DEPARTMENT OF OPERATIONS RESEARCH
UNIVERSITY OF AARHUS

Working Paper no. 2002/3

Production-to-order pricing based on renewal-reward optimization

Søren Glud Johansen

ISSN 1600-8987

Department of Mathematical Science
Telephone: +45 8942 1111
E-mail: institut@imf.au.dk

Building 530, Ny Munkegade
DK-8000 Aarhus C, Denmark
URL: www.imf.au.dk

Production-to-order pricing based on renewal-reward optimization

Søren Glud Johansen*
Department of Operations Research
University of Aarhus, Denmark

February 5, 2002

We analyze two models for the pricing of customized products which we call jobs. The models have discrete time and assume that the demand process is Bernoulli. The random processing time of each job is predictable if and when the job is submitted to the considered system. Its objective is to maximize the gain (the long-run average profit) earned by processing the submitted jobs. The first model restricts the profit margin of each job to be specified in terms of some basic decision variables (e.g. the mark-ups for different job types), whereas the second model has unrestricted prices. We suggest solving the two models by a modified Newton-Raphson algorithm and a tailor-made policy-iteration algorithm, respectively. Both algorithms are designed for renewal-reward optimization and they can easily be adapted to other models than the two analyzed in this paper. The first algorithm relies on our easy-to-compute expressions for the gradient and the Hessian of the gain function. The second algorithm relies on some shadow fees introduced by us. A numerical example illustrates that a few carefully chosen decision variables suffice to obtain a gain close to the maximal one.

Keywords: Make-to-order; Congestion effect; Pricing; Renewal-reward optimization; Restricted policy; Markov decision process.

*Tel.: +45 8942 3547; fax: +45 8613 1769; Email: sgj@imf.au.dk

1 Introduction

Make-to-order firms face the problem that each accepted order has a congestion effect on the delivery lead times for future orders. When the customers are sensitive to the lead times quoted, the congestion effect makes the pricing of the orders difficult. This effect is therefore often neglected. We shall, in this paper, use a numerical example to illustrate that ignorance of the congestion effect can have a severe impact on the profit earned by make-to-order production. The example also illustrates that an expected profit close to the maximal one can be obtained from the solution to a model where the profit margin of each order is specified in terms of two mark-up variables only.

Our focus is on the pricing of customized products. Often such products contain common components, which are produced before the demand occurs. In this paper we neglect leadtime-inventory trade-offs (Glasserman and Wang 1998) and assume that the common components are available at the push-pull point (Silver et al. 1998, page 541) when needed for the make-to-order production of the customized products. The 'production' may comprise assembly only, but it may also include much more, e.g. the activities in an integrated production system. The 'customized product' might be a service or a package of services, cf. Kingsman and de Souza (1997) who have coined the term 'versatile manufacturing'.

Each customized product is a job which has a predictable processing time. Job requests arrive to the considered firm, referred to as the system, from external customers. Each customer is informed of the price to be charged for processing his job and the lead time offered by the system. The customer submits the job to the system when his benefit by doing so is larger than or equal to the price. The gain of the system is the long-run average of the difference between the revenue earned and the direct cost incurred per unit time.

The prices maximizing the gain can be computed as the solution to a Markov decision model. Factors other than economic efficiency, such as cost of analysis and understandability, may cause the decision maker not to implement this solution (Miller and Buckman 1987). We shall therefore also study gain maximization when the prices are restricted to be specified in terms of some basic decision variables. The decision variables may e.g. prescribe the mark-ups for different job types. Our numerical example illustrates that a few carefully chosen decision variables suffice to obtain a gain close to the maximal one.

Our models have discrete time and assume that the system processes the jobs one at a time and in the first-come-first-served order. The base period is the constant time between observation epochs, at which job requests arise from customers according to a stationary Bernoulli process (Çinlar 1975, page 44). The processing times of the jobs are i.i.d. integer numbers of the base periods. For any job with processing time k, the system perceives the customer's benefit as a random variable being stochastically decreasing in the work backlog i (the sum of the remaining processing times for the jobs already submitted), and the lead time is $i + k$.

The price to be charged for a submitted job is the sum of the direct cost incurred by the system if the job is submitted and a profit margin. When the backlog is i, the profit margin $p_{i,k}(\mathbf{x})$ for a submitted job with processing time k is a function of some decision vector $\mathbf{x}$. The components of this vector and the profit margin function can

be specified as desired. If prices are unrestricted, then we let the vector prescribe a profit margin $x_{i,k}$ for each i and k, i.e. $p_{i,k}(\mathbf{x}) = x_{i,k}$. But our main focus is on settings where the number of components of $\mathbf{x}$ is much smaller than the number of i and k combinations. Each setting is modeled as a renewal-reward process. Its state at each observation epoch is the current backlog i and we let $i = 0$ (identifying an observation epoch with no backlog) be the regeneration state.

A cycle for the renewal-reward process is the time interval between two consecutive regeneration epochs. The expected reward $R_0(\mathbf{x})$ earned during a cycle and the expected length $T_0(\mathbf{x})$ of one cycle are functions of the vector $\mathbf{x}$ chosen. The renewal-reward theorem (Tijms 1994, Theorem 1.3.1) provides that the gain is $\gamma(\mathbf{x}) = R_0(\mathbf{x})/T_0(\mathbf{x})$. The objective is to maximize this gain.

We derive easy-to-compute expressions for the gradient $\mathbf{G}(\mathbf{x})$ and the Hessian $\mathbf{H}(\mathbf{x})$ of the gain function at $\mathbf{x}$. These expressions are used in our modified Newton-Raphson algorithm for optimizing $\mathbf{x}$. To the best of our knowledge, such an algorithm has never before been implemented for renewal-reward optimization. The algorithm computes a near-optimal decision vector rapidly when the dimension of the vector is small. Since this is normally not the case if the prices are unrestricted, we then compute the optimal prices as the solution of a Markov decision model. The transition structure of this model is exploited to design a fast tailor-made policy-iteration algorithm. This algorithm allows infinite state and action spaces. A novel feature of the algorithm is that it incorporates shadow fees which ensure that the state values occurring during the iterations become monotone like the final state values.

The paper is organized as follows. Related literature is reviewed in §2. Our expressions for the gradient and the Hessian of the gain function are derived in §3. Here we also present our modified Newton-Raphson algorithm for optimizing a decision vector which specifies restricted prices. A numerical example is presented in §4. Section 5 presents our tailor-made policy-iteration algorithm for optimizing unrestricted prices and illustrates its performance for the numerical example in §4. Section 6 has concluding remarks. Appendices A and B describe procedures to be used by the algorithms presented in §3 and §5, respectively.

2 Related literature

Researchers at Lancaster University have developed a Decision Support System (DSS) that seeks to integrate the marketing and the production functions of make-to-order companies, see Hendry el al. (1998) and their references. The DSS copes with workload control at the job entry level and the job release level. The former level includes quoting lead times and prices (Kingsman and de Souza 1997), but it incorporates many more details than our two models. For example, the job entry level is split into a customer enquiry stage and a customer confirmation stage. This split is important in industries where the customer may take a long time to confirm the quote made at the enquiry stage. Until the customer decides, the system has a contingent order (Easton and Moodie 1999).

Elhafsi (2000) presents a model which incorporates the job entry level only. His model focuses on a single order when it arrives to a system consisting of several work

stations, all subject to random failures and repairs. The aim is to schedule the order among the work stations so as to minimize the total operating cost and/or to assign a due date for the order that does not violate a specified time window. The model is more general than ours, because it deals with a system which has multiple resources. However, the model and the DSS mentioned above neglect the congestion effect. Moreover, the model does not incorporate how the customer reacts to the quoted price (specified as the operating cost plus a profit margin). Elhafsi and Rolland (1999) emphasize that the model can be used as a tool for negotiating the delivery date and the price for a certain upcoming order.

Johansen (1991) studies a discrete-time job-shop consisting of a single work station under various objectives. One of the objectives studied is to maximize the gain defined as the long-run average profit earned per base period. He observes that the complementary distribution of a customer's random benefit may be thought of as a demand function and he suggests an algorithm for computing near-optimal prices in terms of opportunity costs which incorporate the congestion effect. For the profit objective, the opportunity cost of a job is the expected future loss of profit caused when the job is submitted into the shop.

Johansen (1994) studies a continuous-time job-shop system. Its requests for service of jobs are Poisson and the processing time of each job is a continuous random variable. Apart from this the continuous-time system is similar to the discrete-time system studied by Johansen (1991). The latter system can be used to compute approximately the value rate function needed to specify opportunity costs and optimal prices for the former system. Observe that the discrete-time model studied in this paper can also be used to approximate the similar continuous-time model for which the job arrivals are Poisson rather than Bernoulli. Johansen (1996) shows how an $M/D/1$ model for transfer pricing can be approximated and observes for this model that the gain reduction caused by restricting the price to be the same for all jobs is at most 3% in the numerical examples reported.

