2002

Multi-Period Job Selection: Planning Work Loads to Maximize Profit

Herbert F. Lewis
Susan A. Slotnick, *Cleveland State University*

PERGAMON

Computers & Operations Research 29 (2002) 1081–1098

computers & operations research

www.elsevier.com/locate/dsw

Multi-period job selection: planning work loads to maximize profit

Herbert F. Lewis[a], Susan A. Slotnick[b,*]

[a]*W. Averell Harriman School for Management and Policy, State University of New York at Stony Brook, Stony Brook, NY 11794-3775, USA*

[b]*School of Management, Arizona State University West, 4701 West Thunderbird Road, P.O. Box 3100, MC 2451, Phoenix, AZ 85069-37100, USA*

Received 1 May 2000; received in revised form 1 October 2000

Abstract

We examine the profitability of job selection decisions over a number of periods when current orders exceed capacity with the objective of maximizing profit (per-job revenue net of processing costs, minus weighted lateness costs), and when rejecting a job will result in no future jobs from that customer. First we present an optimal dynamic programming algorithm, taking advantage of the structure of the problem to reduce the computational burden. Next we develop a number of myopic heuristics and run computational tests using the DP as benchmark for small problems and the best heuristic as benchmark for larger problems. We find one heuristic that produces near-optimal results for small problems, is tractable for larger problems, and requires the same information as the dynamic program (current and future orders), and another that produces good results using historical information. Our results have implications for when it is more or less worthwhile to expend resources to maintain past records and obtain future information about orders.

Scope and purpose

The purpose of this paper is to investigate trade-offs between accepting or rejecting job orders, completing processing on time and guaranteeing timeliness with money-back guarantees, when job selection decisions will affect future orders. These issues are important to firms that must balance short- and long-term profitability and maintain their customer base in competitive markets. We present a multi-period model that can be solved optimally with a dynamic program, and develop several heuristics which we evaluate with computational studies. We find that our heuristics that use either historical information or future estimates of sales produce good results for relatively large problems without the computational limitation of the optimal procedure. Our studies provide insights into when it is worthwhile for a firm to keep (and process) historical

* Corresponding author. Tel.: + 1-602-543-6121; fax: + 1-602-543-6221.
E-mail address: slotnick@asu.edu (S.A. Slotnick).

0305-0548/02/$ - see front matter
PII: S0305-0548(00)00105-2

sales information, and seek accurate estimates of future orders.

Keywords: Job selection; Scheduling; Heuristics; Dynamic programming

1. Introduction

In increasingly competitive global markets, manufacturing firms and service providers must balance short-term with longer-range profitability while maintaining market share. In order to manage demand as well as capacity, firms must sometimes reject jobs in order to maximize profits. For example, banks have found that while a favored customer may generate over $1000 in annual profit, another customer may actually cost the bank as much as $500 per year. As a result, some banks may actively discourage less profitable customers [1]. Manufacturers also face the issue of whether to reject some orders, and if so, which to reject. Not considering the effect of accepting orders on capacity may result in some jobs costing the firm money [2], as well as affecting the service level for all customers. When on-time delivery of goods and services is an important aspect of the firm's competitive strategy, it may be necessary to reject some jobs in order to avoid processing delays for the most profitable orders. On the other hand, firms must take into account the possible long-term effects of rejecting customers, in terms of reputation and future business [3,4].

We examine the dynamics of the tradeoffs between accepting or rejecting job orders, completing processing on time, and guaranteeing timeliness with money-back guarantees, when the decision of the firm to accept or reject a customer will affect future orders. We model a manufacturing or service facility where customers submit jobs with known processing times, due-dates, and revenues. A customer weight (which reflects the importance of meeting the requested due-date by multiplying the proportional lateness penalty) is also associated with each job. The objective is to maximize profit, which is the sum of the revenue net of processing costs brought in by each job accepted, minus any lateness penalties, and with a premium for early delivery.

Like the earlier studies on which this paper builds [5,6], we use the criterion of lateness (rather than tardiness, which would have no benefit for early delivery) because we rely on optimal sequencing for our procedures. For lateness, we can simply use WSPT order, while for tardiness the sequencing problem is NP-hard [7]. Given an optimal procedure to minimize weighted tardiness, our dynamic program would be unchanged, and the heuristics could be easily adapted.

The firm may reject jobs that it does not consider to be profitable, in terms of the present revenue of the job *and* possible future orders. We model a highly competitive market: if a job is rejected, that customer will never return. Since the one-period problem has been shown to be NP-hard [6], we present an optimal solution method for the multi-period problem for a small number of jobs, and computational studies using heuristic methods for larger problems.

Our optimal procedure is a dynamic program that exploits the structure of the problem to reduce the computational burden. We also develop several heuristics that use varying amounts of information about current, past and future sales, and test them using the dynamic program as

a benchmark. We find that a heuristic that uses the same information as the optimal procedure (future information) produces near-optimal results for small problems, and runs in a fraction of the time needed for the dynamic program. In a second computational study with larger problems, we find that this heuristic still dominates the others. A second heuristic, using historical information, is significantly better than the procedure that only looks at current data. Both of these heuristics are particularly dominant when customer characteristics are relatively more heterogeneous, suggesting that under these conditions, it may be worthwhile for the firm to retain historical data, and to improve its forecasting accuracy along with seeking advance orders from customers.

The next section discusses related work on job selection. Section 3 introduces the model, and Section 4 presents solution methods, including the optimal dynamic programming formulation and myopic heuristics. Section 5 presents the results of our computational studies. Section 6 includes our conclusions and implications for future research.

