# STRATEGIC SAFETY STOCK PLACEMENT IN INTEGRATED PRODUCTION/DISTRIBUTION SYSTEMS

by

Sean Peter Willems

B.S.E. Economics
Wharton School, University of Pennsylvania
1993

Submitted to the Sloan School of Management
in Partial Fulfillment of
the Requirements for the Degree of
Master of Science in Operations Research

at the

Massachusetts Institute of Technology
May 1996



Signature of Author _____
MIT Operations Research Center
May 23, 1996

Certified by _____
Stephen C. Graves
Professor of Management Science
Thesis Supervisor

Accepted by _____
Thomas L. Magnanti
Co-Director, Operations Research Center

# Strategic Safety Stock Placement in Integrated Production/Distribution Systems

by

Sean Peter Willems

Submitted to the Sloan School of Management
on May 23, 1996 in Partial Fulfillment of
the Requirements for the Degree of
Master of Science in Operations Research

## Abstract

An emerging principle for the management of supply chains is that a supply-chain perspective provides the opportunity for significant savings in inventories from better coordination and communication across the supply chain. One component of this savings is due to a coordinated strategy for setting safety stocks to protect against uncertainty and variability; that is, a supply-chain perspective can avert some of the local suboptimization that occurs, for instance, when each stage of a manufacturing or distribution process independently determines its own safety stocks. In this thesis we describe ongoing research to develop tools and general principles for determining how to set safety stocks in a supply chain.

This thesis addresses how to determine the optimal placement of safety stock inventories in a multi-stage supply chain subject to uncertain demand. The thesis first formulates a model, originally given in Simpson (1958), for the simplest supply chain, a multi-stage serial system, and presents a solution procedure to find the optimal safety stocks. We extend Simpson's model to assembly networks in Chapter 3, to distribution networks in Chapter 4, and to integrated assembly/distribution networks in Chapter 5. In Chapter 6, we describe two industrial settings where the model has been used.

Thesis Supervisor: Stephen C. Graves
Title: Professor of Management Science

## Acknowledgments

First and foremost, I would like to thank my advisor, Professor Stephen C. Graves, for giving me the opportunity to work on this research and taking the time to make this thesis presentable.

The research leading to this thesis began in the summer of 1994, when I was given the opportunity to spend the summer at the Elmgrove facility of the Eastman Kodak Company looking at supply chain issues. Since then, many persons at Kodak have played large roles in validating the model and ensuring that the model is useful to Kodak. I extend my sincere thanks to:

- Ron Caldwell, Kodak's technology liaison to MIT, for his willingness to devote time and resources to ensure the success of this project.
- Chuck Petro, of the Industrial Engineering group at Kodak, for his active involvement in identifying departments and personnel at Kodak that would be interested in using the model.
- Bill Carleton, flow team leader of the DES 100, for his willingness to understand the model, request feature improvements, and actually use the model as a decision support tool for the DES 100 product flow.
- Brad Horton, cycle time coordinator for Elmgrove, for helping to determine how best to use the model at Kodak.

Finally, the author gratefully acknowledges the support and resources made available to him through the Leaders for Manufacturing Program, a partnership between MIT and major U.S. manufacturing companies.

# Table of Contents

# 1 Introduction

An emerging principle for the management of supply chains is that a supply-chain perspective provides the opportunity for significant savings in inventories from better coordination and communication across the supply chain. One component of this savings is due to a coordinated strategy for setting safety stocks to protect against uncertainty and variability; that is, a supply-chain perspective can avert some of the local suboptimization that occurs, for instance, when each stage of a manufacturing or distribution process independently determines its own safety stocks. In this thesis we describe ongoing research to develop tools and general principles for determining how to set safety stocks in a supply chain.

In particular, we address how to determine the optimal placement of safety stock inventories in a supply chain subject to uncertain demand. In Chapter 2 we formulate a model, originally given in Simpson (1958), for the simplest supply chain, a multi-stage serial system, and present a solution procedure to find the optimal safety stocks. We extend Simpson's model to assembly networks in Chapter 3, to distribution networks in Chapter 4, and to integrated assembly/distribution networks in Chapter 5. In Chapter 6, we describe two industrial settings where the model has been used. We conclude the thesis in Chapter 7 with a status report on future extensions to the model that we will be working on.

Related work on determining the inventory requirements for a supply chain include Lee and Billington (1993) (and the references therein) and Graves et al. (1996). This paper differs from the earlier work in terms of the underlying assumptions, which result in our focus on where to place strategic inventories that completely decouple the upstream part of the supply chain from the downstream. In contrast Lee and Billington determine how much inventory is needed at each stage in the supply chain, so as to minimize total inventory. And Graves et al.

determine the safety stock for a supply chain that is subject to dynamic requirements planning, driven by a dynamic forecast process.

# 2 Serial System Supply Chain

We first review the Simpson model of a serial production-inventory system and then present a solution procedure. In Section 2.1 we detail the model's assumptions, formulation, and the notation we will be using throughout the paper. In Section 2.2 pipeline stock and safety stock are defined in terms of the inputs from Section 2.1. Section 2.3 presents how the inventory holding cost is determined. In Section 2.4 the math programming formulation of the model is presented. Section 2.5 presents two equivalent solution procedures. The first method solves the problem using dynamic programming. The second method finds the optimal safety stock placement by solving a shortest path problem. Although the two methods produce equivalent solutions, they both help to develop intuition for the more complex networks that we will analyze in subsequent chapters.

## 2.1 Model Assumptions And Notation

### 2.1.1 Network representation

We consider an N-stage serial system, where stage i is the immediate upstream supplier for stage i+1, for i = 1, 2, ..., N-1. Hence, stage 1 is the raw material stage and has no supplier; and stage N is the finished goods inventory stage, from which customer demand is served. Each stage represents a major processing function in the supply chain; a typical stage might represent the manufacturing of a subassembly or the shipment of the finished product from a regional warehouse to the customer's distribution center. The only requirement for stage selection is that the process flow map that results from the selection captures the actual supply chain's characteristics; i.e., the model can be tactical in scope but it should not leave any gaps in how the product gets from one stage to another. A schematic of a typical supply chain is shown in Figure 2-1:

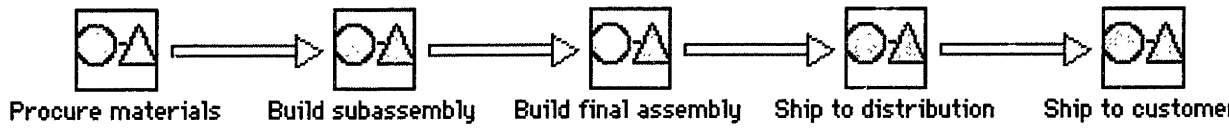


**Figure 2-1: Depiction of a typical serial supply chain**

Figure 2-1 corresponds to a five-stage supply chain. For each stage, the circle represents the stage's processing function and the triangle denotes that the stage is stocking product that has been processed by the stage. For example, for the procure materials stage, which is stage 1, the circle represents the procurement function and the triangle denotes that the stage is stocking purchased materials. Therefore, Figure 2-1 depicts a supply chain where every stage stocks inventory of its completed parts.

Figure 2-2 depicts the same supply chain as in Figure 2-1 for the case where the build subassembly stage, stage 2, does not hold any inventory. (When a stage does not hold any inventory, the triangle disappears to denote that the stage is still performing its processing function, but inventory is no longer held at the stage.)

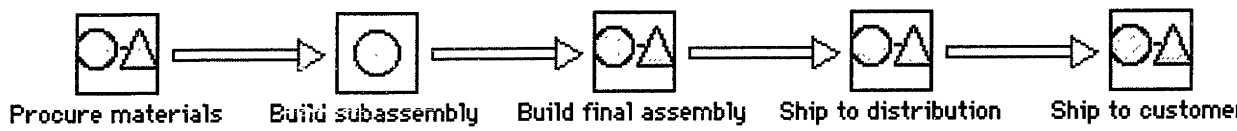


**Figure 2-2: Depiction of supply chain where stage 2 does not stock inventory**

### 2.1.2 Demand characterization

We assume demand each period is an independent normally-distributed random variable with mean $\mu$ and standard deviation $\sigma$. This is the only source of uncertainty in the model.

### 2.1.3 Stage notation and assumptions

#### 2.1.3.1 Inventory policy

Each stage operates with a (echelon) base-stock control policy; that is, each period each stage observes the current customer demand at stage N, and orders a quantity equal to replenish the current period's demand.

In effect, there is a base-stock level associated with each stage. Each period each stage places a replenishment order to bring its on-hand and on-order inventory back to the base-stock level.

#### 2.1.3.2 Capacity constraints

There are no capacity constraints in the supply chain. Production has infinite capacity each period. The buffer size of an inventory location for any stage can be infinite.

#### 2.1.3.3 Production lead-time

For each stage i, we know its production lead-time $T_i$, which we assume is deterministic. Production lead-time includes the time to get the product from the upstream stage, the time to process the product at the current stage, plus any waiting time at the stage.

#### 2.1.3.4 Service time

Each stage i quotes and guarantees a service time $S_i$ by which it will deliver product to its immediate successor. We assume that the finished goods stage provides immediate service from inventory to the final customer; i.e. $S_N = 0$. But for the other stages, these service times are decision variables for the optimization model. Thus, if $S_i = 3$, say, then when an order is placed on stage i at time t, the product will be available for consumption by stage i+1 at time t + 3. If $S_i = 0$, then

when an order is placed on stage i at time t, the product is immediately available for consumption by stage i+1.

If the production lead-times and service times for each stage are known, then the replenishment lead-time for a stage i, denoted by $L_i$, is just $L_i = S_{i-1} + T_i$ for all i, assuming that $S_0$ exists and is equal to 0 by assumption. That is, the replenishment lead-time for a stage is the time it takes the stage to procure its raw materials and perform any required processing function at the stage.

Without capacity constraints there is no reason for stage i to promise a service time longer than its own replenishment lead-time. Therefore, we assume that $S_i \leq S_{i-1} + T_i$, or equivalently $S_i \leq L_i$.

2.1.3.5 Service level

To determine the base-stock level for a stage, we assume that associated with each stage is a service level representing the percentage of time that the guaranteed service time is to be met from inventory. Furthermore we do not attempt to model what happens when the guaranteed service time is violated, i.e., when demand exceeds some maximal level. In effect we assume that the base-stock will be sufficient as long as demand is within the range implied by the desired service level. For example, if the service level is 95%, then we assume that the base-stock will be set to cover a maximum demand equal to the 95th percentile of the demand distribution for any t-period time window; in effect, for setting the base-stock level, we assume the maximum demand over t periods is $t\mu + k_i\sigma\sqrt{t}$ where $k_i = \Phi^{-1}(.95) = 1.645$. We ignore what happens when demand exceeds this level; that is, when demand might be regarded as being extraordinary, we assume that the operation would respond with an equally extraordinary measure, beyond the scope of the model and the assumed base-stock policy. The reasonableness of this assumption is discussed more fully in both Simpson (1958) and Graves (1988).

2.1.3.6 Stage costs

Let $v_i$ denote the direct costs added at stage i. The direct costs at a stage include both direct material and direct labor costs.

We assume that $v_i \geq 0$ for all i. This implies that total cost added is non-decreasing across the supply chain.

## 2.2 Inventory Level Calculations

### 2.2.1 Pipeline stock calculation

The pipeline stock for a stage is the amount of inventory that is currently in-process at the stage; i.e., product that has already been taken from its raw material form but has not yet finished being processed at the stage. For each stage, the expected demand per time unit is $\mu$. Therefore, since demand is assumed to be an independent and identically distributed random variable, the expected amount of pipeline inventory is just the length of the pipeline times the expected demand per time unit; this is just $T_i\mu$ for each stage i.

### 2.2.2 Safety stock calculation

The net coverage time for a stage is the number of time units that the stage's safety stock will have to cover. Given the notation we have already developed, the net coverage time for a stage is just $L_i - S_i$; that is, the net coverage time for stage i is stage i's replenishment lead-time minus its service time.

If we let $\tau$ denote the coverage time for a given stage i, then the demand over the stage's coverage time is a normally distributed random variable with expectation $\tau\mu$ and variance $\sum_{k=1}^{\tau}\sigma_k^2$ where $\sigma_k^2$ is the daily variance of demand on day k. Since demand is assumed to be independent and identically distributed, the variance is equal to $\tau\sigma^2$; therefore, the standard deviation of demand over the coverage time is $\sqrt{\tau\sigma^2} = \sigma\sqrt{\tau}$.

Now that we have the standard deviation of demand over the coverage time, we set the safety stock at stage i as a multiple of this standard deviation, where the multiple ($k_i$) is set to satisfy the specified service level (e.g., $k_i = \Phi^{-1}(.95)$). Therefore, letting $I_i$ denote the expected amount of safety stock held at stage i, $I_i = k_i\sigma\sqrt{L_i - S_i}$.

## 2.3 Inventory Cost Calculations

### 2.3.1 Holding cost calculation

To determine the safety stock cost at stage i, we need to determine stage i's holding cost. For each stage i, we know the direct costs added at the stage. For the purposes of calculating holding costs, it is necessary to determine the total direct costs that have been added from stage 1 up to and including the current stage. For stage i, denote $V_i$ as the total direct cost added up to and including stage i; i.e., the cumulative cost. Since the supply chain is a serial line by assumption, $V_i$ is determined by the trivial recursion $V_i = V_{i-1} + v_i$ for stages i = 2, ..., N and $V_1 = v_1$. If we assume a holding cost rate of $\alpha$ then $h_i$, the holding cost at stage i, equals $\alpha V_i$.

