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

TECHNICAL REPORT NO. 767

January 1988
(Revised October 1989)

Minimizing Separable Convex Objectives on Arbitrarily Directed Trees of Variable Upper Bound Constraints

by

Peter L. Jackson and Robin O. Roundy

Research supported under NSF grants MEA–8121429 and ECS–8404641.

ABSTRACT

An extension of the Economic Order Quantity (EOQ) model to multi–stage production–distribution systems and the isotonic regression problem are known to be equivalent and to be solvable in $O(N^4)$ time. The following specializations of the model are solvable in $O(N \log N)$ time: joint replenishment systems, pure assembly systems, nested policies in pure distribution systems, non–nested policies in one–warehouse multi–retailer systems, nested policies in multi–item, one–warehouse, multi–retailer systems, and systems in which the underlying network is a series–parallel digraph. We show here that a generalization of this problem is solvable in $O(N \log N)$ time whenever the undirected version of the underlying network is a tree. The algorithm we propose is used as a subroutine in an algorithm solving the EOQ problem on general circuitless directed graphs, the subject of a companion paper.

Keywords: production planning, isotonic regression, network optimization, inventory, polynomial–time algorithms.

ACKNOWLEDGEMENTS

The authors wish to extend their sincere thanks to an anonymous referee who devoted considerable effort towards correcting and improving the presentation of these results.

1. INTRODUCTION

Let G denote a circuitless directed graph with node set $N(G)$ and arc set $A(G)$. The graph G is said to be an <u>arbitrarily directed tree</u> if the undirected version of G is a tree. In this paper we present an algorithm to solve problems of the form

$$(P_G) \qquad \underset{\{T_n\,;\ n \in N(G)\}}{\text{minimize}} \ \sum_{n \in N(G)} f_n(T_n) \tag{1.1}$$

$$\text{subject to} \quad T_m \geq T_n \quad \forall \ (m,n) \in A(G) \tag{1.2}$$

where G is an arbitrarily directed tree, and the functions $f_n(\cdot)$, $n \in N(G)$ satisfy Assumptions 2.0–2.2 below. These assumptions are satisfied if, for example, the $f_n(\cdot)$ are strictly convex and continuously differentiable, and if they achieve their minima within a bounded interval.

The motivation for considering problems of the form P_G on arbitrarily directed trees is the subject of a companion paper (Jackson and Roundy, 1987). It is shown there that such "tree problems" arise as sub–problems in an algorithm to solve problems of the form P_G where G is a general circuitless directed graph, not necessarily an arbitrarily directed tree. The importance of such general problems arises from their application in production planning and isotonic regression.

The production planning version of P_G is referred to as PP_G. It is the special case in which the functions $f_n(\cdot)$ are given by EOQ–type cost formulas: for $n \in N(G)$,

$$f_n(T) = K_n/T + g_n T + P(T) \tag{1.3}$$

where $K_n \geq 0$, $g_n \geq 0$, $K_n + g_n > 0$, and $P(\cdot)$ is a penalty function to enforce a non–negativity restriction:

$$P(T) = \begin{cases} 0 & \text{if } \ T \geq 0\,; \\ +\infty & \text{otherwise}\,. \end{cases}$$

Such functions are neither strictly convex nor continuously differentiable but they do satisfy Assumptions 2.0–2.2 below. In production planning applications K_n is the setup cost for operation n , g_n is the holding cost coefficient for the inventory of operation n , and T is the reorder interval for production of operation n. The constraints (1.2) are related to bill of material relationships between different operations.

P_G for general circuitless directed graphs is known to be solvable in at most $O(N^4)$ time where N is the cardinality of $N(G)$ (Maxwell and Muckstadt, 1985; Picard and Queyranne, 1985). The following specializations of PP_G are solvable in $O(N \log N)$ time: pure assembly systems, joint replenishment systems, nested policies in pure distribution systems, non–nested policies in one–warehouse, multi–retailer systems, and nested policies in multi–item, one–warehouse, multi–retailer systems (Jackson, Maxwell, and Muckstadt, 1985; Roundy, 1983, 1984; Muckstadt and Roundy, 1985). For the first three of these specialized models, the graph G is an arbitrarily directed tree.

We unify these latter results by showing that PP_G can be solved in $O(N \log N)$ time whenever G is an arbitrarily directed tree. The so–called Tree Algorithm that we propose also solves P_G if G is an arbitrarily directed tree. In this latter case the running time is $O(N \log N)$ exclusive of at most $2N$ computations of the form

$$\underset{T}{\text{minimize}} \quad f(T,C) = \sum_{n \in C} f_n(T) \tag{1.4}$$

where $C \subseteq N(G)$.

The remainder of the paper is organized as follows. Section 2 introduces the assumptions on the objective functions that are sufficient to guarantee the validity of the algorithm. Section 3 develops the terminology to describe arbitrarily directed trees and to describe the characterization of the solution to P_G. The concepts are illustrated by means of an example. Section 4 presents the Tree Algorithm and illustrates several steps of the algorithm by example. Section 5 establishes the validity of the Tree Algorithm. Section 6 presents an analysis of the running time and space requirements of the Tree Algorithm. Section 7 concludes the paper. A glossary of the notation used is included in the appendices.

2. ASSUMPTIONS

A cluster is defined to be any subset of the node set $N(G)$. Let $\mathscr{N}$ denote the set of all non–empty clusters. Let $\bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty,+\infty\}$.

ASSUMPTION 2.0: The functions $f_n(\cdot)$, $n \in N(G)$ are convex on $\mathbb{R}$.

ASSUMPTION 2.1: There exists a cluster function $T: \mathscr{N} \to \bar{\mathbb{R}}$ such that $T(C)$ uniquely solves (1.4) for each $C \in \mathscr{N}$.

ASSUMPTION 2.2: If C' and C'' are non–empty disjoint clusters with $T(C') < T(C'')$ then

$$T(C') < T(C' \cup C'') < T(C'').$$

The important aspect of Assumption 2.2 is the strictness of the inequalities, as the following lemma points out.

LEMMA 2.3: Under Assumption 2.0 and 2.1, if C' and C'' are non–empty disjoint clusters with $T(C') \leq T(C'')$ then

$$T(C') \leq T(C' \cup C'') \leq T(C'').$$

PROOF: Jackson and Roundy, 1987.

Assumptions 2.0–2.2 are satisfied if, for each $n \in N(G)$, $f_n(\cdot)$ is strictly convex and continuously differentiable, and is minimized by a finite real number.

For the production planning version of the objective function, (1.3), $T(C)$ is given by

$$T(C) = \left[\frac{\sum_{n \in C} K_n}{\sum_{n \in C} g_n} \right]^{1/2} . \qquad (2.1)$$

To simplify notation let $K(C) = \sum_{n \in C} K_n$ and $g(C) = \sum_{n \in C} g_n$. Then $T(C) = (K(C)/g(C))^{1/2}$.

Proposition 2.4: Assuming $K_n, g_n \geq 0$ and $K_n + g_n > 0$ for all $n \in N(G)$, (2.1) is well–defined. Assumptions 2.0 – 2.2 hold for PP_G with $T(C)$ given by (2.1).

Proof: Clearly, Assumption 2.0 is satisfied. The assumption $K_n + g_n > 0$ rules out division of zero by zero, so (2.1) is well defined, though possibly infinite. By examining cases, it is straightforward to verify that (2.1) uniquely minimizes (1.3) over $\bar{\mathbb{R}}$. Thus Assumption 2.1 is satisfied.

Suppose C' and C'' are non–empty disjoint clusters with $T(C') < T(C'')$. The assumption $K_n + g_n > 0 \ \forall \ n \in N(G)$ implies that $0 < T(C' \cup C'') < \infty$. Suppose $T(C') = T(C' \cup C'')$. In that case $g(C')$ and $K(C')$ are both positive. Hence

$$K(C')/g(C') = T(C')^2 = T(C' \cup C'')^2 = (K(C') + K(C''))/(g(C') + g(C'')) ,$$

which implies $T(C') = T(C'')$, a contradiction. Therefore $T(C') < T(C' \cup C'')$. Similarly, $T(C' \cup C'') < T(C'')$. □

The simple structure of (2.1) enables us in Section 6 to tailor the proposed Tree Algorithm to PP_G and bound the running time to $O(N \log N)$. By extension, a family of d–additive convex objectives which includes isotonic regression can also be minimized on G in $O(N \log N)$ time. The following result is Corollary 2.5 of Jackson and Roundy, 1987.

PROPOSITION 2.5: P_G has a unique optimal solution.

3. OPTIMAL PARTITIONS OF ARBITRARILY DIRECTED TREES

In this section we characterize the solution to problem P_G for the special case in which G forms an arbitrarily directed tree. Section 5 below presents a detailed analysis of this characterization. The key results of that analysis are summarized here in preparation to present the Tree Algorithm in the next section. These results are illustrated by means of an example.

The example is based on the arbitrarily directed tree depicted in Figure 1. The results of this section are general to problems of the form P_G but the example is a production planning problem, PP_G, using the setup and holding cost data (K_n, g_n) listed in Table 1. In particular, to simplify the arithmetic, the holding cost factors g_n have been

Node n	Parent p(n)	Sign	K_n	g_n
1	2	L	75	1
2	3	U	65	1
3	15	L	105	1
4	5	L	30	1
5	6	U	20	1
6	10	L	42	1
7	8	L	75	1
8	9	U	69	1
9	10	U	91	1
10	12	L	28	1
11	12	U	99	1
12	13	U	36	1
13	15	U	38	1
14	15	U	85	1
15	16	U	61	1
16	17	L	275	1
17	—	L	75	1

Table 1. Graph and Cost Data for Tree Example.

set to unity, so the optimal reorder interval T(C) of any cluster C is simply the square root of the average setup cost for that cluster. Since reorder intervals are used only for comparing clusters, it is sufficient to report only squared reorder intervals, that is, to report the average setup costs of the clusters.

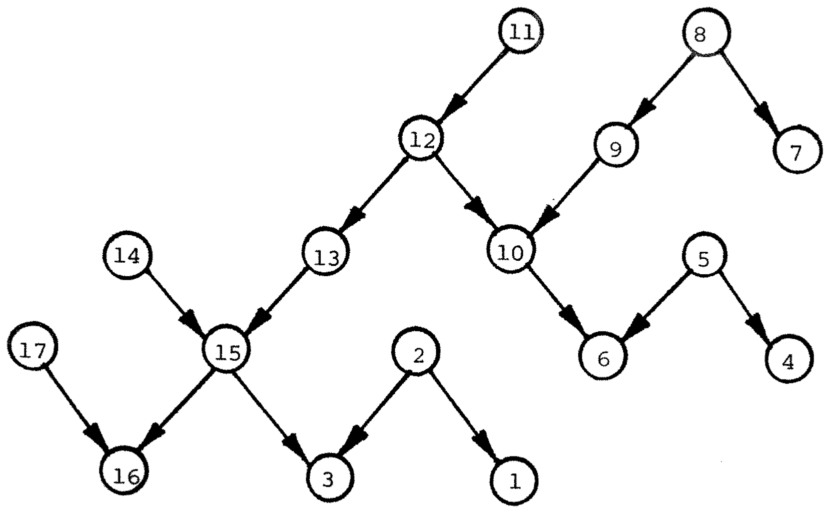


Figure 1. Graph Structure for Tree Example.

In describing the special structure of G and the solutions to P_G we make use of the following definitions. The <u>root of the tree</u> is an arbitrary element of N(G). If there are N nodes in the tree, we assume for convenience that the root is labelled N. Since G is a tree there is a unique simple path connecting any node n with the root N. For $n \neq N$ denote the immediate neighbor, or <u>parent</u>, of n on that path by p(n). Let $s(n) = \{i \in N(G); p(i) = n\}$, the set of <u>sons</u> of n. Let S(n) denote the set of all nodes i in N(G) such that the simple path from N to i includes node n. Thus $n \in S(n)$ and $s(n) \subseteq S(n)$. The elements of S(n) are called the <u>successors</u> of n. A node m is an <u>ancestor</u> of n if $n \in S(m)$. A node is both a successor and an ancestor of itself. In Figure 1, $p(10) = 12$, $s(10) = \{6,9\}$, $S(10) = \{4,5,6,7,8,9,10\}$ and 15 is an ancestor of 10 while 11 is not.

Nodes can be classified by the direction of the arc connecting them to their respective parents. We say a node n is upper, written $n \in U$, if $(n,p(n)) \in A(G)$. We say a node is lower, $n \in L$, if $(p(n),n) \in A(G)$. Arbitrarily we say that the root of the tree is lower, $N \in L$. The sign of a node n, written $\text{sign}(n)$, is its classification, either upper or lower.

Node i is said to be above (resp., below) node n if $i \in S(n)$, $i \neq n$, and the path from i to n, including i but excluding n, consists entirely of upper (resp., lower) nodes. Node i is said to be level with node n, written $i \sim n$, if $i \in S(n)$, $i \neq n$, and the path from i to n, inclusive, consists of both upper and lower nodes. In Figure 1, node 11 is above nodes 12, 13, 15, and 16, and it is level with nodes 16 and 17. Note that node 11 is both above and level with node 16.

