A Study of Generalized Reduced Gradient Method with Different Search Directions

*Hong-Tau Lee  ** Sheu-Hua Chen  *He-Yau Kang

*Department of Industrial Engineering and Management
National Chin-Yi Institute of Technology, Taichung, Taiwan.
E-mail: leeh@chinyi.ncit.edu.tw

**Department of Industrial Engineering and Management
Hsiou-ping Institute of Technology
Taichung, Taiwan.

Abstract

The generalized reduced gradient method has been proven to be a precise and accurate method for solving nonlinear programming problems. To enlarge the capability of this method for solving wilder application problems, in this research, the concept of Conjugate gradient and the Quasi-Newton approach are involved in the GRG algorithm whenever a direction must be generated for its' next search. As for the determination of the step length for each line search operation, the two phases of bracket search and square inner approximating method are provided. A four-criteria comparison is performed, that involving the effectiveness, efficiency, space occupation and solution precision of the three searching directions of the GRG algorithm by testing them on a set of twenty-five testing problems.

Keywords: nonlinear programming, algorithm, optimization, search.

1. Introduction

Many methods had been proven efficient and effective in special fields of application. Such as Simplex method [1,2] in linear programming problems, Conjugate gradient method [3] and Quasi-Newton method [4] in Non-linear Programming problems with no constraints, and
Convex simplex method [5] and Reduced gradient methods [6] in Non-linear programming with linear constraints, etc. Researchers devote their effort to enlarge these methods to wilder application. The Generalized Reduced Gradient method [7] has been developed and proven to be one of the efficient and effective methods for the Non-linear Programming problem with Non-linear constraints. With the superior properties, this method is interesting for other researchers. They also made some improvements for it. For instance Gabride and Ragsdell [8] proposed an algorithm called OPT method that used the Conjugate direction to modify the search direction. Lasdon [9] used the concept of binding constraints to refine the GRG method and involved the concept of Quasi-Newton method for the search direction. Sandgren [10] compared twenty four methods that involve four GRG algorithms for his proposed testing problems. Haggag [11] reported the application of GRG method in the real-life problems. Another researchers compared GRG method with other methods. Saber et. al. [12] compared the partitioning gradient based algorithm that developed from GRG method against the modified pattern search (MPS) method. Kao [13] compared the performance of several nonlinear programming software packges that of course involved the method "GINO" which developed based on reduced gradient. All of the above research support the superior properties of GRG method, though. The comparison for different search directions for GRG method is scarce. In this research, we compare the search directions of the gradient, Conjugate gradient and Quasi-Newton method for the GRG algorithm and employ a two-phases' method proposed by Kao [14] to determent the step length for each iteration in the GRG algorithm.

2. The Basic Idea of GRG Method

The basic concept of GRG method entails linearizing the Non-linear objective and constraint functions at a local solution with Taylor expansion equation. Then, the concept of reduced gradient method is employed which divides the variable set into two subsets of basic and non-basic variable and the concept of implicit variable elimination to express the basic variable by the non-basic variable. Finally, the constraints are eliminated and the variable space is deduced to only non-basic variables. The proven efficient method for non-constraints NLP problems are involved to solve the approximated problem and, then, the next optimal solution for the approximated problem should be found. The processes repeat again until it
fulfills the optimal conditions. No less the generality, the following general form of NLP problem is used for explanation.

\[
\begin{align*}
\text{Minimize} & \quad f(x) \\
\text{subject} & \quad h_i(x) = 0 \quad i = 1, \ldots, m \\
& \quad x_l \leq x \leq x_u \quad k = 1, \ldots, n
\end{align*}
\]

Function \( f(x) \) and \( h_i(x) \) \( i = 1, \ldots, m \) are continuous and differentiable in the domain region \( \{x \mid x_l \leq x \leq x_u \quad k = 1, \ldots, n\} \).