Note that the calculation of holding costs presented in this section ignores potentially important factors like physical handling costs and deterioration. For the purposes of the model which will be presented in Section 2.4, it is only important that $h_i$ exists for each stage. The development in this section has tried to develop some intuition for what might constitute an acceptable derivation of $h_i$, but it is by no means the only acceptable derivation.

### 2.3.2 Pipeline stock cost calculation

To determine the cost of the pipeline stock for the entire supply chain, we multiply the pipeline stock at each stage times the average value of the stock at the stage. If there are N stages numbered from 1 to N, the pipeline stock cost is just $\sum_{i=1}^{N}(\frac{h_{i-1}+h_i}{2})T_i$; note this assumes that $h_0$ exists and is equal to zero.

For the purposes of our analysis we valued a stage's pipeline as the average of the starting and ending inventory costs for the stage. This is certainly an arbitrary valuation function and can be changed to a more complicated function if the conditions justify this.

It is also important to note that regardless of the service times chosen, the pipeline stock, and hence the pipeline stock cost, will not change. This is based on our assumptions of deterministic production times and of a base-stock control policy. For this reason, our analysis and problem formulation will not include pipeline stock. However, it could be added as a constant term in the objective function and it should certainly not be forgotten when assessing the total costs of the supply chain.

### 2.3.3 Safety stock cost calculation

Since the safety stock held at a stage has already been processed by the stage, the safety stock cost for stage i is just $h_i I_i$. Therefore, the total holding cost for safety stock in the supply chain is $\sum_{i=1}^{N} h_i I_i$.

## 2.4 Math Programming Formulation

With these observations and assumptions, we can now formulate the following optimization problem for finding the optimal service times, or equivalently safety stocks, for each stage:

$$\begin{aligned} &\min \sum_{i=1}^{N} h_i I_i \\ &\text{s.t.} \\ &\quad I_i = k_i \sigma \sqrt{S_{i-1} + T_i - S_i} \qquad i = 1,\ldots,N \\ &\quad 0 \le S_i \le S_{i-1} + T_i \qquad i = 1,\ldots,N \end{aligned} \tag{1}$$

The objective function minimizes the total inventory holding cost across the supply chain subject to a constraint that the service time at a stage can not exceed the stage's production lead-time plus the service time quoted to the stage.

This problem formulation was first given by Simpson, who also showed that there is an optimal extreme point solution such that $S_i^* = 0$ or $S_i^* = S_{i-1}^* + T_i$ for all i = 1, 2, ...N-1 (Simpson assumes $S_N$, the service time to the customer, equals zero). Thus, there is an "all or nothing" optimal solution; either a stage has no safety stock ($S_i^* = S_{i-1}^* + T_i$) or the stage has sufficient safety stock ($S_i^* = 0$) to decouple it from its downstream stage.

## 2.5 Solution Procedure

The serial line case can be solved to optimality using dynamic programming or equivalently solving a shortest path problem. The next two sections will describe each approach in turn, while providing insights into the strengths of each approach.

### 2.5.1 Dynamic programming solution procedure

#### 2.5.1.1 Forward recursive formulation

The dynamic program is a forward recursion starting at stage 1 and proceeding to stage N. For each stage, the algorithm finds the service time from the upstream stage that minimizes the cost of the current stage quoting a given service time; this procedure is repeated for each possible service time that the current stage can quote. We define $f_i(\beta)$ as the optimal cost-to-go function from stage 1 up to and including stage i given stage i quotes a service time of $\beta$, as

$$f_i(\beta) = \min_{[\beta - T_i]^+ \le \alpha \le \sum_{j=1}^{i-1} T_j} \left(f_{i-1}(\alpha) + c_i(\alpha,\beta)\right) \qquad 0 \le \beta \le \sum_{j=1}^{i} T_j \quad \text{for } i = 2,\ldots,N \tag{2}$$

$$\text{where} \quad c_i(\alpha,\beta) = k_i h_i \sigma \sqrt{\alpha + T_i - \beta} \qquad \text{for } i = 1,\ldots,N \tag{3}$$

$$\text{and} \quad f_1(\beta) = c_1(\alpha,\beta) \qquad \text{for } 0 \le \beta \le T_1$$

Equation (2) is the optimal cost-to-go function for a given service time $\beta$. The minimum cost up to stage i is the minimum holding cost up to and including stage

i-1, given stage i-1 quotes a service time of $\alpha$, plus the inventory holding cost at stage i for quoting a service time of $\beta$ and receiving a quoted service time of $\alpha$.

Equation (3) is the safety stock cost at stage i given stage i quotes a service time of $\beta$ and is quoted a service time of $\alpha$. For a given $\alpha$ and $\beta$, $L_i$, the replenishment lead-time of stage i, is just $\alpha + T_i$ and the coverage time of stage i is $L_i - \beta$.

By observation, $c_i(\alpha,\beta)$ is strictly decreasing in $\beta$ over the interval $0 \le \beta \le \alpha + T_i$ and $c_i(\alpha,\beta)$ is strictly increasing in $\alpha$ over the interval $[\beta - T_i]^+ \le \alpha \le \sum_{j=1}^{i-1} T_j$.

2.5.1.2 Backward recursive formulation

While we have just presented the forward dynamic program, it is also possible to solve the serial line network using a backward recursion. In the backwards case, we proceed from stage N to stage 1. For each service time that can be quoted to stage i, we find the optimal service time that stage i should quote to its customer. The equation is shown in equation (4), for i = 1, ..., N-1

$$f_i(\alpha) = \min_{0 \le \beta \le \alpha + T_i} \left( f_{i+1}(\beta) + c_i(\alpha,\beta) \right) \qquad \alpha = 0,1..., \sum_{j=1}^{i-1} T_j, \tag{4}$$

$$\text{where} \quad f_N(\alpha) = c_N(\alpha,0)$$

We will see in future sections that solving assembly networks will involve use of the forward recursion approach while solving distribution networks requires solving the backward recursion.

### 2.5.2 Shortest path solution procedure

To solve the math program posed in Section 2.4 as a shortest path problem, we first have to transform the serial supply chain into a network of nodes and arcs. Figure 2-3 shows a four stage serial supply chain and the resulting five node shortest path problem.

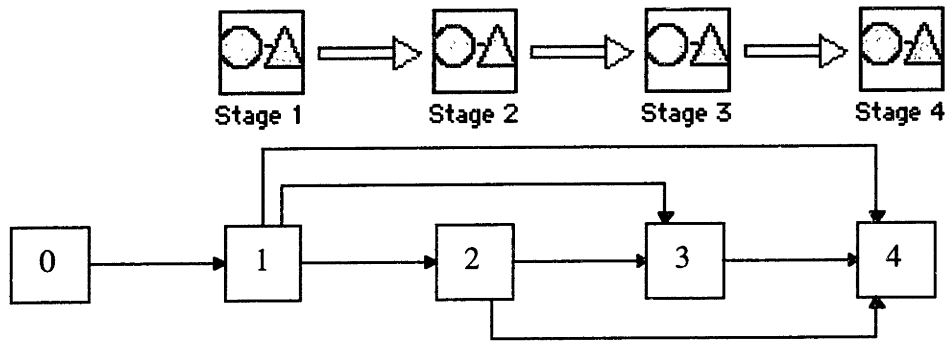


**Figure 2-3: Conversion of supply chain to shortest path problem**

The next few paragraphs detail how the network transformation occurs. To determine the nodes, we assign a node to each stage in the serial supply chain. For the example in Figure 2-3, node i in the shortest path formulation corresponds to stage i in the serial supply chain. We also require an additional node, denoted node 0 whose use will become clear soon.

Recall from Section 2.4 that the optimal service time quoted for any stage i is either equal to 0 or $S_{i-1} + T_i$ (Simpson 1958). In particular, given that $S_N = 0$, we can solve the problem as a shortest path from node 0 to node N on an N+1 node network with arcs (i, j) for all i < j, and i, j = 0, 1, 2, ..., N. The cost of arc (i, j) is the inventory holding cost for having a decoupling inventory at stage j, assuming that the next upstream decoupling inventory is at stage i.

Let $c_{ij}$ equal the cost on arc (i,j). Then, for example, $c_{24}$ is the holding cost for safety stock at stage 4, with $S_4 = 0$, $S_3 = S_2 + T_3$, and $S_2 = 0$:

$$c_{24} = h_4 k_4 \sigma \sqrt{S_3 + T_4 - S_4} = h_4 k_4 \sigma \sqrt{T_3 + T_4}$$

From the extreme point property, $S_3 = 0$ or $S_3 = S_2 + T_3$. Given that arc (2,4) is defined by stages 2 and 4 holding inventory while no intermediate stage holds inventory, this means that $S_3$ will equal $S_2 + T_3$. We can then perform the same substitution for $S_2$. Since stage 2 is required to hold inventory, we know $S_2$ equals zero and we can stop. In general, for any j > i, $c_{ij} = h_j k_j \sigma \sqrt{\sum_{k=i+1}^{j} T_k}$.

Given an optimal solution to the shortest path problem, we have constructed the arcs in such a way that the optimal solution corresponds to the "stock/don't stock" solution we are looking for; each arc represents a link between two stages that quote service times of zero while the stages that are in between the two stages quote their maximum possible service times.

The next step is to use the nodes on shortest path to calculate the service times for each stage in the network. We can reconstruct the service times for each stage by looping over the nodes from node 1 to node N; node 0 is ignored since it was required only for a starting point to exist. If node i is on the shortest path, then we set $S_i = 0$ since a node on the shortest path corresponds to a stage holding inventory. If the node is not on the shortest path, then we set $S_i = S_{i-1} + T_i$. Since a node that is not on the shortest path must not be holding inventory, we have to set its service time to its maximum possible service time given an incoming service time of $S_{i-1}$. Once the algorithm loops over all of the nodes, the algorithm has then calculated the correct service time for each stage. It is easy to verify that this algorithm maintains the extreme point property.

As an example, assume the shortest path for the network presented in Figure 2-3 that returns a shortest path of 0, 1, 4. This corresponds to the following service times: $S_1 = 0$, $S_2 = T_2$, $S_3 = T_2 + T_3$, $S_4 = 0$.

### 2.5.3 Comparison of solution procedures

For a serial system supply chain, the shortest path formulation will solve faster since it exploits the extreme point property of the optimal solution. The dynamic programming formulations both solve for all possible service times at every stage and as such perform many more computations. However, in terms of actually implementing the solution procedures, both of these solution procedures are comparable when solving them by computer for any moderately sized problem; a

typical supply chain will not exceed twenty nodes and the number of nodes can be increased by more than an order of magnitude without any degradation in either solution procedure's computation time.

## 2.6 Extensions to base formulation

Even within the serial line framework, there are several extensions that are necessary before we can study more general networks. This section describes the three most important extensions required for developing future generalizations.

### 2.6.1 Allowing stage 1 to quote nonzero service times

The initial formulation assumes that $S_1 = 0$. This assumption can be relaxed in the dynamic programming formulation simply by letting $S_1$ range between 0 and $T_1$. In the shortest path formulation, we can add arcs between node 0 and each node (1, 2, ... N) where the destination node would be the first stage to hold inventory in the serial line.

### 2.6.2 Allowing stage N to quote nonzero service times

Instead of assuming that the service time for stage N is zero, we now let $U_N$ denote the maximum service time for stage N.

In the dynamic programming case, the only change required to let stage N quote a nonzero service time is to solve equation (2), from Section 2.5.1.1, for $f_N(U_N)$ instead of $f_N(0)$. We can recognize from our previous observation about the monotonicity of equation (3) that there is no need to solve $f_N$ for any value less than $U_N$ since the algorithm is greedy and as such will quote to largest service time possible.

In the case of the shortest path problem, instead of solving a N+1 node shortest path problem we now solve a M+1 node shortest path problem where M is the node that satisfies the two conditions such that $\sum_{k=M}^{N} T_k > U_N \geq \sum_{k=M+1}^{N} T_k$.

Note that these conditions assume $U_N \leq \sum_{k=1}^{N} T_k$. If this is not the case, then the optimal solution is to not stock anywhere since each stage can quote its maximum possible service time. To give a more concrete example of the value of M, assume that we have represented a serial supply chain as follows: $T_0 = 0$, $T_1 = 5$, $T_2 = 3$, $T_3 = 3$, $T_4 = 5$, $T_5 = 2$, $T_6 = 4$. If $U_N = 8$, then $M = 4$. We also have to modify all arcs that terminate at node M. Instead of assuming $S_M = 0$ for all arcs (i,M) we calculate $c_{iM}$ for $S_M = U_N - \sum_{k=M+1}^{N} T_k$; this is the residual amount of nonzero service time that has not been consumed by the upstream stages. For the previous example, $S_M = 2$.

### 2.6.3 Allowing multiple independently-distributed demand origins

Since there are no capacity constraints in the supply chain, the optimal solution does not change if we allow multiple independent demand origins. Given that the demands are independent, the algorithm remains the same but $\sigma_i$ is different for different stages. If we assume that $\upsilon_i$ is the expected demand originating at node i then the total demand at stage i for any period is $\mu_i = \upsilon_i + \mu_{i+1}$ for i = 1,...,N-1 and $\mu_N = \upsilon_N$. Letting $\gamma_i$ be the standard deviation of demand at stage i, we find $\sigma_i = \sqrt{\gamma_i^2 + \sigma_{i+1}^2}$ for i = 1,...,N-1 and $\sigma_N = \gamma_N$.

