Journal of the Operations Research
Society of Japan
Vol. 23, No. 1, March 1980

© 1980 The Operations Research Society of Japan

ALGORITHMS FOR OPTIMAL ALLOCATION PROBLEMS HAVING QUADRATIC OBJECTIVE FUNCTION

Azuma Ohuchi and Ikuo Kaji
Hokkaido University

(Received February 21, 1979; Revised October 25, 1979)

Abstract In this paper, an optimal allocation problem (APQ) with a quadratic objective function, a total resource constraint and an upper and lower bound constraint is considered. The APQ is a very basic and simple model but it can serve as a subproblem in the solution of the generalized allocation problem.

Applying the Lagrange relaxation method, an explicit expression of the dual function associated with the APQ and an equation which the optimal dual variable must satisfy are derived first. Then, some properties of the equation are discussed. Finally, three algortihms for solving the equation are proposed, and some computational results for the APQ are given. These results reveal the effectiveness of the algorithm.

1. Introduction

The allocation problem to be considered in this paper, which we shall call the APQ, is to minimize the quadratic objective function

$$(1) \qquad F(x) = \sum_{i\in M} f_i(x_i) = \sum_{i\in M} (a_i+b_i x_i)x_i$$

subject to

$$(2) \qquad \sum_{i\in M} x_i = B,$$

where

$M = \{1,2,\ldots,m\}$, $x = (x_1,\ldots,x_m)$ and $x_i \in C_i = [l_i,d_i]$ with $0<l_i<d_i$, $b_i>0$ for all $i\in M$.

The APQ is a very basic and simple model but it can serve as a subproblem in the solution of the generalized allocation or transportation problems which have quadratic objective functions. From the standpoint of the mathematical programming theory, the APQ is a strictly convex separable programming problem and is a special class of quadratic programming problem: Therefore, its global optimality is guaranteed.

Our paper was directly influenced by Takahashi's "A method for solving network transportation problems with quadratic cost functions."[6] Takahashi proposed the algorithm based on the separation principle in which the APQ plays a central role. However, he did not give the systematic algorithm for the APQ, and his variables were not restricted from the upper bound, that is, only non-negative constraints were considered. Our treatment is an extension upon which the upper bound constraint on the variables is imposed. This constraint is essential for practical applications.

Other problems in which the APQ plays an important role are the resource allocation problems with integrality condition on the variables.[5] One of the efficient algorithms for this problem requires to solve the APQ as a relaxed subproblem which is obtained from the problem by dropping the integrality condition on the variables.

Furthermore, the APQ itself is an important problem in electrical power network systems. The problem of how to dispatch the load for each generator to minimize the generation cost under the constraint of each generator output limitation is called the economic load dispatch problem (ELD).[2] The ELD is the same as the APQ and is one of the most basic problem for power dispatch operations.

As mentioned above, the APQ is a worthwhile problem to study in detail. In this paper, we would like to consider the algorithm for solving the APQ. Although we can apply general quadratic programming algorithms to the APQ, we will apply the Lagrange relaxation method to the APQ. We can take the advantage of the problem structure, since, there is only one complicating constraint in the sense that the problem would be much easier if it were not present.[3] For example, Wolfe's method for the quadratic programming problems uses the simplex method to solve the system of equations with complementary conditions which represents the Kuhn-Tucker's conditions as applied to quadratic programming problems.[4] But our approach should lead us to a simple result that we only solve an equation with one variable. So even small computers may be used to implement the algorithms.

In section 2, we obtain an explicit expression of the dual function associated with the APQ by applying the Lagrange relaxation method. In section 3, we derive some properties of the derivative of the dual function. Algorithm using these properties are described in section 4. Remarks for implementing the algorithms and computational results are shown in section 5 and section 6, respectively.

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

2. Preliminary

In this section the Lagrange relaxation method is applied to the APQ as preliminary to the following sections

Define the Lagrange function $L(x,\lambda)$ associated with the APQ and its dual function $D(\lambda)$ as follows.

$$L(x,\lambda) = \sum_{i\in M} f_i(x_i)+\lambda(B-\sum_{i\in M} x_i) \tag{3}$$

or

$$L(x,\lambda) = \sum_{i\in M} \{(a_i-\lambda)x_i+b_i x_i^2\}+\lambda B, \tag{4}$$

$$D(\lambda) = \min_{x\in C=\Pi C_i} L(x,\lambda). \tag{5}$$

Let $\bar{\lambda}$ denote λ maximizing $D(\lambda)$, then an $x\in C$ minimizing (4) for $\lambda = \bar{\lambda}$ is an optimal solution for the problem, and is denoted $\bar{x}$. It can be shown

$$D(\bar{\lambda}) = F(\bar{x}) \tag{6}$$

unless $D(\bar{\lambda}) = \infty$, in which case the problem is infeasible. Feasibility, however, can be easily checked by whether $\sum_{i\in M} l_i \leqq B \leqq \sum_{i\in M} d_i$ holds or not.

