Computers & Industrial Engineering xxx (2009) xxx–xxx

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Learning and adaptation of a policy for dynamic order acceptance in make-to-order manufacturing ☆

Facundo Arredondo, Ernesto Martinez *

INGAR(CONICET-UTN), Avellaneda 3657, Santa Fe, S3002 GJC, Argentina

ARTICLE INFO

Article history:
Received 20 October 2008
Received in revised form 11 May 2009
Accepted 21 August 2009
Available online xxxx

Keywords:
Order acceptance
Demand management
Reinforcement learning
Make-to-order manufacturing
Revenue management
Order similarity

ABSTRACT

Order acceptance under uncertainty is a critical decision-making problem at the interface between customer relationship management and production planning of order-driven manufacturing systems. In this work, a novel approach for simulation-based development and on-line adaptation of a policy for dynamic order acceptance under uncertainty in make-to-order manufacturing using average-reward reinforcement learning is proposed. Locally weighted regression is used to generalize the *gain* value of accepting or rejecting *similar* orders regarding attributes such as product mix, price, size and due date. The order acceptance policy is learned by classifying an arriving order as belonging either to the acceptance set or to the rejection set. For exploitation, only orders in the acceptance set must be chosen for shop-floor scheduling. For exploration some orders from the rejection set are also considered as candidates for acceptance. Comparisons made with different order acceptance heuristics highlight the effectiveness of the proposed ARLOA algorithm to maximize the average revenue obtained per unit cost of installed capacity whilst quickly responding to unknown variations in order arrival rates and attributes.

1. Introduction

With the advent of business-to-business electronic commerce and the trend towards client–server relationships in inter-enterprise integration, the value chain has been fragmented both horizontally and vertically. As a result, companies of all sizes and industrial sectors are increasingly shifting towards make-to-order (MTO) production systems (Calosso, Cantamessa, & Gualano, 2004; Calosso, Cantamessa, Vu, & Villa, 2003; Jalora, 2006). In order to face the challenge of MTO manufacturing, order management systems need to account for agility and responsiveness by resorting to decision support tools that integrate selective order negotiation with production planning and scheduling. Aimed at targeting very specific market niches, successful MTO systems should carefully choose which orders are worth accepting or negotiating, and which orders are not from the viewpoint of opportunity costs and revenue management (Defregger & Kuhn, 2007; Quante, Meyer, & Fleischmann, in press). Order-driven production systems increasingly focus on selling customized products and profits generated are thus strongly order-specific. Usually, a MTO company may need to satisfy several customer segments while facing an uncertain demand made up of different types of customers which are willing to accept different product prices.

The success of an MTO system is heavily dependent on the selectivity of an order acceptance policy that seeks to maximize the average revenue per unit cost of requested capacity when demand exceeds capacity. It is noteworthy that the rejection of an order may have strategic repercussions for future customer relations, and may drastically change order arrival rates and patterns. In Fig. 1, the hierarchy of functions involved in an order management system is shown. The decision an MTO system has to make for an incoming order is whether to accept, negotiate or reject it depending on the available capacity, the profit contribution margin of the order and expected arrival times for future orders. A negotiation process of an order is only worth pursuing if accepting it may contribute significantly to generate more revenues per installed capacity and market niche development. Resorting to an order acceptance policy is thus a key decision-making problem at the interface between marketing/sales functions and production planning (Barut & Shridaran, 2004, 2005; Zorzini, Corti, & Pozzetti, 2008).

Unlike the make-to-stock manufacturing mode which holds finished products in stock as a buffer against demand variability, MTO production systems must hold production capacity and work-in-process inventories to accept only orders of the most profitable type. The main issue is thus how to selectively accept/reject/negotiate orders in order to maximize cumulative profit gains. In the absence of a more elaborated policy, orders are accepted on a 'first-come-first-serve' basis. If the available capacity is not enough to process an arriving order on time, the order is rejected. Always

☆ This manuscript was processed by Area Editor Mohamed Y. Jaber, PhD.

* Corresponding author. Tel.: +54 342 4534451; fax: +54 342 4553439.
E-mail address: ecmarti@santafe-conicet.gov.ar (E. Martinez).

0360-8352/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cie.2009.08.005

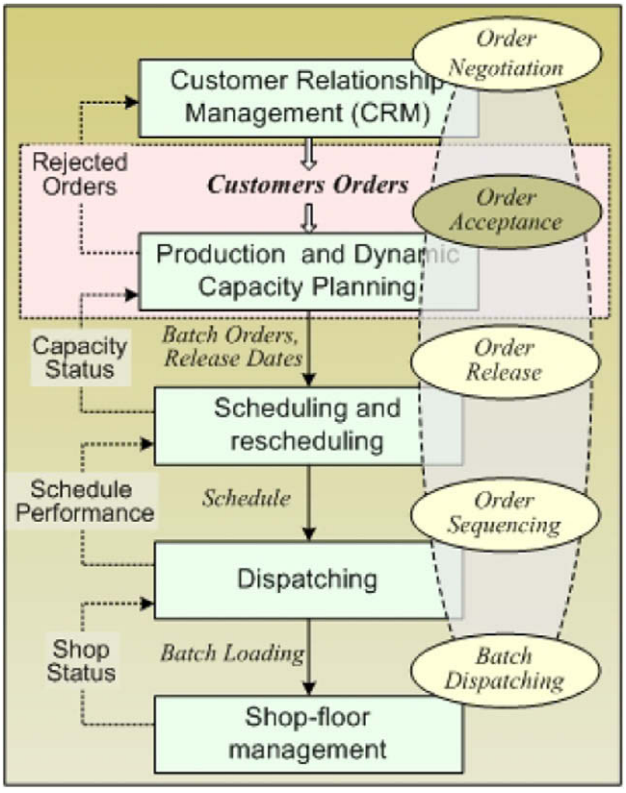

Fig. 1. Hierarchy of functions in an intelligent system for order management.

accepting an order when there exists capacity available myopically commit production capacity so that accepting more profitable orders in the near future is no longer feasible. It is an interesting facet of the order acceptance policies to take opportunity costs into account (Defregger & Kuhn, 2007; Mainegra Hing, van Harten, & Schuur, 2007; Quante et al., in press; Wouters, 1997). How to find a good trade-off between long-term opportunity costs and immediate yield in case of order acceptance under uncertainty is a central problem when demand exceeds capacity. It is thus critical to save plant capacity for future high-valued orders based on cumulative profits obtained from accepting and rejecting "similar" orders.

Despite a clear early concern about workload control (Wight, 1970), order acceptance as problem in itself has received limited attention in the literature until recently. Most papers have focused on alternative methods for releasing jobs to the shop-floor. Philipoom and Fry (1992) studied three order acceptance strategies and found that selective acceptance of orders can yield dramatic improvement in the case of capacity-constrained MTO manufacturing systems. Ten Kate (1994) also found that in severe conditions with short lead times and high utilization rate, order acceptance integrated with scheduling function performs better than operations where these functions are not tightly integrated. Wester, Wijngaard, and Zijm (1992) compared three basic order acceptance approaches and concluded that under heavy workload the one based on detailed scheduling of accepted orders outperforms other aggregate or myopic (e.g. FCFS) approaches. Wang et al. (1994) proposed a multicriteria OA decision tool in which the OA decision rule is based on a prioritization of outstanding orders by a pairwise comparison using a neural network model for ranking. Orders are accepted following the priority ranking if capacity is available, but opportunity costs are not an explicit issue under consideration.

The problem of accepting orders from the point of view of capacity loading decisions has been studied in multipurpose batch process industries by Raaymakers (1999). More recently, Raaymakers, Bertrand, and Fransoo (2000a, 2000b) studied the performance of workload control rules for order acceptance in batch chemical manufacturing. The research of Ivanescu, Fransoo, and Bertrand (2002) was built on these works by investigating order acceptance when processing times are uncertain. Enns (2000) and Enns and Costa (2002) evaluate the input control at the shop-floor based on aggregate workload measures. Using simulation, Nandi and Rogers (2003, 2004) present a make-to-order manufacturing system under a control policy involving an order acceptance/rejection component. More recently, Moreira (2005) has focused on the study of a job-shop as a multiple decision-making problem, where the acceptance/rejection decision is somehow taken into consideration. Calosso et al. (2003) discussed in detail the structure for a standardized negotiation process in electronic commerce for MTO systems. Ebben, Hans, and Olde Weghuis (2005) use a simulation model of a generic job-shop to compare their sophisticated workload control approach with straightforward methods. Uncertainty in order acceptance has started to receive attention just recently. Ivanescu (2004) continued Raaymakers' work by considering the effect of uncertainty, including stochastic order arrivals and processing times.

An important step in considering opportunity costs in addressing order acceptance was made using dynamic programming models. A single server system in continuous time in which opportunity costs play a role was studied by Nawijn (1985). A decision has to be made between starting a new service for an arriving order or rejecting an arriving order depending on its expected processing time. Orders that arrive while the server is busy are lost. A dynamic programming model allows taking into account the opportunity costs in a natural way (Herbots, Herroelen, & Leus, 2007). From this viewpoint, the optimal policy may be formulated as follows: an incoming order is made eligible for acceptance only if its immediate reward per unit of capacity is greater than the average reward. Although it is not a straightforward operational criterion since the average reward is not known beforehand, and it is difficult to estimate analytically, it helps showing that acceptance/rejection of an order only depends on its revenue per unit cost of resources demanded. The major practical problem with this dynamic programming (or optimal control) formulation is that order acceptance is plagued with uncertainty in order attributes, arrival times, processing time and costs (Mainegra Hing et al., 2007). To solve the order acceptance problem in the face of uncertainty reinforcement learning (RL) (Sutton & Barto, 1998) is a promising avenue.

Simulation-based development of order acceptance policies in a production or service environment using RL techniques was first proposed by Snoek (2000) based on a neuro-genetic architecture for acceptance/rejection decisions in a job-shop environment. It was shown that a learning-based approach performs better than two simple heuristics for order acceptance in a changing environment. In Mainegra Hing, van Harten, and Schuur (2001) an RL policy is shown to converge to the optimal policy for a simple OA case with a single server with at most one job in execution. More elaborated applications of RL to the problem of order acceptance have been discussed in Mainegra Hing (2006) and Mainegra Hing et al. (2007). However, there are important drawbacks in the above literature motivating the approach presented in this paper which are worth discussing. Firstly, order types are assumed to be known *a priori*. This is a very weak point for MTO systems where product mix, sizes and revenue of an order are attributes varying over of a continuum and also part of the problem uncertainty to be addressed. Secondly, in the works of Mainegra Hing et al. (2007) and Herbots et al. (2007), the state of processing resources are represented in an explicit way which is unduly complicated for order acceptance while imposing high computational costs without necessity. Finally, the use of neural networks for learning and

Please cite this article in press as: Arredondo, F., & Martinez, E. Learning and adaptation of a policy for dynamic order acceptance in make-to-order manufacturing. *Computers & Industrial Engineering* (2009), doi:10.1016/j.cie.2009.08.005

generalization is quite risky and impractical for on-line adaptation of the order acceptance policy.

