Intelligent Order Acceptance in Make-to-Order Manufacturing using Reinforcement Learning

Facundo Arredondo and Ernesto Martínez

INGAR (CONICET-UTN), Avellaneda 3657, Santa Fe, S3002 GJC, Argentina.
{farredondo, ecmarti}@santafe-conicet.gov.ar

Abstract. Order acceptance (OA) is a critical decision-making problem at the interface between customer relationship management and production planning in make-to-order manufacturing. The main issue in this work is the development of an intelligent OA framework based on reinforcement learning. The attention is focused on iterative development of an OA policy based on evaluative feedback of profits obtained. This policy is first obtained using simulations and later updated on-line based on revenues per unit cost of processing capacity to deal with problem uncertainty. Locally weighted regression is used to generalize cumulative rewards of accepting/rejecting similar orders regarding attributes such as product mix, size and due date. The balance exploration-exploitation of the learned policy is based on classifying an arriving order as belonging either to the acceptance set or to the rejection set. Results highlight the importance of automated learning of an adaptative OA policy using actual rewards when demands exceed capacity

Keywords: Order acceptance, reinforcement learning, make-to-order manufacturing, locally weighted regression, intelligent agent

1 Introduction

With the current trend towards highly segmented markets, companies of all sizes and industrial sectors are increasingly shifting towards Make-to-Order (MTO) production systems [1,2]. In order to face the challenge of MTO manufacturing, order management systems have to dramatically improve agility and responsiveness by developing decision support tools that integrate real time order negotiation with shop-floor production planning and scheduling. Aimed at targeting very specific market niches, successful MTO systems should careful select dynamically which orders are worth accepting or negotiating and which orders are not from the viewpoint of opportunity costs and revenue management [3]. Order-driven production systems increasingly focus on selling customized products and benefits generated are thus strongly order-specific. Usually, a company may satisfy several customer segments, faces a stochastic demand, and a time-varying product price. The main issue is how to selectively accept/reject/negotiate orders under uncertainty in order to maximize cumulative profit gain.

XXXIV Conferencia Latinoamericana de Informática (CLEI 2008), pp. 270-279, 2008.

In recent years, a new paradigm called agent technology has been widely recognized as a promising approach for automating decision making in supply chain negotiation and e-commerce [4]. The main objective of this work is to develop a reinforcement learning algorithm enabling a software agent to carry out autonomous learning and adaptation of a reactive policy which can handle the OA environment in MTO production systems. Reinforcement learning [5] is a new approach where the software agent learns a policy by directly interacting with its environment and responding to rewards received by taking actions which cause state transitions.

2 Intelligent order acceptance

The order acceptance problem deals with accepting or rejecting orders based on some criterion to complete the order within the requested delivery lead time. Orders usually arrive stochastically and have a desired due date (DD) attached. When capacities are limited and demand is high, the company has to decide which orders to accept, which orders to reject and, if it is possible, which orders to negotiate in order to maximize cumulative profit. Accepting an order can prevent the MTO production system from accepting more profitable orders in the future [6]. On the other hand, the rejection of an order may have repercussions for future customer relations. Lack of coordination between the decision to accept/reject/negotiate an order (with its attributes) and the opportunity costs due previous capacity assignments is often the consequence of the usual functional separation between the order-acceptance decision, which is made by the sales department, and capacity planning, which usually lies in the hands of the production department [7].

