SCHOOL OF OPERATIONS RESEARCH
AND INDUSTRIAL ENGINEERING
COLLEGE OF ENGINEERING
CORNELL UNIVERSITY
ITHACA, NEW YORK 14853-7501

TECHNICAL REPORT NO. 751

August 1987
Revised March 1988

**COMPUTING NESTED REORDER INTERVALS
FOR MULTI-ITEM DISTRIBUTION SYSTEMS**

by

Robin O. Roundy

This research was supported by NSF Grant DMC-8451984, by AT&T Information Systems, and DuPont Corporation.

1. INTRODUCTION.

We consider the problem of determining the approximate timing and quantities of shipments in a multi-item, multi-stage inventory distribution system. A number of different items are stocked at each of a number of different locations. External demand for each of the items occurs at a constant, location-dependent rate. All demand must be met without stockouts or backorders.

The costs to be minimized are linear holding costs and order costs. The total order cost incurred at any point in time is a function of the set of items being ordered at that point in time, but not of the quantities of the items being ordered.

The model we consider combines features of models studied earlier [Maxwell and Muckstadt 1985, Roundy 1986]. These authors have shown that a policy whose cost is guaranteed to be within 2% of the cost of an optimal nested policy can be computed in $O(N^4)$ time where N is the number of item-location pairs. The contribution of this paper is to provide an algorithm that computes the same policy in $O(ND \log N)$ time. The number D cannot exceed the number of items in the system, and for real-world problems D will seldom exceed six.

The rest of this section is devoted to describing and modeling the system and to a discussion of the relevant literature.

Problem Description.

A number of items are distributed through a distribution network. The locations of the distribution network are indexed by ℓ , $0 \leq \ell < L$. All goods enter the system at location $L - 1$. Each location $\ell < L - 1$ receives its supply of all items it stocks from a unique *predecessor* location $\varphi(\ell) > \ell$. The locations that are supplied by location ℓ are $\rho(\ell) \equiv \{k: \ell = \varphi(k)\}$, called the *successors* of ℓ . Identifying locations with nodes and predecessor-successor relationships with arcs yields the location arborescence illustrated in Figure 1.

The set of items stocked in the system is $\mathcal{J}$. At a location ℓ , each item $J \in \mathcal{J}$ has a constant external demand rate (possibly zero) and a linear holding cost rate. The holding cost of item J at location ℓ is assumed to be greater than or equal to that of item J at location $\varphi(\ell)$, i.e., the echelon holding cost rates are non-negative.

Demands occur and orders are placed in continuous time. When an order for a set of items is placed at a location ℓ , the required amounts of inventory are instantly transferred from location $\varphi(\ell)$ to location ℓ . The assumption of instant deliveries can be relaxed, and will be discussed later on.

Figure 1: The Location Arborescence.

The cost of placing an order at a location is modeled as follows. Order costs are associated with subsets $F(i)$, $1 \leq i \leq I$ of $\mathcal{J}$ called *families* [Roundy 1986]. We assume that

$$\text{if } F(i) \cap F(i') \neq \emptyset \text{ then either } F(i) \subset F(i') \text{ or } F(i') \subset F(i). \quad (1)$$

Since items interact only through families, we assume without loss of generality that family $F(I)$ contains all of the items. For convenience we also assume that each individual item constitutes a separate family.

We index the location–family pair $(F(i), \ell)$ by $n \equiv i + \ell I$. Associated with each such pair is an order cost $K'_n = K'_{i+\ell I}$. The cost $K'_{i+\ell I}$ is incurred at every point in time at which an order is placed for any item in $F(i)$ at location ℓ . Thus, if $\mathcal{J} \subset \mathcal{I}$ is the set of items ordered at location ℓ at time t , the total order cost incurred at location ℓ is $\sum_{F(i) \cap \mathcal{J} \neq \emptyset} K'_{i+\ell I}$.

To illustrate, suppose that the items stocked are 1 (ice cream), 2 (frozen meat), and 4 (canned peas). Items 1 and 2 require a refrigerated truck. Item 4 can be carried in any type of truck. The cost of providing a truck is incurred every time an order is placed. This cost is associated with the family $\{1,2,4\}$. If either or both of items 1 and 2 are ordered, the truck will have to be refrigerated. The incremental cost of providing a refrigerated truck is associated with the family $\{1,2\}$. In addition, there is an administrative cost for each distinct item included in the shipment. This cost is associated with each of the families $\{1\}$, $\{2\}$, and $\{4\}$.

An infinite–horizon, continuous–time model is assumed. At time zero, the inventory levels of all items at all locations are assumed to be zero. The goal is to determine the timing and quantities of all orders placed over the infinite horizon. Backorders and shortages are not allowed. The quality of a schedule is measured by its long–run average cost, consisting of order costs and holding costs.

A *nested policy* is one in which an order is placed for item J at location ℓ simultaneously with each order for item J at location $\varphi(\ell)$, for all J and all ℓ . Nested policies can be far from optimal, but they tend to be near–optimal when few item–location pairs have low, positive demand rates and high setup costs [Maxwell and Muckstadt 1985, Roundy 1985, Muckstadt and Roundy 1985]. Our goal is to compute a nested policy whose costs are within 2% of the cost of an optimal nested policy.

Extensions of the Model.

The assumption of zero lead times can be relaxed. If the lead time for a shipment from location ℓ to location ℓ' is independent of the composition and size of the shipment, lead times can be incorporated without causing an increase in the relative cost of the policy. The timing of the orders and shipments of the policy computed under the assumption of zero lead times are simply offset to reflect the actual lead times [Roundy 1986].

It is possible to compute a non-nested policy whose long-run average cost is within 2% of the cost of an optimal policy, rather than within 2% of the cost of an optimal nested policy [Roundy 1986]. However the best running time that is currently known for this problem is between $O(N^4)$ and $O(N^4 L^4)$, depending on the topology of the location arborescence.

Review of the Literature.

There is a sizeable literature on lot-sizing models for multi-stage production/inventory systems with constant demand rates (see [Roundy 1986] for additional references). Recently a number of papers have appeared that propose heuristics whose cost can be shown to be within a small percentage of either the cost of an optimal policy, or the cost of an optimal nested policy. These papers all approach the problem in a similar way and prove similar results. The main differences between the papers are in the topology of the bill-of-material network, the way setup costs are modeled, and in the types of policies they compute (nested or non-nested). The papers that compute non-nested policies find a policy whose cost is within either 6% or 2% of the cost of an optimal policy, whereas the papers that compute nested policies find a policy whose cost is within either 6% or 2% of the cost of an optimal nested policy.

The first result of this type dealt with non-nested policies for a single-item, one-warehouse, multi-retailer system [Roundy 1985]. Another special case of the system we study is the multi-item, one-warehouse, multi-retailer system with one multi-item family, for

which Muckstadt and Roundy [1987] computed nested policies. Also, Iyogun [1987] computes non-nested policies for a multi-item distribution system in which each location with external demand stocks only one item, and at each location only one family has a positive order cost.

Several systems that generalize the system we study have been considered in the literature. Maxwell and Muckstadt [1985] study nested policies for a system that allows general bill-of-material networks. In their model order costs are associated with individual items only, not with families. Roundy [1986] studied non-nested policies for a system that generalizes the system considered by Maxwell and Muckstadt by introducing families. Queyranne [1986] studied a system that differs from Roundy's system in that submodular order cost functions are used in place of families. Other papers that consider systems with different types of bill-of-material structures include Jackson, Maxwell, and Muckstadt [1985], Roundy [1983], and Jackson and Roundy [1985].

Anily [1986] and Anily and Federgruen [1986a,b] have used ideas from these papers in their interesting research on one-warehouse, multi-retailer systems with capacity constraints and vehicle routing, and on a capacitated, two-stage multi-item production/inventory problem.

Modelling the Problem.

For each family $F(j)$, we define the *successor* $\beta(j)$ of j by letting $F(\beta(j))$ be the minimal family that properly contains the family $F(j)$. The uniqueness of $\beta(j)$ follows from (1). The set of *predecessors* $\alpha(j)$ of j is defined by $\alpha(j) \equiv \{j' : j = \beta(j')\}$. The family arborescence is defined by associating nodes with families and arcs with predecessor-successor relationships. Figure 2 shows the family arborescence for the simple, three-item example described above. We assume that the families are indexed in such a way that $\beta(i) > i$, and that the locations are indexed in such a way that $\varphi(\ell) > \ell$.

Much of the subsequent analysis depends on the topology of the network $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, defined as follows. The node set of $\mathcal{G}$ is $\mathcal{N} \equiv \{i + \ell I : 1 \leq i \leq I, 0 \leq \ell < L\}$, the set of all location–family pairs. The arc set $\mathcal{A}$ of

Figure 2: The Family Arborescence.

Figure 3: The Network $\mathcal{G}$.

$\mathcal{G}$ consists of all *inventory arcs* of the form $(i + \varphi(\ell)I, i + \ell I)$, and of all *family arcs* of the form $(i + \ell I, \beta(i) + \ell I)$. $\mathcal{G}$ is illustrated in Figure 3. Note that the nodes in each column of Figure 3 correspond to a location and that the nodes in each row correspond to a family.

Two cost parameters are associated with each node $n = i + \ell I$ of $\mathcal{G}$. The order cost $K'_n = K'_{i+\ell I}$ is associated with node n . In addition there is a holding cost coefficient associated with node n . If $F(i)$ is a multi-product family then for all ℓ , the holding cost coefficient associated with node $n = i + \ell I$ is $g'_n = 0$. Let $F(i) = \{J\}$. If item J has echelon demand rate $d_{J\ell}$ and echelon holding cost rate $h_{J\ell}$ at location ℓ then the holding-cost coefficient associated with node $n = i + \ell I$ is $g'_n \equiv \frac{1}{2}h_{J\ell}d_{J\ell}$. We assume that the total system demand for each item is positive and that the holding cost for each item at location $L - 1$ is positive, i.e., that $g'_{i+(L-1)I} > 0$ for all $1 \leq i \leq I$.

We say that an order is placed at a node $n = i + \ell I$, $F(i) = \{J\}$, when an order is placed for item J at location ℓ . We say that an order is placed at a node $n = i + \ell I$, $|F(i)| > 1$, whenever the joint setup cost K_n associated with family $F(i)$ at location ℓ is incurred.

Nested Power-of-Two Policies.

Let $B > 0$ be given. A *power-of-two policy* is one with the following property. For all $n \in \mathcal{N}$, an order is placed at node n at all times $m \times T_n$, $m \geq 0$ and integer, where $T_n = 2^{k(n)} \times B$ for some (possibly negative) integer $k(n)$. B is called the *base planning period*.

Let $\mathcal{T} = (T_1, T_2, \dots, T_N)$ be the reorder intervals of a nested power-of-two policy. Then

$$T_m \geq T_n \text{ for all } (m, n) \in \mathcal{N}. \quad (2)$$

For inventory arcs (m,n) , (2) corresponds to the assumption of nestedness. For family arcs (m,n) , (2) enforces the condition that if $J \in F(i)$ then the order cost $K_{i+\ell}^J$ is incurred every time an order is placed for item J at location ℓ .