2. Intelligent order acceptance

As shown in Fig. 1, order acceptance (OA) is a critical decision-making problem at the interface between customer relationship management and production planning of order-driven manufacturing systems. To solve this problem, the key issue is order selectivity to get the maximum profit by capacity management. As a guideline for selectivity, it is proposed here that when demand exceeds production capacity the logic for OA decision-making should be based on the following simple rule: *only accept orders that help increasing the short-term average revenue obtained per unit cost of installed capacity*. This average will be referred to as ρ hereafter. Using this guideline, credit-assignment in the sequence of acceptance/rejection decisions should be made by rewarding acceptance or rejection of an order based on its marginal contribution to the average revenue ρ. Accordingly, any order whose revenue per unit cost of capacity requested r will increase ρ must be accepted, whereas orders whose r will lower ρ must be rejected. The optimal policy for order acceptance is thus defined as the one that maximizes the average-reward obtained for all decisions taken over time. A negative reward must be given whenever an order that may increase ρ needs to be rejected since it cannot be inserted in the shop-floor schedule. Constantly seeking to maximize the short-term average revenue ρ is a distinctive goal for order acceptance which allows a precise definition of the optimal policy. Any deviation from the optimal policy will inevitably lower ρ, hence it is a sub-optimal policy.

Conceptually, the optimal OA policy can be understood as having complete knowledge about a minimum threshold $b^*(t)$ for the revenue per unit cost of capacity requested r of an order, such that only those orders that comply with $r \geqslant b^*(t)$ should be accepted. The main problem is that this threshold $b^*(t)$ is typically unknown and time-varying since it depends heavily on attributes (price, size and product mix) and arrival rates of the different order types. As a result, OA heuristics such as those proposed by Mainegra Hing et al. (2007) which are based on a fixed threshold b will be necessarily sub-optimal. On one hand, whenever the fixed threshold is set too low, i.e. $b < b^*(t)$, the MTO system accept some orders that lower the average revenue whereas some orders of the most profitable type are necessarily rejected since they cannot be inserted in the shop-floor schedule. Conversely, if the acceptance threshold is chosen such that $b > b^*(t)$, some valuable orders are unduly rejected which give rises to a lower utilization of installed capacity and opportunity costs. Due to the many sources of uncertainty involved and dynamics of the optimal threshold $b^*(t)$, the OA policy must be learned on-line from reinforcements while constantly seeking to increase the short-term average ρ.

For practical reasons, it is assumed hereafter that order acceptance is carried out using decision epochs (see Section 3 below) in such a way that an order_list of outstanding orders is considered at each epoch. When a decision has been made for all orders in the current order_list, a transition to the next decision epoch takes place and a new list of orders is considered. Given an order_list, an important issue is how orders should be ranked so that acceptance/rejection decisions can be taken one at a time to assess individually the feasibility of inserting each order into the current schedule. Intuitively, it can be said that the order_list should be ranked so that at the top of the list are those orders whose acceptance decision is definitively more valuable than order rejection. At the bottom of the order_list are located those orders whose rejection is definitively more convenient for increasing the short-term average revenue ρ. For this purpose, the concept of a *gain* or *value* $\mathscr{R}$ of taking either the acceptance decision $\mathscr{R}(o, acc)$, or the rejection decision $\mathscr{R}(o, rej)$ for an order o, is proposed. The value or utility $\mathscr{R}$ is thus defined as the immediate reward r obtained following order acceptance or rejection, plus the maximum average reward that can be obtained by following an optimal OA policy from then onwards. As explained in the next section, these gain values $\mathscr{R}(o, acc)$ and $\mathscr{R}(o, rej)$ are obtained using a learning rule that resorts to evaluative feedback of decisions taken in a sequence of decision epochs.

Consider a given candidate order o is must be analyzed for acceptance or rejection based on its contributing value to maximizing the average reward per time step using the corresponding gain values. Assuming the gain values for acceptance $\mathscr{R}(o, acc)$ and rejection $\mathscr{R}(o, rej)$ are known beforehand, the decision-making logic proposed in Fig. 2 is rather straightforward. If $\mathscr{R}(o, acc) > \mathscr{R}(o, rej)$, insertion of the order "o" into the current production plans is then attempted. If order insertion for shop-floor processing is feasible, the order is then accepted; otherwise, the due date may be, if possible, negotiated with the client to avoid rejecting it. Alternatively, whenever gain values for the order o are such that $\mathscr{R}(o, acc) < \mathscr{R}(o, rej)$, negotiation of order attributes is required in order to increase the order acceptance value $\mathscr{R}(o, acc)$ so as to exceed $\mathscr{R}(o, rej)$ by either increasing order price, changing order size or product mix, or combination thereof. Otherwise, the order o must be rejected.

To better understand the rationale of the decision-making logic in Fig. 2, it is mandatory to understand the meaning of $\mathscr{R}$-values assigned to a given order as heavily dependent on the environment in which the MTO system operates. For example, the values for acceptance and rejection of an order type depend on the arrival rates of order types which are more valuable than it. For the sake of clarity, let's assume that in the current arrival pattern an order o should be rejected according to the logic in Fig. 2. Imagine the arrival rate of more valuable orders is decreased. Accordingly, the value of acceptance $\mathscr{R}(o, acc)$ for this type of less profitable orders becomes greater than the value for rejection $\mathscr{R}(o, rej)$. As a result, similar orders should be accepted to increase ρ. It is worth noting that $\mathscr{R}$-values are heavily dependent on its attributes such as mix of products, price and size. Thus, the acceptance condition $\mathscr{R}(o, acc) > \mathscr{R}(o, rej)$

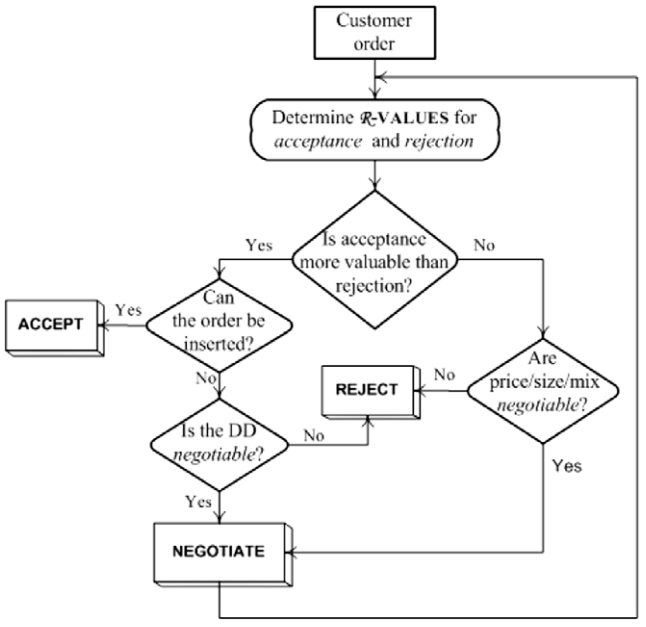

Fig. 2. Order acceptance logic based on $\mathscr{R}$-values for order acceptance and rejection.

Please cite this article in press as: Arredondo, F., & Martinez, E. Learning and adaptation of a policy for dynamic order acceptance in make-to-order manufacturing. *Computers & Industrial Engineering* (2009), doi:10.1016/j.cie.2009.08.005

is a simple, yet powerful guideline for negotiating order attributes. Should the corresponding $\mathscr{R}$-values for acceptance and rejection be perfectly known *a priori* is rather straightforward to decide always optimally. Accordingly, only orders which are contributing the most to maximize the average revenue are accepted, whereas the rest are rejected.

For an over-demanded MTO production system, the corresponding values for acceptance and rejection decisions of an order are independent on the way resource usage is represented in production plans. This is a very important remark which challenges the need for expliciting representing plans and schedules to solve the order acceptance problem as proposed in Mainegra Hing (2006), Mainegra Hing et al. (2007) and Herbots et al. (2007). If an arriving order whose acceptance value is greater than the corresponding value for rejection, but the order cannot be accepted because it is not feasible to insert the order into the current plan or schedule, the only alternative is to negotiate its due date. If this negotiation is not an option, the order must be rejected despite the order is capable of increasing the short-term average revenue, but due to orders previously accepted the available capacity is insufficient.

If the $\mathscr{R}$-values for acceptance and rejection decisions in Fig. 2 were known exactly for a given order, the only information required is whether the order can be inserted or not in the current production plan. Thus, explicitly considering profiles for planned resource usage or any other scheduling or shop-floor details in the decision-making logic for order acceptance are not needed. The main obstacle to the implementation of the order acceptance logic shown in Fig. 2 is knowing the values $\mathscr{R}(o, acc)$ and $\mathscr{R}(o, rej)$ for acceptance or rejection of an arriving order. Due to the many sources of uncertainty involved, reinforcement learning from simulated experience is the alternative of choice for value estimation and generalization based on order similarity. Furthermore, $\mathscr{R}(o, acc)$ and $\mathscr{R}(o, rej)$ can be updated on-line using real experience to account for time-varying order arrival rates and new order types. It is important to highlight the complete equivalence between checking the acceptance condition "$\mathscr{R}(o, acc) > \mathscr{R}(o, rej)$" with resorting the optimal threshold b^*, as it is discussed below for the case study.

3. Learning and adaptation

3.1. Order acceptance decision process

Let's consider an MTO system which is subjected to an uncertain demand that should be answered by acceptance, rejection or negotiation of each arriving order. Orders arrive at the system following some sort of stochastic pattern or regularity, e.g. Poisson process with arrival rate λ. Each order is characterized by a number of attributes such as mix of different products, due date, size (volume) and, if accepted, it generates a revenue. Accordingly, once the decision to accept or reject an order has been made, an immediate reward r is obtained. The MTO system has a limited processing whose scheduled usage state changes every time an order is accepted. A rejection decision will not change the schedule for using the installed capacity but may alter arrival patterns for some order types. When the $\mathscr{R}(o, acc) < \mathscr{R}(o, rej)$ (the order is deemed not valuable enough) or the current production plan does not allow to insert an order one option is rejection. Alternatively, if possible the attributes of an order can be negotiated so that the value for acceptance $\mathscr{R}(o, acc)$ is increased over $\mathscr{R}(o, rej)$. If $\mathscr{R}(o, acc) > \mathscr{R}(o, rej)$, negotiation should pursue a delayed due date to allow the insertion of the order into the current schedule. Orders in the negotiation process are accepted as soon as an agreement is reached on their attributes and insertion in the shop-floor production schedule is feasible. Although order arrivals occur continuously over time, orders are only evaluated at discrete time moments t. At each decision epoch, it is assumed that all orders whose arrival occur after the previous decision epoch are accumulated in the order_list. These decision epochs, labeled by $t, t+1, t+2, \ldots$, correspond to fixed time windows, e.g. 1 or 2 days, for which a list of all arriving orders (order_list) in such period of time is evaluated and acceptance/rejection/negotiation decisions are taken. As a result of decisions taken a sequence of rewards is generated. Fig. 3 summarized the essence of the OA problem in the well-known style of stage-wise dynamic programming.