First of all, the linearizing process at a local feasible solution \( x_1 \) for the objective and constraint functions are performed as follows.

\[
\begin{align*}
\tilde{f}(x, x_1) &= f(x_1) + \nabla f(x_1)(x - x_1) \\
\tilde{h}_i(x, x_1) &= h_i(x_1) + \nabla h_i(x_1)(x - x_1) \quad i = 1, \ldots, m
\end{align*}
\]

By dividing the variable into two subsets one refers to the basic variables and denoted by \( \hat{x} \) another refers to non-basic variables and denoted by \( \bar{x} \). Rearranging the coefficients \( \nabla h_i(x) \) in the constraints to \( \nabla \hat{h}_i \) and \( \nabla \bar{h}_i(x) \) which correspondent to the basic and non-basic variable, respectively. Constructing the following two matrices leads to

\[
B_{m \times m} = \begin{bmatrix}
\nabla \hat{h}_1 \\
\nabla \hat{h}_2 \\
\vdots \\
\nabla \hat{h}_m \end{bmatrix} \quad \text{and} \quad \bar{A}_{m \times (n-m)} = \begin{bmatrix}
\nabla \bar{h}_1 \\
\nabla \bar{h}_2 \\
\vdots \\
\nabla \bar{h}_m \end{bmatrix}
\]

Since \( x_1 \) is a feasible solution for the original problem, it must be a feasible solution for the approximate problem, implying that

\[
\begin{align*}
\tilde{h}_i(x, x_1) &= h_i(x_1) + \nabla h_i(x_1)(x - x_1) \geq 0, \quad i = 1, \ldots, m \quad \text{and} \\
\nabla h_i(x_1)(x - x_1) &= 0, \quad i = 1, \ldots, m.
\end{align*}
\]

Expressing the set of constraints by the following matrix form.

\[
\begin{bmatrix}
B \\
\bar{A}
\end{bmatrix}\begin{bmatrix}
\hat{x} - \hat{x_1} \\
\bar{x} - x_1
\end{bmatrix} = 0
\]

Then, the basic variables set \( \hat{x} \) can be expressed by the following equations

\[\hat{x} - \hat{x_1} = 0\]
A Study of Generalized Reduced Gradient Method with Different Search Directions

\[ \hat{x} = \hat{x}_1 - B^{-1} \hat{A}(x - \hat{x}_1) \]  

(1)

Substituting the basic variables in the objective function with equation (1). The variable space in the objective function is changed from \( n \) to \( n-m \) and the problem is changed to a NLP with non-constraints.

If \( x_1 \) is an optimal solution, the gradient of objective function must be zero, implying that

\[ \frac{\partial f}{\partial x} = \frac{\partial f}{\partial x} \frac{\partial \hat{x}}{\partial \hat{x}_1} + \frac{\partial f}{\partial \hat{x}_1} = \nabla \hat{f}(x_1) - \nabla \hat{f}(x_1)B^{-1} \hat{A} = 0 \]  

(2)

Equation (2) is normally referred to as the reduced gradient. If the reduced gradient at the point \( x_1 \) equal to a zero vector then it also satisfies the Lagrange stable point conditions [15]. Otherwise, walking a little distance along the direction \( \hat{d} = -\nabla \hat{f} \), will reduce the value of the objective function. To keep the new solution feasible for the approximated problem, considering equation (1), the basic variable should walk along the direction \( \hat{d} = -B^{-1} \hat{A} \hat{d} \). Consequently,

\[ d = \begin{bmatrix} \hat{d} \\ \hat{d} \end{bmatrix} \]  

(3)

is the search direction for the algorithm.

3. Determination of the Step Length

From the previous section, the search direction is calculated. The next issue is to determine the step length for each iteration. Himmelblau [16] had concluded after his comparison of many line search methods that the method provided by Davis-Swann-Compey and Powell which usually called DSC-Powell is an efficient once. In this research, a two-phases’ method is proposed to find the step length. In the first phase, a bracket search method developed by Kao [14] in his CVMNLP algorithm is used to fine the possible region of the optimal length. While in the second phase, a square inner-estimate method developed by Powell [17] is applied to find the exact position. The process is described as following:

Phase one,
Assume that we are in a current point \( x \) and the search direction \( d \) for the point \( x \) is determined. The possible maximum step length is divided into ten segments that are denoted by \( D \).

