A heuristic iterated-subspace minimization method with pattern search for unconstrained optimization

Ting Wu, Yingsha Yang, Linping Sun, Hu Shao

Department of Mathematics, Nanjing University, Nanjing, 210093, PR China
Department of Mathematics, China University of Mining and Technology, Xuzhou Jiangsu, 221116, PR China

Abstract

Recently, an increasing attention was paid on different procedures for an unconstrained optimization problem when the information of the first derivatives is unavailable or unreliable. In this paper, we consider a heuristic iterated-subspace minimization method with pattern search for solving such unconstrained optimization problems. The proposed method is designed to reduce the total number of function evaluations for the implementation of high-dimensional problems. Meanwhile, it keeps the advantages of general pattern search algorithm, i.e., the information of the derivatives is not needed. At each major iteration of such a method, a low-dimensional manifold, the iterated subspace, is constructed. And an approximate minimizer of the objective function in this manifold is then determined by a pattern search method. Numerical results on some classic test examples are given to show the efficiency of the proposed method in comparison with a conventional pattern search method and a derivative-free method.

1. Introduction

This paper is aimed to propose a method for solving the following unconstrained optimization problem:

\[
\min_{x \in \mathbb{R}^n} f(x)
\]

where \( f \) is assumed to be continuously differentiable and the information of the derivatives of \( f(x) \) is unreliable or unavailable.

Pattern search methods are regarded as a kind of efficient algorithm for solving the derivative-free optimization problems due to their simplicity and usefulness. A lot of classical pattern search methods have been developed for unconstrained optimization such as multidirectional search of Dennis and Torczon [1], generalized pattern search method of Torczon [2], and grid-based framework of Coope and Price [3]. However, the implementation of these methods is relatively time consuming in comparison with the conventional derivative-based algorithms. This is because there are so many search directions in every pattern search process. To overcome this limitation, many researchers improved the pattern search methods by adopting some techniques in iterate acceptance criterion [4,5] or modifying the search direction set [6]. In the latter case, although positive bases [7] are utilized to replace bases in the pattern search methods, there are at least \( n + 1 \) search directions in every round of pattern search. Hence, when a pattern search point (defined in Section 2) is found, we must implement at least \( n + 1 \) times of function value estimations. In view of this, the implementation of the general pattern search methods may be inefficient particularly for high-dimensional problems. Consequently, to reduce the number...
of function evaluations, we consider implementing pattern search in a subspace with a lower dimension, which motivates us to integrate the iterated-subspace minimization methods into the pattern search framework.

A typical representation of previous work on the iterated-subspace methods is the method presented by Cragg and Levy [8]. They proposed a method to execute the minimization step in a \( k \)-dimensional subspace, which included the steepest descent direction. Shortly afterwards, Dennis and Turner [9] minimized a convex quadratic on a subspace by adding an extra vector which is independent on the existing subspace at each iteration. Then, Dixon et al. [10] constructed a 4-dimension subspace which was defined using the steepest descent direction, the Newton direction, and two other directions which were combinations of the two previous steps. Other related work about iterated-subspace minimization methods can also be found in Miele and Cantrell [11], Vinosme [12], Saad [13] and so on. Despite of the diversification for iterated-subspace method, it should be pointed out that the dimension of the "subspace" is lower than the original one. Taking advantage of this characteristic in the iterated-subspace method, the pattern search method can be improved to achieve a higher efficiency for implementation.

The combination of pattern search methods and iterated-subspace minimization methods can overcome the shortcoming of general pattern search methods. Instead of carrying out pattern search in original space, we implement pattern search in a lower-dimensional manifold, the iterated subspace. It is clearly that this technique can reduce the number of iterations and function evaluations.

The rest of this paper is organized as follows. In Sections 2 and 3, we provide a brief description of pattern search methods and iterated-subspace minimization methods. In the next section, we present framework of the new method together with some discussions on its new features. The corresponding convergence analysis is examined in Section 5. In Section 6, we give numerical results to demonstrate the feasibility and effectiveness of the proposed method. Finally, some concluding remarks are offered.