At each decision epoch, the OA-agent must decide which subset of the orders in the order_list should be accepted or rejected. This means, in principle, that each possible subset of orders may be analyzed to assess if those orders can be inserted in the current production schedule. However, this approach to develop an OA policy does not allow learning order values based on its similarity with previously accepted orders. Thus, instead of focusing on all possible subsets of orders that can be accepted or rejected at once, in this work decisions are taken sequentially whilst the order_list is being processed. The top order in the order_list is considered for acceptance/rejection/negotiation following the logic in Fig. 2, the decision is taken and again the new top order is considered until the list is empty. Accordingly, decisions are taken one order at a time which allows describing order acceptance as a dynamic program in the framework of a semi-markov decision process (SMDP). For a SMDP, decision moments are not restricted to discrete time epochs (like in MDPs) but correspond to any time at which the system enters a new state (Das, Gosavi, Mahavedan, & Marchalleck, 1999). Thus, the MTO system state may change several times between two decision epochs due to capacity assignments. As a result, there are two types of state transitions in the order_list and resource usage profiles (see Fig. 3). *Immediate transitions* as long as acceptance/rejection decisions are taken and *timed transitions* which happen as the discrete time t advances to the next decision epoch when a new order_list is available for processing and shop-floor resource commitments may also change as some orders are being completed.

Order acceptance is a sequential decision-making stochastic process characterized by five elements: decision epochs, states, actions, transition probabilities and rewards. The OA-agent (decision-maker) controls the path of the stochastic process comprising of order arrivals and shop-floor admissions. In fact, at certain points in time in the path, the OA-agent intervenes and takes decisions which alter the course of the future path. These points are called decision epochs and the decisions are called actions. At each decision epoch, the MTO system occupies a decision-making state in which each order present in the order_list is rejected, chosen for service or negotiated. The MTO system state is partly described by an array whose rows are vectors (orders) with order's attributes (e.g. size, price, due date) and its *insertability* as their entries. As a

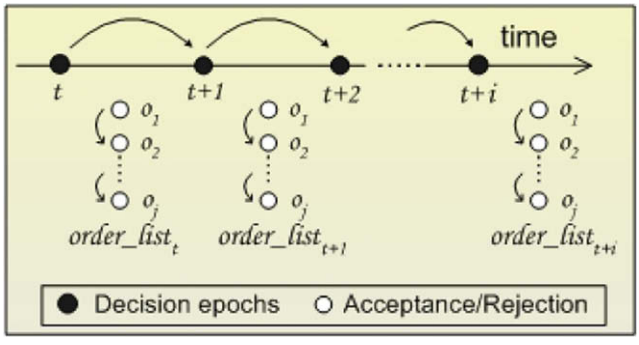

Fig. 3. Sequential decision process in order acceptance.

Please cite this article in press as: Arredondo, F., & Martinez, E. Learning and adaptation of a policy for dynamic order acceptance in make-to-order manufacturing. *Computers & Industrial Engineering* (2009), doi:10.1016/j.cie.2009.08.005

result of taking an action in a state, the decision-maker receives a reward r (which may be positive or negative) and the system goes to the next state with a certain probability which is called the transition probability. A decision rule is a function for selecting an action in each state, while a policy is a collection of such decision rules over the state-space. A policy specifies which action to perform at each state of the system. Implementing a policy generates a sequence of rewards. The SMDP problem is to choose a policy that maximize a function of this reward sequence (optimality criterion), typically an average reward per time step.

In the reinforcement learning literature (Sutton & Barto, 1998), there exist two well-known optimality criteria: the discounted expected cumulative reward and the average-reward criteria. Most works in reinforcement learning are based on expected cumulative reward, where long-term rewards are discounted based on the delay in their occurrence. Discounting future rewards makes perfect sense in some applications, e.g. those dealing with net present value of decisions, where the distant future is indeed less important than the near future (Singh, 1994). However, in the OA problem, when demand exceeds capacity, the goal is to maximize the average-reward obtained per time unit by properly choosing which orders are more profitable in terms of revenue per capacity requested. In this case, all time periods are equally important. For developing the OA policy in MTO systems, the average-reward criterion is used here. The average reward or *gain* of a stationary decision-making policy π, starting at any state system i and continuing with policy π, is defined as follows:

$$\rho^{\pi}(i) = \lim_{N\to\infty} \frac{1}{N} \sum_{i}^{S(\pi)} \left\{ \sum_{t=1}^{N} r(s_t, a_t) \right\} \tag{1}$$

where $S(\pi)$ is the set of system states s_t which can be visited when the policy π is used to choose action a_t which generates the reward $r(s_t, a_t)$; N refers to the number of times each system state is visited when policy π is used to choose actions. The optimal policy can then be characterized simply as the one which provides the maximum average reward ρ^*. As an example, consider the special case when π is a heuristic rule where only orders having revenue per unit of capacity requested r above a threshold b are accepted. The $S(\pi)$ will then correspond to all possible MTO system states (e.g. resource usage profiles) resulting from the application of this rule and ρ^{π} will correspond to the mean revenue which can be obtained using this OA heuristic infinitively.

3.2. Average-reward reinforcement learning

The computational complexity of SMDP algorithms is prohibitive for industrial applications and can grow intractably large with the size of the problem (number of states and action). Furthermore, these model-based techniques require knowing for each action the one step transition probabilities matrix and corresponding rewards, a completely unrealistic assumption bearing in mind the many sources of uncertainty (arrival times, order types, processing times, etc.) involved in the development of a decision-making policy for order acceptance. To face uncertainty, in this work a simulation-based stochastic approximation framework called reinforcement learning (RL) for computing near-optimal policies for SMDPs is used. RL is primarily concerned with how to learn a near-optimal policy from successive interactions with an uncertain environment (Sutton & Barto, 1998). The OA-agent, equipped with a learning rule, uses the outcomes of ongoing interactions with its (actual or simulated) environment to update the $\mathscr{R}$-values for acceptance and rejection decisions so as to maximize ρ over time.

Fig. 4 summarizes the interaction cycle between the OA-agent and its environment in an MTO system. At each decision epoch t, the OA-agent first receives a perception of the environment's state in the form of a list of orders (along with their attributes) which have arrived during a given period of time (1), e.g. day and information about available capacity at the shop-floor (2). Orders that made up the current order list (order_list) are ranked based on their values for acceptance $\mathscr{R}(o, acc)$. The OA-agent must decide and perform an action (3) for each order in the order_list. The action set includes to accept or to reject each order in the order_list. Order selection is made according to the current decision policy developed by the agent. If the order is accepted (4) the commitment of production resources must change. In case the order is rejected (5), capacity available is not affected, but arrival rates of similar orders may decrease. As a result, the environment (customers and production system) makes a transition to a new state and a reinforcement, or reward signal, is given to the OA-agent. The new state includes all orders already belonging to the order_list minus the order that was just accepted or rejected, and the insertability of each order in the current production schedule. The reward signal is received by the OA-agent and through its learning rule is used to update its knowledge base. Consequently the decision-making policy is updated (6). Whilst processing the order list, the OA-agent implements a mapping from states to probabilities of accepting or rejecting a given order bearing in mind the exploration–exploitation trade-off. This mapping or rule is called the OA *policy* π. In reinforcement learning, the agent changes this policy as a result of interactions where the agent takes an action and receives a reward, which is the only hint it has to modify its decision-making policy.

As the decision-making objective is to maximize the cumulative profit gain, the learning process is based on relating revenue to order attributes using a value function $\mathscr{R}(s, a)$, where s is the system state and a stands for the finite set of possible actions in s. When a given action a is chosen at state s, $\mathscr{R}$ corresponds to the average reward that can be gained by choosing action a and acting optimally thereafter. Should the values of $\mathscr{R}(s, a)$ be known *a priori*, the optimal policy is simply to chose accept/reject orders based on which action has the highest $\mathscr{R}$. Unfortunately, the value function $\mathscr{R}$ is unknown *a priori* and can only be estimated by evaluative feedback along the sequence of actions (acceptance/rejection) and credit-assignments to actions taken in the sequel using the actual rewards obtained. Furthermore, even when the action set is finite, MTO system states are defined by a continuum of variables defining product mix, size, volume, etc., which poses a difficult problem

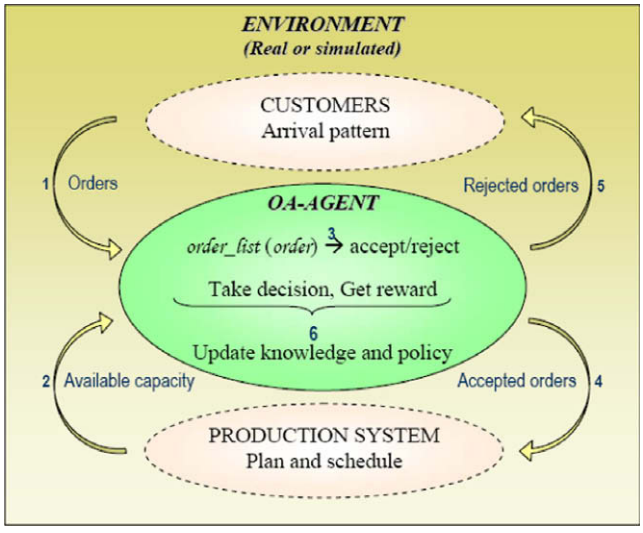

Fig. 4. Interaction cycle of the OA-agent with customers and production scheduling.

Please cite this article in press as: Arredondo, F., & Martinez, E. Learning and adaptation of a policy for dynamic order acceptance in make-to-order manufacturing. *Computers & Industrial Engineering* (2009), doi:10.1016/j.cie.2009.08.005

for generalizing the $\mathscr{R}$-values for acceptance or rejection of orders to new orders that can bear some similarity with orders previously accepted, but are not identical.