2. Related work

The issue of job selection has been a topic of growing interest in the last dozen years. Approaches to this problem include mathematical programming [5,6,8–10], simulated annealing and tabu search [11], simulation [12], and queueing theory [13–17]. About half of these studies use minimization of time-related penalties such as tardiness as the objective. Duenyas and Hopp [15] and Duenyas [16] develop queueing models that allow customers to leave if the due-date offered by the firm is too long. The objective is to maximize profit, and the decisions are sequencing and setting of due-dates. De et al. [10] look at the problem with stochastic processing times and due-dates.

Job selection has also been considered in terms of capacity allocation. Balakrishnan et al. [18] model short-life-cycle orders during a single fixed planning period, and develop policies for capacity rationing that discriminate between two classes of products, and test the sensitivity of the resulting policy to forecast errors [19]. Fransoo et al. [20] use a two-level hierarchical model which selectively allocates capacity by making lot-sizing and scheduling decisions in a facility that produces multiple products.

The present paper is an extension of the one-period, deterministic model of Slotnick and Morton [5]. To our knowledge, it is the first investigation of this problem in a multiperiod setting.

3. The model

Consider a firm that processes jobs, over a set number of time periods (stages) within a given time horizon. The firm has m customers at the beginning of the first period; each customer submits one job at each stage, until one of her jobs is rejected. Each job has a predetermined revenue, and the firm pays back a discount to customers whose jobs are completed past a predetermined due-date; customers are willing to pay a premium for early delivery. Each job has a known processing time and importance weight (multiplied times lateness to determine penalty amount). This weight allows the firm to indicate that certain jobs may have an importance beyond their immediate profit. For example, a potential customer may submit a test order which is not particularly lucrative itself;

however, finishing it on time will result in additional business in the future. Such a job will have a relatively large importance weight, to ensure a high priority in the shop.

The firm has the option of rejecting any job. If a job is rejected, the customer is lost (i.e., never sends another job to be processed within the time horizon). The firm must decide which set of jobs to accept in each time period, in order to maximize profit over the time horizon. At the beginning of each period, the firm has available the set of jobs that have been submitted for processing; it also has information about the characteristics (revenue, due-date, processing time and weight) of past and future jobs. We assume that all jobs finish processing in the same period in which they are submitted. Let $\mathbf{x}_k$ be the subset of jobs to be processed in period k; m-dimensional vector (where m is the number of customers) of 1's (customers whose jobs are accepted in period k) and 0's (customers whose jobs are rejected in k or in a previous period). This is the *state* variable. $x_{k,i}$ is the ith element of $\mathbf{x}_k$. Let $\mathbf{u}_k$ be subset of jobs to reject in period k; m-dimensional vector (where m is the number of customers) of 0's (customers whose jobs are accepted or not submitted in period k) and 1's (customers whose jobs will be rejected in period k). This is the *decision* variable. $u_{k,i}$ is the ith element of $\mathbf{u}_k$.

Constraints on $\mathbf{x}$ *and* $\mathbf{u}$:

$$x_{k+1,i} \leqslant x_{k,i} \quad \forall i,k, \tag{1}$$

$$u_{k+1,i} \leqslant x_{k,i} \quad \forall i,k, \tag{2}$$

$$x_{0,i} = 1 \quad \forall i, \tag{3}$$

$$\sum_{k=1}^{N} u_{k,i} \leqslant 1 \quad \forall i. \tag{4}$$

Constraint (1) indicates that once job i has been rejected, the customer is lost forever. Constraint (2) means that we cannot reject the job of a customer that has been rejected previously. Constraint (3) indicates that all customers submit jobs initially, and constraint (4) says that a customer's job can only be rejected once.

Discrete-time equation:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{u}_{k+1}. \tag{5}$$

This indicates that the next state is the current state minus any jobs rejected in that next period.

Profit for period k:

$$\Pi_k = \sum_{i=1}^{m} x_{k,i}[r_{k,i} - w_{k,i}(C_{k,i} - d_{k,i})], \tag{6}$$

where $r_{k,i}$ is the assigned revenue of job i in period k, $w_{k,i}$ is the weight (proportional lateness discount) of job i in period k, $C_{k,i}$ is the completion time of job i in period k; i.e. $C_{k,i} = \sum_{j=1}^{i} t_{k,j}$, where $t_{k,j}$ is the processing time of job j in period k, with jobs arranged in WSPT order to minimize lateness costs, and $d_{k,i}$ is the due-date for job i in period k.

Profit for N periods:

$$\Pi = \sum_{k=1}^{N} \sum_{i=1}^{m} x_{k,i}[r_{k,i} - w_{k,i}(C_{k,i} - d_{k,i})]. \tag{7}$$

We are looking for an optimal policy for choosing $\mathbf{u}_{k+1}$ for each $\mathbf{x}_k$, that maximizes Π.

4. Solution methods

4.1. Dynamic program

The size of the state space and the action space of the dynamic program is large. For m customers (each of whom submits one job per period) there are 2^m states in each stage, and 3^m actions between consecutive stages, where each state represents a possible subset of jobs to be processed in that period. We can count the total number of actions between stages by grouping the states that have the same number of jobs. In a given stage, there are $\binom{m}{0}$ empty states which can only lead to $1 = 2^0$ state (the empty state) in the next stage; $\binom{m}{1}$ states containing one job which can lead to $2 = 2^1$ states (the state containing that job and the empty state) in the next stage; $\binom{m}{2}$ states containing 2 jobs which can each lead to $4 = 2^2$ states (the state containing both jobs, the state containing one of the jobs, the state containing the other job, and the empty state) in the next stage, and so on. In general, there are $\binom{m}{i}$ states containing i jobs which can each lead to 2^i states in the next stage. Thus, the total number of actions is $\binom{m}{0}2^0 + \binom{m}{1}2^1 + \binom{m}{2}2^2 + \cdots + \binom{m}{m}2^m = (1+2)^m = 3^m$ by the Binomial Theorem [21]. As a result, a 20-job problem has $O(2^{20})$ states per stage (1.05 million) and $O(3^{20})$ actions between stages (3.5 billion).