2. Generalized pattern search methods

Generalized pattern search methods for unconstrained optimization generate a sequence of iterates \( \{x^{(k)}\} \) with non-increasing objective function values. At each iteration, the objective function is evaluated at a finite number of trial points on a grid [3] and the purpose is to look for one point which can yield a lower function value than the current iterate. If such a point is found, set it to be the new iterate, and the iteration is called successful; Otherwise, we declare it unsuccessful, record the current iterate (defined as a pattern search point [14]), refine the grid and update the trial points for the next iteration. Torczon [2] showed that generalized pattern search methods can produce a limit point for which the gradient of the objective function is zero. Coope and Price [3] proved the same conclusion from another point of view.

Now we can use a matrix \( M^{(k)} \in \mathbb{R}^{n \times p^{(k)}} \) to indicate the set of trial directions on the \( k \)-th iteration, where the columns of \( M^{(k)} \) are the trial directions and \( p^{(k)} \) is the number of trial directions. \( \Delta_k \) is a rational scale factor. Then the pattern search process can be showed as follows:

**Generalized Pattern Search:**

Given initial iterate \( x^{(0)} \in \mathbb{R}^n, f(x^{(0)}), M^{(0)} \in \mathbb{R}^{n \times p^{(0)}} \) and \( \Delta^{(0)} > 0 \),

While (Stopping conditions do not hold) do

Step 1. Find a step \( s^{(k)} \in \Delta^{(k)}M^{(k)} \) by using Exploratory moves \( (\Delta^{(k)}, M^{(k)}) \).

Step 2. If \( f(x^{(k)} + s^{(k)}) < f(x^{(k)}) \), then \( x^{(k+1)} = x^{(k)} + s^{(k)} \). Otherwise, \( x^{(k+1)} = x^{(k)} \).

Step 3. Update \( (\Delta^{(k)}, M^{(k)}) \) to \( (\Delta^{(k+1)}, M^{(k+1)}) \), \( k = k + 1 \).

**Remark 2.1.** The trial search direction set \( M^{(k)} \) usually contains a basis of \( \mathbb{R}^n \) and its opposite direction. Recently, basis is replaced by positive basis [7] whose non-negative linear combinations span \( \mathbb{R}^n \). Moreover, the number of positive basis’ cardinality is between \( n + 1 \) and \( 2n \). For example, if \( V \) is a basis of \( \mathbb{R}^n \), then

\[
V_+ = [V, - Ve]
\]

is a positive basis of \( \mathbb{R}^n \), where \( e = [1, 1, \ldots, 1]^T \). Hence, the replacement can help to reduce the search directions.

3. Iterated-subspace minimization methods

Generally speaking, a prototype algorithm for iterated-subspace minimization methods can usually be expressed as follows:

**Iterated-subspace minimization methods:**

Given an initial iterate \( x^{(0)} \):

Step 1. Stop with the current iterate \( x^{(k)} \) if convergence tests are satisfied.

Step 2. Determine a full-rank subspace matrix \( Z^{(k)} \in \mathbb{R}^{n \times z^{(k)}} \), where \( z^{(k)} \ll n \).
Step 3. Approximately solve the $z^{(k)}$-dimensional minimization problem (local minimizer)
\[
\min_{y \in \mathbb{R}^{z^{(k)}}} f(x^{(k)} + Z^{(k)}y),
\]
and set
\[
y^{(k)} = \text{(approximate)} \arg \min_{y \in \mathbb{R}^{z^{(k)}}} f(x^{(k)} + Z^{(k)}y),
\]
\[
x^{(k+1)} = x^{(k)} + Z^{(k)}y^{(k)}
\]
\[k = k + 1, \text{ return to step 1.}
\]

It is worth to point out that the key points of iterated-subspace minimization methods lie in two aspects, i.e., how to determine the iterated-subspace matrix $Z^{(k)}$ and how to solving sub-problem (3.1). Different researchers consider different methods. For example, Conn et al. [15] required that $Z^{(k)}$ contained three components:

- The preconditioned steepest descent direction.
- A number of other conjugate directions which are determined by the preconditioned conjugate-gradient method;
- The overall truncated-Newton direction.