### 2.6.4 Allowing arc multipliers

We can also incorporate arc multipliers using the same ideas from Section 2.6.3. Let $\mu_N$ equal the expected demand at stage N and let $\sigma_N$ equal the standard deviation of demand at stage N. Then, let $\delta_i$ represent the number of end items from stage i that are needed to produce one unit of stage i+1's production requirements. It is easy to verify that $\mu_i = \delta_i \mu_{i+1}$ and $\sigma_i = \delta_i \sigma_{i+1}$ for i = 1,...,N-1.

# 3 Assembly Networks

An assembly network is an acyclic graph in which each node can only have one outgoing arc and one or more incoming arcs. A typical assembly network is shown in Figure 3-1:

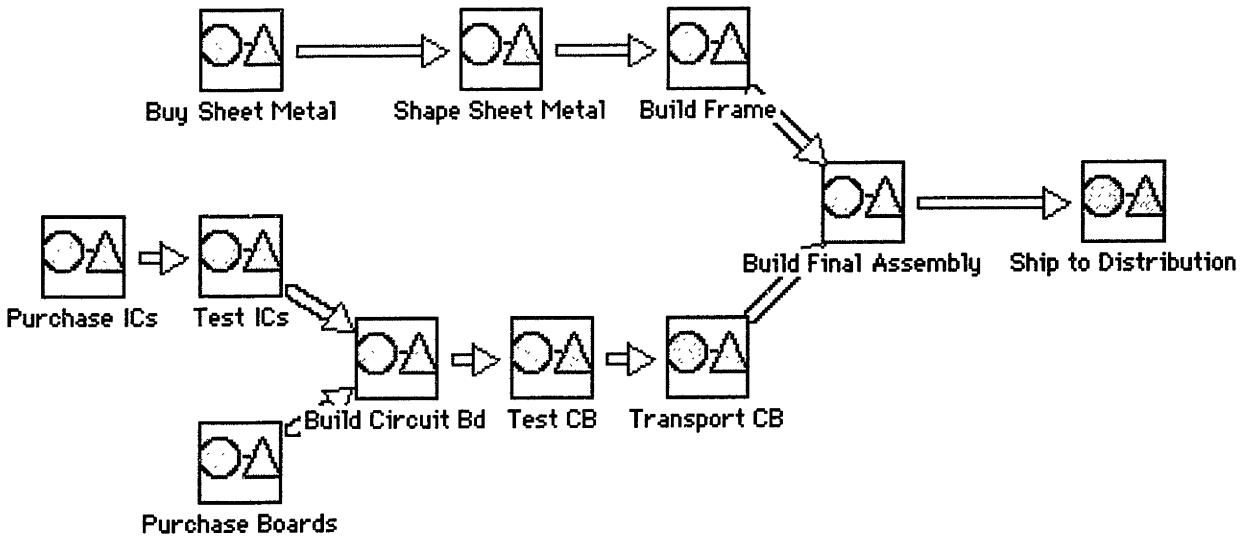


**Figure 3-1: Example of a typical assembly network**

## 3.1 Model Assumptions And Notation

This section only includes notation and assumptions that differ from the serial network case. If the assumptions do not differ, i.e. the inventory control policy, then they have not been repeated.

### 3.1.1 Network representation

We can represent an assembly network by a graph G where N(G) is the node set and A(G) is the arc set. There is a one-to-one mapping between the stages of the supply chain and the nodes in N(G). There is an arc between node i and node j, i. e., $(i, j) \in A(G)$, if and only if stage i is a direct supplier to stage j.

### 3.1.2 Demand characterization

Since there is only one outgoing arc per stage, there is only one point of contact to the final customer. Therefore, as in the serial case, we assume that demand each period is an independent normally-distributed random variable with mean $\mu$ and standard deviation $\sigma$.

### 3.1.3 Service time

For assembly networks, since it is possible to have more than one incoming arc, it is possible for more than one service time to be quoted to a stage. And for more complicated networks like distribution and general networks, it is possible for a stage to have to quote a service time to more than one stage. To easily discern the meaning of service times in these more general cases, we define $S_{ij}$ as the service time that stage i quotes to stage j. As in the serial line case, the service times $S_{ij}$ will be the decision variables for the optimization problem. Mapping our new notation to the serial line case, what is now $S_{i,i+1}$ would have been $S_i$ in Chapter 2; thus we have made the notation more specific, but have in no way changed the definition of service time.

This change in notation does create some ambiguity concerning how to define service times for nodes that have an outdegree of zero; i.e., the stage that serves customer demand in the assembly network case. For example, if node i has an outdegree of zero, there is no j such that $(i,j) \in A(G)$ and hence $S_{ij}$ is not properly defined. To get around this notational difficulty, we can assume that any node i with outdegree zero has an arc to a dummy node N+1, where $(i,N+1) \in A(G)$ but $N+1 \notin N(G)$. This introduction of a dummy node properly defines nodes with an outdegree of zero by adding an arc to the arc set while leaving the node set unchanged.

### 3.1.4 Replenishment lead-time

For each stage i, we define $L_i$ to be replenishment lead-time for stage i; since there may be several upstream suppliers to stage i, we note that the replenishment lead-time equals the production lead-time at stage i, plus the longest service time of its suppliers: $L_i = T_i + \max \{S_{ji}\}$ where the maximization is over all arcs (j, i) $\in$ A(G).

This derivation of replenishment lead-time assumes that production can not begin at a stage until all of the parts that the stage requires are available for consumption by the stage.

## 3.2 Inventory calculations

### 3.2.1 Holding cost calculation

As in the serial case, we let $v_i$ denote the direct cost added at stage i. For assembly networks, we modify the calculation of $V_i$, the cumulative cost up to and including stage i, to reflect the fact that there can be multiple stages that supply stage i. In the case of multiple stages supplying stage i, the total cost up to and including stage i is the sum of stage i's direct costs plus the total cost up to and including each stage that is immediately upstream to stage i. This is represented in equation (5):

$$V_i = v_i + \sum_{\{j:(j,i)\in A(G)\}} V_j \qquad (5)$$

This is equivalent to adding together the cost at stage i and the costs at all other stages that are upstream to stage i.

If we assume a holding cost rate of $\alpha$ then $h_i$, the holding cost at stage i, equals $\alpha V_i$.

### 3.2.2 Safety stock calculation

The net coverage time for a stage is the number of time units that the stage's safety stock will have to cover. Given our change in notation for service times, the

net coverage time for a stage is just $L_i - S_{ij}$ for j such that (i,j) $\in$ A(G). Recall that since the network is assumed to be an assembly, there is only one j such that (i,j) $\in$ A(G).

Letting $I_i$ denote the expected amount of safety stock held at stage i, $I_i = k_i\sigma\sqrt{L_i - S_{ij}}$ for (i,j) $\in$ A(G).

## 3.3 Math Programming Formulation

The mathematical programming formulation for the assembly network case is given by:

$$\min \sum_{i=1}^{N} h_i I_i$$

s.t.

$$\begin{aligned} I_i &= k_i\sigma\sqrt{L_i - S_{ij}} && \forall\, i : i \in N(G), j : (i,j) \in A(G) \\ L_i &\geq S_{ji} + T_i && \forall\, (j,i) \in A(G) \\ 0 &\leq S_{ij} \leq L_i && \forall\, (i,j) \in A(G) \end{aligned} \tag{6}$$

The objective function minimizes the total inventory holding cost across the supply chain. The first constraint equates the inventory at stage i to the safety stock requirements at stage i given an outgoing service time of $S_{ij}$ (recall that there in only one outgoing arc per stage in an assembly network) and a replenishment lead-time $L_i$. The second set of constraints requires the replenishment lead-time at stage i to be greater than or equal to the production lead-time at stage i plus the maximum service time quoted to stage i. The third set of constraints forces all service times to be nonnegative and less than the stage's replenishment lead-time. The first constraint set contains N constraints. The second and third constraint sets each contain $|A(G)|$ constraints.

## 3.4 Solution Procedure

Section 3.4.1 presents the dynamic programming formulation for the math program presented in Section 3.3. In Section 3.4.2, we discuss how to implement the dynamic program. Section 3.4.3 proves the equivalence of the dynamic program and the math programming formulation of the problem. Section 3.4.3 also defines an equivalent dynamic programming formulation to the formulation in Section 3.4.1 that takes advantage of the special structure of assembly networks.

### 3.4.1 Dynamic programming formulation

The assembly network case can be solved to optimality by solving the following dynamic program for each node in the network:

$$f_j(\beta) = \min_{\substack{0 \le \alpha_{ij} \le D_i + T_i \\ [\beta - T_j]^+ \le \max_{(i,j) \in A(G)} \{\alpha_{ij}\}}} \left\{ c_j\left( \max_{\{i:(i,j) \in A(G)\}} (\alpha_{ij}), \beta \right) + \sum_{\{i:(i,j) \in A(G)\}} f_i(\alpha_{ij}) \right\} \quad \begin{matrix} j \in N(G) \\ 0 \le \beta \le D_j + T_j \end{matrix} \qquad (7)$$

where $f_j(\beta) = c_j(0, \beta)$ if j has no predecessors

where $D_j$ is the largest possible service time that can be quoted to stage j. For an assembly network, $D_j$ can be found by creating a subgraph comprised of all nodes contained in the directed in-tree rooted at node j. [Ahuja *et al.*, 1993]. For each node in the subgraph that has an indegree of zero, we add all of the production lead-times on the path from the node to node j; since the network is an assembly network, there can be only one path from each node to node j. The maximum value over all paths emanating from nodes with indegree zero is $D_j$.

Since there can be multiple nodes that are immediately upstream of node j, there are restrictions on the individual values of each service time and a restriction on the set of service times. On an individual basis, each $\alpha_{ij}$ must be non-negative and less than the maximum possible service time that node i can quote; for each upstream node i, this upper bound equals $D_i + T_i$. We also require at least one of the

$\alpha_{ij}$ to be greater than or equal to $[\beta - T_j]^+$. If this condition is not met, node j is quoting a service time $\beta$ larger than the time it takes to produce the part, $T_j$, plus the longest procurement time, max $(\alpha_{ij})$ for all i such that $(i,j) \in A(G)$, to receive its supply. Since this violates an initial assumption we have made concerning the range for service times, the constraint on the set of $\alpha_{ij}$ prevents this.

The cost-to-go function $f_j(\beta)$ is the minimum holding cost for the inventory at stage j and all upstream stages to stage j, given that stage j quotes a service time equal to $\beta$. Within the cost-to-go function, $c_j(\alpha, \beta)$ denotes the inventory holding cost at stage j, given that the longest supplier service time is $\alpha$ and stage j quotes a service time of $\beta$; in the dynamic program formulation $\alpha = \max \{\alpha_{ij}\}$ for all i such that $(i,j) \in A(G)$ and $\beta = S_{jk}$. For a given $\beta$, the algorithm loops over all possible incoming service times and finds the minimum cost solution for stage j and its suppliers. The cost-to-go function needs to be evaluated for all possible values of $\beta$. For each stage in the network, the maximum service time a stage can quote is the maximum service time that can be quoted to the stage plus the stage's production lead-time. Using the notation we have defined earlier in this section, the maximum service time stage j can quote is $D_j + T_j$.

### 3.4.2 Dynamic program implementation

The dynamic program in (7) can be implemented as follows: Find the node with no outgoing arcs and label it node N. Then use a standard labeling algorithm, like breadth first search, to assign every other node a unique label from 1 to N-1. At the same time, determine the depth of each node; the depth of node i is the number of arcs that needs to be traversed to create a path from i to N.

Once the depth of each node in the network is known, start with the nodes having the greatest depth and solve equation (7). Repeat this for every depth,

starting at the greatest depth and working towards node N, which by construction has a depth of zero.

Since the network is assumed to be an assembly network, it follows directly from the definitions of an assembly network and depth that if (i,j) ∈ A(G) then the depth of i equals the depth of j plus one. Therefore, whenever a node j is reached, we know by construction of the algorithm that all its upstream nodes have already been solved.

### 3.4.3 Correctness of the dynamic program's solution

First, note that (7) involves the minimization over each inbound arc. Potentially, this could be a lot of work. Fortunately, we can simplify (7) to (7a)

$$f_j(\beta) = \min_{[\beta - T_j]^+ \le \alpha \le D_j} \left\{ c_j(\alpha,\beta) + \sum_{\{i:(i,j)\in A(G)\}} f_i(\alpha) \right\} \qquad \begin{matrix} j \in N(G) \\ 0 \le \beta \le D_j + T_j \end{matrix} \tag{7a}$$

where the minimization is over one variable $\alpha$ and where we define

$$f_i(\alpha) = f_i(D_i + T_i) \qquad \alpha > D_i + T_i$$

If stage i has no incoming arcs, then we can observe that $f_i(\alpha) = c_i(o,\alpha)$ and thus $f_i(\alpha)$ is non-increasing in $\alpha$. In (7), suppose that $f_i(\alpha)$ are non increasing for all predecessors to j. Then in (7), we can make two observations. First, $\alpha_{ij} = \alpha^*$ for all i such that (i,j) ∈ A(G). This can be argued by contradiction. If this were not true, then $\alpha_{ij} < \alpha^*$ for some i. This $\alpha_{ij}$ can then be increased to $\alpha^*$ without increasing $f_j(\beta)$. Thus, there is an optimal solution with $\alpha_{ij} = \alpha^*$ for all (i,j) ∈ A(G). Second, $f_j(\beta)$ is non-increasing in $\beta$. If $\alpha^* > \beta - T_j$, then $f_j(\beta)$ is non-increasing since the only term that will change is the $\beta$ in $c_j(\alpha,\beta)$ and we have already observed that this function is non-increasing. If $\alpha^* = \beta - T_j$, then $f_j(\beta)$ is also non-increasing since $c_j(\alpha,\beta) = 0$ and the $f_i(\alpha)$ have been assumed to be non-increasing in $\alpha$.

