An Algorithm for the K Best Solutions of the Resource Allocation Problem

N. KATOH

The Center for Adult Diseases, Osaka, Japan

AND

T. IBARAKI AND H. MINE

Kyoto University, Kyoto, Japan

ABSTRACT An algorithm is presented for obtaining the K best solutions of the resource allocation problem with an objective function which is the sum of convex functions of one variable It requires $O(T^* + K\log K + K\sqrt{n\log n})$ time and $O(K\sqrt{n\log n} + n)$ space, where n is the number of variables and T^* is the computational time to obtain the best solution

KEY WORDS AND PHRASES resource allocation problem, K best solutions, computational complexity

CR CATEGORIES 5 25, 5 30, 5 41

1. *Introduction*

The following resource allocation problem (also called the distribution of efforts problem) is extensively studied:

$$P: \quad \text{minimize} \quad z(x) = \sum_{i=1}^{n} f_i(x_i), \qquad \text{subject to} \quad \sum_{i=1}^{n} x_i = N \quad \text{and} \quad x_i: \text{nonnegative integers}, \tag{1.1}$$

where the f_i are convex functions defined over $[0, N]$ and N is a positive integer. This simple integer programming problem has a rather long history, as evidenced by numerous papers [2–13, 15, 17–27]. In addition to the standard procedure of dynamic programming (e.g., [10] and textbooks [2, 26]), which is applicable even if the f_i are not convex, several more efficient algorithms are known. The first one is called the incremental method [4, 5, 9, 11, 18–21, 24]. If each evaluation of $f_i(x_i)$ can be done in constant time, an appropriate implementation of this method runs in time $O(N\log n + n)$. The method of [27] requires $O(c(n, N) + n\log n)$ time, where $c(n, N)$ is the time required to solve the continuous problem P' obtained from P by dropping the integrality condition on the x_i. A third type of approach is exemplified by [6, 7,

Authors' present addresses N Katoh, Department of Management Science, Kobe University of Commerce, Kobe, Japan, T. Ibaraki and H Mine, Department of Applied Mathematics and Physics, Faculty of Engineering, Kyoto University, Kyoto, Japan.

13], the most efficient of which [6] requires only $O(n\log(N/n))$ time if $N \geq n$ and $O(n)$ time if $N < n$.

We consider the following generalization of this problem. Let an integer solution satisfying $\sum_{i=1}^{n} x_i = N$ and $x_i \geq 0$ be called *feasible*. The *kth best solution* $x^k = (x_1^k, x_2^k, \ldots, x_n^k)$ is defined recursively as follows.

(1) x^1 is an optimal solution of P, that is, a feasible solution minimizing the objective value $z(x)$.
(2) x^k with $k \geq 2$ is a feasible solution of P with the minimum objective value among those different from $x^1, x^2, \ldots, x^{k-1}$.

In this paper we present an algorithm for obtaining the K best solutions $x^1, x^2, \ldots, x^K$ of P. In real-life problems the simple structure of the resource allocation problem is usually attained by neglecting some complicating side constraints; hence an optimal solution may not satisfy such side constraints. References [22, 23, 26] contain some examples of such resource allocation problems with more than one constraint. In this respect it is important to obtain more than one good solution of P, preferably the K best solutions, where K is a given positive integer. A best solution satisfying the suppressed side constraints may then be found among the K solutions obtained.

Our algorithm requires $O(T^* + K\log K + K\sqrt{n\log n})$ time and $O(K\sqrt{n\log n} + n)$ space to obtain K best solutions, where T^* denotes the time to obtain x^1. It partitions the solution space into finer and finer subsets and computes $x^1, x^2, \ldots, x^K$ by systematically obtaining the best solutions in the partitioned subsets. The general scheme is based on the framework described in Lawler [16].

Section 2 gives necessary definitions and basic results. Section 3 gives an outline of the entire algorithm. Section 4 gives a detailed description of the algorithm and analyzes the time and space requirements. The algorithm explained in Sections 3 and 4 requires $O(T^* + K\log K + Kn)$ time and $O(Kn)$ space. These are reduced to $O(T^* + K\log K + K\sqrt{n\log n})$ and $O(K\sqrt{n\log n} + n)$, respectively, in Section 5.

2. *Definitions and Basic Concepts*

For an integer x with $0 \leq x \leq N$, let

$$d_i^+(x) = f_i(x+1) - f_i(x), \tag{2.1}$$

$$d_i^-(x) = f_i(x) - f_i(x-1), \tag{2.2}$$

where $f_i(-1) = +\infty$ and $f_i(N+1) = +\infty$ are assumed by convention. Note that $d_i^+(x) = d_i^-(x+1)$, and that $d_i^+(x)$ and $d_i^-(x)$ are both nondecreasing by the convexity of f_i. We assume throughout this paper that each $f_i(x_i)$ can be evaluated in constant time. For a feasible solution $x = (x_1, x_2, \ldots, x_n)$, a pair of indices $[i, j]$ is called an *exchange* if $0 \leq x_i < N$, $0 < x_j \leq N$, and $i \neq j$. Applying an exchange $[i, j]$ to x yields another feasible solution $x' = (x_1, \ldots, x_i + 1, \ldots, x_j - 1, \ldots, x_n)$ with the objective value $z(x) + c(i, j)$, where

$$c(i, j) = d_i^+(x_i) - d_j^-(x_j).$$

LEMMA 2.1 [17]. *A feasible solution x is optimal if and only if there is no exchange with negative cost.*

LEMMA 2.2. *For an optimal solution x^1, let $[i, j]$ be an exchange with minimum cost (which is nonnegative). Then the solution $x^\# = (x_1^1, \ldots, x_i^1 + 1, \ldots, x_j^1 - 1, \ldots, x_n^1)$ obtained by applying $[i, j]$ to x^1 is a second best solution x^2.*

