INTERNATIONAL JOURNAL OF INTELLIGENT CONTROL AND SYSTEMS
VOL. 10, NO. 4, DECEMBER 2005, 286-295

Effective Leadtime-Cost Tradeoff in Supply Chain Management

ARTHUR HSU and DANIEL D. ZENG, *Member, IEEE*

***Abstract*— Electronic commerce has significant impact on enterprise operations. Businesses are eager to tap into new opportunities such as the proliferation of alternatives in the electronic marketplace, real-time access to supply information including both product price and leadtime, and an efficient inter-business electronic collaboration infrastructure. Yet, to gain competitive advantages from these opportunities, business decision makers must focus on their supply chain configurations and operational flexibility. In this paper, we develop a supply chain model based on a conjunctive/disjunctive network. This model is capable of modeling a wide range of supply chain activities including: transportation mode selection, production rate decision, assembly/subassembly, supplier selection and subcontracting. We also present an efficient solution procedure that identifies the entire efficient frontier between leadtime and cost in this supply chain model.**

***Index Terms*— supply chain management, electronic commerce, time-cost tradeoff, conjunctive/disjunctive network.**

1. INTRODUCTION

THE electronic commerce revolution promises to reshape the way in which enterprises conduct business. Products and services become increasingly easy to locate in the electronic marketplace. Consumer-to-business and business-to-business relationships are being redefined. Business transactions are being routinely conducted through electronic means. Businesses themselves are also being reorganized to achieve competitiveness in a new digital economy.

The significant impact of electronic commerce on various business functional areas such as information systems, payment and banking, and marketing has attracted much attention from business and research communities [1]. We argue that electronic commerce is also having a significant impact on enterprise operations in areas such as supplier management, leadtime reduction, inventory, and distribution channel management.

Several operational challenges arise from the electronic commerce environment. Firstly, with customers having increasing expectations for built-to-order, customized products and services, how can a firm configure its supply chain to meet these needs? Secondly, is it possible to develop timely, high-quality operational plans given the explosion in the number of suppliers and dynamic nature of the supplier base? Thirdly, with the increased importance of leadtime as an operational performance measure, how can the tradeoff between cost and time be analyzed and exploited effectively? Lastly, how can an enterprise exploit the increased operational flexibility promised by electronic commerce, and pass this flexibility to its customers?

In this paper, we develop a supply chain model that will provide an enterprise with an analytical framework to help meet these challenges. This model is scalable for organizations of various sizes and can be potentially applied in other business and engineering application domains such as project scheduling. Based on this model, we also develop a computational procedure that explicitly analyzes the relationship between two fundamental performance measures of enterprise operations: cost and leadtime. This solution procedure is computationally efficient and capable of analyzing large-scale supply chain networks.

The rest of the paper is structured as follows. Section 2 discusses the related work on value chain modeling and time cost tradeoffs. Section 3 presents our model of supply chains, called the Leadtime-Cost Tradeoff (LCT) model. This model is capable of modeling a wide range of supply chain activities that are measured by both the time and cost. Section 4 discusses several computational methods to compute the tradeoff function or efficient frontier between leadtime and cost in the LCT model. A detailed numerical example is provided to illustrate the LCT modeling constructs and related computational procedure. We discuss various supply chain analyses and managerial insights made possible by the LCT model in Section 5 and conclude the paper in Section 6 by discussing possible future research.

2. RELATED LITERATURE

The literature contains many qualitative accounts and virtually no quantitative models. Rosenfield et al. discuss the efficient frontier between cost and service/leadtime and how firms align themselves along the frontier [2]. They suggest what the shape of the frontier might look like and even suggest how to find the optimal service level given the cost-service tradeoff but do not discuss how the tradeoff curve can be determined.

Narus and Anderson describe the benefits of an auxiliary support system, essentially a parallel supply chain network that is capable of delivering emergency or one-of-a-kind products

Manuscript received September10, 2005 revised October 21,2005. This work was supported in part by a grant for open research projects (ORP-0303) from the Key Lab of Complex Systems and Intelligence Science, Chinese Academy of Sciences; and a grant (60573078) from the National Science Foundation of China.

A. Hsu is with the United Technologies Research Center, East Hartford, CT 06108 USA (email: hsu@utrc.utc.com)

D. Zeng is with the Management Information Systems Department at the University of Arizona, Tucson, AZ 85721 USA (email: zeng@eller.arizona.edu)

with low leadtime but high cost [3]. The leadtime versus cost issue is prominent in the literature on economic value chain models such as has been popularized by Porter [4].

On the quantitative side, our problem differs from the bicriteria network flow problem that is studied by Srinivasan and Thompson [5], Lee and Pulat [6], [7], and De et al. [8]. Our problem is not a network flow problem and, hence, the constraints are completely different. Also, whereas the two objectives in regular bicriteria problems are computed as the sums of activity levels weighted by two different sets of objective coefficients, one of our objectives, namely cost, is computed in this way, but leadtime is computed based not on activity levels but instead on the length of paths in the network.

Evidently, other researchers have contemplated the same tradeoff issues that we are addressing in this paper. For example, Arntzen et al. recognize that the true time measure of interest is leadtime (or *cycle time*) [9]. However, in their words, "because including cycle time directly in an optimization model complicates things more than warranted here ..., we adopted another measure of time—*weighted activity time*." In other words, they opt to measure time not in terms of cycle time, but instead with the same bicriteria objective as that in [5].

We believe that our work fills a need for a quantitative analysis of the leadtime-cost tradeoff given the importance of such a tradeoff as described in the qualitative accounts, and the lack of existing methods.

3. The Leadtime-Cost Tradeoff (LCT) Model

In this section, we present a model called the Leadtime-Cost Tradeoff (LCT) model that can effectively coordinate supply chain activities with respect to both time and cost measures.

3.1. Model Description and Assumptions

The requirements of the problem under study can be mapped to a simple network or graph representation. We use a directed acyclic graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ where $\mathcal{N}$ is the set of nodes and $\mathcal{A}$ is the set of directed arcs. The model follows an activity-on-arc (also known as activity-on-arrow) representation where each arc represents an activity in the supply chain. Conceptually, the set of arcs $\mathcal{A}$ represents the entire set of possible activities. For instance, each source of each raw material that may be considered in the solution domain constitutes an arc in $\mathcal{A}$.

The direction on each arc implies a precedence relationship such that the activity corresponding to a given arc may begin only after all activities upstream of the tail node of the arc are completed. Thus, each node represents an *event* that implies that all prior activities have been completed. We let $\tau(i)$ denote the tail node (predecessor node) of arc i. Also, we let $\mathcal{I}(n)$ denote the set of incoming arcs incident to a node n. In other words, the head node (successor node) of every arc in $\mathcal{I}(n)$ is n.