The cost of a nested power-of-two policy is

$$c(\mathcal{J}) \equiv \sum_{n \in \mathcal{N}} \left[\frac{K_n'}{T_n} + g_n' T_n \right].$$

The holding cost computations that lead to $c(\mathcal{J})$ are based on echelon inventories and holding costs. They are identical to those in [Maxwell and Muckstadt 1985]. The computation of order costs follows [Roundy 1986]. The problem of finding an optimal nested power-of-two policy can be formulated as follows.

$$(P^*) \quad \min: \quad c(\mathcal{J}) \equiv \sum_{n \in \mathcal{N}} \left[\frac{K_n'}{T_n} + g_n' T_n \right]$$

$$\text{such that:} \quad T_m \geq T_n \text{ for all } (m,n) \in \mathcal{A},$$

$$T_n = 2^{k(n)} \times B, \quad k(n) \text{ integer, for all } n \in \mathcal{N}.$$

The approach usually taken to solving (P^*) is to relax the integrality constraint, resulting in the continuous optimization problem

$$\begin{array}{cc} (P) & \min n: \\ \text{☺☺☺☺} & \text{☺☺☺☺} \end{array} \quad c(\mathcal{T}) \equiv \sum_{n \in \mathcal{N}} \left[\frac{K'_n}{T_n} + g'_n T_n \right]$$

such that: $T_m \geq T_n$ for all $(m,n) \in \mathcal{A}$,

$$T_n \geq 0 \text{ for all } n \in \mathcal{N}.$$

A key result is that the solution to (P) is a lower bound on the long-run average cost of any feasible, nested schedule of orders for the multi-item distribution system, over any finite or infinite time horizon. The proof of this theorem is omitted. It combines ideas from similar proofs in [Maxwell and Muckstadt 1985] and [Roundy 1986] in a straightforward manner.

From an optimal solution to (P), a power-of-two policy whose cost is guaranteed to be within 2% of the solution to (P) can be found if B is a variable, and a policy whose costs are within 6% of the solution to (P) can be found if B is a user-specified constant. Since the solution to (P) is a lower bound on the cost of any feasible, nested schedule, the cost of the policy computed is within either 6% or 2% of the cost of an optimal nested schedule. In either case, the running time of the procedure, given an optimal solution to (P), is $O(N \log N)$ [Roundy 1986].

We will solve (P) in $O(ND \log N)$ time where D is the depth of the family arborescence. This results in an overall running time of $O(ND \log N)$ for the entire procedure. In practice D will usually be very small compared to I . Systems in which the setup costs have the widely studied joint-replenishment structure have $D = 2$.

Note that by inverting the roles of K'_n and g'_n , reversing the orientation of the arcs in $\mathcal{G}$, and using the variable $S_n \equiv 1/T_n$ in place of T_n , we obtain a problem equivalent to (P) and of the same form as (P), except that the location arborescence and the family arborescence have been interchanged. This transformation can be used to implement the

algorithm in such a way that D is bounded by the depth of the location arborescence rather than the depth of the family arborescence. However in practice one would usually expect the family arborescence to have lower depth than the location arborescence.

Section 2 presents important properties of (P), on which our algorithm is based. The algorithm used to solve (P) is presented and its running time is analyzed in Section 3. Possible directions for future research are discussed in Section 4.

2. STRUCTURAL RESULTS.

Perturbation of the Data.

It is common to have $K'_n = g'_n = 0$. For example, if none of the items in family $F(i)$ is stocked at location ℓ then $K'_{i+\ell I} = g'_{i+\ell I} = 0$. The same is true if $F(i)$ is a multi-product family that has a positive order cost at one or more locations, but not at location ℓ . This causes (P) to be highly degenerate.

The degeneracies must be handled with care. If ties are broken arbitrarily, key structural and algorithmic properties will fail to hold. The degeneracies could be handled by explicitly building tie-breaking rules into all of the definitions, results, and algorithms that follow. However doing so would unnecessarily obscure the exposition. Therefore we have chosen to perturb the data of the problem in such a way that the degeneracies disappear, and to give a simple, numerically stable lexicographic scheme that mimics the perturbation and can be used to implement the algorithm.

The data is perturbed as follows. Let

$$K_n \equiv K'_n + \epsilon^{2(N+1-n)} \text{ and } g_n \equiv g'_n + \epsilon \quad \forall \quad 1 \leq n \leq N$$

where ϵ is a small positive number. Note that after the perturbation, all setup and holding costs are strictly positive. For any $C \in \mathcal{N}$ let

$$K(C) \equiv \sum_{n \in C} K_n, \quad g(C) \equiv \sum_{n \in C} g_n, \quad \text{and} \quad T(C) \equiv K(C)/g(C).$$

LEMMA 1. If $B \subset \mathcal{N}$ and $B \neq \emptyset$ then $0 < T(B) < \infty$. If $B, C \subset \mathcal{N}$ and $B \neq C$ then $T(B) \neq T(C)$.

The proof of Lemma 1 is found in Appendix A. Appendix A also contains the lexicographic scheme for implementing the perturbation and a proof that as ϵ tends to zero, the optimal solution to (P) for the perturbed data approaches an optimal solution to (P) for the original data.

Throughout the remainder of this paper we use the perturbed data. Thus K_n and g_n are used in place of K'_n and g'_n in (P), and Lemma 1 holds.

Notation.

The notation used most commonly in the sequel is listed below. Some of these terms have already been defined; some are new.

- L : The number of locations.
- ℓ : A location. $0 \leq \ell < L$.
- $\varphi(\ell)$: The unique location that supplies ℓ . $\varphi(\ell) > \ell$ and $\varphi(L-1) = \emptyset$.
- $\rho(\ell)$: $\{\ell^* : \ell = \varphi(\ell^*)\}$, the set of locations supplied by ℓ .
- $\varphi^*(\ell)$: $[\{\ell\} \cup \varphi(\ell) \cup \varphi(\varphi(\ell)) \cup \dots] = \{\ell\} \cup \{\text{all locations that supply location } \ell, \text{ directly or indirectly}\}$. In Figure 1, $\varphi^*(1) = \{1, 2, 4\}$.
- $\rho^*(\ell)$: $[\{\ell\} \cup \rho(\ell) \cup \rho(\rho(\ell)) \cup \dots] = \{\ell\} \cup \{\text{all locations supplied through location } \ell, \text{ directly or indirectly}\}$. In Figure 1, $\rho^*(2) = \{0, 1, 2\}$.
- I : The number of families.
- i : A family index. $1 \leq i \leq I$.

- $\beta(i)$: The index of the unique minimal family that contains family $F(i)$. $\beta(i) > i$.
- $\alpha(i)$: $\{i^*: i = \beta(i^*)\}$.
- $\beta^*(i)$: $\{i\} \cup \beta(i) \cup \beta(\beta(i)) \cup \dots = \{i\} \cup \{\text{all indices of families that contain family } F(i)\}$. In Figure 2, $\beta^*(2) = \{2, 3, 5\}$.
- $\alpha^*(i)$: $\{i\} \cup \alpha(i) \cup \alpha(\alpha(i)) \cup \dots = \{i\} \cup \{\text{all indices of families that are contained in family } F(i)\}$. In Figure 2, $\alpha^*(3) = \{1, 2, 3\}$.
- D : The depth of the item tree.
- N : $I \times L$ = the number of family–location pairs = the number of nodes in $\mathcal{G}$.
- n : A family–location pair or a node in $\mathcal{G}$. $1 \leq n \leq N$. If $n = i + \ell I$ then n corresponds to family $F(i)$ and location ℓ .
- $\mathcal{N}$: $\{n: 1 \leq n \leq N\}$; the node set of $\mathcal{G}$.
- $\mathcal{A}$: $\{(i + \varphi(\ell)I, i + \ell I)\} \cup \{(i + \ell I, \beta(i) + \ell I)\}$; the arc set of $\mathcal{G}$.
- $\mathcal{G}$: $(\mathcal{N}, \mathcal{A})$, the directed graph with node set $\mathcal{N}$ and arc set $\mathcal{A}$.
- $a(n)$: If $n = i + \ell I$ then $a(n) = \{j + \ell I : j \in \alpha(i)\}$, the predecessors of n in $\mathcal{G}$ that correspond to location ℓ . In Figure 3, $a(13) = \{11, 12\}$ and $a(12) = \emptyset$.
- $b(n)$: If $n = i + \ell I$ then $b(n) = \beta(i) + \ell I$, the successor of n in $\mathcal{G}$ that corresponds to location ℓ . In Figure 3, $b(13) = 15$ and $b(15) = \emptyset$.
- $r(n)$: If $n = i + \ell I$ then $r(n) = \{i + kI : k \in \rho(\ell)\}$, the successors of n in $\mathcal{G}$ that correspond to family $F(i)$. In Figure 3, $r(13) = \{3, 8\}$ and $r(8) = \emptyset$.
- $f(n)$: If $n = i + \ell I$ then $f(n) = i + \varphi(\ell)I$, the predecessor of n in $\mathcal{G}$ that corresponds to family $F(i)$. In Figure 3, $f(13) = 23$ and $f(23) = \emptyset$.
- K_n : The perturbed setup cost associated with node n .
- g_n : The perturbed holding cost coefficient associated with node n .
- $T(C)$: $\left[\sum_{n \in C} K_n \right] / \left[\sum_{n \in C} g_n \right]$ where $C \subset \mathcal{N}$.
- $\mathcal{N}^n$: If $n = i + \ell I$ then $\mathcal{N}^n \equiv \{j + kI : j \in \alpha^*(i), k \in \rho^*(\ell)\}$. Figure 4 illustrates. Note that the subnetwork of $\mathcal{G}$ induced by $\mathcal{N}^n$ is of the same form as $\mathcal{G}$.

- (P^n) : The version of (P) that is restricted to the subnetwork of $\mathcal{G}$ induced by $\mathcal{N}^n$.
- $T(m,n)$: The optimal value of T_m in the solution to (P^n) .
- $\mathcal{C}$: The indices of the clusters that solve (P) ; see Theorem 3 below.
- $\mathcal{C}^n$: The indices of the clusters that solve (P^n) ; see Theorem 3 below.
- C_n : A cluster; see Theorem 3 below. If $n \leq N$ then C_n is the cluster that contains n in the solution to (P^n) .
- $C_n(t)$: Defined in (3) below.

General Structural Results.

We now examine some key, general properties of (P) . The following result appears as Corollary 2.5 in [Jackson and Roundy 1985].

LEMMA 2. (P) has a unique optimal solution.

We say that $B \subset C \subset \mathcal{N}$ is *initial in C* (resp., *final in C*) if there are no arcs $m \rightarrow n \in \mathcal{A}$ (resp., $n \rightarrow m \in \mathcal{A}$), $m \in C \setminus B$, $n \in B$. The following appears as Proposition 1 in [Jackson and Roundy 1985].

Figure 4: $\mathcal{N}^{13}$ and the Network it Induces.

THEOREM 3. T solves (P) if and only if there exist subsets C_j , $j \in \mathcal{C}$ of $\mathcal{N}$ that satisfy the following conditions.

- Partition:* $\{C_j; j \in \mathcal{C}\}$ is a partition of $\mathcal{N}$.
Optimality: $T_n = T(C_j)$ for all $n \in C_j$ and all $j \in \mathcal{C}$.
Convexity: If $(m, n) \in \mathcal{A}$, $m \in C_i$, $n \in C_j$, and $i \neq j$ then $T(C_i) > T(C_j)$.
Concavity: If B is initial in C_j then $T(B) < T(C_j) < T(C_j \setminus B)$.

We refer to the sets C_j as *clusters*. The importance of Theorem 3 is that the clusters entirely determine the solution to (P). Thus Theorem 3 reduces (P) to a purely combinatorial problem, that of finding clusters C_j , $j \in \mathcal{C}$ that satisfy the conditions of Theorem 3. Note that Lemma 1 and the *Optimality* property of Theorem 3 imply that $0 < T_n < \infty$ for all n .

Subproblems (P^n) of (P).

Our approach to solving (P) can be summarized as follows. We define a series of subproblems of (P) of varying sizes, the smallest of which are easily solvable. The solutions to smaller subproblems are used to generate the solutions to successively larger subproblems, until (P) itself has been solved. We begin by defining the subproblems.

The following lemma is illustrated in Figure 5.

LEMMA 4. Let $m \in \mathcal{N}^n \cap \mathcal{N}^{n*}$, $n = i + \ell I$, $n^* = i^* + \ell^* I$, $i \leq i^*$, and $\ell \geq \ell^*$. Then $T(m, n) \leq T(m, n^*)$.