For any cluster C, the root of the cluster is taken to be the node in C closest to N, with ties broken arbitrarily. If C is connected in G then the root of C is the node m uniquely satisfying $m \in C \subseteq S(m)$. A cluster is said to be upper (resp., lower) if its root is upper (resp., lower). For any cluster C, let $s(C) = \{j;\ p(j) \in C,\ j \notin C\}$. The elements of $s(C)$ are called cluster–sons of C. Observe $s(\{n\}) = s(n)$. In Figure 1, $s(\{6,9,10\}) = \{5,8\}$. If C is connected in G with root m then $s(C) \subseteq S(m)$ and

$$C = S(m) \setminus \left[\bigcup_{j \in s(C)} S(j) \right].$$

That is, a connected cluster is defined by its root and cluster–sons.

Let $A(n) = \{(m,j) \in A(G);\ m,j \in S(n)\}$, the arc set associated with $S(n)$. Let $G(n) = (S(n),A(n))$ and observe that $G(n)$ is an arbitrarily directed tree. Let $P_n \equiv P_{G(n)}$, the tree problem associated with the successor set of node n.

THEOREM 3.1: $\{T_j^* ;\ j \in S(n)\}$ is the optimal solution to P_n if and only if there exists an index set $R(n) \subseteq S(n)$ and a set of clusters $\{C_m;\ m \in R(n)\}$ satisfying

(a) $\{C_m; m \in R(n)\}$ is a partition of $S(n)$;

(b) C_m is connected in $G(n)$ and $m \in C_m \subseteq S(m)$ for all $m \in R(n)$;

(c) for each $m \in R(n)$ and each $j \in s(C_m)$,

$$T(C_j) \geq T(C_m) \text{ if } j \in U \text{, and}$$
$$T(C_j) \leq T(C_m) \text{ if } j \in L \text{;}$$

(d) for each $m \in R(n)$ and each $j \in C_m$, $j \neq m$,

$$T(C_m \cap S(j)) < T(C_m) \text{ if } j \in U \text{, and}$$
$$T(C_m \cap S(j)) > T(C_m) \text{ if } j \in L \text{;}$$

(e) if $j \in C_m$ then $T_j^* = T(C_m)$.

Proof: Appendix A.

A set of clusters satisfying (a), (c), and (d) of the theorem is said to be an optimal partition for P_n. By property (b), the clusters in an optimal partition are connected in G. Hence property (b) defines the index set to be the set of roots of clusters in the optimal partition. $R(n)$ is referred to as the optimal index set for P_n and its elements are called optimal nodes for P_n.

By Proposition 2.5, there is a unique optimal solution $\{T_j^*; j \in S(n)\}$ to P_n. We now strengthen Theorem 3.1 by proving the uniqueness of the optimal partition.

Proposition 3.2: If $\{T_j^*; j \in S(n)\}$ is the optimal solution to P_n and $\{C_m; m \in R(n)\}$ is an optimal partition, then $j \in R(n)$ if and only if

$$T(\{k \in S(j); T_k^* = T_j^*\}) = T_j^* \tag{3.1}$$

PROOF: Expressing the argument of $T(\cdot)$ in (3.1) in terms of the optimal partition,

$$\{k \in S(j);\ T_k^* = T_j^*\} = \bigcup_{\substack{k \in R(n) \\ T(C_k)=T_j^*}} (C_k \cap S(j))$$

$$= (C_m \cap S(j)) \cup \left[\bigcup_{\substack{k \in R(n) \cap S(j) \\ T(C_k)=T_j^* \\ k \neq m}} C_k \right]$$

where $j \in C_m$. Repeated application of Lemma 2.3 reveals that

$$T\left[\bigcup_{\substack{k \in R(n) \cap S(j) \\ T(C_k)=T_j^* \\ k \neq m}} C_k \right] = T_j^*.$$

By Assumption 2.2, (3.1) holds if and only if

$$T(C_m \cap S(j)) = T_j^* = T(C_m).$$

By Theorem 3.1(d) this can happen if and only if $j = m \in R(n)$. □

COROLLARY 3.3: There is a unique optimal partition to P_n.

PROOF: By Proposition 2.5, there is a unique optimal solution $\{T_j^*;\ j \in S(n)\}$ to P_n. By Proposition 3.2, this solution uniquely defines the optimal index set. Theorem 3.1(a) and (b) uniquely associate the optimal index set with an optimal partition. □

COROLLARY 3.4: Let $m \in R(n)$. Then $R(m) = R(n) \cap S(m)$ and the clusters containing node m in P_n and P_m coincide.

PROOF: It is easily seen that $R(n) \cap S(m)$ generates a partition satisfying (a)–(d) of Theorem 3.1. Uniqueness of the optimal partition establishes the result. □

Henceforth, C_n denotes the cluster rooted at n in the optimal partition for P_n. It is referred to as the <u>lead cluster</u> for problem P_n. As Corollary 3.4 indicates, the optimal partition for P_n is composed of the lead clusters for problems P_m, $m \in R(n)$.

COROLLARY 3.5: For any $n \in N(G)$,

$$R(n) = \{n\} \cup \left[\bigcup_{m \in s(C_n)} R(m) \right].$$

Corollary 3.5 suggests that problem P_N can be solved by recursively solving problems of the form P_n beginning with the leaves of the tree and working towards the root N. At each node n it is sufficient to identify the lead cluster C_n. The remainder of this section is directed toward illustrating a more explicit characterization of $R(n)$ and C_n.

For the example problem defined by Figure 1 and Table 1, Table 2 reports a complete analysis of P_N and all its subproblems P_n, $n \in N(G)$ by listing the optimal index set R_n, the composition of the lead cluster C_n, and the squared reorder interval of the lead cluster $T(C_n)^2$, for each node n. For example, the solution to P_{10} is characterized by four clusters, C_{10}, C_9, C_7, and C_4, whose squared reorder intervals are 30, 80, 75, and 30, respectively. The properties of optimal partitions as described in Theorem 3.1 and Corollary 3.5 can all be verified with respect to the example. The lead clusters of the example are displayed in Figure 2. In the figure, upper clusters are contained by solid lines and lower clusters are contained by dashed lines. The squared reorder interval of each cluster is listed below the root of the cluster.

A number of important new definitions and properties can be illustrated by the example. They will be taken up in greater detail in Section 5. A node $m \in S(n)$, $m \neq n$ is said to be <u>dominated in $S(n)$</u> if there exists a node $k \in S(n)$ such that $k \neq m$, $m \in C_k$, and k and m have the same sign; otherwise it is said to be <u>undominated in $S(n)$</u>. If m is dominated in $S(n)$ then m cannot be in $R(\ell)$ for any ancestor ℓ of n (see Proposition 5.5). In the example, node 12 is dominated in $S(15)$; node 11 is undominated for all of its ancestors. Let D_n denote the set of nodes dominated in $S(n)$:

$$D_n = \{j \in S(n) \mid \exists\, m \in S(n)\backslash\{j\},\ \text{sign}(j) = \text{sign}(m);\ \text{and}\ j \in C_m\},$$

and let M_n denote the set of nodes undominated in $S(n)$:

$$M_n = S(n)\backslash D_n.$$

A node $m \in S(n)$ is said to be <u>isolated in $S(n)$</u> if the path of nodes from n to m, inclusive, includes both upper and lower nodes (i.e., m is level with n), and if there does not exist any node $k \in S(n)$ such that $k \neq m$, $m \in C_k$, and k is undominated in $S(n)$. If m is isolated in $S(n)$ then $R(m) \subset R(\ell)$ for all ancestors ℓ of n (see Proposition 5.9). For example, node 9 is isolated in $S(10)$ and it appears in the optimal index set for all of the ancestors of node 10. Node 11 is isolated in $S(16)$ but not in $S(15)$, since the path from 15 to 11 consists entirely of upper nodes. Node 7 is an element of C_8, but since 8 is dominated in $S(9)$, node 7 is isolated in $S(9)$. Let I_n denote the set of nodes that are isolated in $S(n)$:

$$I_n = \left\{ \ell \in S(n) \mid \ell \sim n;\ \ell \notin \bigcup_{m \in M_n \backslash \{\ell\}} C_m \right\}.$$

Node n	Optimal Index Set $R(n)$	Lead Cluster C_n	Squared Reorder Interval $T(C_n)^2$
1	1	1	75
2	2	2,1	70
3	3,1	3,2	85
4	4	4	30
5	5	5,4	25
6	6,4	6,5	31
7	7	7	75
8	8	8	72
9	9,7	9,8	80
10	10,9,7,4	10,6,5	30
11	11	11	99
12	12,11,10,9,7,4	12	36
13	13,11,10,9,7,4	13,12	37
14	14	14	85
15	15,14,11,10,9,7,4,2	15,13,12,3	60
16	16,11,10,9,7,4,3,1	16,15,14,13,12	99
17	17,11,10,9,7,4,3,1	17,16,15,14,13,12	95

Table 2. Solutions to P_n, for each node $n \in N(G)$.

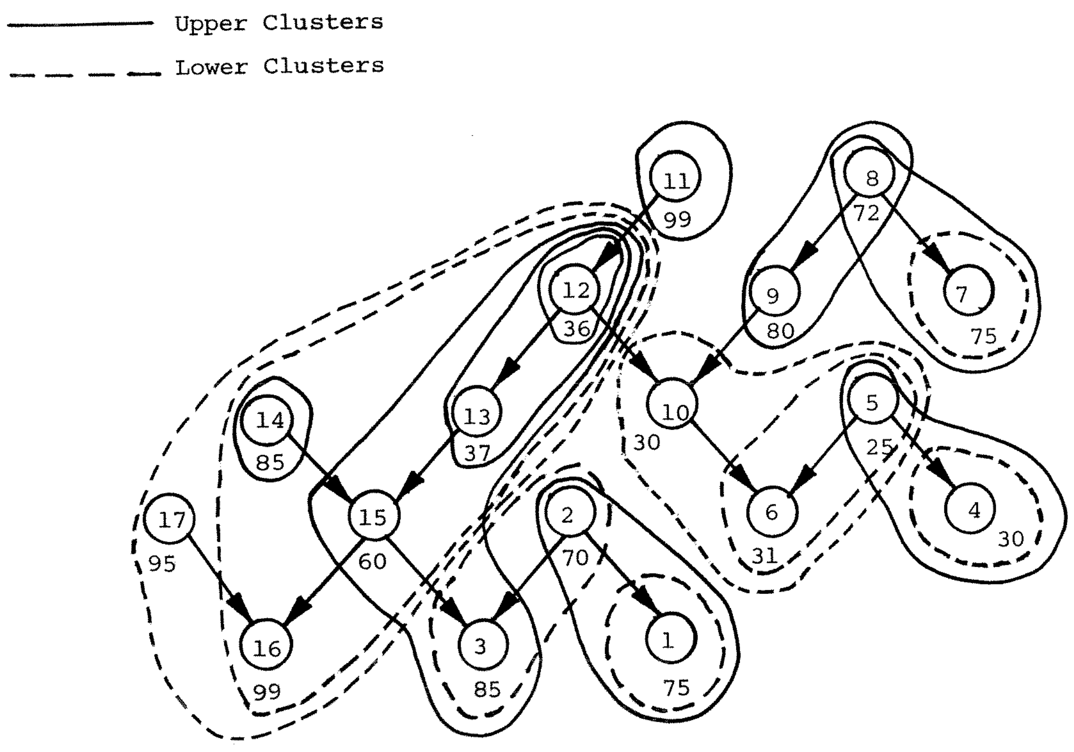


Figure 2. Lead Clusters and Squared Reorder Intervals.

A node $m \in S(n)$ is said to be <u>final in $S(n)$</u> if node m belongs to the optimal index set of any node that is isolated in $S(n)$. Hence isolated nodes are final nodes. Nodes 4 , 7 , 9 , and 10 are final in $S(12)$. Nodes 7 , 9 , and 10 are isolated (and final) in $S(12)$. Node 4 is final by virtue of being an element of $R(10)$. Let F_n denote the set of nodes final in $S(n)$:

$$F_n = \bigcup_{\ell \in I_n} R(\ell).$$

A node $m \in S(n)$ is said to be <u>defeated in $S(n)$</u> if m has an ancestor k that is isolated in $S(n)$ but node m is not in the optimal index set for P_k. If m is defeated in $S(n)$ then k cannot be in $R(\ell)$ for any ancestor ℓ of n (see Corollary 5.11). Node 5 is defeated in $S(12)$ even though it is undominated in $S(12)$. Let E_n denote the set of nodes defeated in $S(n)$:

$$E_n = \left[\bigcup_{\ell \in I_n} S(\ell) \right] \setminus F_n.$$

A node $m \in S(n)$ is said to be a <u>candidate</u> in $S(n)$ if it is not dominated, not final, and not defeated in $S(n)$. Node n is always a candidate in $S(n)$. Node 1 of the example is a candidate in the successor sets of all of its ancestors. Let Q_n denote the set of candidate nodes in $S(n)$:

$$Q_n = M_n \setminus (E_n \cup F_n).$$

Table 3 lists the sets of candidate and final nodes in the successor set of each node in the example. The relationships between C_n, $R(n)$, Q_n, and F_n are taken up in greater detail in Section 5. The following results are summarized from that study.