In order to solve problems of modest size, we will show how we can exploit the structure of the problem to reduce the number of actions which must be searched in each iteration by several orders of magnitude, from 3^m to $2^m + m2^{m-1}$ (for a 20-job problem, from almost 3.5 billion to just over 11.5 million). Intuitively, we keep track of states that have already been visited in a given stage, and so do not have to process them more than once.

We first summarize the algorithm and show formally why our method reduces computation. We then demonstrate how this procedure works on a small sample problem.

We first construct a directed acyclic graph containing 2^m vertices and $m2^{m-1}$ arcs. Each vertex represents a state in a stage of the dynamic program. An arc is drawn from vertex x to vertex y if and only if both of the following properties hold:

(1) All jobs included in the state represented by vertex x are included in the state represented by vertex y.
(2) The state represented by vertex y includes exactly one job *not* included in the state represented by vertex x.

This graph will be used to keep track of which states can lead to other states from stage to stage.

Theorem 1. *There are 2^m vertices and $m2^{m-1}$ arcs in the directed acyclic graph.*

Proof. Since each vertex represents a state (subset of jobs), in an m-customer problem, there are 2^m states (vertices in the directed acyclic graph). The vertices can be grouped according to the

number of jobs in the subset. There are $\binom{m}{0}$ empty states, $\binom{m}{1}$ states containing one job, etc. In general there are $\binom{m}{i}$ states containing i jobs. Arcs go from state x to state y where y contains all jobs in x plus exactly one job not in x. Thus, a given state with i jobs has $(m-i)$ arcs leaving it. There are $\binom{m}{i}(m-i)$ arcs leaving all vertices containing i jobs. The total number of arcs is

$$\sum_{i=0}^{m} \binom{m}{i}(m-i) = m\sum_{i=0}^{m} \binom{m}{i} - \sum_{i=0}^{m} i\binom{m}{i}. \tag{8}$$

It can be shown [21] that

$$\sum_{i=0}^{m} \binom{m}{i} = 2^m$$

and

$$\sum_{i=0}^{m} i\binom{m}{i} = m2^{m-1}.$$

Thus

$$\sum_{i=0}^{m} \binom{m}{i}(m-i) = m2^m - m2^{m-1} = m2^{m-1} \qquad \square \tag{9}$$

Procedure

(1) For the last stage, N, calculate the single-stage profit Π_N for each state (jobs arranged in WSPT order). As the profit for each state is determined, insert the state into a binary tree so that a depth-first search traversal of the tree results in a list of states sorted by decreasing profit.

(2) For each remaining stage k (from $N-1$ to 1), perform a depth-first traversal of the binary tree constructed during stage $k+1$. Consider s, the state currently being visited in the traversal of the binary tree constructed during stage $k+1$. Exactly one of the following conditions will hold:

   (a) State s in stage k has already been processed. In this case, visit the next state in the traversal of the binary tree constructed during stage $k+1$.

   (b) State s in stage k has not been processed. In this case, process state s in stage k as follows:

      (i) Let $\hat{s}_k$ denote the state in stage k that is currently being processed, and s_{k+1} denote the state in stage $k+1$ that is currently being visited in the binary tree.

      (ii) Calculate the single-stage profit, $\Pi(\hat{s}_k)$, associated with state $\hat{s}_k$. That is, find the profit in this stage of this subset of jobs.

      (iii) Add $\Pi^*(s_{k+1})$, the total profit associated with state s_{k+1} (that is, the profit of the state with the highest profit to which the current state can go in the next period) to $\Pi(\hat{s}_k)$ to determine the total profit $\Pi^*(\hat{s}_k)$ for state $\hat{s}_k$ (over all stages processed thus far, i.e., k to N). If we proceed from here to the end of the planning horizon with this decision (subset), this is what the total profit would be. The recursion formula for this decision is

$$\Pi^*(\hat{s}_k) = (\Pi(\hat{s}_k) + \Pi^*(s_{k+1})).$$

      (iv) Insert state $\hat{s}_k$ into the binary tree for stage k so that a depth-first search traversal of the tree will result in a list of states sorted by decreasing total profit.

Table 1
Sample problem: single-stage profits

State	Stage		
	1	2	3
∅	0	0	0
A	3	2	2
B	1	3.5	1
C	2	3	5.5
A, B	4	4.5	4
A, C	3	4	5
B, C	5.5	2	3.5
A, B, C	2	6	4.5

(v) Choose state s_{k+1} as the state in stage $k+1$ to which the current state $\hat{s}_k$ will go, and mark state $\hat{s}_k$ as processed.
(vi) Search the descendant nodes of $\hat{s}_k$, using the directed acyclic graph to keep track of what has been processed so far. Repeat steps (i)–(v) above for descendants which have yet to be processed.

(3) To find the optimal path, find the state with the highest profit in stage 1 (the rightmost node in the tree), and take the path with the highest profit to the final stage. The stage 1 profit is the optimal total profit.

Theorem 2. *If the problem is processed in this manner, the optimal solution will be found with the number of searches during each period reduced from* 3^m *to* $2^m + m2^{m-1}$.

Proof. During each period, each state is searched once directly from the depth-first traversal of the binary tree constructed for the next period, as well as once when each of its immediate predecessors in the directed acyclic graph is processed. The size of the first search corresponds to the number of vertices in the directed acyclic graph, and the size of the second is the number of edges in the directed acyclic graph. From Theorem 1, there are 2^m vertices and $m2^{m-1}$ edges in the directed acyclic graph. Thus the total number of searches per period is $2^m + m2^{m-1}$. □

Example. Consider a three-period problem, with three customers, each of whom submits one job each period. The single-period profits for each state in each stage are given in Table 1. For the three jobs A, B and C in each period, we construct the directed acyclic graph shown in Fig. 1. Note that there are 2^m or 8 nodes and $m2^{m-1}$ or 12 edges. Each node comprises a state in the dynamic programming problem, and represents a choice of subset of jobs to be processed.