Moreover, they selected the size of subspace according to the number of conjugate directions.

Among the conventional methods, “Step 3” in the above problem is usually generated by solving the following Problem, which is minimize the second-order Taylor expansion of the original problem.

\[
\min \psi_k(d) = f_k + d^T g_k + \frac{1}{2} d^T G^{(k)} d \quad d \in \mathbb{R}^n.
\]

Assume that $Z^{(k)}$ is an $n \times z^{(k)}$ nonsingular matrix such that $Z^{(k)T}Z^{(k)} = I$. Then the subspace constraint can be satisfied by setting $d = Z^{(k)}d_z$. Substituting this in (3.3) gives the problem

\[
\min \psi_k(d_z) = f_k + g_k^T d_z + \frac{1}{2} d_z^T G_z d_z \quad d_z \in \mathbb{R}^{z^{(k)}}
\]

where $g_z = Z^{(k)T}g^{(k)}$, $G_z = Z^{(k)T}G^{(k)}Z^{(k)}$ and $\|Z^{(k)}d_z\|_2 = \|d_z\|_2$.

It is obvious that there are a lot of highly efficient algorithms for such a low-dimension unconstrained minimization problem, such as Newton-like second-derivative methods, secant methods and so on.

Motivated by the basic idea of the conventional “Iterated-Subspace minimization methods”, we adopt the pattern search method to solve the “$z^{(k)}$-dimensional minimization problem (local minimizer)” in Step 3 in an attempt to achieve a more efficient method.

4. Algorithmic framework

We have shown above that the biggest defect of general pattern search methods is that the search directions (which are relative to the dimension of objective function) are too numerous and require too many function evaluations. For example, for a 100-dimensional minimization problem, even we use positive bases to replace bases, we have to search at least 101 directions to obtain a pattern search point. Too many times of function evaluations may bring inefficiency for implementation. In order to remove this difficulty, we consider implementing the pattern search in a subspace with a lower dimension to reduce the number of function evaluations. The method proposed here falls into the class of iterated-subspace minimization methods. Typically, we consider the simplest 2-dimensional subspace. Here, at each iteration, we construct a 2-dimensional subspace matrix $Z^{(k)}$ and solve the 2-dimensional minimization problem

\[
\min_{y \in \mathbb{R}^2} f(x^{(k)} + Z^{(k)}y)
\]

by pattern search method. Moreover, when we meet the gradient of function, we will use finite difference methods. There are two kinds of difference formulae to be chosen, i.e., forward difference formula and more accurate central difference formula.

Forward difference formula:
\[
\frac{\delta_i^{(k)}}{\delta_i^{(k)}} = \frac{f(x^{(k)} + \delta_i^{(k)} e_i) - f(x^{(k)})}{\delta_i^{(k)}} + O(\delta_i^{(k)}).
\]

Central difference formula:
\[
\frac{\delta_i^{(k)}}{\delta_i^{(k)}} = \frac{f(x^{(k)} + \delta_i^{(k)} e_i) - f(x^{(k)} - \delta_i^{(k)} e_i)}{2\delta_i^{(k)}} + O(\delta_i^{(k)})^2.
\]
where \( e_i \) (\( i = 1, \ldots, n \)) denote the unit coordinate vectors; \( \delta_i^{(k)} \) is the step size of definite difference. In order to make \( \tilde{s}_i^{(k)} \) approach to the actual gradient, we set \( \delta_i^{(k)} = 0.9\delta_i^{(k-1)} \), which means that the step size will approach zero as the algorithm processes.

**Algorithm PISM** (Pattern Search Based Iterated-Subspace Minimization Algorithm).

Step 1. Initialize \( k = 1, Z^{(1)} = I \). Let \( x^{(1)} \) be the initial point and calculate \( d^{(1)} = -g^{(1)} \) by finite difference method.

