A hybrid method for solving a general constrained optimization problem (Decision Theory in Mathematical Modelling)

Author(s)
Sagaka, Nobuko; Fukushima, Masao

Citation
数理解析研究所講究録 (1999), 1079: 1-14

Issue Date
1999-02

URL
http://hdl.handle.net/2433/62700

Type
Departmental Bulletin Paper

Textversion
publisher

Kyoto University
A hybrid method for solving a general constrained optimization problem

November 16, 1998

Nobuko SAGARA (相良信子)
Department of Management, Aichi University

Masao FUKUSHIMA (福嶋雅夫)
Department of Applied Mathematics and Physics, Kyoto University

1. Introduction

Recently we have proposed an algorithm for solving nonlinear least squares problems with linear constraints [15]. The method proposed there successively constructs trust region constraints, which are ellipsoids centered at the iterative points, in such a way that they lie in the relative interior of the feasible region. The purpose of this paper is to generalize the results of [15] to nonlinear equality constrained problems with non-negative variables.

Over the last decade, various methods using a trust region strategy have been studied for constrained nonlinear optimization problems, including box constrained problems [5, 6, 11, 14, 16] and nonlinear equality constrained problems [2, 4, 7, 8, 9, 13, 17, 19]. In particular, when dealing with equality constraints, adding a trust region constraint directly to the problem may result in an infeasible subproblem. To overcome this difficulty, two approaches have been introduced. The first approach, which was first proposed by Vardi [17], and later used by Byrd, Schnabel and Shultz [2] and M. El-Alem [7], relaxes the linearized equality constraints so as to intersect a trust region constraint. Specifically we first compute the vertical component of a trial step in the range space of the matrix of equality constraint gradients at each iteration and then the horizontal (tangential) component in the null space. The second approach, which was introduced by Celis, Dennis and

1This work was supported in part by the Scientific Grant-in-Aid C-59 from Aichi University.
Tapia [4] and later studied by Powell and Yuan [13], iteratively solves quadratic programming subproblems with some additional constraints using a standard trust region strategy. We adopt the second approach in this paper.

In this paper, we consider the following nonlinearly constrained optimization problem:

\[
\text{minimize } f(x) \quad \text{subject to } \quad c(x) = 0, \quad x \geq 0,
\]

where \( x \in \mathbb{R}^n \) and \( c(x) = (c_1(x), c_2(x), \ldots, c_m(x))^T \). We assume that \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) and \( c_j : \mathbb{R}^n \rightarrow \mathbb{R}, \quad j = 1, \ldots, m \), are continuously differentiable. Note that any optimization problem involving inequality constraints can be transformed into the form (1.1) by using slack variables.

The main aim of this paper is to extend the methods presented in [14, 15] to the general constrained optimization problem (1.1). Similar to the methods of [14, 15], the proposed method constructs trust region constraints that are ellipsoids centered at the iterative points, in such a way that they lie in the interior of the non-negative orthant. Thus the method belongs to the class of interior point methods. However, since solutions of the problem are usually on the boundary of the non-negative orthant, the trust region ellipses may tend to be arbitrarily thin and, as a result, we eventually suffer from numerical instability. To avoid this difficulty, we incorporate the idea of active set strategy into the method. With such a modification, we may still expect that the method retains the advantage of the interior point method, as observed in our previous papers [14, 15] for problems with simple bound constraints on the variables, and problems with linear inequality constraints and non-negative variables.

The paper is organized as follows. Section 2 describes the algorithm. In Section 3, we report some computational results.

2. Description of the Algorithm

Suppose that \( x \) is a current point such that \( x > 0 \). Let \( p \) denote the vector which determines the next point \( x^+ \) from the current point \( x \), that is, \( x^+ = x + p \).

We consider the following subproblem involving a trust region constraint:

\[
\text{minimize}_p \quad g(x)^T p + \frac{1}{2} p^T W p \quad \tag{2.1a}
\]
subject to \[ \| c(x) + A(x)^T p \| \leq \theta, \] \[ \| D(x)p \| \leq \Delta, \]

where \( g(x) = \nabla f(x), \) \( A(x) = \nabla c(x), \) \( D(x) = \text{diag}(1/x_i), \) \( \theta \) and \( \Delta \) are parameters such that \( \theta > 0 \) and \( 0 < \Delta < 1, \) and \( W \) is an \( n \times n \) symmetric matrix. Note that \( \theta, \Delta \) and \( W \) are dependent on the iteration. In particular, \( \theta \) is supposed to satisfy the inequalities

\[
\min_{\| D(x)p \| \leq b_1 \Delta} \| c(x) + A(x)^T p \| \leq \theta \leq \min_{\| D(x)p \| \leq b_2 \Delta} \| c(x) + A(x)^T p \|,
\]

where \( b_1 \) and \( b_2 \) are given constants such that \( 0 \leq b_2 \leq b_1 \leq 1. \) Note that if \( p \) satisfies (2.1c) then \( x^+ := x + p \) remains in the positive orthant, i.e., \( x^+ > 0 \) (see [1]).

