<table>
<thead>
<tr>
<th>Title</th>
<th>An Application of Modular approach to Separable Nonlinear Programming (Mathematical Structure of Optimization Theory)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Iwasaki, Akinori; Hikita, Mitsunori; Nakagawa, Yuji; Narihisa, Hiroyuki</td>
</tr>
<tr>
<td>Citation</td>
<td>数理解析研究所講究録 (1994), 864: 83-90</td>
</tr>
<tr>
<td>Issue Date</td>
<td>1994-04</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/2433/83904">http://hdl.handle.net/2433/83904</a></td>
</tr>
<tr>
<td>Type</td>
<td>Departmental Bulletin Paper</td>
</tr>
<tr>
<td>Textversion</td>
<td>publisher</td>
</tr>
</tbody>
</table>
Abstract

A discrete optimization method, which is called modular approach, is proposed for solving a separable nonlinear programming problem. By dividing search space of variables, the nonlinear programming problem is translated into a discrete optimization problem that is equivalent to nonlinear knapsack problem. When the nonlinear knapsack problem is solved, we do not need the convexity and differentiability of original problem. The nonlinear knapsack problem can be solved efficiently by modular approach. It is shown that modular approach can be applied to a nonlinear programming problem by computational experiments.

1. Introduction

A separable nonlinear programming problem with one constraint function is written as follows:

\[ \begin{align*}
    \text{maximize} & \quad f(x) = \sum_{i \in I} f_i(x_i), \\
    \text{subject to} & \quad g(x) = \sum_{i \in I} g_i(x_i) \leq b, \\
    & \quad x_i \in S_i \quad (i \in I),
\end{align*} \]

where \( I = \{1, 2, \ldots, n\} \), and \( S_i \subset \mathbb{R} \) is a search space, and \( b \) is a maximum amount of available resource.

We divide the search space \( S_i \) into finite set \( A_i \) for each \( i-th \) variable:

\[ \begin{align*}
    \text{maximize} & \quad f(x) = \sum_{i \in I} f_i(x_i), \\
    \text{subject to} & \quad g(x) = \sum_{i \in I} g_i(x_i) \leq b, \\
    & \quad x_i \in A_i \subset S_i \quad (i \in I),
\end{align*} \]
where $I = \{1, 2, \ldots, n\}$, $A_i = \{a_{i1}, a_{i2}, \ldots, a_{ij}, \ldots, a_{ik_i}\}$. The search space $S_i$ is represented by $k_i$ points $\{a_{i1}, \ldots a_{ik_i}\}$.

The original problem $<N>$ is translated into discrete optimization problem $<K>$ that is equivalent to the nonlinear knapsack problem. Solving the nonlinear knapsack problem by discrete optimization method, the convexity and differentiability of original problem are not required.

We use modular approach (MA) for solving the nonlinear knapsack problem $<K>$. MA can solve the large scale nonlinear knapsack problem.

The optimal solution of the problem $<K>$ is a near optimal solution of the original problem $<N>$. The search space $S_i$ of the original problem can be reduced to the neighborhood of the near optimal solution. The new problem with reduced search spaces is created and solved by MA. By repetition of the above procedures, the near optimal solutions converge into the optimal solution of original problem $<N>$.

2. Modular approach

Nakagawa[1] proposed a new solution method called modular approach (MA) for solving discrete optimization problem. MA is a bottom-up scheme as well as Dynamic Programming. First, MA considers an optimization system corresponding to a given discrete optimization problem. Next, MA executes the following items 1) 2) recursively until the number of variables $I$ becomes one.

1) The set $A_i$ is reduced by fathoming tests.

2) Integrate two variables into one variable.

As for fathoming tests, we use dominance test, bounding test and feasibility test, which are techniques commonly used by Branch-and-Bound.

To integrate means to introduce a new set $A_{NEW}$ that is corresponding
to cartesian product of the two sets as follows:

\[ A_{NEW} = A_j \times A_m \]  

(7)

and \( j \) and \( m \) are removed from the set \( I \).

There are four ways to select the sets \( j \) and \( m \) in the set \( I \). Let \( k_i \) be the number of elements in the set \( A_i \).