Nodes are classified into three types. The subset of nodes called *leaf nodes* denoted by $\mathcal{L}$ consists of all nodes n that have no incident arcs ($|\mathcal{I}(n)| = 0$). The other two subsets of nodes are called *conjunction nodes* and *disjunction nodes* and are denoted by $\mathcal{C}$ and $\mathcal{D}$ respectively. The distinction between conjunction and disjunction nodes will be made shortly. Collectively, the three subsets of nodes partition $\mathcal{N}$

$$\mathcal{C} \cup \mathcal{D} \cup \mathcal{L} = \mathcal{N}$$

In a general sense, a solution to the problem of satisfying a customer's order consists of a set of activities $\mathcal{A}'$ that is a subset of the entire set of possible activities $\mathcal{A}$. Together with $\mathcal{N}'$, which is the set of nodes in the graph $\mathcal{G}$ induced by $\mathcal{A}'$, solving the problem involves instantiating a network $\mathcal{G}' = (\mathcal{N}', \mathcal{A}')$ which is a subset of $\mathcal{G}$. In general, not every subset is feasible. Specifically, there is one special node, which we call the *root node* and denote by r, which represents the event associated with the successful completion of a customer's order. Clearly, a feasible solution must be such that $r \in \mathcal{N}'$ or else the supply chain network $\mathcal{G}$ would not contain the activities necessary to fulfill the customer's order. The conditions that must be satisfied in order for an instantiated network to be feasible are given below.

Definition 1: A *feasible network* is any subset $\mathcal{G}' = (\mathcal{N}', \mathcal{A}'), \mathcal{N}' \subseteq \mathcal{N}$, $\mathcal{A}' \subseteq \mathcal{A}$ of the graph G that satisfies the following properties:

1) The root node must be in the subset.
$$r \in \mathcal{N}' \tag{1a}$$
2) All arcs incident to every conjunction node in $\mathcal{N}'$ must be in $\mathcal{A}'$.
$$\mathcal{I}(n) \subseteq \mathcal{A}', \forall\ n \in \mathcal{C} \cap \mathcal{N}' \tag{1b}$$
3) Exactly one arc incident to every disjunction node in $\mathcal{N}'$ must be in $\mathcal{A}'$.
$$|\mathcal{I}(n) \cap \mathcal{A}'| = 1, \forall\ n \in \mathcal{D} \cap \mathcal{N}' \tag{1c}$$
4) The tail node of every arc in $\mathcal{A}'$ must be in $\mathcal{N}'$.
$$\tau(i) \in \mathcal{N}', \forall\ i \in \mathcal{A}' \tag{1d}$$

Again, the purpose of expressing the conditions above is that in order to find a feasible solution to the problem, we must instantiate a network that satisfies the customer's order. Condition (1a), which states that the root node is in the subset $\mathcal{N}'$, corresponds to the completion of the delivery of the final product in the supply chain. This, by itself, is not sufficient. We must ensure that the activities leading up to this event have been instantiated. For example, if r is a conjunction node with two incident arcs, in order for the network to be feasible, certainly those two activities must be in $\mathcal{A}'$. Condition (1d) states that if some activity i is instantiated in the subset of activities $\mathcal{A}'$, then its tail node $\tau(i)$, which represents completion of all predecessor activities, must be in $\mathcal{N}'$. Thus the conditions ensure that not only is r in the instantiated network, but also that activities that lead up the successful completion of the order have been performed.

Conditions (1b) and (1c) distinguish a conjunction node from a disjunction node. For a conjunction node n to be in the subset of nodes $\mathcal{N}'$ of a valid configuration, *all* arcs incident to n must be in $\mathcal{A}'$ whereas for a disjunction node, exactly *one* incident arc must be in $\mathcal{A}'$. A disjunction node n models the selection decision since if $n \in \mathcal{N}'$, then condition (1c) ensures that there is exactly one arc in $\mathcal{I}(n)$ in $\mathcal{A}'$. This

arc corresponds to the selected activity. The conjunction node is required for coordination—when more than one upstream activity (e.g. receiving all components) must be completed before a downstream activity (e.g. assembly) can begin, we must use a conjunction node.

3.2. Model Objective

In the previous section, we laid out the structure of the model including the parameters. The variables in the model essentially correspond to a binary choice for each activity—either we instantiate an arc (i.e. we perform an activity) or we do not. We now discuss the objectives of the problem.

Every activity $i \in \mathcal{A}$ has two parameters: activity time t_i and cost c_i. The activity time t_i represents the time required to complete activity i after all predecessor activities have been completed and before all successor activities may begin. The cost c_i represents the cost incurred by activity i to deliver the final product given that activity i is in the set $\mathcal{A}'$ of activities. Note that c_i "bundles" all costs in the activity i required to satisfy the order for the final product. For example, if an activity i corresponds to machining a part, and each final product requires 4 such parts, and the order consists of 10 final products, then c_i would consist of the costs for machining 40 parts. In cases when an arc may occur in two or more paths leading to a same conjunction node, a proper assignment of cost might not be possible.

We define the two principal performance measures of a supply chain:

Definition 2: The *leadtime* of a supply chain configuration $\mathcal{G}' = (\mathcal{N}', \mathcal{A}')$ is the length of the *longest* of the directed paths from a node n to the root node r among all nodes $n \in \mathcal{N}'$ in the configuration where the length of arc i is the activity time t_i.

Definition 3: The *cost* of a supply chain configuration $\mathcal{G}' = (\mathcal{N}', \mathcal{A}')$ is sum of the costs of all activities in the configuration $\sum_{i \in \mathcal{A}'} c_i$.

In Definition 2, the leadtime is the longest path in the configuration (similar to the notion of the critical path in CPM [10]) such that it represents the least possible time between when an order is placed and when the set of activities in the configurations can be feasibly arranged to deliver the final product.

Our goal is to identify the configurations (solutions) that attain a given leadtime at minimal total cost or, equivalently, configurations that attain a given total cost with minimal leadtime.

It should be noted that we are seeking a description of the tradeoff between leadtime and cost in a functional form, and are not attempting to find the optimal tradeoff point. Indeed, as suggested by Rosenfield et al., the knowledge of the market's demand curve is required and then the optimal leadtime may be determined by maximizing the difference between the leadtime-revenue function and the leadtime-cost function [2].