The role of the constraint (2.1b) may be explained as follows. When using the trust region constraint \( \| D(x)p \| \leq \Delta, \) the linearized constraints \( c(x) + A(x)^T p = 0 \) may have no solution within the trust region. Attempts to overcome this difficulty have been made by several authors [13, 17, 2, 4]. In particular, introducing the parameter \( \theta \) satisfying (2.2) was proposed by Powell and Yuan [13]. Note that, if we assume \( b_1 = 1 \) and \( b_2 = 0, \) then (2.2) reduces to

\[
\min_{\| D(x)p \| \leq \Delta} \| c(x) + A(x)^T p \| \leq \theta \leq \| c(x) \|,
\]

which ensures the existence of a feasible solution to (2.1)

The trust region constraint \( \| D(x)p \| \leq \Delta \) in (2.1) represents an ellipsoid, which is centered at the current point \( x \) and strictly contained in the interior of the non-negative orthant \( \{ x \in \mathbb{R}^n | x \geq 0 \}. \) However, when the current point is close to the boundary of the non-negative orthant, the trust region ellipsoid becomes thin and solution of (2.1) suffers from numerical instability. To overcome this difficulty, we modify (2.1) using the idea of active set strategy [10] in constrained optimization.

For a current point \( x > 0, \) we define the set of indices

\[
I = \{ i | x_i \geq \epsilon_1 \},
\]

where \( \epsilon_1 \) is a sufficiently small positive constant. We also denote \( I = \{ 1, 2, ..., n \} - I. \) According to the above definition, we partition the vectors \( x, p, g(x) \) and the matrices \( W, A(x) \) as

\[
x = \begin{bmatrix} x_I \\ x_{\overline{I}} \end{bmatrix}, \quad p = \begin{bmatrix} p_I \\ p_{\overline{I}} \end{bmatrix}, \quad g(x) = \begin{bmatrix} g_I(x) \\ g_{\overline{I}}(x) \end{bmatrix}, \quad W = \begin{bmatrix} W_{II} & W_{I\overline{I}} \\ W_{\overline{I}I} & W_{\overline{I}\overline{I}} \end{bmatrix}, \quad A(x) = \begin{bmatrix} A_I(x) \\ A_{\overline{I}}(x) \end{bmatrix}.
\]
By adding the extra constraints \( p_{|} = 0 \) to (2.1), we get the following problem:

\[
\begin{align*}
& \text{minimize}_{p_{I}} \quad g_{I}(x)^{T}p_{I} + \frac{1}{2}p_{I}^{T}W_{II}p_{I} \\
& \text{subject to} \quad \| c(x) + A_{I}(x)^{T}p_{I} \| \leq \theta, \\
& \quad \| D_{I}(x)p_{I} \| \leq \Delta,
\end{align*}
\]

(2.4)

where \( \theta \) is chosen, similar to (2.2), to satisfy the condition

\[
\min_{\|D_{I}(x)p_{I}\| \leq b_{2}\Delta} \| c(x) + A_{I}(x)^{T}p_{I} \| \leq \theta \leq \min_{\|D_{I}(x)p_{I}\| \leq b_{1}\Delta} \| c(x) + A_{I}(x)^{T}p_{I} \|, \tag{2.5}
\]

and \( b_{1}, b_{2} \) and \( \Delta \) are parameters such that \( 0 \leq b_{2} \leq b_{1} \leq 1 \) and \( 0 < \Delta < 1 \), and \( D_{I}(x) \) is the diagonal submatrix of \( D(x) \) with elements \( 1/x_{i}, i \in I \).