Now construct a binary tree for period 3, the final stage. First let the root node be the empty set (all jobs rejected). Then calculate the single-stage profit for the first state (say {A}). If the profit is greater than or equal to the profit for the root node, insert the node for this state on a right branch from the root; otherwise insert it on a left branch. Continue to calculate profits and insert nodes

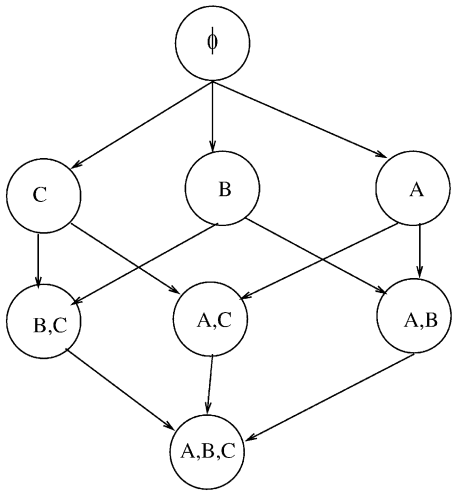

Fig. 1. Directed acyclic graph for sample problem.

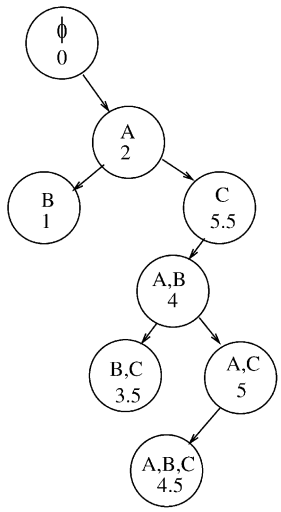

Fig. 2. Binary tree for period 3.

until you have covered all of the eight states. The completed binary tree is shown in Fig. 2 (the profits are indicated below the jobs). Note that a depth-first search of this tree would result in a list of states sorted by decreasing total profit. That is, if you keep branching right, you will reach the highest total profit (here, it is {C} with a profit of 5.5).

Next construct a binary tree for period (stage) 2. Set as root node the node that has the largest profit in period 3. Here it is {C}, which had a profit of 5.5 in period 3. In period 2, {C} has a profit of 3; adding the two profit values yields 8.5. Consider the children of this node (looking at the directed

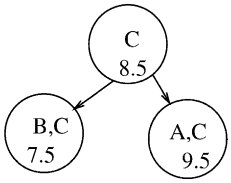

Fig. 3. Partial binary tree for period 2.

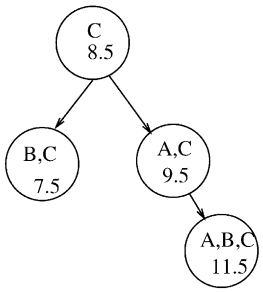

Fig. 4. Partial binary tree for period 2: Children of $\{A, C\}$ and $\{B, C\}$.

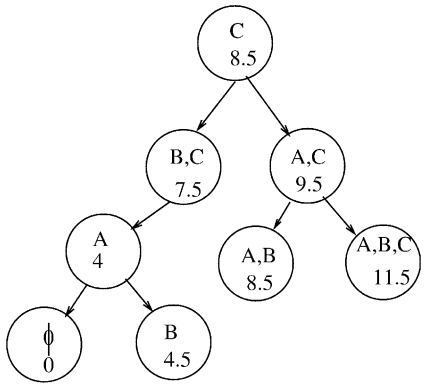

Fig. 5. Binary tree for period 2.

acyclic graph in Fig. 1, these are the nodes $\{A, C\}$ and $\{B, C\}$). Calculate the single-stage profit for each of these nodes, add it to the profit of $\{C\}$, the best state that each of these states can go to period 3, and insert into the binary tree for period 2. The partial binary tree for period 2 is shown in Fig. 3. Next do the same for the child(ren) of $\{A,C\}$ and $\{B, C\}$, namely $\{A, B, C\}$; see Fig. 4.

Now look at the unprocessed node from period 3 that has the next largest profit; this is $\{A, B\}$ (since $\{A, C\}$ and $\{A, B, C\}$ have already been processed). Continue processing nodes in this manner until all of the states in period 2 have been accounted for. The finished binary tree for period 2 is shown in Fig. 5.

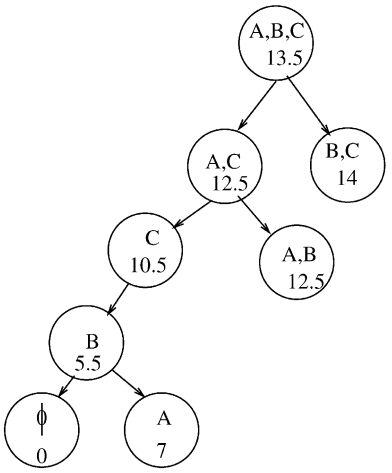

Fig. 6. Binary tree for period 1.