$S(n)$ can be partitioned into three disjoint sets: F_n, Q_n, and the set of all nodes that are either dominated or defeated in $S(n)$. If a node k is dominated (resp., isolated, final, defeated) in $S(n)$ then it is dominated (resp., isolated, final, defeated) in $S(\ell)$ for every ancestor ℓ of n. Consequently, if $k \in Q_n$ then $k \in Q_m$ for all m that are both ancestors of k and descendants of n. For example, since $7 \in F_9$, node 7 is in the optimal index set $R(n)$ for all ancestors of 9, $n = 9, 10, 12, 13, 15, 16, 17$. Node 8 is dominated in $S(9)$, and it is both defeated and dominated in $S(n)$ for $n \in \{10, 11, 12, 13, 15, 16, 17\}$. Node 11 is in Q_n for $n = 11, 12, 13, 15$.

The relationship between optimal index sets and candidate sets is given by (see Proposition 5.13):

$$R(n) = \begin{cases} (Q_n \cap U) \cup F_n & \text{if } n \in U, \\ (Q_n \cap L) \cup F_n & \text{if } n \in L. \end{cases} \tag{3.2}$$

Node n	Optimal Index Set $R(n)$	Candidate Nodes Q_n	Final Nodes F_n
1	1	1	—
2	2	2,1	—
3	3,1	3,2,1	—
4	4	4	—
5	5	5,4	—
6	6,4	6,5,4	—
7	7	7	—
8	8	8,7	—
9	9,7	9	7
10	10,9,7,4	10,5,4	9,7
11	11	11	—
12	12,11,10,9,7,4	12,11	10,9,7,4
13	13,11,10,9,7,4	13,11	10,9,7,4
14	14	14	—
15	15,14,11,10,9,7,4,2	15,14,11,3,2,1	10,9,7,4
16	16,11,10,9,7,4,3,1	16,15,14,3,2,1	11,10,9,7,4
17	17,11,10,9,7,4,3,1	17,15,14,3,2,1	11,10,9,7,4

Table 3. Candidate and Final Nodes for Tree Example.

For example, node 15 is upper, so for $n = 15$ the right hand side of (3.2) is $\{15, 14, 11, 2\} \cup \{10, 9, 7, 4\}$, which is the optimal index set for node 15.

Let $Q^n = \bigcup_{m \in s(n)} Q_m$ and $F^n = \bigcup_{m \in s(n)} F_m$. Then $n \in Q_n \subseteq \{n\} \cup Q^n$, $F^n \subseteq F_n$, and $F_n \subseteq Q^n \cup F^n$. Let $A_n = \{n\} \cup \{i \in S(n);\ i \text{ is above } n\}$ and let $B_n = \{n\} \cup \{i \in S(n);\ i \text{ is below } n\}$. The lead cluster C_n for problem P_n is related to Q^n by the following two equivalent statements (see Corollary 5.22 and Proposition 5.19):

$$C_n = \left[A_n \cup \left[\bigcup_{\substack{m \in Q^n \cap L \\ T(C_m) > T(C_n)}} C_m \right] \right] \setminus \left[\bigcup_{\substack{k \in Q^n \cap U \\ T(C_k) \geq T(C_n)}} C_k \right]; \tag{3.3}$$

and

$$C_n = \left[B_n \cup \left[\bigcup_{\substack{m \in Q^n \cap U \\ T(C_m) < T(C_n)}} C_m \right] \right] \setminus \left[\bigcup_{\substack{k \in Q^n \cap L \\ T(C_k) \leq T(C_n)}} C_k \right]. \tag{3.4}$$

For example, consider node 12. In this case $A_{12} = \{12, 11\}$, $B_{12} = \{12, 10, 6\}$, $Q^{12} = \{11, 10, 5, 4\}$, $Q^{12} \cap U = \{11, 5\}$, and $Q^{12} \cap L = \{10, 4\}$. Noting that $T(C_{11}) > T(C_{12}) > T(C_{10}) = T(C_4) > T(C_5)$, (see Figure 2), the right hand side of (3.4) becomes

$$[B_{12} \cup C_5] \setminus [C_{10} \cup C_4] = [\{12,10,6\} \cup \{5,4\}] \setminus [\{10,6,5\} \cup \{4\}] = \{12\} = C_{12}.$$

The right–hand sides on equations (3.3) and (3.4) are functions of $T(C_n)$, so $T(C_n)$ is a solution to a fixed point problem. It is shown in Section 5 that Assumption 2.2 guarantees that the solution is unique and can be found in a simple one–pass procedure (see Algorithm 4.1). The procedure can be initiated with A_n and used to solve (3.3), or initiated with B_n and used to solve (3.4). If $n \in U$ and (3.4) is solved, or if $n \in L$ and (3.3) is solved, then C_n, Q_n and F_n can be computed in the process of solving the fixed point problem.

4. THE TREE ALGORITHM

In this section we present an algorithm for solving P_G where G is an arbitrarily directed tree. We continue to use the notation of the previous section and we assume that the nodes of G have been indexed by depth–first search. The two key properties of such an indexing scheme are listed below (Aho et al. 1974, pp. 176–179).

PROPERTY 1: $p(n) > n$ for all n.

PROPERTY 2: There exist numbers $\sigma(n)$ such that $S(n) = (m;\ \sigma(n) \leq m \leq n)$.

For any index set $M \subseteq N(G)$ let C_M denote the set of lead clusters indexed by M: $C_M = \{C_n;\ n \in M\}$. Using the depth–first search indexing scheme we define a complete order on C_M as follows. For $m,n \in M$, $C_m < C_n$ if either (a) $T(C_m) < T(C_n)$, or (b) $T(C_m) = T(C_n)$ and any of the following conditions hold: $m \in L$ and $n \in U$; $m,n \in L$ and $m < n$; or $m,n \in U$ and $m > n$. We will make use of two order–related operations:

1. MIN (M): Returns the element k of M that satisfies $C_k \leq C_j$ for all $j \in M$.
2. MAX (M): Returns the element k of M that satisfies $C_k \geq C_j$ for all $j \in M$.

We now present the basic iteration step of the Tree Algorithm.

ALGORITHM 4.1: Find the lead cluster C_n at node n of G.

<u>Input</u>: n, $A = A_n$, $B = B_n$, $Q = Q^n$, $F = F^n$, C_{QUF}.

<u>Output</u>: $Q = Q_n$, $F = F_n$, C_{QUF}.

<u>Method</u>: If $n \in U$ then execute procedure SOLVE_UPPER(n,B,Q,F,C_{QUF}) given in Figure 3; else execute procedure SOLVE_LOWER(n,A,Q,F,C_{QUF}) given in the same figure.

The output of Algorithm 4.1 consists of the index sets Q_n and F_n and the pool of clusters indexed by $Q_n \cup F_n$. By (3.2), this pool contains the optimal clusters for P_n, including cluster C_n.

The validity of Algorithm 4.1 is established in Section 5. Figures 4 to 6 illustrate three applications of the algorithm to the example developed in the preceding section. Figure 4, summarizing the steps for node 12 , is the most interesting because it illustrates (a) the formation of C_{12} by solving (3.4); (b) the identification of an isolated node (node 10); (c) the identification of a node as final even though it is not isolated (node 4); and (d) the elimination from Q of a defeated candidate even though it is undominated (node 5).

Figure 5, summarizing the steps of the algorithm for node 15 , is interesting because it illustrates the elimination from Q of a dominated node (node 13). Figure 6 is included for completeness, illustrating the formation of a lead cluster (C_{16}) by solving (3.3).

Finally, we present the Tree Algorithm.

ALGORITHM 4.2: Solve P_N. (The Tree Algorithm).

Input: An arbitrarily directed tree G indexed by depth–first search and a procedure to compute T(C) for $C \subseteq N(G)$.

Output: $A = A_N$, $B = B_N$, $Q = Q_N$, $F = F_N$, $C_{Q \cup F}$. The optimal partition for problem P_N is given by $C_{(Q \cap L) \cup F}$.

Method: Execute procedure TREE_SOLVE(N,A,B,Q,F,$C_{Q \cup F}$) given in Figure 7.

```
     procedure    SOLVE_UPPER($n,B,Q,F,C_{QUF}$)
                  (resp., SOLVE_LOWER($n,A,Q,F,C_{QUF}$))
     begin
1.       $C \leftarrow B$  (resp., $C \leftarrow A$)
2.       FOUND $\leftarrow$ FALSE
3.       While $Q \neq \phi$ and FOUND = FALSE do begin
4.                Compute $T(C)$
5.                $k \leftarrow MIN(Q)$  (resp., $k \leftarrow MAX(Q)$)
6.                If $k \in L$ and $T(C) \geq T(C_k)$
                  (resp., if $k \in U$ and $T(C) \leq T(C_k)$)
                  then  "PRUNE":
                        begin
6.1                           $Q \leftarrow Q \backslash \{k\}$
6.2                           $F \leftarrow F \cup \{k\}$
6.3                           $C \leftarrow C \backslash C_k$
                        end
7.                else  if $k \in U$ and $T(C) > T(C_k)$
                        (resp., if $k \in L$ and $T(C) < T(C_k)$)
                        then  "GRAFT":
                              begin
7.1                                $Q \leftarrow Q \backslash \{k\}$
7.2                                $C_{QUF} \leftarrow C_{QUF} \backslash \{C_k\}$
7.3                                $C \leftarrow C \cup C_k$
                              end
8.                      else  "STOP": FOUND $\leftarrow$ TRUE
         end
9.       $Q \leftarrow Q \cup \{n\}$
10.      $C_n \leftarrow C$
11.      $C_{QUF} \leftarrow C_{QUF} \cup \{C_n\}$
     end
```

Figure 3. Procedure to Find the Lead Cluster at Node n.

Input: $n = 12$, $A_{12} = \{12,11\}$, $B_{12} = \{12,10,6\}$, $F^{12} = \{9,7\}$ and $Q^{12} = \{11,10,5,4\}$.

Method: SOLVE_UPPER

Q (sorted)	F	C	$T(C)^2$	k = MIN(Q)	Sign of k	$T(C_k)^2$	Action
5,4,10,11	9,7	12,10,6	35.3	5	U	25	GRAFT
4,10,11	9,7	12,10,6,5,4	31.2	4	L	30	PRUNE
10,11	9,7,4	12,10,6,5	31.5	10	L	30	PRUNE
11	10,9,7,4	12	36.0	11	U	99	STOP

Output: $C_{12} = \{12\}$, $F_{12} = \{10,9,7,4\}$ and $Q_{12} = \{12,11\}$.

Figure 4. Finding the Lead Cluster at Node 12.

Input: $n = 15$, $A_{15} = \{15,14,13,12,11\}$, $B_{15} = \{15,3\}$, $F^{15} = \{10,9,7,4\}$ and $Q^{15} = \{14,13,11,3,2,1\}$.

Method: SOLVE_UPPER

Q (sorted)	F	C	$T(C)^2$	k = MIN(Q)	Sign of k	$T(C_k)^2$	Action
13,2,1,3,14,11	10,9,7,4	15,3	83.0	13	U	37	GRAFT
2,1,3,14,11	10,9,7,4	15,13,12,3	60.0	2	L	70	STOP

Output: $C_{15} = \{15,13,12,3\}$, $F_{15} = \{10,9,7,4\}$, and $Q_{15} = \{15,14,11,3,2,1\}$.

Figure 5. Finding the Lead Cluster at Node 15.

Input: $n = 16$, $A_{16} = \{16,15,14,13,12,11\}$, $B_{16} = \{16\}$, $F^{16} = \{10,9,7,4\}$ and $Q^{16} = \{15,14,11,3,2,1\}$.

Method: SOLVE_LOWER

Q (sorted)	F	C	$T(C)^2$	k = MAX(Q)	Sign of k	$T(C_k)^2$	Action
15,2,1,3,14,11	10,9,7,4	16,15,14,13,12,11	99.0	11	U	99	PRUNE
15,2,1,3,14	11,10,9,7,4	16,15,14,13,12	95.0	14	U	85	STOP

Output: $C_{16} = \{16,15,14,13,12\}$, $F_{16} = \{11,10,9,7,4\}$ and $Q_{16} = \{16,15,14,3,2,1\}$.

Figure 6. Finding the Lead Cluster at Node 16.

```
     procedure          TREE_SOLVE($n,A,B,Q,F,C_{QUF}$)
     begin
1.       $A \leftarrow \{n\}$, $B \leftarrow \{n\}$, $Q \leftarrow \phi$, $F \leftarrow \phi$, $C_{QUF} \leftarrow \phi$
2.       For each $m \in s(n)$ in decreasing order of $m$ do
         begin
3.           TREE_SOLVE($m,A',B',Q',F',C_{Q'UF'}$)
4.           If $m \in U$   then $A \leftarrow A \cup A'$
5.                          else $B \leftarrow B \cup B'$
6.           $Q \leftarrow Q \cup Q'$
7.           $F \leftarrow F \cup F'$
7.           $F \leftarrow F \cup F'$
8.           $C_{QUF} \leftarrow C_{QUF} \cup C_{Q' \cup F'}$
         end
9.       If $n \in U$ then SOLVE_UPPER($n,B,Q,F,C_{QUF}$)
                  else SOLVE_LOWER($n,A,Q,F,C_{QUF}$)
         end
```

Figure 7. Recursive Procedure to Solve P_n.

The validity of the Tree Algorithm follows inductively from the validity of Algorithm 4.1, the definitions of A_n, B_n, F^n, and Q^n, and the recursive definition of the procedure TREE_SOLVE.