Duenyas and Hopp (1995) study the problem of quoting lead times in the context of a $GI/G/1$ queue. They assume that all jobs are identical. Each submitted job generates the same net revenue (price minus production cost) and the i.i.d. processing times are unpredictable upon arrival. The proportion of customers that actually submit their job depends on the quoted lead time. The system incurs a penalty if a submitted job is delivered later than promised. The authors prove the optimality of different forms of control limit policies for the case where the jobs are processed in the first-come-first-served (FCFS) order. They also give conditions under which the optimal due-date-quoting/order-scheduling policy will process jobs in earliest due date order. Duenyas (1995) assumes that there are different classes of customers. All customers demand the same product, but they differ with respect to their preferences for price and lead time. For the FCFS case, Duenyas formulates a semi-Markov decision process for optimizing the price and the lead time to be quoted for each customer, depending on the congestion in the system. He also suggests a heuristic for dynamic price and due date setting when the order-scheduling is open for decision making.

Most of the queueing literature on admission control assumes that the processing

times of the arriving jobs are exponential and not predictable upon arrival (Stidham 1985). These assumptions imply that the considered system can be modeled as a semi-Markov decision process observed at the arrival epochs where the state is described by the number of jobs in the system. The action specification depends on the structure of the system. References studying control exercised by charging a price for each submitted job include Low (1974), who considers an $M/M/s$ job-shop with finite capacity and Miller and Buckman (1987), who study transfer pricing in the context of an $M/M/s/s$ queue. The former reference advocates dynamic pricing, whereas the latter restricts the transfer price to be the same for all jobs. Larsen (1998) studies an $M/M/1/\infty$ job-shop controlled by charging the same price for all submitted jobs. He demonstrates how the best price depends on the information system (steady state information or information about the actual number of jobs in the shop) and the objective (profit maximization or welfare maximization). Li and Lee (1994) present a model of the competition on price and delivery speed between two $M/M/1$ queues providing substitutable goods or services in a make-to-order fashion. In many applications, the main reason for varying processing times is varying jobs and it is often easy to make a good estimate of the processing time of a job upon arrival. Therefore we find it unfortunate to assume that the actual processing time of a job is not predictable upon arrival and cannot influence the price charged if the job is submitted. Moreover, we doubt that it is realistic to restrict the processing times to be exponential.

Low (1974) and Johansen (1991) present similar fixed-point algorithms for computing prices which are optimal, if the set of allowable prices is finite, and if not, near-optimal. The algorithms optimize the gain g by solving a sequence of regenerative stopping problems (Miller 1981) with g-revised rewards. The two references initialize and update g differently but both use the fact that each problem in the sequence has a simple transition structure. They exploit implicitly that the transitions are skip-free to the right and to the left, respectively. The former occurs when a one-step transition from state i to state j is not possible if $j > i + 1$ (Weijngaard and Stidham 1986), whereas the latter occurs when such a one-step transition is not possible if $j < i - 1$ (Stidham and Weber 1989). The two skip-free structures imply that the state values can be computed recursively, starting with the smallest state (Low) or the largest state (Johansen). We present in §5 an improved fixed-point algorithm exploiting that the transitions are skip-free to the left. This algorithm resembles the algorithm suggested by Johansen and Larsen (2001) for the computation of a service policy for a single-server queue with homogeneous jobs. But the latter algorithm incorporates a shadow fee (called a test quantity) only when the system is empty, whereas our algorithm can incorporate shadow fees for other states of the system also.

Much recent research on stochastic optimization problems has been based on simulation. Fu (1994) and Fu and Hu (1997) review the early part of this research. It has enabled scholars to tackle problems not tractable by other approaches. The large activity in the area witnesses that the simulation approach is found promising by many researchers, including the author of this paper. But we wish to emphasize that the simulation approach is not the best one for all problems. We think that the pricing

problem studied in this paper is a good example of a problem which can be solved more efficiently by our exact calculations than by the simulation approach. Both approaches focus on the gradient of the objective function. Fox and Glasserman (1991) present a sampling plan for settings where it is difficult to compute the gradient analytically. This is not the case for our discrete-time model. But for the continuous-time model with Poisson requests and continuous processing times, the gradient cannot be computed exactly and the simulation approach may be good because it can handle continuous variables. We shall not in this paper report computations made by this approach. From a computational point of view, we believe that it is more efficient to approximate the continuous-time model with a discrete-time model having Bernoulli requests and processing times which are integer numbers of base periods. The accuracy of the approximation and the computational effort depend on the base period chosen.

3 Optimizing the decision vector

Let K be the generic random variable for the potential input to the considered system at each observation epoch. This input is either 0 or the processing time of one job. The processing times are i.i.d. integer numbers of base periods and take values in a finite set $\mathcal{K}$. Define $q_k = \Pr\{K = k\},\ k \in \{0\} \cup \mathcal{K}$, where q_0 is assumed to be positive. Note that $\Pr\{K > 0\} = 1 - q_0$ is the success probability of the Bernoulli process, describing the requests for service of jobs. The system and/or the customers will not accept that the lead time for a job exceeds an industry standard (Duenyas and Hopp 1995) which is $N + 1$ base periods. Therefore the set $\mathcal{I}$ of states at the observation epochs is the integers from 0 to N. State i identifies an observation epoch where the backlog is i. Define $\mathcal{K}(i) = \{k \in \mathcal{K} |\, i + k \leq N + 1\}$ and let M be the largest integer i for which $\mathcal{K}(i)$ is not empty.

The profit margin $p_{i,k}(\mathbf{x})$ in state i for a submitted job with processing time k is some function of a decision vector $\mathbf{x}$ having components $x_l,\ l = 1, 2, ..., n$. The decision set $\mathcal{X}$ is the n-dimensional Euclidean space of such vectors. The net benefit $B(i, k)$ of the job is the difference between the customer's random benefit and the direct cost incurred by the system if the job is submitted. The direct cost is assumed to be deterministic. Therefore the expected profit earned in state i is

$$r_i(\mathbf{x}) = \sum_{k \in \mathcal{K}(i)} q_k \Pr\{B(i, k) \geq p_{i,k}(\mathbf{x})\} p_{i,k}(\mathbf{x})$$

and the one-step transition probabilities from state $i > 0$ are

$$P_{i,i-1+k}(\mathbf{x}) = \begin{cases} Q_i(\mathbf{x}), & k = 0 \\ q_k \Pr\{B(i, k) \geq p_{i,k}(\mathbf{x})\}, & k \in \mathcal{K}(i) \\ 0, & \text{else} \end{cases}$$

where

$$Q_i(\mathbf{x}) = 1 - \sum_{k \in \mathcal{K}(i)} q_k \Pr\{B(i, k) \geq p_{i,k}(\mathbf{x})\}.$$

Note that $P_{ij}(\mathbf{x}) = 0$ if $j < i-1$, i.e. the transitions are skip-free to the left. The positive one-step transition probabilities from state 0 are $P_{00}(\mathbf{x}) = Q_0(\mathbf{x}) + q_1 \Pr\{B(0,1) \geq p_{0,1}(\mathbf{x})\}$ and $P_{0,k-1}(\mathbf{x}) = q_k \Pr\{B(0,k) \geq p_{0,k}(\mathbf{x})\},\ k \in \mathcal{K}(0) \setminus \{1\}$.

We apply a slightly modified version of the Newton-Raphson method to maximize the gain. This method assumes that the gain function $\gamma(\mathbf{x})$ is twice differentiable, which is the case when, for $i = 0, 1, ..., M$ and $k \in \mathcal{K}(i)$, the customer's benefit $B(i,k)$ has a differentiable probability density function and $p_{i,k}(\mathbf{x})$ is twice differentiable. Let D_l denote the operator taking the first partial derivative with respect to x_l, and note that $D_{k,l} = D_k D_l = D_l D_k$ is the operator taking the second partial derivative with respect to x_k and x_l. For any $\mathbf{x} \in \mathcal{X}$, the Newton-Raphson method uses the gradient $\mathbf{G}(\mathbf{x})$ and the Hessian $\mathbf{H}(\mathbf{x})$ of the gain function at $\mathbf{x}$ to compute a hopefully improved decision vector $\hat{\mathbf{x}}$.

