A Markov Model for Inventory Level Optimization in Supply-Chain Management

Buffett, Scott

Eighteenth Canadian Conference on Artificial Intelligence (AI'2005) [Proceedings], 2005

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC · CNRC

*A Markov Model for Inventory Level Optimization in Supply-Chain Management**

Buffett, S.
May 2005

* published at the Eighteenth Canadian Conference on Artificial Intelligence (AI'2005). pp. 133-144. May 9-11, 2005. Victoria, British Columbia, Canada. NRC 48262.

Canada

A Markov Model for Inventory Level Optimization in Supply-Chain Management

Scott Buffett

Institute for Information Technology – e-Business,
National Research Council Canada
46 Dineen Drive, Fredericton, New Brunswick, Canada E3B 9W4
Scott.Buffett@nrc.gc.ca

Abstract. We propose a technique for use in supply-chain management that assists the decision-making process for purchases of direct goods. Based on projections for future prices and demand, requests-for-quotes are constructed and quotes are accepted that optimize the level of inventory each day, while minimizing total cost. The problem is modeled as a Markov decision process (MDP), which allows for the computation of the utility of actions to be based on the utilities of consequential future states. Dynamic programming is then used to determine the optimal quote requests and accepts at each state in the MDP. The model is then used to formalize the subproblem of determining optimal request quantities, yielding a technique that is shown experimentally to outperform a standard technique from the literature. The implementation of our entry in the Trading Agent Competition-Supply Chain Management game is also discussed.

Keywords: supply-chain management, Markov decision process, dynamic programming, purchasing.

1 Introduction

With the dramatic increase in the use of the Internet for supply chain-related activities, there is a growing need for services that can analyze current and future purchase possibilities, as well as current and future demand levels, and determine efficient and economical strategies for the procurement of direct goods. Such solutions must take into account the current quotes offered by suppliers, likely future prices, projected demand and storage costs in order to make effective decisions on when and from whom to make purchases. Based on demand trends and projections, there is typically a target inventory level that a business hopes to maintain. This level is high enough to be able to meet fluctuations in demand, yet low enough that unnecessary storage costs are minimized (see Shapiro [13] for example). The focus of this paper is to provide an algorithm for purchase decision-making that strives to keep inventory close to its optimal level, while minimizing total cost.

B. Kégl and G. Lapalme (Eds.): AI 2005, LNAI 3501, pp. 133–144, 2005.

In a perfect world, the best strategy for keeping inventory as close to the optimal level as possible would be to delay ordering to the last moment. That is, if demand trends indicate that a new shipment will be needed on some particular day, it would be best to delay ordering as long as possible so that the quantity needed can be assessed with the most certainty. An accurate estimate of the optimal quantity is critical since an inventory shortage may result in lost sales, while excessive inventory could result in unnecessary storage costs. Because of the variance in demand, the quantity needed a few days from now can usually be more accurately assessed than the quantity needed several days from now. Thus by delaying ordering the expected utility of future demand levels is increased. On the other hand, one may want to order earlier if current prices are low, if there will be more selection (i.e. many quotes from which to choose), or simply to ensure timely delivery. Thus there can be incentive to bid both early and late.

In this paper, we propose a decision-theoretic algorithm that advises the buyer when and from whom to buy by looking at possible future decisions. The buyer is advised to take an action if and only if there is no present or future alternative that would yield greater overall expected utility. We consider the request-for-quote (RFQ) model where the buyer requests quotes from suppliers by specifying the quantity needed and the desired delivery date, receives quotes a short time after which specify the price and quantity that can be delivered by the specified date (if not the entire order), and has a fixed period of time to decide whether or not to accept each quote. Factors that are of concern include the projected demand for each day (or whatever time period granularity is desired), current and projected sale prices each day for each supplier, storage costs, and RFQ costs. While there might not be direct costs associated with requesting quotes, indirect costs such as the time taken to compute optimal RFQs, as well as the possibility of being neglected by suppliers if we repeatedly fail to respond to their quotes, must be considered. To compute optimal decisions, we model the problem as a Markov decision process (MDP) [12] and use dynamic programming [3, 9] to determine the optimal action at each decision point. Actions include submitting RFQs to the various suppliers and accepting/rejecting quotes. With this model, the value (i.e. expected utility) of future consequential decisions can be taken into account when determining the value of choices at current decisions. Based on this model, the subproblem of determining optimal quantities to request in an RFQ is formalized and results are presented.