Since the Lagrange function $L(x,\lambda)$ is separable, the $x\in C$ minimizing the $L(x,\lambda)$ is given by

$$x_i(\lambda) = <(\lambda-a_i)/2b_i> \tag{7}$$

and the dual function $D(\lambda)$ is written as

$$D(\lambda) = \sum_{i\in M} \{(a_i-\lambda)x_i(\lambda)+b_i x_i(\lambda)^2\}+\lambda B, \tag{8}$$

where brackets symbol $<Y_i(\lambda)>$ means

$$Y_i(\lambda) = \min\{\max(l_i, Y_i(\lambda)), d_i\} \tag{9}$$

$$= \begin{cases} l_i, & Y_i(\lambda)\leqq l_i \\ Y_i(\lambda), & l_i \leqq Y_i(\lambda) \leqq d_i \\ d_i, & Y_i(\lambda)\geqq d_i \end{cases}$$

The algorithm for the APQ is summarized as follows:

begin

find a maximal point $\bar{\lambda}$ of $D(\lambda)$;

substitute $\bar{\lambda}$ into (7) to obtain the optimal $\bar{x}$;

end

The main part of the algorithm is to find the maximal point of $D(\lambda)$. We shall show some properties for the derivative of $D(\lambda)$ in the next section.

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

3. Properties of derivative of $D(\lambda)$

Let $\alpha_i = a_i+2b_i l_i$, $\beta_i = a_i+2b_i d_i$ for all $i\in M$ and $\alpha = \max_i \alpha_i$, $\beta = \min_i \beta_i$, and define a new sequence $\{\gamma_j \mid \gamma_j = \alpha_i \text{ or } \beta_i\}$ out of α's and β's such that $\gamma_1<\gamma_2<\ldots<\gamma_p$, $(p\leqq 2m)$ and if one or more α_i and β_j are same values, they are considered as identical. Note that since $0\leqq l_i<d_i$, $\alpha_i<\beta_i$ holds for all i, and that $p = 2m$ if all the α's and β's are different.

Put $\Gamma_j = [\gamma_j,\gamma_{j+1}]$, then there are three possible cases in relation with $[\alpha_i,\beta_i]$.

$$[\alpha_i,\beta_i]\cap\Gamma_j = \begin{cases} \Gamma_j \\ \{\gamma_j\} \text{ or } \{\gamma_{j+1}\} \\ \phi \end{cases} \tag{10}$$

We define the index set Λ_j associated with Γ_j,

$$\Lambda_j = \{i \mid [\alpha_i,\beta_i]\supset\Gamma_j\}. \tag{11}$$

Lemma 1.

$$\Lambda_{j+1} = \{\Lambda_j\cup I_\alpha\}-I_\beta = \{\Lambda_j-I_\beta\}\cup I_\alpha,\ j = 1,\ldots,p-1, \tag{12}$$

or

$$\Lambda_j = \{\Lambda_{j+1}-I_\alpha\}\cup I_\beta = \{\Lambda_{j+1}\cup I_\beta\}-I_\alpha,\ j = p-1,\ldots,1. \tag{13}$$

where

$$I_\alpha = \{i \mid \alpha_i = \gamma_{j+1}\} = \Lambda_j^c\cap\Lambda_{j+1},$$
$$I_\beta = \{i \mid \beta_i = \gamma_{j+1}\} = \Lambda_j\cap\Lambda_{j+1}^c.$$

Proof: Note that $\Gamma_j = [\gamma_j,\gamma_{j+1}]$ and γ_{j+1} corresponds to some α_k or/and some β_h, $k\neq h$. If $i\in I_\alpha$ then $i\in\Lambda_j^c\cap\Lambda_{j+1}$. If $i\in I_\beta$ then $i\in\Lambda_j\cap\Lambda_{j+1}^c$. Conversely, if $i\in\Lambda_j^c\cap\Lambda_{j+1}$, then $\alpha_i\leqq\gamma_{j+1}$ and $\alpha_i\geqq\gamma_{j+1}$. Hence, $i\in I_\alpha$. If $i\in\Lambda_j\cap\Lambda_{j+1}^c$, then $\beta_i\leqq\gamma_{j+1}$ and $\beta_i\geqq\gamma_{j+1}$. Hence $i\in I_\beta$. Therefore, $I_\alpha = \Lambda_j^c\cap\Lambda_{j+1}$ and $I_\beta = \Lambda_j\cap\Lambda_{j+1}^c$.