The proof of Lemma 4 is found in Appendix B. Intuitively, the lemma holds because $\mathcal{N}^n \cap \mathcal{N}^{n*}$ is final in $\mathcal{N}^n$ and initial in $\mathcal{N}^{n*}$. This means that the arcs between $\mathcal{N}^n \cap \mathcal{N}^{n*}$, $\mathcal{N}^n \setminus \mathcal{N}^{n*}$, and $\mathcal{N}^{n*} \setminus \mathcal{N}^n$ correspond to constraints in (2) that force $\{T_m : m \in \mathcal{N}^n \cap \mathcal{N}^{n*}\}$ to be small in (P^n) and to be large in (P^{n*}) .

Figure 5: Illustration of Lemma 4.

Solution to (P^n) via Subproblems, given $T(n,n)$.

Having defined subproblems (P^n) of (P) , we want to use the solutions to smaller subproblems to find the solutions to larger ones. It turns out that we can write an explicit expression for the clusters that solve (P^n) as a function of the solutions to the subproblems (P^m) , $m \in a(n) \cup r(n)$, assuming that the value $T(n,n)$ of T_n in the solution to (P^n) is known and is equal to t .

Let C_n be the cluster containing n in an optimal solution to (P^n) . (Later we will see that all clusters of interest are of this type. In the meanwhile we assume that clusters C_{n^*} not of this type are given indices $n^* > N$.) Let $\mathcal{C}^n = \{n^* : C_{n^*} \text{ is one of the clusters that solve } (P^n)\}$. Let $\mathcal{C}^{a(n)} \equiv \bigcup_{m \in a(n)} \mathcal{C}^m$ and $\mathcal{N}^{a(n)} \equiv \bigcup_{m \in a(n)} \mathcal{N}^m$, and let $\mathcal{C}^{r(n)}$ and $\mathcal{N}^{r(n)}$ be similarly defined. Let

$$A(n,t) \equiv \{n^* \in \mathcal{C}^{a(n)} : T(C_{n^*}) \geq t\} \text{ and } R(n,t) \equiv \{n^* \in \mathcal{C}^{r(n)} : T(C_{n^*}) < t\}.$$

Let $C_S \equiv \bigcup_{n^* \in S} C_{n^*}$ for all sets S of cluster indices, and let

$$C_n(t) \equiv \mathcal{N}^n \setminus [C_{A(n,t)} \cup C_{R(n,t)}]. \quad (3)$$

Figure 6 illustrates these definitions for $n = 13$ (also see Figure 4). The main result of this sub-section is that if $t = T(C_n(t))$ then $C_n = C_n(t)$ and $\mathcal{C}^n = \{n\} \cup A(n,t) \cup R(n,t)$. Thus (P^n) is solved, and C_n is the only cluster in the solution to (P^n) that does not appear in the solutions to (P^m) , $m \in a(n) \cup r(n)$. Figure 7 gives the solution to (P^{13}) for the data of Figure 6, assuming that $t = 1 = T(C_{13}(1))$. The next two lemmas lead up to the proof of this result.

Figure 6.

Solution to (P^3) : $\mathcal{C}^3 = \{3, 2, 1\}$.

$$C_1 = \{1\} ; T(C_1) = 3.$$

$$C_2 = \{2\} ; T(C_2) = 1/2.$$

$$C_3 = \{3\} ; T(C_3) = 1/3.$$

Solution to (P^8) : $\mathcal{C}^8 = \{8\}$.

$$C_8 = \{8, 7, 6\} ; T(C_8) = 5.$$

Solution to (P^{11}) : $\mathcal{C}^{11} = \{11, 6\}$.

$$C_{11} = \{11, 1\} ; T(C_{11}) = 2.$$

$$C_6 = \{6\} ; T(C_6) = 1/4.$$

Solution to (P^{12}) : $\mathcal{C}^{12} = \{12, 2\}$.

$$C_{12} = \{12, 7\} ; T(C_{12}) = 4.$$

$$C_2 = \{2\} ; T(C_2) = 1/2.$$

$$\mathcal{C}^{a(13)} = \{12, 11, 6, 2\}$$

$$\mathcal{N}^{a(13)} = \{12, 11, 7, 6, 2, 1\}$$

$$A(13,1) = \{12, 11\}$$

$$C_{A(13,1)} = \{12, 11, 7, 1\}$$

$$\mathcal{C}^{r(13)} = \{8, 3, 2, 1\}.$$

$$\mathcal{N}^{r(13)} = \{8, 7, 6, 3, 2, 1\}.$$

$$R(13,1) = \{3, 2\}.$$

$$C_{R(13,1)} = \{3, 2\}.$$

$$C_{13}(1) = \{13, 8, 6\}.$$

Figure 7: Solution to (P^{13}) if $1 = T(C_{13}(1))$.

$$\mathcal{C}^{13} = \{13, 12, 11, 3, 2\}.$$

$$C_{12} = \{12, 7\} ; T(C_{12}) = 4.$$

$$C_{11} = \{11, 1\} ; T(C_{11}) = 2.$$

$$C_{13} = \{13, 8, 6\} ; T(C_{13}) = 1.$$

$$C_2 = \{2\} ; T(C_2) = 1/2.$$

$$C_3 = \{3\} ; T(C_3) = 1/3.$$

LEMMA 5. $C_{A(n,t)} \cap C_{R(n,t)} = \emptyset$. $C_n(t)$ is final in $C_n(t) \cup C_{A(n,t)}$ and initial in $C_n(t) \cup C_{R(n,t)}$. $C_{A(n,t)}$ is initial in $\mathcal{N}^n$ and $C_{R(n,t)}$ is final in $\mathcal{N}^n$.

PROOF. Let $m \in C_{n^*}$, $n^* \in \mathcal{C}^{a(n)}$ and $m \in C_{n^{**}}$, $n^{**} \in R(n,t)$. The definition of $R(n,t)$ and Lemma 4 imply that $t > T(m, n^{**}) \geq T(m, n^*)$, so $n^* \notin A(n,t)$. Thus

$C_{A(n,t)} \cap C_{R(n,t)} = \emptyset$. By the *Convexity* property of Theorem 3, and the definition of $R(n,t)$, $C_{R(n,t)}$ is final in $\mathcal{N}^{r(n)}$. $\mathcal{N}^{r(n)}$ is final in $\mathcal{N}^n$, so $C_{R(n,t)}$ is final in $\mathcal{N}^n$.

Therefore $C_{R(n,t)}$ is final in $C_n(t) \cup C_{R(n,t)} \subset \mathcal{N}^n$. Similarly, $C_{A(n,t)}$ is initial in $\mathcal{N}^n$ and in $C_n(t) \cup C_{A(n,t)}$. $\square$

The following algebraic fact is easily verified.

$$\text{If } B \cap C = \emptyset \text{ then } \min\{T(B), T(C)\} \leq T(B \cup C) \leq \max\{T(B), T(C)\}. \quad (4)$$

LEMMA 6. If $B \subset C_n(t)$ is final in $C_n(t)$ and $n \notin B$ then $T(B) > t$. Similarly, if $B \subset C_n(t)$ is initial in $C_n(t)$ and $n \notin B$ then $T(B) < t$.

PROOF. Let $n \notin B$ and let $B \subset C_n(t)$ be final in $C_n(t)$. Then $B \subset \mathcal{N}^{r(n)}$ and $B \cap C_{R(n,t)} = \emptyset$, so $B \subset \mathcal{N}^{r(n)} \setminus C_{R(n,t)}$. Since B is final in $C_n(t)$ and $C_n(t)$ is final in $C_n(t) \cup C_{A(n,t)}$, B is final in $C_n(t) \cup C_{A(n,t)}$. But $C_n(t) \cup C_{A(n,t)}$ contains $\mathcal{N}^{r(n)} \setminus C_{R(n,t)}$, so B is final in $\mathcal{N}^{r(n)} \setminus C_{R(n,t)}$. Let $R \equiv \mathcal{E}^{r(n)} \setminus R(n,t)$. Then $B = \bigcup_{n^* \in R} B_{n^*}$ where $B_{n^*} \subset C_{n^*}$ is final in C_{n^*} . By the *Concavity* property of Theorem 3, $T(B_{n^*}) > T(C_{n^*}) \geq t$. Equation (4) implies $T(B) > t$. Similarly, if $n \notin B$ and B is initial in $C_n(t)$ then $T(B) < t$. $\square$

THEOREM 7. If $t = T(C_n(t))$ then $C_n = C_n(t)$ and $\mathcal{E}^n = \{n\} \cup A(n,t) \cup R(n,t)$.

PROOF. The proof is by induction on n . By Theorem 3, the result clearly holds for $n = 1$. Assume that the lemma holds for all $m < n$, and that $t = C_n(t)$. Lemma 2 implies that $T(C_m) \neq t$ for all $m \in A(n,t) \cup R(n,t)$. We will prove that the clusters with index set $\mathcal{E}^n = \{n\} \cup A(n,t) \cup R(n,t)$ satisfy the conditions of Theorem 3 for (P^n) .

The *Partition* property of Theorem 3 holds because Lemma 4 implies that $C_{A(n,t)} \cap C_{R(n,t)} = \emptyset$. The *Convexity* property holds by Lemma 5 and the induction hypothesis. The induction hypothesis implies that *Concavity* holds for all C_m , $m \in A(n,t) \cup R(n,t)$. Since $t = T(C_n(t))$, Lemma 6 implies that *Concavity* holds for $C_n(t)$. $\square$

Theorem 7 implies that

$$\begin{aligned} \mathcal{E}^n = \{n\} \cup [& \bigcup \{C_m : m \in \mathcal{E}^{a(n)}, T(C_m) > T(C_n)\} \\ & \cup \bigcup \{C_m : m \in \mathcal{E}^{r(n)}, T(C_m) < T(C_n)\}]. \end{aligned} \quad (5)$$

Parametric Search for $T(n, n)$.

The task of solving (P^n) given the solutions to (P^m) , $m \in a(n) \cup r(n)$, has been reduced to the problem of finding $T(n, n)$, the value of T_n in the solution to (P^n) . We will find $T(n, n)$ as follows. First we set $t = \infty$ and find $A(n, t)$, $R(n, t)$, and $C_n(t)$. We then decrease the value of t and keep track of the changes that occur in $A(n, t)$, $R(n, t)$, and $C_n(t)$. When we reach a value of t satisfying $t = T(C_n(t))$, we will have solved (P^n) by Theorem 7. In this subsection we describe the way in which $A(n, t)$, $R(n, t)$, and $C_n(t)$ change as t is decreased.

We use $\oplus$ to denote the union of disjoint sets, i.e., $A = B \oplus C$ if and only if $A = B \cup C$ and $B \cap C = \emptyset$.

LEMMA 8. Suppose that $m \in \mathcal{S}^{a(n)} \cup \mathcal{S}^{r(n)}$, that $t > T(C_m) > t'$, and that for all $n^* \neq m$, $n^* \in \mathcal{S}^{a(n)} \cup \mathcal{S}^{r(n)}$, either $T(C_{n^*}) \geq t$ or $t' \geq T(C_{n^*})$. Then

1. $T(C_m) > T(C_n(t))$ if and only if $T(C_m) > T(C_n(t'))$.
2. If $m \in \mathcal{S}^{a(n)} \cap \mathcal{S}^{r(n)}$ then $C_n(t) = C_n(t')$.
3. If $m \in \mathcal{S}^{a(n)} \setminus \mathcal{S}^{r(n)}$ then $C_n(t) = C_m \oplus C_n(t')$.
4. If $m \in \mathcal{S}^{r(n)} \setminus \mathcal{S}^{a(n)}$ then $C_n(t') = C_m \oplus C_n(t)$.