Let \( p_{I} \) be a solution of (2.4). To test whether we should accept the trial step \( p = (p_{I}, 0)^{T} \), we use Fletcher's differentiable exact penalty function

\[
\phi(x; \sigma) = f(x) - \lambda(x)^{T}c(x) + \sigma\|c(x)\|^{2}, \tag{2.6}
\]

where \( \sigma > 0 \) is a penalty parameter and \( \lambda(x) \) is an estimate of the vector of Lagrange multipliers defined by

\[
\lambda(x) = \left( A_{I}(x)^{T}A_{I}(x) \right)^{+}A_{I}(x)^{T}g_{I}(x), \tag{2.7}
\]

where \( A^{+} \) is the generalized inverse of the matrix \( A \). The vector \( \lambda(x) \) minimizes the sum of squared residuals of the Kuhn-Tucker conditions for (2.4), that is,

\[
\lambda(x) = \arg\min_{\lambda \in \mathbb{R}^{m}}\|g_{I}(x) - A_{I}(x)\lambda\|^{2}.
\]

We define the predicted reduction of \( \phi(x + p; \sigma) - \phi(x; \sigma) \) by

\[
\psi_{I}(x, p) = (g_{I}(x) - A_{I}(x)\lambda(x))^{T}p_{I} + \frac{1}{2}p_{I}^{T}W_{II}p_{I} \tag{2.8}
\]

\[
- [\lambda(x + p) - \lambda(x)]^{T}\left[ c(x) + \frac{1}{2}A_{I}(x)^{T}p_{I} \right] \\
+ \sigma \left( \|c(x) + A_{I}(x)^{T}p_{I}\|^{2} - \|c(x)\|^{2} \right),
\]

where \( \hat{p}_{I} \) is the orthogonal projection of \( p_{I} \) onto the null space of \( A_{I}(x)^{T} \). Note that a similar formula for the predicted reduction has been used in [13] and [7]. Here the parameter \( \sigma \) needs to
be chosen so that $\psi_I(x, p) < 0$. More specifically, if the inequality

$$\psi_I(x, p) \leq \frac{1}{2} \sigma \left( ||c(x) + A_I(x)T_p||^2 - ||c(x)||^2 \right)$$

(2.9)
is satisfied, then $\sigma$ remains the same. Otherwise, increase $\sigma$ to the value

$$\sigma := 2\sigma + \max \left\{ 0, \frac{2\psi_I(x, p)}{||c(x)||^2 - ||c(x) + A_I(x)T_p||^2} \right\},$$

(2.10)

which ensures that $\psi_I(x, p)$ with the updated $\sigma$ satisfies (2.9). Note that the right-hand side of (2.9) is negative whenever $p_I \neq 0$. This can be shown in a way similar to Lemma 3.3 in [13].

Next we compute the ratio

$$\rho \equiv \frac{\phi(x + p; \sigma) - \phi(x; \sigma)}{\psi_I(x, p)}.$$

(2.11)

If $\phi(x + p; \sigma)$ is sufficiently smaller than $\phi(x; \sigma)$, then we accept $p$ to determine the next point. Otherwise, we halve $\Delta$ and solve subproblem (2.4) again. More precisely, let $0 < \mu < \eta < 1$, $\gamma > 1$ and $0 < \Delta_{\text{max}} < 1$ be given constants. Compute the ratio $\rho$ defined by (2.11). If $\rho < \mu$, then put $x^+ := x$ and $\Delta^+ := 1/2\Delta$, and then solve subproblem (2.4); if $\mu \leq \rho < \eta$, then put $x^+ := x + p$ and $\Delta^+ := \Delta$; if $\rho \geq \eta$, then put $x^+ := x + p$ and $\Delta^+ := \min(\gamma\Delta, \Delta_{\text{max}})$. To test whether the new point $x := x^+$ is an approximate optimal solution associated with the current index set $I$, we check the inequality

$$||c(x)|| + ||g_I(x) - A_I(x)\lambda(x)|| < \epsilon_2,$$

(2.12)

where $\epsilon_2 > 0$ is a predetermined positive number significantly smaller than $\epsilon_1$. If (2.12) is not satisfied, we update the index set $I$ from the new iterative point $x$. As a result, we have subproblem (2.4) corresponding to the new index set $I$. In this manner, we repeatedly solve subproblems (2.4) as long as (2.12) does not hold. On the other hand, if (2.12) is satisfied, then we proceed to examine optimality of the point $x$ for the original problem (1.1). Namely, we compute an estimate of Lagrange multipliers $\lambda(x)$ by (2.7) and check if $\lambda(x)$ satisfies the inequality

$$g_I(x) - A_I(x)\lambda(x) > -\epsilon_3 e,$$

(2.13)

where $\epsilon_3$ is a sufficiently small positive constant and $e$ is a vector of appropriate dimension whose components are all unity. When the condition (2.13) is violated, we choose an index $i$ from $\overline{I}$ such
that

\[ i := \text{argmax} \{ |(g_I(x) - A_I(x)\lambda(x))_i| \mid (g_I(x) - A_I(x)\lambda(x))_i \leq -\epsilon_3, i \in \overline{I} \}, \]  