On the other hand,

$$\begin{aligned}\Lambda_{j+1} &= \{\Lambda_j\cap\Lambda_{j+1}\}\cup\{\Lambda_j^c\cap\Lambda_{j+1}\} \\ &= \{\Lambda_j-\Lambda_j\cap\Lambda_{j+1}^c\}\cup\{\Lambda_j^c\cap\Lambda_{j+1}\} \\ &= \{\Lambda_j-I_\beta\}\cup I_\alpha \\ &= \{\Lambda_j\ I_\alpha\}-I_\beta. \quad (\because I_\alpha\cap I_\beta = \phi)\end{aligned}$$

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

and

$$\begin{aligned}\Lambda_j &= \{\Lambda_j \cap \Lambda_{j+1}\} \cup \{\Lambda_j \cap \Lambda_{j+1}^c\} \\ &= \{\Lambda_{j+1} - \Lambda_j^c \cap \Lambda_{j+1}\} \cup \{\Lambda_j \cap \Lambda_{j+1}^c\} \\ &= \{\Lambda_{j+1} - I_\alpha\} \cup I_\beta \\ &= \{\Lambda_{j+1} \cup I_\beta\} - I_\alpha. \quad (\because I_\alpha \cap I_\beta = \phi)\end{aligned}$$ Q.E.D.

Our next task is to determine the index set Λ_j, that is, to find all the i's for which $\lambda \varepsilon [\alpha_i, \beta_i]$ for a given λ. One method is to compute all α_i and β_i, then to see whether $\lambda \varepsilon [\alpha_i, \beta_i]$ holds for each i. The second is to compute $x_i^0(\lambda) = (\lambda - a_i)/2b_i$ for λ and test $1_i \leq x_i^0(\lambda) \leq d_i$. The second takes advantage of the fact that $x_i(\lambda)$ defined in (7) is a linear function of λ on $[\alpha_i, \beta_i]$. It is obvious that the index set obtained above is the Λ_j associated with Γ_j which contains λ.

From the definition of bracket symbol and (8), existense of the partial derivative of $D(\lambda)$ is guaranteed. ([3],[6],[7]) Then the maximal point $\bar{\lambda}$ of $D(\lambda)$ can be obtained as a solution to the equation $\partial D(\lambda)/\partial \lambda = 0$,

$$(14) \qquad \partial D(\lambda)/\partial \lambda = -\sum_{i \varepsilon M} <(\lambda - a_i)/2b_i> + B = 0.$$

Thus the following equation (15) is obtained on referring to (7).

$$(15) \qquad \sum_{i \varepsilon M} x_i(\lambda) = \sum_{i \varepsilon M} <(\lambda - a_i)/2b_i> = B.$$

Let $\delta(\lambda)$ be the left hand side of (15) and define the linear function $\delta^0(\lambda)$ by deleting the brackets in the right hand of (15), i.e.,

$$(16) \qquad \delta(\lambda) = \sum_{i \varepsilon M} x_i(\lambda) = \sum_{i \varepsilon M} <(\lambda - a_i)/2b_i>$$

$$(17) \qquad \delta^0(\lambda) = \sum_{i \varepsilon M} x_i^0(\lambda) = \sum_{i \varepsilon M} (\lambda - a_i)/2b_i = k_0 \lambda + c_0,$$

where

$$k_0 = \sum_{i \varepsilon M} 1/2b_i, \quad c_0 = -\sum_{i \varepsilon M} a_i/2b_i.$$

As it is easily seen that the function $\delta(\lambda)$ is a non-decreasing line-segment function with vertices $\{\gamma_j, \delta_j\}$, we write the equation of the line-segment on Γ_j as $y_j = k_j \lambda + c_j$. k_j and c_j are uniquely determined from the fact that y_j passes through the points (γ_j, δ_j) and $(\gamma_{j+1}, \delta_{j+1})$, that is,

$$(18) \qquad \begin{aligned} k_j &= (\delta_{j+1} - \delta_j)/(\delta_{j+1} - \delta_j), \\ c_j &= \delta_j - \gamma_j (\delta_{j+1} - \delta_j)/(\gamma_{j+1} - \gamma_j), \end{aligned}$$

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

where abbreviation $\delta_j = \delta(\gamma_j)$ is used.

On the other hand, k_j and c_j are determined by the index sets Λ_j, Λ_α and Λ_β, i.e.,

$$
\begin{aligned}
k_j &= \sum_{i\in\Lambda_j} 1/2b_i, \\
c_j &= -\sum_{i\in\Lambda_j} a_i/2b_i + \sum_{i\in\Lambda_\alpha} 1_i + \sum_{i\in\Lambda_\beta} d_i,
\end{aligned} \tag{19}
$$

where $\Lambda_\alpha = \{h \mid \alpha_h \geq \gamma_{j+1}\}$ and $\Lambda_\beta = \{h \mid \beta_h \leq \gamma_j\}$.