In Section 6 we argue that none of the lines in Figures 3 and 7 is performed more than $2N$ times, and with the exception of Line 4 in Figure 3, each line in Figures 3 and 7 can be performed in at most $O(\log N)$ time. Consequently the running time of the TREE Algorithm is at most $O(N \log N)$, exclusive of at most $2N$ operations of the form (1.3). In the case of the production planning problem PP_G, we show that the algorithm can be implemented to run in at most $O(N \log N)$ time overall. By extension, certain d–Schur convex objectives including isotonic regression can also be minimized on G in $O(N \log N)$ time.

5. VALIDITY OF THE TREE ALGORITHM

In this section we establish the validity of Algorithm 4.1 and, hence, the validity of Algorithm 4.2, the Tree Algorithm. The section is organized into three parts. Section 5.1 develops a detailed characterization of optimal index sets, Section 5.2 develops a characterization of lead clusters, and Section 5.3 relates these characterizations to Algorithms 4.1 and 4.2.

5.1 Characterization of Optimal Index Sets

In this subsection we define candidate and final nodes, establish the characterization (3.2) of optimal index sets in terms of candidate and final sets, and characterize candidate and final sets in recursive terms that involve the lead clusters.

Let $T^n = \{T^n_j; j \in S(n)\}$ denote the optimal solution to problem P_n and let $R(n)$ denote the corresponding optimal index set. Table 4 lists the squared reorder intervals $(T^n_j)^2$ for three nodes j and for several successive planning problems P_n, taken from the example of Section 3.

The following three lemmas establish several simple relations between the solutions to successive problems. They can be illustrated with reference to Table 4. Note that $T^j_j = T(C_j)$ by Theorem 3.1(e).

LEMMA 5.1: If $j \in S(n)$ then $T^n_j = T^j_j$ if and only if $j \in R(n)$.

PROOF: Corollary 3.4 implies that if $j \in R(n)$ then $T^n_j = T^j_j$. Let $T^n_j = T^j_j$. Then the constraint represented by the arc between j and $p(j)$ is not binding in the solution to P_n, so $T^{nj} = \{T^n_k;\ k \in S(j)\}$ must be optimal for problem P_j. By Proposition 2.5, $T^{nj} = T^j$. Since $j \in R(j)$, by Proposition 3.2 we have

$$T\left[\{k \in S(j);\ T^n_k = T^n_j\}\right] = T\left[\{k \in S(j);\ T^j_k = T^j_j\}\right] = T^j_j = T^n_j.$$

By Proposition 3.2, $j \in R(n)$. □

$(T^n_j)^2$	j	1	2	3
n	Sign	L	U	L
1	L	75^*	X	X
2	U	70	70^*	X
3	L	75^*	85	85^*
15	U	75^*	70^*	60
16	L	75^*	85	85^*
17	L	75^*	85	85^*

X	$j \notin S(n)$
*	$j \in R(n)$

Table 4. Squared Reorder Intervals from Successive Problems.

LEMMA 5.2: Let $j \in S(n)$. Then $T^n_j \geq T^j_j$ if $j \in U$, and $T^n_j \leq T^j_j$ if $j \in L$.

PROOF: If $T^n_j < T^j_j$ and $j \in U$ then we can construct an alternative feasible solution to P_n using $T^{n\backslash j} = \{T^n_k;\, k \in S(n)\backslash S(j)\}$ together with T^j. This solution must also be optimal for P_n, thereby contradicting uniqueness (Proposition 2.5). Therefore $T^n_j \geq T^j_j$ if $j \in U$. The proof for $j \in L$ is similar. □

LEMMA 5.3: Let $m \in S(n)$ and $j \in S(m)$. Then $T^n_j \geq T^m_j$ if $m \in U$, and $T^n_j \leq T^m_j$ if $m \in L$.

PROOF: Suppose $m \in U$ but $T^n_j < T^m_j$. On the path from m to j let node k be the node closest to m for which $T^n_k < T^m_k$. By the previous lemma $k \neq m$. Hence $p(k) \in S(m)$ and $T^n_{p(k)} \geq T^m_{p(k)}$. Suppose $k \in U$. Then $T^m_k > T^n_k \geq T^n_{p(k)} \geq T^m_{p(k)}$. Since $T^m_k > T^m_{p(k)}$, the solution $T^{mk} = \{T^m_\ell;\, \ell \in S(k)\}$ must be optimal for P_k. Hence we can construct an alternative optimal solution to P_n using $T^{n\backslash k} = \{T^n_\ell;\, \ell \in S(n)\backslash S(k)\}$ and T^{mk}. This contradicts uniqueness. Similarly, suppose $k \in L$. Then $T^n_k < T^m_k \leq T^m_{p(k)} \leq T^n_{p(k)}$. Since $T^n_k < T^n_{p(k)}$ the solution $T^{nk} = \{T^n_\ell;\, \ell \in S(k)\}$ must be optimal for P_k. Hence we can construct an alternative optimal solution to P_m using $T^{m\backslash k} = \{T^m_\ell;\, \ell \in S(m)\backslash S(k)\}$ and T^{nk}. This, too, contradicts uniqueness. Therefore $T^n_j \geq T^m_j$. The result for $m \in L$ follows similarly. □

As an example of Lemma 5.3, observe from Table 4 that $T^2_1 \leq T^{16}_1 \leq T^3_1$ since node $2 \in U$, $3 \in L$, and both 2 and 3 are successors of node 16.

An immediate consequence of these lemmas is the following.

COROLLARY 5.4: If $C_m \cap C_n \neq \phi$, $m \in L$, and $n \in U$ then $T(C_m) > T(C_n)$.

PROOF: Since C_m and C_n are each connected in G (Theorem 3.1(b)), either $m \in C_n$ or $n \in C_m$. In either case, $T(C_m) = T(C_n)$ is impossible by Lemma 5.1 and $T(C_m) < T(C_n)$ is impossible by Lemma 5.2. □

As in Section 3, let D_n denote the set of nodes dominated in $S(n)$:

$$D_n = \{j \in S(n) / \exists\ m \in S(n)\backslash\{j\};\ \text{sign}(m) = \text{sign}(j);\ \text{and}\ j \in C_m\}, \tag{5.1}$$

Clearly, if $m \in S(n)$ then $D_m \subseteq D_n$. That is, if j is dominated in $S(m)$ then it is dominated in the successor set for all ancestors of m. The next result establishes that if a node j is dominated in $S(n)$ then it does not belong to the optimal index set of n.

PROPOSITION 5.5: If $j \in D_n$, then $j \notin R(n)$.

PROOF: Since $j \in D_n$, there exists a node m with $m \in S(n)\backslash\{j\}$, $\text{sign}(j) = \text{sign}(m)$, and $j \in C_m$. Suppose $j,m \in U$. By Lemmas 5.2 and 5.3 we have $T_j^n \geq T_j^m \geq T_j^j$. If $T_j^m = T_j^j$ then by Lemma 5.1 we would have $j \in R(m)$. However $j \in C_m$ implies $j \notin R(m)$. Consequently $T_j^n \geq T_j^m > T_j^j$. If $j \in R(n)$ then $T_j^n = T_j^j$ by Lemma 5.1. Hence $j \notin R(n)$. The proof for $j,m \in L$ is similar. □

As in Section 3, let M_n denote the set of nodes undominated in $S(n)$:

$$M_n = S(n)\backslash D_n. \tag{5.2}$$

Clearly, since lead clusters are connected in G, if two nodes k and m are undominated in $S(n)$ and have the same sign then C_k and C_m are disjoint.

COROLLARY 5.6: $R(n) \subseteq M_n$.

A node $\ell \in S(n)$ is said to be level with node n, written $\ell \sim n$, if the path from ℓ to n, inclusive, includes both upper and lower nodes. A node cannot be level with itself. A node ℓ is isolated in $S(n)$ if ℓ is level with n and there does not exist another node $m \in M_n$ such that $\ell \in C_m$. As in Section 3, let I_n denote the set of nodes that are isolated in $S(n)$:

$$I_n = \left\{ \ell \in S(n);\, \ell \sim n\, ,\, \ell \notin \bigcup_{m \in M_n \setminus \{\ell\}} C_m \right\}. \tag{5.3}$$

PROPOSITION 5.7: $I_n \subset R(n)$.

PROOF: By Corollary 5.6, $R(n) \subseteq M_n$. Now $\{C_m;\, m \in R(n)\}$ forms a partition of $S(n)$ but the only cluster in M_n containing an isolated node, $\ell \in I_n$, is C_ℓ. Consequently $\ell \in R(n)\ \forall\ \ell \in I_n$. □

The following lemma and proposition establish that if a node ℓ is isolated in $S(m)$ then it is isolated, and hence is in the optimal index set, for all ancestors n of m.

LEMMA 5.8: If $m \in s(n)$, $\ell \in S(m)$, and $\ell \in I_n \setminus I_m$ then $\mathrm{sign}(n) \neq \mathrm{sign}(\ell)$.

PROOF: Suppose n and ℓ have the same sign. Since $\ell \sim n$, there exists a node on the path from n to ℓ with the opposite sign to both n and ℓ. It follows, therefore, that $\ell \sim m$ and hence $\ell \neq m$. Since $\ell \notin I_m$ there exists a node $k \in M_m$ such that $\ell \in C_k$. Now $\ell \in I_n$ implies $\ell \in R(n)$ by Proposition 5.7 and, hence, $\ell \in M_n$ by Corollary 5.6. This implies ℓ and k have opposite signs. It must also be the case that $k \notin M_n$; else $\ell \notin I_n$. Now, since

$n = p(m)$, $k \in M_m$ and $k \notin M_n$ can happen only if $k \in C_n$ and if k and n have the same sign, contradicting the assumption that n and ℓ have the same sign. □

PROPOSITION 5.9: If $m \in S(n)$ then $I_m \subseteq I_n \subset R(n)$.

PROOF: $I_n \subset R(n)$ by Proposition 5.7. By induction, it suffices to show that $I_m \subseteq I_n$ for $n = p(m)$. Suppose $\ell \in I_m \backslash I_n$. Now $\ell \sim m$ implies $\ell \sim n$, so by definition there exists a node $k \in M_n$ such that $\ell \in C_k$. By (5.1) $D_m \subseteq D_n$, so $M_n \cap S(m) \subseteq M_m$. We must have $k = n$; otherwise $k \in M_m$ and $\ell \notin I_m$. Therefore $\ell \in I_m \backslash I_n$ implies $\ell \in C_n$ and $\ell \notin R(n)$. We will produce a contradiction by showing that $\ell \in R(n)$.

Let j be the node closest to ℓ on the path from m to ℓ such that $\ell \in I_j$. By Lemma 5.8, ℓ and j have opposite signs. By Proposition 5.7, $\ell \in R(j)$, so we have $T^j_\ell = T^\ell_\ell$ by Lemma 5.1. Suppose $\ell \in U$ and $j \in L$. By Lemma 5.3, $T^n_\ell \leq T^j_\ell = T^\ell_\ell \leq T^n_\ell$. Therefore $T^\ell_\ell = T^n_\ell$ and, by Lemma 5.1, $\ell \in R(n)$. A similar argument holds for $\ell \in L$ and $j \in U$. □

A node is final in $S(n)$ if it belongs in the optimal index set of a node isolated in $S(n)$. As in Section 3, the set of final nodes in $S(n)$, F_n, is given by

$$F_n = \bigcup_{\ell \in I_n} R(\ell). \tag{5.4}$$

The following two corollaries follow immediately from Proposition 5.9 and Corollary 3.4.

COROLLARY 5.10: If $m \in S(n)$ then $F_m \subseteq F_n \subset R(n)$.

A node is defeated in $S(n)$ if it is not final in $S(n)$ and it is a successor of a node that is isolated in $S(n)$. As in Section 3, let E_n denote the set of nodes defeated in $S(n)$:

$$E_n = \left[\bigcup_{\ell \in I_n} S(\ell) \right] \backslash F_n.$$

As the following corollary indicates, defeated nodes cannot be in the optimal index set.

COROLLARY 5.11: $E_n \subseteq S(n) \backslash R(n)$.

A node in the successor set of an isolated node is either final or defeated. Hence the set Q_n of candidate nodes in $S(n)$ (that is, the set of nodes that are not dominated, not final, and not defeated in $S(n)$) is given by

$$Q_n = M_n \backslash (E_n \cup F_n). \qquad (5.5)$$

By (5.4) and Theorem 3.1(a),

$$\bigcup_{\ell \in F_n} C_\ell = \bigcup_{j \in I_n} S(j).$$

Furthermore, by (5.3) and Theorem 3.1(b), if $m \in Q_n$ and $j \in I_n$ then $C_m \cap S(j) = \emptyset$. Consequently,

$$\left[\bigcup_{m \in Q_n} C_m \right] \cap \left[\bigcup_{\ell \in F_n} C_\ell \right] = \phi. \qquad (5.6)$$

The following lemma and proposition establish the desired characterization of optimal index sets.

LEMMA 5.12: If $m \in S(n)$ and $\ell \in s(C_m)$ then either $\text{sign}(\ell) = \text{sign}(m)$ or $\ell \in I_n \subsetneq F_n$.