(2.14)

add it to the current inactive index set \( I \), namely,

\[ I := I \cup \{ i \}, \quad \overline{I} := \overline{I} - \{ i \}, \]

and try to improve the current solution by solving subproblem (2.4) with the updated index set \( I \).

If the point \( x \) satisfies both conditions (2.12) and (2.13), then we call it an \( \epsilon \)-optimal solution for problem (1.1), with \( \epsilon = (\epsilon_1, \epsilon_2, \epsilon_3) \).

We are now ready to state an algorithm for finding an \( \epsilon \)-optimal solution of (1.1).

**Algorithm**

**Initialize:** Choose sufficiently small positive constants \( \epsilon_i, i = 1, 2, 3 \) and parameters \( \mu, \eta, \gamma \) and \( \Delta_{\text{max}} \) such that \( 0 < \mu < \eta < 1, \gamma > 1 \) and \( 0 < \Delta_{\text{max}} < 1 \). Also, choose parameters \( b_1 \) and \( b_2 \) with \( 0 < b_2 \leq b_1 < 1 \) to get \( \theta \) in subproblem (2.4). Let \( x \geq \epsilon_1 e, \ \Delta \in (0, 1), \) and \( \sigma > 0 \) be an initial feasible interior point, an initial trust region radius, and an initial penalty parameter, respectively. Let \( I \) be the index set defined by (2.3) and \( \overline{I} \) be the complement of \( I \).

**while** \( x \) is not \( \epsilon \)-optimal **do**

**begin**

1. **while** \( \|c(x)\| + \|g_I(x) - A_I(x)\lambda(x)\| \geq \epsilon_2 \) **do**

2. **begin**

3. Solve subproblem (2.4) to find \( p_I \);

4. \( p_I := 0 \);

5. Compute \( \psi_I(x,p) \) by (2.8)

6. **if** (2.9) does not hold **then**

7. **begin**

8. Update \( \sigma \) by (2.10);

9. **end**

10. **endif**

11. Compute \( \rho \) by (2.11)

**end**

**end while**
if $\rho \geq \mu$ then
    begin
        $x := x + p$
        Update matrix $W$
        if $\rho \geq \eta$ then
            $\Delta := \min(\gamma \Delta, \Delta_{\text{max}})$
            endif
    end
    $I := I - \{i \in I | x_i < \epsilon_1\}$
else
    $\Delta := \frac{1}{2} \Delta$
    go to 2
endif
end

if (2.13) is violated then
    (Comment: if (2.13) holds, the whole algorithm is terminated with an $\epsilon$-optimal solution.)
    begin
        $i := \arg \max \{ |(g_{\overline{I}}(x) - A_{\overline{I}}(x)\lambda(x))_i| \mid (g_{\overline{I}}(x) - A_{\overline{I}}(x)\lambda(x))_i < -\epsilon_3, i \in \overline{I}\}$ ;
        $I := I \cup \{i\}$
    end
endif
end

Note that the sequence of iterates $x$ generated by this algorithm lies in the positive orthant. Here, the inner loop tries to find the solution of the equality constrained optimization problem with some index set $I$. 
3. Numerical Experiments

We executed numerical experiments with the algorithm proposed in Section 2. In this algorithm, the most time consuming task is to solve subproblem (2.4) at each iteration, which is the problem of minimizing a convex quadratic function subject to two quadratic constraints. To solve (2.4) we used the approach by Zhang [18] to reformulate the problem into a univariate nonlinear equation which is continuous, at least piecewise differentiable and monotone.

The program of the algorithm was coded in Fortran77. The computation was carried out using double precision arithmetic on a FACOM-M382 Computer at the Data Processing Center of Kyoto University.