Note that $\Lambda_j \cap \Lambda_\alpha \cap \Lambda_\beta = \phi$, $\Lambda_j \cup \Lambda_\alpha \cup \Lambda_\beta = M$.

Lemma 2.

$$k_0 > \max_{1<j<p-1} k_j \tag{20}$$

Proof

$k_j = \sum_{i\in\Lambda_j} 1/2b_i$ and Λ_j is a subset of $M = \{1,\ldots,m\}$, and $b_i > 0$, we have $k_0 > k_j$, $j = 1,\ldots,p-1$. Hence, $k_0 > \max k_j$. Q.E.D.

Define $\phi(\lambda)$ as follows.

$$\phi(\lambda) = \delta^0(\lambda) - \delta(\lambda) = \sum_{i\in M} \{(\lambda-a_i)/2b_i - <(\lambda-a_i)/2b_i>\}. \tag{21}$$

Then we have Lemma 3. (Illustrated in Fig. 1)

Lemma 3.

If $\alpha<\beta$, then there exists a single subinterval $\Gamma_k = [\alpha,\beta]$ such that $\phi(\lambda) = 0$, for all $\lambda\in\Gamma_k$. If $\alpha\geq\beta$, then there exists a single point λ such that $\phi(\lambda) = 0$, $\beta\leq\lambda\leq\alpha$.

Proof:

We first note that $\phi(\lambda)$ is also a non-decreasing continuous line-segment function on $-\infty<\lambda<+\infty$ since $k_0 - k_j \geq 0$ for all subinterval Γ_j from Lemma 2 and both $\delta^0(\lambda)$ and $\delta(\lambda)$ are continuous. It is also easily seen that $\phi(\lambda)<0$ for some $\lambda<\min_i \alpha_i$ and $\phi(\lambda)>0$ for some $\lambda>\max_i \beta_i$.

Applying the intermediate value theorem we can obtain the result that there exists at least a point at which $\phi(\lambda) = 0$.

The case $\alpha<\beta$.

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

From the definitions of α,β and the assumption $\alpha<\beta$, we can write all subintervals as $\Gamma_j = [\alpha_i,\alpha_h]$ or $[\beta_i,\beta_1]$ except $\Gamma_k = [\alpha,\beta]$. By noting that $\alpha_i \leqq \alpha < \beta \leqq \beta_i$, we obtain $\Lambda_k = M$ bacause $\Gamma_k = [\alpha,\beta] \subset [\alpha_i,\beta_i]$ for all i.

From (21), we have $\phi(\lambda) = 0$ for $\lambda \varepsilon [\alpha,\beta]$.

On all other subintervals Γ_j, $j \neq k$,

$$k_j = \sum_{i\varepsilon\Lambda_j} 1/2b_i < \sum_{i\varepsilon M} 1/2b_i = k_0.$$

Therefore, the subinterval $[\alpha,\beta]$ is the only subinterval where $\phi(\lambda) = 0$ occurs.

The case $\alpha \geqq \beta$.

To prove the latter part of Lemma 3, we assume that $\alpha = \alpha_k$, $\beta = \beta_h$. Then,

$$[\alpha_n,\beta_n] \cap [\alpha_k,\beta_k] \subset \{\alpha\} \text{ or } \{\beta\}.$$

Each Γ_j is either contained in $[\alpha_h,\beta_h]$ or not. If the former is the case,

$$\Gamma_j \cap [\alpha_k,\beta_k] \subset [\alpha_h,\beta_h] \cap [\alpha_k,\beta_k] \subset \{\alpha\} \text{ or } \{\beta\},$$

that is, $\Gamma_j \not\subset [\alpha_k,\beta_k]$. If the latter is the case, it is trivial that $\Gamma_j \not\subset [\alpha_h,\beta_h]$. Therefore, there exists at least an interval $[\alpha_i,\beta_i]$ such that $\Gamma_j \not\subset [\alpha_i,\beta_i]$ for any j. Hence, $|\Lambda_j| < |M|$ for all j, which means that there exists no interval Γ_j such that $\phi(\lambda) \equiv 0$ for $\lambda \varepsilon \Gamma_j$ by the non-decreasing property of $\phi(\lambda)$. Thus we conclude that there exists only one point λ^0 where $\phi(\lambda^0) = 0$.

Such a point λ^0 lies in the interval $[\beta,\alpha]$ if $\phi(\beta) \leqq 0$ and $\phi(\alpha) \geqq 0$, which will be shown to be the case in the following.