This completes our proof by induction that for any stage in the network, $f_j(\beta)$ is non-increasing in $\beta$.

# 4 Distribution Networks

A distribution network is an acyclic graph in which each node can have only one incoming arc but one or more outgoing arcs. A typical distribution network is shown in Figure 4-1:

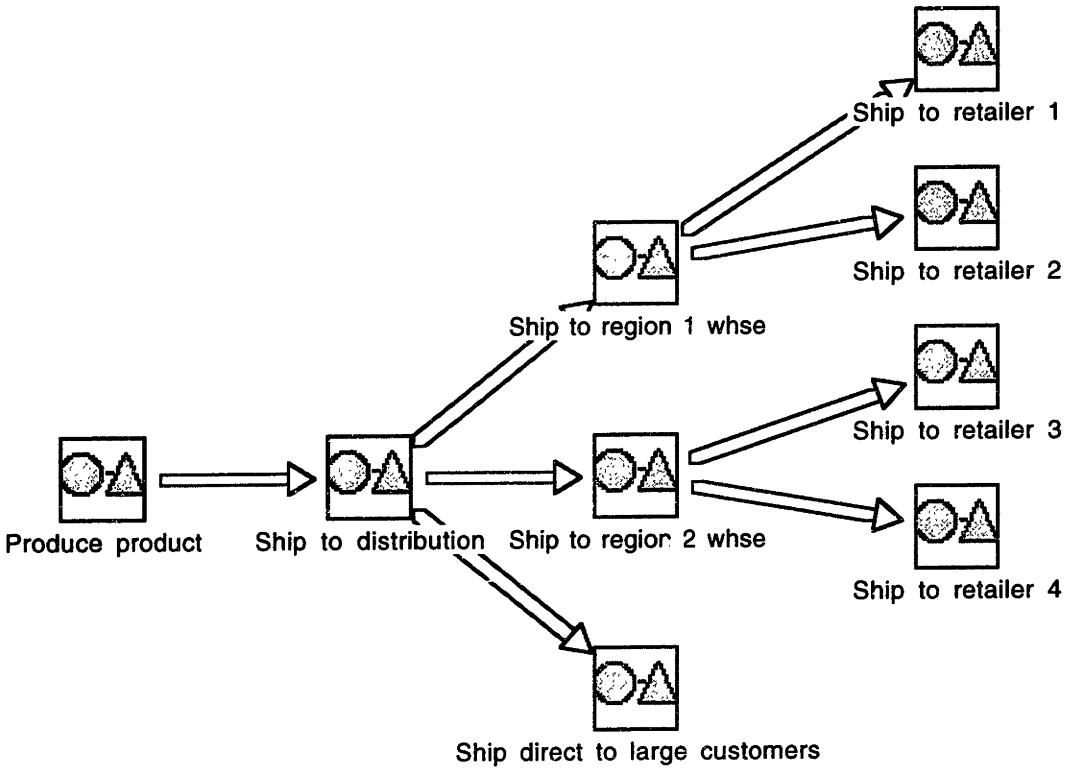


**Figure 4-1: Depiction of a typical distribution network**

## 4.1 Model Assumptions And Notation

### 4.1.1 Network representation

We will use the same network terminology and definitions used for the assembly network case. We can represent an assembly network by a graph G where N(G) is the node set and A(G) is the arc set. There is a one-to-one mapping between the stages of the supply chain and the nodes in N(G). There is an arc between stage i and stage j, i. e., (i, j) $\in$ A(G), if and only if stage i is a direct supplier to stage j.

Furthermore, let F(i) denote the forward arc adjacency matrix of i. Symbolically, this is defined as $F(i) = \{j: (i,j) \in A(G)\}$ for all nodes i in N(G).

### 4.1.2 Demand characterization

In a distribution network, unlike a serial line or an assembly network, there will be more than one stage that directly serves customer demand. This is due to the fact that each stage has at most one supplier but each stage may now feed several downstream stages. In the case when a stage is producing product for more than one demand origin, the stage will see the convolution of several demand streams and will need to have safety stock to protect against the variation in aggregate demand. Therefore, the notation for the standard deviation of demand, and hence calculation of safety stock, will be slightly more complex than the notation presented in the previous two chapters.

Now the problem is how to characterize demand seen by stage i, where $|F(i)| \geq 1$. For nodes with $|F(i)| = 0$, we assume they see normal, independent, identically distributed demand with parameters $\mu_i$, $\sigma^2_i$.

We denote $\sigma^2_{ij}$ as the one period variance of demand stage i sees from stage j. If we denote $\sigma^2_i$ as the one period variance of demand at stage i, we can calculate $\sigma^2_{ij}$ and $\sigma_i$ as follows:

$$\sigma^2_{ij} = \sigma^2_j \qquad \forall (i,j) \in A(G) \tag{9}$$

Equation (9) states the demand variance stage i sees from stage j is equal to the total demand variance of stage j. Now,

$$\sigma^2_i = \sum_{(i,j)\in A(G)} \sigma^2_{ij} \qquad \text{if } |F(i)| \geq 1 \tag{10}$$

Equation (10) states that the one period standard deviation of demand is equal to the square root of the summation of the variances from all the stage's forward adjacent nodes.

For these formulas to be correct, we once again assume that demand each period is an independent identically-distributed random variable with finite mean and variance. However, we now have the added assumption that demand between different demand origins are independent from each other. This is a necessary assumption to preclude the case where the mean and variance at different stages could be correlated.

### 4.1.3 Stage cost calculation

The recursion to determine the cumulative cost up to and including stage i is similar to equation (5) in the assembly case, but simpler since there is only one upstream stage to any stage. Therefore, $V_i = v_i + V_j$ where $(j,i) \in A(G)$.

If we assume a holding cost rate of $\alpha$ then $h_i$, the holding cost at stage i, equals $\alpha V_i$.

### 4.1.4 Safety stock calculation

As in the assembly network case, we define $S_{ij}$ as the service time that stage i quotes to stage j. For each stage i, we define $L_i$ to be replenishment lead-time for stage i. As in the serial case, there can only be one upstream supplier to each stage i, so the replenishment lead-time equals the production lead-time at stage i plus its supplier's service time: $L_i = T_i + S_{ji}$ where $(j, i) \in A(G)$.

Let $I_i$ denote the amount of safety stock held at stage i. Then based on the notation specified in the previous sections, $I_i = k_i \sqrt{\sum_{j:(i,j)\in A(G)} (L_i - S_{ij})\sigma_i^2}$.

## 4.2 Math Programming Formulation

The mathematical programming formulation for the distribution network case is given by:

$$\min \sum_{i=1}^{N} h_i I_i$$

s.t.

$$I_i = k_i \sqrt{\sum_{(i,j)\in A(G)} (L_i - S_{ij})\sigma_{ij}^2} \qquad \forall\, i : i \in N(G), j \in F(i) \tag{12}$$

$$L_i = S_{ji} + T_i \qquad \forall\, i : i \in N(G), (j,i) \in A(G)$$

$$0 \le S_{ij} \le L_i \qquad \forall\, (i,j) \in A(G)$$

The objective function minimizes the total inventory holding cost across the supply chain. The first constraint calculates the safety stock requirements for stage i by multiplying the safety factor at stage i times the standard deviation over stage i's coverage time given that stage i quotes service times of $S_{ij}$ to each forward adjacent stage j. The second constraint requires the replenishment lead-time at stage i to be greater than the production lead-time at stage i plus the service time quoted to stage i. The third set of constraints forces the stage's service times to be nonnegative and less than the stage's replenishment lead-time. The first constraint set contains N constraints. The second and third constraint sets each contain $|A(G)|$ constraints.

## 4.3 Solution Procedure

For the special case where we assume that each stage specifies the same service time to each of its downstream customers, we can formulate a dynamic program similar in spirit to the formulation given in Chapter 3 for assembly networks. In particular, the dynamic programming recursion is a "mirror image" of that for the assembly network. In the assembly system the recursion finds the best possible service time to quote the current stage for each possible service time that the current stage can quote. In the distribution network case, the algorithm does the exact opposite: the algorithm loops over each possible service time that can be quoted to the current stage and finds the best possible service time that the current stage can quote. This is made more rigorous in the following formulation:

$$f_i(\alpha) = \min_{0 \le \beta \le \min(\alpha+T_i, U_i)} \left( c_i(\alpha,\beta) + \sum_{\{j:(i,j)\in A(G)\}} f_j(\beta) \right) \qquad i \in N(G),\ 0 \le \alpha \le D_i \tag{12}$$

where $f_i(\alpha) = c_i(\alpha, 0)$ for j with no successor

where $\alpha$ denotes the service time that is being quoted to stage i and $\beta$ is the service time that stage i quotes to all its adjacent downstream stages j. If $U_i$ is infinity, then for a given choice of $\alpha$ we can minimize over all values of $\beta$ that are bounded by zero and $\alpha + T_i$. This is just imposing the math programming constraint $0 \le S_{ij} \le L_i$ into the dynamic program. Let $U_i$ represent a user-defined maximum possible service time that stage i can quote. It is necessary to specify $U_i$ because if $U_i$ does not exist, then for each incoming service time $\alpha$ quoted to stage i, stage i would quote a service time to $T_i + \alpha$. The use of $U_i$ for those stages that are demand origins prevents this uninteresting case from being realized.

### 4.3.1 Implementing the dynamic program

The dynamic program in (12) can be implemented as follows: Find the node with no incoming arcs and label it node 1. Then use a standard labeling algorithm, like breadth first search, to assign every other node a unique label from 2 to N. At the same time, determine the depth of each node; the depth of node i is the number of arcs that need to be traversed to create a path from 1 to i.

Once the depth of each node in the network is known, start with the nodes having the greatest depth and solve equation (12). Repeat this for every depth, starting at the greatest depth and working towards node 1, which by construction has a depth of zero.

Since the network is assumed to be a distribution network, it follows directly from the definitions of a distribution network and depth that if $(i,j) \in A(G)$ then the depth of j equals the depth of i plus one. Therefore, whenever a node i is reached, we know by construction of the algorithm that all its downstream nodes have already been solved.

# 5 Integrated Production-Distribution Networks

An integrated production-distribution network is a supply chain composed of a pure assembly network feeding a pure distribution network. An example of a production-distribution network is shown below in Figure 5-1:

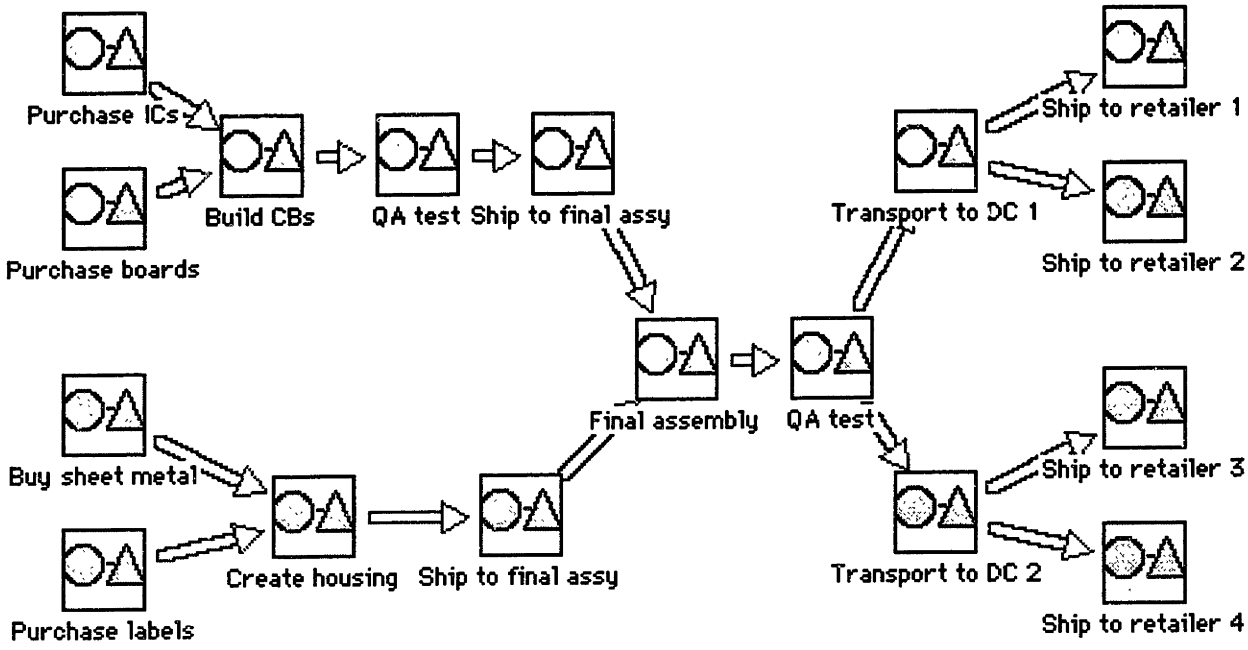


**Figure 5-1: Example of typical production-distribution network**

We can recognize that networks of the type shown in Figure 5-1 have a very special structure. Namely, they can be split into two separate subgraphs; one of the subgraphs is a pure assembly network while the other subgraph is a pure distribution network. Therefore, once the two subgraphs are identified, we can solve the assembly network using the techniques developed in Chapter 3 and we can use the techniques from Chapter 4 for the distribution subgraph.