1) \( j \) and \( m \) such that \( k_j \) and \( k_m \) are the least.

2) \( j \) such that \( k_j \) is the least, and \( m \) such that \( k_m \) is the most.

3) \( j \) and \( m \) such that \( k_j \) and \( k_m \) are the most.

4) \( j \) and \( m \) in order of \( i \in I \).

We choose the item 2) that is the fastest and can solve the largest scale problem. [3]

MA written by pseudo code is shown in figure 1.

The input of \( Modular\ Approach() \) is a data sequence of Problem \(< PC > \) and Quasi-Optimal Solution \(< NEAR > \). Problem \(< PC > \) contains a data sequence of current problem \(< P > \) and a data sequence of \(< T > \) that is required to translate the current problem \(< P > \) into primal problem. The Quasi-Optimal Solution \(< NEAR > \) is given by Recursive Greedy method[2]. Function \( Fathom() \) reduces the set \( A_i \) by fathoming tests, and renews the current problem\(< P > \). Function \( ChoiceIM() \) selects two sets \( A_j \) and \( A_m \). Function \( Integrate() \) integrates the two sets \( A_j \) and \( A_m \) into one set \( A_{NEW} \). After repeating the fathoming tests and integration, function \( FindOptimalSolution() \) gives the optimal solution of the one variable problem.

3. Computational experiments

We divide the search space \( S_i \) of given problem into finite set \( A_i \). We create the nonlinear knapsack problem from the set \( A_i \). The next two
steps are repeated until required precision is given.

1) MA is applied to the nonlinear knapsack problem, and near optimal solution of given problem is given.

2) The neighborhood of the near optimal solution is divided, and the new nonlinear knapsack problem with reduced search spaces is created.

3.1 Example 1

We consider a convex and differentiable problem as follows:

\[
\begin{align*}
\text{maximize} & \quad f(x) = \sum_{i=1}^{10} (a_i + b_i x_i)^2, \\
\text{subject to} & \quad g(x) = \sum_{i=1}^{10} (c_i + d_i x_i)^2 \leq e, \\
& \quad x_i \in R.
\end{align*}
\]

(8) (9) (10)

Coefficients $a_i, b_i, c_i, d_i$ and $e$ are shown in Table 1.

This problem is solved by numerical computation and MA. Each results are shown in Table 2. Generally the results of MA are agreement with the results of numerical computation.

3.2 Example 2

We consider the nonconvex and undifferentiable problem as follows:

\[
\begin{align*}
\text{maximize} & \quad f(x) = \sum_{i=1}^{10} f_i(x_i), \\
\text{subject to} & \quad g(x) = \sum_{i=1}^{10} g_i(x_i), \\
& \quad f_i(x_i) = \begin{cases} 
  a_{i1} \left| \sin(x_i + b_{i1}) \right| & (0.0 \leq x_i < 1.0) \\
  a_{i2} \left| \cos(x_i + b_{i2}) \right| & (1.0 \leq x_i < 2.0) \\
  a_{i3} \ln(x_i + b_{i3}) & (2.0 \leq x_i < 3.0) \\
  a_{i4} \sqrt{x_i + b_{i4}} & (3.0 \leq x_i < 4.0) \\
  a_{i5} \exp(x_i/5 + b_{i5}) & (4.0 \leq x_i < 5.0),
\end{cases}
\end{align*}
\]

(11) (12) (13)
\[ g_i(x_i) = (x_i + c_i)^2 \leq e. \] (14)

Coefficients \( a_{i1}, \ldots, a_{i5}, b_{i1}, \ldots, b_{i5}, c_i \) and \( e \) are shown in Table 3.

This problem is solved by MA and the results are shown in Table 4. First, the search space is divided into 100 elements, and the near optimal solution is given by MA. Next, the neighborhood of the near optimal solution is divided into 100 elements, and the second near optimal solution is also given by MA. The second near optimal solution exhausts the resource of constraint.

4. Concluding remarks

Solving two examples, it is shown that MA can solve nonconvex and undifferentiable nonlinear programming problem.