Define index sets $N(\lambda)$ and $P(\lambda)$ as

$$N(\lambda) = \{i \mid \lambda \leqq \alpha_i\},$$

$$P(\lambda) = \{i \mid \lambda \geqq \beta_i\}.$$

We then evaluage the i-th term of $\phi(\lambda)$ for $\lambda \varepsilon \Gamma_j$.

$$(\lambda-a_i)/2b_i - <(\lambda-a_i)/2b_i> = \begin{cases} (\lambda-a_i)/2b_i - l_i, & \text{if } i\varepsilon N(\lambda) \ldots (a) \\ (\lambda-a_i)/2b_i - (\lambda-a_i)/2b_i = 0, & \text{if } i\varepsilon\Lambda_j \\ (\lambda-a_i)/2b_i - d_i, & \text{if } i\varepsilon P(\lambda) \ldots (b) \end{cases}$$

In particular, if $\lambda = \beta$, then $N(\beta) = \{i \mid \beta \leqq \alpha_i\}$ and $P(\beta) = \{i \mid \beta \geqq \beta_i\}$, and from the definition of β, $\beta = \min \beta_i$, $P(\beta)$ has only elements $\beta_i = \beta$, and for which (b) vanishes. On the other hand, $N(\beta)$ is not empty since $\alpha \geqq \beta$ from the assumption, furthermore,

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

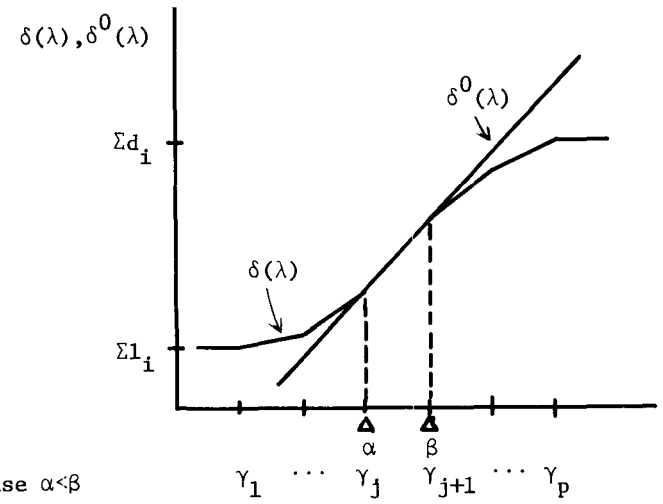
$(\beta-a_i)/2b_i-l_i \leqq 0$, for $i\in N(\beta)$,

the equality holds only for $\alpha_i = \beta$. Thus we have

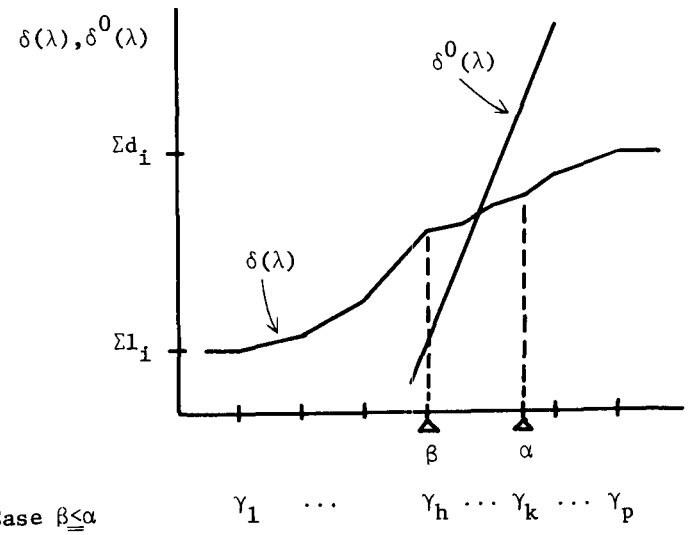
$$\phi(\beta) = \sum_{i\in N(\beta)} \{(\beta-a_i)/2b_i-l_i\} \leqq 0,$$

the equality holds only if $\alpha_i = \beta$ for all $i\in N(\beta)$.

Similarly, it can be proved that $\phi(\alpha)\geqq 0$ and the equality holds only if $\beta_i = \alpha$ for all $i\in P(\alpha)$. Q.E.D.



(a) Case $\alpha<\beta$



(b) Case $\beta\leqq\alpha$

Fig. 1 Relation of $\delta(\lambda)$ and $\delta^0(\lambda)$

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

Lemma 4.

$\delta(\lambda)$ is a monotone increasing line-segment convex function for $\gamma_1 \leqq \lambda \leqq \beta$ and $\delta(\lambda)$ is a monotone increasing line-segment concave function for $\alpha \leqq \lambda \leqq \gamma_p$.