PROOF: By Proposition 5.9 it suffices to show that either $\mathrm{sign}(\ell) = \mathrm{sign}(m)$ or $\ell \in I_m$. By Corollaries 3.5 and 5.6, $\ell \in s(C_m)$ implies $\ell \in R(m)$ and hence $\ell \in M_m$. Suppose m and ℓ have opposite signs. In that case, either $\ell \in I_m$ or there exists a node $k \in M_m$ on the path from m to ℓ, $k \neq \ell$, such that $\ell \in C_k$. Now $\ell \in M_m$ implies that ℓ and k have opposites signs. However, $\ell \in s(C_m)$ implies that $k \in C_m$. Since $k \in M_m$, we must have $\mathrm{sign}(k) \neq \mathrm{sign}(m)$. Consequently ℓ and m must have the same sign, which is a contradiction. Hence if ℓ and m have opposite signs we must have $\ell \in I_m$. □

PROPOSITION 5.13: If $m \in F_n \cup Q_n$ then

$$R(m) = \begin{cases} F_n \cap S(m) , & \text{if } m \in F_n , \\ (F_n \cup (Q_n \cap U)) \cap S(m) , & \text{if } m \in Q_n \cap U , \\ (F_n \cup (Q_n \cap L)) \cap S(m) , & \text{if } m \in Q_n \cap L. \end{cases}$$

PROOF: The result follows from a straightforward induction argument based on Corollary 3.5 and Lemma 5.12. □

Observe that $n \in Q_n$ so that (3.2) follows immediately from the proposition.

Proposition 5.13 characterizes optimal index sets in terms of final nodes and candidate nodes. We turn now to the problem of recursively computing F_n and Q_n. Corollary 5.14 below is found to be the key relationship in solving this problem.

COROLLARY 5.14: If $n \in U$ then

$$T(C_m) \geq T(C_n) \;\; \forall \; m \in Q_n \cap U \quad \text{and} \quad T(C_m) > T(C_n) \;\; \forall \; m \in Q_n \cap L.$$

If $n \in L$ then

$$T(C_m) \leq T(C_n) \;\forall\; m \in Q_n \cap L \quad \text{and} \quad T(C_m) < T(C_n) \;\forall\; m \in Q_n \cap U.$$

PROOF: Suppose $n \in U$ and $m \in Q_n$. By (5.6), $m \notin \bigcup_{\ell \in F_n} C_\ell$. Suppose $m \in U$. By Proposition 5.13, $m \in R(n)$. Let k denote the root of the optimal cluster next to C_m on the path from n to m. That is, $k \in R(n)$ and $p(m) \in C_k$. Since $m \in U$, $T(C_m) \geq T(C_k)$. By Proposition 5.13 and by (5.6), $k \in Q_n \cap U$. By induction, therefore, $T(C_m) \geq T(C_n)$. Suppose $m \in L$. Then $m \in C_k$ for some $k \in Q_n \cap U$; otherwise m is isolated in $S(n)$. By Corollary 5.4, $T(C_m) > T(C_k)$. As just established, $T(C_k) \geq T(C_n)$ since $k \in Q_n \cap U$. Hence $T(C_m) > T(C_n)$. A parallel argument establishes the result for $n \in L$. □

As in Section 3, let $Q^n = \bigcup_{m \in s(n)} Q_m$ and let I^n and F^n be similarly defined. By Corollary 5.10, $F^n \subseteq F_n$. The set difference $F_n \setminus F^n$ is the set of nodes identified as final in $S(n)$ but not final in $S(m)$ for any son m of n.

$$F_n \setminus F^n = \left[\bigcup_{\ell \in I_n} R(\ell)\right] \setminus \left[\bigcup_{k \in I^n} R(k)\right]$$

$$= \bigcup_{\ell \in I_n \setminus I^n} \left[R(\ell) \setminus \left[\bigcup_{k \in I^n} R(k)\right]\right] \subseteq \bigcup_{\ell \in I_n \setminus I^n} \left[R(\ell) \setminus \left[\bigcup_{k \in I_\ell} R(k)\right]\right]$$

since $I_\ell \subseteq I^n$ for all $\ell \in S(n) \setminus \{n\}$. Therefore

$$F_n \setminus F^n \subseteq \bigcup_{\ell \in I_n \setminus I^n} (R(\ell) \setminus F_\ell). \tag{5.7}$$

LEMMA 5.15: All elements of $F_n \setminus F^n$ have the opposite sign of node n.

PROOF: By Lemma 5.8, $I_n \backslash I^n$ consists entirely of nodes which have the opposite sign of n. By Proposition 5.13, $R(\ell) \backslash F_\ell$ consists entirely of nodes with the same sign as ℓ. By (5.7), therefore, $F_n \backslash F^n$ consists of nodes with the same sign of nodes in $I_n \backslash I^n$ and the opposite sign of node n. □

The next two propositions reveal that final sets and candidate sets can be computed recursively, given the $T(\cdot)$ value of the current lead cluster.

PROPOSITION 5.16: For $n \in N(G)$,

$$F_n = \begin{cases} F^n \cup \{\ell \in Q^n \cap L\colon\ T(C_\ell) \le T(C_n)\} & \text{if } n \in U, \\ F^n \cup \{\ell \in Q^n \cap U\colon\ T(C_\ell) \ge T(C_n)\} & \text{if } n \in L. \end{cases}$$

PROOF: As observed, $F^n \subseteq F_n$. Let $\ell \in F_n \backslash F^n$. By Lemma 5.15, n and ℓ have opposite signs. Suppose $n \in U$; then $\ell \in L$ and, in particular, $\ell \in Q^n \cap L$. Let m be the ancestor of ℓ that is a cluster–son of C_n: $m \in s(C_n)$, $\ell \in S(m)$. Since $\ell \notin F^n$, $\ell \in Q_m \cap L$. Now $m \in s(C_n)$ implies $m \in R(n)$ and $\ell \in F_n \subset R(n)$ then implies $\ell \in R(m)$. Therefore, by Proposition 5.13, ℓ and m have the same sign; else $\ell \in F^m$. By Corollary 5.14, $T(C_\ell) \le T(C_m)$. By Theorem 3.1(c), $T(C_m) \le T(C_n)$. Hence $T(C_\ell) \le T(C_n)$. The result follows similarly if $n \in L$.

Now suppose $n \in U$, $\ell \in Q^n \cap L$, and $T(C_\ell) \le T(C_n)$. By Corollary 5.14, $\ell \notin Q_n \cap L$. Node ℓ cannot be dominated in $S(n)$ because $\ell \in Q^n$ and ℓ and n are of opposite sign. Suppose ℓ is defeated in $S(n)$. In that case, by (5.4) and Theorem 3.1(a), there exists a node $m \in F_n$ such that $\ell \in C_m$. It cannot be the case that $m \in F^n$ because $\ell \in Q^n$. Thus, $m \in F_n \backslash F^n$. By Lemma 5.15, $m \in L$, implying ℓ is dominated in $S(m)$, a contradiction. If ℓ is not dominated, not defeated, and not a candidate in $S(n)$ then it must be final in $S(n)$; $\ell \in F_n$. The result follows similarly for $n \in L$, $\ell \in Q^n \cap U$, and $T(C_\ell) \ge T(C_n)$. □

PROPOSITION 5.17: If $n \in U$ then

$$Q_n = \{n\} \cup \{k \in Q^n \cap L;\ T(C_k) > T(C_n)\} \cup \{k \in Q^n \cap U;\ T(C_k) \geq T(C_n)\}.$$

If $n \in L$ then

$$Q_n = \{n\} \cup \{k \in Q^n \cap U;\ T(C_k) < T(C_n)\} \cup \{k \in Q^n \cap L;\ T(C_k) \leq T(C_n)\}.$$

PROOF: It is easily seen that $Q_n \subseteq \{n\} \cup Q^n$. Suppose $n \in U$. If $k \in Q_n \backslash \{n\}$ then $T(C_k) > T(C_n)$ if $k \in L$ and $T(C_k) \geq T(C_n)$ if $k \in U$, by Corollary 5.14. Hence Q_n is a subset of the right hand side of the first equation.

Suppose $k \in Q^n \cap L$ and $T(C_k) > T(C_n)$. By Proposition 5.16, $k \notin F_n$. If k is defeated in $S(n)$ then $\exists\, \ell \in F_n \backslash F^n$ such that $k \in C_\ell$. By Lemma 5.15, $\ell \in L$, implying k is dominated in $S(\ell)$, a contradiction. If k is dominated in $S(n)$ then there exists $m \in \{n\} \cup Q^n$ such that $k \in C_m$ and m and k have the same sign. If $m \in Q^n$ then $k \notin Q^n$, a contradiction. Nodes k and n have opposite signs. Therefore k is undominated, undefeated, and not final in $S(n)$: $k \in Q_n$.

Suppose $k \in Q^n \cap U$ and $T(C_k) \geq T(C_n)$. By Proposition 5.16, $k \notin F_n$. If k is defeated in $S(n)$ then $\exists\, \ell \in F_n \backslash F^n$ such that $k \in C_\ell$. By Lemma 5.15, $\ell \in L$. By Corollary 5.14, $T(C_\ell) > T(C_k)$ since $k \in Q^n \cap S(\ell) \subseteq Q_\ell$. Hence $T(C_\ell) > T(C_n)$, violating Proposition 5.16. Consequently k is undefeated in $S(n)$. If k is dominated in $S(n)$ then $k \in C_n$ since $k \in Q^n$. But $T(C_k) \geq T(C_n)$ violates Lemma 5.1 if the equality holds and violates Lemma 5.2 if the strict inequality holds. Therefore k is undominated, undefeated, and not final in $S(n)$, i.e., $k \in Q_n$.

A parallel proof establishes the result for $n \in L$. □

5.2 Characterization of Lead Clusters

In this section we establish that C_n, the lead cluster of problem P_n, is the unique solution to a fixed point problem.

Let $C_n(T)$ be the point to set mapping defined by

$$C_n(T) = \{n\} \cup \left[\bigcup_{m \in Q^n} C_m \right] \Bigg\backslash \left[\left[\bigcup_{\substack{k \in Q^n \cap U \\ T(C_k) \geq T}} C_k \right] \cup \left[\bigcup_{\substack{k \in Q^n \cap L \\ T(C_k) \leq T}} C_k \right] \right]. \tag{5.8}$$

LEMMA 5.18: The clusters in $\{C_k$; either $k \in Q^n \cap U$ and $T(C_k) \geq T$, or $k \in Q^n \cap L$ and $T(C_k) \leq T\}$ are all mutually disjoint.

PROOF: Since candidate nodes are undominated, the clusters $\{C_k: k \in Q^n \cap U\}$ are mutually disjoint, as are the clusters $\{C_k: k \in Q^n \cap L\}$. Suppose $k \in Q^n \cap U$, $m \in Q^n \cap L$, $C_k \cap C_m \neq \phi$, and $T(C_k) \geq T \geq T(C_m)$. By Corollary 5.4, $T(C_m) > T(C_k)$, a contradiction. □

The fixed point problem is to find a value T^o such that $T(C_n(T^o)) = T^o$. The existence of a fixed point is ensured by the following.

PROPOSITION 5.19: If $T^o = T(C_n)$ then $C_n(T^o) = C_n$.

PROOF: Setting $T = T^\circ$ in (5.8) and by employing (5.6), we have

$$C_n(T^\circ) = \{n\} \cup \left[\bigcup_{\ell \in Q^n \cup F^n} C_\ell\right] \Bigg\backslash \left[\left[\bigcup_{k \in F^n} C_k\right] \cup \left[\bigcup_{\substack{k \in Q^n \cap U \\ T(C_k) \geq T^\circ}} C_k\right] \cup \left[\bigcup_{\substack{k \in Q^n \cap L \\ T(C_k) \leq T^\circ}} C_k\right]\right]$$

$$= S(n) \Bigg\backslash \left[\left[\bigcup_{k \in F^n} C_k\right] \cup \left[\bigcup_{\substack{k \in Q^n \cap U \\ T(C_k) \geq T^\circ}} C_k\right] \cup \left[\bigcup_{\substack{k \in Q^n \cap L \\ T(C_k) \leq T^\circ}} C_k\right]\right]$$

since $R(m) \subseteq Q^n \cup F^n$ for all $m \in s(n)$ by Proposition 5.13. Suppose $n \in U$. By Propositions 5.16 and 5.17, if $T^\circ = T(C_n)$ then

$$C_n(T^\circ) = S(n) \Bigg\backslash \left[\left[\bigcup_{k \in F_n} C_k\right] \cup \left[\bigcup_{k \in Q_n \cap U \backslash \{n\}} C_k\right]\right].$$

Hence, by Proposition 5.13,

$$C_n(T^\circ) = S(n) \backslash \left[\bigcup_{k \in R(n) \backslash \{n\}} C_k\right].$$

By Corollary 3.5 and Theorem 3.1(a),

$$C_n(T^\circ) = S(n) \Bigg\backslash \left[\bigcup_{\substack{k \in R_m \\ m \in s(C_n)}} C_k\right] = S(n) \Bigg\backslash \left[\bigcup_{m \in s(C_n)} S(m)\right] = C_n.$$

The proof for $n \in L$ is parallel. □

Proposition 5.19 implies that $T(C_n)$ is a fixed point of the function $T(C_n(T))$. The following proposition states that this is the only fixed point of $T(C_n(T))$. The proof uses the strict monotonicity property of Assumption 2.2.