## 5.1 Identification of subgraphs

For each production-distribution network, there is at least one node that marks the end of the assembly network and the beginning of the distribution network.

That is, we can find a node that forms a directed in-tree for the assembly subgraph and a directed out-tree for the distribution subgraph; we call this node the anchor node.

The anchor node need not be unique. For example, in Figure 5-1, both Final assembly and QA test could be the anchor node. However, the algorithm we will present will always find the anchor node that gives the assembly network the maximum depth over all choices for the anchor node; in Figure 5-1, this is the QA test node.

To find the anchor node, start with any node that has no incoming arcs. Check to see if the node has more than one outgoing arc. If it does, this node is the anchor and the algorithm terminates. If not, move to the node that the current node is adjacent to; i.e., there must be an arc emanating from the current node so the algorithm now moves from the tail node of the arc to the head node. Repeat the procedure of first verifying the number of outgoing arcs and then labeling the current node as the anchor node or moving on to the next node. Since we assume that the network is the combination of an assembly network and a distribution network, we know that the first node reached that has more than one outgoing arc must be the beginning of the distribution subgraph.

It is worth mentioning that if the network is a serial line, then the algorithm will terminate without finding an anchor node. In this case, we can use any solution procedure that we have previously described in the thesis.

## 5.2 Solution procedure for production-distribution networks

Assume the anchor node has been determined and it is labeled node N. We can then separately solve the assembly subgraph and distribution subgraph rooted at the anchor node. Let $f_a(\alpha)$ equal the minimum cost for the assembly subgraph with the anchor node quoting a service time of $\alpha$, and let $f_d(\alpha)$ equal the minimum cost for

the distribution subgraph given the anchor node quotes a service time of $\alpha$. To find the minimum cost of the supply chain, we choose $\alpha$ to minimize:

$$\min_{0 \le \alpha \le D_N + T_N} \{f_a(\alpha) + f_d(\alpha)\}$$

Thus, once each subgraph is solved separately, we only need to loop over all the possible service times for the anchor node, adding the results from each subgraph and choosing the minimum.

# 6 Case Studies

Two companies have tested and validated the models that have been presented in this thesis. This chapter describes two of the supply chains that have been analyzed with the model. Section 6.1 presents a brief overview of the software application we have developed. Both of the companies that we refer to in this chapter have used the software application on-site.

The first example, presented in Section 6.2, describes the use of the model at a specific supply chain in the equipment division of the Eastman Kodak Company. The department has used the model to determine the location and size of safety stocks in the company's internal supply chain.

Section 6.3 describes the use of the tool at a communications equipment company that we will refer to as Company A. Company A is using the model to look at reducing redundant inventories between the company and one of its major customers.

The data for both cases has been disguised in order to protect any proprietary information. However, the altered data still captures the key characteristics of each supply chain, as well as the recommendations that were made to the companies.

## 6.1 Software application background

The model has been programmed in C for the Macintosh. The graphical interface allows the user to quickly construct various supply chain configurations. The program allows the user to either use the program as an optimizer that determines optimal service times or as a calculator where the user can enter service times and see what impact the inputs have on the total inventory costs for the network.

A full-screen display of the supply chain for the product we will be analyzing in Section 6.2 is shown below:

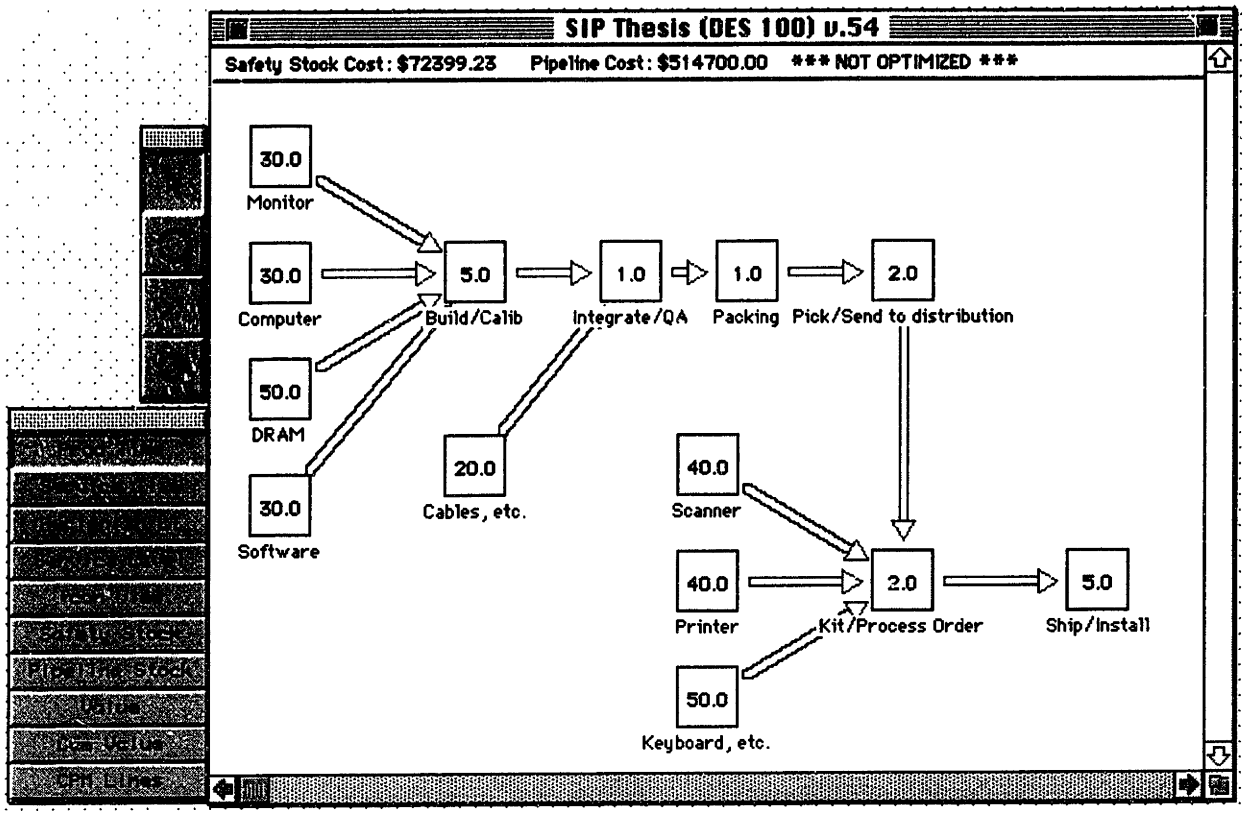


**Figure 6-1: Full-screen display of a typical supply chain**

The graphical interface contains two tool palettes. The four-button tool palette (top left) allows the user to quickly add and delete stages and arcs between stages; this is the tool palette that creates and modifies the network. The second tool palette allows the user to change the current label for each stage; the term label refers to the information displayed in the stage's box. The labels on the tool palette correspond to the inputs and outputs that we have already discussed in the thesis. For example, Figure 6-1 shows a supply chain where the production lead-time for each stage is displayed. Finally, the user changes a stage's inputs by double-clicking on the stage's icon and editing the inputs directly.

The user-friendly nature of the software has been a critical success factor in the case studies to date.

## 6.2 Kodak Digital Enhancement Station 100

### 6.2.1 Product background

This section presents an analysis of the Digital Enhancement Station 100 (DES 100) supply chain at the Eastman Kodak Company.

The DES 100 is used to digitize photographs and then transform the digital image in various ways. The picture is scanned into the computer where it can be edited. Typical editing functions include removing red-eye from the photograph, cropping objects out of the photograph, changing the background in a photograph, and printing out enlargements of the original photograph. The digital image can then be printed out using a photograph-quality color thermal printer.

The DES 100 provides a complete turn-key solution for its customers. The DES 100 includes the computer and monitor to store and view the image, the scanner to input the image, the printer to produce the high quality output, and the software to manipulate the digitized image.

Typical DES 100 customers are one hour photo labs and corporate graphic design teams. The one hour photo lab can use the DES 100 to reconstruct damaged photographs or to reproduce a photograph when the negative is no longer available. The corporate design staff can use the DES 100 to modify photographs to make brochures more appealing.

### 6.2.2 DES 100 supply chain

Figure 6-2 presents the product flow map for the DES 100 supply chain:

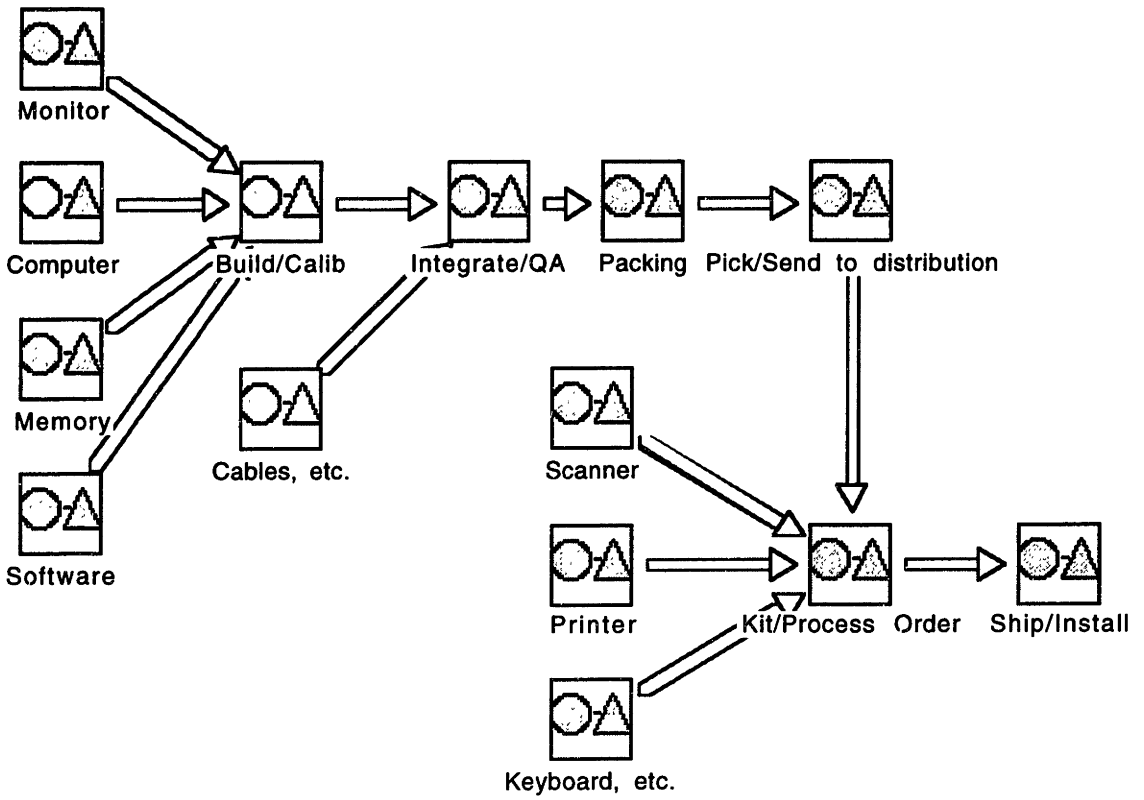


**Figure 6-2: DES 100 supply chain**

This sixteen-stage supply chain is a high level view of the supply chain but for the purposes of the model it captures the key drivers of the supply chain. Stages without incoming arcs are procurement functions; for example scanner represents the work involved in placing an order for a scanner plus the time it takes to receive the scanner once the order has been placed. Stages with both an incoming and an outgoing arc represent processing functions that occur at Kodak's production facility. The stage without an outgoing arc is the point of contact to the customer.

The flow through the supply chain can be described as follows: The monitor, computer, memory and software are taken from an inventory vault and brought to the build/calibrate station. At the build/calibrate station, memory and digital imaging software are added to the computer's base configuration. The computer and monitor are calibrated to ensure that their colors match perfectly. Once the calibration process is completed, the computer and monitor that were calibrated

together will not be separated from one another. The computer/monitor pairing is then tested by quality control. It is then packed and shipped to a central distribution center at Kodak. When an order is received by the central warehouse, the warehouse picks a printer, scanner, country-specific keyboard and manual as well as a calibrated monitor/computer pairing and ships them to the customer location. Ship/install represents the final step in the supply chain. At this point, a Kodak technician travels to the customer to install the DES 100 and to provide the customer with any necessary training on how to use the product.

For the DES 100, only the build/calib, integrate/QA and packing stages are manufacturing functions that represent assembly operations. Several stages represent moving the inventory from one location to another and eight of the stages represent the procurement of materials.

It is also important to note that this product flow encompasses all of the stages that the assembly department has direct control over. Some of the products that are procured are actually purchased from other departments at Kodak. For the first phase of this model's use, the assembly department only wanted to consider those stages that were under its direct control.