Proof:

Assume that $\alpha = \gamma_k$, $\beta = \gamma_h$, than we can suppose that γ_j corresponds to

$$\gamma_j = \alpha_{i_j}, \quad j = 1,\ldots,h-1$$
$$\gamma_{k+j} = \beta_{n_j}, \quad j = 1,\ldots,p-k.$$

Then each index set Λ_j, $j = 1,\ldots,h-1$ and $j = k,\ldots,p-1$ are determined as follows:

Set $\Lambda_0 = \{\phi\}$, then from (12) and (13) in Lemma 1,

$$\Lambda_j = \Lambda_{j-1} \cup I_\alpha, \quad j = 1,\ldots h-1$$

and

$$\Lambda_{k+j} = \Lambda_{k+j-1} - I_\beta, \quad j = 1,\ldots,p-k-1.$$

Hence,

$$|\Lambda_1| < |\Lambda_2| < \ldots < |\Lambda_{h-1}|$$

and

$$|\Lambda_k| > |\Lambda_{k+1}| > \ldots > |\Lambda_{p-1}|.$$

Therefore, from (19),

$$k_1 < k_2 < \ldots < k_{h-1}$$

and

$$k_k > k_{k+1} > \ldots > k_{p-1}.$$ Q.E.D.

4. Algorithms

We shall propose three algorithms for finding the optimal λ; (i) polynominal approximations (PA), (ii) sequential search (SS) and (iii) hybrid of (i) and (ii) (HB). All algorithms are described by PASCAL-like language.

4.1 Algorithm PA

By Lemma 4 the function $\delta(\lambda)$ can be generally divided into three parts, convex part for $\lambda < \beta$, concave part for $\lambda > \alpha$ and neither convex nor concave part for $\beta < \lambda < \alpha$. Theoretically, any convenient convex (cancave or linear) function

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

may be available as an approximation function for the convex (concave or neither convex nor concave) part. For paractical purpose a function which is a monotone increasing part of convex (concave) quadratic function for $\lambda<\beta$ (for $\lambda>\alpha$) is quite sufficient.

When a linear approximation function is used for all λ, the algorithm is just the algorithm so called regula falsi method which computes the root of monotone functions.

If quadratic functions are used, the approximating function can be determined by the information that it passes through the two points $(\lambda^0,\delta_0)$and(λ^1,δ_1) and its tangent at λ^0 or λ^1. We can use the line-segment on the subinterval containing λ^0 or λ^1 as the tangent of the approximating function. As noted in section 3, this line-segment can be determined by Λ_j, which can be determined by $x_i^0(\lambda)$, and $x_i^0(\lambda)$ have been already computed when δ_0 or δ_1 were evaluated.

```
begin
    compute δ⁰(λ) = B and let λ⁰ be the solution;
    if δ(λ⁰)≠B then
    begin
        compute α = max αᵢ and β = min βᵢ;
        if λ⁰<β then compute the solution by using convex approximating
                    functions;
        else
        begin
            if λ⁰>α then compute the solution by using concave approximating
                        functions;
            else compute the solution by using linear approximating functions;
        end
    end
end
```

4.2 Algorithm SS

The initial λ, λ^0, can be easily obtained from the solution to the equation $\delta^0(\lambda) = B$. Then the subinterval Γ_j, $\Gamma_j \ni \lambda^0$, and index set Λ_j, $\Lambda_j = \{i \mid [\alpha_i,\beta_i]\} \supset \Gamma_j$, can be determined, furthermore, Γ_{j+1} and Λ_{j+1}, Γ_{j+2} and Λ_{j+2},..., or Γ_{j-1} and Λ_{j-1}, Γ_{j-2} and Λ_{j-2},... are sequentially determined by Lemma 1. If we find the interval $[\delta_k,\delta_{k+1}] \ni B$, then the optimal λ can be computed form the equation $y_k = k_k\lambda+c_k = B$, where y_k is the equation of the

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

line-segment on the subinterval Γ_k.

begin

given initial λ^0;

determine the subinterval Γ_j and corresponding index set Λ_j such that $\Gamma_j \ni \lambda^0$;

if $\delta(\lambda^0) < B$ then search Γ_k by increasing k until $[\delta_k, \delta_{k+1}] \ni B$ is found;

else search Γ_k by decreasing k until $[\delta_k, \delta_{k+1}] \ni B$ is found;

solve the equation $k_k\lambda + c_k = B$;

end

4.3 Algorithm HB

The algorithm PA is efficient for the global estimation of the optimal λ. On the other hand, the algorithm SS is efficient for the local computation of the optimal λ. The hybrid of these two leads to a very efficient algorithm. The algorithm HB performs the algorithm PA r times, then uses the algorithm SS by starting from λ^0 which has been obtained by the algorithm PA.

begin

Execute algorithm PA r times and let λ^0 be the current solution;

Execute algorithm SS starting from λ^0;

end

5. Remarks for implementing the algorithms

We shall point out a few important points about implementing the above algorithms.

