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 algorithms 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

\[ F(x) = \sum_{i \in M} f_i(x_i) = \sum_{i \in M} (a_i + b_i x_i) x_i \]

subject to

\[ \sum_{i \in M} x_i = B, \]

where

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

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.
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.

\[
L(x, \lambda) = \sum_{i \in M} f_i(x_i) + \lambda(B - \sum_{i \in M} x_i)
\]

or

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

\[
D(\lambda) = \min_{x \in C} L(x, \lambda).
\]

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

\[
D(\lambda) = F(\bar{x})
\]

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

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

\[
x_i(\lambda) = <(\lambda - a_i)/2b_i>.
\]

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

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

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

\[
<Y_i(\lambda)> = \begin{cases} 
1_i, \quad Y_i(\lambda) \leq 1_i \\
Y_i(\lambda), \quad 1_i < Y_i(\lambda) \leq d_i \\
d_i, \quad Y_i(\lambda) > d_i
\end{cases}
\]

The algorithm for the APQ is summarized as follows:

\begin{itemize}
  \item find a maximal point \( \lambda \) of \( D(\lambda) \);
  \item substitute \( \lambda \) into (7) to obtain the optimal \( \bar{x} \);
\end{itemize}

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

Let $a_i = a_i + 2b_i = 1, \beta_i = a_i + 2b_i = 1$ for all $i \in M$ and $\alpha = \max \alpha_i, \beta = \min \beta_i$, and define a new sequence $\{y_j | y_j = \alpha_i or \beta_i\}$ out of $\alpha$'s and $\beta$'s such that $y_1 < y_2 < \ldots < y_p$, $(p \leq 2m)$ and if one or more $\alpha_i$ and $\beta_j$ are the same values, they are considered as identical. Note that since $0 \leq d_i, \alpha_i < \beta_i$ holds for all $i$, and that $p = 2m$ if all the $\alpha$'s and $\beta$'s are different.

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

$$[\alpha_i, \beta_i] \cap \Gamma_j = \begin{cases} \Gamma_j & \text{if } y_j \notin \Gamma_j \\ \{y_j\} \text{ or } \{y_{j+1}\} & \text{if } y_j \in \Gamma_j \\ \emptyset & \text{if } y_j + 1 \notin \Gamma_j \end{cases}$$

We define the index set $A_j$ associated with $\Gamma_j$,

$$A_j = \{i | [\alpha_i, \beta_i] \cap \Gamma_j\}.$$ 

**Lemma 1.**

(12) $A_j + 1 = \{A_j \cup I_x\} \cap I_\beta = \{A_j \cap I_\beta\} \cup I_\alpha$, $j = 1, \ldots, p-1,$

or

(13) $A_j = \{A_j + 1 \cap I_\alpha\} \cup I_\beta = \{A_j + 1 \cap I_\beta\} - I_\alpha$, $j = p-1, \ldots, 1.$

where

$I_\alpha = \{i | \alpha_i = y_{j+1}\} = A_j \cap I_\alpha$, $j = 1, \ldots, p-1,$

$I_\beta = \{i | \beta_i = y_{j+1}\} = A_j \cap I_\beta$, $j = 1, \ldots, p-1.$

**Proof:** Note that $\Gamma_j = [y_j, y_{j+1}]$ and $y_{j+1}$ corresponds to some $\alpha_k$ or and some $\beta_h$, $k \neq h$. If $i \in I_\alpha$ then $i \in A_j \cap A_{j+1}$. If $i \in I_\beta$ then $i \in A_j \cap A_{j+1}$. Conversely, if $i \in A_j \cap A_{j+1}$, then $i \leq y_{j+1}$ and $i \geq y_{j+1}$. Hence, $i \in I_\alpha$.

If $i \in A_j \cap A_{j+1}$, then $\beta_i \leq y_{j+1}$ and $\beta_i \geq y_{j+1}$. Hence $i \in I_\beta$.

Therefore, $I_\alpha = A_j \cap A_{j+1}$ and $I_\beta = A_j \cap A_{j+1}$.