3.3. Modeling of Supply Chain Activities

We demonstrate that the network-based LCT model can represent a wide range of supply chain activities. Consider the network in Fig. 1. Observe that the root node in this case

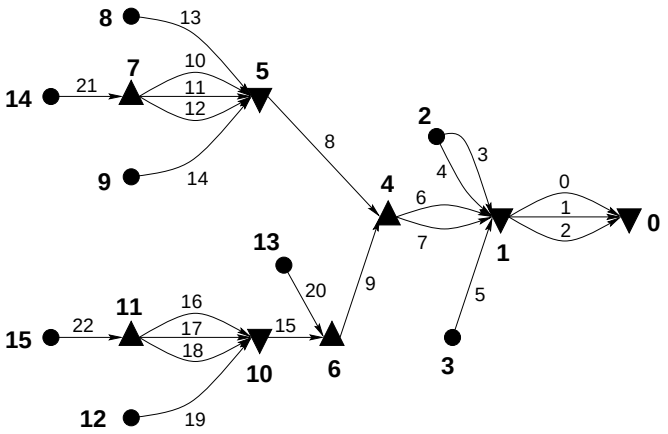

Fig. 1. An Example of a Supply Chain.

is node 0 and that there is directed path from every node to the root node. There are four conjunction nodes (4, 6, 7 and 11) indicated by a ▲ symbol and four disjunction nodes (0, 1, 5 and 10) indicated by a ▼ symbol. Since there is only one arc incident to nodes 7 and 11, they could be modeled as disjunction nodes equivalently. The other nodes indicated by a ● symbol are leaf nodes, each of which may be modeled as either a conjunction or disjunction node equivalently.

Using the hypothetical supply chain in Fig. 1, we will develop an example that gives meaning to the arcs and nodes by demonstrating how supply chain activities can be modeled. We will later refer to this example to illustrate insights gained from solving the model.

- *Transportation mode selection*. In the context of the leadtime versus cost focus of this paper, the choice of which transportation mode to select to transport a product from one location to another will be based on the mode's leadtime and cost parameters. This is a common decision faced by logistics managers. We assume that the parameters for all options are known and deterministic. For example, we may have the three options for sending a product from a particular origin to destination: via the postal service at cost 5 with a leadtime of 4 days, a second-day air service with cost 17 and leadtime of 2 days, or an overnight delivery service with cost 20 and leadtime of 1 day. In order to represent this decision in our model, we use three arcs, one for each mode, leading from one node to a disjunction node. The first node represents the

event that the product is ready to be shipped from the origin. The disjunction node represents the completed delivery of the product to the destination. We use a disjunction node since only *one* mode is required.

For example, in Fig. 1, there are four instances of transportation mode selection. Instance 1 is between nodes 1 and 0 with three modes represented by arcs 0, 1 and 2. Similar instances are between nodes 7 and 5 with arcs 10, 11 and 12 and between nodes 11 and 10 with arcs 16, 17 and 18. One other case is with the two arcs 3 and 4 between nodes 2 and 1.

• *Assembly/subassembly*. An assembly or subassembly operation in a supply chain typically requires more than one component to be ready and available before the operation can begin. In this case, we use a conjunction node because it requires that all arcs representing the delivery of the components be part of the configuration before allowing the assembly to take place. In our example, node 6 represents a subassembly which requires two components—one via arc 15 and another via arc 20. Also, node 4 represents an assembly operation, also with two components whose delivery is indicated by arcs 8 and 9. The distinction between assembly and subassembly is that subassembly operations are generally located farther upstream in the supply chain.

• *Production rate decision*. Production rate decisions correspond to the choice between using faster facilities or more expensive, high-capacity machines versus slower but cheaper facilities or machines. We model these decisions similar to the transportation mode selection. Again, we use an arc for every possible production rate option leading to a disjunction node. For example, node 4 represents the event that the two components are ready for assembly, while arcs 6 and 7 represent two possible options of production activity. One of them may be faster but more expensive, thus the disjunction node 1 needs to choose only one of them (among other options).

• *Supplier selection*. In procuring raw materials from a supplier, cost and leadtime are crucial issues. In our example, there are three sets of supplier selection decisions. For one raw material, the leaf nodes 8, 9 and 14 represent the suppliers. The parameters of arcs 13 and 14 represent the delivered leadtime and delivered cost for the suppliers at nodes 8 and 9 respectively. For the supplier at node 14, the leadtime and cost for the undelivered material is represented by arc 21 and the delivery leadtime and cost is represented by the three transportation mode options in arcs 10, 11 and 12. In terms of "free on board" (f.o.b.) pricing in logistics, the suppliers at node 8 and 9 offer f.o.b. destination pricing while the supplier at node 14 offers f.o.b. factory pricing.

For a second raw material, node 10 represents the supplier selection between the two suppliers at nodes 12 and 15. A third raw material at node 13 does not require a selection decision since there is only one supplier.

• *Subcontracting decision*. This decision is a generalization of the "make or buy" decision since it encompasses the "buy from which subcontractor" decision. In our example, the disjunction node 1 represents a decision among five possibilities. By selecting arcs 6 or 7, we are deciding on "in-house" production of the product, with the two arcs modeling the two production rates as discussed earlier. By selecting arcs 3 or 4, we are deciding to subcontract the product to the subcontractor at node 2, with the two arcs modeling two transportation mode decisions. By selecting arc 5, we are deciding on a second subcontractor at node 3.

A final remark about the modeling capability of our LCT model is that two well-known network models are special cases of the LCT model. The first is the *extended bill of materials* (BOM) in materials requirements planning (MRP) which is essentially a LCT model where all nodes are conjunction nodes. The BOM model represents the product structure with the nodes representing assembly or subassembly operations. Without disjunction nodes, there are no decisions to be made about which activities to engage in; however, the problem is not trivial since the issue remains of *when* activities should start to determine the production schedule.

A second special case of the LCT model arises in the Critical Path Method (CPM) in project management. The underlying network representation of a CPM or PERT chart is similar to the LCT model only that there are only conjunction nodes since all activities must be completed. The nodes establish precedence relationships as in the LCT model in that all activities prior to a node must be completed before activities after the node may begin. Like the LCT model and the BOM, the network has a single terminal node (similar to the root node). In project management, both cost and leadtime are essential issues and time costing may be modeled using disjunction nodes in the LCT model with "parallel" incident arcs (all of which originate from one node and terminating at the disjunction node). Each arc would have parameters representing the activity time and cost of different options similar to the production rate decisions in supply chain management. The issue of subcontracting, which involves a more general use of the disjunction node, is typically not embedded in CPM techniques. Since subcontracting is a real consideration in project management, we propose that the LCT model can be applied to analyze subcontracting options in project management as well as in supply chain management.

As stated in the introduction, the LCT model does not explicitly consider inventory levels. However, that is not to suggest that the model cannot be applied whenever there is inventory buffer in the supply chain. Indeed, if a particular component is held in inventory at some location, then that component is considered available at the location with zero leadtime. In other words, a location with inventory is a leaf node in the model. On the other hand, if the stocking location has zero inventory, then the supply chain $\mathcal{G}$ would include those activities preceding the stocking location required to replenish the inventory. See [11] for detailed technical discussions.

4. Solution Method: Computing the Efficient Frontier

We now develop computational methods that explicitly compute the *tradeoff function* or the *efficient frontier* between leadtime and cost in the LCT model.