Let \( k=1 \)
\[
FA = F(x) \quad DA = 0 \\
FB = F(x) \quad DB = 0 \\
FC = F(x) \quad DC = 0
\]

Step 1. \( y = x + Dd \) and \( FY = F(y) \)
- let \( FC = FB \) \quad \( DC = DB \)
- \( FB = FA \) \quad \( DB = DA \)
- \( FA = FY \) \quad \( DA = D \)

Step 2. If \( FA < FB \) then \( k = k + 1 \), \( D = kD \). Back to step 1.
- If \( FA > FB \) and \( k = 1 \) then
  - let \( DB = 0.5(DA + DC) \) and calculates
    \( y = x + DB \cdot d \) and \( FY \).
  - Let \( FB = FY \) back to main program.
- If \( FA > FB \) and \( k > 1 \) then back to main program.

Phase two,

From the phase one bracket search, the optimal region \((x_c, x_a)\) that envelops the optimal point is available. In this phase, the quadratic inner estimate method proposed by Powell is used to find the exact position.

Step 1. Use the results of phase one. The optimal position \( x^* \) is approximated as following:
\[
x^* = \frac{1}{2} \left( x_a^2 - x_c^2 \right) FA + \left( x_a^2 - x_c^2 \right) FB + \left( x_a^2 - x_c^2 \right) FC \\
2 \left( x_c - x_a \right) FA + \left( x_a - x_c \right) FB + \left( x_b - x_a \right) FC
\]

Step 2. Allow \( x_m = \{ x_i \mid f(x_i) = \text{Min} (FA, FB, FC) \} \)
- If \( \| x^* - x_m \| < \varepsilon \) or \( |f(x^*) - f(x_m)| < \varepsilon \) then stop and back to main program.
Otherwise, discard the worst one of \( x_a, x_b, x_c \) and add the point \( x^* \). Rearrange the three points to the order \( x_a > x_b > x_c \). and back to step 1.

4. Pull Back into The Feasible Region

Take the optimal step length along the searching direction \( d \). The new solution reduces the value of objective function and keep it feasible for the approximate problem. As for the original problem, the new solution may stretch out the feasible region and cause an unfeasible solution. If the situation occurs, keep the value of non-basic variable because of their decreasing property for the objective function, then using the Newton method to pull the unfeasible solution back into the feasible region by changing the value of basic variables with the following operations.

Allow the following \( m \) constraints

\[
h_i(x) = 0 \quad i = 1, \ldots, m.
\]

Then

\[
x^{i+1} = x^i - \nabla h^{-1}(x^i)h(x^i).
\]

The process repeats itself until \( \| x^{i+1} - x^i \| < \epsilon \). If the stopping condition is satisfied, the new solutions stay out of the feasible region. The step length must be cut to a shorter once and go back to the pulling process.

5. Three Search Directions

For the reduced gradient method, the direction \( d \) defined in equation (3) is the original search direction. It processes some good properties for searching a better solution. The efficiency and effectiveness can be improved if the search direction modified by some other well-known methods. Such as the concept of Conjugate direction that has been proven that in a quadratic function of \( n \) variables, after \( n \) times optimal line search along a set of \( n \) conjugate directions, the process will converge to the optimal solution [18]. Although the Conjugate direction method is effective for the quadratic problems, the set of \( n \) Conjugate directions is difficult to know before we start processing the problem. While this concept is normally be used by researchers as they developed the searching method. Fletcher and Reeves [4] modified the Cauchy's method searching the next solution along the gradient of objective
function, to the direction that conjugate to the previous once. Finally, the searching direction can be expressed as the following equation:

$$ s^k = -g^k + \frac{g^k - g^{k-1}}{\|g^k - g^{k-1}\|} s^{k-1}, $$

In which $g^k$ and $s^k$ are the gradient of the objective function and the searching direction at iteration $k$, respectively.