On the other hand,

$$A_j + 1 = \{A_j \cap A_{j+1}\} \cup (A_j \cap A_{j+1}) = \{A_j \cap A_{j+1}\} \cup (A_j \cap A_{j+1}) = \{A_j \cap I_\beta\} \cup I_\alpha = \{A_j \cap I_\alpha\} - I_\beta. \quad (\because I_\alpha \cap I_\beta = \emptyset)$$

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.
and
\[
A_j = \{A_j \cap A_{j+1}\} \cup \{A_j \cap A_{j+1}^c\} \\
= \{A_{j+1} \cap A_j^c\} \cup \{A_j \cap A_{j+1}\} \\
= \{A_{j+1} \cap I^c_B\} \cup \{I^c_B\} \\
= \{A_{j+1} \cup I^c_B\} \setminus I^c_B. \quad \text{Q.E.D.}
\]

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

From the definition of bracket symbol and (8), existence of the partial derivative of \(D(\lambda)\) is guaranteed. ([3],[6],[7]) Then the maximal point \(\lambda_0\) of \(D(\lambda)\) can be obtained as a solution to the equation \(\partial D(\lambda)/\partial \lambda = 0\),

\[
\partial D(\lambda)/\partial \lambda = - \sum_{i \in M} \frac{<\lambda - a_i>}{2b_i} > + B = 0.
\]

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

\[
\sum_{i \in M} x_i^1(\lambda) = \sum_{i \in M} \frac{<\lambda - a_i>}{2b_i} = B.
\]

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.,

\[
\begin{align*}
\delta(\lambda) &= \sum_{i \in M} x_i^1(\lambda) = \sum_{i \in M} \frac{<\lambda - a_i>}{2b_i} \\
\delta^0(\lambda) &= \sum_{i \in M} x_i^0(\lambda) = \sum_{i \in M} \frac{(\lambda - a_i)}{2b_i} = k_0\lambda + c_0,
\end{align*}
\]

where
\[
k_0 = \sum_{i \in M} 1/2b_i, \quad c_0 = - \sum_{i \in M} a_i/2b_i.
\]

As it is easily seen that the function \(\delta(\lambda)\) is a non-decreasing line-segment function with vertices \(\{y_j, \delta_j\}\), we write the equation of the line-segment on \(I^j\) 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 \((y_j, \delta_j)\) and \((y_{j+1}, \delta_{j+1})\), that is,

\[
\begin{align*}
k_j &= \frac{(\delta_{j+1} - \delta_j)}{(\delta_{j+1} - \delta_j)} \\
c_j &= \delta_j - y_j \frac{(\delta_{j+1} - \delta_j)}{(y_{j+1} - y_j)}
\end{align*}
\]
where abbreviation $\delta_j = \delta(y_j)$ is used.

On the other hand, $k_j$ and $c_j$ are determined by the index sets $\Lambda_j$, $\Lambda_\alpha$ and $\Lambda_\beta$, i.e.,

$$k_j = \sum_{i \in \Lambda_j} \frac{1}{2} b_i,$$

(19)

$$c_j = -\sum_{i \in \Lambda_j} a_i + \sum_{i \in \Lambda_\alpha} i + \sum_{i \in \Lambda_\beta} d_i,$$

where $\Lambda_\alpha = \{ h|a_i \geq \gamma_j \}$ and $\Lambda_\beta = \{ h|\beta_i < \gamma_j \}$.

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

Lemma 2.

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

Proof

$$k_j = \sum_{i \in \Lambda_j} \frac{1}{2} b_i$$

and $\Lambda_j$ 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.

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

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 $\lambda$ 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 $k_0 - k_j > 0$ for all subinterval $\Gamma_j$ from Lemma 2 and both $\delta^0(\lambda)$ and $\delta(\lambda)$ are continuous. It is also easily seen that $\phi(\lambda) < 0$ for some $\lambda < \min a_i$ and $\phi(\lambda) > 0$ for some $\lambda > \max \beta_i$.

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 $\alpha, \beta$ and the assumption $\alpha < \beta$, we can write all subintervals as $\Gamma_j = [\alpha_i, \alpha_i]$ or $[\beta_i, \beta_i]$ except $\Gamma_k = [\alpha, \beta]$. By noting that $\alpha_i < \alpha < \beta < \beta_i$, we obtain $\Lambda_k = M$ because $\Gamma_k = [\alpha, \beta] = [\alpha_i, \beta_i]$ for all $i$.

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

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

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

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

The case $\alpha > \beta$.

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

Then,

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

Each $\Gamma_j$ is either contained in $[\alpha_k, \beta_k]$ or not. If the former is the case,

$$\Gamma_j \cap [\alpha_k, \beta_k] \subseteq [\alpha_k, \beta_k] \cap [\alpha_k, \beta_k] \subseteq \{ \alpha \} \text{ or } \{ \beta \},$$

that is, $\Gamma_j \cap [\alpha_k, \beta_k]$. If the latter is the case, it is trivial that $\Gamma_j \not\subseteq [\alpha_k, \beta_k]$.

Therefore, there exists at least an interval $[\alpha_i, \beta_i]$ such that $\Gamma_j \not\subseteq [\alpha_i, \beta_i]$ for any $j$. Hence, $|\Lambda_j| < |M|$ for all $j$, which means that there exists no interval $\Gamma_j$ such that $\phi(\lambda) \equiv 0$ for $\lambda \in \Gamma_j$ by the non-decreasing property of $\phi(\lambda)$. Thus we conclude that there exists only one point $\lambda^0$ where $\phi(\lambda^0) = 0$.