The new Trading Agent Competition-Supply Chain Management game (TAC-SCM) [1, 2] now provides a vehicle for testing various techniques related to supply-chain management in a competitive environment. While the theory in this paper deals with supply chain management in general, we briefly discuss how the techniques can be implemented for our entry in the competition, NaRC.

The paper is organized as follows. In section 2 we give a formal description of the problem. In section 3, we formulate the problem as an MDP and define the dynamic programming model. The subproblem of determining optimal request quantities is presented in section 4, and results of a few experiments are given. In section 5 we discuss the TAC-SCM game and describe how the research discussed in this paper fits. Finally, in section 6 we offer a few conclusions and outline plans for future work.

2 Problem Formalization

We consider the model where the buyer wants to purchase multiple units of a single good for resale (perhaps first being assembled with other items). Let $SUP = \{sup_1, \ldots, sup_m\}$ be the set of suppliers from whom the good can be obtained. Let $d = 0, 1, \ldots n$ denote the days over the procurement period (e.g. the next fiscal year, etc.). These could instead be hours, weeks, etc., depending on the desired granularity of time. Also, let $k \in Z$ be an integer denoting the inventory on a particular day d, and let h be the holding cost per unit per day. That is, if k' units are left over at the end of the day, they are held at a cost of hk'. Also, let $uk(k, d)$ be the utility of holding k units at the start of day d. This is a function of the expected income for d, taking into consideration the expected demand on d and the expected cost of holding the leftover inventory at the end of the day. This function will be maximized with higher k during high-demand periods and lower k over low-demand periods.

Our research is placed in the context of the request-for-quote (RFQ) procurement model. At any time, the buyer can send an RFQ to various suppliers. A subset of those suppliers will then respond to the request by offering a quote which specifies the terms of the offer. Let each RFQ be a tuple $\langle sup_i, q, d_{del} \rangle$ specifying the supplier sup_i, the quantity q needed and the day d_{del} on which to deliver. Let each quote be a tuple $\langle sup_i, p, q_{del}, d_{del}, d_r \rangle$ specifying the supplier sup_i, the price p of the order, the quantity q_{del} that can be delivered on d_{del} (in case the entire order cannot be filled by that day), and the day d_r on which the quoted price will be rescinded if the buyer has not yet responded. Let c be the small cost associated with each RFQ. Payment for the order is assumed to be made when the quote is accepted.

Also, for the purposes of projecting future outcomes, assume we have three probability distribution functions that are used to predict future outcomes: the demand distribution function, the supply distribution function and the price distribution function. The demand distribution function $df(d, q)$ takes a day d and a quantity q and returns the probability of selling q units on d. The supply distribution function $sf(sup, d, d', q)$ takes a supplier sup, days d and d' and a quantity q and returns the probability that sup can (and will agree to) to deliver q units on day d' if they were ordered on day d. We assume that if the supplier does agree to this delivery, then all q units will arrive on d' with certainty. The model could, however, be extended to allow for late deliveries by using a probability measure over all possible d'. Finally, the price distribution function $pf(sup, d, d', q, p)$ takes a supplier sup, days d and d', a quantity q and a monetary amount p and returns the probability that sup will quote a price of p for q units ordered on d to be delivered on day d'. Each of these functions can be constructed by examining market history, supplier history, or by using statistical projection techniques.

