International Journal of Computer and Communication Technology

Volume 8 | Issue 1 Article 6

January 2017

e-Procurement and Purchase Algorithm for Supply Chain Management

Himanshu Sekhar Moharana
Raajdhani Engineering College, Bhubaneswar, Odisha, India, moharana_himanshu@rediffmail.com

J. S. Murty
RRL, Bhubaneswar, Odisha, India, jsmurty@gmail.com

D.K Sahoo
Raajdhani Engineering College, Bhubaneswar, Odisha, India, dksahoo@gmail.com

K. Khuntia
Raajdhani Engineering College, Bhubaneswar, Odisha, India, kkhuntia@gmail.com

e-Procurement and Purchase Algorithm for Supply Chain Management

Himanshu S. Moharana[1], J. S. Murty[2], D. K. Sahoo[3] & K. Khuntia[4]

[1,3&4]Raajdhani Engineering College, Bhubaneswar, Odisha, India
[2]RRL, Bhubaneswar, Odisha, India
E-mail: moharana_himanshu@rediffmail.com

Abstract **-** A technique is developed 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, Request 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, 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 Markov Decision Process. A mathematical algorithm for purchasing can also be developed which is suitable for manufacturing companies to solve the supply chain management problems.

Keywords **-** *Supply-chain management, Markov decision process, mathematical algorithm for purchasing.*

I. INTRODUCTION

With the advent of 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 ordered 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. 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, or simply to ensure timely delivery. Thus there can be incentive to bid both early and late. 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 period of time to decide whether or not to accept each quote. Factors that are of concern include the projected demand for each day, 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 and use dynamic programming 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.

II. PROBLEM FORMULATION

Consider the model where the buyer wants to purchase multiple units of a single good for resale being assembled with other items. Let SUP = fsup1: be the set of suppliers from whom the good can be obtained. Let d = 0; 1: 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 span of time. Also, let k 2 Z is an integer denoting the inventory on a particular day d, and let h be the holding cost per unit per day. That is, if k0 units are left over at the end of the day, they are held at a cost of hk0. 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 .The same is placed in the context of the request-for-quote 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 ordering a quote which specifies the terms of the order. Let each RFQ be a tuple hsupi; q; ddeli specifying the supplier supi, the quantity q needed and the day ddel on which to deliver. Let each quote be a tuple hsupi; p; qdel; ddel; dri specifying the supplier supi, the price p of the order, the quantity qdel that can be delivered on ddel (in case the entire order cannot be filled by that day), and the day dr 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; d0; x) takes a supplier sup, days d and d0 and an integer x and returns the probability that sup can deliver x units on day d0 if they were ordered on day d. Finally, the price distribution function pf(sup; d; d0; x; y) takes a supplier sup, days d and d0, 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 d0. 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.

III. MODELING OF THE PROBLEM

We bear 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 sup1 and sup2. 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 ordered 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. 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 s0. Otherwise the action is stochastic, and the transition function maps (s; a) to states according to a probability function Pr, where Pr (s0js; a) is the probability of occupying s0 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 fi S! 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.

A. *State*

Each state s in the MDP is a tuple hI; Q; C; d; ki 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 2 I is a tuple hq; di 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.

B. *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.The set A of actions is then the union of these two sets. Any subset A0 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 As.

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 _ Z ! < 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 k0 is the optimal number of units to hold on d (thus maximizing uk for d), then for k < k0 inventory may not be high enough to meet the demand so money may be lost, and for k > k0 inventory may be too high and too costly to be worth holding.

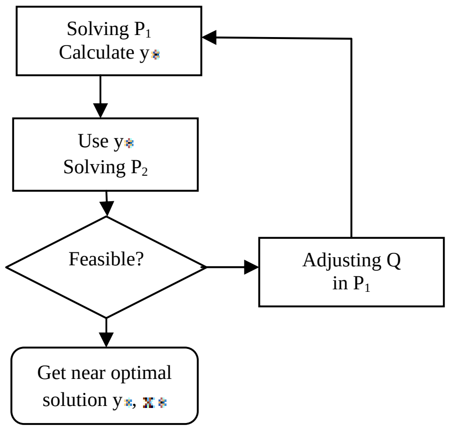
IV. NaRC AGENT

NaRC (J. F. Shapiro, 2001) is intended to use the Markov Decision Process model 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 full 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 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. 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 orders 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 midgame, 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 k0 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.