Table 2
Sample problem: results

State	Period					
	1		2		3	
	Best profit	Next state	Best profit	Next state	Best profit	Next state
∅	0	∅	0	∅	0	—
A	7	A	4	A	2	—
B	5.5	B	4.5	B	1	—
C	10.5	C	**8.5**	**C**	**5.5**	—
A, B	12.5	A, B	8.5	A, B	4	—
A, C	12.5	A, C	9.5	C	5	—
B, C	**14**	**C**	7.5	C	3.5	—
A, B, C	13.5	A, B, C	11.5	C	4.5	—

Now connect each node in period 2 with the node in the next period that has the highest total profit; this will be the optimal path. For example, for node $\{C\}$ the optimal path from period 2 to period 3 is $\{C\} \rightarrow \{C\}$; for $\{A, B, C\}$ it is $\{A, B, C\} \rightarrow \{C\}$. We only need to keep track of these paths on the backward iterations of the dynamic program. Keep processing backward through the first period, and then trace the optimal path forward starting at the state in period 1 with the highest total profit. The binary tree for period 1 is shown in Fig. 6.

Now we can determine the optimal path by working from the state with the highest profit in period 1 (i.e. $\{B, C\}$) through period 2 (i.e. $\{C\}$) to period 3 ($\{C\}$). Thus the optimal path is $\{B, C\} \rightarrow \{C\} \rightarrow \{C\}$, with a profit of 14. See Table 2, where "best profit" is the optimal profit from

this stage to the last one, and "next state" is the best state in the next stage; the optimal path is indicated in bold-face type. This means that in the first period, jobs B and C should be accepted for processing (and A rejected); in the second and third periods, only process C (reject B in period 2).

4.2. Heuristic methods

Even with our reduction in search space, the combinatorics of the problem preclude the solution of problems of any considerable size with our dynamic programming method. In order to solve larger problems (and solve smaller problems much faster), we develop a number of heuristics.

These heuristics use previous results [5] and extend them to the multi-period case. The myopic index indicates which jobs are definitely members of the optimal set in the one-period problem, and which are possibly not part of the optimal solution, by considering the combination of each job's contribution to revenue and to the lateness of the jobs sequenced after it. We first determine whether a job is worth keeping in a single-period case (step 1). We only consider rejecting those jobs that we are not certain are members of the optimal set (see Slotnick and Morton [5]), using information that we have about this customer in the past and the future.

4.2.1. Current myopic index

This first heuristic (MCH) simply rejects jobs, one by one, that will improve profitability in the current period if they are rejected. Once the available set has only jobs whose rejection will decrease profit, the heuristic terminates. In each stage the number of operations is $O(m^2)$ where m is the number of customers, significantly smaller than the dynamic program.

(1) For each period, put jobs into WSPT order (to minimize weighted lateness) and calculate the myopic index $\psi_{k,i}$ for each job i:

$$\psi_{k,i} = t_{k,i} \sum_{j=i+1}^{m} w_{k,j} + w_{k,i}(C_{k,i} - d_{k,i}) - r_{k,i}.$$

(2) For the subset of jobs for which $\psi_{k,i} > 0$, reject the job for which $\psi_{k,i}$ is largest.
(3) If a job is rejected, recalculate the $\psi_{k,i}$'s for the remaining jobs, and return to step 2.
(4) When $\psi_{k,i} \leqslant 0$ for all remaining jobs, stop. This is the subset of jobs to accept this period.

4.2.2. Historic myopic average

For this heuristic (MHH), we first determine whether a job is worth keeping in a single-period case (step 1). If we may want to discard it, we look at how jobs from this customer have contributed to profit and lateness in previous periods (steps 2 and 3), so that we are considering a profile of the customer rather than just the current job. In each stage the number of operations is $O(m^3)$ where m is the number of customers; so this heuristic runs significantly more quickly than the dynamic programming algorithm.

(1) For each period, put jobs into WSPT order (to minimize weighted lateness) and calculate the myopic index $\psi_{k,i}$ as above.
(2) For those jobs for which $\psi_{k,i} > 0$, consider the historic myopic average (average of this index for all jobs from this customer, including this one, through this period).

(3) If the myopic average is nonnegative, then delete this job starting in the current period. Results from Slotnick and Morton [5] show that if a job's myopic index calculated for any set is positive (i.e., that job may not be profitable in this period), it will also be positive for any subset, and so by using the full set of jobs to calculate the index for each period we are being conservative about rejecting jobs.
(4) If a job is deleted, recalculate the $\psi_{k,i}$'s for the remaining jobs, and return to step 2.
(5) When $\psi_{k,i} \leqslant 0$ for all remaining jobs, stop. This is the subset of jobs to accept this period.

4.2.3. Future myopic average

This method (MFH) uses the myopic index in a forward-looking manner. If we have some knowledge of characteristics (weight, processing time, due-date and revenue) of future jobs, we can base our acceptance of current jobs on the value of a customer's business in the future. This method assumes that we know characteristics of incoming jobs until the end of the planning horizon. In each stage, the number of operations is $O(nm^3)$ where n is the number of periods in the planning horizon and m is the number of customers.

(1) For each period, put jobs into WSPT order (to minimize weighted lateness) and calculate the myopic index $\psi_{k,i}$ as above.
(2) For those jobs for which $\psi_{k,i} > 0$, consider the future myopic average (average of this index for all jobs from this customer, including this one, from this period onward). In order to calculate this average, assume that all jobs are accepted in each future period.
(3) If the myopic average is nonnegative, then reject this job in the current period.
(4) If a job is rejected, recalculate the $\psi_{k,i}$'s for the remaining jobs, and return to step 2.
(5) When $\psi_{k,i} \leqslant 0$ for all remaining jobs, stop. This is the subset of jobs to accept this period.