PROOF. Let $\tilde{x}$ be a second best solution not equal to $x^{\#}$. Then there exists a pair of indices p, q such that $\tilde{x}_p > x^1_p$ and $\tilde{x}_q < x^1_q$. From $d^+_p(\tilde{x}_p - 1) \geq d^+_p(x^1_p)$ and $d^-_q(\tilde{x}_q + 1) \leq d^-_q(x^1_q)$, it follows that a feasible solution $x' = (\tilde{x}_1, \ldots, \tilde{x}_p - 1, \ldots, \tilde{x}_q + 1, \ldots, \tilde{x}_n)$ has an objective value not greater than $\tilde{x}$ because $z(\tilde{x}) - z(x') = d^+_p(\tilde{x}_p - 1) - d^-_q(\tilde{x}_q + 1) \geq d^+_p(x^1_p) - d^-_q(x^1_q) \geq d^+_i(x^1_i) - d^-_j(x^1_j) \geq 0$ by the optimality of x^1 and the minimality of exchange $[i, j]$. Repeating this, we eventually obtain a feasible solution $\hat{x}$ such that $z(\tilde{x}) \geq z(\hat{x})$, and $\hat{x}_l = x^1_l + 1$ and $\hat{x}_m = x^1_m - 1$ for some l and m but $\hat{x}_k = x^1_k$ for all $k \neq l, m$. Then $z(\tilde{x}) \geq z(\hat{x}) \geq z(x^{\#})$ follows from $z(\hat{x}) - z(x^{\#}) = (d^+_l(x^1_l) - d^-_m(x^1_m)) - (d^+_i(x^1_i) - d^-_j(x^1_j)) \geq 0$ by the minimality of $[i, j]$. This implies that $z(\tilde{x}) = z(x^{\#})$ and therefore that $x^{\#}$ is also a second best solution. □

To construct an algorithm for computing x^k for $k \geq 3$, we need to generalize Lemma 2.2 as follows. The proof is similar to Lemma 2.2.

LEMMA 2.3. *Let two n-dimensional integer vectors $\bar{x}$ and $\underline{x}$ with $0 \leq \underline{x} \leq \bar{x}$ be given. Let $x = x^*$ be a best feasible solution of P among those satisfying*

$$\underline{x}_i \leq x_i \leq \bar{x}_i, \qquad i = 1, 2, \ldots, n.$$

Then a second best solution $\tilde{x}$ satisfying the above constraint is obtained by applying $[i', j']$ to x^*, where $[i', j']$ is a minimum exchange satisfying $x^*_{i'} < \bar{x}_{i'}$ and $x^*_{j'} > \underline{x}_{j'}$.

3. *The Outline of the Entire Algorithm*

Our algorithm consists of routines COMPBS and KBS. COMPBS computes x^k when the first $k - 1$ best solutions $x^1, x^2, \ldots, x^{k-1}$ are given. KBS generates all the K best solutions using COMPBS as a subroutine.

Now assume that $x^1, x^2, \ldots, x^{k-1}$, $k > 1$, have been generated. The set of remaining feasible solutions is partitioned into $k - 1$ disjoint subsets,

$$P^m(k-1) = \{x^l \mid l > k-1, \underline{x}^m(k-1) \leq x^l \leq \bar{x}^m(k-1)\}, \qquad m = 1, 2, \ldots, k-1. \tag{3.1}$$

As will be discussed shortly, these sets have the property that x^k is equal to a solution with the minimum objective value among those obtained as best solutions in the respective $P^m(k-1)$.

Vectors $\underline{x}^m(k-1)$ and $\bar{x}^m(k-1)$ are recursively defined as follows. Initially, when $k = 2$ (i.e., only x^1 is obtained), $\bar{x}^m(1)$ and $\underline{x}^m(1)$ for $m = 1$ are given by

$$\begin{aligned} \bar{x}^1(1) &= (N, N, \ldots, N), \\ \underline{x}^1(1) &= (0, 0, \ldots, 0). \end{aligned} \tag{3.2}$$

In general, let $\bar{x}^m(k-1)$ and $\underline{x}^m(k-1)$, $1 \leq m \leq k-1$, be given, and assume that x^k is obtained from x^{m^*} by applying an exchange $[i^*, j^*]$. Then $\bar{x}^m(k)$ and $\underline{x}^m(k)$ are defined as follows:

$$\begin{aligned} \bar{x}^{m^*}(k) &= (\bar{x}^{m^*}_1(k-1), \ldots, x^{m^*}_{i^*}, \ldots, \bar{x}^{m^*}_n(k-1)), \\ \underline{x}^{m^*}(k) &= \underline{x}^{m^*}(k-1), \\ \bar{x}^k(k) &= \bar{x}^{m^*}(k-1), \\ \underline{x}^k(k) &= (\underline{x}^{m^*}_1(k-1), \ldots, x^k_{i^*}(=x^{m^*}_{i^*} + 1), \ldots, \underline{x}^{m^*}_n(k-1)), \\ \bar{x}^l(k) &= \bar{x}^l(k-1), \qquad \underline{x}^l(k) = \underline{x}^l(k-1), \qquad \text{for all } l \neq m^*, k. \end{aligned} \tag{3.3}$$

These new sets define $P^m(k)$ for $m = 1, 2, \ldots, k$. From this definition and Lemma 2.3, the next lemma is obvious.

LEMMA 3.1. *Let k be $2 \le k \le K$.*

(1) *For $m = 1, 2, \ldots, k-1$, x^m is a best feasible solution satisfying*

$$\underline{x}^m(k-1) \le x \le \bar{x}^m(k-1). \tag{3.4}$$

Furthermore, no other x^l, $l = 1, 2, \ldots, m-1, m+1, \ldots, k-1$, satisfies (3.4).

(2) *Any feasible solution x of P satisfies* (3.4) *for exactly one m with $1 \le m \le k-1$.*

Lemma 3.1(2) asserts that

$$\bigcup_{m=1}^{k-1} P^m(k-1) = (\text{the set of all feasible solutions of } P) - \{x^1, x^2, \ldots, x^{k-1}\},$$

and $P^m(k-1) \cap P^l(k-1) = \emptyset$ for $m \neq l$, since the condition $l > k-1$ in each $P^m(k-1)$ excludes $x^1, x^2, \ldots, x^{k-1}$. Thus, letting $\tilde{x}^m$ be a best feasible solution in each $P^m(k-1)$, x^k is given as a best one in the set $\{\tilde{x}^m \mid m = 1, 2, \ldots, k-1\}$. Lemma 3.1(1) says that $\tilde{x}^m$ is a second best solution among those satisfying (3.4).