(1) Algorithm PA

We shall consider the case of using quadratic functions as an approximating function. First, we will show how to obtain the points through which the initial quadratic approximating function passes through. Let λ^0 be the root of the equation $\delta^0(\lambda) = B$, then (λ^0, δ_0) can be used as one of the points. The other point can be determined by the non-decreasing property of $\delta(\lambda)$ as follows:

if $\delta(\lambda^0) > B$ then $\lambda^1 := \min_i \alpha_i$ else $\lambda^1 = \max_i \beta_i$;

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

Secondly, we consider the roots of the quadratic equation. If we write the quadratic approximating function as $\bar{\delta}(\lambda) = p\lambda^2+q\lambda+r$, the equation $\bar{\delta}(\lambda) = B$ has two roots, say λ^a and λ^b. However, it is sufficient to take only one root by the non-decreasing property of $\delta(\lambda)$ as follows:

```
begin
  if δ(λ^0)<B then
                begin
                  if p>0 then λ: = max{λ^a,λ^b};
                  if p<0 then λ: = min{λ^a,λ^b};
                end
  if δ(λ^0)>B then
                begin
                  if p>0 then λ: = min{λ^a,λ^b};
                  if p<0 then λ: = max{λ^a,λ^b};
                end
end
```

(2) Algorithm SS

An important part of algorithm SS is to determine the linear functions on the subintervals Γ_j's by Lemma 1 and eq. (12) or (13). That is, when searching for subintervals $\Gamma_j \to \Gamma_{j+1} \to \ldots$ or $\Gamma_j \to \Gamma_{j-1} \to \ldots$, we must determine coefficients k_j and c_j of the linear function $y_j = k_j\lambda+c_j$ define on the subinterval Γ_j. As noted in section 4.1, k_j and c_j can be determined by the index sets Λ_j, Λ_α and Λ_β in eq. (19). We can reduce the amount of computer storage required to store the index sets by providing three lists L_1, L_2 and L_3 consisting of m-bits corresponding to the index sets Λ_j, Λ_α and Λ_β, respectively. We set i-th bit in the list L_j as follows:
if index i is in Λ_j, then i-bit of L_j is set to 1, otherwise 0. This requires only 3m bits to store the index sets. Furthermore, we can compute one list from two other lists by using the property that the index sets $\Lambda_j, \Lambda_\alpha$ and Λ_3 are the mutually exclusive and collectively exhaustive sets of M. For example, we can get L_3 by taking a compliment of the Boolean sum of L_1 and L_2; i.e., $L_3 = \overline{L_1 \oplus L_2}$, where $\oplus$ means m-bitswise Boolean sum and $\overline{\quad}$ is a compliment.

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

6. Computational results

In order to test the relative efficiency of the algorithms, 200 randomly generated problems were solved by each algorithm. The results are shown in Figure 2 and Table 1. Figure 2 shows the computer times in milli-seconds and Table 1 shows the number of searched subintervals in algorithms SS and HB. The following observations were made on their performance:

1. The computer times of algorithm PA and HB appeared to be proportional to the number of m variables. On the other hand, algorithm SS looked like the order of logarithm computer times because it contained a part that arranged all α_i's and β_i's in increasing order to make the sequence $\{\gamma_j\}$. This sorting needed an average of $O(2mcLog\ 2m)$, $c>0$, comparisons.[1] However, when we already had the sequence $\{\gamma_j\}$ and only the total amount of resource B was changed, only Log 2m comparisons by the binary search algorithm were needed in order to find the interval $[\delta_k, \delta_{k+1}]$ such that $[\delta_k, \delta_{k+1}] \ni B$

2. It is clear from Table 1 that algorithm HB needs much less interval search than algorithm SS. Generally speaking, as the number of variables increased, the function $\delta(\lambda)$ became a smooth logistic curve rather than line-segments and the quadratic function gave a fairly good approximate solution in one or two iterations.

Summarizing the comparisons of the algorithms, it can be seen that algorithm HB is efficient when the APQ must be solved for many data sets of coefficients or constants. On the other hand, algorithm SS is useful for problems when only the right hand quantity B is changed.