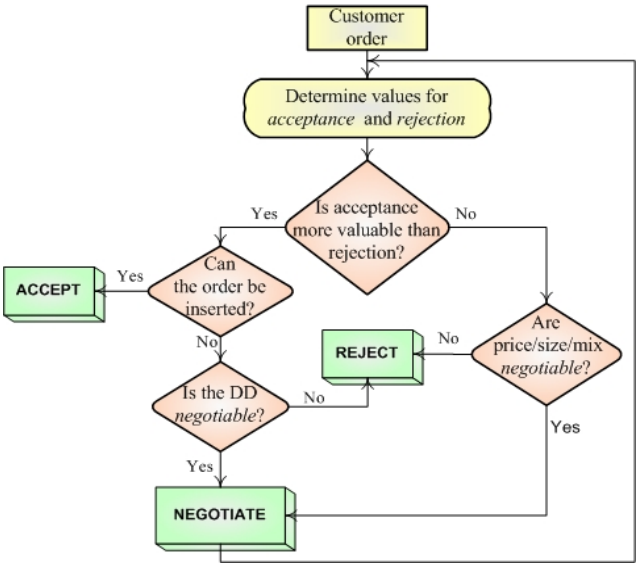

Fig. 1. Order acceptance decision

One of most important attributes of an intelligent agent is its ability to deal with uncertain environments by adaptation. Thus, the intelligent agent for OA should be able to selectively accept, reject or negotiate orders so as to accumulate profits without a model of the OA environment. In this work, it is argue that the logic of the agent for decision-making should be mainly based on the concept of *values* for order acceptance and rejection decisions as proposed in Fig. 1. An arriving order is first considered for acceptance or rejection based on its contributing value to discounted reward accumulation. If the order is having enough value for being accepted, its insertion into the current production plans is then attempted. If order insertion for shop-floor processing is achieved, the order is then accepted; otherwise, the due date may be negotiated with the client. Whenever the value for order acceptance is smaller than the corresponding value for order rejection, negotiation of order attributes is attempted to increase the order acceptance value by either increasing its revenue, changing its size or product mix.

The value Q for an order acceptance decision (or rejection) is defined here as the corresponding reward (or penalty) plus the cumulative relative revenue that can be obtained by acting optimally thereafter. Should the corresponding values for acceptance and rejection for any order be perfectly known at the arrival time is rather straightforward to always act optimally since only orders which are contributing the most to profit accumulation and can be inserted in production plans are accepted and the rest are rejected. Moreover, if values are known *a priori*, would be possible to assign the perfect reward (or penalty) based on the action that has been taken. It is worth noting that the values for acceptance and rejection of an order depend on the arrival rates of all types of orders. Also, the value of an order is heavily dependent on its attributes such as mix of products, price and size. For an order that should be rejected, the negotiation process should try to increase enough the value for acceptance so as to exceed the value for order rejection.

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 [6] and [8]. If an arriving order whose acceptance value is greater than its rejection value 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 not because of its profit contributing value but because available processing capacity is insufficient due previous accepted orders. If values for acceptance and rejection decisions in Fig. 1 are known exactly the only information required is whether the order can be inserted or not in the current production plan. There is not need for explicitly considering profiles for planned resource usage or any other planning or scheduling details in the decision-making logic for order acceptance. The main issue to successfully carry out order acceptance as shown in Fig. 1 is to know the values for acceptance and rejection of an arriving order. Due to the many sources of uncertainty involved, reinforcement learning, from simulated experience, is proposed for value estimation and generalization based on order similarity. Estimated values for acceptance and rejection decisions can be updated on-line using real experience to reflect time-varying order arrival rates and observed profitability.

3 Policy learning and adaptation

3.1 OA problem description

Following the idea given in Section 2, our OA problem can be described more formally as follow. Consider a case where an MTO system is offering different products. Arrivals orders take place at any continuous time moment following some regularity pattern. In what follows it is assumed that orders arrive at the system according to a Poisson process with global arrival rate λ. Each order *i* is characterized by a mix of different products and has a due date ($\mathcal{DD}_i$), size (w_i) and profit per total processing costs that generates a reward (r_i) upon acceptance decisions are taken. Orders arrive continuously in the time, but acceptance/rejection decisions are only considered at discrete time moments *t*. In these moments, called decision epochs, the agent must decide which orders in current *order_list* are accepted, rejected or negotiated based on the estimated values. Decision epochs, labelled by *t, t+1, t+2...*, correspond to fixed time windows, e.g. days, for which a list of all arriving orders (*order_list*) in the time period is created and decisions are taken. The system has a limited processing capacity. When the available capacity is not sufficient for an arriving order and their attributes are not negotiable, the only option is rejection.