In order to examine performance of the proposed algorithm, we have solved the following four test problems. These test problems except Example 3 have been constructed by modifying the problems in [3]. In particular, the last problem is an expansion of the large scale problem No.7 in [3]. Throughout the experiments, we set the parameter values as follows: \(\epsilon_1 = 0.001\), \(\epsilon_2 = 0.01\), \(\epsilon_3 = 0.001\), \(\mu = 0.2\), \(\eta = 0.6\), \(\gamma = 1.2\) and \(\Delta_{\text{max}} = 0.99\). Also we set \(b_1 = 0.8\) and \(b_2 = 0.2\), and let \(\theta\) be the mean value of the both sides of (2.5). The symmetric matrix \(W_{II}\) in subproblem (2.4) was updated by BFGS formula.

Example 1.

\[
\begin{align*}
\text{minimize } f(x) & = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^2 + (x_3 - x_4)^4 + (x_4 - x_5)^4 \\
& + (x_5 - x_6)^4 + (x_6 - x_7)^4 + (x_7 - x_8)^4 + (x_8 - x_9)^4 + (x_9 - x_{10})^4, \\
\text{subject to } c_1(x) & = x_1 + x_2^2 + x_3^3 - 2 - \sqrt{18} = 0 \\
c_2(x) & = x_2 - x_3^3 + x_4 + 2 - \sqrt{8} = 0 \\
c_3(x) & = x_1x_5 - 2 = 0 \\
c_4(x) & = x_2x_6 - 3 = 0 \\
c_5(x) & = x_3x_7 - 4 = 0 \\
c_6(x) & = x_4x_8 - 5 = 0 \\
c_7(x) & = x_5x_9 - 6 = 0
\end{align*}
\]
\[ c_8(x) = x_6x_{10} - 7 = 0 \]
\[ x_i \geq 0, \ i = 1, 2, ..., 10. \]

The numerical results for several initial points \( x^0 \) are shown given in the following table.

<table>
<thead>
<tr>
<th>( x^0 )</th>
<th>iter.</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, ..., 1)</td>
<td>11</td>
<td>92</td>
</tr>
<tr>
<td>(2, ..., 2)</td>
<td>9</td>
<td>54</td>
</tr>
<tr>
<td>(3, ..., 3)</td>
<td>13</td>
<td>70</td>
</tr>
<tr>
<td>(4, ..., 4)</td>
<td>13</td>
<td>86</td>
</tr>
<tr>
<td>(5, ..., 5)</td>
<td>17</td>
<td>104</td>
</tr>
<tr>
<td>(6, ..., 6)</td>
<td>18</td>
<td>100</td>
</tr>
<tr>
<td>(7, ..., 7)</td>
<td>19</td>
<td>99</td>
</tr>
<tr>
<td>(8, ..., 8)</td>
<td>20</td>
<td>116</td>
</tr>
</tbody>
</table>

iter. = number of iterations
CPU time = CPU time in msec

Example 2.

\[ \text{minimize } f(x) = x_1x_2x_3 + x_3x_4x_5 + x_5x_6x_7 + \cdots + x_{n-2}x_{n-1}x_n \]
subject to
\[ c_1(x) = x_1x_{n-1} - 1 = 0 \]
\[ c_2(x) = x_2x_{n-1} - 2 = 0 \]
\[ c_3(x) = x_3x_{n-1} - 3 = 0 \]
\[ \vdots \]
\[ c_{m-1}(x) = x_{m-1}x_{m+1} - (m - 1) = 0 \]
\[ c_m(x) = x_m^2 - m = 0 \]
\[ x_i \geq 0, \ i = 1, 2, ..., n, \]

where \( n \) is odd, \( m = \left[ \frac{n}{2} \right] + 1 \) and \([k]\) is the maximum integer not exceeding \( k \). We experimented with an initial vector \( x^0 \in \mathbb{R}^n \) whose elements are all 2.0. The numerical results are shown in the following table.
Example 3.

\[
\begin{align*}
\text{minimize } f(x) &= (x_1 - 1)^2 + (x_1 - x_2)^2 + \cdots + (x_{n-1} - x_n)^2 \\
\text{subject to } c_1(x) &= x_1 + x_2 + \cdots + x_{n-1} - 5n = 0 \\
& \quad c_2(x) = x_1^2 + x_2^2 + \cdots + x_{\frac{n}{2} - 2}^2 - 20n = 0 \\
& \quad c_3(x) = x_{\frac{n}{2} + 3}^2 + \cdots + x_n^2 - 20n = 0 \\
& \quad x_i \geq 0, \quad i = 1, 2, \ldots, n
\end{align*}
\]

where \( n \) is even. We experimented with an initial vector \( x^0 \in \mathbb{R}^n \) whose elements are all 5.0. The numerical results are shown in the following table.
Example 4.