Recall that the gain $\gamma(\mathbf{x})$ is the ratio of the expected reward $R_0(\mathbf{x})$ earned during a cycle and the expected length $T_0(\mathbf{x})$ of one cycle. Therefore $\mathbf{G}(\mathbf{x})$ is the n-dimensional vector with components

$$G_l(\mathbf{x}) = D_l \gamma(\mathbf{x}) = \frac{D_l V_0(\mathbf{x})}{T_0(\mathbf{x})} \tag{1}$$

and $\mathbf{H}(\mathbf{x})$ is the $n \times n$ matrix with components

$$H_{k,l}(\mathbf{x}) = D_{k,l} \gamma(\mathbf{x}) = \frac{D_{k,l} V_0(\mathbf{x}) - G_k(\mathbf{x}) D_l T_0(\mathbf{x}) - G_l(\mathbf{x}) D_k T_0(\mathbf{x})}{T_0(\mathbf{x})} \tag{2}$$

where $V_0(\mathbf{x})$ is defined with $g = \gamma(\mathbf{x})$ by $V_0(\mathbf{x}) = R_0(\mathbf{x}) - g T_0(\mathbf{x})$. We shall generalize this definition with the same g to specify the value $V_i(\mathbf{x})$ of being in any state $i \in \mathcal{I}$ when decision $\mathbf{x}$ is applied. We define $V_i(\mathbf{x}) = R_i(\mathbf{x}) - g T_i(\mathbf{x})$ where $R_i(\mathbf{x})$ is the expected reward earned until the first visit into state 0 and $T_i(\mathbf{x})$ is the expected time until this visit. By conditioning on the next state following state i, it can be concluded that

$$T_i(\mathbf{x}) = 1 + \sum_{j \in \mathcal{J}} P_{ij}(\mathbf{x}) T_j(\mathbf{x}),\ i \in \mathcal{I}, \tag{3}$$

and

$$V_i(\mathbf{x}) = r_i(\mathbf{x}) - g + \sum_{j \in \mathcal{J}} P_{ij}(\mathbf{x}) V_j(\mathbf{x}),\ i \in \mathcal{I}, \tag{4}$$

where $\mathcal{J} = \mathcal{I} \setminus \{0\}$. The expected times $T_i(\mathbf{x})$ and the state values $V_i(\mathbf{x})$ for decision $\mathbf{x}$ are the unique solutions to the linear equations in (3) and (4), respectively.

For any $\mathbf{x} \in X$, the strong unichain assumption holds for the Markov chain describing the states at the observation epochs because $q_0 = \Pr\{K = 0\} > 0$. Therefore this chain has equilibrium probabilities $\pi_j(\mathbf{x})$, which are the unique solution to the equilibrium equations $\pi_j(\mathbf{x}) = \sum_{i \in \mathcal{I}} \pi_i(\mathbf{x}) P_{ij}(\mathbf{x}),\ j \in \mathcal{I}$, and the normalizing equation $\sum_{j \in \mathcal{I}} \pi_j(\mathbf{x}) = 1$ (Tijms 1994, Theorem 2.3.3). The expected number of visits into state j during a cycle is $f_j(\mathbf{x}) = \pi_j(\mathbf{x}) / \pi_0(\mathbf{x})$. Starting with $f_0(\mathbf{x}) = 1$, we conclude from Corollary 2.3.6 in Tijms (1994) that $f_j(\mathbf{x})$ can be computed recursively by

$$f_j(\mathbf{x}) = \frac{\sum_{i=0}^{j-1} f_i(\mathbf{x}) \sum_{k=j}^{N} P_{ik}(\mathbf{x})}{P_{j,j-1}(\mathbf{x})},\ j = 1, 2, ..., N. \tag{5}$$

The expected length of one cycle is $T_0(\mathbf{x}) = \sum_{i\in\mathcal{I}} f_i(\mathbf{x})$ and the expected reward earned during a cycle is $R_0(\mathbf{x}) = \sum_{i\in\mathcal{I}} f_i(\mathbf{x})r_i(\mathbf{x})$. Lemma 1 offers expressions for the derivatives needed in (1) to compute the components of $\mathbf{G}(\mathbf{x})$ and in (2) to compute the components of $\mathbf{H}(\mathbf{x})$.

Lemma 1 *For fixed $\mathbf{x} \in X$ and $l = 1, 2, ...n$, the partial derivatives $D_l V_i(\mathbf{x})$, $i \in \mathcal{I}$, are the unique solution to the following linear equations in these derivatives*

$$D_l V_i(\mathbf{x}) = D_l r_i(\mathbf{x}) + \sum_{j\in\mathcal{J}} \left([D_l P_{ij}(\mathbf{x})]V_j(\mathbf{x}) + P_{ij}(\mathbf{x})D_l V_j(\mathbf{x})\right), \; i \in \mathcal{I}. \tag{6}$$

Moreover,

$$D_l V_0(\mathbf{x}) = \sum_{i\in\mathcal{I}} f_i(\mathbf{x}) \left(D_l r_i(\mathbf{x}) + \sum_{j\in\mathcal{J}} [D_l P_{ij}(\mathbf{x})]V_j(\mathbf{x}) \right), \tag{7}$$

$$D_l T_0(\mathbf{x}) = \sum_{i\in\mathcal{I}} f_i(\mathbf{x}) \sum_{j\in\mathcal{J}} [D_l P_{ij}(\mathbf{x})]T_j(\mathbf{x}) \tag{8}$$

and, for $k = 1, 2, ...n$,

$$D_{k,l} V_0(\mathbf{x}) = \sum_{i\in\mathcal{I}} f_i(\mathbf{x}) \left(D_{k,l} r_i(\mathbf{x}) + U_{i,k,l}(\mathbf{x})\right) \tag{9}$$

where

$$U_{i,k,l}(\mathbf{x}) = \sum_{j\in\mathcal{J}} \left([D_{k,l} P_{ij}(\mathbf{x})]V_j(\mathbf{x}) + [D_l P_{ij}(\mathbf{x})]D_k V_j(\mathbf{x}) + [D_k P_{ij}(\mathbf{x})]D_l V_j(\mathbf{x})\right).$$

Proof. Following Fox and Glasserman (1991), we apply the operator D_l to get (6) from (4). After multiplication with $f_i(\mathbf{x})$ in (6) and summing over all states i, we get (7) because $f_0(\mathbf{x}) = 1$ and

$$\sum_{i\in\mathcal{J}} f_i(\mathbf{x})D_l V_i(\mathbf{x}) = \sum_{j\in\mathcal{J}} f_j(\mathbf{x})D_l V_j(\mathbf{x}) = \sum_{i\in\mathcal{I}} f_i(\mathbf{x}) \sum_{j\in\mathcal{J}} P_{ij}(\mathbf{x})D_l V_j(\mathbf{x}).$$

Similarly, we get (8) and (9) from (3) and (6), respectively.

For a current decision vector $\mathbf{x} \in \mathcal{X}$ with nonzero $\mathbf{G}(\mathbf{x})$, the Newton-Raphson method (Polak 1971, page 38) assumes that $\mathbf{H}(\mathbf{x})$ is invertible and computes the hopefully improved decision vector $\hat{\mathbf{x}}$ as $\mathbf{x}+\mathbf{d}$ where the components d_l, $l = 1, 2, ..., n$, of the vector $\mathbf{d}$ are the unique solution to

$$\sum_{l=1}^{n} H_{k,l}(\mathbf{x})d_l = -G_k(\mathbf{x}), \; k = 1, 2, ..., n. \tag{10}$$

Unfortunately, for our system it cannot be concluded in general (i) that $\mathbf{H}(\mathbf{x})$ is invertible and (ii) that, when $\mathbf{G}(\mathbf{x})$ is nonzero, $\gamma(\hat{\mathbf{x}}) > \gamma(\mathbf{x})$ for the vector $\hat{\mathbf{x}}$ specified above. But it is our experience that a near-optimal decision vector can be computed rapidly by repeated use of the Newton-Raphson method, when a vector $\mathbf{x}$ satisfying

the conditions (i) and (ii) is found. It is also our experience that a steepest ascent algorithm easily finds a vector $\mathbf{x}$ for which (10) has a unique solution $\mathbf{d}$ satisfying the condition

$$\sum_{k=1}^{n}\sum_{l=1}^{n} d_k H_{k,l}(\mathbf{x}) d_l = -\sum_{k=1}^{n} d_k G_k(\mathbf{x}) < 0 \text{ if } \mathbf{d} \text{ is nonzero} \tag{11}$$

and that this inequality remains satisfied for non-optimal decision vectors with larger gain than $\gamma(\mathbf{x})$. Therefore our iterative algorithm for computing a near-optimal decision vector assumes that the inequality in (11) is satisfied for the vectors occurring during the iterations. The algorithm is started with g set equal to the gain $\gamma(\mathbf{x})$ for the initial vector $\mathbf{x}$ and it consists of the following four steps.