Step 2. Implement inexact line search along \( d^{(k)} \) to look for \( x^{(k+1)} \). \( k = k + 1 \).

Step 3. If the condition for termination is achieved, then stop.

Step 4. Construct the subspace matrix \( Z^{(k)} = [-g^{(k)}, d^{(k-1)}] \), where \( Z^{(k)} \in \mathbb{R}^{n \times 2} \) and \( g^{(k)} = g(x^{(k)}) \) is calculated by finite difference method. If \( Z^{(k)} \) has full column rank, implement Schmidt orthonormalization such that \( Z^{(k)T}Z^{(k)} = I \); otherwise, let \( Z^{(k)} = Z^{(k-1)} \).

Step 5. Let \( f_i(d_z) = f(x^{(k)} + Z^{(k)}d_z) \). Minimize \( f_i(d_z) \) by pattern search method.

Step 6. Let \( d^{(k)} = Z^{(k)}d_z \).

Step 7. Do inexact line search again along \( d^{(k)} \) to obtain a new iterate \( x^{(k+1)} \).

\[ k = k + 1 \]. Go to Step 3.

**Remark 4.1.** Specifically, Step 5 can be described as follows: Initialize \( m = 1, l = 1, d_z^{(1)} = 0, H = \xi \), where \( \xi \) is a positive constant.

While (Stopping condition does not hold) do

Step 1. Choose grid mesh \( \tilde{h}^{(i)} \), positive basis \( V_+^{(i)} \). Set \( i = 1 \), \( num = 0 \), \( h^{(m)} = \tilde{h}^{(i)} \). Let \( \eta^{(i)} = |V_+^{(i)}| \), where \( |V_+^{(i)}| \) denotes the number of columns in matrix \( V_+^{(i)} \). Let \( \Sigma^{(i)} \) denote the grid generated by current iterate \( d_z^{(m)} \), grid mesh size \( \tilde{h}^{(i)} \) and positive basis \( V_+^{(i)} \).

Step 2. While (\( num < \eta^{(m)} \)) do

(a) Calculate the values of \( f_i(d_z) \) on a finite number of points on grid \( \Sigma^{(i)} \) including \( \{d_z^{(m)} + h^{(m)} \nu_+^{(i)} \} \), where \( \nu_+^{(i)} \) denotes the \( i \)th column vector in \( V_+^{(i)} \).

If there is any point \( y \) which satisfies

\[ f_i(y) < f_i(d_z^{(m)}) \],

let it be \( d_z^{(m+1)} \). Set \( h^{(m+1)} = \phi h^{(m)} \), where \( \phi \) is a positive integer. If \( h^{(m+1)} > H \),

\[ m = m + 1 \],

\[ num = 0 \].

else,

\[ num = num + 1 \].

(b) Set \( i = i + 1 \).

If \( i > \eta^{(m)} \), set \( i = 1 \).

end

Step 3. \( d_z^{(i)} = d_z^{(m)} \)

\[ l = l + 1 \].

End

This subroutine consists of two loops, i.e., inner loop and outer loop. The inner loop (Step 2) searches the new iterate along all the members of positive basis \( V_+^{(i)} \) in turn. In Step 2(a) the finite number of points on grid could be only \( d_z^{(m)} + \tilde{h}^{(m)} \nu_+^{(i)} \) or include some other points which are randomly chosen on the grid. \( num \) is the number recording the times of failed search. When \( num = \eta^{(m)} \), it means that no improved point can be found around the current iterate. Hence, a pattern search point, just the current iterate, is obtained. Then the inner loop ends and outer loop begins. Compared with inner loop, the outer loop is used to choose grid mesh size and positive bases. In a word, these two loops work together to look for pattern search points and positive bases.

Besides, we use simple decrease in Step 2(a). Alternatively, it can be replaced by sufficient decrease [16] on the basis of some modifications to the subroutine according to the frame-based methods [17].