References


DATADEF
< K >= \{I, \{K_1, K_2, \ldots, K_J\}\};
< f >= \{\{f_1(1), \ldots, f_1(K_1)\}, \ldots, \{f_I(1), \ldots, f_I(K_I)\}\};
< g >= \{\{g_1(1), \ldots, g_1(K_1)\}, \ldots, \{g_I(1), \ldots, g_I(K_I)\}\};
< P >= \{< K >, < f >, < g >, b\};
< PC >= \{< P >, < T >\};
< NEAR >= \{f^{NEAR}_{\{x_{NEAR}^1, \ldots, x_{NEAR}^J\}}\};
< OPT >= \{f^{OPT}_{\{x_{OPT}^1, \ldots, x_{OPT}^J\}}\};
< M >= \{m_1, m_3\};
ENDDEF

FUNCTION ModularApproach()
INPUT Problem< PC >, Quasi-Optimal Solution< NEAR >;
BEGIN
yes ← 1; No ← 0; isNearSolOptimal ← No;
WHILE I ≥ 2 DO
  \{< PC >, < NEAR >\} ← Fathom\{< PC >, < NEAR >\};
  IF exists i ∈ \{1, \ldots, I\} such that K_i = 0 THEN
    isNearSolOptimal ← Yes;
  \{< OPT >\} ← \{< NEAR >\};
  EXITWHILE
ENDWHILE
IF isNearSolOptimal = No THEN
  \{< OPT >\} ← FindOptimalSolution\{< PC >\};
  IF f^{OPT} < f^{NEAR} THEN
    \{< OPT >\} ← \{< NEAR >\};
ENDIF
ENDIF
RETURN Optimal Solution< OPT >
END

figure 1. Modular approach

<table>
<thead>
<tr>
<th>i</th>
<th>a_i</th>
<th>b_i</th>
<th>c_i</th>
<th>d_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-2.2073</td>
<td>2.8969</td>
<td>-0.7442</td>
<td>-1.0398</td>
</tr>
<tr>
<td>2</td>
<td>2.4402</td>
<td>4.3</td>
<td>3.7626</td>
<td>-4.3934</td>
</tr>
<tr>
<td>3</td>
<td>-0.7114</td>
<td>5.52</td>
<td>-1.8898</td>
<td>1.871</td>
</tr>
<tr>
<td>4</td>
<td>1.2999</td>
<td>7.832</td>
<td>1.9367</td>
<td>3.4018</td>
</tr>
<tr>
<td>5</td>
<td>-2.2473</td>
<td>5.5</td>
<td>3.2593</td>
<td>4.4436</td>
</tr>
<tr>
<td>6</td>
<td>-4.9337</td>
<td>-4.98</td>
<td>1.0066</td>
<td>2.43</td>
</tr>
<tr>
<td>7</td>
<td>2.9042</td>
<td>2.9</td>
<td>3.1389</td>
<td>-1.7736</td>
</tr>
<tr>
<td>8</td>
<td>1.3991</td>
<td>-4.2178</td>
<td>2.8799</td>
<td>2.4</td>
</tr>
<tr>
<td>9</td>
<td>-1.0905</td>
<td>3.0</td>
<td>-3.4813</td>
<td>-1.283</td>
</tr>
<tr>
<td>10</td>
<td>-0.6581</td>
<td>4.54</td>
<td>-3.4915</td>
<td>-2.3995</td>
</tr>
</tbody>
</table>