PROOF. Assertion 2 follows directly from (3). If $m \in \mathcal{S}^{a(n)} \setminus \mathcal{S}^{r(n)}$ then $R(n, t) = R(n, t')$ and $m \in A(n, t')$. By Lemma 4, $\emptyset = C_m \cap C_{R(n, t')} = C_m \cap C_{R(n, t)}$. Since $m \notin A(n, t)$ we have $C_m \subset C_n(t)$. This and (3) imply Assertion 3. Similarly, if $m \in \mathcal{S}^{r(n)} \setminus \mathcal{S}^{a(n)}$ then $C_m \cap C_{A(n, t')} = \emptyset$ and $m \notin R(n, t')$, so $C_m \subset C_n(t')$ and Assertion 4 holds. Assertion 1 follows from (4) and from Assertions 2 – 4. $\square$

General Description of the Algorithm.

The algorithm that solves (P) can be informally described as follows. We solve (P^n) for each n , starting with $n = 1$ and proceeding up to $n = N$. Because of the way the nodes of $\mathcal{G}$ are indexed, when we solve (P^n) we have already solved (P^m) for all $m \in a(n) \cup r(n)$.

We solve (P^n) as follows. If $a(n) \cup r(n) = \emptyset$ then we set $C_n = \{n\}$ and $\mathcal{C}^n = \{n\}$. Otherwise we solve (P^n) by searching parametrically for the value of t satisfying $t = T(C_n(t))$ and applying Theorem 7. Lemma 8 is used to update the values of $A(n,t)$, $R(n,t)$, and $C_n(t)$ as t decreases from ∞ to $T(C_n)$.

If the algorithm described above is implemented in a straightforward manner in which the basic data structures are the sets $\mathcal{C}^n$, the running time will be unacceptably high. The reason is that nodes in $\mathcal{C}^{a(n)} \cap \mathcal{C}^{r(n)}$ will need to be processed in performing the parametric search for $t = T(C_n)$, even though Assertion 2 of Lemma 8 tells us that they have no effect on $C_n(t)$. To make the algorithm efficient we need data structures that give us access to the sets $\mathcal{C}^{a(n)} \setminus \mathcal{C}^{r(n)}$ and $\mathcal{C}^{r(n)} \setminus \mathcal{C}^{a(n)}$ directly, without requiring us to process the elements of $\mathcal{C}^{a(n)} \cap \mathcal{C}^{r(n)}$.

Figure 8: The Network Induced by $\mathcal{N}^*(7)$.

The Structure of the Sets $\mathcal{E}^{a(n)} \setminus \mathcal{E}^{r(n)}$ and $\mathcal{E}^{r(n)} \setminus \mathcal{E}^{a(n)}$.

In order to devise efficient data structures for accessing $\mathcal{E}^{a(n)} \setminus \mathcal{E}^{r(n)}$ and $\mathcal{E}^{r(n)} \setminus \mathcal{E}^{a(n)}$ directly, we need to study the structure of the set $\mathcal{S}(m) = \{n : m \in \mathcal{E}^n\}$. Let $m = i^* + \ell^*I$, and let $\mathcal{N}^*(m) \equiv \{i + \ell I : i \in \beta^*(i^*), \ell \in \varphi^*(\ell^*)\}$. See Figure 8. Note that $\mathcal{N}^*(m) = \{n : m \in \mathcal{N}^n\}$, so that $\mathcal{S}(m) \subset \mathcal{N}^*(m)$. Also note that the network induced by $\mathcal{N}^*(m)$ has the topology of a planar grid with arcs oriented from left to right and from top to bottom, with m at the upper right-hand corner.

For a fixed location ℓ , the set $c(\ell, m) \equiv \{i + \ell I \in \mathcal{N}^*(m) : 1 \leq i \leq I\}$ is called a *column* of $\mathcal{N}^*(m)$. These correspond to the columns in Figure 8. Similarly, for a fixed item i the set $\{i + \ell I \in \mathcal{N}^*(m) : 0 \leq \ell < L\}$ is called a *row* of $\mathcal{N}^*(m)$. In Figure 8 the columns $c(\ell, m)$ of $\mathcal{N}^*(n)$ appear in decreasing order of ℓ , from left to right. The nodes of a column appear in increasing order of n , from top to bottom.

LEMMA 9. Let $m = i^* + \ell^*I$. Then

1. The columns $c(\ell, m)$ that have a non-empty intersection with $\mathcal{S}(m)$ are consecutive, i.e., there is an $\ell(m) \geq \ell^*$ such that $\{\ell \in \varphi^*(\ell^*) : \mathcal{S}(m) \cap c(\ell, m) \neq \emptyset\} = \{\ell \in \varphi^*(\ell^*) : \ell \leq \ell(m)\}$.
2. The nodes in $c(\ell, m)$ that are in $\mathcal{S}(m)$ are consecutive, i.e., if $\ell(m) \geq \ell \in \varphi^*(\ell^*)$ then there exist $u(\ell, m)$ and $v(\ell, m)$ such that $\{i : i + \ell I \in \mathcal{S}(m)\} = \{i \in \beta^*(i^*) : u(\ell, m) \leq i \leq v(\ell, m)\}$.
3. $u(\ell, m) \leq u(\varphi(\ell), m) \leq v(\ell, m) \leq v(\varphi(\ell), m)$, assuming all of these numbers are defined.

PROOF. Let $r(n, m) \equiv r(n) \cap \mathcal{N}^*(m)$ and $a(n, m) \equiv a(n) \cap \mathcal{N}^*(m)$. The topology of $\mathcal{N}^*(m)$ implies that either $|r(n, m)| = 1$ or $r(n, m) = \emptyset$, and either $|a(n, m)| = 1$ or $a(n, m) = \emptyset$. By (5), if $n \in \mathcal{S}(m)$ and $n \neq m$ then $\mathcal{S}(n) \cap [a(n, m) \cup r(n, m)] \neq \emptyset$. This implies that

$$\begin{aligned}
& \text{if } n \in \mathcal{S}(m) , \text{ there is a sequence } \{n = n_0 , n_1 , \dots , n_K = m\} \\
& \text{such that } n_k \in \mathcal{S}(m) \text{ and } n_{k+1} \in [a(n_k, m) \cup r(n_k, m)] \\
& \text{for all } 0 \leq k < K .
\end{aligned} \tag{6}$$

(6) implies that Assertion 1 holds. By (5) we have that

$$\text{if } \emptyset \neq a(n, m) \in \mathcal{S}(m) \text{ and } \emptyset \neq r(n, m) \in \mathcal{S}(m) \text{ then } n \in \mathcal{S}(m). \tag{7}$$

Assume that Assertion 2 above is false. Then Q is nonempty where $Q \equiv \{n \in c(\ell, m): 0 \leq \ell < L, n \notin \mathcal{S}(m), \text{ there is an } n' \in \mathcal{S}(m) \cap c(\ell, m) \text{ such that } n' < n, \text{ and there is an } n' \in \mathcal{S}(m) \cap c(\ell, m) \text{ such that } n' > n\}$. Let n^* be the element of Q with minimal index. The definition of Q implies that n^* is not in the same row as m . If n^* were in the same column as m , the fact that there is an $n' > n, n' \in \mathcal{S}(m) \cap c(\ell, m)$ would contradict (6). Thus $a(n^*, m) \neq \emptyset$ and $r(n^*, m) \neq \emptyset$. The minimality of n^* implies that $a(n^*, m) \in \mathcal{S}(m)$ and $b(n^*, m) \in \mathcal{S}(m)$, which contradicts (7). Thus Q is empty and Assertion 2 above holds.

Assertion 2 and (6) imply that $u(\ell, m) \leq u(\varphi(\ell), m) \leq v(\ell, m)$. Assertion 2 and (7) imply that $v(\ell, m) \leq v(\varphi(\ell), m)$, so Assertion 3 holds. $\square$

If the values of $u(\ell, m)$ and $v(\ell, m)$ of Lemma 9 are known for all ℓ, m then we can readily identify $\mathcal{S}^{a(n)} \setminus \mathcal{S}^{r(n)}$ and $\mathcal{S}^{r(n)} \setminus \mathcal{S}^{a(n)}$. Our algorithm uses data structures that are closely related to $u(\ell, m)$ and $v(\ell, m)$.

3. THE ALGORITHMS.

We are now ready to present the algorithms that solve (P). Algorithm 1 is the main algorithm, described in general terms in Section 2. It computes $T(C_n)$ for all n .

Algorithm 2 computes the set $\mathcal{S}$ of indices of the optimal clusters, the optimal clusters $\{C_n : n \in \mathcal{S}\}$, and the reorder intervals $(T_n : n \in \mathcal{N})$.

Algorithm 1 appears in Figure 9 below. The loop starting with Line 2 and ending with Line 13 computes $T(C_n)$, starting with $n = 1$ and proceeding up to $n = N$. In Line 3 we set $K^n = K(C_n(\infty))$ and $g^n = g(C_n(\infty))$. The loop starting with Line 4 and ending with Line 12 performs the changes that occur in $K^n = K(C_n(t))$ and $g^n = g(C_n(t))$ as t is parametrically decreased from ∞ to $T(C_n)$, as described in Lemma 8. When Line 13 is executed, $T(C_n)$ has been found.

Figure 9: Algorithm 1.

```

For  $\ell = 0$  to  $L - 1$ .
  For  $i = 0$  to  $I - 1$  do:
     $n \leftarrow i + \ell I$ .
1     $\mathcal{U}(n) \leftarrow \bigcup_{m \in r(n)} \mathcal{U}(m)$  and  $\mathcal{L}(n) \leftarrow \bigcup_{m \in r(n)} \mathcal{L}(m)$ .
  Next  $i$ .
2  For  $i = 1$  to  $I$  do:
     $n \leftarrow i + \ell I$ .
3     $K^n \leftarrow \sum_{m \in \mathcal{N}^n \setminus \mathcal{N}^{r(n)}} K_m$ ;  $g^n \leftarrow \sum_{m \in \mathcal{N}^n \setminus \mathcal{N}^{r(n)}} g_m$ .
4    If  $\mathcal{U}(n) \cup \mathcal{L}(n) = \emptyset$  then goto Line 13.
5    Let  $m \in \mathcal{U}(n) \cup \mathcal{L}(n)$  and  $K^m/g^m \geq K^{m^*}/g^{m^*} \ \forall \ m^* \in \mathcal{U}(n) \cup \mathcal{L}(n)$ .
    If  $K^n/g^n > K^m/g^m$  then goto Line 13.
6    If  $m \in \mathcal{L}(n)$  then do:
         $K^n \leftarrow K^n - K^m$ ,  $g^n \leftarrow g^n - g^m$ .
7         $\mathcal{L}(n) \leftarrow \mathcal{L}(n) \setminus \{m\}$ ,  $\mathcal{L}(b(n)) \leftarrow \mathcal{L}(b(n)) \cup \{m\}$ .
        End,
8    Else do:
9         $K^n \leftarrow K^n + K^m$ ,  $g^n \leftarrow g^n + g^m$ ,  $\mathcal{U}(n) \leftarrow \mathcal{U}(n) \setminus \{m\}$ .
10       If  $m \notin \mathcal{L}(b(n))$  then  $\mathcal{U}(b(n)) \leftarrow \mathcal{U}(b(n)) \cup \{m\}$ ,
11       Else  $\mathcal{L}(b(n)) \leftarrow \mathcal{L}(b(n)) \setminus \{m\}$ .
        End.
12    Goto Line 4.
13     $\mathcal{U}(n) \leftarrow \mathcal{U}(n) \cup \{n\}$ ;  $\mathcal{L}(b(n)) \leftarrow \mathcal{L}(b(n)) \cup \{n\}$ ;  $T(C_n) \leftarrow \sqrt{K^n/g^n}$ .
  Next  $i$ 
Next  $\ell$ .