The problem is to decide each day 1) which quotes that have already been obtained to accept, and 2) whether to request new quotes, and if so, how the RFQ's should be formulated. That is, we must decide on which days we will likely need new shipments, and also what the optimal quantity is. The goal is to make

decisions that maximize the overall inventory utility (i.e. keep the inventory close to optimal each day), while minimizing the total amount spent on orders over the duration of the purchase period.

3 Modeling the Problem as a Markov Decision Process

In this paper we capitalize on the idea of examining exactly what information will be known at future choice points when determining the optimal actions. For example, consider two suppliers sup_1 and sup_2. If we choose to request a quote for k units from each of them on some future day d, at the time we receive the quotes we will know the exact price being offered by each supplier. Based on this knowledge, plus the knowledge of the expected utility of not ordering at all, we can choose either to accept the cheaper quote or pass altogether. While the expected utility of any course of action on day d may not be as high as the expected utility of any action at the current decision point (i.e. current quotes), it is possible that the overall expected utility of waiting until day d to take action is higher. This is due to the fact that more information will be known on d than is known now, which will allow the decision-maker to make a more informed decision, thus increasing expected utility.

To determine the optimal quotes to accept and RFQs to submit, the problem is modeled as a Markov decision process (MDP) [12]. An MDP is a mathematical tool used to aid decision-making in complex systems. In an MDP, the possible *states* S that the decision-making agent can occupy are defined, as well as the set of *actions* A that the agent can take in each state. If action a is deterministic in state s, then the transition function maps (s, a) to a new state s'. Otherwise the action is stochastic, and the transition function maps (s, a) to states according to a probability function Pr, where $Pr(s'|s, a)$ is the probability of occupying s' given that a is performed in s. Also, some or all of the states may have an associated *reward*. The purpose of modeling a problem as an MDP is to determine a *policy* function $\pi : S \rightarrow A$, which takes any state and specifies the action such that the expected sum of the sequence of rewards is maximized. Dynamic programming is used to determine the optimal action in each state.

3.1 States

Each state s in the MDP for our problem is a tuple $\langle I, Q, C, d, k \rangle$ where

- I is the set of incoming orders. That is, I contains the orders known to be coming in on the day specified in s or on some future day. Each $i \in I$ is a tuple $\langle q, d \rangle$ where d is the day of the shipment and q is the quantity.
- Q is the set of currently open quotes.
- C is the total amount spent on purchases thus far.
- d is the day.
- k is the current inventory.

3.2 Actions

Actions consist of accepting quotes and sending RFQs. Since quote rescind times are always known (i.e. quotes are not pulled without warning), we assume that decisions on whether or not to accept a quote are delayed to the last possible moment, to allow decisions to be as informed as possible. Thus quotes are only accepted the day before they are to be rescinded. We also assume that at most one RFQ is sent to each supplier each day. This assumption is put in place merely to reduce the number of possible actions at each state, and could easily be lifted if desired. Let $req(rfq)$ represent the act of submitting a request-for-quote rfq, and let $acc(qu)$ represent the act of accepting quote qu. For a state s with quotes Q_s and day d_s, let $\{req(\langle sup, q, d_{del}\rangle) \mid sup \in SUP, q_{min} \leq q \leq q_{max}, d_s < d_{del} \leq d_n\}$ be the set possible quote requests, where q_{min} and q_{max} are the minimum and maximum quantities that can be ordered, respectively, and d_n is the final day of the procurement period. Also let the set $\{acc(\langle s, p, q, d_{del}, d_r\rangle) \mid \langle s, p, q, d_{del}, d_r\rangle \in Q_s, d_r = d_s + 1\}$ be the set of possible quote acceptances. The set A of actions is then the union of these two sets. Any subset A' of the actions in A for a state s can be performed with the restriction that at most one RFQ is submitted to each supplier. Let the set of these valid subsets for a state s be denoted by A_s.