To learn $\mathscr{R}$-values from (simulated or actual) experience the basic *$\mathscr{R}$-learning* algorithm (Schwartz, 1993) is adapted for order acceptance under uncertainty, where the return associate to any OA policy is the average reward per time step received by the OA-agent. This algorithm aims to compute an optimal policy that yield the highest average reward. Initially the acceptance $\mathscr{R}$-values for all order types are set to an optimistic high value. At each state in the sequence, the learning agent chooses the order acceptance action that has the highest $\mathscr{R}$-value (accept/reject), except that sometimes it chooses an exploratory, non-optimal action. $\mathscr{R}$-values are adjusted after each action, based on the following learning rule:

$$\mathscr{R}_{new}(s,a) \leftarrow \mathscr{R}_{old}(s,a) + \alpha\left[r - \rho + \max_{a'} \mathscr{R}_{old}(s',a') - \mathscr{R}_{old}(s,a)\right] \quad (2)$$

where r is the reward resulting of taking the action a at the state s and $0 < \alpha < 1$ is the learning rate. This rule differs from the Q-learning algorithm (the most popular algorithms used in RL, see Sutton & Barto, 1998 for details) only in that the average reward ρ is subtracted from the immediate reward r, and by the fact that discount factor γ is not required. Using revenues obtained from recently accepted orders, the average reward ρ for order acceptance in the learning rule (2) is increasingly updated using the cumulative revenue obtained per installed capacity over $\mathscr{H}$ previous decision epochs as follows:

$$\rho = \frac{\sum_{t-\mathscr{H}}^{t-1} revenue}{\mathscr{H} \times cap\_\max} \quad (3)$$

where cap_max is the installed capacity available per time step. As no revenue is get from rejected orders, $\rho = 0$ when the learning rule (2) is used to update $\mathscr{R}(o, rej)$ for a rejection decision. In Fig. 5, the basic *$\mathscr{R}$-learning* algorithm for order acceptance is described.

The way acceptance/rejection decisions are given rewards or penalties is the only evaluative feedback from the environment to the learning OA-agent. In the present work, whenever an order is accepted, the reward r is exactly the same revenue obtained per unit cost of processing capacity requested (p_i). However, for rejected orders the reward is defined as follows:

$$r_i = \begin{cases} -p_i & \text{if } \mathscr{R}(o_i, acc) \geq \mathscr{R}(o_i, rej) \\ 0 & \text{if } \mathscr{R}(o_i, acc) < \mathscr{R}(o_i, rej) \end{cases} \quad (4)$$

where $\mathscr{R}(o, acc)$ and $\mathscr{R}(o, rej)$ are current $\mathscr{R}$-values corresponding to acceptance and rejection decisions, respectively. It is worth noting that based on the estimated $\mathscr{R}$-values for an order, a penalty (negative reward) is given to the decision of rejecting an order when it should be accepted, that is, whenever $\mathscr{R}(o, acc) \geqslant \mathscr{R}(o, rej)$. A word of caution is in order about the basic algorithm for order acceptance: learning assumes that there exist different order types. Thus, $\mathscr{R}$-values actually correspond to order types rather individual orders. If an arriving order is different from any previously accepted orders, a new order type must be created for which the corresponding $\mathscr{R}$-values must be learned.

Set ρ and $\mathscr{R}(s,a)$, for all s, a arbitrarily
Repeat forever
$s \leftarrow$ current state
Choose action $a \in \mathscr{A}(s)$
Take action a, receive s' and r values

$$\mathscr{R}_{new}(s,a) \leftarrow \mathscr{R}_{old}(s,a) + \alpha \times \left[r - \rho + \max_{a'} \mathscr{R}_{old}(s',a') - \mathscr{R}_{old}(s,a)\right]$$

Each time an order is accepted, update the average reward using:

$$\rho = \frac{\sum_{t-\mathscr{H}}^{t-1} revenue}{\mathscr{H} \times cap\_\max}$$

Fig. 5. Basic $\mathscr{R}$-learning algorithm for OA.

3.3. Value generalization and order similarity

For MTO systems with a small number of order types, estimates of the $\mathscr{R}$-values can be represented as a table. However, as the number of order types increases the table of $\mathscr{R}$-values scales up significantly in size; updating information accurately in such a large table may be a significant problem for fast learning and adaptation. Often orders have attributes that vary over a continuum. However, there exist clusters of similar orders that favor inductive learning. To generalize $\mathscr{R}$-values based on order similarity we resort to *locally weighted regression* (*LWR*) (Atkeson et al., 1997; Smart & Pack Kaelbling, 2000). *LWR* is a variation of standard regression techniques, in which training points close to the query point have more influence over fitted regression surface than those which are further away. Performing a regression over all of the training data for every order query would be extremely expensive. Therefore, *LWR* attempts to fit the training data only in a small region around the location of the query point. Data points are weighted according to a function of their distance from the query point. This function is typically a kernel function (k), with 'width' parameter known as the bandwidth (h):

$$k(d,h) = e^{-(d/h)^2} \quad (5)$$

where d is the Euclidean distance between the query point and each point in the training dataset. Since most of data points are likely to be far away from the query point, they will have little effect on $\mathscr{R}$-values estimation for an order. Typically, a minimum number of points is needed for estimating $\mathscr{R}(o,\ acc)$ and $\mathscr{R}(o,\ rej)$. With few data points, regression is not reliable and a default $\mathscr{R}$-values must be returned.

The key issue in local propagation of $\mathscr{R}$-values is order similarity. Accordingly, learning in order acceptances is based on the assumption of clustering orders in the attribute space with similar values for $\mathscr{R}(o,\ acc)$ and $\mathscr{R}(o,\ rej)$. In the *LWR* algorithm these clusters of similar orders are defined using some sort of distance norm (e.g. Euclidean) between a query point and orders in its neighborhood. Thus, order similarity relates the notion of locality of order attributes with their corresponding $\mathscr{R}(o,\ acc)$ and $\mathscr{R}(o,\ rej)$. It is noteworthy that even when orders are not segmented by type *a priori*, the proposed learning algorithm will discover the existence of such order clusters. The proposed integration of *LWR* with the basic OA *$\mathscr{R}$-learning* algorithm makes possible to propagate $\mathscr{R}$-values updates to all accepted orders in the region of influence around the query point, as defined by the kernel function in Eq. (5).

3.4. Exploitation vs. exploration

To effectively learn a policy in the face of uncertainty the OA-agent must address the dilemma of exploitation vs. exploration. This means that to discover better orders to profit from it is necessary to accept orders whose rejection seems, at first glance, more valuable than acceptance. Without purposeful exploration the agent can only exploit knowledge from accepted orders. To exploit more in the near future, the OA-agent should accept apparently less profitable orders to know more about their profitability based on order similarity. The trade-off between exploitation and exploration can be achieved in many different ways. The easiest alternative is the so-called *ε-greedy*. With probability $(1 - \varepsilon$ the decision

Please cite this article in press as: Arredondo, F., & Martinez, E. Learning and adaptation of a policy for dynamic order acceptance in make-to-order manufacturing. *Computers & Industrial Engineering* (2009), doi:10.1016/j.cie.2009.08.005

with highest value is selected whereas with probability ε, any of the alternative decisions is taken. More specifically, for learning the OA policy, the balance between exploration and exploitation is based on ranking the order_list using $\mathscr{R}(o,\ acc)$. Also, orders in the order_list are classified as belonging either to the *acceptance set* ($\mathscr{R}(o,\ acc) > \mathscr{R}(o,\ rej)$) or to the *rejection set* ($\mathscr{R}(o,\ rej) > \mathscr{R}(o,\ acc)$). For exploitation, only orders in the acceptance set must be chosen for shop-floor insertion. For exploration some orders from the rejection set are also considered as candidates for acceptance. Accordingly, with probability $(1-\varepsilon)$ the order at the top of the acceptance set is selected. If the order can be inserted into the shop-floor schedule, then it is accepted. On the other hand, with a small probability ε any of the orders in the rejection set is chosen to check the feasibility of its insertion. As usual, if the capacity available allows inserting the order, then it is accepted.

To stabilize the learning curve it is important to lessen the effect of exploration as the agent-environment interaction progresses. To this aim, the value of ε is increasingly lowered as follows (Mainegra Hing et al., 2007):

$$\varepsilon = \frac{\varepsilon_0}{1 + t/\tau_0} \tag{6}$$

where ε_0 is the initial value of ε at the beginning of the learning curve and τ_0 is a parameter which defines the progressive decay of exploration. The lower is parameter τ_0, the greater is the exploration decay rate towards only exploitation of the acceptance set whiles the order_list is being processed.

For responsiveness, it is important to reset the exploration parameter $\varepsilon \leftarrow \varepsilon_0$ as soon as shop-floor utilization or the average revenue ρ experiment a sustained decrease. In the first case, resource utilization may decrease because the arrival rate of the most profitable type has diminished and the OA policy should change to accept more orders of less profitable types. A lowering of ρ may be due to a profitability loss of some order types which demands updating the $\mathscr{R}$-values of order clusters that are being rejected with the current policy. Whatever the case, the rationale for resetting to a maximum level of exploration is that environmental changes may have changed the actual values for acceptance of some orders which are not being accepted unless exploration is done. For better updating the values of those orders, a more aggressive strategy for exploration is thus required. Finally, a special type of exploration strategy is needed to handle the arrival of orders of a new type for which estimating acceptance/rejection values is not feasible using *LWR*. In the ARLOA algorithm (see below), when an arriving order has a set of attributes such that there not exist enough data points to support a reliable estimation of its $\mathscr{R}$-values, a different kind of exploration is carried out. To guarantee that some orders of this new type are processed by the MTO system, an optimistic (high) default value is assigned to $\mathscr{R}(o,\ acc)$.

3.5. ARLOA algorithm

The overall logic of the proposed ARLOA (Average-reward Reinforcement Learning for Order Acceptance) algorithm is shown in Fig. 6. The values $\mathscr{R}(o, acc)$ and $\mathscr{R}(o, rej)$ of an incoming order o are first estimated using *LWR*, based on the similarity of its attributes with previously accepted orders. At each decision epoch, the order_list is first prioritized using order values for acceptance, namely $\mathscr{R}(o, acc)$. While there exists enough processing capacity available orders in the order_list are accepted using an OA policy that combines exploitation with exploration based on the exploration parameter ε. If the order cannot be inserted and its due date cannot be delayed, the order is rejected. If $\mathscr{R}(o, acc) < \mathscr{R}(o,\ rej)$ and order attributes are not negotiable, the order is rejected. Details about the integration of average-reward reinforcement learning algorithm with locally weighted regression (*LWR*) for value generalization are given in Fig. 6. The novel algorithm ARLOA is the key contribution of this work. ARLOA is capable of effectively handling different sources of uncertainty in the environment of a MTO system by transforming the simple guideline stated in Section 2 into a systematic procedure to answer order processing requests.