In order to compute $\tilde{x}^m$ according to Lemma 2.3, we maintain two sets of labels,

$$\begin{aligned} D_+^m(k-1) &= \{d_i^+(x_i^m) \mid 1 \le i \le n,\ x_i^m < \bar{x}_i^m(k-1)\}, \\ D_-^m(k-1) &= \{d_j^-(x_j^m) \mid 1 \le j \le n,\ x_j^m > \underline{x}_j^m(k-1)\}, \end{aligned} \tag{3.5}$$

for $m = 1, 2, \ldots, k-1$ (if $d_i^+(x_i^m) = d_{i'}^+(x_{i'}^m)$, both are stored). $D_+^m(k-1)$ and $D_-^m(k-1)$ contain at most $O(n)$ labels. A minimum exchange $[i', j']$ of Lemma 2.3 for $\bar{x} = \bar{x}^m(k-1)$ and $\underline{x} = \underline{x}^m(k-1)$ is then determined as follows. Let i_1 and i_2 be the indices of the first and second minimum $d_i^+(x_i^m)$ in $D_+^m(k-1)$, respectively, and j_1 and j_2 be the indices of the first and second maximum $d_j^-(x_j^m)$ in $D_-^m(k-1)$. Then

$$[i', j'] = \begin{cases} [i_1, j_1] & \text{if } i_1 \neq j_1, \\ [p, q] & \text{if } i_1 = j_1 \text{ and } c(p, q) = \min[c(i_1, j_2), c(i_2, j_1)], \\ & \text{where } p \in \{i_1, i_2\},\ q \in \{j_1, j_2\}. \end{cases} \tag{3.6}$$

Note that i_1, i_2, j_1, and j_2 are computed in $O(\log n)$ time if the $D_\pm^m(k-1)$ use appropriate data structure (any efficient priority queue discussed in [1, 14] for example). $[i', j']$ is then computed by (3.6), and $\tilde{x}^m$ is obtained from x^m by applying exchange $[i', j']$.

When x^k is obtained as $\tilde{x}^{m^*}$ (which is generated from x^{m^*} by exchange $[i^*, j^*]$), the $D_\pm^m(k)$ are computed from the $D_\pm^m(k-1)$ as follows (see (3.3)):

$$D_+^{m^*}(k) = D_+^{m^*}(k-1) - \{d_{i^*}^+(x_{i^*}^{m^*})\}, \tag{3.7}$$

$$D_-^{m^*}(k) = D_-^{m^*}(k-1), \tag{3.8}$$

$$D_+^k(k) = D_+^{m^*}(k-1) \cup \{d_{i^*}^+(x_{i^*}^{m^*} + 1),\ d_{j^*}^+(x_{j^*}^{m^*} - 1)\} - \{d_{i^*}^+(x_{i^*}^{m^*}),\ d_{j^*}^+(x_{j^*}^{m^*})\}, \tag{3.9}$$

$$D_-^k(k) = D_-^m(k-1) \cup \{d_{j^*}^-(x_{j^*}^{m^*} - 1)\} - \{d_{i^*}^-(x_{i^*}^{m^*}),\ d_{j^*}^-(x_{j^*}^{m^*})\}, \tag{3.10}$$

$$D_+^m(k) = D_+^m(k-1), \qquad D_-^m(k) = D_-^m(k-1) \qquad \text{for } m \neq m^*, k. \tag{3.11}$$

(Although not explicitly written, it should be understood in (3.9) and (3.10) that those $d_i^\pm(x_i^m)$ whose variables violate the condition of (3.5) are not included in $D_\pm^k(k)$.) Using the appropriate data structure for a priority queue, the deletion of an element from $D_\pm^m(k-1)$ and the addition of an element to $D_\pm^m(k-1)$ are each done in $O(\log n)$ steps (e.g., [1, 14]). Thus the $D_\pm^{m^*}(k)$ are computed in $O(\log n)$ steps and the $D_\pm^k(k)$ are computed in $O(n)$ steps (since $D_\pm^{m^*}(k-1)$ must be copied and it requires $O(n)$ steps). Other $D_\pm^m(k)$ do not require any computation time because the $D_\pm^m(k-1)$ can be used as $D_\pm^m(k)$ with no change.

We note that each $P^m(k-1)$ is represented in our algorithm by the following list:

$$P^m(k-1) = (c', [i', j'], x^m, \underline{x}^m(k-1), \bar{x}^m(k-1), D^m_-(k-1), D^m_+(k-1)), \quad (3.12)$$

where $c' = z(x^m) + c(i', j')$ and $[i', j']$ is a minimum exchange with respect to $\bar{x}^m(k-1)$ and $\underline{x}^m(k-1)$. This list uses $O(n)$ space.

The computation of $x^k = \bar{x}^{m^*}$, $\underline{x}^m(k)$, $\bar{x}^m(k)$, and $D^m_\pm(k)$ for each k described above constitutes Subroutine COMPBS. COMPBS is repeated for $k = 2, 3, \ldots, K$. The entire procedure is organized as KBS.

4. *Algorithms KBS and COMPBS*

This section describes algorithms KBS and COMPBS in an ALGOL-like language and then analyzes its running time.

```
    Procedure KBS(P, K);
    begin
        comment This procedure computes x^1, x^2, . . , x^K together with their costs c^1, c^2, . . . , c^K. If P
                does not have K feasible solutions, KBS terminates after generating all feasible solutions;
1       Find an optimal solution x^1 for problem P,
2       x̄^1(1) ← (N, N, . . . , N), x_^1(1) ← (0, 0, . . , 0); compute D_^1(1) and D_+^1(1);
3       Find a minimum exchange [i', j'] with cost c(i', j'),
4       P^1(1) ← (z(x^1) + c(i', j'), [i', j'], x^1, x_^1(1), x̄^1(1), D_^1(1), D_+^1(1)),
5       for k = 2 until K do call COMPBS(P^m(k − 1) | m = 1, 2, . . , k − 1),
    end KBS,

    Subroutine COMPBS(P^m(k − 1) | m = 1, 2, . . , k − 1),
    begin
1       Find P^{m*}(k − 1) = (c*, [i*, j*], x_^{m*}, x^{m*}(k − 1), x̄^{m*}(k − 1), D_^{m*}(k − 1), D_+^{m*}(k − 1)) with the
          minimum cost c* among P^m(k − 1), m = 1, 2, . . , k − 1;
2       if c* = ∞ then stop (all feasible solutions have been output and P has only k − 1 feasible
          solutions);
3       else begin
                comment Computation of x^k,
4               Output [i*, j*], m*, and c* as x^k (x^k is obtained from x^{m*} by exchange [i*, j*]) and c^k,
                  respectively;
                comment Updating P^{m*}(k);
5               Construct x_^{m*}(k) and x̄^{m*}(k) by (3.3);
6               Construct D_^{m*}(k) and D_+^{m*}(k) by (3.7) and (3.8);
7               Using (3.6), find a minimum exchange [i', j'] for D_±^{m*}(k);
8               if such [i', j'] exists then P^{m*}(k) ← (c^{m*} + c(i', j'), [i', j'], x^{m*}, x_^{m*}(k), x̄^{m*}(k),
                  D_^{m*}(k), D_+^{m*}(k));
9               else (i.e., D_+^{m*}(k) = ∅ or D_^{m*}(k) = ∅) P^{m*}(k) ← (∞, ∅, ∅, ∅, ∅, ∅, ∅);
                comment Computation of P^k(k);
10              Construct x_^k(k) and x̄^k(k) by (3.3);
11              Construct D_^k(k) and D_+^k(k) by (3.9) and (3.10);
12              Using (3.6), find a minimum exchange [i', j'] for D_±^k(k),
13              if such [i', j'] exists then P^k(k) ← (c^k + c(i', j'), [i', j'], x^k, x_^k(k), x̄^k(k), D_^k(k), D_+^k(k));
14              else P^k(k) ← (∞, ∅, ∅, ∅, ∅, ∅, ∅);
                comment Computation of other P^m(k);
15              P^m(k) ← P^m(k − 1) for m ≠ m*, k;
16          end
        return
    end COMPBS;
```