3.2 Reinforcement Learning

To overcome uncertainty in the development of an order acceptance policy without *a priori* knowledge resorting to reinforcement learning [5] is proposed. RL deals with the problem of how an autonomous agent can learn to select proper actions for achieving its goals through continuously interacting with its environment. A situated learning agent is characterized by a knowledge structure, a learning method or rule to update its knowledge and a decision-making policy. The simulated environment is characterized by states, rewards and probabilistic transitions.

Figure 3 summarizes the interaction between the learning agent and its environment in the OA problem. At each decision step, the agent receives a perception of the environment state (1). The received information is made up of orders (and their attributes) which had arrived during a given period of time, e.g. day. These orders form the current order list (*order_list*). On that basis, the learning agent must select and perform an action (2). The action set includes acceptance or rejection of orders in the *order_list*. Selection is made according to the current decision policy developed by the agent. As a result of the action taken, the simulated environment makes a transition (3) to a new state and a reinforcement or reward signal is generated (4). The new state includes all orders belonging to the *order_list* minus the order that was just considered for acceptance or rejection. The reward signal and the new state (Updated Order List) are both received by the RL-agent (5) and through its learning rule are used to update its knowledge about the environment, and consequently it can update its decision-making policy (6). At each epoch decision, the agent implements a mapping from states to probabilities of selecting each possible action. This mapping is

called the *agent's policy*, π. In reinforcement learning, the agent changes this policy as a result of its experience (adaptation). This process is repeated until that the *order_list* is emptied by acceptance or rejection decisions.

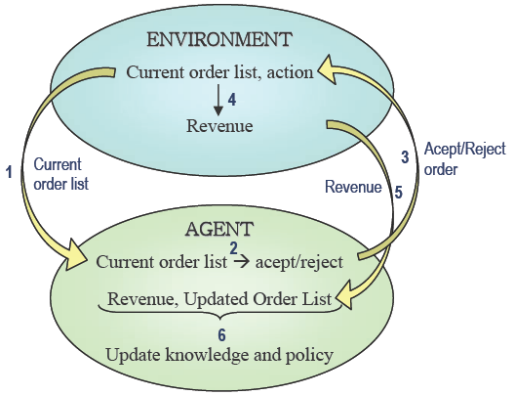

Fig. 2. Agent-Environment interaction for OA problem

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 $Q(s,a)$, where s is the order state and a stands for the finite set of possible actions in s. When a given action is chosen at state s, Q corresponds to the cumulative reward that can be gain by choosing a and acting optimally thereafter. Should the values of $Q(s,a)$ are known, the optimal policy is simply to chose accept/reject based on which action has the highest Q. Unfortunately, the value function Q is unknown *a priori* and can only be estimating by evaluative feedback along the sequence of actions (acceptance/rejection) and credit assignments to actions in the sequel using the actual profits obtained. Furthermore, even when the action set is finite, the states are defined by a continuum of product mixes which poses a difficult problem for generalizing the values of actions taken at the visited states to unseen ones.

The logic of the iterative update of Q-values for accepted orders and "similar" ones in their neighbourhood is shown in Fig. 3. The value of each order is estimated using a similarity of its attributes with accepted orders. The *order_list* is prioritized using their *Q-values* for acceptance. While there is processing capacity available orders in the list are accepted using a policy that combines exploitation with exploration (see description below). If there is not exists enough capacity and DD is not negotiable, the order is rejected. Details about the integration of reinforcement learning algorithm and locally weighted regression (LWR) for value generalization are given in Fig. 3.

3.2 Q-Learning

One of the most widely basic algorithms used in reinforcement learning [5] is the one allowing updating the values of state-action pairs using real or simulate experience. In its simple form, *one-step Q-learning*, is defined by:

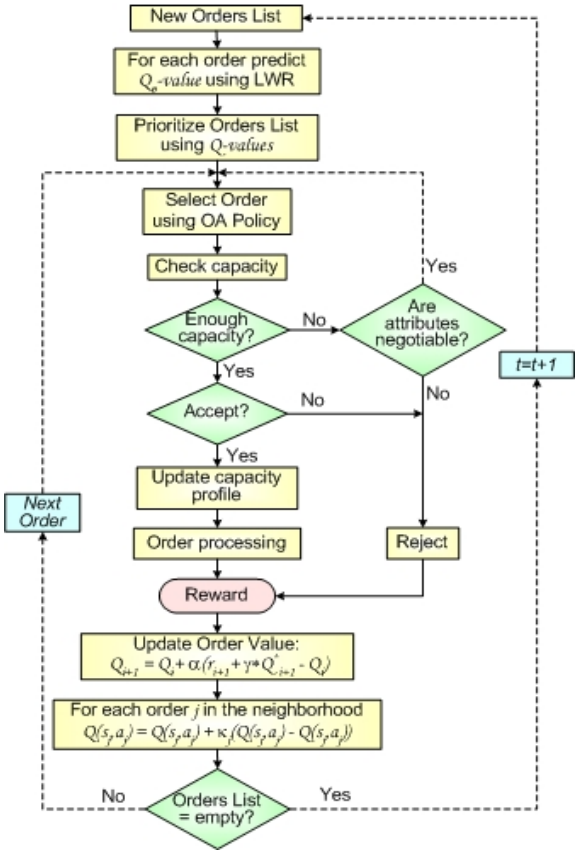

Fig. 3. Reinforcement learning of an adaptive OA policy using LWR

$$Q(s_i, a_i) \leftarrow Q(s_i, a_i) + \alpha\left[r_{i+1} + \gamma \max_a Q(s_{i+1}, a) - Q(s_i, a_i)\right] \quad (1)$$

where r_{i+1} is reward resulting of taking the action a_i at the state s_i and $0<\alpha<1$ is the learning rate. The parameter $0 <\gamma< 1$ is the discount rate which defines the present net value of future rewards (profits). Thus, if γ is close to zero short-term rewards are worthier than more distant ones, as $\gamma \rightarrow 1$ the policy becomes more farsighted.

3.3 State generalization

For a small RL problem, the estimates of value functions can be represented as a table with one entry for each state-action pair. However, for a large problem with a large number of states or actions, updating information accurately in such a large table may be a significant problem. One problem with the basic *Q-learning* algorithm is that it is assumed that the number of actions and states are finite. In our application the number

of actions is indeed finite but the order list states are not. To generalize across a continuum of states we resort to *locally weighted regression* (LWR) [9,10]. 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 further away. Performing a regression over all of the training data for every query would be extremely expensive. Therefore, this model attempts to fit the training data only in a region around the location of the query point. Training points in LWR are weighted according to a function of their distance from the query point. This function is typically a kernel function *(k)*, with 'width' parameter know as the bandwidth *(h)*:

$$k(d,h) = e^{-\left(d/h\right)^2} \tag{2}$$

where d is the Euclidian distance between the query point and each point in the training dataset. Since most of data points are likely to be far from the query point, they have little effect on *Q*-value estimation for an order. Typically, a minimum number of points are needed for estimating the value of accepting/rejecting an order. With fewer data points, a default Q-value of zero is returned.

A key issue in local value propagation is order similarity which allows learning to form group of orders in the attribute space with similar values for acceptance and rejection. In the LWR algorithm these clusters of similar orders are defined using some short of distance (e.g. euclidean) between a query point and orders in the neighborhood. Thus, order similarity integrates the locality or order attributes with their corresponding values. It is noteworthy that even when orders are not segmented by type *a priori*, the proposed learning algorithm will readily discovered the existence of such order types. The proposed integration of LWR with the basic *Q-learning* algorithm makes possible to propagate *Q-value* updates to all accepted orders in the region of influence around the query point, defined by the kernel function in Eq. (2).

3.4 Exploration vs Exploitation

To effectively learn a policy in the face of uncertainty the RL-agent must address the dilemma of exploitation vs exploration. This means that to discover better policies to exploit the agent should choose orders were rejection seems more valuable than acceptance. Without exploration the agent can only exploit knowledge from accepted orders. To exploit more in the future the agent should accept apparently less profitable orders to know more about the OA environment. 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- ε) the action with highest value is chosen whereas with probability ε any of the non-optimal action is chosen. To address the OA problem, the balance between exploration and exploitation is based on ranking the *order_list* using *Q-values* for order acceptance. Accordingly, with probability (1-ε) the order at the top of the *order_list* is accepted, if enough processing capacity is available. On the other hand, with probability ε, any of the other order in the current list is accepted regardless of its *Q-value*. 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 increasingly lowered using [5]:

$$\varepsilon = \frac{\varepsilon_0}{1 + t/t_0} \tag{3}$$

where ε_0 is the initial value of ε before any interaction and t_0 is a parameter which defines the progressive decay of exploration. The lower is t_0, the greater is the exploration decay rate towards only exploitation in OA.