The ARLOA algorithm has been developed bearing in mind agent-based automation of the interface between customer relationship management and production planning as part a of business-to-business electronic commerce project for make-to-order supply chains. However, having access to a minimum of data from the shop-floor planning system in order to assess the insertability of a candidate order, the ARLOA algorithm can also be applied as a standlone application based only on an user interface, a standard spreadsheet and a small database application to accumulate information on previous accepted orders. Small and medium enterprises working as MTO systems can benefit sensibly from a standlone implementation of ARLOA for revenue management.

4. Case study and results

4.1. Description

To illustrate the proposed approach, order acceptance in a *single-stage process plant* adapted from Musier and Evans (1989) is used. These authors argued that this class of process structure is the typical production structure found in a broad range of batch process industries, and it is also a key building block in bottleneck resource scheduling in multi-stage production plants. Our case study is related to a manufacturing environment in which production is driven by multiple-product orders. Plant equipment items are semi-continuous extruders which process orders for four different products. Each order consists of a mix of products, has a due date and generates a revenue following acceptance. After a decision (acceptance or rejection) is taken, a reward is generated based on the order revenue and requested capacity (processing time in this case). There is a maximum regular capacity of hours per working day due to operator shift and there is not available extra capacity. It is assumed that all accepted orders must be completed without violating their corresponding due date. Working weeks are made up of 5 days.

The product mix of an arriving order is randomly generated from up to five well-differentiated order clusters or types. This fact is unknown *a priori* to the OA-agent and is part of the learning process. For the Case 1 below (initially unknown to the OA-agent) preferences by order type are as follows: $4 \prec 3 \prec 2 \prec 5 \prec 1$, based on their corresponding revenue per unit of capacity requested (p_i). For all order types, due dates are generated using a uniform distribution between 5 and 10 working days. At each decision epoch, the MTO system state is defined by the order_list prioritized by the corresponding $\mathscr{R}$-values for acceptance; the individual insertability of each order is also an entry in the system state vector. As the order_list is being processed using the ARLOA algorithm (see Fig. 6), orders in the acceptance set are accepted only if they can be inserted. This corresponds to exploiting the knowledge an OA-agent has about the $\mathscr{R}$-values for acceptance and rejection of each order in the current order_list. When exploration is carried out, orders from the rejection set are also made eligible for processing. Penalties to rejection are given based on the revenue generated per unit of capacity requested (see Eq. (4)). Orders of all types arrive following a Poisson process with mean $\lambda = 10$ order/day. Arrival rates (in percentage of λ of each order type for the base case (Case 1) are shown in Table 1.

Please cite this article in press as: Arredondo, F., & Martinez, E. Learning and adaptation of a policy for dynamic order acceptance in make-to-order manufacturing. *Computers & Industrial Engineering* (2009), doi:10.1016/j.cie.2009.08.005

1) Set epoch decision tag: t=0. Initialize all $\mathcal{R}(o,acc)$ with an optimistic initial value. Set the cumulative revenue obtained per installed capacity: ρ=0. Run the MTO system.
2) For each arriving order in [t-1, t] estimate $\mathcal{R}$-values using LWR and prioritized them in the *order_list* using $\mathcal{R}(o,acc)$.
3) While *order_list* is not empty do
   i) With probability 1-ε, select the order at the top of the *order_list* (EXPLOIT), otherwise choose a random order from the *rejection set* (EXPLORE). Set i as the selected order.
   ii) Given an order, decide to *accept*, *reject* or *negotiate* it. If $\mathcal{R}(i,acc) > \mathcal{R}(i,rej)$ and the order can be inserted, then *action* = *accept* and go to iii); if the order cannot be inserted negotiate its due date and return to i).
   If $\mathcal{R}(i,acc) < \mathcal{R}(i,rej)$ and order attributes (price, mix, size, etc.) can be negotiated, then *action* = *negotiate* and return to i). Otherwise, *action* = *reject* and go to iii).
   iii) Take the chosen action (*accept/reject*). Let r be the immediate reward received as a result of taking action a.
   iv) Calculate $\mathcal{R}(o,a)$ using:

$$\mathcal{R}_{new}(o,a) \leftarrow \mathcal{R}_{old}(o,a) + \alpha \times \left[r - \rho + \max_{a'} \mathcal{R}_{old}(o',a') - \mathcal{R}_{old}(o,a) \right]$$

   v) Update $\mathcal{R}$-value for each order j in the neighbourhood of order i:

$$\mathcal{R}_{new}(o_j,a_j) \leftarrow \mathcal{R}_{old}(o_j,a_j) + \kappa_j \times \left[\mathcal{R}_{new}(o_i,a_i) - \mathcal{R}_{old}(o_j,a_j) \right]$$

   vi) Set current state s to the new state s', $\mathcal{R}(o,a)_{old} \leftarrow \mathcal{R}(o,a)_{new}$.
   vii) If *order_list* is not empty go to step i), else go to step 4)
4) Update average reward:

$$\rho \leftarrow \frac{\sum_{t-\mathcal{H}}^{t-1} revenue}{\mathcal{H} \times cap\_max}$$

go step 2)

Fig. 6. The ARLOA algorithm.

4.2. Results

Different scenarios for the demand to the MTO system have been evaluated to compare the ARLOA algorithm with threshold heuristics of the type proposed by Mainegra Hing et al. (2007). In some scenarios, the proposed method is also compared with the simple rule "*First Come First Serve*" (FCFS) and *Q*-learning based on discounted cumulative rewards with a discount factor $\gamma = 0.99$. The initial value of the exploration parameter is set $\varepsilon_0 = 0.2$, whereas decay exploration rate is defined by choosing $\tau_0 = 10$. Learning rate is set to $\alpha = 0.5$ for both *Q*-learning and ARLOA algorithms. At each decision epoch, the short-term average ρ is calculated based on orders accepted in 10 previous epochs.

Table 1
Demand data for the base case.

	P1	P2	P3	P4
Products attributes				
Processing rate (kg/day)	30	25	25	30
Revenue ($/kg)	5	2	1	0.5

	#1	#2	#3	#5	#4
Order type attributes					
Size (kg)	15				
Composition (% dominant products)	$N \sim (60,5)$				$N \sim (22,3)$
Due date (days)	$X \sim U(5,10)$				
Arrival rates (% of λ)	10	20	30	30	10

4.2.1. Case 1: base case

In Table 1 data values for this base case are given. Orders are generated from clusters of similar orders. Each order has a dominant product and the remaining product mix is evenly distributed among the other three products. Order types #1, #2, #3 and #5 have as dominant products P1, P2, P3 and P4, respectively. For each order type, the percentage of the dominant product is randomly obtained from a normal probability distribution with mean $\mu = 60\%$ and standard deviation $\sigma = 5\%$, whereas for the other products in the order are evenly distributed. For order type #4, the mix of products is roughly balanced as they are all obtained through sampling a normal distribution with $\mu = 22\%$ and $\sigma = 3\%$. The percentage of λ corresponding to the arrival rate of each order type is given in Table 1.

In Fig. 7, the learning curve in this illustrative example is shown for the ARLOA policy using the weekly revenue obtained from accepted orders. Learning curves were obtained by averaging five independent simulation runs. A comparison is made between the ARLOA algorithm, an optimal threshold heuristic $r \geqslant 55$ ($b = 55$), the *Q*-learning rule and the first-come-first-served (FCFS) acceptance policy. As it is shown, after a learning phase, the ARLOA

Please cite this article in press as: Arredondo, F., & Martinez, E. Learning and adaptation of a policy for dynamic order acceptance in make-to-order manufacturing. *Computers & Industrial Engineering* (2009), doi:10.1016/j.cie.2009.08.005

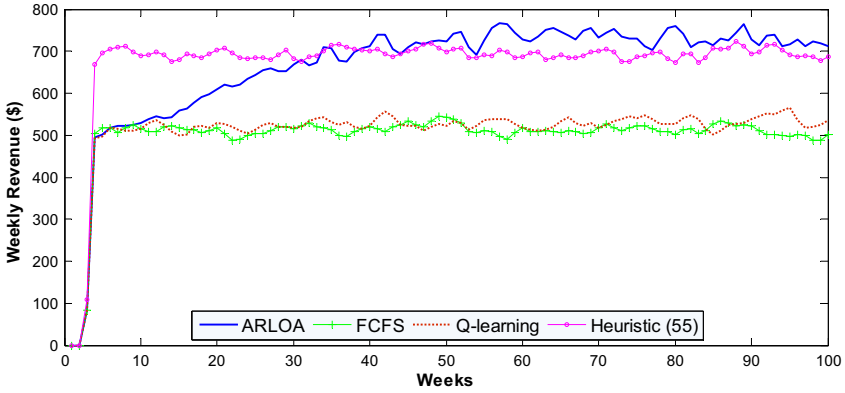

Fig. 7. Profit learning curves for Case 1. ARLOA and *Q*-learning OA-agents compared to FCFS policy and a near-optimal threshold heuristic with $b = 55$.

policy not only match the revenue get by the optimal heuristic rule but actually performs better. The sub-optimality of FCFS and the *Q*-learning rule is quite noticeable. Note that the optimal threshold value ($b = 55$) was obtained after a tedious and computationally expensive tuning of the parameter b by trial-and-error. To assess the selectivity of the ARLOA policy, an independent simulation run was made without allowing further changes to the $\mathscr{R}$-values. Results obtained for the acceptance rate of each order type are shown in Fig. 8. The ARLOA policy is highly selective since almost every type #1 order arriving to the system is accepted. Selectivity of the ARLOA policy also seeks to increase revenue generation from the installed capacity by accepting near 50% of type #5 orders which are second in the preference scale. Note that the ARLOA policy only accepts a very small number of orders from types #2, #3 and #4. It is worth mentioning that selectivity is very important for consolidating market segments in MTO systems. The optimal heuristic rule is by far less selective than ARLOA. OA policies using *Q*-learning and FCFS are not selective at all. Selectivity and responsiveness are the two most requirements for successfully implementing the hierarchy of functions in Fig. 1.

4.2.2. Case 2: adaptability

To assess adaptiveness of the policy generated by ARLOA, at time 500 days (100 working weeks) a significant reduction in the arrival rates and revenues of order types #1 and #5 are made as it is shown in Table 2. Order preferences are still $4 \prec 3 \prec 2 \prec 5 \prec 1$, though. To maintain constant the overall arrival rate λ,

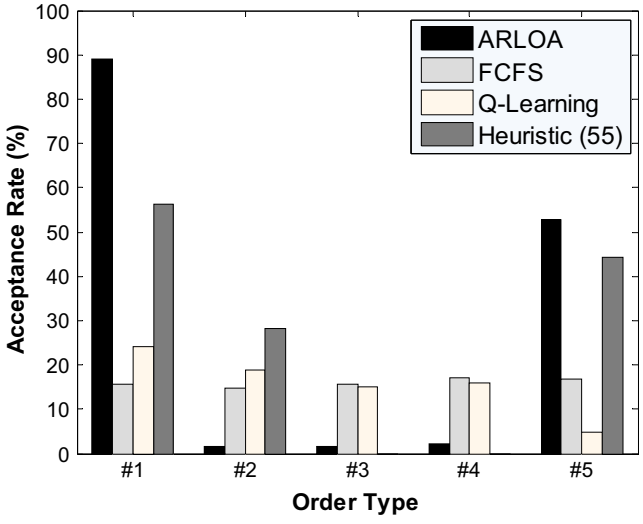

Fig. 8. Selectivity compared for different OA policies in Case 1.