LEMMA 4.1. *For each $k = 2, 3, \ldots, K$, COMPBS correctly computes x^k and $P^m(k)$ for $m = 1, 2, \ldots, k$ in $O(\log K + n)$ time.*

PROOF. The correctness is obvious from the discussion in Sections 2 and 3. The time requirement is considered here. Line 1 of COMPBS is done in $O(\log(k-1))$

$\leq O(\log K)$ time if the $P^m(k-1)$ are linked in the form of an efficient priority queue [1, 14]. Lines 2 and 3 are executed in constant time. Line 4 is executed in constant time since x^k is output only by m^* and $[i^*, j^*]$ instead of the full vector $x^k = (x_1^k, \ldots, x_n^k)$. Line 5 is executed in constant time, as is obvious from (3.3). Line 6 requires $O(\log n)$ time as explained after (3.11). Lines 7 and 12 require $O(\log n)$ time, as explained after (3.6). Hence $P^{m^*}(k)$ of lines 8 and 9 is obtained in $O(\log n)$ time. Line 10 requires $O(n)$ time, as is obvious from (3.3) (note that $O(n)$ is necessary to copy $\underline{x}^{m^*}(k-1)$ and $\bar{x}^{m^*}(k-1)$ beforehand). Line 11 requires $O(n)$ time, as explained after (3.11). This implies that $P^k(k)$ of line 13 or 14 is constructed in $O(n)$ time. Line 15 requires constant time because it is accomplished by keeping the previous data. The adjustment of links among $P^m(k)$, $m = 1, 2, \ldots, k$, (which are used at line 1) as a result of the addition of $P^k(k)$ and the modification of $P^{m^*}(k)$ is done in $O(\log k)$ ($\leq O(\log K)$) time. Thus all computation in COMPBS is done in $O(\log K + n)$ steps. □

THEOREM 4.2. *KBS correctly generates the K best solutions from x^1 to x^K in $O(T^* + Kn + K \log K)$ time and $O(Kn)$ space, where T^* is the time required to compute x^1. If P does not have K feasible solutions, KBS terminates after generating all feasible solutions.*

PROOF. The correctness of KBS follows from the previous discussion. The time complexity is analyzed here. Line 1 of KBS requires T^* time. Line 2 obviously requires $O(n)$ time since the initial construction of a priority queue is done in $O(n)$ time [1, 14]. Line 3 requires $O(\log n)$ time, as explained after (3.6), and line 4 requires $O(n)$ time. Line 5 calls COMPBS $K-1$ times and requires $O(K \log K + Kn)$ time in total by Lemma 4.1. Thus the total time is $O(T^* + Kn + K \log K)$. Finally, $O(Kn)$ space is required to store $P^m(k)$, $m = 1, 2, \ldots, k$, since each $P^m(k)$ needs $O(n)$ space; the space for other data is obviously dominated by $O(Kn)$. □

The time T^* was discussed in Section 1. The most time consuming part is the generation of $P^k(k)$ at lines 10–14 of COMPBS; this requires $O(n)$ time since it must copy $\underline{x}^{m^*}(k-1)$, $\bar{x}^{m^*}(k-1)$, and $D_{\pm}^{m^*}(k-1)$. The time requirement of this part will be further reduced in the next section by introducing a sophisticated data structure for $P^m(k)$.

5. *Time and Space Reduction*

The time and space required for KBS will be reduced to

$$O(T^* + K \log K + K\sqrt{n \log n}) \quad \text{and} \quad O(K\sqrt{n \log n} + n),$$

respectively, in this section. To attain these bounds, it is essential to output x^k by $[i^*, j^*]$ and m^* (as indicated at line 4 of COMPBS). If the n-dimensional vectors x^k are to be directly output, $O(Kn)$ time is required only for this purpose.

5.1 DERIVATION TREE T AND SOME DEFINITIONS. First we represent the process of deriving $x^2, x^3, \ldots$ from x^1 by a rooted tree T defined as follows: x^{m^*} is the *father* of x^k (or x^k is a *son* of x^{m^*}) if x^k is obtained from x^{m^*} by an exchange. x^l is an *ancestor* of x^m (or x^m is a *descendant* of x^l) if there is a sequence x^{l_1} $(=x^l)$, $x^{l_2}, \ldots, x^{l_p}$ $(=x^m)$ such that x^{l_i} is the father of $x^{l_{i+1}}$, $i = 1, 2, \ldots, p-1$. x^m and $x^{m'}$, $m \neq m'$, are brothers if their fathers coincide; x^m is placed to the left of $x^{m'}$ if $m < m'$. Obviously x^1 is the root of this tree. For any ancestor x^l of x^m, $\pi(l, m)$ denotes the path from x^l to x^m in T. For any x^p, let $[i^*(p), j^*(p)]$ denote the exchange with which x^p is obtained from its father.