### 6.2.3 Model inputs

For the purpose of this thesis, we are defining every input in terms of a common planning period. It might be helpful to think of the planning period as a day or a week but in reality the planning period is a fictitious time unit that we have used to help us disguise the data.

#### 6.2.3.1 Global parameters

The expected demand per time unit is 20 units with a standard deviation of 4. The holding cost rate is assumed to be 20%.

6.2.3.2 Stage parameters

Figure 6-3 shows the production lead-time for each stage.

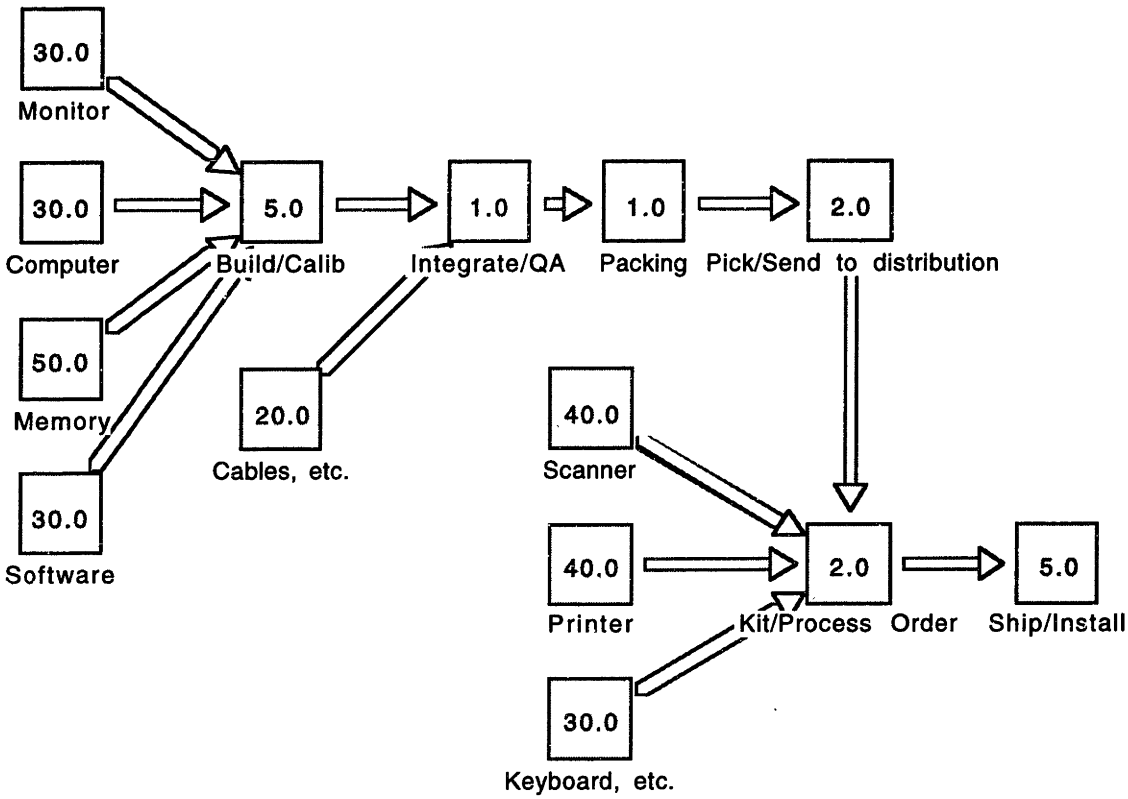


**Figure 6-3: Production lead-times for DES 100**

For the stages that represent procurement functions, it is helpful to think of the production lead-time as procurement lead-time since this is what the majority of the time at each procurement stage is devoted to; for example, it might only take one time unit for a materials management buyer to place an order for memory but because memory is a part that is on allocation, the vendor will quote a delivery time 49 time units into the future.

From the data in Figure 6-3, it is clear that the procurement function will have a big impact on the performance of the supply chain. Procurement times for parts like memory exceed the production lead-time of the entire assembly process by a factor of five.

The following figure displays the direct costs added at each stage:

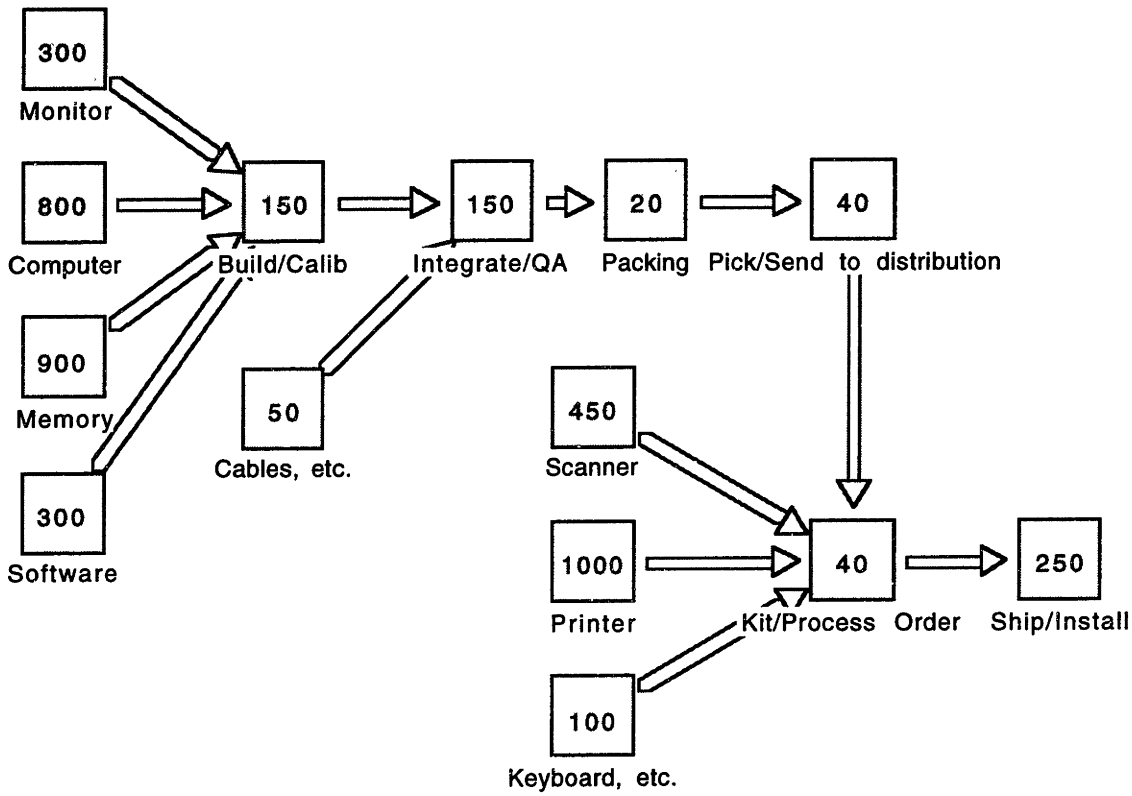


**Figure 6-4: Cost added per stage for DES 100**

From the cost added information, purchased products account for nearly 88% of the product's direct costs.

Each stage in the supply chain plans its inventory levels assuming the stage will provide a 95% service level.

Finally, when a customer orders a product, the maximum service time they will find acceptable is seven time units. The reason that customers do not want shipment immediately is that they usually have to put some infrastructure in place before the DES 100 arrives.

### 6.2.4 Model analysis

#### 6.2.4.1 Current inventory policy

When we started working with the DES 100 supply chain team, they had just begun a more formal analysis of where to place safety stocks in the supply chain. The current inventory policy held finished computer/monitor sets after the packing operation and at the distribution center.

Inventory levels at both of these locations had been set primarily based on heuristics. Between the two safety stock locations, the system planned for a total of 40 computer/monitor pairs in the system.

#### 6.2.4.2 Optimal inventory policy

This section will discuss the optimal solution to the DES 100 supply chain. Since this was not the solution that was implemented, this section is mainly concerned with developing some intuition for the structure of the optimal solution. The next two figures show the service times and resulting safety stock inventories that result from optimizing the service times of the DES 100 supply chain.

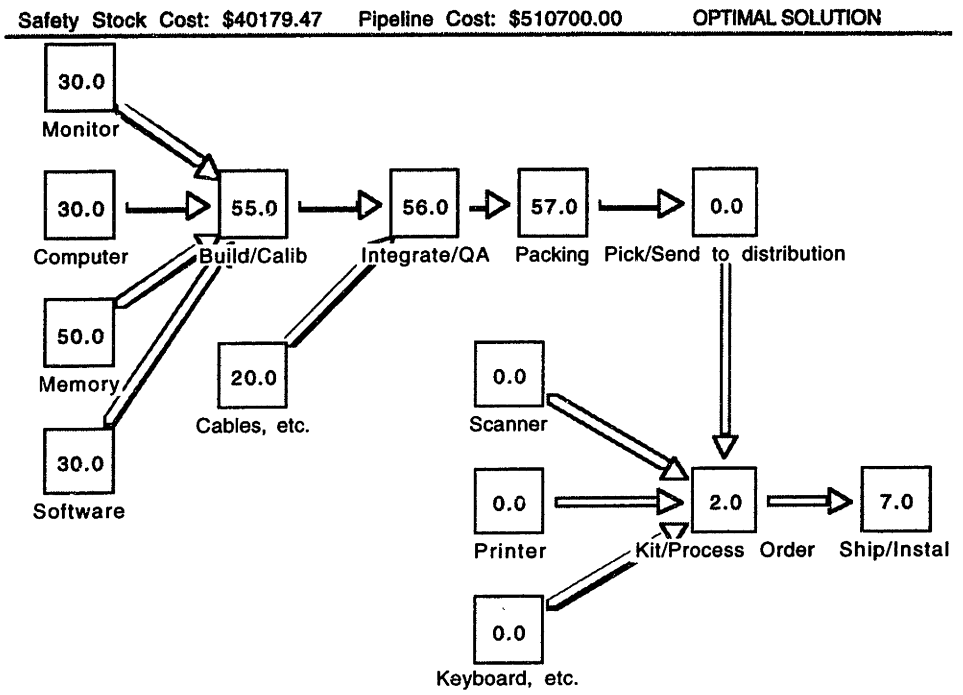


**Figure 6-5: Service times for the optimal solution**

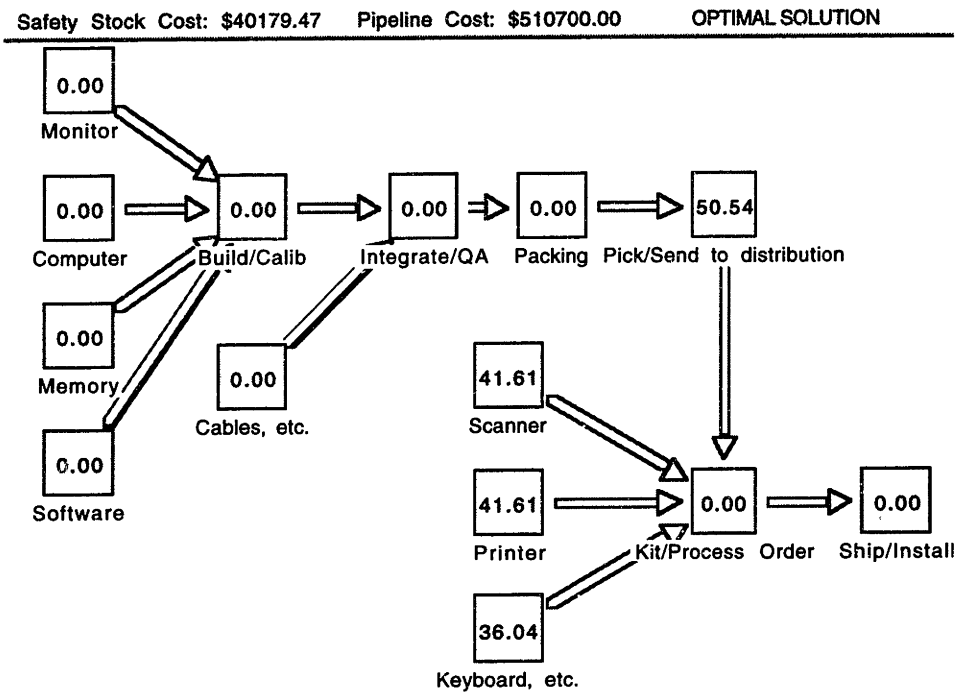


**Figure 6-6: Safety stock inventories for the optimal solution**

Thus, the optimal solution is to hold a safety stock of completed computer/monitor sets as well as a safety stock for all of the key bundled components like the scanner and printer.

There is some intuition for this optimal solution. First, note that kit/process order and ship/install both hold no inventory; that is, they each quote their maximum service time given the service times that are quoted to kit/process order. Since both stages can quote a service time that requires them to hold no inventory while still meeting the maximum service time constraint to the final customer, it makes economic sense for these stages to hold no inventory.