4.1. Enumeration and the Number of Feasible Solutions

We will explore a possible approach for finding the tradeoff function that enumerates all feasible solutions that satisfy the conditions for a valid configuration in (1).

Definition 4: An *efficient solution* is a valid configuration whose leadtime and total cost is not dominated by any other valid configuration. In other words, an efficient solution is such that there is no other valid configuration with both lower leadtime and total cost.

We begin by addressing the question of how many valid configurations or feasible solutions there are for a given supply chain $\mathcal{G} = (\mathcal{N}, \mathcal{A})$.

Let us define $S(n)$ to be number of combinations such that all nodes and arcs "upstream" of n satisfy conditions (1b)–(1d). By "upstream of n," we mean a node or arc which has a directed path to n. We refer to $S(n)$ as the number of valid upstream combinations (VUC's).

We first give the expressions for $S(n)$ and then explain them afterwards:

If $n \in \mathcal{L}$, then

$$S(n) = 1 \tag{2a}$$

Otherwise, if $n \in \mathcal{C}$ then

$$S(n) = \prod_{i \in \mathcal{I}(n)} S(\tau(i)) \tag{2b}$$

and for $n \in \mathcal{D}$,

$$S(n) = \sum_{i \in \mathcal{I}(n)} S(\tau(i)) \tag{2c}$$

The total number of valid configurations in the supply chain is $S(r)$ where r is the index of the root node. In the example in Fig. 1, there are a total of 129 valid configurations, which is easily verified in the table below showing $S(n)$ for all non-leaf nodes:

Node n	11	10	7	6	5	4	1	0
$S(n)$	1	4	1	4	5	20	43	129

Eq. (2a) defines $S(n)$ for each leaf node. The only valid upstream combination is the leaf node itself hence $S(n) = 1$ in this case. We explain Eq. (2c) using the example in Fig. 1. Here, node 11 has one VUC consisting of arc 22 thus $S(11) = 1$. Node 10 has four VUC's consisting of: arcs 22 and 16, arcs 22 and 17, arcs 22 and 18, and arc 19. Each incoming arc to the disjunction node corresponds with a set of upstream configurations. Since the disjunction node requires only one of its incident arcs to be in the configuration, the set of VUC's at the disjunction node is the union of all the VUC's of its incident arcs hence the number of VUC's is the sum of the number of VUC's of its incident arcs' tail node. In this case, given the four arcs incident to node 10, $S(10) = S(11) + S(11) + S(11) + S(12) = 4$.

We explain Eq. (2b) using node 4 as an example. The VUC's of node 4 are the five VUC's of node 5 *crossed with* the four VUC's of node 6. For instance, one VUC of node 4 is the union of the set of arcs 21 and 11 (one VUC of node 5) with the set of arcs 19, 15 and 20 (one VUC of node 6). There are $4 \times 5 = 20$ distinct combinations and so $S(4) = 20$.

Note that the expression for $S(n)$ is dependent only on the topology of $\mathcal{G}$ and not on the cost and leadtime parameters. In fact, for each of the $S(r)$ supply chain configurations, there exists a set of parameters $(t_i, i \in \mathcal{A})$ and $(c_n, n \in \mathcal{N})$ where the configuration is efficient. Furthermore, it should be noted that $S(r)$ does not count valid but not minimal configurations where: (i) $\mathcal{N}'$ contains a non-root node n with no outgoing arc in $\mathcal{A}'$, and (ii) $\mathcal{A}'$ contains an arc whose head node is not in $\mathcal{N}'$ It can be shown that valid configurations having either property (i) or (ii) are always dominated by some other valid configuration. Thus, $S(r)$ represents the number of *potentially efficient* supply chain configurations, which in general is considerably less than the total number of valid configurations satisfying (1).

When the number of potentially efficient configurations is relatively small, it would seem reasonable to propose a solution approach where we enumerate all feasible solutions that satisfy conditions (1), then determine the performance measures of leadtime and total cost of the activities in the configuration, and then find the efficient frontier. To illustrate, consider Fig. 2 which shows the same supply chain as Fig. 1 only that each arc is shown with its activity time and cost parameters (t_i, c_i). Using these parameters, we can determine

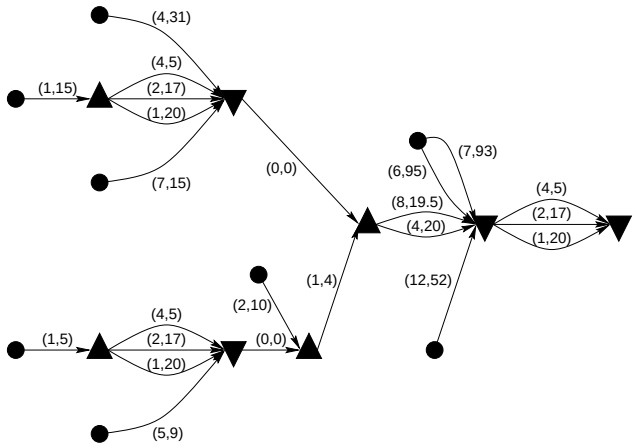

Fig. 2. Example Showing Arcs Labeled with Activity Time and Cost Parameters.

the leadtime and total cost for each of the 129 potentially efficient valid configurations and plot them in Fig. 3 below. Since our objective is to minimize total cost and leadtime, the efficient frontier is the lower-left (southwest) rectilinear envelope of all the points. Observe that the efficient frontier is completely defined by the set of efficient solutions, which are shown in Fig. 3 as those points on the lower-left "corners" of

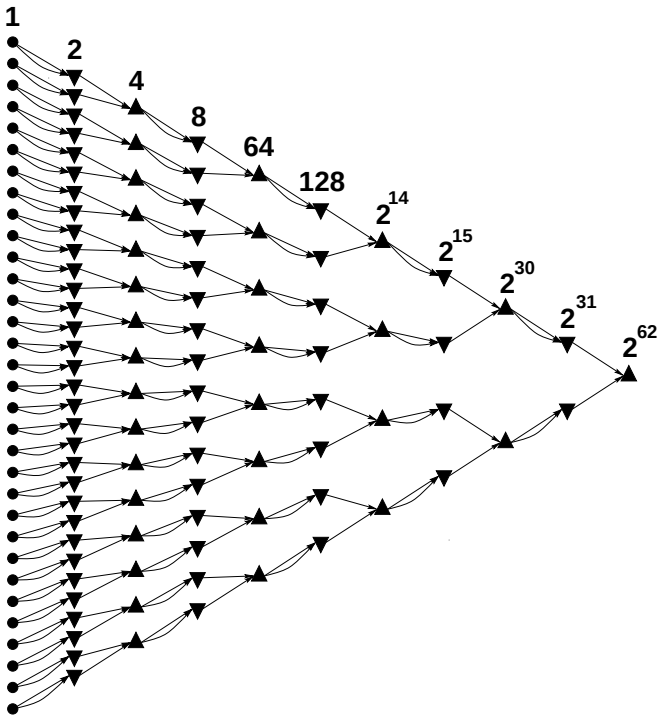

Fig. 4. Example Showing $S(n)$ at Each Level.