3.3 Rewards

The value of a state in an MDP is equal to the reward for that state plus the expected rewards of future states. The optimal action at each state is then the one defined to yield the highest expected value. Our technique aims to optimize the utility of the inventory held each day, and minimize the total cost over the entire purchase period. Thus there are two types of rewards given in the MDP. To assess the reward to be assigned to each state, two utility functions are used: the inventory utility function uk and the cost utility function uc.

The inventory utility function $uk : Z \times Z \rightarrow \Re$ takes an inventory level k and a day d and returns the utility of holding k units on d. This utility is determined by measuring the ability of meeting the expected demand for day d with k units against the expected costs associated with holding the leftover units. For example, if k' is the optimal number of units to hold on d (thus maximizing uk for d), then for $k < k'$ inventory may not be high enough to meet the demand so money may be lost, and for $k > k'$ inventory may be too high and too costly to be worth holding.

As an example, let the demand function be such that either 1 or 2 units will be sold, each with 0.5 probability, on day d. Also let the sale price of each unit be 10, and the inventory holding cost be $h = 1$/unit/day. The expected net income (revenue - minus inventory cost) $E(x, d)$ for x units on day d is 0 if $x = 0$ (since no units are sold and no units are held), 10 if $x = 1$ (the one item will be sold with certainty, since the demand function says that 1 or 2 units will be sold today), and $16.5 - x$ if $x \geq 2$ (taking into account losses incurred by possible leftover inventory). The utility function uk is then a function of $E(x, d)$ (perhaps concave to indicate aversion to risk).

The cost utility function $uc : Z \rightarrow \Re$ is a monotonically decreasing function that takes a cost c and returns the utility of spending c. It is typically a concave function reflecting the risk-averseness of the decision-maker.

For each state s, the *inventory reward* is given. That is, if k is the inventory for s and d is the day, then the inventory reward for s is $uk(k, d)$. For each terminal state a *cost reward* is given, which is the utility $uc(C)$ of spending a total of C over the duration of the procurement period.

The value of each state is then a function of the expected cost reward and the expected inventory rewards for the remainder of the procurement period, given that the state is reached.

3.4 The Transition Function

The transition function specifies which states can follow from an action in a given state in the MDP. Let $T(s, a)$ be this function which takes a state $s \in S$ and action $a \in A_s$, and returns the set of states that can be occupied as a result of performing a in s. Let $Pr(s'|s, a)$ be the transition probability function, which specifies the probability of occupying state $s' \in T(s, a)$ directly after a is performed in s. These two functions are computed as follows.

Let $s = \langle I, Q, C, d, k\rangle$ be a state and $a \in A_s$ an action where a is a valid subset of requests and acceptances that can be performed in s. Then $s' = \langle I', Q', C', d', k'\rangle \in T(s, a)$ if

- I' contains the incoming orders from I, minus those offers that arrived on day d, plus new incoming orders that result from the quotes accepted in a. More formally, let $I_{old} = \{\langle q, d_{del}\rangle \mid \langle q, d_{del}\rangle \in I, d_{del} = d\}$ be the orders that came in on d, and let $I_{new} = \{\langle q, d_{del}\rangle \mid acc(\langle sup, p, q, d_{del}, d\rangle) \in a\}$ be the new incoming orders that arise as a result of accepting quotes. Then $I' = I \setminus I_{old} \cup I_{new}$.
- Q' contains the quotes from Q, minus those that were rescinded on day d, plus those that are received as a result of the requests in a. Let $Q_{old} = \{\langle sup, p, q, d_{del}, d_r\rangle \mid \langle sup, p, q, d_{del}, d_r\rangle \in Q, d_r = d'\}$ be the orders that are rescinded on d', and let $Q_{new} = \{\langle sup, p, q, d_{del}, d{+}1{+}ql\rangle \mid req(\langle sup, q, d_{del}\rangle) \in a\}$ be the quotes received in response to the requests in a, where ql is the quote length (i.e. the number of days for which the quote is valid). This could be assumed to be constant over all suppliers. Thus $Q' = Q \setminus Q_{old} \cup Q_{new}$. Note that there may be several possible values for the price p and the deliverable quantity q in the quotes in Q_{new}. The transition probability function will consider the probability of each outcome in determining the probability of the state as a whole.
- C' is the amount spent C by day d, plus the amount spent on accepted quotes in a, plus the RFQ costs. Thus $C' = C + \sum p + c_{req}$ over all $acc(\langle sup, p, q, d_{del}, d + 1\rangle) \in a$, where c_{req} is the cost of requests in a.
- k' is the starting inventory k for day d, minus the units sold t_d on d, plus those received via incoming orders in I_{new}. Thus $k' = k - t_d + \sum q$ for all $\langle q, d_{del}\rangle \in I_{new}$. Note that there may be several possible values for t_d, each