Now to understand the rest of the service times in the supply chain, we first consider the case where the supply chain has been consolidated as follows:

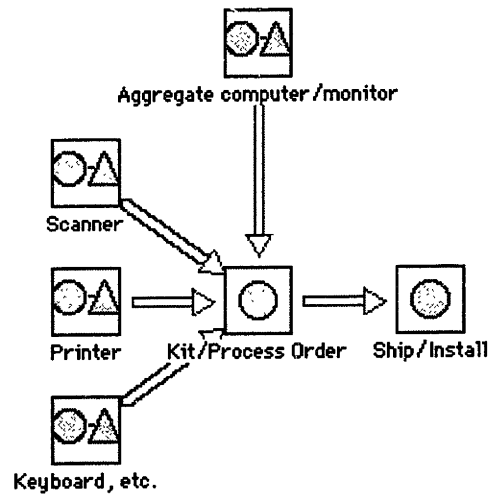


**Figure 6-7: Consolidated supply chain**

In this supply chain, the aggregate computer/monitor represents a consolidation of all the stages in the original supply chain that were required to produce the computer/monitor pairing; i.e., this stage is the consolidation of all the stages that are in the original supply chain but are absent from the supply chain shown in Figure 6-7. Computing the cost added at the aggregate stage as the sum of the costs from the stages that have been consolidated, the cost added is $2710. The production

lead-time is the longest time path in the original time path up to and including the pick/send to distribution stage; this is equal to 59 time units.

Therefore, in this consolidated supply chain, the aggregate computer/monitor has a longer lead-time than any of the other three component stages and a cost added that is greater than the summation of the three stages.

This insight can be mapped back to the original supply chain as follows: The computer/monitor portion of the supply chain dominates the other purchased components. Therefore, we can think of the optimization as first optimizing the computer/monitor section of the supply chain plus the kit/process order and ship/install stages. Once the service times for these stages have been determined, the service times for the remaining stages are chosen such that their costs are minimized while not increasing the costs of the stages whose service times have already been optimized.

#### 6.2.4.3 Implemented inventory policy

Two members of the DES 100 supply chain team spent several months becoming familiar with the model's assumptions in an effort to understand how the make the model of use to them. After they understood the model, they felt comfortable enough with the software to help them determine a more optimal stocking strategy for the DES 100.

Although they understood the results from optimal solution, they felt that stocking fifty DES 100s did not make sense given two real-world considerations not captured by the model; the model does not incorporate depreciation or lot sizing effects when ordering raw materials.

Therefore, they used the model to look at the case where all raw materials were stocked in inventory; i.e., all procurement stages quote a service time of zero.

The resulting optimal service times and resulting safety stock inventories are shown in the next two figures.

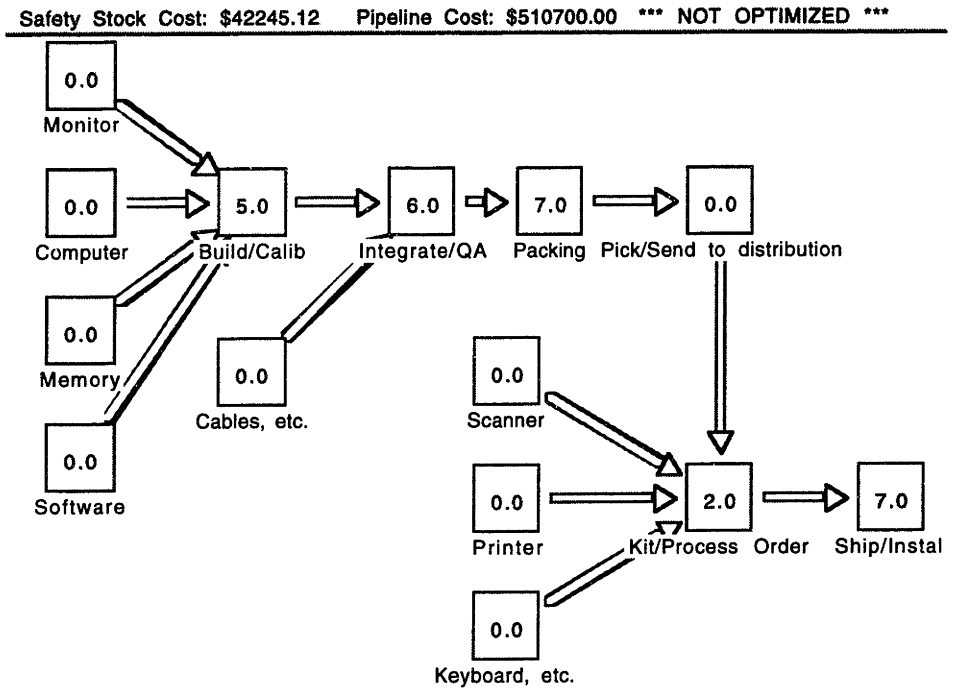


**Figure 6-8: Service times for implemented policy**

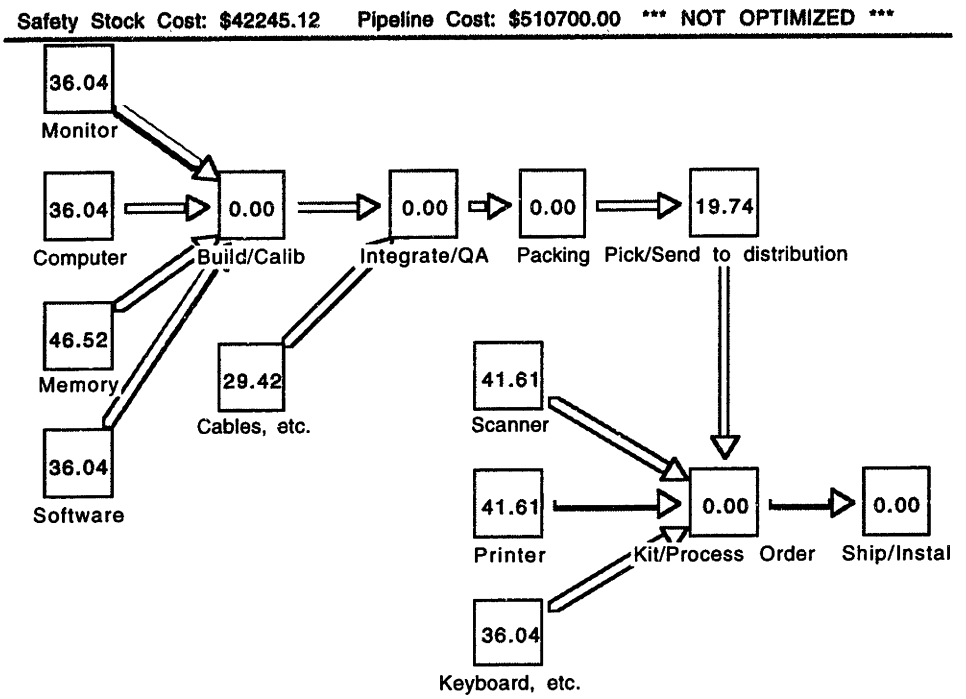


**Figure 6-9: Safety stock inventories for implemented policy**

By implementing this policy, the optimal safety stock strategy is to hold all completed components required to make a DES 100 at the distribution center. When a computer/monitor set is consumed, a production order is initiated to pull another set through the assembly process.

The size of the inventory is half the 40 units the supply chain team had initially predicted it would need. The system has been running at 20 units of safety stock for several months now without any problems.

## 6.3 Company A's intercompany supply chain

### 6.3.1 Product background

Company A makes equipment for switching networks. Company A makes the equipment for several other companies, but has a primary customer we will call Company B. We will refer to the product Company A produces as part XYZ.

### 6.3.2 Part XYZ supply chain

The supply chain for part XYZ is shown in Figure 6-10:

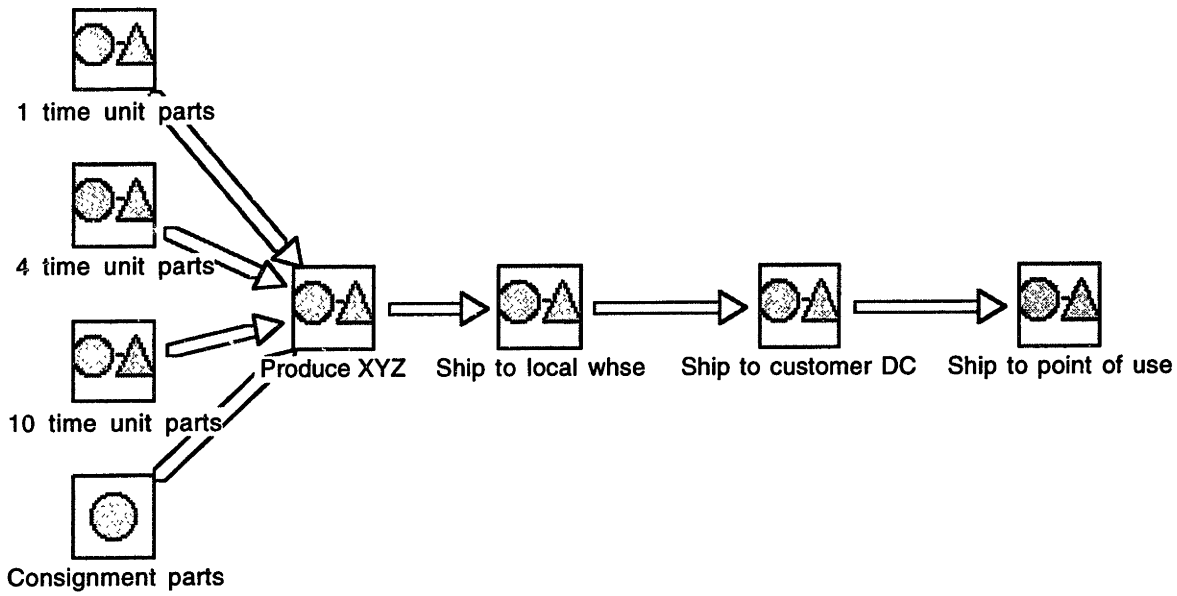


**Figure 6-10: Product flow map for part XYZ**

Raw materials for the product are grouped according to procurement lead-times. Parts are either on consignment at Company A or they take one, four, or ten time units to arrive once they are ordered. When a production run is initiated, the parts are pulled from an inventory vault and produced in an assembly operation at Company A's production facility. Once the part is completed, it is sent to a local warehouse owned by Company A. Thus, the supply chain from raw materials up to and including ship to local whse is owned by Company A. When Company B places an order for part XYZ, the product is first shipped from Company A's warehouse to a central distribution center owned by Company B. When the part is actually needed at the point of use, Company B ships the part from the central distribution center to point of use.

### 6.3.3 Model inputs

#### 6.3.3.1 Global parameters

The expected demand per time unit is 30 units with a standard deviation of 7. The holding cost rate is assumed to be 20%.

#### 6.3.3.2 Stage parameters

For each stage, the service level is set at 99.8% for each stage in the supply chain. Production lead-times and cost added are displayed in the next two figures.