PROPOSITION 5.20: If $T^o \in \bar{\mathbb{R}}$ satisfies $T^o = T(C_n(T^o))$ then T^o is the unique point satisfying $T = T(C_n(T))$.

PROOF: By (5.8), $T(C_n(T))$ is a piecewise–constant function of T with a finite number of discontinuities. For arbitrary T^*, let $C = C_n(T^*)$, $C^- = C_n(T^*-\epsilon)$, and $C^+ = C_n(T^*+\epsilon)$ where ϵ is a very small positive number. By (5.8), $C^- = C \cup C_-$ for some (possibly empty) set C_-, and $C^+ = C \cup C_+$ for some (possibly empty) set C_+. Corollary 5.4 implies that

$$C_- = \bigcup_{\substack{k \in Q^n \cap L \\ T(C_k)=T^*}} C_k \text{ and } C_+ = \bigcup_{\substack{k \in Q^n \cap U \\ T(C_k)=T^*}} C_k .$$

By Lemma 2.3, $T(C_-) = T^*$ if $C_- \neq \emptyset$, and $T(C_+) = T^*$ if $C_+ \neq \emptyset$. Assumption 2.2 implies that

$$\text{if } T^* > T(C) \text{ then } T^* > T(C^-) \text{ and } T^* > T(C^+) ,$$

$$\text{if } T^* < T(C) \text{ then } T^* < T(C^-) \text{ and } T^* < T(C^+) \text{, and}$$

$$\text{if } T^* = T(C) \text{ then } T^* = T(C^-) \text{ and } T^* = T(C^+) .$$

In the latter case T^* is not a discontinuity of $T(C_n(T))$.

Let $T^o = T(C_n) = T(C_n(T^o))$. Then $T(C_n(T))$ is continuous at T^o. Since $T(C_n(T))$ is piecewise constant with a finite number of discontinuities,

$T(C_n(T^\circ+\epsilon)) = T^\circ < T^\circ + \epsilon$ for all sufficiently small $\epsilon > 0$. By induction on the discontinuities of $T(C_n(T))$, the preceding paragraph implies that $T(C_n(T)) < T$ for all $T > T^\circ$. Similarly, $T < T(C_n(T))$ for all $T < T^\circ$. □

We are now in a position to establish (3.3) and (3.4) as alternative equivalent statements of the fixed point problem. As defined in Section 3, $A_n = \{n\} \cup \{k \in S(n);\ k \text{ is above } n\}$ and $B_n = \{n\} \cup \{k \in S(n);\ k \text{ is below } n\}$.

Lemma 5.21:

$$A_n \cup \left[\bigcup_{k\in Q^n\cap L} C_k\right] = \{n\} \cup \left[\bigcup_{m\in Q^n} C_m\right] = B_n \cup \left[\bigcup_{k\in Q^n\cap U} C_k\right].$$

Proof: By (5.3), (5.4), and Proposition 5.13 we have $A_n\backslash\{n\} \subseteq \bigcup_{m\in Q^n} C_m$. Let $\ell \in \bigcup_{m\in Q^n} C_m$ and $\ell \in s(A_n)$. Clearly, $\ell \in Q^n \cap L$. By Proposition 5.13 and (5.6),

$$\bigcup_{k\in Q^n\cap S(\ell)} C_k = \bigcup_{k\in Q^n\cap L\cap S(\ell)} C_k.$$

Since $A_n \cap Q^n \cap L = \phi$ we have

$$\{n\} \cup \left[\bigcup_{m\in Q^n} C_m\right] = A_n \cup \left[\bigcup_{\ell\in s(A_n)} \left[\bigcup_{k\in Q^n\cap L\cap S(\ell)} C_k\right]\right] = A_n \cup \left[\bigcup_{k\in Q^n\cap L} C_k\right].$$

The result involving B_n follows similarly. □

Corollary 5.22: If $T^\circ = T(C_n)$ then $C_n(T^\circ)$ is given equivalently by the right hand sides of (3.3) and (3.4).

PROOF: The result follows upon substituting $T(C_n)$ for T in (5.8), employing the substitutions suggested by Lemma 5.21, and noting that $A_n \cap Q^n \cap L = B_n \cap Q^n \cap U = \phi$ and Lemma 5.18 permit reordering the set difference and set union operations. □

5.3 Validity of the Tree Algorithm

In this subsection we relate the characterizations of the previous two subsections to Algorithms 4.1 and 4.2.

Let $q_n = |Q^n|$ and let $Q^n = \{k_n^1, k_n^2, \dots, k_n^{q_n}\}$ where $C_{k_n^i} < C_{k_n^{i+1}}$ for all i, using the ordering defined in Section 4. Let $r_n^i = \{k_n^j : 1 \leq j < i\}$ and $s_n^i = \{k_n^j : i \leq j \leq q_n\}$ for $i = 1, 2, \dots, q_{n+1}$. Note that $r_n^1 = s_n^{q_n+1} = \phi$. Let

$$C_n^i = \{n\} \cup \left[\bigcup_{\ell \in Q^n} C_\ell\right] \setminus \left[\left[\bigcup_{k \in r_n^i \cap L} C_k\right] \cup \left[\bigcup_{k \in s_n^i \cap U} C_k\right]\right]. \tag{5.9}$$

LEMMA 5.23: For each $i = 1, 2, \dots, q_n+1$, the clusters $\{C_k : k \in (r_n^i \cap L) \cup (s_n^i \cap U)\}$ are mutually disjoint.

PROOF: The proof is parallel to that of Lemma 5.18. □

The following proposition indicates that the clusters C_n^i can be computed recursively for both increasing and decreasing values of i.

PROPOSITION 5.24: $C_n^1 = B_n$ and $C_n^{q_n+1} = A_n$. For $i = 1, 2, \dots, q_n$, if $k_n^i \in U$ then $C_n^i = C_n^{i+1} \setminus C_{k_n^i}$ and $C_n^{i+1} = C_n^i \cup C_{k_n^i}$, and if $k_n^i \in L$ then $C_n^i = C_n^{i+1} \cup C_{k_n^i}$ and $C_n^{i+1} = C_n^i \setminus C_{k_n^i}$.

PROOF: Since $r_n^1 = \phi$ and $s_n^1 = Q^n$, Lemma 5.21 and (5.9) imply

$$C_n^1 = B_n \cup \left[\bigcup_{k \in Q^n \cap U} C_k\right] \setminus \left[\bigcup_{k \in Q^n \cap U} C_k\right].$$

Clearly $B_n \cap C_k = \phi \ \forall \ k \in Q^n \cap U$. Hence $C_n^1 = B_n$. Similarly, $C_n^{q_n+1} = A_n$.

Lemma 5.23 implies $C_{k_n^i}$ is disjoint from $\{C_k;\ k \in (r_n^i \cap L) \cup (s_n^{i+1} \cap U)\}$. Therefore if $k_n^i \in U$ then $C_{k_n^i} \subseteq C_n^{i+1}$, and if $k_n^i \in L$ then $C_{k_n^i} \subseteq C_n^i$. The result now follows by (5.9). □

We next establish conditions under which the constructs of this subsection, r_n^i, s_n^i, and C_n^i, match up with the desired sets, Q_n, F_n, and C_n.

Let $t_n^i = T\left[C_{k_n^i}\right]$ for $i = 1, \dots, q_n$, $t_n^0 = -\infty$, and $t_n^{q_n+1} = +\infty$. We say that i is <u>critical in</u> n if the following conditions hold:

$$\begin{cases} t_n^{i-1} < T(C_n^i), & \text{if } i > 1 \text{ and } k_n^{i-1} \in U; \\ t_n^{i-1} \leq T(C_n^i), & \text{if } i > 1 \text{ and } k_n^{i-1} \in L; \\ T(C_n^i) \leq t_n^i, & \text{if } i \leq q_n \text{ and } k_n^i \in U; \text{ and} \\ T(C_n^i) < t_n^i, & \text{if } i \leq q_n \text{ and } k_n^i \in L. \end{cases} \tag{5.10}$$

PROPOSITION 5.25: If i is critical in n then

(a) $r_n^i \cap U = \{\ell \in Q^n \cap U: T(C_\ell) < T(C_n)\}$,

(b) $r_n^i \cap L = \{\ell \in Q^n \cap L: T(C_\ell) \leq T(C_n)\}$,

(c) $s_n^i \cap L = \{\ell \in Q^n \cap L: T(C_\ell) > T(C_n)\}$,

(d) $s_n^i \cap U = \{\ell \in Q^n \cap U: T(C_\ell) \geq T(C_n)\}$,

(e) $C_n^i = C_n$,

(f) $$F_n = \begin{cases} F^n \cup (r_n^i \cap L) & \text{if } n \in U, \\ F^n \cup (s_n^i \cap U) & \text{if } n \in L, \end{cases}$$

(g) $$Q_n = \begin{cases} \{n\} \cup s_n^i & \text{if } n \in U, \\ \{n\} \cup r_n^i & \text{if } n \in L. \end{cases}$$

PROOF: If i is critical in n then it is easily seen that

$$r_n^i \cap L \subseteq \{\ell \in Q^n \cap L\colon T(C_\ell) \le T(C_n^i)\}.$$

Suppose that there is an $\ell \in s_n^i \cap L$ such that $T(C_\ell) \le T(C_n^i)$. Since $T(C_n^i) \le t_n^i$ we must have $T(C_\ell) = T(C_n^i) = t_n^i$. By the ordering rule, $\ell \in L$ implies $k_n^i \in L$ and hence $T(C_n^i) < t_n^i$, a contradiction. Therefore

$$r_n^i \cap L = \{\ell \in Q^n \cap L\colon T(C_\ell) \le T(C_n^i)\}.$$

Similarly

$$s_n^i \cap U = \{\ell \in Q^n \cap U\colon T(C_\ell) \ge T(C_n^i)\}.$$

By (5.8) and (5.9), therefore, $C_n^i = C_n(T(C_n^i))$ and by Proposition 5.20, $C_n^i = C_n$. Properties (a)–(e) follow immediately. Properties (f) and (g) follow from Propositions 5.16 and 5.17. □

The last major step in establishing the validity of the Tree Algorithm is to show that the termination criteria of the algorithm are equivalent to the condition of finding a value of i that is critical in n. For this purpose we make use of the strict monotonicity property of Assumption 2.2.

PROPOSITION 5.26: If either of the following sets of conditions hold then i is critical in n:

$$\begin{cases} t_n^{i-1} < T(C_n^i)\,, & \text{if } i > 1 \text{ and } k_n^{i-1} \in U; \\ t_n^{i-1} \le T(C_n^i)\,, & \text{if } i > 1 \text{ and } k_n^{i-1} \in L; \\ T(C_n^{i+1}) \le t_n^i & \text{if } i \le q_n \text{ and } k_n^i \in U\,; \text{ and} \\ T(C_n^{i+1}) < t_n^i & \text{if } i \le q_n \text{ and } k_n^i \in L\,; \end{cases} \tag{5.11}$$

or

$$\begin{cases} t_n^{i-1} < T(C_n^{i-1}) & \text{if } i > 1 \text{ and } k_n^{i-1} \in U; \\ t_n^{i-1} \le T(C_n^{i-1}) & \text{if } i > 1 \text{ and } k_n^{i-1} \in L; \\ T(C_n^i) \le t_n^i & \text{if } i \le q_n \text{ and } k_n^i \in U\,; \text{ and} \\ T(C_n^i) < t_n^i & \text{if } i \le q_n \text{ and } k_n^i \in L\,. \end{cases} \tag{5.12}$$

PROOF: Consider the first set of conditions, (5.11). If $k_n^i \in U$ then by Proposition 5.24, $C_n^{i+1} = C_n^i \cup C_{k_n^i}$. Since C_n^i and $C_{k_n^i}$ are mutually disjoint, $T(C_n^i) > T\left[C_{k_n^i}\right]$ would imply $T(C_n^{i+1}) > T\left[C_{k_n^i}\right]$, by Assumption 2.2, and thereby violate the third condition of (5.11). Consequently $T(C_n^i) \le t_n^i$. If $k_n^i \in L$ then by Proposition 5.24, $C_n^i = C_n^{i+1} \cup C_{k_n^i}$. By Assumption 2.2, $T(C_n^i) < T\left[C_{k_n^i}\right] = t_n^i$. It follows that i is critical in n. A parallel proof establishes the result for the second set of conditions. □

THEOREM 5.27: Algorithm 4.1 returns with $Q = Q_n$, $C = C_n$, and $F = F_n$.

PROOF: If $k = k_n^{i-1}$ after step 5 of SOLVE_LOWER, or if $Q = \phi$ (i.e., $i = 1$), then a simple induction proof shows that $C = C_n^i$, $F = F^n \cup (s_n^i \cap U)$, $Q = r_n^i$, $T(C_n^{i+1}) \leq t_n^i$ if $k_n^i \in U$, and $T(C_n^{i+1}) < t_n^i$ if $k_n^i \in L$. If SOLVE_LOWER then terminates, either because $Q = \phi$ or step 8 of SOLVE_LOWER is executed for this value of k, then by (5.11) of Proposition 5.26, i is critical in n and by Proposition 5.25, $C = C_n$, $F = F_n$, and $Q = Q_n \backslash \{n\}$. Step 9 then results in $Q = Q_n$. A parallel proof establishes the result for SOLVE_UPPER using (5.12) of Proposition 5.26. □

The validity of the Tree Algorithm is an easily established consequence of Theorem 5.27. We conclude this section by proving the following lemma, which is used in Section 6.