Polak and Ribiere [19] remodified the above direction and developed another approximated one as equation (4). In this research, equation (4) is employed to modify the search direction for the GRG algorithm prior to the new iteration be performed.

$$ s^k = -g^k + \frac{(g^k - g^{k-1})' (g^k)}{\|g^k - g^{k-1}\|} s^{k-1} $$

(4)

Another efficient search method for non-constraint problem is the Newton's method which can be expressed simply as the following equation.

$$ x^{k+1} = x^k - \alpha_k H^{-1}(x^k) \nabla f(x^k) $$

Where the $H^{-1}(x^k)$ denotes the Hessian matrix for objective function $f$ at point $x^k$. Since the Hessian matrix requires the two-degree differential operation on objective function, it must cost much time. Consequently, the approximated approaches for the Hessian matrix are studied and proposed. Broyden, Fletcher, Goldfard and Shanno four researchers provided an approximate estimation, usually called BFGS method, for the Hessian Matrix whenever we need to find it for the next iteration. The approximated estimation approach can be expressed as follows:

$$ H_{k+1} = H_k + \left(1 + \frac{q_k H_k q_k}{q_k^T p_k} \right) p_k q_k^T + \frac{q_k H_k q_k}{q_k^T q_k} p_k q_k^T $$

(5)

In which, $p_k = x_{k+1} - x_k$, $q_k = g_{k+1} - g_k$, and matrix $H_{k+1}$ represents the approximated Hessian Matrix at iteration $k+1$ which is used at $k+2$ iteration. The approximated Hessian Matrix has been proven that it will always have the property of positive definition. Hence, by walking a little distance along the direction $-H^{-1}_k(x^k) \nabla f(x^k)$, the value of objective function can be reduced.
6. The Main Steps in GRG Algorithm

After the discussions in the previous sections, the main process for the GRG method can be summarized as the following steps.

Step 1. Find a feasible solution and divided it into basic and non-basic subsets and denoted them with $\bar{x}^k$ and $\bar{x}^k$, respectively.

This step can be handled by GRG algorithm itself. However, the objective function is defined as the sum of square of the constraints. The processes stop as the values of the objective function equal zero. Then, the solution is an initial feasible solution for the original problem.

Step 2. Find the search direction.

(a) Using equation (2) to calculate the gradient for non-basic variable. Consider the boundary condition for the variable and take an adequate modification as follows:

\[
\bar{d} = \begin{cases} 
-\nabla \tilde{f} & \text{if } \nabla \tilde{f} > 0 \text{ and } \bar{x} - \bar{x}_l > \varepsilon \\
-\nabla \tilde{f} & \text{if } \nabla \tilde{f} < 0 \text{ and } \bar{x}_u - \bar{x} > \varepsilon \\
0 & \text{Otherwise}
\end{cases}
\]

(b) Checking the optimal condition. If $|\bar{d}| < \varepsilon$ then stop. Otherwise, modify $\bar{d}$ with equation (4) or (5).

Using equation (1) to calculate $\hat{d}$.

Finally, constructing the searching direction $d$.

Step 3. Perform a line search according to the two-phases method discussed in section 3.

Step 4. Check the feasibility.

If at least one constraint violates the feasibility condition, use the Newton method discussed in section 4 to pull the solution back to the feasible region.

Step 5. Changing the basis.
If variable approach to its boundary limits, e.g. $x_k - x_l < \varepsilon$ or $x_{u_k} - x_k < \varepsilon$ , The variable must leave out the basic variable set and become a nonbasic variable. Another non-basic variable with the largest absolute value of gradient is selected to enter the basic variable set. The corresponding basic and non-basic matrices $B$ and $A$ must also be changed. Then go back to step 2.