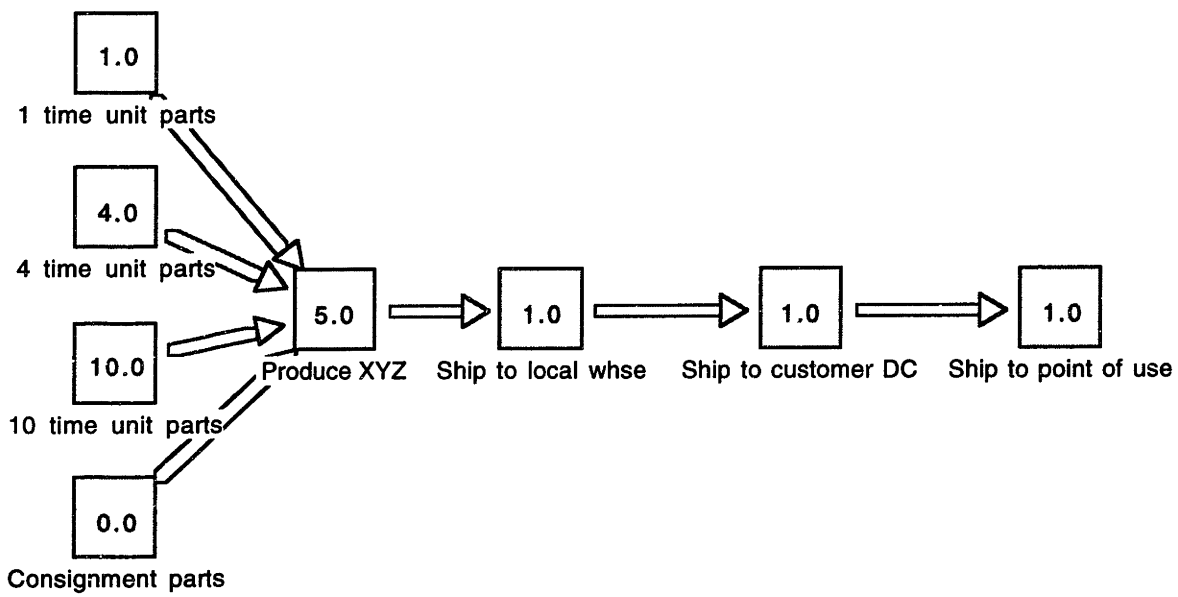


**Figure 6-11: Production lead-times for product XYZ**

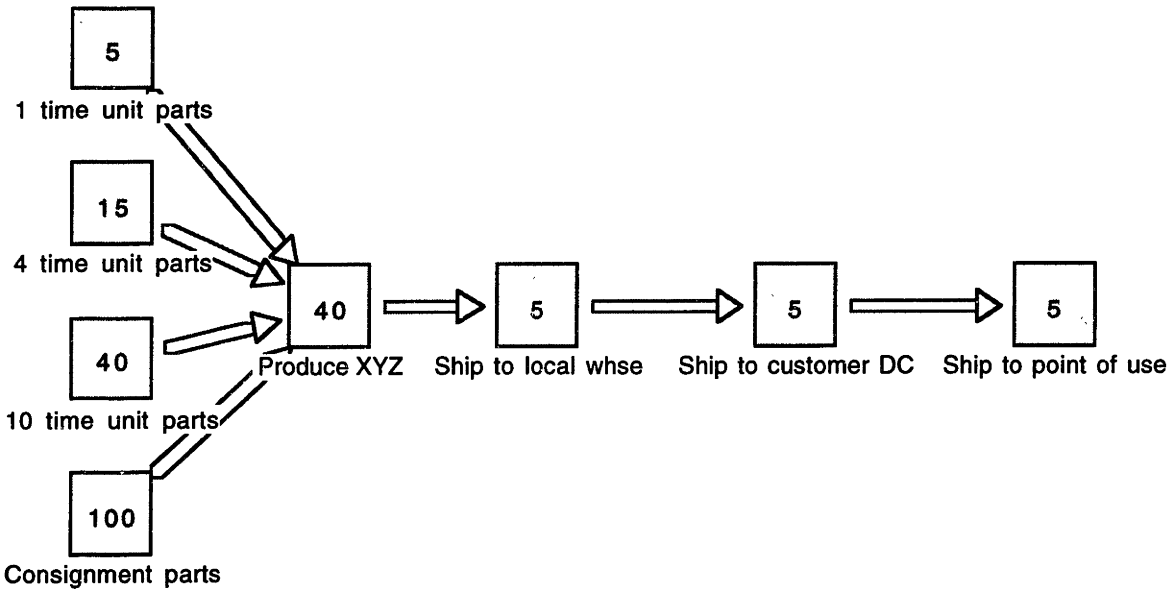


**Figure 6-12: Stage costs for part XYZ**

### 6.3.4 Model analysis

#### 6.3.4.1 Current inventory practice

Because failures in the field are very costly, the typical supply chain dynamic was for each stage in the supply chain to carry a large inventory of parts. The current inventory practice is shown in Figure 6-13:

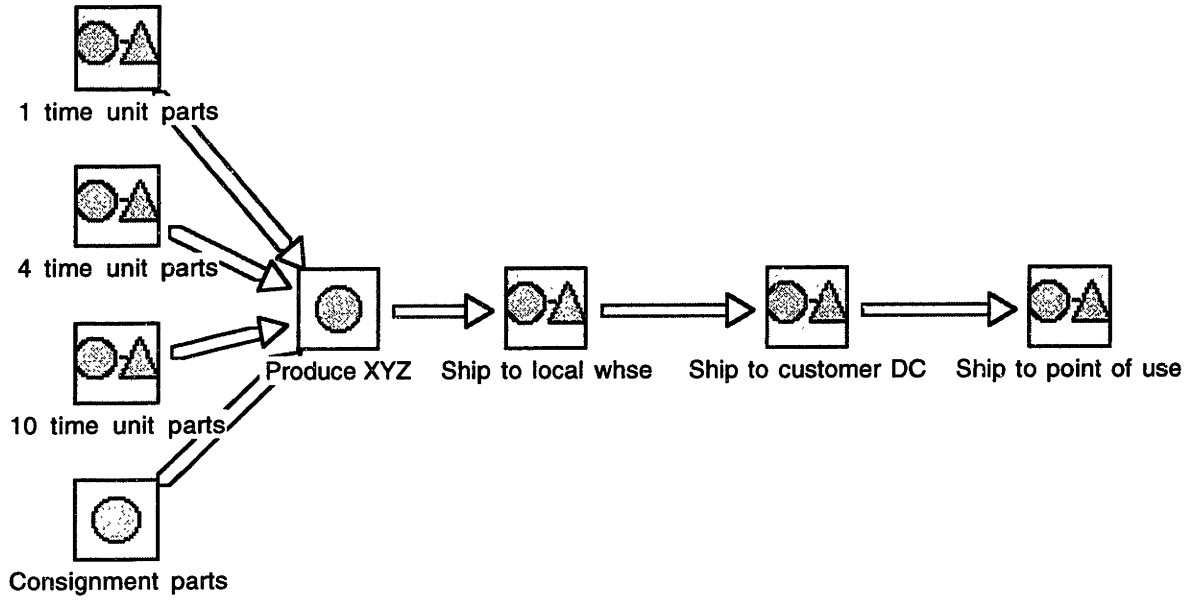


**Figure 6-13: Current inventory practice for part XYZ**

Thus, the current safety stock cost is almost $4,400 where completed XYZs are being held by Company A and at two locations by Company B.

Note that there is no inventory for parts held on consignment since these parts are not owned by Company A until they are pulled from the inventory location and consumed by the production stage.

6.3.4.2 Optimal inventory policy

The following figures show the optimal service times and safety stock inventories for the part XYZ supply chain:

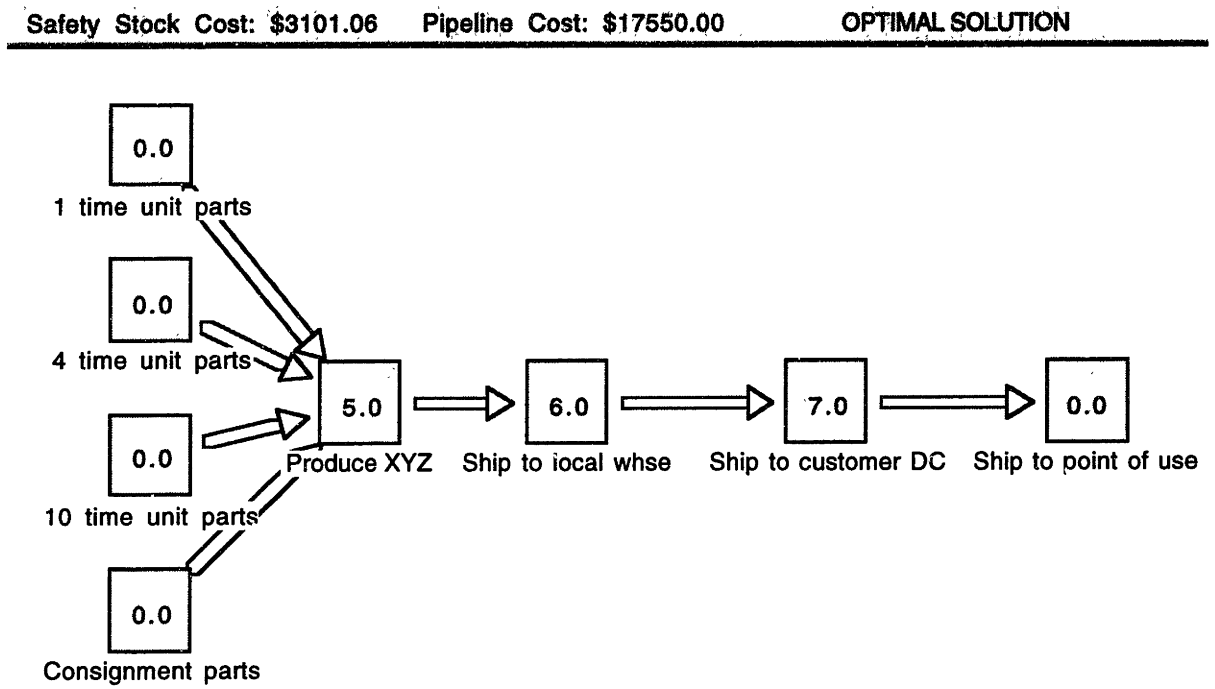


**Figure 6-14: Optimal service times for part XYZ**

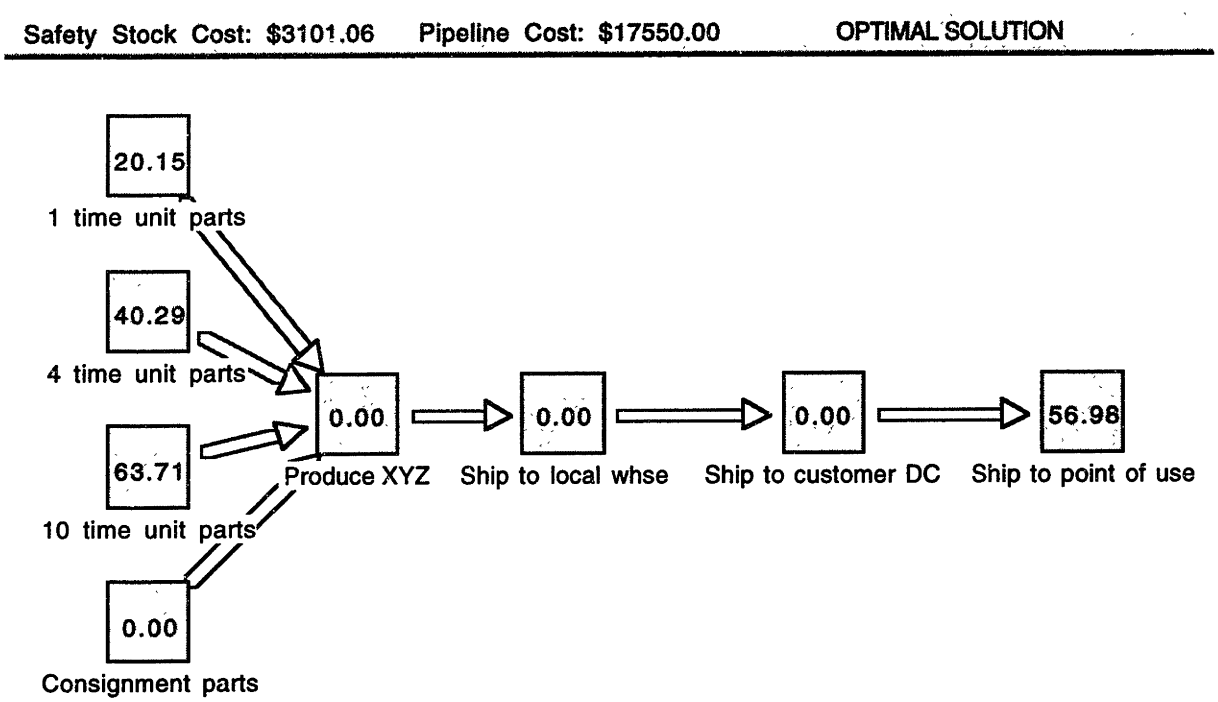


**Figure 6-15: Optimal safety stock inventories for part XYZ**

The optimal solution is to hold raw materials in inventory and finished part XYZs at the point of use. Consumption of the parts at the point of use then initiates production at Company A.

#### 6.3.4.3 How Company A is using the model

Company A is using the model to reduce inventories at both Company A and Company B. Company A and B are trying to determine compensation schemes that will make this cooperation mutually beneficial to both companies.

## 6.4 Lessons learned

The lessons learned in this project were that the software is a useful tool in determining safety stock requirements in intercompany and intracompany supply chains. This model is helping quantify management's intuition that a global view of the supply chain has benefits that can be measured in terms of cost reductions.

Finally, the process of gathering the data and understanding the inputs to the model has helped each company gain a better understanding of the key drivers of their supply chain.

# 7 Future Extensions

In this section, we briefly comment on our ongoing work to improve the solution algorithms and to extend the model to incorporate more realistic assumptions.

The first extension involves solving networks that have component commonality between adjacent echelons. The solution procedure will involve breaking the network into multiple assembly and distribution subgraphs. After these subgraphs are solved, we are working on intelligent ways to recombine these subgraphs with the remaining nodes that do not belong in either an assembly or distribution subgraph. When solving these more complicated networks, a key difficulty in the construction of an algorithm lies in how to traverse the network. In particular we need to assure that once a stage is reached in the algorithm, that all of the relevant associated stages, either upstream or downstream, have been evaluated.

Another area of research is to develop computationally efficient ways to allow a stage to quote different service times to its immediate successor nodes.

We are also looking at how to incorporate lot sizing effects in the model.

We will also be examining how to extend the model to incorporate stochastic production lead-times and capacity constraints.

# 8 References

Ahuja, R.K., T.L. Magnanti and J.B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, NJ.

Graves, S. C., "Safety Stocks in Manufacturing Systems," *Journal of Manufacturing and Operations Management*, 1 (1988), 67-101.

Graves, S. C., D. B. Kletter and W. B. Hetzel, "A Dynamic Model for Requirements Planning with Application to Supply Chain Optimization," working paper, March 1994, revised March 1995, January 1996.

Lee, H. L. and C. Billington, "Material Management in Decentralized Supply Chains," *Operations Res.*, 41 (1993), 835-847.

Simpson, K. F., "In-process Inventories," *Operations Res.*, 6 (1958), 863-873.

# THESIS PROCESSING SLIP

FIXED FIELD: ill _____ name _____

index _____ biblio _____

▶ COPIES (Archives) Aero (Dewey) Eng Hum

Lindgren Music Rotch Science

TITLE VARIES ▶☑ see degree book

NAME VARIES: ▶☐ _____

IMPRINT: (COPYRIGHT) _____

▶ COLLATION: 56 l

▶ ADD. DEGREE: _____ ▶ DEPT.: _____

SUPERVISORS: _____

NOTES:

cat'r: date:

page: J180

▶ DEPT: O.R.

▶ YEAR: 1996 ▶ DEGREE: M.S.

▶ NAME: WILLEMS, Sean Peter