V. PURCHASE ALGORITHM

We also derive the algorithm for the original problem. However the solution derived through this algorithm is an approximation at best. Take the original problem, P. The truly optimal solution has to consider the revenue and cost parts in the objective function simultaneously, not sequentially as in our partitioning approach: the optimal amount of production, i.e., y1, should depend on not only the revenue, but also the material cost side. Thus, partitioning the whole problem into the revenue and the cost related ones may not be as accurate as the holistic method. Nevertheless, we think our approach a good approximation: it should be true, before proceeding to the numerical examples; we elaborate more on our numerical analysis technique solving $P_{1.}$ First, we employ the most widely used method i.e. Newton-Raphson method, when solving the nonlinear equations. In order to simplify the solution process, when deriving the KKT conditions for P_1, we implicitly impose $\sum_{l=1}^{L} h_l y_l = Q$. Imposing this enables us to reach the near optimal solution much faster. This method should not distort the solution significantly as long as Q is a valuable resource so that the firm tries to utilize it as much as possible.



Once we solve $P_{1,}$ the Rest of the algorithm progresses straight forward since P_2 is just an_ordinary LP problem.

We recapitulate the entire solution procedure as follows.

Step 1. Partition P into two, P_1 and $P_{2.}$

Step 2. Use the Newton-Raphson method to solve P_1 and calculate y^*_l.

Step 3. With y^*_l, solve $P_{2.}$ If P_2 is infeasible, go to step4 otherwise go to step5.

Step 4. Adjust Q. Reduce it by a predetermined magnitude. Go to step2.

Step 5. Calculate x*.

Step 6. Report an optimal solution(y^*_l, x^*).

If shows how we can utilize the mathematical algorithm laid out. In order to demonstrate the capability of the algorithm, we first present an example with multiple products, raw materials, and suppliers. Then we can focus on the practical applicability to draw managerial insights. Using the mathematical algorithm developed, we can concentrate on the following relationships-

(a) How the optimal value of the objective function changes as the manufacturer's capacity, Q changes.

(b) Manufacturing company's procurement amount from each supplier given a particular capacity, Q .

(c) Total material procurement cost of each product.

(d) Production quantity of each product given a particular Q.

(e) How the changes in a product's demand uncertainty affects the other products production quantities.

(f) How the change in a supplier capacity affects the manufacturer's procurement cost from other suppliers.

VI. CONCLUSIONS

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, 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. We can develop a mathematical algorithm to solve a supply chain management faced by a manufacturing company that assembles and sells multiple products using materials from several suppliers. In order to show the utility of the algorithm, we can present numerical examples using a set of parameters and data. It indicates that the manufacturing company has to reach an optimal supply decision by taking into account such key factors as its production capacity and under stocking and over stocking costs, market demand uncertainty,, supply costs, and suppliers' capacities. The manufacturer's procurement is decision on not only its own capacity but also its suppliers'. We also observed there exist tradeoffs between products as their demand uncertainties change unequally: a product's demand uncertainty has an adverse effects on its optimal production amount. It is not just the manufacturer who has to pay close attention to its supply chain partners (suppliers). Each supplier also has to consider the manufacturer's capacity since that could determine which supplies the manufacturer procures from which supplier. We can state in order to optimize the supply chain performance, decisions made by the manufacturing company and its suppliers need to be integrated fully. Developing a more effective algorithm to alleviate the potential problem is a definite improvement, although the specific context is more relevant to the case of a single decision period.

REFERENCES

[1] R. Bellman. Dynamic Programming. Princeton University Press, Princeton, NJ, 1957.

[2] 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.

[3] Krishnamurthy, S. 2002. E-Commerce Management: Text and Cases. Mason, Ohio, South-western

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

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

❑❑❑