3.4 Reward function

The way acceptance/rejection decisions are given rewards or penalties is the only feedback from the OA environment to the learning agent. In this proposal, whenever an order is accepted, the reward is defined as the profit per unit cost of processing capacity (p_i). For rejected orders, the reward is defined as follows:

$$r_i = \begin{cases} -p_i & \text{if } Q_a \geq Q_r \\ 0 & \text{if } Q_a < Q_r \end{cases} \tag{4}$$

where p_i is the estimated reward based on similar orders. Q_a and Q_r are the estimated values corresponding to acceptance and rejection decisions, respectively. It is worth noting that using estimated values a penalty is given to orders which are rejected when they should be accepted ($Q_a \geq Q_r$).

4 Example

Our case is about a manufacturing environment in which production is driven by multiple-product orders. Plant equipment items are semi-continuous extruders which process orders for 4 products. Each order consists of a mix of different products, a due date and a size and generates a revenue upon decision is taken. There is a maximum regular capacity of hours per working day due to operator shift. It is assumed that all orders are always completed without violating their due dates which may require extra shift working and cost. Product mix of an arriving order is randomly generated from 5 well-differentiated clusters, a fact which is unknown *a priori* by the agent and is part of the learning process. The (assumed unknown by the agent) order preferences by type are as follows: $1 \prec 5 \prec 2 \prec 3 \prec 4$ based on the revenue per unit of capacity requested. Order due dates are generated using a uniform distribution between 5 and 10 days. At each decision epoch, the state is defined by the *order_list* prioritized by *Q*-values and the insertability of each order. As the list is being processed, always actions are to accept or to reject each order assuming it can be inserted. In some case,

orders more profitable can be negotiate and they are reconsidered to acceptance. Rewards are calculated as was explicated in 3.4. Orders arrive following a Poisson process with mean inter arrival times of 0.1 days. In the base case, order types are evenly distributed in the pool arrival process.

In Fig. 4, the learning curve for this illustrative example is given using the cumulative profit gain from accepted orders. A comparison is made between the OA policy using reinforcement learning (with and without negotiation) and the first-come-first-served (FCFS) acceptance policy. As can be seen, there is an important improvement in the cumulative profit using RL. Also, it is remarkable the gain that can be obtained if it is possible to negotiate the due date of those orders that pay off larger revenue per unit of requested capacity (in this case orders type #1). In Fig. 5, the proportion of each order type which is accepted is shown to highlight the selectivity of the learning policy. In Fig 6 and Fig 7 comparison is made between the OA policy using RL with negotiation and the FCFS acceptance policy. Figures shows the results obtained in a second case considered, were at time 250 days a 10% reduction in the arrival rate of the more profitable order type # 1 is made. To maintain constant the overall arrival rate the other order types are subjected to a 2.5% increase in their arrival rates. Results shows that are accepted almost 100% of orders type #1. Moreover, as can seen in Fig 6 and Fig 7, the RL-based OA policy selectively increases mainly the acceptance rate for orders type #5 to compensate for fewer order arrivals of the most profitable type #1 in total agreement with profitability preferences. Only small orders of type #3 and #4 are accepted. It is worth noting in Fig. 5 the fast response of the adaptive OA policy to the lowering of the arrival rate of the more profitable type #1.

5 Concluding Remarks

In this work, an intelligent system for OA problem has been proposed. An agent develops a policy for order acceptance problem under uncertainty using reinforcement learning without *a priori* segmentation of order types. Results emphasize the importance of negotiating those orders which are more profitable. Moreover, the RL-based policy readily responds to significant changes in arrival rates and order attributes whilst exhibiting a superior performance compared to the FIFS rule.