with some probability of occurring. The probability of any t_d greater than $k + \sum q$ is 0.
- $d' = d + 1$.

Let s be a state and let $T(s, a)$ contain the states that can follow from performing a in s. Then for each state $s' \in T(s, a)$, the probability $P(s'|s, a)$ of occupying s' after a is performed in s is the probability of receiving the new quotes in s' given the requests in a, multiplied by the probability of the sales realized in the transition from s to s'. Let d be the day specified in s, let Q_{new} be the set of new quotes received on day $d + 1$ (i.e. the quotes that are in s' but not in s), and let t_d be the number of units sold on day d, which is the inventory in s' minus the sum of the inventory in s and the units received (i.e. in I_{new}). Let the demand distribution function df, supply distribution function sf and price distribution function pf be as defined in section 2. Then the probability of getting the quotes in Q_{new} is

$$Prob(Q_{new}) = \prod_{qu_i \in Q_{new}} sf(sup_i, d+1, d_{del_i}, q_i) \cdot pf(sup_i, d+1, d_{del_i}, q_i, p_i) \tag{1}$$

where $qu_i = \langle sup_i, p_i, q_i, d_{del_i}, d_{r_i} \rangle$. Note that there must be a qu_i for every request in a. Unanswered or rejected requests should have a corresponding quote $qu_i = \langle sup_i, 0, 0, d_{del_i}, d_{r_i} \rangle$ in Q_{new}. Since the probability of selling t_d units on d is $df(d, t_d)$, the probability of s' occurring given that a is performed in s is

$$P(s'|s, a) = Prob(Q_{new}) \cdot df(d, t_d) \tag{2}$$

3.5 The Dynamic Programming Model

The value iteration method of dynamic programming is used to determine the optimal action at each state. Let $v : S \to \Re$ be the value function that assigns to each state its value (i.e. utility), let $\pi : S \to Q$ be the optimal policy and let $s = \langle I, Q, C, d, k \rangle$ be a state. Then

$$v(s) = \begin{cases} f_d(uk(k, d), uc(C)) & \text{if } d = d_n \\ \max\limits_{a \in A_s} \sum\limits_{s' \in T(s,a)} f_d(uk(k, d), v(s')) \cdot P(s'|s, a) & \text{otherwise} \end{cases}$$

$$\pi(s) = \begin{cases} null & \text{if } d = d_n \\ \arg\max\limits_{a \in A_s} \sum\limits_{s' \in T(s,a)} f_d(uk(k, d), v(s')) \cdot P(s'|s, a) & \text{otherwise} \end{cases} \tag{3}$$

where f_d is the function for computing the value of the state in terms of the inventory reward of the current state and the expected value of the following states, and arg is the operator that returns the maximizing a. This function may be constant or variable and can be constructed to factor in the decision maker's relative importance for optimizing either cost or inventory level.

4 Using the Model to Determine RFQ Quantities

4.1 Modeling the Subproblem

While the Markov model presented in the previous section laid a framework for all decision-making involved in optimizing target inventories, in several situations the model may be too complex to solve in a reasonable length of time. In this section we show how the model can be used to solve a more manageable piece of the puzzle, and formalize the subproblem of determining the optimal quantity to request in a given RFQ.