```

Let $n = i + \ell$. When Line 3 of Algorithm 1 is executed for n , $\mathcal{U}(n) = \mathcal{E}^{r(n)} \setminus \mathcal{E}^{a(n)}$ and $\mathcal{L}(n) = \mathcal{E}^{a(n)} \setminus \mathcal{E}^{r(n)}$. After line 13 is executed for n , $\mathcal{U}(n) = \mathcal{E}^n \setminus \mathcal{E}^{a(n)}$ and $\mathcal{L}(n) = \mathcal{E}^{a(n)} \setminus \mathcal{E}^n$. At this point, using the notation of Lemma 9, $m \in \mathcal{U}(n)$ if and only if $n = u(\ell, m)$ and $m \in \mathcal{L}(n)$ if and only if $v(\ell, m) \in a(n)$. After Line 1 is executed for n , $\mathcal{U}(n) = \mathcal{E}^{r(n)} \setminus \mathcal{E}^{r(a(n))}$ and $\mathcal{L}(n) = \mathcal{E}^{r(a(n))} \setminus \mathcal{E}^{r(n)}$.

Suppose that Line 5 has just been executed, and that t and t' satisfy the conditions of Lemma 8. Lines 6 to 12 of Algorithm 1 convert $K(C_n(t))$ and $g(C_n(t))$ to $K(C_n(t'))$ and $g(C_n(t'))$, respectively. Lines 6 through 7 correspond to Assertion 3 of Lemma 8. The changes made to $\mathcal{L}(n)$ and $\mathcal{L}(b(n))$ reflect the fact that $m \in \mathcal{E}^n$, but $m \notin \mathcal{E}^{r(n)}$. Lines 8 through 11 correspond to Assertion 4 of Lemma 8. If $m \notin \mathcal{L}(b(n))$, the changes made to $\mathcal{U}(n)$ and $\mathcal{U}(b(n))$ in Lines 9 and 10 reflect the fact that $m \notin \mathcal{E}^n$, but $m \in \mathcal{E}^{r(n)}$. If $m \in \mathcal{L}(b(n))$ then $m \in \mathcal{E}^{r(n)}$ and $m \notin \mathcal{E}^n \cup \mathcal{E}^{r(b(n))}$. Lemma 9 implies that the bottom left-hand corner $v(\ell(m), m)$ of $\mathcal{S}(m)$ is in $r(n)$, so $m \notin \mathcal{E}^{n'}$ for any $n' \geq n$. The changes made in Lines 9 and 11 reflect this fact.

In Line 1, if $r(n) = \emptyset$ then $\mathcal{U}(n) = \mathcal{L}(n) = \emptyset$. If $b(n)$ does not exist, instructions relating to $\mathcal{U}(b(n))$ and $\mathcal{L}(b(n))$ in Lines 7, 10, 11, and 13 are ignored.

THEOREM 10. Algorithm 1 computes $T(C_n)$ correctly for all $1 \leq n \leq N$.

The proof of Theorem 10 is given in the Appendix C.

Running Time of Algorithm 1.

Clearly the running time of Algorithm 1 is dominated by the time required to manipulate the datasets $\mathcal{U}(n)$ and $\mathcal{L}(n)$ and to perform the additions in Line 3. In Algorithm 1 the datasets $\mathcal{U}(n)$ and $\mathcal{L}(n)$ are not created until Line 1 is executed for n . When Line 1 is executed for $n = f(n')$, $\mathcal{U}(n')$ and $\mathcal{L}(n')$ are no longer needed. We choose

to cannibalize them, using them as sub-structures of the newly-created $\mathcal{U}(n)$ and $\mathcal{L}(n)$. Consequently at any point in time during the execution of the algorithm, $\mathcal{U}(n')$ and $\mathcal{L}(n')$ only exist if Line 1 has been executed for $n = n'$, but not for $n = f(n')$.

LEMMA 11. At all times during the execution of the algorithm,

$$\begin{aligned} |\cup_{\{n: \mathcal{U}(n) \text{ currently exists}\}} \mathcal{U}(n)| &\leq N \text{ and} \\ |\cup_{\{n: \mathcal{L}(n) \text{ currently exists}\}} \mathcal{L}(n)| &\leq N. \end{aligned}$$

In addition, Line 7 and the right half of Line 10 are executed at most ND times.

PROOF. During the execution of the algorithm, m is added to a set $\mathcal{U}(n)$ in one of three ways. When Line 13 is executed for $n = m$, m is added to $\mathcal{U}(m)$. This happens exactly once for each m . In Line 1, m may belong to a newly-created set $\mathcal{U}(n)$. In this case, m also belongs to a set $\mathcal{U}(n')$, $n' \in r(n)$, which is destroyed in the process of creating $\mathcal{U}(n)$. Finally, in Lines 9 and 10, m may be removed from a set $\mathcal{U}(n)$ and added to the set $\mathcal{U}(b(n))$. This can not happen more than D times for each m where D is the depth of the item tree. Thus the right half of Line 10 is executed at most ND times. In addition, at all times during the execution of the algorithm, m belongs to at most one of the sets $\mathcal{U}(n)$ that currently exist. Consequently $|\cup_{\{n: \mathcal{U}(n) \text{ currently exists}\}} \mathcal{U}(n)| \leq N$. Similarly $|\cup_{\{n: \mathcal{L}(n) \text{ currently exists}\}} \mathcal{L}(n)| \leq N$ and Line 7 is executed at most ND times. $\square$

The sets $\mathcal{U}(n)$ and $\mathcal{L}(n)$ are implemented using 2 – 3 trees with unordered leaves [Aho et al, 1974]. The operations performed on these sets, the running times of these operations, and the lines of the algorithm the operations appear in are shown in Table 1.

Table 1.

Operation	Running Time	Lines in the Algorithm
Disjoint Union	$O(\log N)$	1,7,10
Find Largest Element	$O(\log N)$	5
Delete an Element	$O(\log N)$	7,9,11
Determine whether m is in the Set	$O(1)$	10

LEMMA 12. Algorithm 1 can be implemented to run in $O(ND \log N)$ time.

PROOF. The total number of disjoint unions required to implement Line 1 is $O(N)$ because each m satisfies $m \in r(n)$ for at most one n . In light of Table 1 and Lemma 11, it suffices to show that Lines 5, 9, and 11 of Algorithm 1 are executed at most $O(ND)$ times each, and that Line 3 can be implemented in the required time. The proof of Lemma 11 implies that m enters $\cup_{\{n: \mathcal{U}(n) \text{ currently exists}\}} \mathcal{U}(n)$ only once, so it can leave this set only once. This occurs when Line 11 is executed for m . Therefore Line 11 is executed at most N times. The algorithm and Lemma 11 imply that Lines 5 and 9 are executed at most $O(ND)$ times.

With regard to Line 3, we can write

$$K^*(n) \equiv \sum_{m \in \mathcal{N}^n \setminus \mathcal{N}^{r(n)}} K^m = K^n + \sum_{m \in a(n)} K^*(m).$$

Since for each m there is a unique $n = b(m)$ such that $m \in a(n)$, we can use this formula to compute $K^*(n)$ for all n in $O(N)$ time. Thus Line 3 of Algorithm 1 can be implemented to run in $O(N)$ time. $\square$

Algorithm 2.

Given the values $T(C_n)$, $n \in \mathcal{N}$, Algorithm 2 computes the cluster indices $\mathcal{C}$, the clusters C_n , $n \in \mathcal{C}$, and the vector of reorder intervals $(T_n: n \in \mathcal{N})$. It appears in Figure 10 below.

THEOREM 13. Algorithm 2 computes $(T_n: n \in \mathcal{N})$, $\mathcal{C}$, and $\{C_n: n \in \mathcal{C}\}$.

The proof of Theorem 13 is found in Appendix D. Algorithm 2 can be implemented to run in $O(N)$ time.

Figure 10: Algorithm 2.

$\mathcal{C} = \emptyset$.

For $n = N$ down to 1 do:

 If $f(n)$ exists and $T(C_n) > T_{f(n)}$ then do:

$T_n \leftarrow T_{f(n)}$ and $C_m \leftarrow C_m \cup \{n\}$ where $f(n) \in C_m$,

 If $b(n)$ exists and $T(C_n) < T_{b(n)}$ then do:

$T_n \leftarrow T_{b(n)}$ and $C_m \leftarrow C_m \cup \{n\}$ where $b(n) \in C_m$,

 If neither of the above conditions hold then do:

$\mathcal{C} \leftarrow \mathcal{C} \cup \{n\}$, $T_n \leftarrow T(C_n)$, and $C_n \leftarrow \{n\}$.

Next n .

4. DIRECTIONS FOR FURTHER WORK.

In addition to the network structure studied in this problem, $O(N \log N)$ algorithms exist for (P) when the network $\mathcal{G}$ has the structure of a series-parallel digraph [Lawler 1978, Roundy 1986] or the structure of a directed tree [Jackson and Roundy 1985]. There may be other useful network structures for which efficient algorithms exist. There may be a class of networks that includes all of these network structures as special cases, and an efficient algorithm that solves (P) for this more general class. As is discussed in

[Roundy 1986], such an algorithm would also be useful in solving the isotonic regression problem and the precedence—constrained, single—machine, weighted—tardiness scheduling problem.

For this and several other types of multi—stage inventory systems, nested schedules are much easier to compute than non—nested schedules, but are potentially much more expensive. In these cases an efficient, effective algorithm which takes a good nested schedule and improves it by selectively relaxing certain of the nestedness constraints would potentially be of great value.

Zheng [1987] has pointed out that 94%—effective policies can be computed for problems of this sort by solving a limited number of max—flow problems. If a linear—time max—flow algorithm could be found for this network, Zheng's approach would probably be competitive with this algorithm.

Finally, it may be possible to extend this work by allowing backorders [Mitchell 1987].

Acknowledgement.

The author wishes to thank the referees for their careful review of the paper and their insightful comments. This research was supported by the National Science Foundation under grant DMC—8451984 and by research grants from AT&T Information Systems and DuPont Corporation.

APPENDIX A: THE PERTURBATION.

In this appendix we give a lexicographic scheme for implementing the algorithm, which uses perturbed data, in a numerically stable manner. We also prove Lemma 1 and show that as the perturbation tends to zero, the solution to (P) for the perturbed data approaches an optimal solution to (P) for the original data.

Let $K'(C) \equiv \sum_{n \in C} K'_n$ and $g'(C) \equiv \sum_{n \in C} g'_n$. Then

$$T(C) = \left[K'(C) + \sum_{n \in C} \epsilon^{2(N+1-n)} \times 1(n \in B) \right] / \left[g'(C) + \epsilon \times |C| \right]$$

where $1(S)$ is one if S is true and zero if S is false. Let

$$V(C) \equiv \frac{1}{g'(C) + \epsilon |C|} [K'(C), 1(N \in C), 1(N-1 \in C), \dots, 1(1 \in C)]. \quad (8)$$

Clearly, for ϵ sufficiently small, $T(B) > T(C)$ if and only if $V(B)$ is lexicographically greater than $V(C)$. This proves Lemma 1.

LEMMA 1. If $B \subset \mathcal{N}$ then $0 < T(B) < \infty$. If $B, C \subset \mathcal{N}$ and $B \neq C$ then $T(B) \neq T(C)$.

The lexicographic scheme for implementing the algorithms goes as follows. In Algorithm 1 or Algorithm 2, we only need to compare $T(B)$ with $T(C)$ if B, C are nonempty and if the elements of B and C with largest index and are known and distinct. Let

$$W(C) \equiv \left[\frac{K'(C)}{g'(C)}, \frac{K'(C)}{|C|}, \max \{n: n \in C\} \right].$$

In comparing $W(C)$ to $W(B)$ for subsets C, B of $\mathcal{E}$, we take comparisons of the form $\frac{0}{0} = \frac{x}{y}$ to be true for all $x, y \in \mathbb{R}^+$.