LEMMA 5.28: If $k_n^i \in U$ then $S(k_n^i) \cap C_n^{i+1} = C_{k_n^i}$. If $k_n^i \in L$ then $S(k_n^i) \cap C_n^i = C_{k_n^i}$.

PROOF: In view of Proposition 5.24, it suffices to show that if $k_n^i \in U$ then $S(k_n^i) \cap C_n^{i+1} \subseteq C_{k_n^i}$, and that if $k_n^i \in L$ then $S(k_n^i) \cap C_n^i \subseteq C_{k_n^i}$. Let $k_n^i \in U$ and let $m \in S(k_n^i) \backslash C_{k_n^i}$. Then $m \in C_\ell$ for some $\ell \in R(k_n^i)$, $\ell \neq k_n^i$. By Lemma 5.13, either $\ell \in F^n$ or $\ell \in Q^n \cap U$. If $\ell \in F^n$ then by (5.6), $m \notin C_n^{i+1}$. If $\ell \in Q^n \cap U$ then $\ell = k_n^j$ for some $j \neq i$. But $\ell = k_n^j \in S(k_n^i)$ implies $k_n^j \in Q^{k_n^i} \cap U$. By Corollary 5.14, $T(C_{k_n^j}) \geq T(C_{k_n^i})$, so $j > i$. By (5.9), $m \notin C_n^{i+1}$. The proof for $k_n^i \in L$ is similar. □

6. IMPLEMENTATION AND RUNNING TIME

In this section we show that the Tree Algorithm can be implemented in $O(N^2)$ time in $O(N)$ space. For a very wide class of objective functions it runs in $O(N \log N)$ time and $O(N)$ space.

6.1 Data Structures and Operations

Three types of data structures are used in the algorithm, as shown below.

Type $\mathscr{C}$

Used for: C, C_n, A_n, and B_n.

Operations Performed:

Union (Figure 7: Lines 4 and 5, Figure 3: Line 7.3);

Set Differences (Figure 3: Line 6.3).

Total size of all type $\mathscr{C}$ structures: $\leq 4N$.

Structure Used: 2–3 trees with ordered leaves.

Type $\mathscr{Q}$

Used for: Q.

Operations Performed:

Union (Figure 7: Line 6, Figure 3: Line 9);

Find smallest element, Find largest element (Figure 3: Line 5);

Remove smallest element, Remove largest element (Figure 3: Lines 6.1, 7.1).

Total size of all type $\mathscr{Q}$ structures: $\leq N$.

Structure Used: 2–3 trees with unordered leaves.

Type $\mathscr{F}$

Used for: F.

Operations Performed:

Union (Figure 7: Line 7);

Add an element (Figure 3: Line 6.2).

Total size of all type $\mathscr{F}$ structures: $\leq$ N.

Structure Used: List.

In the tree algorithm, we store C_m only for those m currently in a type $\mathscr{Q}$ data structure. In Figure 7, we discard the data structures A′ (resp., B′, Q′, F′) after Line 4 (resp., 5, 6, 7) has been executed. Roughly speaking, we only store A_m (resp., B_m , Q_m , F_m) for those m for which A_m (resp., B_m , Q_m , F_m) has been created, but $A_{p(m)}$ (resp., $B_{p(m)}$, $Q_{p(m)}$, $F_{p(m)}$) has not been created. These facts limit the total size of all data structures to O(N).

THEOREM 6.1: The Tree Algorithm can be implemented to run in O(N) space and in O(N log N) time, exclusive of at most 2N computations of the form (1.4).

PROOF: Note that Procedure TREE_SOLVE is called exactly once for each node n in the tree, in reverse depth–first–search order (from n = N down to n = 1). Either SOLVE_UPPER or SOLVE_LOWER is called once for each node in the tree, in standard depth–first–search order (from n = 1 up to n = N).

We first show that the Tree Algorithm can be implemented in O(N) space, and that the number of times that either Line 6.1 or Line 7.1 of Figure 3 is executed is $\leq$ N. Note that each node n enters a type $\mathscr{Q}$ data structure at most once, in Line 9 of Figure 3. Since Q′ is discarded in Line 6 of Figure 7, no node can be in two different type $\mathscr{Q}$ data structures at the same time. Therefore the total size of all type $\mathscr{Q}$ data structures is at most N . Similarly, the total size of all type "A" and type "B" data structures cannot exceed N. Furthermore each node can be deleted from a type $\mathscr{Q}$ data structure at most once, so the total number of times that either Line 6.1 or 7.1 of Figure 3 is executed is at most N. Since a node enters a type $\mathscr{F}$ data structure only if it is simultaneously deleted from a type $\mathscr{Q}$ data structure (see Lines 6.1 and 6.2 of Figure 3), the total size of all type $\mathscr{F}$ data structures can not exceed N.

Lemma 5.28 implies that a node is never in more than two type "C" or type "C_n" data structures at any one time. Therefore the total number of nodes in all type $\mathscr{C}$ data structures can not exceed 4N. Since the space required by these data structures is linear in the number of elements they contain (Aho et.al., 1974), the algorithm can be implemented in O(N) space.

We now claim that each line in Figures 3 and 7 is executed at most 2N times. Clearly all lines of Figure 7 and lines 1, 2, and 9–11 of Figure 3 are performed exactly N times, once for each node in the tree. The number of times that the loop of Figure 3 Lines 3–8 is performed is at most N , plus the number of times that an element is removed from Q in either Line 6.1 or Line 7.1 . A $\mathscr{Q}$ data structure can be expanded only N times (Figure 7, line 6). Once removed from a $\mathscr{Q}$ data structure, an element cannot re–enter. Therefore, deletions from type $\mathscr{Q}$ data structures cannot occur more than N times, so each of Lines 3–8 is executed at most 2N times.

Clearly, each of the operations listed above is executed at most O(N) times. It therefore suffices to show that each of the operations can be implemented in at most O(log N) time. It is well known that this is true for type $\mathscr{Q}$ and type $\mathscr{F}$ data structures (Aho et.al., 1974).

Consider the set unions and set differences performed on type $\mathscr{C}$ data structures. By Lemma 5.28 and Proposition 5.24, the set unions are all of the type $C \leftarrow C' \cup C''$ where $C'' \subset S(m)$ and $S(m) \cap C' = \phi$ for some $m \in N(G)$, and the set differences are all of the form $C' = C \backslash C''$ where $C'' = C \cap S(m)$ for some $m \in N(G)$. The nodes in a type $\mathscr{C}$ data structure are stored in increasing order of n. For both unions and set differences, Property 2 of the depth–first indexing of the nodes (see Section 4) implies that the nodes in C'' are consecutive in C. Thus both the set unions and the set differences can be performed in $O(\log N)$ time by a combination of split and concatenate operations (Aho et.al., 1974). □

6.2 Overall Running Time: Production Planning

For the production planning problem PP_G, $T(C)$ is given by (2.1). Let $K(C) = \sum_{n \in C} K_n$ and $g(C) = \sum_{n \in C} g_n$. $T(C)$ can be computed in constant time if $K(C)$ and $g(C)$ are known. Let $K^j = K(C_j)$ and $g^j = g(C_j)$ for all $j \in Q^n$. Thus we need to maintain $K(C)$ and $g(C)$ for C, a type $\mathscr{C}$ data structure. Proposition 5.29 implies that all set unions performed on type $\mathscr{C}$ data structures involve disjoint sets, and all set differences involve computing $C \backslash C'$ where C' is a subset of C. This justifies the following changes to the algorithms.

Figure 7, Line 1: $A \leftarrow \{n\}$ becomes $K(A) \leftarrow K_n$; $g(A) \leftarrow g_n$. $B \leftarrow \{n\}$ is similarly changed.

Figure 7, Line 4: $A \leftarrow A \cup A'$ becomes $K(A) \leftarrow K(A) + K(A')$; $g(A) \leftarrow g(A) + g(A')$.

Similar changes are made to Figure 7, Line 5 and to Figure 3, Line 7.3.

Figure 3, Line 6.3: $C \leftarrow C \backslash C_k$ becomes $K(C) \leftarrow K(C) - K(C_k)$; $g(C) \leftarrow g(C) - g(C_k)$.

Note that these modifications completely eliminate the need for type $\mathscr{C}$ data structures. The sets $\{C_n \; ; n \in R(N) = F_N \cup (Q_N \cap L)\}$ can easily be computed by a simple search procedure once the algorithm terminates and the set $R(N)$ has been determined.

6.3 Overall Running Time: d–Schur Convex Objectives

In Appendix B we show that if f is a proper convex function of $T = \{T_n : n \in N(G)\}$, the dual of

$$(PS_G) \qquad \text{minimize:} \quad f(T) + \sum_n g_n T_n$$

$$\text{such that:} \quad (1.2) \text{ holds}$$

is

$$(DS_G) \qquad \text{minimize:} \quad f^*(z)$$

such that:

$$\sum_{(n,k) \in A(G)} x_{nk} - \sum_{(m,n) \in A(G)} x_{mn} - z_n = g_n \text{ for all } n \in N(G), \tag{6.3}$$

$$x_{mn} \geq 0 \text{ for all } (m,n) \in A(G) \tag{6.4}$$

where f^* is the conjugate function of f. Veinott (1971) has shown that if f is proper, convex, and d–Schur convex then f^* is proper, convex, and d–Schur convex. (Not all d–Schur convex functions are convex.) He also showed that a single vector $(x_{mn}: (m,n) \in A(G))$ is optimal for every d–Schur convex function f^*. The corresponding set of optimal clusters in PS_G is therefore optimal for every convex, d–Schur convex function f.

An interesting subclass of the d–Schur convex functions is the d–additive convex functions, i.e., functions of the form $f(T) = \sum_n d_n g(T_n/d_n)$ where g is convex. The objective function (1.1) of the production planning problem PP_G is d–additive convex with $g(x) = x^{-1}$ and $d = \left[\sqrt{K_n}: n \in N(G)\right]$. Since PP_G can be solved in $O(N \log N)$ time, optimal clusters for PS_G can be found for any convex, d–Schur convex function f in $O(N \log N)$ time. Optimal values of T are then found by solving

(PC) minimize: $f(T) + \sum_n g_n T_n$

such that: $T_m = T(C_n)$ for all $m \in C_n$, $n \in R(N)$.

If the optimal value of T in PC is not unique, care should be taken to ensure that a solution is selected for which (1.2) holds (Veinott, 1971).

The isotonic regression problem is PS_G with $f(T) = \Sigma_n \frac{1}{2} w_n T_n^2$ and $g_n = -w_n \mu_n$ where w_n is the size of a sample from random variable n and μ_n is the sample mean. Adding α to all sample means μ_n simply translates the solution to (PS_G) by α, so we can and do assume that $\mu_n < 0$ for all n. Since $f(T)$ is d–additive convex with $d_n = w_n$ and with $g(x) = \frac{1}{2}x^2$, the isotonic regression problem is equivalent to the production planning problem PP_G. The proper choice of parameters is $K_n = d_n^2 = w_n^2$ and $g_n = -w_n \mu_n$. Note that (2.1) becomes

$$T(C) = \left[\frac{-\sum_{n \in C} w_n \mu_n}{\sum_{n \in C} w_n} \right]^{-1/2} .$$

Thus $T(C) > T(C')$ if and only if

$$\frac{\sum_{n \in C} w_n \mu_n}{\sum_{n \in C} w_n} > \frac{\sum_{n \in C'} w_n \mu_n}{\sum_{n \in C'} w_n} ,$$

i.e., if the weighted average of the sample means for all $n \in C$ is greater than the corresponding average for all $n \in C'$. Consequently the isotonic regression problem can be solved in $O(N \log N)$ time overall. The same is true of many other d–additive objective functions.

7. CONCLUSIONS

Both an extension of the Economic Order Quantity (EOQ) model to multi–stage production–distribution systems and the isotonic regression problem are known to be equivalent and to be solvable in $O(N^4)$ time. We have shown that these problems are solvable in $O(N \log N)$ time whenever the undirected version of the underlying network is a tree, and that a generalization of these problems is solvable in $O(N \log N)$ time exclusive of at most $2N$ side computations. This algorithm is used as a subroutine in an algorithm for solving the EOQ problem and the isotonic regression problem on general circuitless directed graphs, the subject of a companion paper.

REFERENCES

[1] Aho, A.V., J.E. Hopcroft, and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison–Wesley, Reading, Massachusetts, 1974.

[2] Avriel, M., *Nonlinear Programming: Analysis and Methods*, Prentice–Hall, Englewood Cliffs, New Jersey, 1976.

[3] Federgruen, A. and H. Groenevelt, "Polynomial Network Flow Models with Multiple Sinks: Transformation to Standard Network Models," Working Paper Series No. QM 8531, Graduate School of Management, The University of Rochester, July 1985.

[4] Federgruen, A. and H. Groenevelt, "Two Algorithms for Maximizing a Separable Concave Function over a Polymatroid Feasible Region," Working Paper Series No. QM 8532, Graduate School of Management, The University of Rochester, August 1985.

[5] 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, February 1984.

[6] Jackson, P.L., W.L. Maxwell, and J.A. Muckstadt, "The Joint Replenishment Problem with a Powers–of–Two Restriction," *IIE Transactions*, Vol. 17, No. 1 (March 1985), pp. 25–32.