In this case, elements such as quoted costs are not considered, and thus decision-making does not depend on the onerous task of enumerating all outcomes for price. Instead, an *acceptance rate* is used, which is a static measurement of the likelihood any quote will be accepted based on its price. For example, if it is found that the quotes are accepted 55% of the time (or in the case when multiple quotes are solicited simultaneously for purpose of comparison, that *some* quote is accepted), then the acceptance rate is 55%. While demand typically changes each day, we assume that the acceptance rate is the average taken over the procurement period. The only dynamic factor under consideration here is the current inventory. The question is then, based on the current inventory, how many units should be requested?

The problem is stated more formally as follows: Given a current inventory level k, shipping time st (in days), daily inventory utility function $uk(k, d)$, daily demand function $df(d, k)$ indicating the probability that k units are sold on day d, and acceptance rate α, if an RFQ were to be submitted, how many units should be ordered? The MDP for this problem is then a portion of the MDP for the general problem. Each state s is a tuple $\langle I, d, k\rangle$ where

- I is the set of incoming orders
- d is the day.
- k is the current inventory.

An action is an RFQ $rfq = \langle q, d + st\rangle$ for a particular quantity q to arrive on day $d + st$. Every state has an associated reward equal to $uk(k, d)$ where k is the starting inventory on day d. The optimal action $\pi(s)$ then specifies the optimal RFQ given state s.

4.2 Testing the Performance

To assess the potential performance of using this model, the method was tested against a method from the literature that uses a Monte Carlo algorithm [13]. With this method, a reorder point r and a quantity q are randomly chosen from some distribution. Market behaviour is then simulated within the specified parameters, where an order for q units is placed each time the inventory falls below r. This process continues for several (r, q) pairs, and the optimal result is noted.

Tests were run over a 150 day procurement period, with an inventory utility function $uk(k, d) = \max\{0, 15-|15-k|\}$ (utility maximized at $k = 15$, minimized

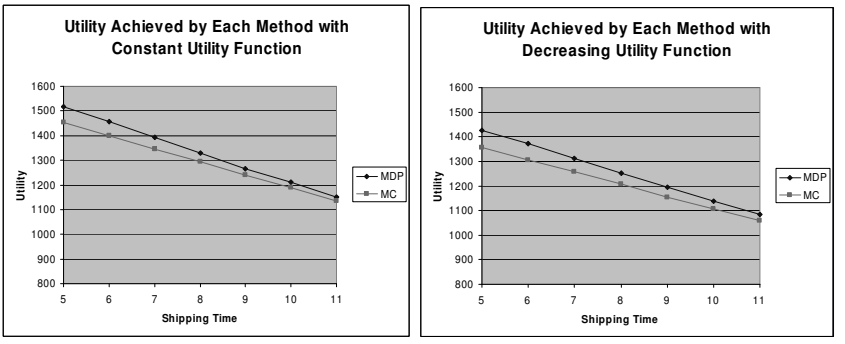

Fig. 1. (a) Utility achieved using our method (MDP) and Monte Carlo (MC) using a constant utility function over all 150 days in the procurement period, (b) Utility achieved using our method (MDP) and Monte Carlo (MC) using a decreasing utility function where 0 inventory is desired on day 150

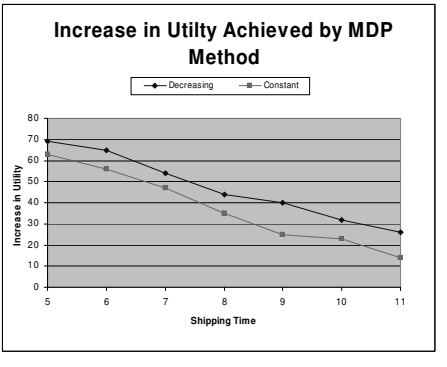

Fig. 2. The increase in utility achieved by our method over Monte Carlo for a constant and decreasing utility function

at $k = 0$, $k \geq 30$) for all d, and a quote acceptance rate of $\alpha = 50\%$. Shipping time was varied throughout the tests. A summary of the results is demonstrated in Figure 1(a).