**Remark 4.2.** Positive basis is a set of vectors whose positive combinations can span \( \mathbb{R}^n \), and the number of its cardinality is between \( n + 1 \) and \( 2n \). For example, if \( V \) is a basis of \( \mathbb{R}^n \), then

\[ V_+ = [V, -Ve] \quad (4.1) \]

and

\[ V_+ = [V, -V] \quad (4.2) \]

are both positive bases of \( \mathbb{R}^n \), where \( e = [1, 1, \ldots, 1]^T \).
To reduce the number of search directions and the number of function evaluations, we choose the first generation of positive bases in this paper. Moreover, because 2-dimensional subspace is considered in this paper, there are only three directions in every pattern search.

**Remark 4.3.** Conjugate-gradient method belongs to iterated-subspace minimization method in a sense. At kth iteration, the search direction \( d^{(k)} \) is the linear combination of the minus gradient direction of current iterate \(-g^{(k)}\) and the search direction of last iteration \( d^{(k-1)}\). Consequently, \( d^{(k)} \) can be viewed as being found in the subspace spanned by \(-g^{(k)}\) and \( d^{(k-1)}\). In this paper, we construct the 2-dimensional subspace according to this method. It is worth to point out that because we can only use function values, the minus gradient direction will be calculated by finite difference method. Furthermore, similar to this method, we can use the quasi-Newton direction to replace the minus gradient direction to establish the subspace. At every iteration, we construct the subspace by the quasi-Newton direction of current iterate \( x^{(k)} \) and the search direction of previous direction \( d^{(k-1)}\). Moreover, this quasi-Newton direction can be obtained by finite difference and conjugate factorization of approximating Hessian matrices [18].

**Remark 4.4.** It should be pointed out there is a probability that \( Z^{(k)} = Z^{(k-1)} \) will not be executed every time. That is to say the method cannot get trapped in a 2-dimensional subspace forever. Such a case will occur, if the following three conditions are satisfied: (a) the sequence of iterates is converging (possibly in subsequence) to a non-stationary point; (b) if the error in the gradient estimates goes to zero as \( k \) goes to infinity; (c) \( \{B^{(k)}\} \) is a sequence of positive definite matrices that are bounded and bounded away from singularity. Such three conditions will make the quasi-Newton direction \( -B^{(k)}^{-1}g^{(k)} \) not lie in the subspace spanned by \( Z^{(k)} \) for large \( k \).

**Remark 4.5.** Because Algorithm PISM is a derivative-free algorithm, we use Himmelblau's stopping criterion:

If \( \|x^{(k)}\| > \varepsilon_1 \) and \( f(x^{(k)}) > \varepsilon_1 \),

\[
\frac{\|x^{(k)} - x^{(k-1)}\|}{\|x^{(k-1)}\|} \leq \varepsilon_1, \quad \frac{|f(x^{(k)}) - f(x^{(k-1)})|}{|f(x^{(k-1)})|} \leq \varepsilon_1
\]

are used; Otherwise,

\[
\|x^{(k)} - x^{(k-1)}\| \leq \varepsilon_1, \quad |f(x^{(k)}) - f(x^{(k-1)})| \leq \varepsilon_1
\]

are used, where \( \varepsilon_1 = 10^{-5} \).

For the subroutine of pattern search, there are many choices for stopping criterion. In this paper, we choose \( \hat{h}^{(i)} < \varepsilon_2 \) as the stopping criterion, where \( \varepsilon_2 = 10^{-5} \). Hence, when the mesh size \( \hat{h}^{(i)} < \varepsilon_2 \), the subroutine will terminate.

5. Convergence analysis

Algorithm PISM is a combination of iterated-subspace minimization method and pattern search method. Hence, its convergence is closely related to the convergence of the two methods.

Torczon [2] exploits the characteristics of pattern search methods to establish a global convergence theory. Coope and Price [3] propose a class of grid-based methods which are similar to Torczon’s generalized pattern search method [2], but permit greater freedom in the orientation and scaling of successive grids. And they also prove the convergence of their methods in another way. Furthermore, Price and Coope [19] combine grid-based methods [3] and frame-based methods [17]. In this paper, the convergence of the subroutine of pattern search in Algorithm PISM could be guaranteed according to Theorem 4.1 in [3].