5. Computational studies

In order to compare our quick heuristics with optimal or near-optimal benchmarks, we ran a number of computational tests. We divided the jobs to be processed into three different types, to represent different types of customers or orders. Dividing customers into various categories is a standard practice for service providers [22]; similar classification systems are used to schedule workforces [23] and prioritize inventory management [24]. We varied job parameters in order to test the heuristic(s) in different types of market scenarios. Homogeneous customers (tests 1–50) serve as the baseline scenario (B). We then differentiated between three customer types, with regard to processing times, revenue and weight. Processing times were i.i.d. according to an exponential distribution. Varying the mean (tests 51–100) allowed us to represent a situation in which customers are differentiated by the average length of their jobs (P). For revenue and weight, we used a lognormal distribution and varied both the mean (for which the numbers in the table indicate the mean of the underlying normal distribution) and standard deviation (S.D.) (tests 101–300), in order to consider more or less heterogeneous job submissions, in which customers are differentiated by the amount of revenue brought in by their typical jobs, as well as the importance weights assigned (R1, R2, W1 and W2). Table 3 shows the various scenarios that were generated.

Table 3
Computational test: various scenarios

Scenario	Test number	Proc. time Mean			Revenue Mean (S.D.)			Weight Mean (S.D.)		
	Customer type	1	2	3	1	2	3	1	2	3
B	1–50	5.0	5.0	5.0	1.0 (1)	1.0 (1)	1.0 (1)	1.0 (1)	1.0 (1)	1.0 (1)
P	51–100	**2.5**	**5.0**	**7.5**	1.0 (1)	1.0 (1)	1.0 (1)	1.0 (1)	1.0 (1)	1.0 (1)
R1	101–150	5.0	5.0	5.0	**0.0 (1)**	**1.0 (1)**	**2.0 (1)**	1.0 (1)	1.0 (1)	1.0 (1)
R2	151–200	5.0	5.0	5.0	**0.0 (2)**	**1.0 (2)**	**2.0 (2)**	1.0 (1)	1.0 (1)	1.0 (1)
W1	201–250	5.0	5.0	5.0	1.0 (1)	1.0 (1)	1.0 (1)	**0.0 (1)**	**1.0 (1)**	**2.0 (1)**
W2	251–300	5.0	5.0	5.0	1.0 (1)	1.0 (1)	1.0 (1)	**0.0 (2)**	**1.0 (2)**	**2.0 (2)**

Table 4
Summary of the results from the pilot study

Scenario	DP	MFH		MHH		MCH		ALL
	Average profit	Average profit	Average optimal (%)	Average profit	Average optimal (%)	Average profit	Average optimal (%)	Average Profit
B	\$237	\$212	87.9	\$138	61.4	\$130	60.0	− \$374
P	\$276	\$253	89.6	\$180	66.6	\$153	59.8	− \$322
R1	\$298	\$257	83.6	\$168	55.9	\$160	54.7	− \$278
R2	\$898	\$858	92.4	\$471	58.5	\$434	52.7	\$409
W1	\$393	\$346	81.8	\$235	59.2	\$227	58.3	− \$457
W2	\$1865	\$1786	86.6	\$1074	56.3	\$1058	55.7	− \$1115

Bold-face indicates the attributes that are different for a given test. The procedures were coded in C, using the Microsoft C compiler, and run on a Pentium-based PC using a DOS/Windows platform.

5.1. Pilot study

The purpose of the pilot study was to test the myopic heuristics (MCH, MHH, MFH) against the optimal benchmark provided by the dynamic program (DP). We also included a procedure that simply accepted all jobs (ALL). For the six scenarios described above, we ran 50 sets of 6-job problems for 5 periods. This was the largest problem that we could solve using the dynamic program; the state space of larger problems exceeded available memory.

In Table 4, we summarize the results of the pilot study. The average profits produced by the dynamic program, by each myopic heuristic, and by ALL are listed for each scenario. Note that all the average profits produced by ALL are negative except for scenario R2, suggesting that job

Table 5
95% confidence intervals for the proportion of times each heuristic found the optimal solution, and the average difference of the heuristic's solution value and that of the optimal

Scenario	Statistic	MFH	MHH	MCH
B	Proportion optimal	0.28 ± 0.12	0.00 ± 0.00	0.02 ± 0.04
	Avg difference from optimal	$\$25 \pm \$10$	$\$99 \pm \$31$	$\$107 \pm \$32$
P	Proportion optimal	0.32 ± 0.13	0.04 ± 0.05	0.06 ± 0.07
	Avg difference from optimal	$\$22 \pm \$9$	$\$96 \pm \$30$	$\$122 \pm \$34$
R1	Proportion optimal	0.08 ± 0.08	0.00 ± 0.00	0.00 ± 0.00
	Avg difference from optimal	$\$41 \pm \$13$	$\$130 \pm \$30$	$\$137 \pm \$32$
R2	Proportion optimal	0.28 ± 0.12	0.00 ± 0.00	0.00 ± 0.00
	Avg difference from optimal	$\$40 \pm \$20$	$\$427 \pm \$183$	$\$464 \pm \$183$
W1	Proportion optimal	0.12 ± 0.09	0.02 ± 0.04	0.02 ± 0.04
	Avg difference from optimal	$\$47 \pm \$22$	$\$158 \pm \$60$	$\$166 \pm \$62$
W2	Proportion optimal	0.30 ± 0.13	0.02 ± 0.04	0.04 ± 0.05
	Avg difference from optimal	$\$79 \pm \$69$	$\$791 \pm \$513$	$\$807 \pm \$517$

Table 6
Results of Marascuilo's procedure for comparison among heuristics

	MFH − MHH	MFH − MCH	MCH − MHH
Proportion difference	0.2167	0.2067	0.0100
Critical range	0.0616	0.0632	0.0268

rejection is usually advisable for the scenarios that we generated. We have also calculated the average percentage of optimal profit produced by each heuristic for each scenario. Table 5 shows the sample proportion of instances in which each heuristic found the optimal solution, with a 95% confidence interval for each scenario, and the average difference between the optimal profit and the profit produced by each heuristic with a 95% confidence interval for each scenario.