The Monte Carlo method is quite rigid, since at any given time either q or 0 is ordered. Even though the optimal quantity and the optimal times at which to order are utilized, it still does not perform as well as our method, which adapts to the situation and determines the appropriate amount. Notice that both methods perform worse as the shipping time is lengthened because of increased uncertainty. To further demonstrate the advantages of our technique, Figure 1(b) shows results of tests where the utility function is not constant. In particular, this test utilized a decreasing utility function over time. This models the situation where no inventory is desired at the end of the procurement period.

Our method performs even better, since order quantities can be adjusted to accommodate a changing utility function, where the Monte Carlo method uses a static q and r values. Figure 2 demonstrates the overall increase realized by our method for each of the constant and dynamic utility functions.

5 The TAC-SCM Game

5.1 Game Description

The Trading Agent Competition has occurred annually since 2000. The competition was designed to encourage research in trading agent problems, and it provides a method for direct comparison of different approaches, albeit in an artificial environment. The original competition focused on acquiring a package of items in set of auctions, but in 2003 the "Supply Chain Management" (SCM) [1, 2] game was introduced. The TAC-SCM game charges the competing agent with the task of successfully managing a computer dealership: acquiring components from suppliers, assembling these components into complete PCs, and selling these PCs to a group of customers. Starting with an initial bank balance of 0 and unlimited borrowing capabilities, the agents' goal in the competition is to make the most profit. To compete successfully, agents must be quite complex and able handle different purchasing models. To win contracts with customers, agents must win a first-price sealed-bid auction. To acquire goods successfully from suppliers, agents must be able to effectively judge pricing trends. At the same time, they must also consider that supply is limited, and thus rejecting an offer could result in inability to acquire goods. Several other stochastic factors such as customer demand, customer reserve values and delivery delays must also be handled for the agent to be successful.

The procurement model of the TAC-SCM game loosely reduces to our MDP model. Each day, an agent receives quotes from suppliers based on the previous day's requests, as well as quote requests from customers. From the procurement point of view, the goal is to determine which quotes from suppliers to accept and what new quotes to request, to optimize inventory and cost. Quotes and RFQs take the same form as those described in this paper. Based on the bidding strategy used in response to customer requests (we do not focus on bidding in this paper, only procurement), the agent can judge the demand function by assessing how likely it is to sell certain quantities each day. Based on previous dealings with the various suppliers, the agent can also model the supply and price distribution functions, and build the MDP. Dynamic programming is then used to determine the optimal accepts and requests.

5.2 Our "NaRC" Agent

NaRC [7] competed in the 2004 TAC-SCM competition in New York. While we qualified for the tournament (top 24 teams out of about 35), we were eliminated in the quarter-final round (the first of three days of competition). Aspects of NaRC utilized the MDP model described in this paper. In particular, the

purchase decision-making engine modeled the sequence of subsequent purchase decisions as an MDP in order to determine the value of current quotes. While thus far untested in the TAC-SCM competition, the technique of using an MDP to compute optimal RFQs has been shown above to be quite promising. We plan to implement the method in our agent for future installments of the competition.

6 Conclusions and Future Work

In this paper we present a mathematical model for determining when to request quotes from suppliers, how to construct the RFQs, and which of the resulting quotes to accept. Decisions are made in such a way as to optimize the level of inventory each day, while lowering total cost. The problem is modeled as a Markov decision process (MDP), which allows for the computation of the utility of actions to be based on the utilities of consequential future states. Each action is considered to be a set containing quote requests and accepts. Dynamic programming is then used to determine the optimal action at each state in the MDP. The model is then used to formalize the subproblem of determining optimal request quantities, and experiments show that the technique performs better than a standard technique from the literature. The TAC-SCM game is also discussed, and the implementation details for own agent, NaRC, are briefly described.