It should be pointed out that the convergence of Algorithm PISM could not be guaranteed in general case. However, in some cases, if the function values can decrease in each search direction of the subspace, Algorithm PISM is convergent. The convergence proof is presented in the following two steps. Firstly, it is shown that the direction \( d \) obtained from Step 6 in Algorithm PISM is a descent direction. Then, the convergence of Algorithm PISM can be obtained based on the convergence of general iterated-subspace minimization methods.

**Lemma 5.1.** If \( f_{\xi}(d_{\xi}) \) can decrease in each search directions of subspace, the direction \( d \) obtained from Step 6 satisfies \( \nabla f(x^{(k)})d < 0 \).

**Proof.** Because \( f_{\xi}(d_{\xi}) = f(x^{(k)} + Z^{(k)}d_{\xi}) \), there is

\[
\nabla f_{\xi}(d_{\xi}) = Z^{(k)T}\nabla f(x^{(k)} + Z^{(k)}d_{\xi}). \tag{5.1}
\]

Let \( d_{\xi} = 0 \), then we have

\[
\nabla f_{\xi}(0) = Z^{(k)T}\nabla f(x^{(k)}). \tag{5.2}
\]
Furthermore, there is
\[
\nabla f_z(0)^T d_z = \nabla f(x^{(k)})^T Z^{(k)} d_z.
\]
(5.3)

It follows from \( d = Z^{(k)} d_z \) that
\[
\nabla f_z(0)^T d_z = \nabla f(x^{(k)})^T d.
\]
(5.4)

Because \( f_z(d_z) \) can decrease in every search direction, which are columns of positive bases. Therefore,
\[
\nabla f_z(0)^T d_z < 0.
\]
Hence we can get
\[
\nabla f(x^{(k)})^T d < 0.
\]

The proof is completed. \( \square \)

**Theorem 5.1.** Algorithm PISM is convergent.

**Proof.** From Lemma 5.1 we have \( \nabla f(x^{(k)})^T d < 0 \) which means that \( d \) is a descent direction.

Similar to the method in [15], the convergence of Algorithm PISM can be guaranteed under fairly general assumptions. The proof is completed. \( \square \)

6. Numerical results

To show the efficiency of our method, some examples of classic optimization problems are adopted in this paper. These examples are taken from the set of test examples frequently used in the literature. All tests are implemented on a PC with 2.66 GHz Pentium 4 and 256 MB SDRAM using Matlab 6.5.

In our algorithm, we let \( \epsilon_1 = \epsilon_2 = 10^{-5} \). In the subroutine of pattern search method we set \( n^{(m)} = 400, \xi = 10 \) and fix the positive basis as \([I, -le] \), where \( I \) is unit matrix and \( e = [1, 1, \ldots, 1]^T \).

All the test functions are outlined as follows:

| TF. 1: Beale [20] |
| TF. 2: Hilbert [21] |
| TF. 3: Dixon [22] |
| TF. 4: Power [20] |
| TF. 5: Tridia [23] |
| TF. 6: Variably Dimensioned [20] |
| TF. 7: Modified Cragg [24] |
| TF. 8: Brown and Dennis [25] |
| TF. 9: Trigonometric [20] |
| TF. 10: Box 3-dimensional [26] |
| TF. 11: Biggs (n = 4) [20] |
| TF. 12: Biggs (n = 6) [20] |

For each test function, Table 6.1 reports the numerical results of Algorithm PISM, including the dimension of the objective function argument \((N)\), the number of iterations \((N_t)\), the total number of function evaluations \((N_f)\), the value of the objective function at the termination point \(x_t(f_t)\). For comparison purpose, the relative data about general pattern search algorithm(GPS) which is the subroutine of Algorithm PISM are also given in Table 6.1.