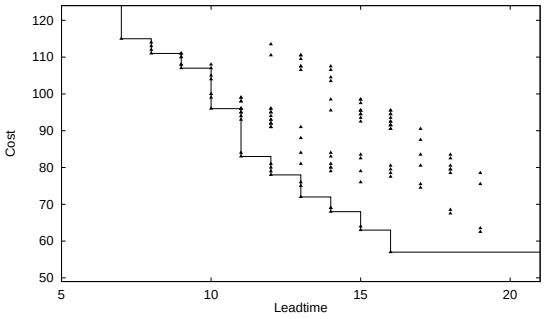

Fig. 3. Enumeration Approach Reveals the Efficient Frontier

the frontier. There are only ten efficient solutions. Fewer than 129 points appear in Fig. 3 since there are some solutions with overlapping values of cost and leadtime.

While the enumeration approach may be viable in this example, the number of combinations can grow astronomically large very quickly. Consider a LCT model whose network is a 5-level binary tree such that there are exactly 5 arcs between any leaf node and the root node, and every node except the leaf node has two incident arcs. There are 63 nodes and 62 arcs in the model. If all nodes are conjunction nodes, then there is only one valid configuration which includes all of the arcs ($S(r) = 1$). Now let us alter the same model by allowing two options for each of the 62 activities. Each arc in the model is now replaced by inserting a disjunction node with two parallel arcs, with one dummy arc connecting the disjunction node with the head node of the original arc as in Fig. 4. Now, we have 31 conjunction nodes and 32 leaf nodes as we did before, but we have 62 new disjunction nodes and we have 186 arcs instead of 62. The number of VUC's is shown in the figure for a node at each "level" of the graph computed according to (2). For the entire supply chain, we see that there are $2^{62} \approx 4.6 \times 10^{18}$ potentially efficient valid configurations. In fact, if every disjunction node and conjunction node were to have three incident arcs instead of the two, the number of potentially efficient valid configurations balloons to 1.6×10^{173}. This is for a supply chain with only 727 nodes and 1452 arcs.

These examples are intended to show that the number of potentially efficient valid configurations can be extremely large even for moderate sized supply chains. This suggests that the enumeration approach is not practical. In the following section, we discuss an alternative approach that is efficient and can deal with large supply chains.

4.2. Recursive Algorithm Approach

As an alternative to the enumeration approach, we now present a solution method that is simple and efficient. This approach makes the following assumption.

Assumption 1 (Discretization Assumption): One of the two performance measures can be discretized such that the number of possible values is bounded by a constant B. Essentially, B can be viewed as the ratio of the maximum possible magnitude of the measure to the unit of quantization of the measure.

The exposition that follows discretizes the leadtime measure although it is possible to discretize the cost measure instead. Our rationale for discretizing leadtime is that it is reasonable to consider leadtimes in units of days and for typical instances of supply chains, the overall leadtime might be a year or less resulting in the bound constant B being no more than 365. If

we discretize cost instead, the value of B is typically much larger. We are interested in a relatively small value of B since the running time depends on the size of B.

Our algorithm follows a dynamic programming approach. We define the state space, the value functions, the boundary conditions and the recursions. The state space is on the measure that is discretized—in this case, the leadtime. Hence, the state space is the set $\{0, 1, \ldots, B\}$ where B is the maximum possible value of the leadtime of the supply chain. The independent axis of the value functions is defined on this space.

The value functions have the identical interpretation as the tradeoff function or efficient frontier. Specifically, a value function for an element of the network represents the lowest cost over all feasible upstream configurations attainable for a given leadtime. The algorithm employs two families of value functions—one for the nodes and another for the arcs. Let $f_n(t)$ be the value function for node $n, n \in \mathcal{N}$ such that $f_n(t)$ is the lowest cost for all predecessor activities to node n that can be completed by time t in all valid configurations.

The other family of value functions is for the arcs. We define $h_i(t)$ to be the value function at the head side (i.e., downstream side) of arc $i, i \in \mathcal{A}$. By specifying the "head side," we mean to include cost c_i and leadtime t_i of activity i itself such that $h_i(t)$ is the lowest cost for completing activity i and its predecessor activities by time t in all valid configurations.

The boundary conditions are defined on the "leaf nodes" of the supply chain which are the nodes comprising the set $\mathcal{L} \subset \mathcal{N}$ having no incident arcs. (Mathematically, $n \in \mathcal{L}$ if and only if $|\mathcal{I}(n)| = 0$.) The boundary condition is stated as:

$$\text{For all } n \in \mathcal{L}, f_n(t) = \begin{cases} 0, & t \geq 0 \\ \infty, & \text{otherwise} \end{cases} \tag{3}$$

Since $n \in \mathcal{L}$ has no predecessor activities, the cost of predecessor activities is 0 as long as any activities subsequent to n begin at time $t \geq 0$. However, in order to prevent infeasible solutions that would begin before time 0, the cost of a solution that requires n to be completed before time 0 is set to infinity. The fact that $f_n(t) = 0$ not simply for $t = 0$ but also $t > 0$ implies that there is no penalty for starting activities with no predecessors after time 0.

Finally, we present simple definitions for the recursions:

1) For all $i \in \mathcal{A}$:

$$h_i(t) = c_i + f_{\tau(i)}(t - t_i) \tag{4a}$$

2) For all $n \in \mathcal{C}$:

$$f_n(t) = \sum_{i \in \mathcal{I}(n)} h_i(t) \tag{4b}$$

3) For all $n \in \mathcal{D}$:

$$f_n(t) = \min_{i \in \mathcal{I}(n)} h_i(t) \tag{4c}$$

In Eq. (4a), the value function at the head side of activity i is the same as the value function at the tail side of activity i only that it is shifted upwards by c_i and to the right by t_i. These shifts account for the activity cost c_i and the activity time t_i. In Eq. (4b), the value function for the conjunction node n is the *sum* of the value functions at the completion of activities corresponding to all the incident arcs to n. In other words, the minimal cost $f_n(t)$ to attain a leadtime of t at node n is the sum of the cost $h_i(t)$ to attain that leadtime t over all the incident arcs. In Eq. (4c), the disjunction node allows us to pick the cheapest from the incident arcs and, hence, we can pick the best set of valid predecessor activities, which satisfies the disjunction at the given leadtime.