7. Experimental Results

Four factors [16] are usually considered to compare algorithms. Such as effectiveness discusses the ability of each algorithm to converge the searching process to a steady state for the testing problems; efficiency that refers to the execution time for each algorithm to converge the searching process on the testing problems; memory space occupation, which considers the necessary space for each algorithm to memorize the information while they perform their processes; precision that is the values of the sum of the deviation of the objective function between the experimental and theoretical results and the violation of the constraints. The three search directions are involved in the GRG algorithm and implemented on a set of twenty five selected testing problems in which seven with linear constraints the other eighteen's with non-linear constraints. The source of the problems, the execution times on a personal computer and the sum of deviation between theoretical and experimental value of objective function and the violation of feasibility's for each searching direction are described in Table 1. According to table 1, five problems do not converge to a stable point within a reasonable time when GRG algorithm with the original gradient direction, while two same problems do not converge to a stable point when GRG algorithm using the approximated Conjugate or BFGS to modify the original gradient direction. Owing to the same converging condition for the three searching direction, from the effectiveness perspective that the GRG algorithm with the approximated Conjugate or BFGS approaches are better than merely using the original gradient for searching direction. From the execution time in Table 1, eight problems' execution time for GRG with gradient direction are longer than the average time of three directions. Adding them with the five non-converging problems, there are thirteen problems' execution time longer then average time. In the same manner, thirteen and nine problems' execution time are longer than the average time for GRG with approximated Conjugate and BFGS directions, respectively. From above discussion, GRG
with the approximated BFGS direction have the best efficiency. As for the space occupation, the problems must be changed to the form of standard simplex method before implementing the GRG algorithm on it. Assume that there are $n$ variables and $m$ constraints in a problem, another $m$ slack or surplus variables should be needed for transferring the problem to the standard form. Consequently, $(m+1)(n+m)$ units of space is occupied for the GRG algorithm. Since the approximated conjugate directions have to memorize the searching direction just used, the BFGS directions must also memorize the estimated Hessian matrix, the space units for each approximate direction is $(m+1)(n+m)+n$ and $(m+1)(n+m)+n^2$, respectively. Hence, the GRG with approximate BFGS direction costs much memory space when handling a larger problem. The differences of precision at converging point for the three directions are insignificant. From the data in the parenthesis of Table 1, the number of problems that the total violation is larger than the average value for the gradient, Conjugate and BFGS are 7,6 and 7, respectively. One interesting phenomenon is that regardless of whether the problem converges to a good or bad point, the three searching directions converge to a similar point that causes a similar violation.

8. Conclusion

Many algorithms have been used to solve general nonlinear problems. The algorithm developed herein is based on the concept of reduced gradient, and is an effective and efficient approach. This algorithm can be improved when applying the concept of Conjugate or Newton descents directions to modify the original gradient of GRG method. However, two problems of four Dembo’s [20] geometry problems can not converge to a stable point. In the other testing problems, the better properties of Conjugate and Newton directions have been expressed completely. Nevertheless, this approach can be used to improve other algorithms which have a similar search mechanism.

References

2. G. Hadley, 1962, Linear programming. Addison-Wesley, USA.


A Study of Generalized Reduced Gradient Method with Different Search Directions


Table 1  The source of the testing problems, the execution time and total violation for each problem using GRG algorithm with three different search directions.