Table 1. Coefficient of Example 1

e=1000
<table>
<thead>
<tr>
<th></th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
<th>$x_7$</th>
<th>$x_8$</th>
<th>$x_9$</th>
<th>$x_{10}$</th>
<th>$f$</th>
<th>$g$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Numerical</strong></td>
<td>6.498558</td>
<td>1.0223189</td>
<td>17.132994</td>
<td>-1.120442</td>
<td>-0.962257</td>
<td>0.0719372</td>
<td>2.9085138</td>
<td>-1.976559</td>
<td>-7.244958</td>
<td>-2.47795</td>
<td>10301.902</td>
<td>1000</td>
</tr>
<tr>
<td><strong>Solution</strong></td>
<td>-6.56</td>
<td>1</td>
<td>17.12</td>
<td>-1.12</td>
<td>-0.96</td>
<td>0.08</td>
<td>2.88</td>
<td>-1.96</td>
<td>-7.32</td>
<td>-2.48</td>
<td>10301.736</td>
<td>999.99833</td>
</tr>
<tr>
<td><strong>Errors</strong></td>
<td>0.061442</td>
<td>0.0223189</td>
<td>0.0129939</td>
<td>-0.000442</td>
<td>-0.002257</td>
<td>-0.008063</td>
<td>0.0285138</td>
<td>-0.016559</td>
<td>0.075042</td>
<td>0.0020505</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>N=1000</strong></td>
<td>-6.53</td>
<td>1.02</td>
<td>17.13</td>
<td>-1.12</td>
<td>-0.96</td>
<td>0.06</td>
<td>2.91</td>
<td>-1.97</td>
<td>-7.25</td>
<td>-2.48</td>
<td>10301.888</td>
<td>999.9996</td>
</tr>
<tr>
<td><strong>Solution</strong></td>
<td>0.031442</td>
<td>0.0023189</td>
<td>0.0029939</td>
<td>-0.000442</td>
<td>-0.002257</td>
<td>0.0119372</td>
<td>-0.001486</td>
<td>-0.000559</td>
<td>0.005042</td>
<td>0.0020505</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Errors</strong></td>
<td>0.009442</td>
<td>0.001681</td>
<td>0.0009939</td>
<td>-0.000442</td>
<td>-0.0017431</td>
<td>-6.28E-05</td>
<td>0.0045138</td>
<td>-0.00559</td>
<td>-0.000958</td>
<td>0.0020505</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>N=4000</strong></td>
<td>-6.495</td>
<td>1.022</td>
<td>17.133</td>
<td>-1.12</td>
<td>-0.962</td>
<td>0.072</td>
<td>2.909</td>
<td>-1.977</td>
<td>-7.248</td>
<td>-2.478</td>
<td>10301.901</td>
<td>1000</td>
</tr>
<tr>
<td><strong>Solution</strong></td>
<td>-0.003558</td>
<td>0.003189</td>
<td>-6.14E-06</td>
<td>-0.000442</td>
<td>-0.000257</td>
<td>-6.28E-05</td>
<td>-0.000486</td>
<td>0.0004414</td>
<td>0.003042</td>
<td>5.047E-05</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Errors</strong></td>
<td>0.000802</td>
<td>-8.11E-05</td>
<td>3.386E-05</td>
<td>-0.000282</td>
<td>-1.69E-05</td>
<td>9.721E-05</td>
<td>0.0001938</td>
<td>-7.86E-05</td>
<td>2.034E-06</td>
<td>-2.95E-05</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>N=1000*1000</strong></td>
<td>-6.49936</td>
<td>1.0224</td>
<td>17.13296</td>
<td>-1.12016</td>
<td>-0.96224</td>
<td>0.07184</td>
<td>2.90832</td>
<td>-1.97648</td>
<td>-7.24496</td>
<td>-2.47792</td>
<td>10301.902</td>
<td>1000</td>
</tr>
<tr>
<td><strong>Solution</strong></td>
<td>0.000802</td>
<td>-8.11E-05</td>
<td>3.386E-05</td>
<td>-0.000282</td>
<td>-1.69E-05</td>
<td>9.721E-05</td>
<td>0.0001938</td>
<td>-7.86E-05</td>
<td>2.034E-06</td>
<td>-2.95E-05</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Errors</strong></td>
<td>0.000318</td>
<td>-1.05E-06</td>
<td>-4.61E-05</td>
<td>-3.779E-05</td>
<td>-1.69E-05</td>
<td>9.721E-05</td>
<td>0.0001938</td>
<td>1.374E-06</td>
<td>-7.8E-05</td>
<td>0.0001305</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

N: The number of division. (4000*1000 means 1000 division after 4000 division)
d: Division width
Table 3. Coefficients of Example 2