The goal of the algorithm is to find $F(t)$, the tradeoff function of the entire supply chain. For the LCT model, the tradeoff function is equivalent to the value function of the root node.

$$F(t) = f_r(t) \tag{5}$$

Two final remarks are that the expressions for the value function in (3) and (4) do not assume that the state space has been discretized and hold in general when the leadtime is real. Secondly, since the equations for the recursive equations in (4) depend on other value functions computed previously, there is the issue of sequencing the evaluations of the recursions. We refer to methods on topological sorting of directed acyclic graphs such as the one described in [12]. The topological sort produces a linear ordering of all nodes such that if the set of arcs $\mathcal{A}$ contains an arc (i, j), then i appears before j in the ordering. We now formally describe the algorithm.

4.2.1) Recursive LCT Algorithm:

Inputs: We are given LCT model with the graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ and leadtime and cost parameters t_i and c_i for $i \in \mathcal{A}$. The size of the state space B is given.

Output: Find the tradeoff function $F(t)$ for the LCT model.

1) Apply the topological sorting algorithm on $\mathcal{A}$ to find a permutation vector σ on the nodes in $\mathcal{N}$. Therefore, if $(i, j) \in \mathcal{A}$, then $k < l$ where $i = \sigma(k)$ and $j = \sigma(l)$.
2) For $k = 1, \ldots, |\mathcal{N}|$: *(visit nodes in order)*
   a. Let $n = \sigma(k)$.
   b. If $|\mathcal{I}(n)| = 0$: *(leaf node)*
      i) For $t = 0, \ldots, B$, $f_n(t) = 0$.
   c. Else:
      i) For $i \in \mathcal{I}(n)$: *(incident arcs)*
         a. For $t = 0, \ldots, (t_i - 1)$, $h_i(t) = \infty$.
         b. For $t = t_i, \ldots, B$,
            $h_i(t) = c_i + f_{\tau(i)}(t - t_i)$.
      ii) If $n \in \mathcal{C}$ then: *(conjunction node)*
         For $t = 0, \ldots, B$, $f_n(t) = \sum_{i \in \mathcal{I}(n)} h_i(t)$.
      iii) Else: *(disjunction node)*
         For $t = 0, \ldots, B$, $f_n(t) = \min_{i \in \mathcal{I}(n)} h_i(t)$.

According to (5), the desired output $F(t)$ is obtained from $f_r(t)$. The algorithm is a fairly straightforward translation of the recursions onto the discretized state space $\{0, \ldots, B\}$. Step 2.b.i corresponds to (3) over the state space domain. Step 2.c.i has two parts—step 2.c.1.a takes care of the case when the domain of $f_{\tau(i)}(\bullet)$ would otherwise be negative in step 2.c.1.b so the value function is infinite.

There are several remaining issues to be discussed regarding the algorithm. We will discuss the algorithm's complexity in Section 4.2.2, refinements in Section 4.2.3, its performance

in computational experimentation in Section 4.2.4, and how to derive a supply configuration from the solution in Section 4.2.5.

4.2.2) Analysis of Complexity:

Proposition 1: Given Assumption 1, the algorithm runs in linear time with respect to the size of the graph $|\mathcal{N}| + |\mathcal{A}|$ in the worst case.

Proof.: Step 1 involves a topological sort which has been shown by Cormen et al. to take $\mathcal{O}(|\mathcal{N}| + |\mathcal{A}|)$ time. Step 2.a, involving a constant time operation, is executed in $\mathcal{O}(|\mathcal{N}|)$. Steps 2.b, 2.c.ii and 2.c.iii are executed $|\mathcal{N}|$ times in total (with each node $n \in \mathcal{N}$ processed in one of the steps), which we will show takes $\mathcal{O}(|\mathcal{N}| + |\mathcal{A}|)$ time in total. What remains is step 2.c.i which we will show takes $\mathcal{O}(|\mathcal{A}|)$ time.

To show that steps 2.b, 2.c.ii and 2.c.iii runs in $\mathcal{O}(|\mathcal{N}|+|\mathcal{A}|)$ time, we observe that in total, the number of assignment operations of $f_n(t)$ is exactly $(B+1)|\mathcal{N}|$ since $f_n(t)$ is computed over the domain $0, \ldots, B$ exactly once for each node $n \in \mathcal{N}$. Steps 2.c.ii and 2.c.iii further involve either an addition operation or a $\min$ operation (both of which take constant time) with $h_i(t)$ over incident arcs $\mathcal{I}(n)$. Observe that in total, for any value of t, $h_i(t)$ is accessed exactly $|\mathcal{A}|$ times (once for each $i \in \mathcal{A}$) since an arc i occurs in exactly one node's set of incident arcs $\mathcal{I}(n)$. Since the number of values of t is bounded by $B+1$, then the number of addition operations is $(B+1)|\mathcal{A}|$ and the total number of constant-time operations performed in steps 2.b, 2.c.ii and 2.c.iii is $(B+1)(|\mathcal{N}|+|\mathcal{A}|)$. Since B is a constant, the order of complexity for the three steps is $\mathcal{O}(|\mathcal{N}| + |\mathcal{A}|)$.

To show that step 2.c.i runs in $\mathcal{O}(|\mathcal{A}|)$ time, observe that the step is executed $|\mathcal{A}|$ times since $h_i(\bullet)$ is computed once for each arc $i \in \mathcal{A}$. Again, this is because an arc appears in exactly one node's set of incident arcs $\mathcal{I}(n)$. In every execution of the step, there are at most $B+1$ assignments and $B+1$ accesses to $f_{\tau(i)}(t-t_i)$, each of which takes constant time. Hence, the total number of constant-time operations performed in step 2.c.i is no greater than $(B+1)|\mathcal{A}|$. Since B is a constant, the order of complexity of the step is $\mathcal{O}(|\mathcal{A}|)$.

Since the total number of operations in each step of the algorithm runs in complexity that is no more than $\mathcal{O}(|\mathcal{N}| + |\mathcal{A}|)$, the algorithm runs in linear time with respect to the size of the graph $|\mathcal{N}| + |\mathcal{A}|$ in the worst case. ■

Corollary 1: Given Assumption 1, the asymptotically tight bound on the complexity of Algorithm 1 is $\Theta(|\mathcal{N}| + |\mathcal{A}|)$.

Proof.: The asymptotic upper bound of $\mathcal{O}(|\mathcal{N}| + |\mathcal{A}|)$ was established in Proposition 1. Simply to read in the input to the algorithm is necessarily linear in the size of the graph, thus an asymptotic lower bound is $\Omega(|\mathcal{N}|+|\mathcal{A}|)$. Therefore, the asymptotically tight bound of the algorithm is $\Theta(|\mathcal{N}| + |\mathcal{A}|)$. ■

Note that while the constant B does not affect the asymptotic bound of the algorithm, the operations count depends on B so it does affect the actual running time of the algorithm.

One further remark is that if the LCT model is connected, which must be true since the model definition stated there is a directed path from every node to the root node, then $|\mathcal{N}| \leq |\mathcal{A}|+1$. Thus, the asymptotically tight bound on the complexity of the algorithm reduces to $\Theta(|\mathcal{A}|)$ time.