<table>
<thead>
<tr>
<th>No</th>
<th>Source</th>
<th># Variables</th>
<th># Constraints</th>
<th>Gradient</th>
<th>Conjugate</th>
<th>BFGS</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ryan (1974)</td>
<td>3</td>
<td>8</td>
<td>0.063(^a) (0.0)</td>
<td>0.064 (0.0)</td>
<td>0.068 (0.0)</td>
<td>0.065 (0.0)</td>
</tr>
<tr>
<td>2</td>
<td>Colville (1969) #4</td>
<td>4</td>
<td>8</td>
<td>----- (c)</td>
<td>0.759 (0.0)</td>
<td>0.848 (0.0)</td>
<td>0.803 (0.0)</td>
</tr>
<tr>
<td>3</td>
<td>Colville (1969) #1</td>
<td>5</td>
<td>15</td>
<td>0.381 (0.0)</td>
<td>0.470 (0.0)</td>
<td>0.430 (0.0)</td>
<td>0.427 (0.0)</td>
</tr>
<tr>
<td>4</td>
<td>Himmelblau (1972) #17</td>
<td>10</td>
<td>20</td>
<td>0.201 (0.0)</td>
<td>0.201 (0.0)</td>
<td>0.201 (0.0)</td>
<td>0.201 (0.0)</td>
</tr>
<tr>
<td>5</td>
<td>Bracken (1968)</td>
<td>10</td>
<td>13</td>
<td>1.757 ((2.741\times10^{-5}))</td>
<td>1.780 ((2.741\times10^{-5}))</td>
<td>1.834 ((2.741\times10^{-5}))</td>
<td>1.790 ((2.741\times10^{-5}))</td>
</tr>
<tr>
<td>6</td>
<td>Colville (1969) #7</td>
<td>16</td>
<td>40</td>
<td>7.067 ((1.638\times10^{-5}))</td>
<td>1.887 ((1.726\times10^{-5}))</td>
<td>2.506 ((1.891\times10^{-5}))</td>
<td>3.820 ((1.751\times10^{-5}))</td>
</tr>
<tr>
<td>7</td>
<td>Miele (1969) #7</td>
<td>2</td>
<td>7</td>
<td>0.046 ((1.455\times10^{-5}))</td>
<td>0.042 ((3.221\times10^{-6}))</td>
<td>0.041 ((3.220\times10^{-6}))</td>
<td>0.043 ((2.632\times10^{-6}))</td>
</tr>
<tr>
<td>8</td>
<td>Himmelblau (1972) #3</td>
<td>2</td>
<td>7</td>
<td>0.082 (0.0)</td>
<td>0.092 (0.0)</td>
<td>0.099 (0.0)</td>
<td>0.091 (0.0)</td>
</tr>
<tr>
<td>9</td>
<td>Rosenbrock (1960)</td>
<td>2</td>
<td>3</td>
<td>1.792 (0.0)</td>
<td>0.068 (0.0)</td>
<td>0.067 (0.0)</td>
<td>0.642 (0.0)</td>
</tr>
<tr>
<td>10</td>
<td>Himmelblau (1972) #5</td>
<td>3</td>
<td>5</td>
<td>2.958 ((1.866\times10^{-5}))</td>
<td>0.531 ((1.998\times10^{-5}))</td>
<td>1.610 ((2.000\times10^{-5}))</td>
<td>1.690 ((1.961\times10^{-5}))</td>
</tr>
<tr>
<td>11</td>
<td>Colville (1969) #8</td>
<td>3</td>
<td>20</td>
<td>1.109 (0.0)</td>
<td>1.024 ((2.179\times10^{-12}))</td>
<td>0.688 (0.0)</td>
<td>0.940 ((7.26\times10^{-13}))</td>
</tr>
<tr>
<td>12</td>
<td>Rosen (1965)</td>
<td>4</td>
<td>3</td>
<td>2.125 ((4.459\times10^{-5}))</td>
<td>1.533 ((4.460\times10^{-5}))</td>
<td>1.203 ((2.525\times10^{-5}))</td>
<td>1.620 ((3.723\times10^{-5}))</td>
</tr>
<tr>
<td>13</td>
<td>Himmelblau (1972) #9</td>
<td>4</td>
<td>6</td>
<td>-----</td>
<td>10.514 (0.0)</td>
<td>3.527 (0.0)</td>
<td>7.021 (0.0)</td>
</tr>
<tr>
<td>14</td>
<td>Miele (1969) #4</td>