<table>
<thead>
<tr>
<th>i</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$a_4$</th>
<th>$a_5$</th>
<th>$b_1$</th>
<th>$b_2$</th>
<th>$b_3$</th>
<th>$b_4$</th>
<th>$b_5$</th>
<th>$c_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.6</td>
<td>3.5</td>
<td>1.6</td>
<td>1.1</td>
<td>0.7</td>
<td>4.5</td>
<td>2.2</td>
<td>1.5</td>
<td>0.8</td>
<td>0.4</td>
<td>0.8</td>
</tr>
<tr>
<td>2</td>
<td>2.4</td>
<td>2.2</td>
<td>0.6</td>
<td>0.2</td>
<td>0.2</td>
<td>2.0</td>
<td>1.9</td>
<td>0.6</td>
<td>0.3</td>
<td>0.7</td>
<td>1.5</td>
</tr>
<tr>
<td>3</td>
<td>5.0</td>
<td>4.8</td>
<td>3.1</td>
<td>2.5</td>
<td>0.8</td>
<td>4.0</td>
<td>4.2</td>
<td>0.8</td>
<td>1.0</td>
<td>0.5</td>
<td>2.0</td>
</tr>
<tr>
<td>4</td>
<td>2.0</td>
<td>1.9</td>
<td>0.7</td>
<td>0.2</td>
<td>0.1</td>
<td>3.6</td>
<td>4.1</td>
<td>0.2</td>
<td>0.2</td>
<td>0.6</td>
<td>0.5</td>
</tr>
<tr>
<td>5</td>
<td>4.5</td>
<td>4.0</td>
<td>3.0</td>
<td>2.4</td>
<td>0.7</td>
<td>2.9</td>
<td>1.2</td>
<td>0.9</td>
<td>0.9</td>
<td>0.5</td>
<td>0.2</td>
</tr>
<tr>
<td>6</td>
<td>2.8</td>
<td>3.0</td>
<td>1.1</td>
<td>0.8</td>
<td>0.5</td>
<td>4.7</td>
<td>1.6</td>
<td>1.2</td>
<td>0.5</td>
<td>0.8</td>
<td>0.9</td>
</tr>
<tr>
<td>7</td>
<td>3.9</td>
<td>4.1</td>
<td>2.9</td>
<td>2.5</td>
<td>1.0</td>
<td>4.4</td>
<td>3.8</td>
<td>0.4</td>
<td>1.3</td>
<td>0.3</td>
<td>1.3</td>
</tr>
<tr>
<td>8</td>
<td>3.2</td>
<td>3.4</td>
<td>1.2</td>
<td>0.9</td>
<td>0.6</td>
<td>3.3</td>
<td>0.7</td>
<td>1.8</td>
<td>1.1</td>
<td>0.9</td>
<td>1.8</td>
</tr>
<tr>
<td>9</td>
<td>4.7</td>
<td>5.0</td>
<td>2.7</td>
<td>2.7</td>
<td>0.6</td>
<td>0.8</td>
<td>3.6</td>
<td>0.1</td>
<td>0.7</td>
<td>0.4</td>
<td>0.7</td>
</tr>
<tr>
<td>10</td>
<td>2.7</td>
<td>2.7</td>
<td>1.0</td>
<td>0.5</td>
<td>0.1</td>
<td>1.1</td>
<td>2.3</td>
<td>1.9</td>
<td>0.6</td>
<td>0.2</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Table 4. Results of Example 2

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
<th>$x_7$</th>
<th>$x_8$</th>
<th>$x_9$</th>
<th>$x_{10}$</th>
<th>$f$</th>
<th>$g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>N=100</td>
<td>0.10</td>
<td>0.00</td>
<td>3.40</td>
<td>0.85</td>
<td>3.90</td>
<td>0.00</td>
<td>3.80</td>
<td>4.90</td>
<td>3.90</td>
<td>0.35</td>
<td>39.437</td>
</tr>
<tr>
<td>N=100*100</td>
<td>0.182</td>
<td>0.000</td>
<td>3.526</td>
<td>0.998</td>
<td>4.000</td>
<td>0.000</td>
<td>3.866</td>
<td>4.999</td>
<td>4.000</td>
<td>0.442</td>
<td>39.444</td>
</tr>
</tbody>
</table>

N: The number of division