Now consider the time instance when x^k is attached to x^{m^*} in T, corresponding to the generation of x^k from x^{m^*}. Special attention needs to be paid to the fact that $\bar{x}^{m^*}(k)$ of (3.3) is modified whenever its son is created. This implies that each $\bar{x}^m(k)$ is generally dependent upon the entire history of how $x^1, x^2, \ldots, x^k$ have been generated (i.e., the structure of T). Let x^l be an ancestor of x^m in T. Then we see from (3.3) that $\bar{x}^m(k)$ is computed from $\bar{x}^l(l)$ by taking into account the effects caused by those nodes in T which are either left brothers of some x^q on $\pi(l, m)$ or are sons of x^m. To carry out this computation, we define

$$f(m): \quad x^{f(m)} \text{ is the father of } x^m; \tag{5.1}$$

$$s(m): \quad x^{s(m)} \text{ is the rightmost son of } x^m \quad (s(m) = \varnothing \text{ if } x^m \text{ has no son}); \tag{5.2}$$

$$b(m): \quad x^{b(m)} \text{ is the brother immediately to the left of } x^m \quad (b(m) = \varnothing \text{ if } x^m \text{ is the leftmost son}); \tag{5.3}$$

$$I_b(l, m) = \{i^*(p) \mid x^p \text{ is a left brother of some } x^q \text{ (note } p \neq q) \text{ on } \pi(l, m), \text{ and } q \neq l\}; \tag{5.4}$$

$$I_s(m) = \{i^*(p) \mid x^p \text{ is a son of } x^m\}; \tag{5.5}$$

$$J(l, m) = \{j^*(p) \mid x^p \text{ is on } \pi(l, m) \text{ and } p \neq l\}; \tag{5.6}$$

$$I(l, m) = \{i^*(p) \mid x^p \text{ is on } \pi(l, m) \text{ and } p \neq l\}. \tag{5.7}$$

Then we have, by the definition of T and (3.3),

$$x^m = x^l + \sum_{i \in I(l,m)} e_i - \sum_{j \in J(l,m)} e_j, \qquad \text{where} \quad e_q = (0, \ldots, 0, \overset{q}{1}, 0, \ldots, 0), \tag{5.8}$$

$$\underline{x}_i^m(k) = \begin{cases} x_i^{p^*}, & \text{where } p^* = \max\{p \mid i^*(p) \in I(l, m) \text{ and } i^*(p) = i\}, \\ \underline{x}_i^l(k), & \text{if } p^* \text{ is not defined}, \quad i = 1, 2, \ldots, n, \end{cases} \tag{5.9}$$

$$\bar{x}_i^m(k) = \begin{cases} x_i^{f(p^*)}, & \text{where } p^* = \max\{p \mid i^*(p) \in I_b(l, m) \cup I_s(m) \text{ and } i^*(p) = i\}, \\ \bar{x}_i^l(l), & \text{if } p^* \text{ is not defined}, \quad i = 1, 2, \ldots, n. \end{cases} \tag{5.10}$$

5.2 Types 1 and 2 Data Structures of $P^m(k)$. On the basis of the above notations we now introduce two types of data structure of $P^m(k)$ with which the time and space reduction is attained. Call the following $P^m(k)$ *type* 1:

$$P^m(k) = (c', [i', j'], [i^*(m), j^*(m)], x^m, \underline{x}^m(k), \bar{x}^m(k), D_-^m(k), D_+^m(k), \bar{x}^m(m), D_+^m(m), f(m), f'(m), b(m), s(m)), \tag{5.11}$$

where c' and $[i', j']$ are defined after (3.12) and $f'(m)$ will be explained later in Section 5.3. Note that x^m is obtained from $x^{f(m)}$ by an exchange $[i^*(m), j^*(m)]$. A type-1 $P^m(k)$ requires $O(n)$ space.

We also use a simplified data structure called *type* 2:

$$P^m(k) = (c', [i', j'], [i^*(m), j^*(m)], f(m), f'(m), b(m), s(m)). \tag{5.12}$$

A type-2 $P^m(k)$ requires only constant space.

Consider now the instance immediately after x^k is obtained from x^{m^*}. It is shown in this section and the rest of Section 5 how a type-2 $P^m(k)$ (though only the cases of $m = m^*, k$ are our concern because $P^m(k) = P^m(k-1)$ for $m \neq m^*, k$, as discussed

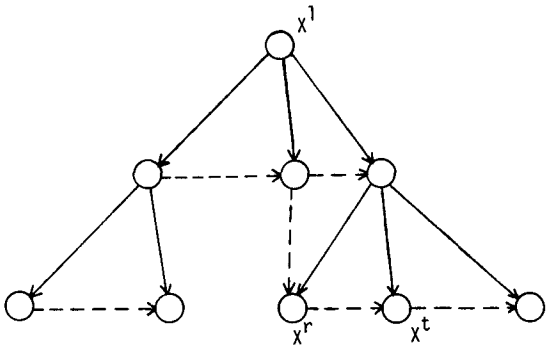

FIG 1 The relation between a rooted tree T and the directed forest T' (solid lines denote T and broken lines denote T')

above) can be constructed from type-1 $P^l(k)$, where x^l is an ancestor of x^m such that no node x^p on $\pi(l, m)$, $p \neq l$, has type-1 $P^p(k)$. The essential part is the computation of $[i', j']$ and c' of $P^m(k)$, which is done after temporarily constructing $D^m_{\pm}(k)$.

On the basis of (5.9) and (3.5), $D^m_{-}(k)$ is constructed in $O(|\pi(l, m)| \log n)$ time as follows ($|\pi|$ denotes the length of path π). $\underline{x}^m(k)$ is obtained from $\underline{x}^l(k)$ by changing $\underline{x}^l_{i^*(p)}(k)$ to $x^p_{i^*(p)}$ for each x^p on $\pi(l, m)$ by following $\pi(l, m)$ from x^l to x^m. This requires $O(|\pi(l, m)|)$ time, since a change of an element is done in constant time. Corresponding to the changes in $\underline{x}^m(k)$, $D^m_{-}(k)$ is obtained by appropriately modifying $D^l_{-}(k)$ in $O(|\pi(l, m)| \log n)$ time, since a deletion or a change of an element in $D^l_{-}(k)$ is done in $O(\log n)$ time.

Note that after type-2 $P^m(k)$ is computed (which will be explained in Section 5.3), the modified $\underline{x}^l(k)$ and $D^l_{-}(k)$ in $P^l(k)$ (which now become $\underline{x}^m(k)$ and $D^m_{-}(k)$, respectively) must be set back to the original form. This is also done in $O(|\pi(l, m)| \log n)$ time just by following $\pi(l, m)$ in reverse.