\[
\text{minimize } f(x) = \frac{100}{2} \sum_{j=1}^{L} (x_{5j+2} - x_{5j+1}^2)^2 + \frac{1}{2} \sum_{j=1}^{L} (1 - x_{5j+1})^2
\]

subject to

\[
c_{3k-2}(x) = x_{5k-4}x_{5k-3} - 1.0 + x_{5k-2}^2 = 0
\]

\[
c_{3k-1}(x) = x_{5k-3}^2 + x_{5k-4} - x_{5k-5}^2 = 0
\]

\[
c_{3k}(x) = -x_{5k-4} - x_{5k}^2 + 0.5 = 0, \quad k = 1, 2, ..., L
\]

\[
x_i \geq 0, \quad i = 1, 2, ..., n,
\]

where \( L \) is a positive integer, \( n = 5L \) and \( m = 3L \).

The numerical results are summarized in the following table.

| \( n \) | iter. | CPU time | \(|I|\) |
|--------|-------|----------|--------|
| 10     | 77    | 243      | 8      |
| 20     | 80    | 969      | 14     |
| 30     | 54    | 1898     | 21     |
| 40     | 78    | 6365     | 27     |
| 50     | 75    | 10424    | 34     |
| 60     | 82    | 18882    | 40     |
| 70     | 89    | 33071    | 46     |
| 80     | 87    | 39488    | 53     |
| 85     | 92    | 57826    | 56     |
| 90     | 99    | 57907    | 59     |

iter. = number of iterations

CPU time = CPU time in msec

\(|I| = \) number of indices \( i \) such that \( x_i \geq \epsilon_1 \).
<table>
<thead>
<tr>
<th>$L$</th>
<th>$n$</th>
<th>$m$</th>
<th>$x^0$</th>
<th>iter.</th>
<th>CPU time</th>
<th>$B/A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>25</td>
<td>15</td>
<td>(0.5,...,0.5)</td>
<td>34</td>
<td>1055</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(0.6,...,0.6)</td>
<td>34</td>
<td>1048</td>
<td>0.0019</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(1.0,...,1.0)</td>
<td>37</td>
<td>1256</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(1.5,...,1.5)</td>
<td>37</td>
<td>1415</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(2.0,...,2.0)</td>
<td>38</td>
<td>1239</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>50</td>
<td>30</td>
<td>(0.5,...,0.5)</td>
<td>44</td>
<td>8547</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(0.6,...,0.6)</td>
<td>44</td>
<td>7813</td>
<td>0.0008</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(1.0,...,1.0)</td>
<td>47</td>
<td>9102</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(1.5,...,1.5)</td>
<td>49</td>
<td>9942</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(1.9,...,1.9)</td>
<td>51</td>
<td>11442</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>60</td>
<td>(0.5,...,0.5)</td>
<td>57</td>
<td>80684</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(0.6,...,0.6)</td>
<td>56</td>
<td>82920</td>
<td>0.0003</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(0.9,...,0.9)</td>
<td>58</td>
<td>90307</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>250</td>
<td>150</td>
<td>(0.5,...,0.5)</td>
<td>77</td>
<td>1834208</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(0.6,...,0.6)</td>
<td>76</td>
<td>1822075</td>
<td>0.0001</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(0.8,...,0.8)</td>
<td>77</td>
<td>1892914</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>300</td>
<td>180</td>
<td>(0.6,...,0.6)</td>
<td>81</td>
<td>3374823</td>
<td>0.0001</td>
</tr>
</tbody>
</table>

iter. = number of iterations
CPU time = CPU time in msec to get an $\epsilon$-solution
$A$ = CPU time to get an $\epsilon$-solution
$B = $ CPU time to recover an accurate solution from an $\epsilon$-solution

For this example, we considered a correction phase that recovers more accurate solution from the obtained $\epsilon$-optimal solution, because the proposed method merely finds $\epsilon$-optimal solutions. The CPU time spent ($B$) in the correction phase was always less 1% of the CPU time ($A$) spent to obtain an $\epsilon$-solution. Moreover the ratio $B/A$ decreases as the problem size increases. Over all, the proposed method solved successfully the above test problems even if a starting point was far from the solution or the problem size were large, though it consumed relatively large computation time.
References