LEMMA 14. If $\max \{n: n \in B\} \neq \max \{n: n \in C\}$ then $T(B) > T(C)$ if and only if $W(B)$ is lexicographically greater than $W(C)$.

PROOF. The result follows easily by using (8) and considering the following cases:

$$\begin{aligned} g'(B) &= g'(C) = 0; \\ g'(B) &> 0, g'(C) > 0; \\ g'(B) &= 0, g'(C) > 0, K'(B) > 0; \\ g'(B) &= 0, g'(C) > 0, K'(B) = 0, K'(C) = 0; \\ g'(B) &= 0, g'(C) > 0, K'(B) = 0, K'(C) > 0. \quad \square \end{aligned}$$

Finally we prove that a solution to (P) for the perturbed data yields an optimal solution to (P) for the original data. In evaluating the cost of a solution to (P), we assume that $\frac{0}{0} = 0$.

LEMMA 15. Let $\mathcal{T}^*(\epsilon)$ solve (P) for the ϵ -perturbation of the data described above. Then there is a limit point $\mathcal{T}^*$ of $\mathcal{T}^*(\epsilon)$ as ϵ tends to zero, and $\mathcal{T}^*$ solves (P) for the original data.

PROOF. Let $c(\mathcal{T}, \epsilon) \equiv \sum_n \left[\left[K'_n + \epsilon^{2(N+1-n)} \right] / T_n + (g'_n + \epsilon) T_n \right]$, the objective function of (P) for the perturbed data, as a function of the perturbation ϵ . Clearly $c(\mathcal{T}, \epsilon)$ is non-decreasing in ϵ . Let $e = (1, \dots, 1)$, and let $\mathcal{D} \equiv \{ \mathcal{T} \in \mathbb{R}^N : \mathcal{T} \geq 0, c(\mathcal{T}, 0) \leq c(e, 1) \}$. Clearly all optimal solutions to (P) for the original data are in $\mathcal{D}$.

Claim 1: $\mathcal{D}$ is bounded. Since $g'_n > 0$ for all nodes n with no predecessors in $\mathcal{G}$, and all terms of $c(\mathcal{T}, \epsilon)$ are non-negative, the claim follows from (2).

Claim 2: $\mathcal{D}$ is closed. Let $\alpha \equiv \min \{K'_n : K'_n > 0\}$ and $D \equiv \{\mathcal{T} \in \mathbb{R}^N : \mathcal{T} \geq 0 \text{ and } T_n \geq \alpha/c(e,1) \text{ if } K'_n > 0\}$. D is closed, $c(\mathcal{T},0)$ is continuous in $\mathcal{T}$ on D , and $\mathcal{D}$ is a subset of D . Therefore $\mathcal{D}$ is closed.

Claim 3: $\mathcal{T}^*(\epsilon) \in \mathcal{D}$ for all $0 < \epsilon \leq 1$. $c(\mathcal{T}^*(\epsilon),0) \leq c(\mathcal{T}^*(\epsilon),\epsilon) \leq c(e,\epsilon) \leq c(e,1)$.

Let $\mathcal{T}^*$ be a limit point of $\mathcal{T}^*(\epsilon)$ as $\epsilon \rightarrow 0$. Then there are $\{\epsilon_1, \epsilon_2, \dots\}$ such that $\epsilon_k \rightarrow 0$ and $\mathcal{T}^*(\epsilon_k) \rightarrow \mathcal{T}^*$ as $k \rightarrow \infty$. We will see that $\mathcal{T}^*$ solves (P) for the original data.

Claim 4: $c(\mathcal{T}^*,0) \leq \liminf_{k \rightarrow \infty} c(\mathcal{T}^*(\epsilon_k),\epsilon_k)$. Let $\mathcal{T}^*(\epsilon) = (T_1^*(\epsilon), \dots, T_N^*(\epsilon))$.

Clearly $(g'_n + \epsilon_k) \times T_n^*(\epsilon_k) \rightarrow g'_n T_n^*$ as $k \rightarrow \infty$. If $T_n^* > 0$ then

$\left[K + \epsilon^{2(N+1-n)} \right] / T_n^*(\epsilon_k) \rightarrow K'_n / T_n^*$. If $T_n^* = 0$ then $K'_n = 0$, so

$$K'_n / T_n^* = 0 \leq \left[K'_n + \epsilon^{2(N+1-n)} \right] / T_n^*(\epsilon_k).$$

Let $\mathcal{T}$ be an arbitrary element of $\mathcal{D}$. We need to show that $c(\mathcal{T}^*,0) \leq c(\mathcal{T},0)$.

Let $\mathcal{T}(\epsilon) = (T_1(\epsilon), \dots, T_N(\epsilon))$ be given by $T_n(\epsilon) = \max \{\epsilon, T_n\}$.

Claim 5: $c(\mathcal{T}(\epsilon),\epsilon) \rightarrow c(\mathcal{T},0)$ as $\epsilon \rightarrow 0$. Clearly $(g'_n + \epsilon) \times T_n(\epsilon) \rightarrow g'_n T_n$. If

$T_n > 0$ then $\left[K'_n + \epsilon^{2(N+1-n)} \right] / T_n(\epsilon) \rightarrow K'_n / T_n$. Since $\mathcal{T} \in \mathcal{D}$, if $T_n = 0$ then $K'_n = 0$, so $K'_n / T_n = 0$ and $\left[K'_n + \epsilon^{2(N+1-n)} \right] / T_n(\epsilon) = \epsilon^{2(N+1-n)} / \epsilon \rightarrow 0$ as $\epsilon \rightarrow 0$.

Thus $c(\mathcal{T}^*,0) \leq \liminf_{k \rightarrow \infty} c(\mathcal{T}^*(\epsilon_k),\epsilon_k) \leq \liminf_{k \rightarrow \infty} c(\mathcal{T}(\epsilon_k),\epsilon_k) = c(\mathcal{T},0)$. $\square$

APPENDIX B: PROOF OF LEMMA 4.

LEMMA 4. Let $m \in \mathcal{N}^n \cap \mathcal{N}^{n^*}$, $n = i + \ell I$, $n^* = i^* + \ell^* I$, $i \leq i^*$, and $\ell \geq \ell^*$. Then $T(m, n) \leq T(m, n^*)$.

PROOF. Let $n'' = i + \ell^* I$. In Figure 5, $n = 23$, $n^* = 15$, and $n'' = 13$. Then $\mathcal{N}^{n''} = \mathcal{N}^n \cap \mathcal{N}^{n^*}$, and $\mathcal{N}^{n''}$ is final in $\mathcal{N}^n$ and initial in $\mathcal{N}^{n^*}$. Recall that $T(m, n^*)$ is the optimal value of T_m in the solution to (P^{n^*}) . Let $\mathcal{J}^* \equiv (T(m, n) : m \in \mathcal{N}^n \setminus \mathcal{N}^{n''})$, $\mathcal{J}^n \equiv (T(m, n) : n \in \mathcal{N}^{n''})$, and $\mathcal{J}^{n''} \equiv (T(m, n'') : n \in \mathcal{N}^{n''})$. Let $\mathcal{J}^N$ be the element-wise minimum of $\mathcal{J}^n$ and $\mathcal{J}^{n''}$ and let $\mathcal{J}^X$ be their element-wise maximum. Let $f(\mathcal{J}^n) \equiv \sum_{m \in \mathcal{N}^{n''}} [K_m/T(m, n) + g_m T(m, n)]$, and let $f(\mathcal{J}^{n''})$, $f(\mathcal{J}^N)$, and $f(\mathcal{J}^X)$ be similarly defined.

Since $\mathcal{J}^X$ is feasible for $(P^{n''})$, $f(\mathcal{J}^X) \geq f(\mathcal{J}^{n''})$. The constraints linking $\mathcal{N}^n \setminus \mathcal{N}^{n''}$ and $\mathcal{N}^{n''}$ are all of the form $T_m \geq T_n$, $m \in \mathcal{N}^n \setminus \mathcal{N}^{n''}$, $n \in \mathcal{N}^{n''}$. Therefore $\mathcal{J}^N \cup \mathcal{J}^*$ is feasible for (P^n) , so $f(\mathcal{J}^N) \geq f(\mathcal{J}^n)$. The additivity of f implies that $f(\mathcal{J}^X) + f(\mathcal{J}^N) = f(\mathcal{J}^n) + f(\mathcal{J}^{n''})$, so $f(\mathcal{J}^X) = f(\mathcal{J}^{n''})$ and $f(\mathcal{J}^N) = f(\mathcal{J}^n)$. Applying Lemma 2 to $(P^{n''})$ and (P^n) we have $\mathcal{J}^X = \mathcal{J}^{n''}$ and $\mathcal{J}^N = \mathcal{J}^n$, so $\mathcal{J}^{n''} = \mathcal{J}^X \geq \mathcal{J}^N = \mathcal{J}^n$ and $T(m, n) \leq T(m, n'')$ for all $m \in \mathcal{N}^{n''}$. The proof that $T(m, n^*) \leq T(m, n'')$ for all $m \in \mathcal{N}^{n''}$ is similar. $\square$

APPENDIX C: PROOF OF THEOREM 10.

In this appendix we prove Theorem 10. The following lemma is used in the proof.

LEMMA 16. For each m, ℓ the following three properties hold whenever Line 4 of Algorithm 1 is executed.

- a. There is at most one i , $0 \leq i < I$ such that $m \in \mathcal{U}(i + \ell I)$.
- b. There is at most one j , $0 \leq j < I$ such that $m \in \mathcal{L}(j + \ell I)$.
- c. If $m \in \mathcal{L}(j + \ell I)$ then $m \in \mathcal{U}(i + \ell I)$ for some $i < j$.

PROOF. The proof follows directly from Algorithm 1 by induction on the iterations of Line 4. $\square$

THEOREM 10. Algorithm 1 computes $T(C_n)$ correctly for all $1 \leq n \leq N$.

PROOF. In order to describe the evolution of the sets $\mathcal{U}(n)$ and $\mathcal{L}(n)$ we introduce the following notation. Let $\mathcal{U}(n^*, n'')$ and $\mathcal{L}(n^*, n'')$ be the values of $\mathcal{U}(n^*)$ and $\mathcal{L}(n^*)$ just before Line 3 is executed for $n = n'' + 1$, i.e., after $T(C_{n''})$ has been computed and before $T(C_{n''+1})$ is computed. Let $\ell'' I \leq n'' < I + \ell'' I$, i.e., let node $n'' + 1$ be in the column of $\mathcal{G}$ that corresponds to location ℓ'' . Then $\mathcal{U}(n^*, n'')$ and $\mathcal{L}(n^*, n'')$ exist if $n^* \leq I + \ell^* I$, i.e., if n^* is in a column of $\mathcal{G}$ that corresponds to a location $\ell^* \leq \ell''$.

Since $b(n) > n$, Algorithm 1 implies that

$$\mathcal{U}(n^*, n) = \mathcal{U}(n^*, n^*) \text{ and } \mathcal{L}(n^*, n) = \mathcal{L}(n^*, n^*) \text{ for all } n \geq n^*. \quad (9)$$

Lemma 16 implies that

$$\mathcal{U}(n^*, n) \cap \mathcal{L}(n^*, n) = \emptyset \text{ for all } n^*, n \in \mathcal{N}. \quad (10)$$

For all node pairs (n^*, n) we define

$$a(n^*, n) \equiv \{m : m \in a(n^*) , m \leq n\} \cup \{m : m \in r(k) , k \in a(n^*) , k > n\}.$$

Note that

$$a(n^*, n) = a(n^*) \text{ for all } n \geq n^* - 1 . \quad (11)$$

We will show that the following two properties hold for all n .

Property A: When Line 13 is executed for n , $K^n = K(C_n)$ and $g^n = g(C_n)$.