The construction of $D^m_{+}(k)$ from $D^l_{+}(k)$ is more involved and is discussed in Section 5.3.

5.3 DIRECTED FOREST T' AND CONSTRUCTION OF A TYPE-2 $P^m(k)$. To compute $D^m_{+}(k)$ efficiently, we introduce a directed forest T' defined as follows (see Figure 1). T' has nodes $x^1, x^2, \ldots, x^{k-1}$, and x^r is the father of x^t in T' (denoted by $r = f'(t)$) if $r = b(t)$ or if $b(t) = \emptyset$ and $r = b(q)$, where x^q is the closest ancestor of x^t in T with $b(q) \neq \emptyset$. Note that an x^p is a root in T' if and only if x^p is on the leftmost path in T. The path from x^u to x^v in T' (i.e., x^u is an ancestor of x^v in T') is denoted by $\pi'(u, v)$. Then we have by (5.4) and (5.5) that

$$I_b(l, m) \cup I_s(m) = \{i^*(p) \mid x^p \text{ is on } \pi'(u, v)\}, \tag{5.13}$$

where

$$v = \begin{cases} s(m) & \text{if } s(m) \neq \emptyset \quad \text{(see (5.2))}, \\ b(q) & \text{otherwise}, \end{cases}$$

$$q = \max\{p \mid x^p \text{ is on } \pi(l, m),\ p \neq l \text{ and } b(q) \neq \emptyset\};$$

$$u = \min\{r \mid x^r \text{ is an ancestor of } x^v \text{ in } T' \text{ and is a descendant of } x^l \text{ in } T\}.$$

The situation is illustrated in Figure 2. If x^u and x^v are not defined by this (i.e., the set on the right-hand side of $\underline{x}^q$ is empty), $I_b(l, m) \cup I_s(m) = \emptyset$ is concluded. x^v can be computed in $O(|\pi(l, m)|)$ time. Let

$$w = \begin{cases} \max\{q \mid x^q \text{ is on } \pi'(u, v) \text{ and is of type 1}\}, \\ u \qquad \text{if a type-1 node is not on } \pi'(u, v). \end{cases} \tag{5.14}$$

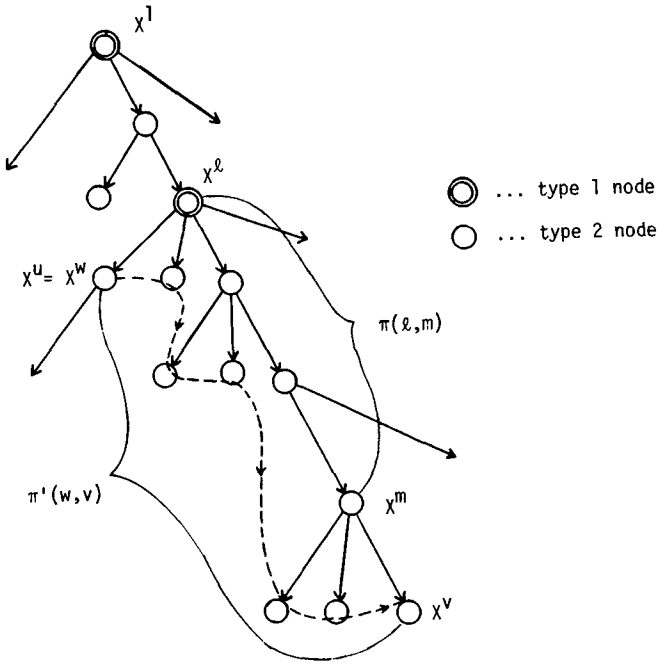

FIG 2 Relative positions of the nodes in T and T' used for the computation of type-2 $P^m(k)$

x^w is obtained in $O(|\pi'(w, v)|)$ time by following T' upward from x^v and checking at each node x^r whether $f(r) \geq l$ (i.e., x^r is a descendant of x^l in T) or not.

Case A. If x^w is of type 2 (i.e., $w = u$), $f(w) = f(u) = l$ holds. $\bar{x}^m(k)$ and $D_+^m(k)$ are obtained as follows. Starting with $\bar{x}^l(l)$ and $D_+^l(l)$ stored at $P^l(k-1)$, follow the path $\pi(l, m)$ from x^l $(=x^{f(w)})$ to x^m. At each vertex, first modify $\bar{x}^l(l)$ and $D_+^l(l)$ by (3.3) and (3.7), respectively, to take into account the generation of each son which is contained in $\pi'(w, v)$, and then move to its son on $\pi(l, m)$ while modifying $\bar{x}^l(l)$ and $D_+^l(l)$ by (3.3) and (3.9), respectively. When this process is completed, $\bar{x}^l(l)$ and $D_+^l(l)$ have been changed to $\bar{x}^m(k)$ and $D_+^m(k)$, respectively. The time required is $O(|\pi(f(w), m)| \log n + |\pi'(w, v)| \log n)$, because (3.3) is executed in constant time per vertex, and (3.7) or (3.9) is executed in $O(\log n)$ time per vertex.

Case B. If x^w is of type 1 (see Figure 3), we have

$$\bar{x}_i^m(k) = \begin{cases} x_i^{f(p^*)}, & \text{where } p^* = \max\{p \mid p(\neq w) \text{ is on } \pi'(w, v) \\ & \qquad \text{and } i^*(p) = i\}, \\ \bar{x}_i^{f(w)}(w), & \text{if } p^* \text{ is not defined,} \end{cases} \tag{5.15}$$

by (5.10) and (5.13). First $\bar{x}^{f(w)}(w)$ and $D_+^{f(w)}(w)$ are obtained in $O(\log n)$ time from $\bar{x}^w(w)$ and $D_+^w(w)$ by using (3.3), (3.7), and (3.9). Then $\bar{x}^m(k)$ and $D_+^m(k)$ are obtained in $O(|\pi(f(w), m)| \log n + |\pi'(w, v)| \log n)$ time from $\bar{x}^{f(w)}(w)$ and $D_+^{f(w)}(w)$ in a manner similar to case A.

