An Algorithm for Procurement in Supply Chain Management
Buffett, Scott; Scott, N.

National Research
Council Canada
Conseil national de
recherches Canada

Canada

NRC·CNRC

*An Algorithm for Procurement in Supply Chain Management **

Buffett, S., and Scott, N.
July 2004

Canada

An Algorithm for Procurement in Supply-Chain Management

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

Nathan Scott
National Research Council Canada
Institute for Information Technology - e-Business
46 Dineen Drive
Fredericton, New Brunswick, Canada
E3B 9W4
nathan.scott@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, RFQs 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. We also discuss the implementation of our entry in the TAC-SCM game, NaRC, and demonstrate how the general technique presented can be specialized for use in our TAC-SCM agent.

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 [11] 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.

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 and 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 the 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 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 [10] and use dynamic programming [2, 7] 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.

The new Trading Agent Competition-Supply Chain Management game (TAC-SCM) [1] 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 show how the model is reduced to that of the TAC-SCM game, and demonstrate how our technique is 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. In section 4 we discuss the TAC-SCM game and describe how the research discussed in this paper fits. Finally, in section 5 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 due upon receipt of the goods.

Also, for 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 $df(d, x)$ takes a day d and an integer x and returns the probability

of selling x units on d. The supply distribution function $sf(sup, d, d', x)$ takes a supplier sup, days d and d' and an integer x and returns the probability that sup can deliver x units on day d' if they were ordered on day d. Finally, the price distribution function $pf(sup, d, d', x, y)$ takes a supplier sup, days d and d', an integer x and a monetary amount y and returns the probability that sup will quote a price of y for x 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) [10]. 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 is 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 on each day in the procurement period.

3.1 States

Each state s in the MDP 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 on their rescind days. 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 $\{req(\langle sup, q, d_{del} \rangle) \mid sup \in SUP, q_{min} \leq q \leq q_{max}, d_s < d \leq d_n\}$ be the set possible quote requests, where q_{min} and q_{max} are the minimum and maximum quantities that can be ordered, respectively, and let the set $\{acc(\langle s, p, q, d_r \rangle) \mid \langle s, p, q, d_r \rangle \in Q_s, d_r = d_s\}$ 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 two things: the utility of the inventory held each day, and 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.

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. The expected net income (revenue - minus inventory cost) $E(x, d)$ for x units on day d is 0 if $x = 0$, 10 if $x = 1$ (since the one item will be sold with certainty), 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-aversity 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 for the procurement period and the expected inventory rewards for subsequent days.

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 were rescinded, 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 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.

- $d' = d + 1$.

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 determined as follows. Let d be the day specified in s, let Q_{new} be the set of new quotes received on day $d+1$, 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, supply distribution function and price distribution function be as defined in section 2. Then the probability of getting the quotes in Q_{new} is

$$Prob(Q_{new}) =$$

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

where $qu_i = \langle sup_i, p_i, q_i, d_{del_i}, d_{r_i} \rangle$, and the probability of selling t_d units on d is $df(d, t_d)$. Thus the probability of s' occurring is $P(s'|s,a) = Prob(Q_{new}) \times df(d, t_d)$.

3.5 The Dynamic Programming Model

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

$$v(s) =$$

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

$$\pi(s) =$$

$$\arg \max_{a \in A_s} \sum_{s' \in T(s,a)} f_d(uk(k,d), v(s')) \times P(s'|s,a) \quad \text{if } d < d_n$$

where arg is the operator that returns the maximizing a, and f_d is the function for computing the value of the state in terms of the utility of the current inventory and the value of the following states. 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. THE TAC-SCM GAME

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) 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.

4.1 Game Description

The unit of time in a TAC-SCM game is a single day. Each day, all competing agents receive a variety of information and can choose to perform several different actions. These all occur in three main areas: Purchasing (from suppliers), sales (to customers), and production and delivery.

The customers send a package of RFQs each day. Each RFQ specifies a PC type, quantity, due date, reserve price and penalty amount. Each agent may then choose to bid on the RFQ. The customers choose the lowest bid it receives, as long as the bid is below the reserve price, and accepts that bid the following day. Once a bid is accepted, the agent has committed to meet the bid by the specified due date: failure to do so results in penalty fees for late or cancelled orders.

The production process takes a set of components in inventory (that are never sought individually by customers) and assembles them to create a finished product which can be sold to customers. Different PC types require different components and use different amounts of "assembly cycles". All agents have the same daily capacity for production. Once assembled, PCs can be delivered freely, at any time, with no limit on numbers. Additionally, inventory holding costs are charged for all components and finished PCs held in inventory.