Modified Newton-Raphson algorithm

Step 1 For the current vector $\mathbf{x}$, compute the gradient $\mathbf{G}(\mathbf{x})$ and its norm $||\mathbf{G}(\mathbf{x})|| = \sqrt{\sum_{l=1}^{n} G_l(\mathbf{x})^2}$. Stop if $||\mathbf{G}(\mathbf{x})||$ is less than some pre-specified tolerance. Otherwise, compute the Hessian $\mathbf{H}(\mathbf{x})$ and compute $\mathbf{d}$ as the solution to (10).

Step 2 For $\hat{\mathbf{x}} = \mathbf{x} + \mathbf{d}$, compute the gain $\hat{g} = \gamma(\hat{\mathbf{x}})$.

Step 3 While $\hat{g} \leq g$, bisect $\mathbf{d}$ and repeat Step 2.

Step 4 Go to Step 1 with $\mathbf{x}$ and g set equal to $\hat{\mathbf{x}}$ and $\hat{g}$, respectively.

Our procedures to be used in Step 1 for solving (3), (4) and (6) exploit that the transitions are skip-free to the left. The procedures are described in Appendix A. We apply Gauss elimination in Step 1 to solve (10). We have investigated various approaches for what to do whenever in Step 2 it turns out that $\hat{g} \leq g$. It is our experience that the simple approach described in Step 3 performs well and that bisections of $\mathbf{d}$ are often prescribed in the first iteration only, or not at all. It is also our experience, even for a small pre-specified tolerance, that our algorithm needs only few iterations to compute a near-optimal decision vector.

4 A numerical example

We shall illustrate how the modified Newton-Raphson algorithm can be used to compute profit margins for the numerical example presented in Johansen (1991). Requests for service of jobs are Bernoulli with the success probability $\frac{9}{10}$, i.e. $q_0 = 0.10$. The processing times of the jobs belong to the set $\mathcal{K} = \{2, 3, 4\}$ and the conditional probabilities $\Pr\{K = k | K \in \mathcal{K}\}$ are $\frac{3}{10}$, $\frac{4}{10}$ and $\frac{3}{10}$ for the three elements in $\mathcal{K}$, i.e. $q_2 = q_4 = 0.27$ and $q_3 = 0.36$. No customer will accept that the lead time exceeds 15 base periods, i.e. $M = 13$ and $N = 14$. The net benefit $B(i,k)$ is uniformly distributed over the interval $[0, 100k]$. The maximum gain for unrestricted prices is 59.491 (see Table 3). This is obtained when the profit margin in state i for a job with processing time k is $(C_{\mathrm{OPP}}(i,k) + 100k)/2$ where $C_{\mathrm{OPP}}(i,k) = v_{i-1} - v_{i-1+k}$ is an opportunity cost which measures the congestion effect of the job. Here $v_{-1} = v_0 = 0$

iteration	$\mathbf{G}(\mathbf{x})$	$\|\|\mathbf{G}(\mathbf{x})\|\|$	b	$\hat{\mathbf{x}}$	$\hat{g}$
1	(11.7987,13.2747,7.6082)	19.3212	2	(123.08,198.11,291.51)	56.594
2	(-2.5038,-6.0864,-7.7604)	10.1754	0	(125.57,190.61,257.47)	58.125
3	(0.2528,0.1195,-0.2262)	0.3596	0	(126.52,191.14,255.58)	58.129
4	(-0.0005,-0.0007,-0.0005)	0.0010	0	(126.52,191.14,255.58)	58.129

Table 1: The values computed and the numbers b of bisections prescribed during 4 iterations of the modified Newton-Raphson algorithm started with $\mathbf{x} = (100, 150, 200)$ and $g = 49.784$.

and, for $i > 0$, v_i is the value of being in state i rather than in state 0 when the optimal policy is applied. The reference has computed $\frac{v_{i-1}-v_i}{100}$ for $i = 0, 1, .., 14$.

Suppose that the prices are not allowed to depend on the actual backlog. Let $\mathcal{X}$ be the set of three-dimensional vectors $\mathbf{x}$ and let component x_{k-1} prescribe the profit margin for a submitted job with processing time $k \in \mathcal{K}$. If the opportunity costs caused by the congestion effect are neglected, then the best margin vector is $\mathbf{x} = (100, 150, 200)$ and the gain is $\gamma(\mathbf{x}) = 49.784$. This gain is 16.32% less than the maximum gain 59.491 for unrestricted prices. Hence ignorance of the congestion effect has a severe impact on the profit. Starting with the computed vector $\mathbf{x}$ and $g = 49.784$, we obtain during 4 iterations of our algorithm the results reported in Table 1. If the tolerance for the norm $||\mathbf{G}(\mathbf{x})||$ is below 0.3596 and above 0.0010, then the algorithm is stopped in Step 1 of the fourth iteration and the margin vector computed is $(126.52, 191.14, 255.58)$ with gain 58.129. This gain is 2.29% less than the maximum gain 59.491 for unrestricted prices. If each component of the computed vector is rounded to the nearest integer, then the gain becomes $\gamma(127, 191, 256) = 58.128$. The very small gain reduction caused by this rounding indicates that the function γ is flat around its maximum.

In the numerical example, the net benefit $B(i, k)$ is (stochastically) proportional to k when $i+k \leq 15$. Therefore it is not surprising that the best decision computed by the algorithm lets the mark-ups for the jobs be almost proportional to their processing times. This result makes it interesting to investigate what happens for $n = 1$ when the profit margin function is $p_{i,k}(x) = kx$ where x specifies a common mark-up per base period. The modified Newton-Raphson algorithm computes that the best common mark-up per base period is $x = 63.68$ and that the gain with this mark-up is 58.127. It is very close to the maximum gain 58.129 for the best margins computed before.

We shall now show that the gain for the numerical example can be increased by introducing a pair (x_1, x_2) of mark-ups per base period and a threshold z for the backlog. We consider three different methods for computing the backlog to be compared with z for each job. The first method (BEFORE) focuses on the backlog i upon request and its margin function is

$$p_{i,k}(x_1, x_2) = \begin{cases} kx_1 & \text{if } i < z \\ kx_2 & \text{else.} \end{cases}$$

The second method (INCREMENTAL) focuses on how the backlog is increased and

	BEFORE		INCREMENTAL		AFTER	
z	(x_1, x_2)	gain	(x_1, x_2)	gain	(x_1, x_2)	gain
0	$(-, 63.7)$	58.127	$(-, 63.7)$	58.127	$(-, 63.7)$	58.127
1	$(53.4, 65.1)$	58.584	$(32.0, 65.0)$	58.568	$(-, 63.7)$	58.127
2	$(54.4, 65.8)$	58.764	$(46.6, 65.6)$	58.729	$(51.3, 64.0)$	58.240
3	$(55.6, 66.6)$	58.910	$(51.1, 66.4)$	58.904	$(52.7, 64.7)$	58.471
4	$(56.7, 67.4)$	59.014	$(53.4, 67.2)$	59.048	$(54.2, 65.7)$	58.735
5	$(57.6, 68.3)$	59.076	$(55.0, 68.1)$	59.149	$(55.4, 66.4)$	58.884
6	$(58.5, 69.2)$	**59.104**	$(56.3, 69.1)$	59.213	$(56.5, 67.2)$	58.995
7	$(59.3, 70.2)$	59.098	$(57.4, 70.2)$	59.245	$(57.4, 68.1)$	59.066
8	$(60.1, 71.3)$	59.056	$(58.3, 71.4)$	**59.246**	$(58.3, 69.0)$	59.101
9	$(60.8, 72.7)$	58.974	$(59.2, 72.9)$	59.215	$(59.1, 70.0)$	**59.104**
10	$(61.5, 74.1)$	58.838	$(60.0, 74.7)$	59.147	$(59.9, 71.1)$	59.071
11	$(62.3, 75.7)$	58.629	$(60.8, 77.2)$	59.032	$(60.6, 72.4)$	58.999
12	$(63.1, 77.3)$	58.347	$(61.5, 81.4)$	58.867	$(61.3, 73.9)$	58.886
13	$(63.6, 78.5)$	58.172	$(62.3, 88.9)$	58.654	$(62.1, 75.5)$	58.694
14	$(63.7, -)$	58.127	$(63.0, 106.1)$	58.413	$(62.9, 77.0)$	58.423

Table 2: The best pair (x_1, x_2) of two mark-ups per base period for the three pricing methods and different thresholds z.

its margin function is

$$p_{i,k}(x_1, x_2) = \sum_{j=i}^{i+k-1} [x_1 + (x_2 - x_1)\mathbf{1}(j \geq z)]$$

where the indicator $\mathbf{1}(j \geq z)$ is 1 if $j \geq z$ and 0 else. The third method (AFTER) focuses on the lead time $i + k$ (the backlog if the job is submitted) and its margin function is

$$p_{i,k}(x_1, x_2) = \begin{cases} kx_1 & \text{if } i + k \leq z \\ kx_2 & \text{else.} \end{cases}$$

For each of the three methods and $z = 0, 1, ..., 14$, we have used the modified Newton-Raphson algorithm with $\mathcal{X}$ specified as the set of all pairs (x_1, x_2) to compute the best pair and its gain. Our results are reported in Table 2 where a bar for one of the components in a pair indicates that this component has no influence on the gain. For example, if $z = 0$, then the gain is independent of x_1, and all the three methods specify that the best value of x_2 is 63.7 and has gain 58.127 (confirming the result reported earlier for a common mark-up per base period). The maximum gain for each method is written in bold face. The three maxima are, respectively, 0.65%, 0.41% and 0.65% less than the maximum gain 59.491 for unrestricted prices. These small percentages tell that most of the gain reduction, caused by restricting the profit margins to be independent of the actual backlog or to be a common mark-up per base period multiplied by the processing time, can be eliminated by the considered pricing methods. The best method is INCREMENTAL, but the other two are also rather good. The three methods perform well because they allow the price of a job

to depend on the actual backlog and they demonstrate that simple specifications of this dependence can provide a gain close to the maximal one.

