Scholars Journal of Engineering and Technology (SJET) **ISSN 2321-435X (Online)**
Sch. J. Eng. Tech., 2014; 2(4B):591-596 **ISSN 2347-9523 (Print)**

(An International Publisher for Academic and Scientific Resources)
www.saspublisher.com

Research Article

An Efficient Modified Derivative-Free Method on Bound Constrained Optimization

Xiaoli Zhang, Qinghua Zhou*
College of Mathematics and Computer Sciences, Hebei University, Baoding, 071002, Hebei Province, China

***Corresponding author**
Qinghua Zhou
Email: qinghua.zhou@gmail.com

Abstract: This paper introduces an efficient modified derivative-free method for bound constrained optimization problems. It is based on the coordinate search method. During the running of the algorithm, it incorporates the progressive obtained local information into the current iteration. Actually, after we find two different suitable descent directions, we introduce the composite expansion step. By doing these, a new point is produced through some kind of line search techniques. Then we test the efficiency of our new method on the benchmarking. The computational results shows the efficiency of the modified algorithm.
Keywords: bound constrained optimization; derivative-free method; line search

INTRODUCTION

In this work, we consider the bound constrained minimization problem

$$Min\ f(x) \qquad (1)$$

$$s.t.\ l \le x \le u \qquad (2)$$

Where $x, l, u \in R^n$, with $l < u$, and $f : R^n \to R$, a continuously differentiable function. We suppose that the first order derivatives can not be explicitly calculated or approximated. We set both $l^i = -\infty$ and $u^i = \infty$ for some $i \in \{1, 2, \cdots, n\}$ to denote the relevant variables are unbounded. $F = \{x \in R^n : l \le x \le u\}$ means the feasible set. $\|\cdot\|$ and $\langle \cdot, \cdot \rangle$ signify the Euclidean norm and inner product, respectively. The identity matrix is denoted by I and its i-th column by e_i. The $j-th$ component of a vector $v \in R^n$ is indicated by v_j. Finally, ∂F denotes the boundary of the feasible set F.

Derivative-free optimization methods (also named direct search methods) which neither compute nor approximate derivatives are attracted more and more attentions in recent years. In order to overcome the lack of gradient information, many globally convergent derivative-free algorithms are proposed in literature. And most of them are based on the idea of performing finer and finer samplings of the objective function along suitable sets of search directions [1-3].

In 2002, Lucidi [4] proposed an algorithm to solve the above bound constrained optimization problem. Whenever a "suitable" descent feasible coordinate direction is detected, a new point is produced by performing a derivative free line search along this direction (expansion step). Li and Zhou [5] improved the algorithm in [4] by the following ways. When the algorithm detects the second decent direction, then it does not use line search any more, but to introduce a composite search directions and execute the expansion step. The results show the efficiency of the simple strategy. Inspired by them, we introduce a different composite search technique. We think the strategies in [4] and [5] neglect the local information to some extent circling the current iteration point. Considering the intuitive of simplex methods, we construct a composite direction whenever the algorithm find two different descent directions and then execute the expansion step.

The main features of our algorithm are the following: In order to reduce the complexity of the algorithm, we do not perform the "sufficiently" large step length along every direction after we find a "suitable" point. We calculate the new point by performing a line search along composite direction of the obtained two descent directions, but not the detected one. According to this simple skill, we make considerable progress in reducing the numbers of function evaluations.

The paper is organized as follows. In Section 2, we describe our algorithm for solving the bound constrained problem by combining the local information which is progressively obtained during the iterations of the algorithm. In Section 3, the primary numerical results are presented. Finally a short discussion about conclusion and future work is given in section 4.

THE MODIFIED ALGORITHM

At this part, we present a new derivative-free algorithm for the minimization of a continuously differentiable function, in the case of that some of (or all) the variables are bounded. As we all know, the successfully coordinate directions must be the set of descent directions, which we can search and remain feasible for a sufficiently long distance in the box constraints. On the basis of the previous constrained methods, we propose a different composite search technique. Moreover, It samples the objective function along the coordinate directions, with the aim of detecting a feasible direction. Furthermore, this set of search directions and the used particular sampling technique allow us to overcome the lack of gradient information and ensure that every limit point of the sequence produced is a stationary point for problem [1]. Then, we have all we need to state our algorithm now.

Algorithm 1 The modified algorithm

Start with $x_0 \in F,\ \theta \in (0,1),\ \gamma > 0,\ 0 < \alpha_0^i < \infty,\ d_i = e_i,\ i = 1,2,\cdots,n$. Let $k = 0, i = 1$ be given.

(1) Compute $\alpha_{i\max}$, s.t. $x_k + \alpha_{i\max} d_i \in \partial F$ and set $\alpha_i = \min\left\{\alpha_k^i, \alpha_{i\max}\right\}$. If $\alpha_i > 0$ and $f\left(x_k + \alpha_i d_i\right) \le f\left(x_k\right) - \gamma\alpha_i^2$, then go to step (4).

(2) Compute $\alpha_{i\max}$, s.t. $x_k - \alpha_{i\max} d_i \in \partial F$ and set $\alpha_i = \min\left\{\alpha_k^i, \alpha_{i\max}\right\}$. If $\alpha_i > 0$ and $f\left(x_k - \alpha_i d_i\right) \le f\left(x_k\right) - \gamma\alpha_i^2$, then go to step (4); else go to step (3).