Upon constructing $D_+^m(k)$ and $D_-^m(k)$ it is easily seen that c' and $[i', j']$ can be computed in $O(\log n)$ time, as noted in Section 3 following (3.6). Other data,

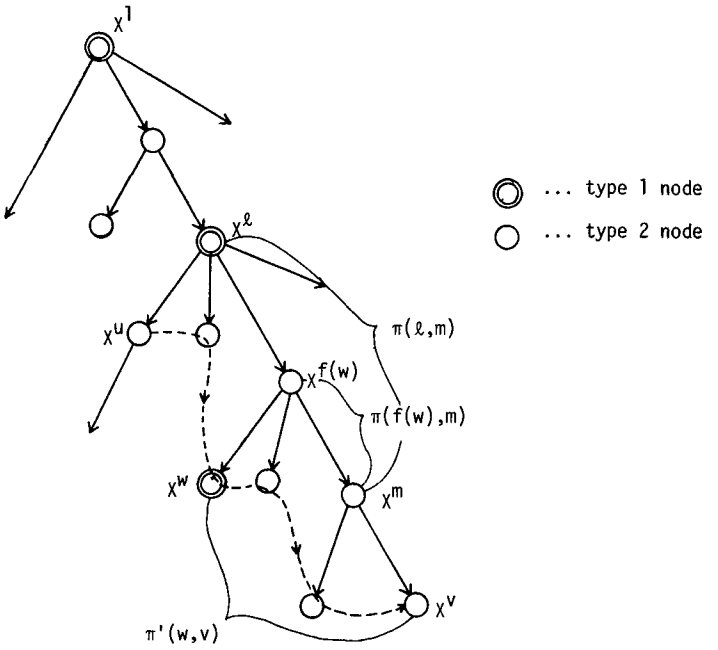

FIG 3 The relative positions of the nodes in T and T' when $\pi'(u, v)$ has a type-1 node.

$[\iota^*(m), j^*(m)], f(m), b(m)$, and $s(m)$, in type-2 $P^m(k)$ can be obtained in constant time as follows. For $m = k$,

$$\begin{aligned} &[\iota^*(k), j^*(k)] \leftarrow [\iota', j'] \qquad (\text{in } P^{m^*}(k-1)), \\ &f(k) \leftarrow m^*, \qquad f'(k) \leftarrow \begin{cases} s(m^*) & \text{if } s(m^*) \neq \varnothing, \\ f'(m^*) & \text{otherwise,} \end{cases} \\ &b(k) \leftarrow s(m^*), \qquad s(k) \leftarrow \varnothing. \end{aligned} \tag{5.16}$$

For $m = m^*$,

$$s(m^*) \leftarrow k, \tag{5.17}$$

and other data in type-2 $P^{m^*}(k-1)$ do not change. From the above discussion the next lemma follows.

LEMMA 5.1. *A type-2 $P^m(k)$ is constructed in*

$$O(|\pi(f(w), m)| \log n + |\pi'(w, v)| \log n)$$

time, where x^v, x^w, and x^m are defined as above. A type-2 $P^m(k)$ requires constant space.

5.4 SCHEME FOR CREATING A TYPE-1 DATA STRUCTURE. Let y be a prespecified positive integer not larger than K. Assume that $P^1(1)$ is of type 1 and all the other $P^m(k)$ are initially type 2. A $P^m(k)$, $m > 1$, is then altered from type 2 to type 1 if one of the following three conditions is satisfied.

(a) $|\pi(l, m)| = y$ and

$$(\text{the number of descendants of } x^m \text{ in } T) \geq y \tag{5.18}$$

hold, where x^l is the closest ancestor of x^m in T with type-1 data structure.

(b) x^m is on the leftmost path of T (i.e., a root in T'), and

$$\text{(the number of descendants of } x^m \text{ in } T') \geq y. \tag{5.19}$$

(c) x^m is not on the leftmost path of T and satisfies

$$|\pi'(w, m)| = y \quad \text{and} \quad (5.19), \tag{5.20}$$

where x^w is the closest ancestor of x^m in T' with type-1 data structure. (Note that if an x^m of (c) satisfies (5.19), there is an ancestor in T' satisfying the condition of (b); therefore the above x^w always exists.)

Consequently it always holds for any x^m that

$$|\pi(l, m)| \leq 2y, \tag{5.21}$$

$$|\pi'(w, m)| \leq 2y. \tag{5.22}$$

LEMMA 5.2 *The numbers of $P^m(k)$ of type* 1 *and* 2 *are $O(K/y)$ and $O(K)$, respectively.*

PROOF. The result for type 2 is obvious. We show that the number of type-1 $P^m(k)$ is $O(K/y)$. Let type 1a, 1b, and 1c denote the type-1 data structure generated by the above rules (a), (b), and (c), respectively. Consider $P^1(1)$ to be of type 1a for convenience. Define a set,

$$D(m) = \{x^p \mid x^p \text{ is a node not of type 1a whose closest ancestor in } T \text{ with type-1a data structure is } x^m\},$$

for a type-1a $P^m(k)$. By the generation rule (5.18), $D(m)$ contains at least $y - 1$ nodes. Since the $D(m)$'s for type 1a nodes x^m are mutually disjoint and their union is the set of all nodes not of type 1a, the number of type-1a nodes is $O(K/y)$. It is similarly shown that the number of type-1b or -1c nodes is $O(K/y)$. Thus the total number of type-1 nodes is $O(K/y)$. □

5.5 TIME NEEDED TO CREATE A TYPE-1 DATA STRUCTURE. We analyze here the time needed to create a type-1 data structure. It is not difficult to see that the conditions (5.18)–(5.20) for altering type 2 to type 1 can be detected in constant time by introducing some additional data such as the numbers of descendants of x^m in T and T', respectively. When it is decided to change type-2 $P^m(k)$ to type 1, let x^l be the closest type-1 ancestor of x^m in T, and let x^v and x^w be defined by (5.13) and (5.14) (see Figure 2). As discussed in Sections 5.2 and 5.3, x^m, $\underline{x}^m(k)$, $D^m_-(k)$, $\bar{x}^m(k)$, and $D^m_+(k)$ are obtained in $O(y \log n + n)$ time (including the time to copy these data), since $|\pi(f(w), m)| \leq |\pi(l, m)| \leq 2y$ and $|\pi'(w, v)| \leq 2y$ hold by (5.21) and (5.22). $\bar{x}^m(m)$ and $D^m_+(m)$ are also obtained in $O(y \log n + n)$ time from $\bar{x}^m(k)$ and $D^m_+(k)$ by a similar procedure. The other data can be obtained in constant time by (5.16). Consequently the following lemma is proved.