It can be seen from Table 6.1 that Algorithm PISM is relatively efficient while compared with the general pattern search method. In all tests except for TF.1 and TF.7, the total number of function evaluations is reduced when using PISM. The higher the function dimension is, the more the number of function evaluations is reduced. For example, in TF.5, the differences of the total number of function evaluations \( N_f \) between PISM and GPS are respectively 256 (495—239) and 19733 (38209—18476) for dimension 10 and 1000. However, it should be pointed out that for a lower dimension problem the efficiency of the proposed method is not obvious. In an extreme case, i.e., a 2-dimensional problem (say example TF. 1), the total number of function evaluations is almost equal between PISM and GPS. This is because, for 2-dimensional problems, the PISM is just the GPS. Besides, it should be pointed out that GPS can have an artificial advantage in some cases. For example, TF. 3, TF. 4, TF. 5 and TF. 7 have their initial points and solutions differing by integer values in each dimension. Hence if GPS uses an integer grid spacing then the solution point is also a grid point and GPS can step exactly to the solution very quickly. This puts GPS at a massive advantage. However, in a real life problem this advantage would not exist. Furthermore, it can be seen from the numerical results that GPS does not find the solution on all problems, such as TF. 6. The numerical results also indicate that total number of iterations \( N_t \) will increase if the PISM is used. That is to say, the integration of the iterated-subspace minimization method into the pattern search method will increase the number of iterations although the total number of function evaluations is reduced.
Table 6.1
Numerical results of Algorithm PISM.

<table>
<thead>
<tr>
<th>Function</th>
<th>Dimension, N</th>
<th>Algorithm</th>
<th>( N_t )</th>
<th>( N_f )</th>
<th>( f_t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF. 1</td>
<td>2</td>
<td>PISM</td>
<td>6</td>
<td>757</td>
<td>0.116 \times 10^{-7}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPS</td>
<td>229</td>
<td>724</td>
<td>0.167 \times 10^{-7}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NGOCSRSR1</td>
<td>14</td>
<td>81</td>
<td>0.339 \times 10^{-10}</td>
</tr>
<tr>
<td>TF. 2</td>
<td>4</td>
<td>PISM</td>
<td>5</td>
<td>45</td>
<td>0.320 \times 10^{-8}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPS</td>
<td>391</td>
<td>3432</td>
<td>0.160 \times 10^{-6}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NGOCSRSR1</td>
<td>4</td>
<td>47</td>
<td>0.598 \times 10^{-11}</td>
</tr>
<tr>
<td>TF. 3</td>
<td>10</td>
<td>PISM</td>
<td>28</td>
<td>782</td>
<td>0.273 \times 10^{-19}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPS</td>
<td>393</td>
<td>5972</td>
<td>0.954 \times 10^{-13}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NGOCSRSR1</td>
<td>45</td>
<td>971</td>
<td>0.353 \times 10^{-10}</td>
</tr>
<tr>
<td>TF. 4</td>
<td>20</td>
<td>PISM</td>
<td>12</td>
<td>294</td>
<td>0.212 \times 10^{-9}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPS</td>
<td>2</td>
<td>420</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NGOCSRSR1</td>
<td>35</td>
<td>1480</td>
<td>0.699 \times 10^{-10}</td>
</tr>
<tr>
<td>TF. 5</td>
<td>10</td>
<td>PISM</td>
<td>11</td>
<td>239</td>
<td>0.622 \times 10^{-8}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPS</td>
<td>12</td>
<td>495</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NGOCSRSR1</td>
<td>11</td>
<td>255</td>
<td>0.188 \times 10^{-11}</td>
</tr>
<tr>
<td>TF. 6</td>
<td>20</td>
<td>PISM</td>
<td>118</td>
<td>1474</td>
<td>0.284 \times 10^{-9}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPS</td>
<td>20</td>
<td>2109</td>
<td>0.111 \times 10^{-8}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NGOCSRSR1</td>
<td>49</td>
<td>5053</td>
<td>0.647 \times 10^{-18}</td>
</tr>
<tr>
<td>TF. 7</td>
<td>4</td>
<td>PISM</td>
<td>70</td>
<td>645</td>
<td>0.802 \times 10^{-8}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPS</td>
<td>2</td>
<td>1920</td>
<td>0.633 \times 10^{-4}</td>
</tr>
<tr>
<td>TF. 8</td>
<td>4</td>
<td>PISM</td>
<td>10</td>
<td>156</td>
<td>0.068 \times 10^{-10}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPS</td>
<td>2</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NGOCSRSR1</td>
<td>49</td>
<td>455</td>
<td>0.726 \times 10^{-10}</td>
</tr>
<tr>
<td>TF. 9</td>
<td>5</td>
<td>PISM</td>
<td>26</td>
<td>195</td>
<td>0.858222 \times 10^{5}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPS</td>
<td>396</td>
<td>3367</td>
<td>0.87547 \times 10^{5}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NGOCSRSR1</td>
<td>18</td>
<td>209</td>
<td>0.858222 \times 10^{5}</td>
</tr>
<tr>
<td>TF. 10</td>
<td>3</td>
<td>PISM</td>
<td>15</td>
<td>490</td>
<td>0.294 \times 10^{-8}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPS</td>
<td>272</td>
<td>1852</td>
<td>0.756 \times 10^{-7}</td>
</tr>
<tr>
<td>TF. 11</td>
<td>4</td>
<td>PISM</td>
<td>20</td>
<td>692</td>
<td>0.220 \times 10^{-11}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPS</td>
<td>76</td>
<td>709</td>
<td>0.435 \times 10^{-11}</td>
</tr>
<tr>
<td>TF. 12</td>
<td>6</td>
<td>PISM</td>
<td>5</td>
<td>91</td>
<td>0.289 \times 10^{-8}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPS</td>
<td>390</td>
<td>4446</td>
<td>0.690 \times 10^{-8}</td>
</tr>
</tbody>
</table>
Furthermore, we compare Algorithm PISM with some other derivative-free algorithms, such as Algorithm NGOCSSR1 [27]. Table 6.1 shows that our algorithm performs rather well in TFs. 3, 6, 7, 8 and 9. For other test problems, some of the function values in PISM are larger than those of NGOCSSR1. Therefore, our algorithm may not be better than NGOCSSR1 method. Similar situation also occurred for the comparison of GPS method in Table 6.1. Such a finding motivates us to further investigate the applicable condition of PISM algorithm in future study.