Table 2
Demand changes (prices and arrival rates) for Case 2.

	P1	P2	P3	P4
Products attributes				
Revenue (before change)	5	2	1	0.5
Revenue (after change)	4	1	0.5	0.5

	#1	#2	#3	#4	#5
Order type attributes					
Arrival percent (before change) (%)	15	20	25	25	15
Arrival percent (after change) (%)	5	20	35	35	5

specific arrival rates for order types #3 and #4 are subjected to an identical increase in their arrival rates (see Table 2). As type #1 orders are still the most profitable ones, weakly revenues will be necessarily lower after the change. In Fig. 9, the average weakly revenues based on five independent runs are shown for the ARLOA policy and three threshold heuristics with $b = 40$, $b = 55$ and $b = 65$. As can be seen, ARLOA provides the best OA policy both before and after the change. The heuristic rule with $b = 55$, which was comparable to ARLOA before the change is made, performs sensibly sub-optimal afterwards, though. As can be expected, lowering the threshold for acceptance is needed to handle this kind of environmental changes. The effect on order selectivity for the ARLOA policy resulting from these changes to arrival rates and prices are shown in Fig. 10. The way ARLOA handles demand changes is by increasing the number of type #4 orders which are accepted. Also, more orders of the less profitable types are accepted in accordance with their preferences as shown in Fig. 10b.

Changes to arrival rates are quickly detected by a lowering of the resource utilization index for extruders. This fact triggers a resetting of the exploration strategy of ARLOA and a new learning curve begins as can be seen in Fig. 9. In Fig. 11, the effect of demand changes on resource utilization for ARLOA is compared with resource usage lowering when a heuristic rule with $b = 65$ is used. Note that the ARLOA policy is able to recover a high level of utilization after the environment changes. It is remarkable the severe detrimental effect on resource utilization of a sub-optimal setting of the threshold b. High levels of utilization can only be restored by reoptimizing the value of b.

4.2.3. Case 3: a new order type

In this scenario, the OA policy should respond to an environmental change related to the arrival of a new order type which is even more profitable than the current best type. The MTO system is initially receiving four order types which are numbered #2–#5 with order preferences defined as follows: $4 \prec 3 \prec 2 \prec 5$ (see Table 3). The overall arrival rate λ is maintained constant throughout by

Please cite this article in press as: Arredondo, F., & Martinez, E. Learning and adaptation of a policy for dynamic order acceptance in make-to-order manufacturing. *Computers & Industrial Engineering* (2009), doi:10.1016/j.cie.2009.08.005

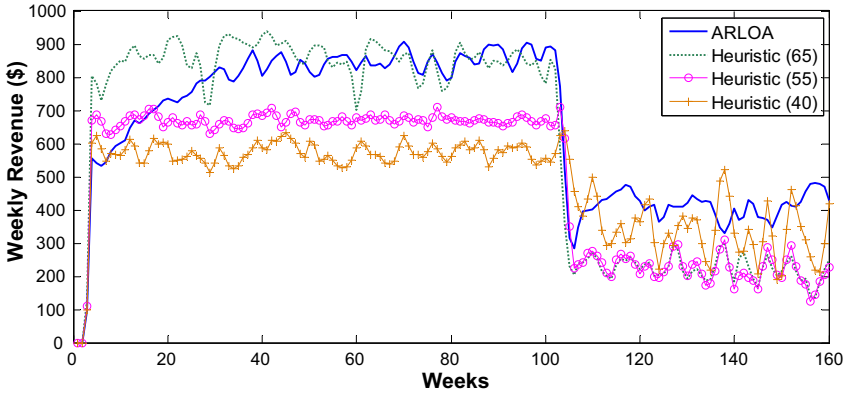

Fig. 9. ARLOA adaptiveness to a significant reduction of the arrival rates of more profitable orders.

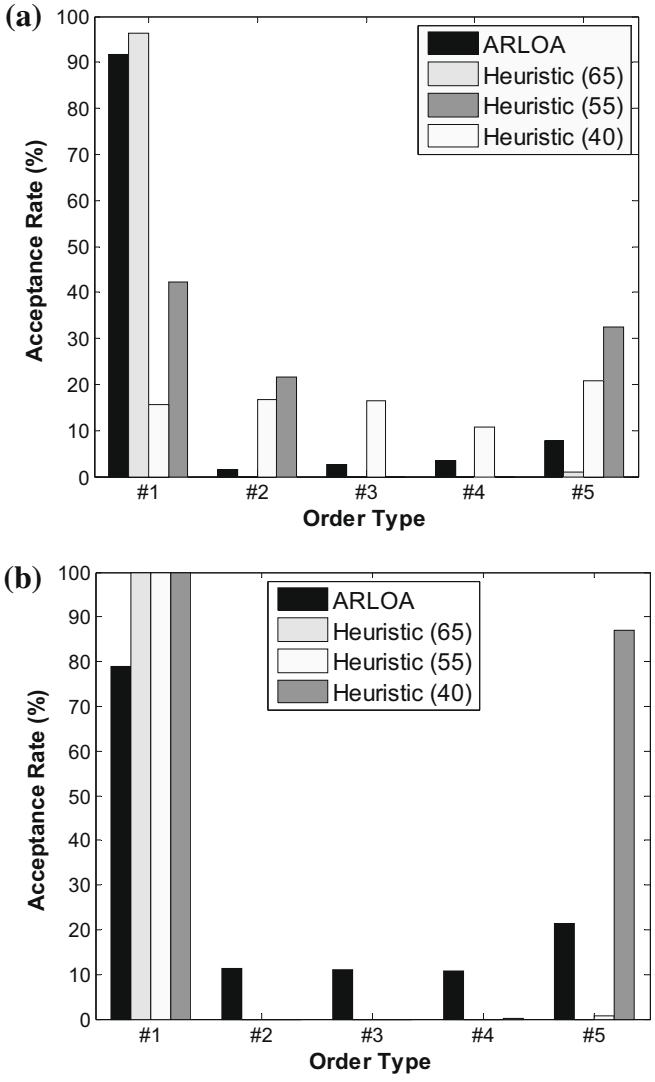

Fig. 10. Selectivity of ARLOA and threshold heuristics compared in Case 2: (a) before change and (b) after change.

lowering the arrival rates of order types #3 and #4. Arrivals of the new order type #1 occur from week #80 onwards. The exploration strategy in the ARLOA algorithm provides a high default value for acceptance $\mathscr{R}(o, acc)$ to these new orders which allows the OA policy to adapt quickly and grasp the opportunity for increasing the revenue obtained as it is shown in Fig. 12. It is noticeable how a near-optimal setting for the heuristic threshold ($b = 50$) degrades significantly when arrivals of this new order type take place. As expected, the way to handle this demand change would be to increase the threshold (see learning curve for $b = 50$). However, the difficult issue to be addressed is choosing fast the best value for the threshold so that revenues are maximized.

Fig. 13 provides a clear picture of the ARLOA distinctive capability for selective order acceptance. After the arrival of the new order type, more than 60% of order type #1 are accepted, some orders are lost because their due dates are too tight to be inserted. The only way to increase the number of accepted orders of the more profitable types #1 and #5 is attempting due date negotiation with clients so as to consolidate this market segment.

4.2.4. Case 4: due date negotiation

To take advantage of ARLOA selectivity, the demand scenario in Table 4 is considered for product pricing and arrival rates.

Negotiation is modeled here as offering clients of selected orders the possible postponement of the order due date so as to guarantee order insertion in the production scheduling. It is assumed that the 90% of the due dates offered are accepted by customers. An important remark is the implemented logic chosen to decide which orders are offered a delayed due date. Only orders for which the estimated $\mathscr{R}$-value for acceptance is greater than the average revenue ρ are negotiated ($\mathscr{R}(o,\ acc) > \rho$. Figs. 14 and 15 vividly depict the benefits resulting from selective negotiation of due dates when the ARLOA algorithm is used. It is worth noting that even though the heuristic threshold b has been defined quite conveniently, negotiation does not provide a meaningful advantage in this case. The explanation is rather simple: as the heuristic rule is not selective above the threshold, negotiation can only provide marginal improvement as there exist alternative order types to profit from without engaging in due date negotiation. The remarkable selectivity of the ARLOA policy when due date negotiation is allowed is shown in Fig. 15.

4.2.5. Case 5: changes in product prices

Market conditions constantly change order profitability. The OA policy should be able to detect and to adapt to these changes by accepting orders in the rejection set to look out for opportunities. Let's assume that at time 400 days (80 working weeks), significant changes to product prices are made as it is detailed in Table 5. The demand change is first detected using ARLOA based on a steady decrease in the average revenue ρ obtained. As a result, a resetting event of the exploration parameter $\varepsilon \leftarrow \varepsilon_0$ is triggered. The weekly revenue obtained using the OA policy generated by the ARLOA is compared in Fig. 16 with two competitive heuristic rules. The

Please cite this article in press as: Arredondo, F., & Martinez, E. Learning and adaptation of a policy for dynamic order acceptance in make-to-order manufacturing. *Computers & Industrial Engineering* (2009), doi:10.1016/j.cie.2009.08.005

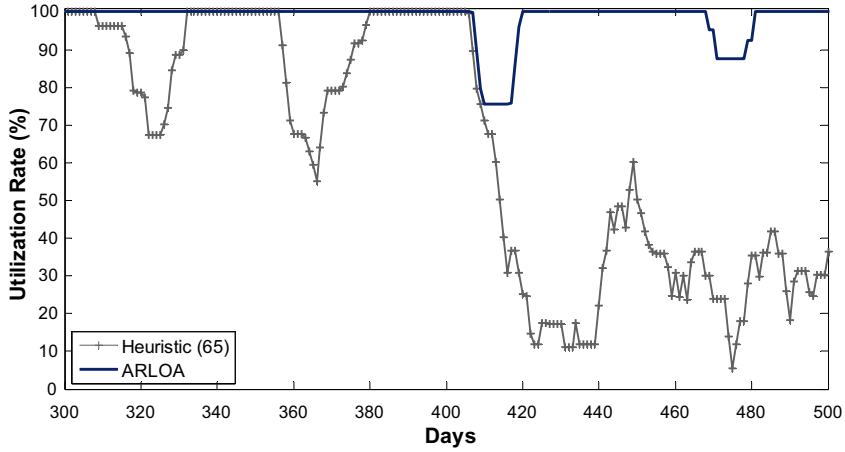

Fig. 11. Utilization rate for ARLOA and the threshold heuristic (65) compared in Case 2.