Property B: Let $\mathcal{U}(n^*, n)$ and $\mathcal{L}(n^*, n)$ exist. Then if $n^* > n$ we have

$$\begin{aligned} \mathcal{U}(n^*, n) &= \mathcal{C}^{r(n^*)} \setminus \mathcal{C}^{a(n^*, n)} \text{ and} \\ \mathcal{L}(n^*, n) &= \mathcal{C}^{a(n^*, n)} \setminus \mathcal{C}^{r(n^*)}, \end{aligned} \quad (12)$$

and if $n^* = n$ we have

$$\begin{aligned} \mathcal{U}(n, n) &= \mathcal{C}^n \setminus \mathcal{C}^{a(n)} = \mathcal{C}^n \setminus \mathcal{C}^{a(n, n)} \text{ and} \\ \mathcal{L}(n, n) &= \mathcal{C}^{a(n)} \setminus \mathcal{C}^n = \mathcal{C}^{a(n, n)} \setminus \mathcal{C}^n . \end{aligned} \quad (13)$$

Since Property A implies that Theorem 10 holds, this will complete the proof. By (11) and (9), if (13) holds then for all $n^* \leq n$ we have

$$\mathcal{U}(n^*, n) = \mathcal{U}(n^*, n^*) = \mathcal{E}^{n^*} \setminus \mathcal{E}^{a(n^*)}, \text{ and}$$

$$\mathcal{L}(n^*, n) = \mathcal{L}(n^*, n^*) = \mathcal{E}^{a(n^*)} \setminus \mathcal{E}^{n^*}. \quad (14)$$

The proof is by induction on n . We first show that Property B holds for $n = 0$.

$\mathcal{U}(n^*, 0)$ and $\mathcal{L}(n^*, 0)$ are defined if $n^* \leq I$. Assuming that $n^* \leq I$ we have

$\mathcal{U}(n^*, 0) = \mathcal{L}(n^*, 0) = r(n^*) = a(n^*, 0) = \emptyset$, so (12) holds. Since $n^* \neq n$ for all $n \in \mathcal{N}$, (13) holds vacuously.

Assume that Property B holds for $n - 1$. We will show that Property A holds for n .

By the induction hypothesis and (11),

$$\mathcal{U}(n) = \mathcal{U}(n, n - 1) = \mathcal{E}^{r(n)} \setminus \mathcal{E}^{a(n, n - 1)} = \mathcal{E}^{r(n)} \setminus \mathcal{E}^{a(n)}.$$

$$\mathcal{L}(n) = \mathcal{E}^{a(n)} \setminus \mathcal{E}^{r(n)}.$$

Let $t_0 = \infty$ and let $\{T(C_m) : m \in \mathcal{U}(n) \cup \mathcal{L}(n)\} = \{t_k : 1 \leq k \leq K\}$ where $t_1 > t_2 > t_3 \dots > t_K$. By induction on k , Lemma 8 implies that the k^{th} time that Line 5 is executed for n , $K^n = K(C_n(t))$ and $g^n = g(C_n(t))$ for all $t_{k-1} < t < t_k$.

We need to show that the loop from Line 4 to Line 12 will terminate with

$t = T(C_n(t))$. $T(C_n(t))$ is piecewise constant in t and continuous except at $t = T(C_m)$, $m \in \mathcal{U}(n) \cup \mathcal{L}(n)$. $T(C_n(0)) > 0$ because $K_n > 0$ and $T(C_n(\infty)) < \infty$ because $g_n > 0$, so $t^* \equiv \inf \{t : t \geq T(C_n(t))\}$ exists and is positive. By Assertion 1 of Lemma 8, $t^* = T(C_n(t^*))$. Therefore the loop from Line 4 to Line 12 will terminate, and when it does, Line 13 of Algorithm 1 will be executed with $K^n = K(C_n(t^*))$ and $g^n = g(C_n(t^*))$.

Theorem 7 implies that Property A holds for n .

We now assume that Property A holds for all $n' \leq n$ and that Property B holds for all $n' < n$, and show that Property B holds for n . Two facts need to be shown. The first is that if $\mathcal{U}(n^*, n - 1)$ and $\mathcal{L}(n^*, n - 1)$ exist then (12) and (13) hold for n^* and n . The second is that (12) holds for newly-created sets $\mathcal{U}(n^*)$ and $\mathcal{L}(n^*)$, i.e., if $n = \ell I$ then (12) holds for all n^* , $1 + \ell I \leq n^* \leq I + \ell I$.

The first fact easily implies the second fact. If $n'' \in r(n^*)$, $\ell I < n^* \leq I + \ell I$, and $n = \ell I$ then (14) holds for $\mathcal{U}(n'', n)$ and $\mathcal{L}(n'', n)$. Thus (12) holds for $\mathcal{U}(n^*, n)$ and $\mathcal{L}(n^*, n)$ and the second fact is proven. We conclude our proof of Theorem 10 by proving the first fact.

Algorithm 1 implies that $\mathcal{U}(n^*, n) = \mathcal{U}(n^*, n-1)$ and $\mathcal{L}(n^*, n) = \mathcal{L}(n^*, n-1)$ unless $n^* = n$ or $n^* = b(n)$. The definition of $a(n^*, n)$ implies that $a(n^*, n) = a(n^*, n-1)$ if $n^* \neq b(n)$. Therefore it suffices to show that

$$\mathcal{U}(n, n) = \mathcal{E}^n \setminus \mathcal{E}^{a(n)} \equiv D, \quad (15a)$$

$$\mathcal{L}(n, n) = \mathcal{E}^{a(n)} \setminus \mathcal{E}^n \equiv E, \quad (15b)$$

$$\mathcal{U}(b(n), n) = \mathcal{E}^{r(b(n))} \setminus \mathcal{E}^{a(b(n), n)} \equiv G, \text{ and} \quad (15c)$$

$$\mathcal{L}(b(n), n) = \mathcal{E}^{a(b(n), n)} \setminus \mathcal{E}^{r(b(n))} \equiv H. \quad (15d)$$

Since $a(n, n-1) = a(n, n) = a(n)$, the induction hypothesis implies

$$\mathcal{U}(n, n-1) = \mathcal{E}^{r(n)} \setminus \mathcal{E}^{a(n)} \equiv d, \quad (16a)$$

$$\mathcal{L}(n, n-1) = \mathcal{E}^{a(n)} \setminus \mathcal{E}^{r(n)} \equiv f, \quad (16b)$$

$$\mathcal{U}(b(n), n-1) = \mathcal{E}^{r(b(n))} \setminus \mathcal{E}^{a(b(n), n-1)} \equiv g, \text{ and} \quad (16c)$$

$$\mathcal{L}(b(n), n-1) = \mathcal{E}^{a(b(n), n-1)} \setminus \mathcal{E}^{r(b(n))} \equiv h. \quad (16d)$$

The definition of $a(n^*, n)$ implies that

$$a(b(n), n-1) \oplus \{n\} = a(b(n), n) \oplus r(n) \quad (17)$$

where $\oplus$ denotes the union of two disjoint sets. A useful fact that follows directly from (17) is the following.

If either $m \in \mathcal{E}^{r(n)} \cap \mathcal{E}^n$ or $m \notin \mathcal{E}^{r(n)} \cup \mathcal{E}^n$ then

$$m \in G \Leftrightarrow m \in g \text{ and } m \in H \Leftrightarrow m \in h. \quad (18)$$

We claim that

$$\text{if } m \in \mathcal{E}^{r(n)} \setminus \mathcal{E}^n \text{ then } m \in \mathcal{E}^{a(b(n), n-1)} \setminus \mathcal{E}^{a(b(n), n)}. \quad (19)$$

In (17), let $W \equiv a(b(n), n-1) \setminus r(n) = a(b(n), n) \setminus \{n\}$. The topology of $\mathcal{G}$ implies that $\mathcal{N}^W \cap \mathcal{N}^n = \mathcal{N}^W \cap \mathcal{N}^{r(n)} = \emptyset$. Therefore $\mathcal{E}^W \cap \mathcal{E}^n = \mathcal{E}^W \cap \mathcal{E}^{r(n)} = \emptyset$. This and (17) imply that (19) holds.

Algorithm 1 implies that $\mathcal{U}(n, n) \subset \mathcal{U}(n, n-1)$ and $\mathcal{L}(n, n) = \mathcal{L}(n, n-1)$. Let

$$X \equiv \mathcal{L}(n, n-1) \setminus \mathcal{L}(n, n)$$

$$Y \equiv \left[\mathcal{U}(n, n-1) \setminus \mathcal{U}(n, n) \right] \setminus \mathcal{L}(b(n), n-1)$$

$$Z \equiv \left[\mathcal{U}(n, n-1) \setminus \mathcal{U}(n, n) \right] \cap \mathcal{L}(b(n), n-1)$$

By (10),

$$\mathcal{L}(n,n), \mathcal{U}(n,n), X, Y, Z \text{ are all mutually disjoint.} \quad (20)$$

Algorithm 1 implies that

$$\mathcal{L}(n,n-1) = \mathcal{L}(n,n) \oplus X \text{ and } \mathcal{U}(n,n-1) = \mathcal{U}(n,n) \oplus Y \oplus Z, \quad (21)$$

$$\mathcal{L}(b(n),n-1) \oplus X = \mathcal{L}(b(n),n) \oplus Z, \quad (22)$$

$$\mathcal{U}(b(n),n-1) \oplus Y = \mathcal{U}(b(n),n), \quad (23)$$

$$m \in \mathcal{L}(n,n) \cup \mathcal{U}(n,n) \Rightarrow T(C_m) < T(C_n), \text{ and} \quad (24)$$

$$m \in X \cup Y \cup Z \Rightarrow T(C_m) > T(C_n). \quad (25)$$

For a given m and for each equation in (15a – d), we will show that m is in the left side of the equation if and only if it is in the right side.

Case 1: $m = n$. Note that $n \in a(b(n),n)$. Clearly $m = n$ is in D and H , but not in E or G . The algorithm implies that m is in $\mathcal{U}(n,n)$ and $\mathcal{L}(b(n),n)$, but not in $\mathcal{L}(n,n)$ or $\mathcal{U}(b(n),n)$.

Case 2: $m \notin \mathcal{U}(n,n-1) \cup \mathcal{L}(n,n-1)$. By (21), $m \notin \mathcal{U}(n,n)$ and $m \notin \mathcal{L}(n,n)$. By (16a,b) and (5), $m \in \mathcal{C}^{a(n)} \Leftrightarrow m \in \mathcal{C}^{r(n)} \Leftrightarrow m \in \mathcal{C}^n$. Thus $m \notin D$ and $m \notin E$, so (15a,b) hold. By (21), $m \notin X \cup Y \cup Z$. By (22) and (23),

$m \in \mathcal{U}(b(n), n) \Leftrightarrow m \in \mathcal{U}(b(n), n-1)$ and $m \in \mathcal{L}(b(n), n) \Leftrightarrow m \in \mathcal{L}(b(n), n-1)$. Since $m \in \mathcal{E}^{r(n)} \Leftrightarrow m \in \mathcal{E}^n$, (18) implies that $m \in G \Leftrightarrow m \in g$ and $m \in H \Leftrightarrow m \in h$, so (16c,d) imply (15c,d).

Case 3: $m \in \mathcal{L}(n, n)$. (10) implies that $m \notin \mathcal{U}(n, n)$. By (21) and (16b), $m \in \mathcal{E}^{a(n)} \setminus \mathcal{E}^{r(n)}$. Thus (24) and (5) imply that $m \notin \mathcal{E}^n$, so $m \notin D$, $m \in E$, and (18a,b) hold. Since $m \notin \mathcal{E}^n \cup \mathcal{E}^{r(n)}$, $m \in G \Leftrightarrow m \in g$ and $m \in H \Leftrightarrow m \in h$ by (18). (20), (22), and (23) imply that $m \in \mathcal{L}(b(n), n) \Leftrightarrow m \in \mathcal{L}(b(n), n-1)$ and $m \in \mathcal{U}(b(n), n) \Leftrightarrow m \in \mathcal{U}(b(n), n-1)$. Thus (16c,d) imply (15c,d).