Out of 300 problem instances, the MFH found the optimal solution 69 times (23%), the MHH found the optimal solution 4 times (1.3%), and the MCH found the optimal solution 7 times (2.3%). We performed a χ^2 test to determine whether the heuristics are different with respect to the proportion of problem instances in which the optimal solution was found. The results of this test indicate that there is significant difference among the heuristics (overall χ^2 was 110.83 with 2 degrees of freedom and a p-value < 0.00005). Specifically, using Marascuilo's procedure [25,26] at $\alpha = 0.05$, the proportion of problem instances in which the MFH produced the optimal solution was significantly higher than that using either the MHH or the MCH. However, there was no significant difference between the performance of the MHH and that of the MCH. Table 6 shows the results from Marascuilo's procedure.

We next performed a χ^2 test to determine whether the proportion of problem instances in which the optimal solution was found differed among scenarios for each of the three heuristics. The results

Table 7
Summary of the results of the analysis of variance for each scenario of the pilot study

Scenario	F-value	df	p-value
B	21.75	(2, 98)	< 0.00005
P	28.61	(2, 98)	< 0.00005
R1	33.57	(2, 98)	< 0.00005
R2	19.59	(2, 98)	< 0.00005
W1	14.05	(2, 98)	< 0.00005
W2	7.71	(2, 98)	< 0.00009

of this test indicate that there is a significant difference among the scenarios using the MFH (overall $\chi^2 = 14.85$ with 5 degrees of freedom and a p-value $= 0.0110$). However, there is no significant difference among the scenarios using the MHH (overall $\chi^2 = 5.07$ with 5 degrees of freedom and a p-value $= 0.4077$) or the MCH (overall $\chi^2 = 6.00$ with 5 degrees of freedom and a p-value $= 0.3065$). We used Marascuilo's procedure to compare the performance of the MFH between pairs of scenarios. The results showed only weak significance between scenario P and R1 at $\alpha = 0.079$ and between scenarios R1 and W2 at $\alpha = 0.130$.

To compare the performance of the three myopic heuristics, we performed a randomized block design two-way analysis of variance without replication for each scenario. We used profit as the dependent variable and heuristic as the treatment. Problem instance was the block. See Table 7 for a summary of the results.

For all scenarios, the average profits differed among the heuristics. Specifically, using Tukey's HSD at $\alpha = 0.05$, the MFH produced significantly higher average profits than did either the MHH or the MCH for all scenarios. This indicates that having information on future jobs, whether by taking advanced orders or developing an accurate forecasting methodology, can significantly increase profit. However, there was no significant difference between the performance of the MHH and the MCH in all scenarios.

5.2. Large-scale study

In order to investigate the performance of our heuristics on larger problems than we could solve with the dynamic program, we ran 50 sets of 30-job problems for 30 periods for the three myopic heuristics and ALL.

In Table 8, we summarize the results of the large-scale study. The average profits produced by each myopic heuristic and by ALL are listed for each scenario. Note that all of the average profits produced by ALL are negative, indicating that we have generated scenarios in which it is profitable to reject jobs. We have also provided the average of the best-known profits (i.e., the best answer given by any of the heuristics) for each scenario and calculated the average percentage of the best-known profit produced by each myopic heuristic for each scenario. We note that the MFH did not always get the highest profit (the MHH did better 15 times or 30% for scenario B, 8 times or 16% for scenario P, 10 times or 20% for scenario R1, 4 times or 8% for scenario R2, 7 times or 14% for scenario W1, and 1 time or 2% for scenario W2).

Table 8
Summary of the results from the large-scale study

Scenario	Best known	MFH		MHH		MCH		ALL
	Average profit	Average profit	Average best (%)	Average profit	Average best (%)	Average profit	Average best (%)	Average profit
B	\$787	\$753	95.5%	\$611	81.2%	\$390	55.2%	− \$91,098
P	\$1325	\$1301	97.7%	\$999	78.9%	\$544	46.0%	− \$84,015
R1	\$1200	\$1178	97.8%	\$980	82.6%	\$538	49.3%	− \$88,206
R2	\$9607	\$9595	99.7%	\$4783	56.1%	\$1946	22.9%	− \$65,022
W1	\$1504	\$1481	97.9%	\$989	71.0%	\$704	53.8%	− \$97,265
W2	\$24,302	\$24,293	99.4%	\$6525	39.6%	\$3890	25.6%	− \$158,101

Table 9
Summary of the results of the analysis of variance for each scenario of the large-scale study

Scenario	F-value	df	p-value
B	38.05	(2, 98)	< 0.00005
P	79.92	(2, 98)	< 0.00005
R1	84.63	(2, 98)	< 0.00005
R2	91.07	(2, 98)	< 0.00005
W1	40.07	(2, 98)	< 0.00005
W2	39.01	(2, 98)	< 0.00005

To compare the performance of the three myopic heuristics, we performed a randomized block design two-way analysis of variance without replication for each scenario. We used profit as the dependent variable and heuristic as the treatment. Problem instance was the block. See Table 9 for a summary of the results.

As in the pilot study, the average profits differed among the heuristics for all scenarios. Using Tukey's HSD at $\alpha = 0.05$, the MFH produced significantly higher average profits than either the MHH or the MCH for all scenarios. This reinforces the results of the pilot study, indicating the importance of accurate forecasting or advance orders. In contrast to the pilot study, the MHH performed significantly better than the MCH in all but the W2 scenario, indicating that the historic information may be valuable over the long haul and thus the cost of storing and processing this information may be worthwhile.