Table 3
Arrival of a new order type with high profitability (Case 3).

	#1	#2	#3	#4	#5
Order type attributes					
Arrival percent (before change) (%)	0	20	35	35	10
Arrival percent (after change)	10	20	30	30	10

policy based on ARLOA performs always better than any well-tuned heuristics due to its sensitivity and selectivity as can be seen in Fig 17. It is worth noting that type #2 orders (which are mostly rejected before prices are changed), are the preferred ones after product prices are modified.

5. Final remarks

Based on the key role of order acceptance in MTO manufacturing systems, a novel hierarchy of functions for intelligent order management when demand exceeds capacity has been proposed. As the key piece of this hierarchy, the OA-agent has the role of deciding which orders are worth negotiating and which orders can be released to the shop-floor so as to attempt their insertion into the current production schedule. For negotiation, the OA-agent resorts to the clear-cut criterion that the order value for acceptance $\mathscr{R}(o, acc)$ should be greater than $\mathscr{R}(o, rej)$ to pinpoint which attributes (size, revenue, product mix, etc.) must be changed in order to make the order part of the acceptance set, i.e. where $\mathscr{R}(o, acc) > \mathscr{R}(o, rej)$. The attractiveness (value) for acceptance of a given order is thus heavily dependent on its attributes regarding the objective of maximizing the short-term revenue but also depends on market factors such as arrival rates of orders with higher $\mathscr{R}$-values for acceptance. Thus, values for acceptance and rejection of an order are dynamic entities which require that the OA-agent includes an active exploration strategy which allows conveniently updating the OA policy to cope successfully with environmental changes. The proposed algorithm ARLOA integrates all the means required for effectively learning $\mathscr{R}$-values under uncertainty.

The importance of developing an OA policy under uncertainty using the ARLOA algorithm have been highlighted for MTO systems subjected to unknown order types and environmental changes related to profitability and arrival rates. To deal with many sources of uncertainty for revenue management, the idea of integrating order similarity and locally weighted learning for dynamic order acceptance in an over-demanded MTO manufacturing system is proposed. Since order attributes vary over a continuum, the presented work deliberatively resorts to attribute similarity in order to estimate order $\mathscr{R}$-values from previously accepted orders using a distance (kernel) function. This is very important for defining a reward function which not only provides rewards for accepted orders but also for rejected ones. In previous related works by Mainegra Hing (2006) and Mainegra Hing et al. (2007),

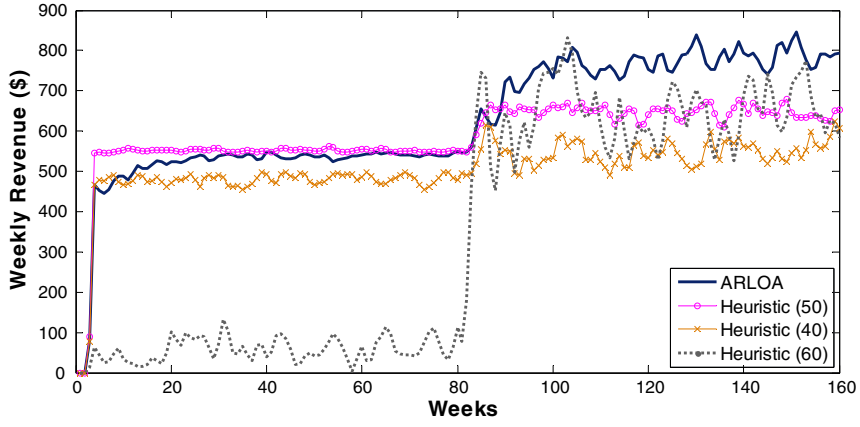

Fig. 12. Learning curves for ARLOA and three heuristic rules to accommodate arrivals of a new order type (Case 3).

Please cite this article in press as: Arredondo, F., & Martinez, E. Learning and adaptation of a policy for dynamic order acceptance in make-to-order manufacturing. *Computers & Industrial Engineering* (2009), doi:10.1016/j.cie.2009.08.005

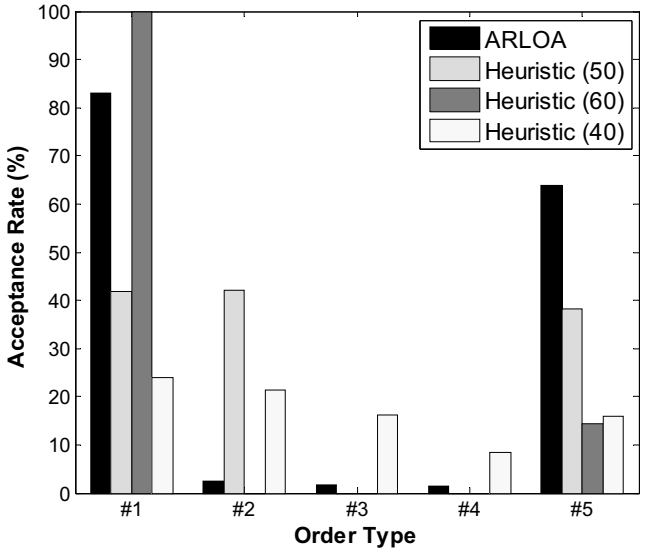

Fig. 13. Selectivity compared for ARLOA and some OA heuristic rules once the new type orders are already arriving (Case 3).

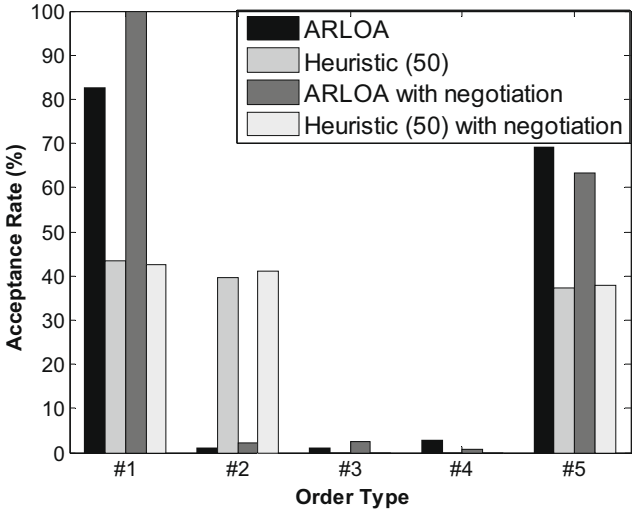

Fig. 15. Selectivity for ARLOA and OA *heuristic* (*50*) compared in Case 4.

Table 4
Due date negotiation of more profitable orders (Case 4).

	P1	P2	P3	P4
Products attributes				
Revenue ($/kg)	5	2	1	0.5
	#1	#2	#3	#4	#5
Order type attributes					
Arrival percent (%)	5	20	30	30	15

Table 5
Product price changes (Case 5).

	P1	P2	P3	P4
Products attributes				
Revenue (before change)	5	2	1	0.5
Revenue (after change)	3	6	1.5	1
	#1	#2	#3	#5	#4
Order type attributes					
Arrival percent (%)	10	20	30	30	10

zero reward is given to rejection decisions which is no indicative at all of the goodness or badness of rejecting some order types. As a result, the proposed ARLOA algorithm is very effective to quickly learn which orders are worth accepting and to distinguish them from those orders whose rejection is mandatory to accept more profitable orders in the near future.

The issue of policy adaptation has been addressed by providing a special meaning to the notion of exploitation and exploration along with conveniently resetting the maximum level of exploration. To this aim, the order_list found by the OA-agent at each decision epoch has been divided into the acceptance set and the rejection set. For exploitation, only orders in the acceptance set should be accepted. However, to discover new order types, and to adapt to lower arrival rates of more profitable order types, the OA-agent should also continuously explore by choosing orders from the rejection set. The trade-off between exploitation and exploration is a tricky issue that has been successfully addressed here by resetting. The ARLOA algorithm resorts to a steady lowering of the exploration parameter ε until a resetting event occurs. Exploration resetting is mainly based on monitoring certain indices such as the short-term average revenue or a bottleneck resource utilization. There can be more elaborated strategies for resetting exploration, but this one is very simple and was able to provide a quick enough response to environmental changes affecting the

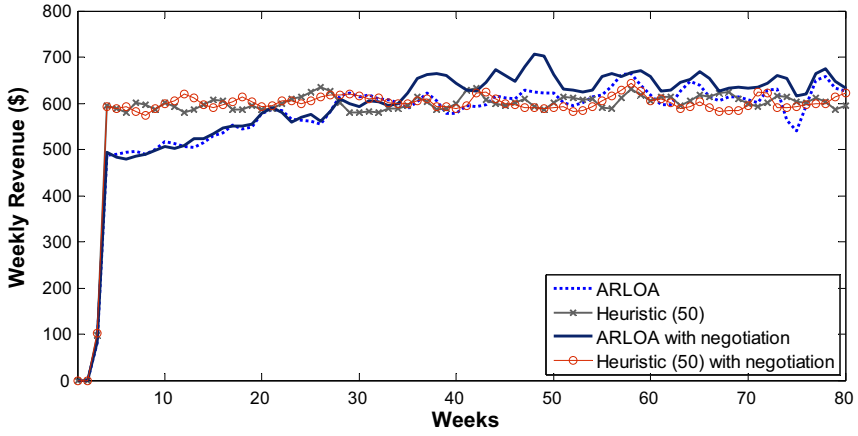

Fig. 14. Learning curve for ARLOA and *heuristic* (*50*) when due date negotiation of more profitable orders is allowed (Case 4).

Please cite this article in press as: Arredondo, F., & Martinez, E. Learning and adaptation of a policy for dynamic order acceptance in make-to-order manufacturing. *Computers & Industrial Engineering* (2009), doi:10.1016/j.cie.2009.08.005

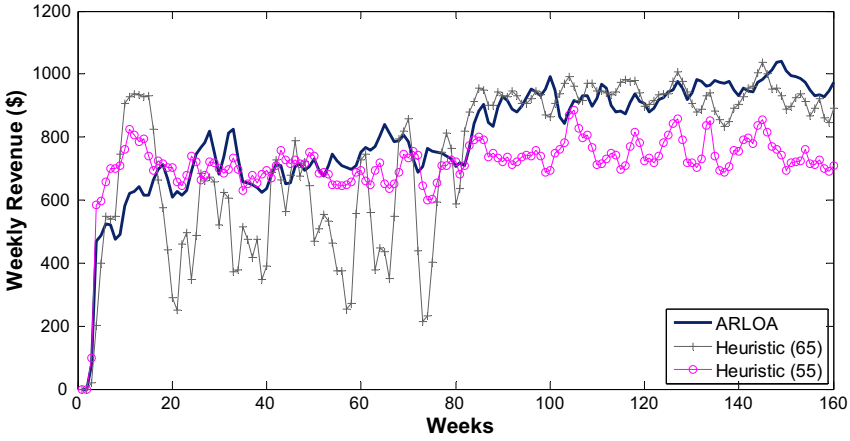

Fig. 16. Learning curves for ARLOA and *heuristics* in Case 5.