7. Conclusions

This paper has presented a new pattern search method for unconstrained optimization problems. Conventional pattern search methods may be very inefficient when solving some high-dimensional problems. Specifically, for an n-dimensional problem there needs at least \( n + 1 \) times of function evaluations to find a pattern search point. To reduce the number of function evaluations, the iterated-subspace minimization method is integrated into the framework of the conventional pattern search method. At each iteration, a low-dimensional manifold, the iterated subspace, is constructed. In such a subspace, the pattern search method can be easily implemented to find an approximate minimizer of the objective function. Meanwhile, a method for the construction of a 2-dimensional subspace is also proposed in this paper. Hence, there are only three search directions in every pattern search under the basis of positive bases. As such, the number of function evaluations can be considerably reduced. Although the convergence of the new method cannot be guaranteed in general case, we have proven that for a class of functions, whose values decrease in each search direction in subspace, the new method is convergent. Numerical results indicate that our method is efficient and competitive when compared with the conventional pattern search method and the derivative-free method.

It should be pointed out that a 2-dimensional subspace might not be the most appropriate one in the proposed method. Subspaces with a higher dimension may also be adopted in our method, but a higher dimension may result in more number of function evaluations. How to choose a subspace with an appropriate dimension so as to improve the efficiency of the pattern search methods still reveals interesting and significant investigations in our further studies.

Moreover, this paper can also be regarded as an extended application of pattern search methods for unconstrained optimization problems. We have seen some merits of this method in the numerical results. The pattern search method has also been developed in the literature (see [5, 14]). The comparison of these methods could also deserve further investigation.

References