5 Optimizing unrestricted prices

The dimension n of the decision set $\mathcal{X}$ equals the number of all i and k combinations when the prices are unrestricted. If this number is big, the computational burden becomes overwhelming when we try to compute near-optimal unrestricted prices by the modified Newton-Raphson algorithm. Therefore we suggest to compute unrestricted prices as the solution to a Markov decision model with states i. The action in each state i is a vector $\mathbf{a}$. Its kth component is the price to be charged for a job with processing time k. We shall present a tailor-made policy-iteration algorithm for computing near-optimal prices. As in the original policy-iteration algorithm (Howard 1960), we decompose, in each iteration of our algorithm, the computation of an improved policy into the states. Hence, improved prices are computed separately for each state. This is much faster than computing improved prices for all i and k combinations simultaneously.

Our algorithm does not assume that the integers M (the largest state in which the system accepts jobs) and N (the largest state of the system) are pre-specified. These integers are now decision variables. For a pre-specified small $\epsilon > 0$, the algorithm computes the integers M and N and unrestricted prices, for which the gain is at most ϵ below the maximum gain. The algorithm assumes that the net benefit $B(i,k)$ is distributed as the difference between the customer's benefit $(\tilde{B} - \tilde{h}i)k$ and the direct cost $\tilde{c}k$ incurred by the system if a job with processing time k is submitted. Here $\tilde{h}$ and $\tilde{c}$ are nonnegative constants and $\tilde{B}$ is a random variable, having differentiable probability density function f with increasing failure rate. Let $\Re$ denote the real numbers. For any $s \in \Re$, the failure (or hazard) rate is $f(s)/\bar{F}(s)$ where $\bar{F}(s) = \int_s^\infty f(t)\,dt = \Pr\{\tilde{B} \geq s\}$. The normal and many other distributions have increasing failure rates (Barlow and Proschan 1975).

Before presenting the details of our policy-iteration algorithm, we shall explain how it differs from Howard's algorithm (as it is described by Tijms (1994, pages 193-194)). The latter algorithm assumes that the state and action spaces are finite. Each iteration of the algorithm consists of three steps: value determination (of the initial policy in the first iteration and otherwise of the policy from the previous iteration), value improvement and convergence test. Our algorithm allows infinite state and action spaces. Rather than starting the iterations with a policy, our algorithm starts each iteration with an estimate g of the maximal gain. This estimate is specified from the outset in the first iteration. Otherwise it is the gain of the policy found in the previous iteration. The first two steps in our algorithm are designed to compute the integers M and N and to compute a value v_i for each state i from N to 0 (and in that order except if and when the initial N is increased in Step 2). The second step also computes actions prescribing a pricing policy which has gain g and is $\frac{\epsilon}{2}$-optimal for an adapted version of the considered model. The adapted model has the same characteristics as the considered model but, in order to obtain the gain g and to get monotone state values, the adapted model also incorporates shadow fees. We know

from Theorem 4.2 in Johansen (1991) that the state values for the optimal policy are non-increasing in i. We ensure that the state values of the adapted model have the same monotonicity property by incorporating a shadow fee y_i for the states i from m to 0. Here m is the largest state i where the property is violated (which is indicated by a negative number Y_i computed for state i), before the model is adapted. If the property is not violated for a positive state i, then m is set equal to 0. The third step in our algorithm computes the true gain for the policy found in the second step. The fourth step is a convergence test. We have experienced from numerous numerical examples that the shadow fees introduced by us make the number of iterations needed to compute the desired ϵ-optimal prices small, even when the algorithm is started with an arbitrarily chosen g-value.

Suppose that k_i is the largest processing time which is accepted for jobs submitted in state i. State i's action set $\mathcal{A}(k_i)$ is the vectors $\mathbf{a}$ with components a_k, $k = 1, 2, ..., k_i$. The expected profit earned in state i by choosing the action $\mathbf{a} \in \mathcal{A}(k_i)$ is

$$r_i(k_i, \mathbf{a}) = \sum_{k=1}^{k_i} q_k \bar{F}\left(\tilde{h}i + \frac{a_k}{k}\right)(a_k - \tilde{c}k)$$

and if $i > 0$ then the transition probabilities are

$$P_{ij}(k_i, \mathbf{a}) = \begin{cases} Q_i(k_i, \mathbf{a}), & j = i - 1 \\ q_{j-i+1}\bar{F}\left(\tilde{h}i + \frac{a_{j-i+1}}{j-i+1}\right), & j = i, i+1, ..., i+k_i-1 \\ 0, & \text{else} \end{cases}$$

where

$$Q_i(k_i, \mathbf{a}) = 1 - \sum_{k=1}^{k_i} q_k \bar{F}\left(\tilde{h}i + \frac{a_k}{k}\right), \ \mathbf{a} \in \mathcal{A}(k_i).$$

State 0's transition probability $P_{0j}(k_0, \mathbf{a})$ is $Q_0(k_0, \mathbf{a}) + q_1\bar{F}(a_1)$ if $j = 0$ and $q_{j+1}\bar{F}\left(\frac{a_{j+1}}{j+1}\right)$ if $1 \le j \le k_0 - 1$.

As already explained, the iterations of our algorithm start with some estimate g of the maximum gain. Each iteration consists of the following four steps.

Tailor-made policy-iteration algorithm

Step 1 Compute the integer M so that

$$\bar{F}(\tilde{h}(M+1) + s)(s - \tilde{c} - g) \le \frac{\epsilon}{2E[K]} \text{ for all } s \in \Re. \tag{12}$$

Set $N = M$ and $Y_{M+1} = g$. When needed in Step 2, compute v_i as $g(M - i)$ for $i \ge M$.

Step 2 Recursively for $i = M, M-1, ..., 0$, first compute the largest processing time k_i, the action $\mathbf{x}_i \in \mathcal{A}(k_i)$ and the number $Y_i \le Y_{i+1}$ so that

$$\bar{F}(\tilde{h}i + s)\left(s - \tilde{c} - \frac{Y_i + v_i - v_{i+k_i-1}}{k_i}\right) \le \frac{\epsilon}{4E[K\mathbf{1}(K > k_i)]} \text{ for all } s \in \Re, \tag{13}$$

$$S_i(k_i, \mathbf{x}_i) + Q_i(k_i, \mathbf{x}_i)Y_i = v_i \tag{14}$$

and, for all $\mathbf{a} \in \mathcal{A}(k_i)$,

$$S_i(k_i, \mathbf{a}) + Q_i(k_i, \mathbf{a})Y_i \leq v_i + \frac{\epsilon}{4} \tag{15}$$

where

$$S_i(k_i, \mathbf{a}) = r_i(k_i, \mathbf{a}) - g + \sum_{j=0}^{i+k_i-1} P_{ij}(k_i, \mathbf{a})v_{\max\{i,j\}}. \tag{16}$$

Second, if $N < i + k_i - 1$, set $N = i + k_i - 1$. And third, if $i > 0$, set $v_{i-1} = \max\{Y_i, 0\} + v_i$ and $y_i = \max\{-Y_i, 0\}$. If $i = 0$, then set $y_0 = -Y_0$.

Step 3 If $y_0 > 0$ and in the first iteration also if $y_0 < -\frac{\epsilon}{2}$, first compute the equilibrium probabilities π_j for the Markov chain having transition probabilities $P_{ij}(k_i, \mathbf{x}_i)$. Next, compute

$$\hat{g} = g + \sum_{j=0}^{M} \pi_j Q_j(k_j, \mathbf{x}_j)y_j \tag{17}$$

and set g equal to $\hat{g}$.