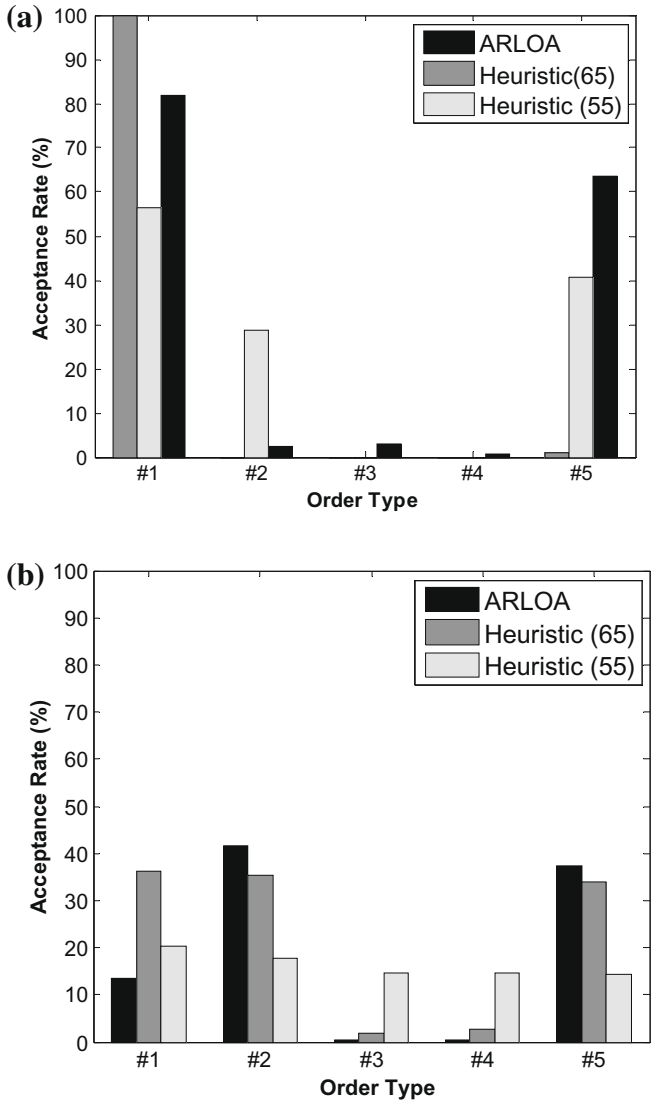

Fig. 17. Selectivity compared for Case 5. (a) Before change and (b) after change.

$\mathscr{R}$-values for acceptance and rejection upon which the OA policy is defined.

There are several avenues which are being pursued to further the research presented in this work. The issue of developing a meta-learning layer based on adaptive clustering techniques is one problem currently being addressed. There are many advantages of explicitly identifying clusters of similar orders based on values for acceptance and rejection. Firstly, the OA policy can be easily understood in the attribute space using order types which are part of the learning process. Secondly, OA heuristics can be generated and human supervision of the OA policy is thus much easier. Also, propagation of $\mathscr{R}$-values using adaptive clustering is a very effective approach which may provide a definitive advantage in terms of learning rate and policy adaptation. Another research avenue which is being tackled is introducing order recognizers to define archetypes (features) of order attributes for automatic characterization of *a priori* unknown order type. Finally, agent roles such as negotiation, order acceptance and order insertion, defined in the intelligent OA hierarchy of Fig. 1, are currently being used to define a multi-agent architecture for order management in MTO systems.

References

Atkeson, C., Moore, A., & Schaal, S. (1997). Locally weighted learning. *Artificial Intelligence Review, 11*, 11–73.

Barut, M., & Sridharan, V. (2004). Design and evaluation of a dynamic capacity apportionment procedure. *European Journal of Operational Research, 155*, 112–133.

Barut, M., & Sridharan, V. (2005). Revenue management in order-driven production systems. *Decision Sciences, 36*, 287–316.

Calosso, T., Cantamessa, M., Vu, D., & Villa, A. (2003). Production planning and order acceptance in business to business electronic commerce. *International Journal of Production Economics, 85*, 233–249.

Calosso, T., Cantamessa, M., & Gualano, M. (2004). Negotiation support for make-to-order operations in business-to-business electronic commerce. *Robotics and Computer-Integrated Manufacturing, 20*, 405–416.

Das, T., Gosavi, A., Mahavedan, S., & Marchalleck, N. (1999). Solving semi-markov decision problems using average reward reinforcement learning. *Management Science, 45*(4), 560–574.

Defregger, F., & Kuhn, H. (2007). Revenue management for a make-to-order company with limited inventory capacity. *OR Spectrum, 29*, 137–156.

Ebben, M., Hans, E., & Olde Weghuis, F. (2005). Workload based order acceptance in job-shop environments. *OR Spectrum, 27*, 107–122.

Enns, T. (2000). Evaluating shop-floor input control using rapid modelling. *International Journal of Production Economics, 63*(3), 229–241.

Enns, S., & Costa, M. (2002). The effectiveness of input control based on aggregate versus bottleneck work loads. *Production Planning and Control, 13*, 614–624.

Herbots, J., Herroelen, W., & Leus, R. (2007). Dynamic order acceptance and capacity planning on a single bottleneck resource. *Naval Research Logistics, 54*(8), 874–889.

Ivanescu, C., Fransoo, J., & Bertrand, J. (2002). Makespan estimation and order acceptance in batch process industries when processing times are uncertain. *OR Spectrum, 24*, 467–495.

Ivanescu, C. (2004). Order acceptance under uncertainty in batch process industries. PhD thesis, Technische Universiteit Eindhoven, Eindhoven.

Jalora, A. (2006). Order acceptance and scheduling at a make-to-order system using revenue management. PhD thesis, Texas A&M University.

Please cite this article in press as: Arredondo, F., & Martinez, E. Learning and adaptation of a policy for dynamic order acceptance in make-to-order manufacturing. *Computers & Industrial Engineering* (2009), doi:10.1016/j.cie.2009.08.005

Ten Kate, H. (1994). Towards a better understanding of order acceptance. *International Journal of Production Economics, 37*, 139–152.

Mainegra Hing, M., van Harten, A., & Schuur, P. (2001). Order acceptance with reinforcement learning. Technical Report 66. University of Twente, Netherlands.

Mainegra Hing, M. (2006). Order acceptance under uncertainty: A reinforcement learning approach. PhD thesis, Universiteit Twente, Technische Universiteit Eindhoven, Netherlands.

Mainegra Hing, M., van Harten, A., & Schuur, P. (2007). Reinforcement learning versus heuristics for order acceptance on a single resource. *Journal of Heuristics, 13*, 167–187.

Moreira, M. (2005). Planning and controlling job-shop operations. PhD dissertation, Faculty of Economics, University of Porto, Portugal.

Nandi, A., & Rogers, P. (2003). Behavior of an order release mechanism in a make-to-order manufacturing system with selected order acceptance. In *Proceedings of the 2003 winter simulation conference* (pp. 1251–1259).

Nandi, A., & Rogers, P. (2004). Using simulation to make-to-order acceptance/rejection decision. *Simulation, 80*(3), 131–142.

Nawijn, W. (1985). The optimal look-ahead policy for admission to a single server system. *Operations Research, 33*(3), 625–643.

Philipoom, P., & Fry, T. (1992). Capacity-based order review/release strategies to improve manufacturing performance. *International Journal of Production Research, 30*(11), 2559–2572.

Quante, R., Meyer, H., & Fleischmann, M. (in press). Revenue management and demand fulfilment: Matching applications, models, and software. *OR Spectrum*. doi: 10.1007/s00291-008-0125-8.

Raaymakers, W. (1999). Order acceptance and capacity loading in batch process industries. PhD thesis, Technische Universiteit Eindhoven, Netherlands.

Raaymakers, W., Bertrand, J., & Fransoo, J. (2000a). The performance of workload rules for order acceptance in batch chemical manufacturing. *Journal of Intelligent Manufacturing, 11*, 217–228.

Raaymakers, W., Bertrand, J., & Fransoo, J. F. (2000b). Using aggregate estimation models for order acceptance in a decentralized production control structure for batch chemical manufacturing. *IIE Transactions, 32*, 989–998.

Schwartz, A. (1993). A reinforcement learning method for maximizing undiscounted rewards. In *Proceedings of the tenth international conference on machine learning* (pp. 298–305).

Singh, S. (1994). Reinforcement learning algorithms for average-payoff markovian decision processes. In *Proceedings of the 12th national conference in artificial intelligent, 2002–2007*. MIT Press.

Smart, W., & Pack Kaelbling, L. (2000). Practical reinforcement learning in continuous spaces. In *Proceedings of the 17th international conference on machine learning* (pp. 903–910). Morgan Kaufmann.

Sutton, R., & Barto, A. (1998). *Reinforcement learning: An introduction*. London, England: MIT Press.

Snoek, M. (2000). Neuro-genetic order acceptance in a job shop setting. In *Proceedings of 7th international conference on neural information processing*, Korea (pp. 815–819).

Wang, J., Yang, J., & Lee, H. (1994). Multicriteria order acceptance decision support in over demanded job-shops: A neural network approach. *Mathematical and Computer Modelling, 19*(5), 1–19.

Wester, F., Wijngaard, J., & Zijm, W. (1992). Order acceptance strategies in a production-to-order environment with setup times and due-dates. *International Journal of Production Research, 30*(6), 1313–1326.

Wight, O. (1970). Input/output control: A real handle on lead time. *Production and Inventory Management Journal, 11*(3), 9–30.

Wouters, M. (1997). Relevant cost information for order acceptance decisions. *Production Planning and Control, 8*(1), 2–9.

Zorzini, M., Corti, D., & Pozzetti, A. (2008). Due date (DD) quotation and capacity planning in make-to-order companies: Results from an empirical analysis. *International Journal of Production Economics, 112*(2), 919–933.

Please cite this article in press as: Arredondo, F., & Martinez, E. Learning and adaptation of a policy for dynamic order acceptance in make-to-order manufacturing. *Computers & Industrial Engineering* (2009), doi:10.1016/j.cie.2009.08.005