Case 4: $m \in \mathcal{U}(n, n)$. This case is similar to Case 3. (10) implies that $m \notin \mathcal{L}(n, n)$. By (21) and (16a), $m \in \mathcal{E}^{r(n)} \setminus \mathcal{E}^{a(n)}$. Thus (24) and (5) imply that $m \in \mathcal{E}^n$, so $m \in D$, $m \notin E$, and (15a,b) hold. Since $m \in \mathcal{E}^n \cup \mathcal{E}^{r(n)}$, $m \in G \Leftrightarrow m \in g$ and $m \in H \Leftrightarrow m \in h$ by (18). (20), (22), and (23) imply that $m \in \mathcal{L}(b(n), n) \Leftrightarrow m \in \mathcal{L}(b(n), n-1)$ and $m \in \mathcal{U}(b(n), n) \Leftrightarrow m \in \mathcal{U}(b(n), n-1)$. Thus (16c,d) imply (15c,d).

Case 5: $m \in X$. (20) implies that $m \notin \mathcal{U}(n, n) \cup \mathcal{L}(n, n)$. By (21) and (16b), $m \in \mathcal{E}^{a(n)} \setminus \mathcal{E}^{r(n)}$. Thus (25) and (5) imply that $m \in \mathcal{E}^n$, so $m \notin D \cup E$ and (15a,b) hold. (17) and $m \in \mathcal{E}^n \setminus \mathcal{E}^{r(n)}$ imply that $m \in \mathcal{E}^{a(b(n), n)}$, so $m \notin G$. Lemma 9 and $m \in \mathcal{E}^{a(n)} \setminus \mathcal{E}^{r(n)}$ imply that $m \notin \mathcal{E}^{r(b(n))}$, so $m \in H$. (22) implies that $m \in \mathcal{L}(b(n), n)$, so $m \notin \mathcal{U}(b(n), n)$ by (10). Thus (15c,d) hold.

Case 6: $m \in Y$. (20) implies that $m \notin \mathcal{U}(n, n) \cup \mathcal{L}(n, n)$. By (21) and (16a), $m \in \mathcal{E}^{r(n)} \setminus \mathcal{E}^{a(n)}$. Thus (25) and (5) imply that $m \notin \mathcal{E}^n$, so $m \notin D \cup E$ and (15a,b) hold. (23) implies that $m \in \mathcal{U}(b(n), n)$, so $m \notin \mathcal{L}(b(n), n)$ by (10). By (19),

$m \in \mathcal{E}^{a(b(n),n-1)} \setminus \mathcal{E}^{a(b(n),n)}$. The definition of Y implies that $m \notin \mathcal{L}(b(n),n-1)$, so (16d) implies $m \in \mathcal{E}^{r(b(n))}$. Thus $m \in G$ and $m \notin H$, so (15c,d) hold.

Case 7: $m \in Z$. The proofs of (15a,b), $m \in \mathcal{E}^{r(n)} \setminus \mathcal{E}^{a(n)}$, $m \notin \mathcal{E}^n$, and $m \in \mathcal{E}^{a(b(n),n-1)} \setminus \mathcal{E}^{a(b(n),n)}$ are identical to Case 6. (22) implies that $m \notin \mathcal{L}(b(n),n)$ and that $m \in \mathcal{L}(b(n),n-1)$. (10) then implies that $m \notin \mathcal{U}(b(n),n-1)$, so by (23) we have $m \notin \mathcal{U}(b(n),n)$. Since $m \in \mathcal{L}(b(n),n-1)$, by (16d) we have $m \notin \mathcal{E}^{r(b(n))}$. Thus $m \notin G \cup H$, so (15c,d) hold. $\square$

APPENDIX D: THE PROOF OF THEOREM 13.

The proof of Theorem 13 relies on the Lemma 17 below. Intuitively, Lemma 17 holds because otherwise $C_m \cap C_n$ and C_n would violate the *Concavity* property of Theorem 3.

LEMMA 17. If $n \neq m \in C_n$ and either $f(m) \notin C_n$ or $f(m) = \emptyset$ then $T(C_m) < T(C_n)$. Similarly, if $n \neq m \in C_n$ and either $b(m) \notin C_n$ or $b(m) = \emptyset$ then $T(C_m) > T(C_n)$.

PROOF. Recall that $T(m, n)$ is the value of T_m in the solution to (P^n) . The proof is for a fixed m , by induction on n . It is vacuously true for $n \leq m$. Assuming it is true for all $n^* < n$, we will prove it for n . Let $n \neq m \in C_n$ and either $f(m) \notin C_n$ or $f(m) = \emptyset$.

We claim that $m \notin r^*(n) \equiv \{n\} \cup r(n) \cup r(r(n)) \cup \dots$. If $m \in r^*(n)$ then $f(m) \neq \emptyset$ and $T(C_n) = T(m, n) \leq T(f(m), n) \leq T(f(f(m)), n) \leq \dots \leq T(n, n) = T(C_n)$. This contradicts $f(m) \notin C_n$, so the claim holds.

Since $m \notin r^*(n)$ we have $m \in \mathcal{N}^{a(n)}$, so $m \in C_k$ for some $k \in \mathcal{E}^{a(n)}$. We claim that either $f(m) = \emptyset$ or $f(m) \notin C_k$. Suppose otherwise. Since $f(m) \notin C_n$ we have $T(f(m), n) > T(m, n) = T(C_n)$. Since $f(m) \in \mathcal{N}^{a(n)}$, Theorem 3 and (5) imply that $f(m) \in C_{k^*}$ for some $k^* \in \mathcal{E}^{a(n)}$, $T(C_{k^*}) > T(C_n)$. But (5), $m \in C_n \cap C_k$, and $k \in \mathcal{E}^{a(n)}$ imply that $T(C_n) > T(C_k)$, so $k^* \neq k$ and $f(m) \notin C_k$.

By the induction hypothesis for $n^* = k$, either $m = k$ or $T(C_m) < T(C_k)$. In either case, $T(C_m) \leq T(C_k) < T(C_n)$. This proves the first assertion. The proof of the second assertion is similar. $\square$

A node m is said to be a root of C_n if $m \in C_n$, $f(n) \notin C_n$, and $b(m) \notin C_n$. Clearly n is a root of C_n .

COROLLARY 18. n is the only root of C_n .

PROOF. Lemma 4 implies that if $m \neq n$ is a root of C_n then $T(C_m) < T(C_n) < T(C_m)$, a contradiction. $\square$

THEOREM 13. Algorithm 2 computes $(T_n; n \in \mathcal{N})$, $\mathcal{E}$, and $\{C_n; n \in \mathcal{E}\}$.

PROOF. Let $(T_n; n \in \mathcal{N})$ be the optimal solution to (P). It suffices to show that

$$T_{f(m)} < T(C_m) \Rightarrow T_m = T_{f(m)}, \quad (26a)$$

$$T_{b(m)} > T(C_m) \Rightarrow T_m = T_{b(m)}, \text{ and} \quad (26b)$$

$$T_{f(m)} > T(C_m) > T_{b(n)} \Rightarrow T_m = T(C_m) \Leftrightarrow T_{f(m)} > T_m > T_{b(m)}. \quad (26c)$$

We first show that the equivalence in (26c) holds. By Lemma 1 and Theorem 3, $T_m = T(C_m) \Rightarrow T_{f(m)} > T_m > T_{b(m)}$. The reverse implication is a consequence of Theorem 3 and Corollary 18.

If $T_{f(m)} = T_{b(m)}$ then $T_m = T_{f(m)} = T_{b(m)}$ and (26a – c) hold. Assume that $T_{f(m)} > T_{b(m)}$. The three conditions in the left-hand sides of (26a – c) are clearly mutually exclusive and all-encompassing, as are the three conditions in the right-hand sides of (26a – c). Thus it suffices that the right-hand sides imply the left-hand sides.

Let $T_m = T_{f(m)} > T_{b(m)}$ and let $T_{f(m)} = T(C_n)$ for some n . Lemma 17 implies that $T(C_m) > T(C_n) = T_{f(m)}$.

Let $T_m = T_{b(m)} < T_{f(m)}$ and let $T_{b(m)} = T(C_n)$ for some n . Lemma 17 implies that $T(C_m) < T(C_n) = T_{b(m)}$.

The right-hand side of (26c) implies the left-hand side by Theorem 3. $\square$

REFERENCES

- Aho, A.V., J.E. Hopcroft, and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass. 1974.
- Anily, S., "Integrating Inventory Control and Transportation Planning", Ph.D. dissertation, Columbia University, New York, NY 10027, 1986.
- Anily, S. and A. Federgruen, "Structured Partitioning Problems; Part 1: Theory," Working Paper, Graduate School of Business, Columbia University, New York, NY, 10027, December 1986.
- Anily, S. and A. Federgruen, "Structured Partitioning Problems; Part 2: Applications to Capacitated Two-Stage Multi-Item Production/Inventory Models," Working Paper, Graduate School of Business, Columbia University, New York, NY, 10027, December 1986.
- Iyogun, P.O., "Lower Bounds for Production/Inventory Problems by Cost Allocation," Ph.D. dissertation, University of British Columbia, British Columbia, CANADA, 1987.
- Jackson, P.L., W.L. Maxwell and J.A. Muckstadt, "The Joint Replenishment Problem with a Powers of Two Restriction," *IIE Transactions* 17 No. 1 (1985), 25-32.
- Jackson, P.L., W.L. Maxwell and J.A. Muckstadt, "Determining Optimal Reorder Intervals in Capacitated Production-Distribution Systems," Technical Report No. 624, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY, February 1984 (Forthcoming in *Management Science*).
- Jackson P.L. and R.O. Roundy, "Constructive Algorithms for Planning Production in Multi-Stage Systems with Constant Demand," Technical Report No. 632, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY, August 1985.
- Lawler, E.L., "Sequencing Jobs to Minimize Total Weighted Completion Time Subject to Precedence Constraints," *Annals of Discrete Mathematics* 2 (1978), 75-90.
- Maxwell, W.L., and J.A. Muckstadt, "Establishing Consistent and Realistic Reorder Intervals in Production-Distribution Systems," *Operations Research* 33 (1985), 1316-1341.
- Mitchell, J.S.B., "98%-Effective Lot-Sizing for One-Warehouse, Multi-Retailer Inventory Systems with Backlogging," *Operations Research* 35 (1987), 399-404.
- Muckstadt, J.A. and R.O. Roundy, "Multi-Item, One-Warehouse, Multi-Retailer Distribution Systems," Technical Report No. 646, School of Operations Research and Industrial Engineering, Cornell University, January 1985 (Forthcoming in *Management Science*).
- Queyranne, M., "A Polynomial-Time, Submodular Extension to Roundy's 98%-Effective Heuristic for Production/Inventory Systems", Working Paper No. 1136, Faculty of Commerce and Business Administration, University of British Columbia, August, 1985.

- Roundy, R.O., "94%—Effective Integer—Ratio Lot—Sizing for Multi—Stage Assembly Systems," Technical Report No. 674, School of Operations Research and Industrial Engineering, Cornell University, September 1983, revised August 1985.
- Roundy, R.O., "98%—effective Integer—Ratio Lot—Sizing for One—Warehouse Multi—Retailer Systems," *Management Science* 31 (1985), 1416—1430.
- Roundy, R.O., "A 98%—effective Lot—sizing Rule for a Multi—Product, Multi—Stage Production/Inventory System," *Mathematics of Operations Research* 11 (1986), 699—727.
- Zheng, Y.S., "Replenishment Strategies for Production/Distribution Networks with General Joint Setup Costs," Ph.D. dissertation, Columbia University, New York, NY 10027, 1987.