[7] Jackson, P.L. and R.O. Roundy, "Constructive Algorithm for Planning Production in Multi–Stage Systems with Constant Demand," Technical Report No. 632, School of Operations Research and Industrial Engineering, Cornell University, September 1987.

[8] Maxwell, W.L. and J.A. Muckstadt, "Establishing Consistent and Realistic Reorder Intervals in Production – Distribution Systems," *Operations Research*, Vol. 33, No. 6 (November–December 1985), pp. 1316–1341.

[9] Muckstadt, J.A. and R.O. Roundy, "Planning Shipping Intervals in Multi–Item, One–Warehouse, Multi–Retailer Distribution Systems," Technical Report No. 646, School of Operations Research and Industrial Engineering, Cornell University, January 1985.

[10] Picard, J–C and M. Queyranne, "Integer Minimization of a Separable Convex Function Subject to Variable Upper Bound Constraints," Faculty of Commerce and Business Administration, University of British Columbia, Vancouver, BC, Canada, April 1985.

[11] Rockafellar, R.T., *Convex Analysis*, Princeton University Press, Princeton, New Jersey, 1970.

[12] Roundy, R.O., "A 98%–Effective Lot Sizing Rule for a Multi–Product, Multi–Stage Production/Inventory System," Technical Report No. 642, School of Operations Research and Industrial Engineering, Cornell University, December 1984.

[13] Roundy, R.O., "98%–Effective Integer–Ratio Lot–Sizing for One–Warehouse Multi–Retailer Systems," forthcoming in *Management Science*.

[14] Veinott, A.F., Jr., "Least d–Majorized Network Flows with Inventory and Statistical Applications," *Management Science*, Vol. 17, No. 9 (May 1971), p. 547.

APPENDIX A

THEOREM 3.1: $\{T_j^* ; j \in S(n)\}$ is the optimal solution to P_n if and only if there exists an index set $R(n) \subseteq S(n)$ and a set of clusters $\{C_m; m \in R(n)\}$ satisfying

(a) $\{C_m; m \in R(n)\}$ is a partition of $S(n)$;

(b) C_m is connected in G and $m \in C_m \subseteq S(m)$ for all $m \in R(n)$;

(c) for each $m \in R(n)$ and each $j \in s(C_m)$,

$$T(C_j) \geq T(C_m) \text{ if } j \in U \text{, and}$$
$$T(C_j) \leq T(C_m) \text{ if } j \in L \text{;}$$

(d) for each $m \in R(n)$ and each $j \in C_m$, $j \neq m$,

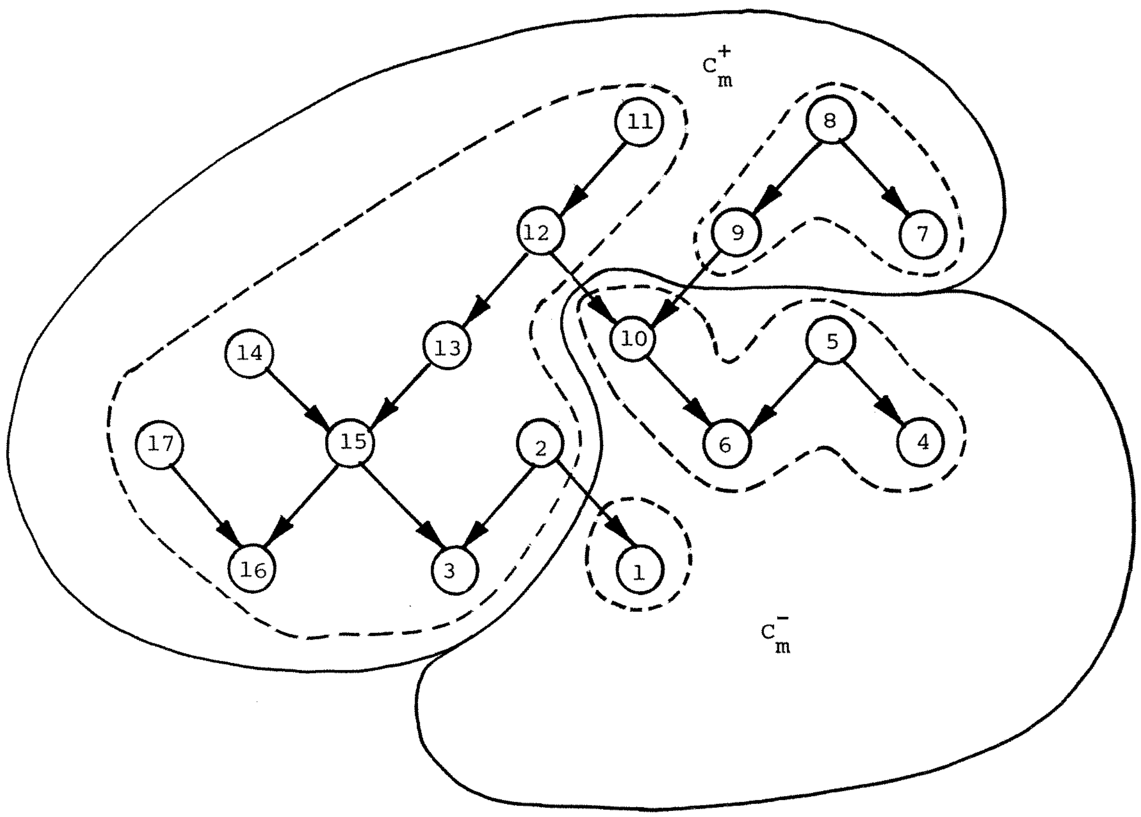
$$T(C_m \cap S(j)) < T(C_m) \text{ if } j \in U \text{, and}$$
$$T(C_m \cap S(j)) > T(C_m) \text{ if } j \in L \text{;}$$

(e) if $j \in C_m$ then $T_j^* = T(C_m)$.

PROOF: Theorem 2.3 in Jackson and Roundy (1987) gives four properties, labeled (a)–(d), which also use clusters to characterize solutions to P_n. We will show the equivalence of those properties to ours. Properties (a), (c), and (e) above are equivalent to properties (a), (b), and (d), respectively, of Theorem 2.3 in Jackson and Roundy (1987). Properties (c) and (d) above imply that each cluster is connected in G and therefore has a unique root satisfying property (b) above. The index set $R(n)$ is simply the collection of the roots of the clusters in the optimal partition. It remains to establish the equivalence of property (d) above and Theorem 2.3(c) in Jackson and Roundy (1987).

A <u>directed cut</u> of a node set $C \subset N(G)$ is a partition of C into nonempty sets C^+, C^- such that there is no arc $(i,j) \in A(G)$, $i \in C^-$, $j \in C^+$. Theorem 2.3(c) in Jackson and Roundy (1987) states that for each cluster C_m, $m \in R(n)$, there is no directed cut (C^+,C^-) of C_m satisfying $T(C^+) \geq T(C^-)$. Trivially, a violation of property (d) above implies a violation of Theorem 2.3(c) in the companion paper. Suppose, for an arbitrary connected cluster $C_m \subseteq N(G)$ with root m, that there exists a directed cut (C^+,C^-) satisfying $T(C^+) \geq T(C^-)$, or equivalently by Assumption 2.2 and Lemma 2.3, $T(C^+) \geq T(C_m)$. We will show that property (d) is also violated for this cluster.

Let $\{C(j); j \in H^+\}$ denote the connected components of C^+ in $G(C_m)$, and let $\{C(j); j \in H^-\}$ denote the connected components of C^-. Let j be the root of $C(j)$. Figure 8 illustrates for $C_m = N(G)$ in the example of Figure 1. If $T(C(j)) < T(C_m)$ for some $j \in H^+$ then we set $C_+ \leftarrow C^+ \backslash C(j)$ and $C_- \leftarrow C^- \cup C(j)$. Lemma 2.3 implies that $C_+ \neq \phi$ and $T(C_+) \geq T(C_m) \geq T(C_-)$. Consequently we can and do assume that $T(C(j)) \geq T(C_m)$ for all $j \in H^+$. Similarly, we can assume that $T(C^-) \leq T(C_m)$ for all $j \in H^-$.



$H^+ = \{17,9\}, \ T(C_m^+)^2 = 89.5;$

$H^- = \{1,10\}, \ T(C_m^-)^2 = 39.0.$

Figure 8. Connected Components of a Directed Cut.

Since C_m is connected, $G(C_m)$ is a tree and there must exist at least one root $j \in H^+ \cup H^-$ such that $C_m \cap S(j) = C(j)$; otherwise one could construct a circuit in $G(C_m)$. In Figure 8, nodes 1 and 9 are the only such roots. If $j = m$ then $C(j) = C_m$ and either $H^+ = \phi$ or $H^- = \phi$. Consequently $j \neq m$. Suppose $j \in H^+$. Observe that $p(j) \in C^-$; else j cannot be the root of $C(j)$. Thus $j \in U$, and $T(C_m \cap S(j)) = T(C(j)) \geq T(C_m)$, violating (d) above. Similarly, if $j \in H^-$ then (d) is violated. □

APPENDIX B

In this appendix we show that the dual of PS_G is DS_G. Following Avriel (1976), let

$$\varphi(T,\zeta) = \begin{cases} f(T) + \sum_n g_n T_n & \text{if } T_m - T_n \geq \zeta_{mn} \text{ for all } (m,n) \in A(G), \\ +\infty & \text{otherwise.} \end{cases}$$

The dual of PS_G is to minimize

$$\varphi^*(x) = \sup_{T,\zeta} [x\zeta - \varphi(T,\zeta)]. \tag{B.1}$$

where $x = (x_{mn}: (m,n) \in A(G))$ and $\zeta = (\zeta_{mn}: (m,n) \in A(G))$.

Note that $\varphi^*(x) = \infty$ if $x_{mn} < 0$ for some $(m,n) \in A$ (let $\zeta_{mn} \to -\infty$). Assuming $x \geq 0$, it is optimal to choose $\zeta_{mn} = T_m - T_n$ in (B.1), so we have

$$\varphi^*(x) = \sup_T \left[\sum_n T_n z_n - f(T)\right] = f^*(z)$$

where z_n is defined by (6.3). Therefore the dual of PS_G is to minimize $f^*(z)$ subject to (6.3) and (6.4).

GLOSSARY

Section 1

$G = (N(G), A(G))$	Graph with node set $N(G)$ and arc set $A(G)$
T_n	Decision variable, $n \in N(G)$
$f_n(\cdot)$	Cost function, $n \in N(G)$
P_G	Constrained minimization problem defined on G
K_n, g_n	Coefficients in EOQ cost function
$P(\cdot)$	Penalty function
PP_G	Production planning problem defined on G
N	Cardinality of $N(G)$
$f(T,C) = \sum_{n \in C} f_n(T)$	Cost of common decision, T, over $C \subseteq N(G)$

Section 2

$\bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$	
$T(C)$	Cost minimizing decision for $C \subseteq N(G)$
$K(C) = \sum_{n \in C} K_n$	Setup cost for cluster
$g(C) = \sum_{n \in C} g_n$	Holding cost factor for cluster

Section 3

N	Root of tree
$p(n)$	Parent of node n
$s(n)$	Sons of node n
$S(n)$	Successors of node n
U	Set of upper nodes

L	Set of lower nodes
Sign (n)	Sign (upper/lower) of node n
$i \sim n$	Node i is level with node n
C_m	Cluster with root m. (Later, the unique cluster in optimal partition rooted at m.)
$s(C)$	Cluster–sons of cluster C
$A(n)$	Arc set associated with $S(n)$
$G(n)$	Sub–tree rooted at node n
P_n	Cost minimization problem defined on $G(n)$
$R(n)$	Optimal index set for P_n
$\{C_m; m \in R(n)\}$	Optimal partition for P_n
$\{T_j^*; j \in S(n)\}$	Optimal solution to P_n
D_n	Set of nodes dominated in $S(n)$
M_n	Set of nodes undominated in $S(n)$
I_n	Set of nodes isolated in $S(n)$
E_n	Set of nodes defeated in $S(n)$
F_n	Set of nodes final in $S(n)$
Q_n	Set of candidate nodes in $S(n)$
A_n	Set of nodes above n, including n
B_n	Set of nodes below n, including n

Section 4

$C_M = \{C_n; n \in M\}$	Collection of lead clusters for arbitrary index set, M
C, F, Q, A, B	Working sets to build C_n, F_n, Q_n, A_n, and B_n, respectively

Section 5

$T^n = \{T_j^n; j \in S(n)\}$	Optimal solution to P_n
$C_n(T)$	Point–to–cluster function (5.8)
T^o	Fixed point solution: $T(C_n(T^o)) = T^o$
q_n	Cardinality of Q^n
r_n^i	the $(i-1)$–st smallest elements of Q^n
s_n^i	the $(q_n - i + 1)$–st largest elements of Q^n
C_n^i	Index to cluster function (5.9)
t_n^i	$T(C_k)$ where k is the i–th smallest element of Q^n

Section 6

$\mathscr{C}$	2–3 trees with ordered leaves
$\mathscr{Q}$	2–3 trees with unordered leaves
$\mathscr{F}$	Lists
PS_G	d–Schur convex cost minimization problem defined on G
DS_G	Dual of PS_G