6. Conclusions and future research

We set out to investigate the profitability of job selection decisions when a firm has too many jobs to complete on time, and when rejecting jobs would result in loss of future business. While we

can generate optimal results with a dynamic programming algorithm, the combinatorics limit the size of the problems that we can consider. Using the same information that would be provided to the dynamic program (predictions of future sales, which could come from advance orders or demand forecasting), our future myopic average heuristic does not share these computational limitations, and gets results for small problems that are on average always less than 20% worse than optimal. For larger problems, it clearly dominated the other heuristics. Another myopic heuristic that only requires past sales information (MHH) is clearly better than the strictly myopic method (MCH) as problems grow large.

Our results provide insights about the value of information to a manufacturing firm under differing market conditions. When jobs are relatively heterogenous with regard to revenue and customer weight (R2, W2), the heuristics that use more than simply current information (MFH, MHH) significantly outperform the others (MCH, ALL). This implies that when customers are widely differentiated by per-job revenue and weight, it may be worthwhile for a firm to expend resources to maintain and process historical sales information, as well as improving the accuracy of its forecasts, and working with customers to gain as much information as possible about future orders.

Extensions of the present model might include more complicated "rules" for job submission. For example, we might decrease submission frequency from a given customer if previous orders have been delivered late, or if many jobs are delivered late in any given period (reputation effect). We might also allow customers to enter as well as leave.

Acknowledgements

We would like to thank Thomas R. Sexton of the State University of New York at Stony Brook for his helpful comments in general and his assistance with the statistical analysis of the computational studies in particular.

References

[1] Brooks R. Alienating customers isn't always a bad idea, many firms discover. Wall Street Journal 7 January 1999;A1.
[2] Guerrero HH, Kern GM. How to more effectively accept and refuse orders. Production and Inventory Management 1988;29(4):59–63.
[3] Shapiro B, Moriarty RT, Kline CE, Fabtek(A). Boston, MA, 1992. Harvard Business School Case Study No. 9-592-095.
[4] Shapiro B. Fabtek(A) & (B). Boston, MA, 1992. Harvard Business School Teaching Note No. 5-593-006.
[5] Slotnick SA, Morton TE. Selecting jobs for a heavily loaded shop with lateness penalties. Computers & Operations Research 1996;23(2):131–40.
[6] Ghosh JB. Job selection in a heavily loaded shop. Computers & Operations Research 1997;24(2):141–5.
[7] Pinedo M. Scheduling: theory, algorithms, and systems. Englewood Cliffs, NJ: Prentice-Hall, 1995.
[8] Pourbabi B. A short term production planning and scheduling model. Engineering Costs and Production Economics 1989;18:159–67.
[9] Pourbabi B. Optimal selection of orders in a just-in-time manufacturing environment: a loading model for a computer integrated manufacturing system. International Journal of Computer Integrated Manufacturing 1992;5(1):38–44.

[10] De P, Ghosh JB, Wells CE. Job selection and sequencing on a single machine in a random environment. European Journal of Operational Research 1993;70:425–31.
[11] Woodruff DL. Subcontracting when there are setups, deadline and tooling costs. In: Scherer WT, Brown DE, editors. Proceedings of the Intelligent Scheduling Systems Symposium. November 1992. p. 337–53.
[12] Wester FAW, Wijngaard J, Zijm WHM. Order acceptance strategies in a production-to-order environment with setup times and due-dates. International Journal of Production Research 1992;30(6):1313–26.
[13] Wein LM. Due-date setting and priority sequencing in a multiclass M/G/1 queue. Management Science 1991;37(7):834–50.
[14] Deallaert NP. Due-date setting and production control. International Journal of Production Economics 1991;23:59–67.
[15] Duenyas I, Hopp WC. Quoting customer lead times. Management Science 1995;41(1):43–57.
[16] Duenyas I. Single facility due date setting with multiple customer classes. Management Science 1995;41(4):608–19.
[17] Garbe R, Glazebrook KD. Submodular returns and greedy heurisitcs for queueing scheduling problems. Operations Research 1998;46(3):336–46.
[18] Balakrishnan N, Patterson JW, Sridharan SV. Rationing capacity between two product classes. Decision Sciences 1996;27(2):185–214.
[19] Balakrishnan N, Patterson JW, Sridharan SV. Robustness of capacity rationing policies. European Journal of Operational Research 1999;115:328–38.
[20] Fransoo JF, Sridharan V, Bertrand JWM. A hierarchical approach for capacity coordination in multiple products single-machine production systems with stationary stochastic demands. European Journal of Operational Research 1995;86(1):57–72.
[21] Tucker A. Applied combinatorics. New York: Wiley, 1980.
[22] Coleshaw J. Getting the best out of customers. Management Today, May 1986.
[23] Bucholtz C. Pac Bell uses force to cut response time. Telephony 1996;231(2):7.
[24] Russell RS, Taylor BW. Operations management, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 1998.
[25] Marascuilo LA. Large-sample multiple comparisons. Psychological Bulletin 1966;65:280–90.
[26] Marascuilo LA, McSweeney M. Nonparametric and distribution-free methods for the social sciences. Pacific Grove, CA: Brooks/Cole, 1977.

Herbert F. Lewis is a Lecturer at the W. Averell Harriman School of Management and Policy at the State University of New York at Stony Brook. He received his Ph.D. from the Department of Applied Mathematics and Statistics at the State University of New York at Stony Brook. His research interests include solution methods for combinatorial problems in the areas of scheduling, vehicle routing, and facility location, as well as productivity and efficiency analysis using data envelopment analysis.

Susan A. Slotnick is Assistant Professor at the School of Management at Arizona State University West. She received her Ph.D. from the Graduate School of Industrial Administration at Carnegie Mellon University. She also holds a Ph.D. in Linguistics from Columbia University. Her research interests include heuristic and knowledge-based approaches to scheduling and quantitative modeling of managerial decision-making in the areas of quality management and new product development.