Agents RFQ suppliers much like customers RFQ agents. Each day, agents may choose to send an RFQ for a specific component, quantity and due date. The difference is in the pricing. When the suppliers receive an RFQ, they respond with a price based on the quantity requested, the due date, and its projected capacity for production. Also, if the supplier cannot complete an order by the requested date, they may respond with a partial completion quote (for a lower quantity) or an earliest completion quote (at a later due date) or both.

The agent receives a response to its RFQs the following day. When an agent receives an offer from a supplier, it must choose to accept or decline the offer that same day. While there is no immediate cost associated with making an RFQ, there is a hidden cost that results from a simple "reputation" model that each supplier uses for the agents. The model uses the ratio of the quantity requested by an agent to the quantity actually ordered, over the entire game, to decide which RFQs to process first. Being processed later than other agents can lead to higher prices and less reliable dates.

There is additional information provided such as market and price reports that can help gaage demand levels, but the main focus of day-to-day decision making is: Bid on customers, assemble and deliver PCs, send suppliers RFQs, and choose supplier offers to accept.

4.2 Our "NaRC" Agent

NaRC is intended to use the Markov Decision Process model described in this paper for dealing with the supplier end of the SCM game. The customer sales portion of our agent sets the ground rules for this aspect. It is in charge of gauging the demand levels and setting the utility functions. In other words, the sales portion tells the purchasing portion what inventory levels it wished it had based on how the game has been going (the inventory utility function). The sales portion also decides what prices should be considered to be "too high" based on market conditions (the cost utility function). The purchaser then uses the MDP model to fulfill those requests as best it can. Whatever inventory it does manage to acquire is used by the sales portion when it comes down to actually bidding on customers.

The description of a state in the MDP provided in 3.1 maps neatly to the SCM game with a few refinements. First of all, the SCM is much more complex than the single-unit model that we describe. There are several different computers sold in the market, each of which composed of multiple components. We reduce the SCM problem to our model as follows. Each day, several subsets of the quotes received are examined to determine the optimal subset to accept. The value of each set is assessed by determining the optimal allocation of components that would be received as a result, as well as those already in inventory, to the various computers that can be built. If the optimal allocation will give us x units of computer type A with assembly day d (i.e. the day that the last of the necessary components will arrive), then an MDP is built for A with an initial incoming order of x units on day d. The value of the current state of this MDP is our value for A. This is done for every computer given this allocation, giving the total value for the allocation.

Another difference is that the set Q of open quotes will not consist of a group of overlapping offers that expire at different times. Quotes given by suppliers in the SCM game always arrive the day after the RFQ, and always close on the same day. Thus decisions on open quotes must be made immediately, and so considering accepting actions need only be done on days immediately following a day we chose an RFQ action.

Market conditions change from game to game, and even mid-game, so the inventory utility function $uk(k, d)$ will need to be dynamic. The utility of an inventory vector k on day d is based on comparing it to an estimated optimal inventory vector k' for day d. The optimal inventory is estimated based on the market conditions and production capacity. The cost utility function $uc(C)$ could be based on the base prices for each component (which are known at the beginning of the game and do not change) or could be dynamic, based on the average prices for the component for everyone during that game.

Section 3.4 describes the transition function, which uses three separate distributions in its calculations: the demand distribution function, supply distribution function and price distribution function. These are modeled by the agent each game based on the actual history of the game up to the decision point. The demand distribution function represents the expected "sales rate" for each component. The sales rate for future days is based on our performance history, as well as any confirmed orders that we have in the near future. The supply and price distribution functions will also be based on past history with suppliers. The formula for pricing is given in the TAC-SCM documentation, and so will be easy to predict if we can accurately estimate the supplier's capacity. This estimation will also provide us with the key to the supply distribution function.

5. CONLUSIONS 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 TAC-SCM game is also discussed, and the implementation of technique for own agent, NaRC, is described.

The idea of modeling problems similar to this as an MDP has been done before. Boutilier *et al.* [3, 4], Byde [6], and Buffett and Grant [5] 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.* [8, 9] 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 exceed 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.

6. 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] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[3] 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.

[4] 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.

[5] 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*, London, ON, Canada, 2004.

[6] A. Byde. A dynamic programming model for algorithm design in simultaneous auctions. In *WELCOM'01*, Heidelburg, Germany, 2001.

[7] R.A. Howard. *Dynamic Programming and Markov Processes*. M.I.T. Press, Cambridge, Mass., 1960.

[8] 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.

[9] 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.

[10] M.L. Puterman. *Markov Decision Processes*. Wiley, 1994.

[11] J. F. Shapiro. *Modeling the Supply Chain*. Duxbury, Pacific Grove, CA, 2001.