Table 1

Number of searched intervals

Algorithms	Number of variables									
	10	20	30	40	50	60	70	80	90	100
SS	3	4	6	9	11	14	16	19	21	21
HB	1	1	1	2	3	3	3	3	3	4

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

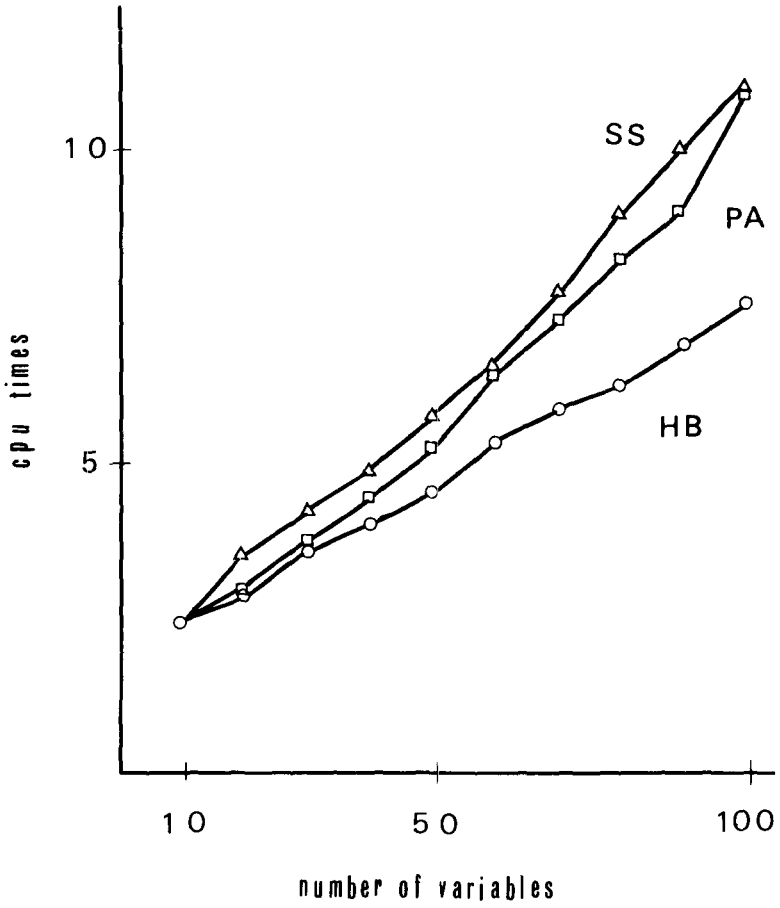


Fig. 2 CPU times (CPU milliseconds on a FACOM 230-75)
Results are the average of twenty problems for each m

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

7. Conclusion

We have discussed some properties of the dual function $D(\lambda)$ associated with the APQ and have given three algorithms for finding the optimal λ with remarks for implementing the algorithms.

The important properties are that the root of the equation $\phi(\lambda) = 0$ is characterized by the magnitude of α and β, and that the function $\delta(\lambda)$ can be divided into three parts: convex for $\lambda<\beta$; concave for $\lambda>\alpha$; and neither convex nor concave for $\beta<\lambda<\alpha$.

Our computational results show that algorithm HB is very efficient and suggest that this algorithm can be applied to more complicated allocation type problems, for example, to transportation network problems having the quadratic objective function.

Some properties obtained here will be applied to general differentiable strictly convex objective function cases and reported on in a subsequent paper.

8. Acknowledgements

The authors wish to thank Mr. Akihiko Takiguchi for his help in executing the numerical computations and Professor Masanao Kitamura for his critical reading of the paper.

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

References

[1] Aho, A.V., J.E. Hopcroft and J.D. Ullman: The design and analysis of computer algorithms. Johnson-Wesley, (1974).

[2] Elgard, O.I.: Electric energy systems theory. McGraw-hill, (1971).

[3] Geoffrion, A.M.: Elements of large-scale mathematical programming. Management Science, Vol. 16, No. 11 (1970), 652-675.

[4] Hadley, G.: Nonlinear and dynamic programming. Addison-wesley, (1964).

[5] Katoh, N., Ibaraki, T. and Mine, H.: A polynomial time algorithm for the resource allocation problem with convex objective function. Journal of the Operational Research, Vol. 30 (1979), 449-455.

[6] Takahashi, I.: A method for solving network transportation problems with quadratic cost functions. Bulletin of the Institute for research in Productivity, Waseda University, Vol. 1, No. 1 (1970)

[7] _____ : Variable separation principle for mathematical programming. Journal of the Operations Research Society of Japan, Vol. 6, No. 1 (1964), 82-105.

Azuma Ohuchi: Department of Electrical Engineering, Faculty of Engineering, Hokkaido University, Kita 13 Jyo, Nishi 8 Chome, Kita-ku, Sapporo, Japan.

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.