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

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

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

$$P(\lambda) = \{ i \mid \lambda > \alpha_i \}.$$  

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

$$\phi(\lambda) = \left\{ \begin{array}{ll}
(\lambda-a_i)/(2b_i-\lambda-b_i), & \text{if } i \in N(\lambda) \\
(\lambda-a_i)/(2b_i-\lambda-b_i), & \text{if } i \in P(\lambda)
\end{array} \right.$$  

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

\[(\beta - a_i)/2b_i - l_i \leq 0, \text{ for } i \in \mathbb{N}(\beta),\]

the equality holds only for \(a_i = \beta\). Thus we have

\[\phi(\beta) = \sum_{i \in \mathbb{N}(\beta)} \{(\beta - a_i)/2b_i - l_i \leq 0,\]

the equality holds only if \(a_i = \beta\) for all \(i \in \mathbb{N}(\beta)\).

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

\[
\begin{align*}
\delta(\lambda), \delta^0(\lambda) & \quad \delta^0(\lambda) \\
\Sigma d_i & \quad \delta(\lambda) \\
\Sigma l_i & \quad \delta^0(\lambda) \\
\end{align*}
\]

(a) Case \(\alpha < \beta\)

\[
\begin{align*}
\gamma_1 \cdots \gamma_j \gamma_{j+1} \cdots \gamma_p
\end{align*}
\]

(b) Case \(\beta \leq \alpha\)

\[
\begin{align*}
\gamma_1 \cdots \gamma_h \gamma_k \cdots \gamma_p
\end{align*}
\]

Fig. 1 Relation of \(\delta(\lambda)\) and \(\delta^0(\lambda)\)
Lemma 4.

δ(λ) is a monotone increasing line-segment convex function for γ₁≤λ≤β and δ(λ) is a monotone increasing line-segment concave function for α≤λ<γₚ.

Proof:
Assume that α = γₖ, β = γₙ, then we can suppose that γᵢ corresponds to

γᵢ = αᵢ, j = 1,...,h-1
γₖ+j = βₙⱼ, j = 1,...,p-k.

Then each index set Λⱼ, j = 1,...,h-1 and j = k,...,p-1 are determined as follows:
Set Λ₀ = {ϕ}, then from (12) and (13) in Lemma 1,

Λⱼ = Λⱼ₋₁ ∪ Iₜ, j = 1,...,h-1

and
Λₖ₋₁ = Λₖ₋₁ ∪ Iₜ, j = 1,...,p-k-1.

Hence,
|Λ₁| < |Λ₂| < ... < |Λₙ₋₁|
and
|Λₖ| > |Λₖ₊₁| > ... > |Λₚ₋₁|.

Therefore, from (19),

k₁ < k₂ < ... < kₙ₋₁
and
kₖ > kₖ₊₁ > ... > kₚ₋₁.

Q.E.D.

4. Algorithms

We shall propose three algorithms for finding the optimal λ; (i) polynomial 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 δ(λ) can be generally divided into three parts, convex part for λ≤β, concave part for λ>α and neither convex nor concave part for β<λ<α. Theoretically, any convenient convex (concave or linear) function
may be available as an approximation function for the convex (concave or neither convex nor concave) part. For practical purposes 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 $\lambda$, 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 $(\lambda^1, \delta^1)$ and its tangent at $\lambda^0$ or $\lambda^1$. We can use the line-segment on the subinterval containing $\lambda^0$ or $\lambda^1$ as the tangent of the approximating function. As noted in section 3, this line-segment can be determined by $\Lambda_j$, which can be determined by $x^0_i(\lambda)$, and $x^0_i(\lambda)$ have been already computed when $\delta^0$ or $\delta^1$ were evaluated.

begin

compute $\delta^0(\lambda) = B$ and let $\lambda^0$ be the solution;

if $\delta(\lambda^0) \neq B$ then

begin

compute $\alpha = \max \alpha_i$ and $\beta = \min \beta_i$;

if $\lambda^0 < \beta$ then compute the solution by using convex approximating functions;

else

begin

if $\lambda^0 > \alpha$ 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 $\lambda^0$, can be easily obtained from the solution to the equation $\delta^0(\lambda) = B$. Then the subinterval $\Gamma_j$, $\Gamma_j = \lambda^0$, and index set $\Lambda_j$, $\Lambda_j = \{i | [\alpha_i, \beta_i) \supseteq \Gamma_j\}$, can be determined, furthermore, $\Gamma_{j+1}$ and $\Lambda_{j+1}$, $\Gamma_{j+2}$ and $\Lambda_{j+2}$, $\ldots$, or $\Gamma_{j-1}$ and $\Lambda_{j-1}$, $\Gamma_{j-2}$ and $\Lambda_{j-2}$, $\ldots$ are sequentially determined by Lemma 1. If we find the interval $[\delta_k, \delta_{k+1}] \supseteq B$, then the optimal $\lambda$ can be computed from the equation $y_k = k^\lambda + c_k = B$, where $y_k$ is the equation of the
line-segment on the subinterval $\Gamma_k$.