4.2.3) Refinements: The algorithm can be refined to exploit sparseness in the value functions. Thus far, the value functions $f_n(t)$ and $h_i(t)$ have been treated as a dense array over the range $0, \ldots, B$. The value function is most compactly defined by the set of efficient solutions on the value function, the size of which is often much smaller than $B+1$. We do not describe here the details on exploiting this fact although the data structures and computations are quite simple. We have implemented this refinement for the purposes of reporting computational experimentation in the following section.

4.2.4) Computational Efficiency: In section 4.2.2, we demonstrated linear time complexity of the algorithm, which is theoretically desirable. In this section, we discuss the computation performance of an implementation of the algorithm using a Sun SPARCstation 20/61 workstation.

Our first experiment is on the example from Fig. 2. The algorithm, as expected, computes the same efficient frontier as the enumeration procedure as shown in Fig. 3 of Section 4.1. The same ten efficient configurations are found with the following pairs of leadtime and total cost values: $(7, 115)$, $(8, 111)$, $(9, 107)$, $(10, 96)$, $(11, 83)$, $(12, 78)$, $(13, 72)$, $(14, 68)$, $(15, 63)$, and $(16, 57)$. The computation time was less than the smallest measurable unit of CPU time, which is $\frac{1}{50}$ of a CPU second but this example is small and has only 129 valid configurations.

To test the algorithm further, we use a test problem created by a problem generator that generates a directed acyclic graph with branches at conjunction nodes and where all incident arcs to the same disjunction node have the same tail node. A special case of this type of graph is shown in Fig. 4. For this test problem, the program randomly generates the number of incident arcs to each conjunction node and disjunction node (unlike the problem in Fig. 4 where the number is fixed at 2) and also randomly generates activity cost and leadtime data. Within the set of parallel arcs that are between the same pair of nodes, each arc has nontrivial parameters in the sense that the leadtime values are distinct among the arcs in the set and an arc with a higher activity time parameter has a lower cost parameter.

In the test problem, there are 1825 nodes (296 conjunction, 912 disjunction and 617 leaf nodes) and 5161 arcs. The path between each leaf node and the root node has up to ten arcs. The activity times range from 1 to 394 (and, less importantly, the activity costs range from 6.05 to 1361.75). The efficient frontier consists of 71 efficient solutions with leadtimes ranging from 231 to 546. The largest leadtime value in any of the value functions is 546, hence the constant bound $B = 546$ would be applicable in this case. The number of feasible solution $S(r)$ is in this case 7.3×10^{601} which would make the enumeration approach completely impractical, and yet the amount of computation time required to find the efficient frontier is only 1.0 CPU seconds according to the Unix `time` command.

The largest problem we have tested (due to memory limitations on the workstation) is a model with 49217 nodes and 300798 arcs. The entire efficient frontier is computed in 61.3 CPU seconds. Therefore, we conclude that the algorithm is capable of solving for the efficient frontier in large-scale

supply chains.

4.2.5) Deriving An Efficient Configuration: It is often the case in dynamic programming formulations that the algorithm is designed primarily to find the optimal value and not the optimal solution itself. Given the value functions $f_n(t)$ for all $n \in \mathcal{N}$ and $h_i(t)$ for all $i \in \mathcal{A}$, we can proceed in the opposite direction of the recursions, starting at the root node and build up the subset of nodes and arcs that form the configuration $(\mathcal{N}', \mathcal{A}')$ by scanning the graph in the opposite direction of the orientation of the arcs.

By applying that procedure, we can show the efficient solution with leadtime 14 and cost 68 for the example in Figs. 1 and 2. This is depicted in Fig. 5 where the arcs in $\mathcal{A}'$ are displayed with thick lines. Another configuration

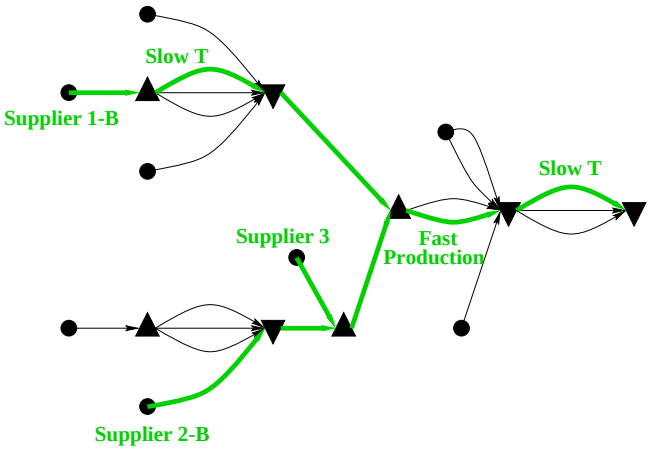

Fig. 5. The Configuration Attaining Leadtime 14 and Cost 68.

corresponding to the efficient solution with leadtime 7 and cost 115 consists of a completely different subset of activities. In this case, there are two arcs (namely, arcs 2 and 4 in Figs. 1 and 2) corresponding to a subcontracting option. It is a simple matter to verify the supply chain's leadtime and total cost in both cases and that the configuration is valid according to (1).

5. Insights Gained

The tradeoff function computed by the LCT model provides several insights. Consider the set of activities corresponding to a particular efficient solution such as the one shown in Fig. 5. Now, consider the union of all activities corresponding to all efficient solutions which is shown in Fig. 6 for the ten efficient solutions from the example in Fig. 2. Observe that there are some activities that do not appear in any efficient solution. This immediately implies that those activities should never be part of any solution.

A related insight concerns the triplets of transportation mode arcs between nodes 7 and 5, nodes 11 and 10, and nodes 1 and 0 (see Fig. 1). Observe in Fig. 2 that the activity time and cost parameters for the triplet of arcs is identical. However, Fig. 6 (which is computed using the parameters in Fig. 2) shows that the fast mode (leadtime 1) is never efficient between nodes 7 and 5, whereas the slow mode (leadtime 4) is never efficient between nodes 11 and 10, whereas the medium mode (leadtime 2) is never efficient between nodes 1 and 0. The insight is that contracting for all transportation services with a carrier that operates a single mode is always inefficient regardless of the leadtime-cost tradeoff in this case.

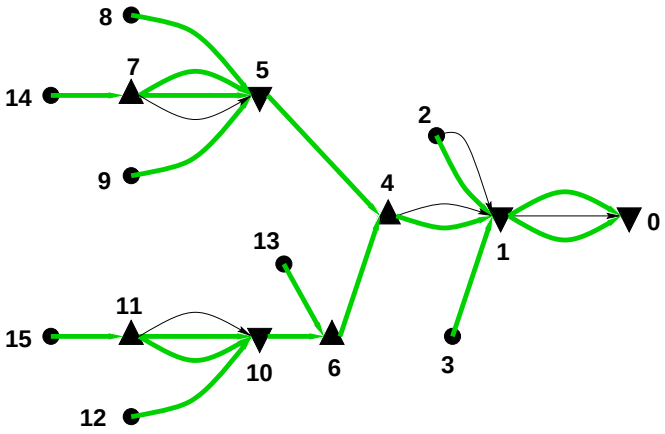

Fig. 6. Thick Arcs Occur in Some Efficient Supply Chain Configuration.