Step 4 Let m denote the largest state i for which y_i is nonzero. Stop in the first iteration, if $m = 0$ and $-\frac{\epsilon}{2} \leq y_0 \leq \frac{\epsilon}{2}$. Stop in later iterations, if $m = 0$ and $y_0 \leq \frac{\epsilon}{2}$. Otherwise, go to Step 1.

Our procedures for the computations in Steps 1 and 2 are described in Appendix B. The equilibrium probabilities in Step 3 are computed by (5) and the normalizing equation. We shall use the following lemma to argue that the pricing policy computed in Step 2 is $\frac{\epsilon}{2}$-optimal for the adapted model. This result provides that an ϵ-optimal policy is found in the iteration where the algorithm stops in Step 4.

We consider how the adapted model is specified in terms of the shadow fees y_i (which equal 0 for $i > m$). For any integers $i \geq 0$ and $k > 0$ and any action $\mathbf{a} \in \mathcal{A}(k)$, the adapted reward is

$$\hat{r}_i(k, \mathbf{a}) = r_i(k, \mathbf{a}) - Q_i(k, \mathbf{a})y_i$$

and we define

$$\hat{V}_i(k, \mathbf{a}) = \hat{r}_i(k, \mathbf{a}) - g + \sum_{j=0}^{i+k-1} P_{ij}(k, \mathbf{a})v_j.$$

It follows from (14) that

$$\hat{V}_i(k_i, \mathbf{x}_i) = v_i \text{ for } i = 0, 1, ..., M. \tag{18}$$

Lemma 2 *For any integers $i \geq 0$ and $k > 0$,*

$$\hat{V}_i(k, \mathbf{a}) - \frac{\epsilon}{2} \leq v_i \text{ for all } \mathbf{a} \in \mathcal{A}(k). \tag{19}$$

Proof. If $i > M$ then

$$\hat{V}_i(k,\mathbf{a}) - v_i \leq \hat{V}_{M+1}(k,\mathbf{a}) - v_{M+1} = \sum_{l=1}^{k} l q_l \bar{F}\Big(\tilde{h}(M+1) + \frac{a_l}{l}\Big)\Big(\frac{a_l}{l} - \tilde{c} - g\Big) \leq \frac{\epsilon}{2}$$

where the last inequality follows from (12). If $i \leq M$ and $k \leq k_i$, then (19) follows trivially from (15). Finally, if $i \leq M$ and $k > k_i$, then

$$\hat{V}_i(k,\mathbf{a}) \leq v_i + \frac{\epsilon}{4} + \sum_{l=k_i+1}^{k} q_l \bar{F}\Big(\tilde{h}i + \frac{a_l}{l}\Big)(a_l - [\tilde{c}l + Y_i + v_i - v_{i+l-1}]) \qquad (20)$$

where the inequality is established by (15). Observe for $l > k_i$ that

$$Y_i + v_i - v_{i+l-1} = \sum_{j=i}^{i+l-1} u_j \geq \frac{l}{k_i} \sum_{j=i}^{i+k_i-1} u_j = \frac{l}{k_i}(Y_i + v_i - v_{i+k_i-1})$$

where $u_i = Y_i$ and $u_j = v_{j-1} - v_j \geq u_{j-1},\ j = i+1, i+2, ..., i+l-1$. Therefore, the last expression in (20) is at most

$$\sum_{l=k_i+1}^{k} l q_l \bar{F}\Big(\tilde{h}i + \frac{a_l}{l}\Big)\left(\frac{a_l}{l} - \left[\tilde{c} + \frac{Y_i + v_i - v_{i+k_i-1}}{k_i}\right]\right).$$

Because (13) provides that $\frac{\epsilon}{4}$ is an upper bound for this expression, we conclude from (20) that the inequality in (19) is valid.

We obtain from (18) and Theorem 3.2.1 in Tijms (1994) that the gain is g for the adapted model under the policy specified by the computed actions $\mathbf{x}_i$. Moreover, the same theorem and our Lemma 2 provide that $g + \frac{\epsilon}{2}$ is an upper bound for the gain of the adapted model under any stationary policy. Hence the computed actions $\mathbf{x}_i$ specify an $\frac{\epsilon}{2}$-optimal policy for the adapted model. For the un-adapted model, the gain $\hat{g}$ under this policy is specified by (17). If $m = 0$ then $g + \frac{\epsilon}{2} + \max\{y_0, 0\}$ is an upper bound for the maximum gain. Therefore, the algorithm is stopped in Step 4 of the first iteration (where g can be any scalar) if $-\frac{\epsilon}{2} \leq y_0 \leq \frac{\epsilon}{2}$ because then an ϵ-optimal policy is found. When the algorithm is stopped in Step 4 of a later iteration, then either the last policy found is ϵ-optimal because $-\frac{\epsilon}{2} \leq y_0 \leq \frac{\epsilon}{2}$ or else the policy found in the previous iteration is ϵ-optimal.

The tailor-made policy-iteration algorithm becomes simpler when the set $\mathcal{K}$ and the integer N are pre-specified. Then M is the largest integer i, for which $\mathcal{K}(i)$ is not empty and k_i is the largest element in $\mathcal{K}(i)$, i.e. the computations of M satisfying (12) in Step 1 and of k_i satisfying (13) in Step 2 are not needed. If $\tilde{B}$ is uniformly distributed, then for each i we can in Step 2 explicitly compute

$$Y_i = \min_{\mathbf{a}\in\mathcal{A}(k_i)} \left\{ \frac{v_i - S_i(k_i,\mathbf{a})}{Q_i(k_i,\mathbf{a})} \right\} \qquad (21)$$

and fix $\mathbf{x}_i$ as the vector $\mathbf{a}$ for which the minimum is attained.

iteration	$y_0, ..., y_m$	$\hat{g}$
1	$33.9, 33.9, 33.9, 33.9, 33.9, 33.9, 33.9, 30.9, 21.4, 4.2$	56.402
2	$20.81, 20.81, 20.81, 19.39, 14.21, 4.31$	58.549
3	$15.343, 10.820, 2.431$	59.328
4	3.5870	59.485
5	0.12204	59.491
6	0.000156	59.491

Table 3: The first $1 + m$ components of the shadow fee vector $\mathbf{y}$ and the gain $\hat{g}$ computed during 6 iterations of the tailor-made policy-iteration algorithm started with $g = 49.784$.

	start with $g = 59.104$		start with $g = 59.246$	
iteration	$y_0, ..., y_m$	$\hat{g}$	$y_0, ..., y_m$	$\hat{g}$
1	$9.0241, 0.3774$	59.457	5.584327	59.477
2	0.701220	59.490	0.280407	59.490
3	0.005056	59.491	0.000818	59.491

Table 4: The first $1 + m$ components of the shadow fee vector $\mathbf{y}$ and the gain $\hat{g}$ computed during 3 iterations of the tailor-made policy-iteration algorithm started with $g = 59.104$ and $g = 59.246$.

We have implemented our algorithm with the above mentioned simplifications for the numerical example in §4. Table 3 reports the first $1 + m$ components (with the accuracy depending on the iteration number) of the fee vector $\mathbf{y}$ computed in Step 2 and the gain $\hat{g}$ computed in Step 3 during 6 iterations, when the algorithm is started with $g = 49.784$ (the gain of the price policy which neglects the congestion effect). If the desired accuracy ϵ for the gain is below 0.24408 and above 0.000312, then the algorithm is stopped in Step 4 of the last reported iteration where the computed gain is 59.491, confirming the information in §4 about the maximum gain.

We know from Table 2 that the maximum gain for the pricing methods BEFORE and AFTER is 59.104, whereas the maximum gain for INCREMENTAL is 59.246. If we start with g set equal to these values, then during 3 iterations of the algorithm we get the results reported in Table 4.

Tables 3 and 4 illustrate that our tailor-made policy-iteration algorithm rapidly finds a near-optimal policy. It is our experience that the algorithm is much faster than the fixed-point algorithm presented by Johansen (1991).

6 Conclusion

We have analyzed and illustrated two models for the pricing of customized products, which are called jobs by us. For the model with profit margins specified in terms of

some basic decision variables, we presented a modified Newton-Raphson algorithm for optimizing the decision variables. For the model with unrestricted prices we presented a tailor-made policy-iteration algorithm relying on some shadow fees introduced by us. Both algorithms were designed for renewal-reward optimization and they can easily be adapted to other models than the two analyzed in this paper.

Our numerical example illustrated that few carefully chosen decision variables suffice to obtain a gain close to the maximal one. We introduced a pair (x_1, x_2) of mark-ups per base period and a threshold z for the backlog. We used three different methods for computing the backlog to be compared with z for each job and we illustrated how they perform for the numerical example when z is varied. All three methods performed rather well for their best z-value.