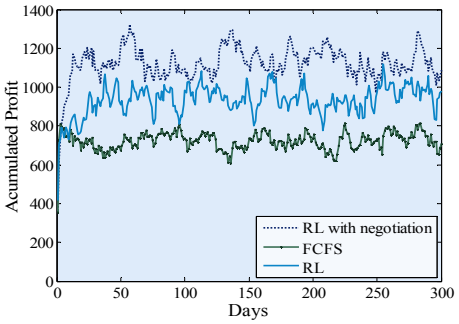

Fig. 4. Profit learning curve compared

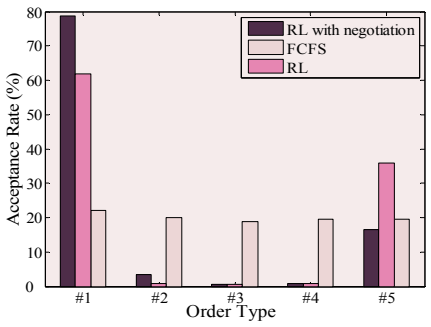

Fig. 5. Selectivity of the RL-based OA policy

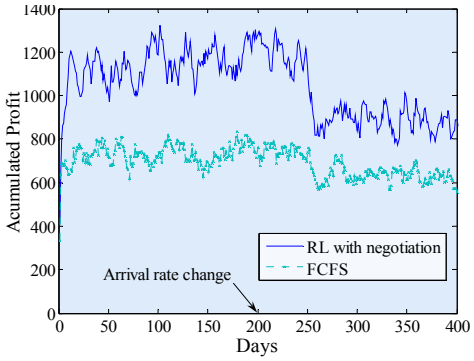

Fig. 6. OA policy response to a reduction of more profitable orders (type #1) arrival rate

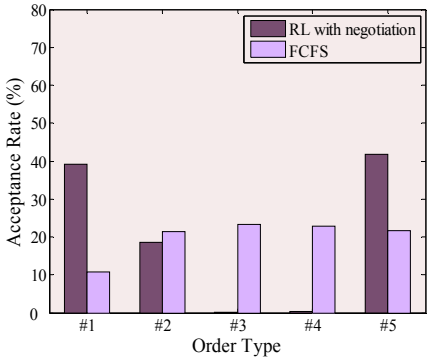

Fig. 7. Selectivity response to a reduction of the arrival rate of more profitable orders

References

1. Calosso, T., Cantamessa, M., Vu, D. and Villa, A.: Production planning and order acceptance in business to business electronic commerce. International Journal Production Economics, 85, 233--249 (2003).
2. Jalora, A.: Order acceptance and scheduling at a make-to-order system using revenue management. PhD Thesis, Texas A&M University (2006).
3. Defregger, F., Kuhn, H.: Revenue management for a make-to-order company with limited inventory capacity. OR Spectrum, 29, 137--156 (2007).
4. Wang, Y., Usher, M.: Application of reinforcement learning for agent-based production scheduling. Engineering Applications of Artificial Intelligence, 18, 73–82 (2005).
5. Sutton R., Barto A.: Reinforcement Learning: An introduction. MIT Press, Cambridge (1998).
6. Mainegra Hing, M., van Harten, A., Schuur, P.: Reinforcement learning versus heuristics for order acceptance on a single resource. J Heuristics, 13, 167-87 (2007).
7. Barut, M., Sridharan, V.: Revenue management in order-driven production systems. Decision Sciences, 36, 287--316 (2005).
8. Herbots, J., Herroelen, W., Leus, R.: Dynamic order acceptance and capacity planning on a single bottleneck resource. Naval Research Logistics, 54, 874--889 (2007).
9. Atkeson C, Moore A., Schaal S.: Locally Weighted Learning, Artificial Intelligence Review, 11, 11--73 (1997).
10. Smart W., Pack Kaelbling L.: Practical Reinforcement Learning in Continuous Spaces. In: Proc of the 17º International Conference on Machine Learning, pp. 903-910. Stanford University (2000).