The idea of modeling problems similar to this as an MDP has been done before. Boutilier *et al.* [4, 5], Byde [8], and Buffett and Grant [6] have previously used MDPs for auction decision-making. However our model differs from these works in two ways: 1) we consider the request-for-quote model rather than the auction model, and 2) we buy items for resale with the extra aim of maintaining a certain level of inventory, in addition to cost minimization. Other techniques have been presented by Priest *et al.* [10, 11] for purchasing items for resale; however, these works do not attempt to measure the value of current choices based on the value of consequential future decisions.

For future work, we intend to test the technique against other strategies to determine under what conditions and situations the technique performs well and not so well. Such strategies range from the more naïve where quotes are requested simply when inventories reach certain levels and the cheapest quote is immediately accepted, to the more sophisticated where massive amounts of inventories are built up (regardless of overhead costs) and intelligent selling methods are employed to maximize profit. We believe that the latter type of strategy, which was employed by several agents in the TAC-SCM game in 2003, might not yield as much profit per unit as our technique, but could surpass our technique in total profit because of the higher volume of transactions. As far the potential success of using our technique in the actual TAC-SCM game, we believe that while these high-volume agents may monopolize supply early in the game, in the long run our agent will perform better, especially in low-demand games. Only after experimentation with real-world examples as well as the TAC-SCM will these questions be answered.

References

1. R. Arunachalam, J. Eriksson, N. Finne, S. Janson, and N. Sadeh. The supply chain management game for the trading agent competition 2004. http://www.sics.se/tac/tacscm_04spec.pdf. Date accessed: Apr 8, 2004, 2004.
2. Raghu Arunachalam and Norman Sadeh. The 2003 supply chain management trading agent competition. In *Proc. International Conference on Electronic Commerce (ICEC2004)*, pages 113–120, Delft, The Netherlands, 2004.
3. R. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, NJ, 1957.
4. C. Boutilier, M. Goldszmidt, and B. Sabata. Continuous value function approximation for sequential bidding policies. In *the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 81–90, Stockholm, 1999.
5. C. Boutilier, M. Goldszmidt, and B. Sabata. Sequential auctions for the allocation of resources with complementaries. In *the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 527–534, Stockholm, 1999.
6. S. Buffett and A. Grant. A decision-theoretic algorithm for bundle purchasing in multiple open ascending price auctions. In *the Seventeenth Canadian Conference on Artificial Intelligence (AI'2004)*, pages 429–433, London, ON, Canada, 2004.
7. S. Buffett and N. Scott. An algorithm for procurement in supply chain management. In *Proc. of the Trading Agent Design and Analysis Workshop (TADA'04)*, pages 9–14, New York, NY, 2004.
8. A. Byde. A dynamic programming model for algorithm design in simultaneous auctions. In *WELCOM'01*, Heidelburg, Germany, 2001.
9. R.A. Howard. *Dynamic Programming and Markov Processes.* M.I.T. Press, Cambridge, Mass., 1960.
10. C. Preist, C. Bartolini, and A. Byde. Agent-based service composition through simultaneous negotiation in forward and reverse auctions. In *Proceedings of the 4th ACM Conference on Electronic Commerce*, pages 55–63, San Diego, California, USA, 2003.
11. C. Priest, A. Byde, C. Bartolini, and G. Piccinelli. Towards agent-based service composition through negotiation in multiple auctions. In *AISB'01 Symp. on Inf. Agents for Electronic Commerce*, 2001.
12. M.L. Puterman. *Markov Decision Processes.* Wiley, 1994.
13. J. F. Shapiro. *Modeling the Supply Chain.* Duxbury, Pacific Grove, CA, 2001.