To be published elsewhere we have developed a modified Newton-Raphson algorithm for optimizing a pair $(\mathbf{x}, \mathbf{z})$ of decision vectors where $\mathbf{x}$ has real-valued components (like x_1 and x_2 in our example), whereas $\mathbf{z}$ has integer components (like z in our example).

Appendices

A Procedures for the modified Newton-Raphson algorithm

We shall describe the procedures used by the modified Newton-Raphson algorithm to compute, for the current decision vector $\mathbf{x}$ and each $i \in \mathcal{I}$, the expected time $T_i(\mathbf{x})$ and the state value $V_i(\mathbf{x})$ (Procedure A1) and the partial derivatives $D_l V_i(\mathbf{x})$ (Procedure A2.l) for $l = 1, 2, ..., n$.

Because the transitions are skip-free to the left, for $t_0 = -T_N(\mathbf{x})$ and $t_i = T_i(\mathbf{x}) + t_0,\ i \in \mathcal{J}$, it can be concluded from (3) that

$$t_{i-1} = \frac{t_i - 1 - \sum_{j=i}^{N} P_{ij}(\mathbf{x}) t_j}{P_{i,i-1}(\mathbf{x})},\ i \in \mathcal{J}. \tag{22}$$

Moreover, for $v_i = V_i(\mathbf{x}) - V_N(\mathbf{x}),\ i \in \mathcal{I}$, it can be concluded from (4) that

$$v_{i-1} = \frac{v_i - r_i(\mathbf{x}) + g - \sum_{j=i}^{N} P_{ij}(\mathbf{x}) v_j}{P_{i,i-1}(\mathbf{x})},\ i \in \mathcal{J}. \tag{23}$$

Starting with $t_N = 0$ and $v_N = 0$, Procedure A1 first uses (22) and (23) to compute t_{i-1} and v_{i-1} recursively for $i = N, N-1, ..., 1$. Next, the procedure computes $T_0(\mathbf{x}) = \sum_{j=0}^{N} f_j(\mathbf{x})$ $(= 1 + \sum_{j=1}^{N} P_{0j}(\mathbf{x})(t_j - t_0)])$, $V_0(\mathbf{x}) = 0$ and, for $i = 1, 2, ..., N$, $T_i(\mathbf{x}) = t_i - t_0$ and $V_i(\mathbf{x}) = v_i - v_0$.

For $l = 1, 2, ..., n$, define $w_{l,0} = -D_l V_N(\mathbf{x})$ and $w_{l,i} = D_l V_i(\mathbf{x}) + w_{l,0},\ i \in \mathcal{J}$. It can be concluded from (6) that

$$w_{l,i-1} = \frac{w_{l,i} - D_l r_i(\mathbf{x}) - \sum_{j=i-1}^{N} [D_l P_{ij}(\mathbf{x})] V_j(\mathbf{x}) - \sum_{j=i}^{N} P_{ij}(\mathbf{x}) w_{l,j}}{P_{i,i-1}(\mathbf{x})},\ i \in \mathcal{J}. \tag{24}$$

Starting with $w_{l,N} = 0$, Procedure A2.l first uses (24) to compute $w_{l,i-1}$ recursively for $i = N, N-1, ..., 1$. Next, the procedure computes $D_l V_0(\mathbf{x})$ by (7) and $D_l V_i(\mathbf{x}) = w_{l,i} - w_{l,0}$ for $i = 1, 2, ..., N$.

B Steps 1 and 2 of the tailor-made policy iteration algorithm

The computations in Steps 1 and 2 of the tailor-made policy-iteration algorithm rely on the following evaluation of the expected net profit earned per unit time in state i. Suppose that the state values v_j are computed for $j \geq i$. Consider a job with processing time k. Define

$$c_{i,k} = \tilde{c} + \frac{v_i - v_{i+k-1}}{k}$$

and

$$\Delta_{i,k}(s, Y) = \int_{\tilde{h}i+s}^{\infty} \left(\rho(t) - \tilde{h}i - c_{i,k} - \frac{Y}{k} \right) f(t)\, dt, \ (s, Y) \in \Re^2,$$

where

$$\rho(t) = t - \frac{\bar{F}(t)}{f(t)}$$

is an expected marginal revenue per unit time. Observe that $\rho'(t) > 1$ because f is differentiable with increasing failure rate and that

$$\begin{aligned} \Delta_{i,k}(s, Y) &= \int_{\tilde{h}i+s}^{\infty} \left(t - \tilde{h}i - c_{i,k} - \frac{Y}{k} \right) f(t)\, dt - \int_{\tilde{h}i+s}^{\infty} \bar{F}(t)\, dt \\ &= \left(s - c_{i,k} - \frac{Y}{k} \right) \bar{F}(\tilde{h}i + s) \end{aligned} \tag{25}$$

where integration by parts establishes the second equality.

Lemma 3 *Suppose that s_i is chosen so that $\rho(\tilde{h}i + s_i) \leq \tilde{h}i + c_{i,k} + \frac{Y}{k}$. Then*

$$\Delta_{i,k}(s, Y) \leq \frac{\bar{F}(\tilde{h}i + s_i)^2}{f(\tilde{h}i + s_i)} \text{ for all } s \in \Re.$$

Proof. Define $s^* = \rho^{-1}(\tilde{h}i + c_{i,k} + \frac{Y}{k}) - \tilde{h}i$ where ρ^{-1} is the inverse of ρ. For all $s \in \Re$, observe that

$$\Delta_{i,k}(s, Y) \leq \Delta_{i,k}(s^*, Y) = \Delta_{i,k}(s_i, Y) + \int_{\tilde{h}i+s_i}^{\tilde{h}i+s^*} \left(\tilde{h}i + c_{i,k} + \frac{Y}{k} - \rho(t) \right) f(t)\, dt$$

and that $s_i \leq s^*$. Hence, for $\tilde{h}i + s_i \leq t \leq \tilde{h}i + s^*$, the expression in parenthesis is at most

$$\tilde{h}i + c_{i,k} + \frac{Y}{k} - \rho(\tilde{h}i + s_i) = c_{i,k} + \frac{Y}{k} - s_i + \frac{\bar{F}(\tilde{h}i + s_i)}{f(\tilde{h}i + s_i)} \geq 0$$

and it can be concluded that

$$\Delta_{i,k}(s^*, Y) \leq \left(s_i - c_{i,k} - \frac{Y}{k} \right) \bar{F}(\tilde{h}i + s_i) + \left(c_{i,k} + \frac{Y}{k} - s_i + \frac{\bar{F}(\tilde{h}i + s_i)}{f(\tilde{h}i + \hat{s})} \right) \bar{F}(\tilde{h}i + \hat{s}).$$

The expression to the right of the inequality equals the desired upper bound.

Because $v_i = g(M - i)$ for $i \geq M$, it is easily seen that the expression to the left of the inequality in (12) equals $\Delta_{M+1,k}(s, g)$ for any positive integer k. Motivated by Lemma 3 our procedure for Step 1 chooses M and s_{M+1} so that the conditions

$$\rho(\tilde{h}(M+1) + s_{M+1}) \leq \tilde{h}(M+1) + \tilde{c} + g \tag{26}$$

and

$$\frac{\bar{F}(\tilde{h}(M+1) + s_{M+1})^2}{f(\tilde{h}(M+1) + s_{M+1})} \leq \frac{\epsilon}{2E[K]} \tag{27}$$

are satisfied.

In order to present our procedure for the computations in Step 2, for $0 \leq i \leq M$, $j \geq 1$ and $\mathbf{b} \in \mathcal{A}(j)$ we define

$$Z_{i,j}(\mathbf{b}) = \frac{g - \sum_{k=1}^{j} k q_k \bar{F}(\tilde{h}i + b_k)(b_k - c_{i,k})}{1 - \sum_{k=1}^{j} q_k \bar{F}(\tilde{h}i + b_k)}$$

and

$$U_{i,j}(\mathbf{b}) = \sum_{k=1}^{j} q_k \left| k[\rho(\tilde{h}i + b_k) - \tilde{h}i - c_{i,k}] - Z_{i,j}(\mathbf{b}) \right|.$$

For each state i, the purpose of the procedure is to compute the integer k_i, the vector $\mathbf{b}_i \in \mathcal{A}(k_i)$ and the scalar s_i so that

$$\rho(\tilde{h}i + s_i) \leq \tilde{h}i + c_{i,k_i} + \frac{Z_{i,k_i}(\mathbf{b}_i)}{k_i}, \tag{28}$$

$$\frac{\bar{F}(\tilde{h}i + s_i)^2}{f(\tilde{h}i + s_i)} \leq \frac{\epsilon}{4E[K\mathbf{1}(K > k_i)]} \tag{29}$$

and

$$U_{i,k_i}(\mathbf{b}_i) \leq \frac{\epsilon}{4}.$$

Then, for $Y_i = Z_{i,k_i}(\mathbf{b}_i)$, Lemma 3 provides that (13) is satisfied. Moreover, when $\mathbf{x}_i$ is the vector with components $x_{ik} = kb_{ik},\ k = 1, 2, ..., k_i$, it can be concluded that (14) and (15) are satisfied.