The model can also be used to evaluate the efficiency of the current operations. By observing the leadtime and total cost among activities under the current operating procedures and comparing it with the tradeoff frontier, insights can be gained into the efficiency of the current operating procedures.

The tradeoff function can also determine the value added by each activity in the supply chain. We have already determined that we can delete arcs that appear in no efficient frontier without affecting performance. For every arc, we can compare the tradeoff function $F(t)$ to the tradeoff function $F^{\mathcal{A}\setminus\{i\}}(t)$ with activity i removed. Necessarily, $F^{\mathcal{A}\setminus\{i\}}(t) \geq F(t)$. By measuring the difference between $F^{\mathcal{A}\setminus\{i\}}(t) - F(t)$ at some leadtime t, we have a measure of the value added by activity i in terms of *cost*. Conversely, if $G(c)$ represents the inverse function of $F(t)$, then the value added by activity i in terms of *leadtime* is $G^{\mathcal{A}\setminus\{i\}}(c) - G(c)$ for some total cost c. Therefore, the fact that we can compute the tradeoff function allows us to perform sensitivity analysis of a firm's value chain.

A final insight is that the efficient frontier, which represents

the firm's production function with respect to different service levels, can be coupled with the market demand function to determine an optimal offering of price and service points.

6. Concluding Remarks

The LCT model and related solution procedure presented in this paper enable an enterprise to take advantage of abundant choices and flexibility that come with electronic commerce. Our main technical contributions are two-fold. First, the LCT model is capable of modeling a wide range of supply chain activities and explicitly addresses time and cost measures in an integrated manner. Second, we have developed a recursive solution procedure that can be used to identify the entire efficient frontier between leadtime and cost for large supply chain networks, suitable for real-world application in electronic commerce. We conclude this paper by presenting some extensions to the LCT model.

Other performance measures of supply chains such as product quality, service levels, and environmental impact [13], are not addressed in the current model. We plan to enrich our model to address these additional measures. In addition, we are interested in strategic aspects of supply chain management. In the research reported in this paper, we assume that all leadtime/cost parameters are given. In cases where supply chain agents are responsible for making these pricing decisions, strategic interactions among business entities become critical. We are currently working on extending our previous work on learning and negotiation [14] to address issues that arise from adaptive multi-issue negotiation in inter-organizational electronic commerce.

References

[1] M. Totty, "E-commerce (a special report): Overview — the next phase," in *Wall Street Journal, May 21*. New York, 2001.

[2] D. Rosenfield, R. Shapiro, and R. Bohn, "Implications of cost-service trade-offs on industry logistics structures," *Interface*, vol. 15, no. 6, pp. 47–59, 1985.

[3] J. Narus and J. Anderson, "Rethinking distribution: Adaptive channels," *Harvard Business Review*, vol. 74, no. 4, pp. 112–120, 1996.

[4] M. Porter, *Competitive Advantage: Creating and Sustaining Superior Performance*. The Free Press, 1985.

[5] V. Srinivasan and G. L. Thompson, "Determining cost vs. time pareto-optimal frontiers in multi-modal transportation problems," *Transportation Science*, vol. 11, pp. 1–19, 1977.

[6] H. Lee and P. Pulat, "Bicriteria network flow problems: Continuous case," *European Journal of Operational Research*, vol. 51, pp. 119–126, 1991.

[7] ——, "Bicriteria network flow problems: Integer case," *European Journal of Operational Research*, vol. 66, pp. 148–157, 1993.

[8] P. De, E. Dunne, J. Ghosh, and C. Wells, "The discrete time-cost tradeoff problem revisited," *European Journal of Operational Research*, vol. 81, pp. 225–238, 1995.

[9] B. Arntzen, G. Brown, T. Harrison, and L. Trafton, "Global supply chain management at digital equipment corporation," *Interface*, vol. 25, no. 1, pp. 69–93, 1995.

[10] T. Morton and D. Pentico, *Heuristic Scheduling Systems: With Application to Production Systems and Product Management*. New York, N.Y.: John Wiley and Sons Inc., 1993.

[11] D. Zeng, "Managing flexibility for inter-organizational electronic commerce," *Electronic Commerce Research Journal*, vol. 1, no. 2, pp. 33–51, 2001.

[12] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*. The MIT Press, 1990.

[13] M. Dotoli, M. Fanti, C. Meloni, and M. Zhou, "Design and optimization of integrated e-supply chain for agile and environmentally conscious manufacturing," *IEEE Transactions on Systems, Man, and Cybernetics: Part A*, vol. 36, no. 1, pp. 62–75, Jan 2006.

[14] D. Zeng and K. Sycara, "Bayesian learning in negotiation," *International Journal of Human-Computer Studies*, vol. 48, pp. 125–141, 1998.

Arthur Hsu received a Ph.D. degree in Industrial Engineering and Management Sciences from Northwestern University (1995) and a S.M. degree in Transportation from M.I.T. (1990). He is currently a Principal Engineer at the United Technologies Research Center and is leading advanced technology projects in elevator group control algorithms and optimized elevator configuration design in high-rise buildings. In addition, he has led projects in service operations and supply chain design. His research interests include computational methods for optimization, service operations management and distribution problems.

Daniel Zeng received the M.S. and Ph.D. degrees in industrial administration from Carnegie Mellon University, Pittsburgh, PA, in 1994 and 1998, respectively, and the B.S. degree in economics and operations research from the University of Science and Technology of China, Hefei, China, in 1990. Currently, he is an Associate Professor and the Director of the Intelligent Systems and Decisions Laboratory in the Department of Management Information Systems at the University of Arizona. He is currently directing several National Science Foundation (NSF)-funded research projects as PI or co-PI. His research interests include intelligence and security informatics, multi-agent systems, distributed optimization, computational support for auctions and negotiations, intelligent information integration and caching, and recommender systems. He has co-edited five books and published about 70 peer-reviewed articles in Management Information Systems and Computer Science and Engineering journals, edited books, and conference proceedings. He served as conference or program co-chair for the first four Intelligence and Security Informatics conferences (ISI-2003, ISI-2004, ISI-2005, and ISI-2006). He is a member of IEEE, INFORMS, AAAI, and ACM, and serves on the editorial board of Decision Support Systems, Journal of Database Management, and several other IT-related journals. He is the VP for Technical Activities of the IEEE Intelligent Transportation Systems Society. He is also the VP for Academic Activities and Publications of the Chinese Association for Science and Technology, USA.