<td>5</td>
<td>3</td>
<td>0.171 ((4.616\times10^{-5}))</td>
<td>0.226 ((2.100\times10^{-5}))</td>
<td>0.144 ((3.562\times10^{-5}))</td>
<td>0.127 ((3.426\times10^{-5}))</td>
</tr>
<tr>
<td>15</td>
<td>Colville (1969) #3</td>
<td>5</td>
<td>16</td>
<td>0.129 ((3.349\times10^{-5}))</td>
<td>0.099 ((4.639\times10^{-5}))</td>
<td>0.153 ((3.364\times10^{-5}))</td>
<td>0.127 ((2.392\times10^{-5}))</td>
</tr>
<tr>
<td>16</td>
<td>Powell (1964)</td>
<td>5</td>
<td>3</td>
<td>0.416 ((6.794\times10^{-5}))</td>
<td>0.592 ((3.654\times10^{-5}))</td>
<td>0.346 ((3.809\times10^{-5}))</td>
<td>0.451 ((2.710\times10^{-5}))</td>
</tr>
<tr>
<td>17</td>
<td>Miele (1969) #3</td>
<td>5</td>
<td>2</td>
<td>4.648 ((4.085\times10^{-5}))</td>
<td>0.730 ((1.581\times10^{-5}))</td>
<td>0.640 ((1.648\times10^{-5}))</td>
<td>2.086 ((1.212\times10^{-5}))</td>
</tr>
<tr>
<td>18</td>
<td>Dembo (1976) #4</td>
<td>8</td>
<td>12</td>
<td>1.250 ((1.970\times10^{-5}))</td>
<td>0.474 ((2.701\times10^{-1}))</td>
<td>2.438 ((1.143\times10^{-1}))</td>
<td>1.387 ((5.366\times10^{-2}))</td>
</tr>
<tr>
<td>19</td>
<td>Himmelblau (1972)</td>
<td>9</td>
<td>14</td>
<td>10.179 ((1.236\times10^{-5}))</td>
<td>36.355 ((1.149\times10^{-5}))</td>
<td>23.833 ((1.130\times10^{-5}))</td>
<td>23.450 ((1.171\times10^{-5}))</td>
</tr>
<tr>
<td>20</td>
<td>Colville (1969) #2</td>
<td>15</td>
<td>20</td>
<td>14.249 (0.0)</td>
<td>18.907 (0.0)</td>
<td>12.301 (0.0)</td>
<td>15.152 (0.0)</td>
</tr>
<tr>
<td>21</td>
<td>Himmelblau (1972)</td>
<td>20</td>
<td>17</td>
<td>-----</td>
<td>84.565 ((2.941\times10^{-1}))</td>
<td>35.671 ((8.417\times10^{-1}))</td>
<td>60.118 ((1.891\times10^{-1}))</td>
</tr>
<tr>
<td>22</td>
<td>Dembo (1976) #1</td>
<td>11</td>
<td>15</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>24</td>
<td>Dembo (1976) #5</td>
<td>8</td>
<td>14</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>25</td>
<td>Himmelblau (1972)</td>
<td>30</td>
<td>75</td>
<td>35.613 ((1.904\times10^{7}))</td>
<td>35.585 ((1.164\times10^{7}))</td>
<td>39.072 ((1.748\times10^{7}))</td>
<td>36.756 ((1.605\times10^{7}))</td>
</tr>
</tbody>
</table>

Note: a, execution time  b, total violation at steady state  c, fail to converge
不同尋求方向對梯度消減法執行成效之研究

*李鴻濤 **陳秀華 ***康鶴耀
*國立勤益技術學院 工業工程與管理系
**修平技術學院工業管理系
連絡人 E-mail：leeht@chinyi.ncit.edu.tw

摘要

在求解非線性數學規劃問題時，梯度消減法(generalized reduced gradient method)是一有效且準確的方法。為了擴展此方法，使之能更廣泛的應用在不同類型問題之求解上，本研究在現有之梯度消減法上加入共軛梯度(conjugate gradient)及近似牛頓(quasi-newton)等搜尋方法。在搜尋步長上則採用二次函數估計與兩階段線性尋求方法決定之。本研究測試二十五個難度很高的非線性規劃問題，以有效性、效率性、儲存空間需求與解答之準確性等四評估準則來比較一般梯度方向、共軛梯度、及近似牛頓等三種搜尋方向之成效。

關鍵字：非線性規劃，演算法，最佳化，搜尋。