Procedure for state i

Step 0 If $i = M$, then set $s_M = s_{M+1}$, compute the smallest k_M satisfying (29) for $i = M$ and let $\mathbf{b} \in \mathcal{A}(k_M)$ have the components $b_k = s_M,\ k = 1, 2, ..., k_M$. If $i < M$, then set $k_i = k_{i+1} + 1$ and let $\mathbf{b} \in \mathcal{A}(k_i)$ have the components $b_k = b_{i+1,k},\ k = 1, 2, ..., k_{i+1}$, and $b_{k_i} = s_{i+1}$.

Step 1 While $U_{i,k_i}(\mathbf{b}) > \frac{\epsilon}{4}$, repeat to replace $\mathbf{b}$ with a vector $\hat{\mathbf{b}}$ (computed as described below) having the property $Z_{i,k_i}(\hat{\mathbf{b}}) < Z_{i,k_i}(\mathbf{b}) < Y_{i+1}$.

Step 2 Compute s_i so that (28) is satisfied for $\mathbf{b}_i = \mathbf{b}$. If (29) is satisfied, then stop after setting $\mathbf{b}_i = \mathbf{b}$ and $Y_i = Z_{i,k_i}(\mathbf{b})$. Otherwise, first repeat to increase k_i by one and to set $b_{k_i} = s_i$ until (29) is satisfied. Next, go to Step 1.

The computation in Step 1 of $\hat{\mathbf{b}}$ for the current vector $\mathbf{b}$ relies on approximations. For each $k = 1, 2, ..., k_i$, define $\alpha_k = f(\tilde{h}i + b_k)$ and $\beta_k = b_k + \bar{F}(\tilde{h}i + b_k)/\alpha_k$. We approximate $\bar{F}(\tilde{h}i + s)$ by $\alpha_k(\beta_k - s)$ and conclude from (25) that $\Delta_{i,k}(s, Y)$ is approximated by $(s - c_{i,k} - \frac{Y}{k})\alpha_k(\beta_k - s)$. For any scalar Y, the maximum value of this quadratic function is $\alpha_k(\beta_k - b_k^*(Y))^2$, where

$$b_k^*(Y) = \frac{\beta_k + c_{i,k} + \frac{Y}{k}}{2}$$

is the s-value for which the maximum is attained. Let $\mathbf{a}^*(Y) \in \mathcal{A}(k_i)$ have the components $kb_k^*(Y)$, $k = 1, 2, ..., k_i$, and note that we can approximate

$$r_i(k_i, \mathbf{a}^*(Y)) - g + Q_i(k_i, \mathbf{a}^*(Y))Y + \sum_{j=0}^{i+k_i-1} P_{ij}(k_i, \mathbf{a}^*(Y))v_{\max\{i,j\}} - v_i$$

by

$$Y + \sum_{k=1}^{k_i} kq_k\alpha_k \left(\frac{\beta_k - c_{i,k} - \frac{Y}{k}}{2} \right)^2 - g.$$

The largest root of this quadratic function is

$$\hat{Y} = \bar{Y} + \sqrt{\bar{Y}^2 - \frac{\sum_{k=1}^{k_i} kq_k\alpha_k(\beta_k - c_{i,k})^2 - 4g}{\sum_{k=1}^{k_i} \frac{q_k\alpha_k}{k}}}$$

where

$$\bar{Y} = \frac{\sum_{k=1}^{k_i} q_k\alpha_k(\beta_k - c_{i,k}) - 2}{\sum_{k=1}^{k_i} \frac{q_k\alpha_k}{k}}.$$

The vector $\hat{\mathbf{b}}$ is computed so that

$$\left| k[\rho(\tilde{h}i + \hat{b}_k) - \tilde{h}i - c_{i,k}] - \hat{Y} \right| \leq \frac{\epsilon}{4}, \ k = 1, 2, ..., k_i.$$

References

[1] Barlow, R.E., F. Proschan. 1975. *Statistical Theory of Reliability and Life Testing*. Holt, Reinehart and Winston, New York.

[2] Bazaraa, M.S., H.D. Sherali, C.M. Shetty. 1993. *Nonlinear Programming*, 2nd Ed. John Wiley and Sons, New York.

[3] Çinlar, E. 1975. *Introduction to Stochastic Processes.* Prentice-Hall, Englewood Cliffs.

[4] Duenyas, I. 1995. Single facility due date setting with multiple customer classes. *Management Sci.* **41**(4) 608-619.

[5] —, W.J. Hopp. 1995. Quoting customer lead times. *Management Sci.* **41**(1) 43-57.

[6] Elhafsi, M. 2000. An operational decision model for lead-time and price quotation in congested manufacturing systems. *European J. Oper. Res.* **126** 355-370.

[7] —, E. Rolland. 1999. Negotiating price delivery date in a stochastic manufacturing environment. *IIE Transactions* **31**(3) 255-270.

[8] Easton, E.F., D.R. Moodie. 1999. Pricing and lead time decisions for make-to-order firms with contingent orders. *European J. Oper. Res.* **116**(2) 305-318.

[9] Fox, B.L., P. Glasserman. 1991. Estimating derivatives via Poisson's equation. *Probability in the Engineering and Informational Sciences* **5** 415-428.

[10] Fu, M. 1994. Optimization via simulation: A review. *Annals of Oper. Res.* **53** 199-247.

[11] —, J.-Q. Hu. 1997. *Conditional Monte Carlo.* Kluwer Academic Publishers, Boston.

[12] Glasserman, P., Y. Wang. 1998. Leadtime-inventory trade-offs in assembly-to-order systems. *Oper. Res.* **46**(6) 858-871.

[13] Hendry, L.C., B.G. Kingsman, P. Cheung. 1998. The effect of workload control (WLC) on performance in make-to-order companies. *J. Oper. Management* **16**(1) 63-75.

[14] Howard, R.A. 1960. *Dynamic Programming and Markov Processes.* John Wiley and Sons, New York.

[15] Johansen, S.G. 1991. Optimal prices of a job shop with a single work station: a discrete time model. *International J. Production Econom.* **23** 129-137.

[16] —. 1994. Optimal prices of an $M/G/1$ jobshop. *Oper. Res.* **42**(4) 765-774.

[17] —. 1996. Transfer pricing of a service department facing random demand. *International J. Production Econom.* **46-47** 351-358.

[18] — , C. Larsen. 2001. Computation of a near-optimal service policy for a single-server queue with homogeneous jobs. *European J. Oper. Res.* **134**(3) 648-663.

[19] Kingsman, B.G., A.A. de Souza. 1997. A knowledge-based decision support system for cost estimation and pricing decisions in versatile manufacturing companies. *International J. Production Econom.* **53**(2) 119-139.

[20] Larsen, C. 1998. Investigating sensitivity and the impact of information on pricing decisions in an M/M/1/∞ queueing model. *International J. Production Econom.* **56-57** 365-377.

[21] Li, L., Y.S. Lee. 1994. Pricing and delivery-time performance in a competitive environment. *Management Sci.* **40**(5) 633-646.

[22] Low, D.W. 1974. Optimal dynamic pricing policies for an $M/M/s$ queue. *Oper. Res.* **22** 545-561.

[23] Miller, B.L. 1981. Countable-state average-cost regenerative stopping problems. *J. Appl. Prob.* **18** 361-377.

[24] —, A.G. Buckman. 1987. Cost allocation and opportunity costs. *Management Sci.* **33**(5) 626-639.

[25] Polak, E. 1971. *Computational Methods in Optimization.* Academic Press, New York.

[26] Press, W.H., et al. 1989. *Numerical Recipes in Pascal.* Cambridge University Press, Cambridge.

[27] Silver, E.A., D.F. Pyke, R. Peterson. 1998. *Inventory Management and Production Planning and Scheduling*, 3rd Ed. John Wiley and Sons, New York.

[28] Stidham, S. 1985. Optimal control of admission to a queueing system. *IEEE Trans. Auto. Control* **AC-30** 705-713.

[29] —, R.R. Weber, 1989. Monotonic and insensitive optimal policies for control of queues with undiscounted costs. *Oper. Res.* **37**(4) 611-625.

[30] Tijms, H.C. 1994. *Stochastic Models.* John Wiley and Sons, Chichester.

[31] Wijngaard, J., S. Stidham. 1986. Forward recursion for Markov decision processes with skip-free-to-the-right transitions, part I: Theory and algorithms. *Math. Oper. Res.* **11**(2) 295-308.