LEMMA 5.3. *The time required to alter type-*2 *$P^m(k)$ to type* 1 *is $O(y \log n + n)$.*

The results of Lemmas 5.1–5.3 are summarized in Table I.

5.6 TIME AND SPACE COMPLEXITY OF THE ENTIRE ALGORITHM. This section analyzes the time and space requirement of Algorithm KBS and Subroutine COMPBS implemented with the above data structure. First consider the time requirement of COMPBS for each iteration. The time required for lines 5–14 is reduced to $O(y \log n)$ by Lemma 5.1, (5.21), and (5.22). The other lines are not

TABLE I. THE TIME AND SPACE REQUIREMENT OF THE MODIFIED DATA STRUCTURE

Type	Time for a $P^m(k)$	Space for a $P^m(k)$	Total number
1	$O(n + y \log n)$	$O(n)$	$O(K/y)$
2	$O(y \log n)$	constant	$O(K)$

changed. Thus COMPBS requires $O(y \log n + \log K)$ time for every iteration. Since COMPBS is called $K - 1$ times, the total time is

$$O(K) \cdot O(y \log n + \log K) = O(Ky \log n + K \log K). \tag{5.23}$$

In addition to the time consumed by COMPBS, the modified KBS with the new data structure must take care of the alterations of some $P^m(k)$ from type 2 to type 1. The total time required for this process is

$$O(K/y) \cdot O(y \log n + n) = O(K \log n + Kn/y), \tag{5.24}$$

by Lemmas 5.2 and 5.3. Thus the modified KBS requires

$$O(Ky \log n + K \log K + Kn/y)$$

in total.

The space requirement for all $P^m(k)$ is also easily obtained from Table I:

$$O(K) + Q(K/y) \cdot O(n) = O(K + Kn/y).$$

Other space is obviously dominated by this.

Letting $y = \min(K, \sqrt{n/\log n})$ in the above discussion results in the next theorem.

THEOREM 5.4. *KBS (with subroutine COMPBS) can be implemented with* $O(T^* + K \log K + K\sqrt{n \log n})$ *time and* $O(K\sqrt{n \log n} + n)$ *space, where* T^* *is the time to obtain* x^1.

The $+n$ in the space complexity is added since at least $O(n)$ space is necessary even if $K < \sqrt{n/\log n}$. It is not added to the time complexity because T^* is at least $O(n)$.

REFERENCES

1. AHO, A.V., HOPCROFT, J.E., AND ULLMAN, J D. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, Mass., 1974
2. DREYFUS, S.E., AND LAW, A.M. *The Art and Theory of Dynamic Programming* Academic Press, New York, 1977
3. DUNSTAN, F D.J. An algorithm for solving a resource allocation problem. *Oper Res Q 28* (1977), 839–851.
4. EINBU, J.M. On Shih's incremental method in resource allocations. *Oper. Res Q. 28* (1977), 459–462
5. FOX, B. Discrete optimization via marginal analysis. *Manage. Sci. 13* (1966), 210–216.
6. FREDERICKSON, G.N., AND JOHNSON, D.B Optimal algorithms for generating quantile information in $X + Y$ and matrices with sorted columns. Tech. Rep CS-79-45, Computer Science Dep., Pennsylvania State Univ., College Park, Pa., 1979.
7. GALIL, Z., AND MEGIDDO, N A fast selection algorithm and the problem of optimum distribution of effort. *J. ACM 26*, 1 (Jan. 1979), 58–64
8. GROSS, O. A class of discrete-type minimization problems. Tech. Rep RM-1644-PR, The RAND Corporation, Santa Monica, Calif., 1956.
9. HARTLEY, R On an algorithm proposed by Shih *Oper. Res. Q 27* (1976), 389–390
10. JOHNSON, S.M. Sequential product planning over time at minimum cost. *Manage Sci. 3* (1957), 435–437.

11. KAO, E.P. On incremental analysis in resource allocation. *Oper. Res. Q. 27* (1976), 759–763
12. KARUSH, W On a class of minimum-cost problems. *Manage. Sci. 4* (1958), 136–153
13. KATOH, N., IBARAKI, T., AND MINE, H. A polynomial time algorithm for the resource allocation problem with convex objective function *J. Oper. Res. Soc. 30* (1979), 449–455.
14 KNUTH, D.E. *The Art of Computer Programming, Vol 3: Sorting and Searching.* Addison-Wesley, Reading, Mass , 1973.
15 KOOPMAN, B.O The optimum distribution of effort. *Oper. Res. 1* (1953), 52–63.
16. LAWLER, E.L. A procedure for computing the K best solutions to discrete optimization problems and its application to the shortest path problem. *Manage Sci. 18* (1972), 401–406.
17. MICHAELI, I., AND POLLATSCHECK, M.A. On some nonlinear knapsack problems *Ann. Discrete Math. 1* (1977), 403–414
18. MJELDE, K.M. The optimality of an incremental solution of a problem related to distribution of effort. *Oper Res Q. 26* (1975), 867–870.
19. PROLL, L.G. Marginal analysis in resource allocations. *Oper Res. Q. 27* (1976), 765–767.
20. SAATY, T L *Mathematical Methods of Operations Research* McGraw-Hill, New York. 1959
21. SHIH, W. A new application of incremental analysis in resource allocations *Oper. Res. Q. 25* (1974), 587–597.
22. SHIH, W. A branch and bound procedure for a class of discrete resource allocation problems with several constraints. *Oper. Res. Q 28* (1977), 439–451.
23 SIVAZLIAN, B D., AND STANFEL, L.E. *Optimization Techniques in Operations Research* Prentice-Hall, Englewood Cliffs, N.J., 1975, pp. 432–434.
24. VEINOTT, A.F Production planning with convex costs: A parametric study *Manage. Sci. 10* (1964), 441–460.
25. VEINOTT, A.F. The status of mathematical inventory theory. *Manage Sci. 12* (1966), 745–777
26. WAGNER, H M *Principles of Operations Research.* Prentice-Hall, Englewood Cliffs, N J , 1969.
27. WEINSTEIN, I.J., AND YU, O.S. Comment on an integer maximization problem. *Oper. Res. 21* (1973), 648–649

RECEIVED MAY 1979; REVISED JULY 1980; ACCEPTED JULY 1980