begin

given initial $\lambda^0$;
determine the subinterval $\Gamma_j$ and corresponding index set $A_j$ such that $\Gamma_j \supseteq \lambda^0$;
if $\delta(\lambda^0) < B$ then search $\Gamma_k$ by increasing $k$ until $[\delta_k, \delta_{k+1}] \supseteq B$ is found;
else search $\Gamma_k$ by decreasing $k$ until $[\delta_k, \delta_{k+1}] \supseteq B$ is found;
solve the equation $\kappa \lambda + c = B$;
end

4.3 Algorithm HB

The algorithm PA is efficient for the global estimation of the optimal $\lambda$. On the other hand, the algorithm SS is efficient for the local computation of the optimal $\lambda$. 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 $\lambda^0$ which has been obtained by the algorithm PA.

begin
Execute algorithm PA $r$ times and let $\lambda^0$ be the current solution;
Execute algorithm SS starting from $\lambda^0$;
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 $\lambda^0$ be the root of the equation $\delta^0(\lambda) = B$, then $(\lambda^0, \delta^0_0)$ can be used as one of the points. The other point can be determined by the non-decreasing property of $\delta(\lambda)$ as follows:

$$ \text{if } \delta(\lambda^0) > B \text{ then } \lambda^1 = \min_{i} \alpha_i \text{ else } \lambda^1 = \max_{i} \beta_i; $$
Secondly, we consider the roots of the quadratic equation. If we write the quadratic approximating function as \( \delta(\lambda) = p\lambda^2 + q\lambda + r \), the equation \( \delta(\lambda) = B \) has two roots, say \( \lambda^a \) and \( \lambda^b \). However, it is sufficient to take only one root by the non-decreasing property of \( \delta(\lambda) \) as follows:

\[
\text{begin}
\text{if } \delta(\lambda^0) < B \text{ then}
\text{begin}
\text{if } p > 0 \text{ then } \lambda = \max\{\lambda^a, \lambda^b\};
\text{if } p < 0 \text{ then } \lambda = \min\{\lambda^a, \lambda^b\};
\text{end}
\text{if } \delta(\lambda^0) > B \text{ then}
\text{begin}
\text{if } p > 0 \text{ then } \lambda = \min\{\lambda^a, \lambda^b\};
\text{if } p < 0 \text{ then } \lambda = \max\{\lambda^a, \lambda^b\};
\text{end}
\text{end}
\]

(2) Algorithm SS

An important part of algorithm SS is to determine the linear functions on the subintervals \( \Gamma_j \)'s by Lemma 1 and eq. (12) or (13). That is, when searching for subintervals \( \Gamma_j \rightarrow \Gamma_{j+1} \rightarrow \ldots \) or \( \Gamma_j \rightarrow \Gamma_{j-1} \rightarrow \ldots \), we must determine coefficients \( k_j \) and \( c_j \) of the linear function \( y_j = k_j x + c_j \) define on the subinterval \( \Gamma_j \).

As noted in section 4.1, \( k_j \) and \( c_j \) can be determined by the index sets \( \Lambda_j, \Lambda_\alpha \) and \( \Lambda_\beta \) 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 \( \Lambda_j, \Lambda_\alpha \) and \( \Lambda_\beta \), respectively. We set \( i \)-th bit in the list \( L_j \) as follows:

if index \( i \) is in \( \Lambda_j \), 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 \( \Lambda_\beta \) 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.,

\[ L_3 = L_1 \oplus L_2, \text{ where } \oplus \text{ means } m \text{-bitswise Boolean sum and } \oplus \text{ is a compliment.} \]
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 αₖ's and βₖ's in increasing order to make the sequence \{γⱼ\}. This sorting needed an average of \(O(2mc\log 2m)\), \(c>0\), comparisons.\[1\] However, when we already had the sequence \{γⱼ\} 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 \([δ_k, δ_{k+1}]\) such that \([δ_k, δ_{k+1}]\) ⊆ 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>
<thead>
<tr>
<th>Algorithms</th>
<th>Number of variables</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td>SS</td>
<td>3</td>
</tr>
<tr>
<td>HB</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1

Number of searched intervals

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
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 $\lambda$ 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 $\alpha$ and $\beta$, 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.
References


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.