(3) Set $\alpha_k = 0, \alpha_{k+1}^i = \theta\alpha_i$, $i = \mathrm{mod}\left(i, n\right) + 1,$ go to step (1).

(4) Use the similar program as (1)、（2）to find another descent direction d_j.

(5) Compute α_{com} by composite expansion step $\left\{d_i, d_j, \alpha_i, \alpha_j, \gamma\right\}$.Set $x_{k+1} = x_k + \alpha_{com}\left(d_i + d_j\right)$.

(6) $j = \mathrm{mod}\left(j, n\right) + 1, i = j,$ go to step (1).

Composite expansion step $\left\{d_i, d_j, \alpha_i, \alpha_j, \gamma\right\}$

Data. $\delta, \delta_1, \delta_2 \in \left(0,1\right)$.

Step1. Let

$$\bar{\alpha}_i = \min\left\{\alpha_{i\max}, \frac{\alpha_i}{\delta_1}\right\}$$

$$\bar{\alpha}_j = \min\left\{\alpha_{j\max}, \frac{\alpha_j}{\delta_2}\right\}$$

If $\alpha_i = \alpha_{i\max}$, then compute α_k by the expansion step $\left(d_j, \alpha_j, \alpha_{j\max}, \gamma\right)$;

If $\alpha_j = \alpha_{\text{jmax}}$, then compute α_k by the expansion step $\left(d_i, \alpha_i, \alpha_{\text{imax}}, \gamma\right)$;

If $f\left(x_k + \hat{\alpha}_i d_i + \hat{\alpha}_j d_j\right) > f\left(x_k\right) - \gamma\left(\hat{\alpha}_i^2 + \hat{\alpha}_j^2\right)$, update the best point and stop.

Step2. If $\alpha_i = \hat{\alpha}_i, \alpha_j = \hat{\alpha}_j$, go to step1.

Expansion step $\left\{d_i, \alpha_i, \alpha_{i\max}, \gamma\right\}$

Data. $\delta \in (0,1)$

Step1. let $\hat{\alpha} = \min\left\{\alpha_{i\max}, \frac{\alpha_i}{\delta}\right\}$. If $\alpha_i = \alpha_{i\max}$ or

$f\left(x_k + \hat{\alpha} d_i\right) > f\left(x_k\right) - \gamma\alpha^2$, set $\alpha_k = \alpha_i$ and stop.

Step2. set $\alpha_i = \hat{\alpha}$, go to step1.

Particularly, the steps of the algorithm can be detailed explanation as follows:

In the first step, we compute the maximum feasible step-length α_{imax}, the direction d_i is checked on the aim of determining a feasible point where the objective function is sufficiently decreased. Next, the trial stepsize α_i is determined by choosing the minimum between $\alpha_{i\max}$ and α_k^i. Certainly, the scalar α_k^i has been computed on the basis of the behavior of the objective function along the same direction showed in the previous iterations. Therefore, the scalar α_k^i should provide a promising initial stepsize for the direction d_i. Finally, it is verified if the moving of length α_i along d_i produces a feasible point where the function is sufficiently reduced. If such a point is produced then we could directly go to find another descent point immediately. Otherwise, the direction $-d_i$ is considered (Step 2).

The second step is similar to the first one by replacing d_i with its opposite direction $-d_i$. In this case, we still need to calculate the maximum feasible step-length α_{imax} and decide the trial stepsize α_i along with the direction $-d_i$. If the trial point $x_k - \alpha_i d_i$ can produce a sufficient decrease of f then we go to the fourth step directly. Otherwise, we perform the next step.

In the third step, let the stepsize α_k is equal to zero and reduce the scalar α_k^i, which means the line search is failed to find a possible descent point in the current direction. Then, the algorithm goes back to the first step to check other directions.

The purpose of the fourth step is to find another descent direction, where the main process is quite similar to steps (1) and (2).

In the fifth step, we use a composite search technique. Details are as follows, a comparative larger stepsize α_{com} is computed by the composite expansion step, the aim is to accelerate the algorithm and make suitable progress in reducing the numbers of function evaluations.

NUMERICAL RESULTS

In this section, we implement our new algorithm and compare it with the method of Lucidi [4] and Li [5]. The test problems are those given by [6], which are obtained from the set of functions suggested in [7]. During all the tests, the parameters appeared in the algorithm model have been set as follows:

$$\gamma = 10^{-6},\ \delta = 0.25,\ \delta_1 = \delta_2 = 0.5, \theta = 0.5, \alpha_0^i = 0.5,\ i = 1, \cdots, n$$

Note that we have not performed an extensive empirical tuning of the parameters in the algorithm. We have adapted choices usually adopted in linesearch techniques of gradient based algorithms.

About the stopping criterion, we have adapted the same approach proposed in [6]. Let f_0 be the value of f at the starting point x_0 and f^* be the best known function value. Then we introduce the quotient

$$q_k = \frac{f_k - f^*}{f_0 - f^*} \qquad (2)$$

Which can be considered a measure of the convergence speed and we have terminated an algorithm whenever

$$q_k \le \varepsilon \qquad (3)$$

Where ε is a prefixed value. We can got an idea on the efficiency of an algorithm when the values of ε are different. However, in some test problems the global minimum is not the unique stationary point. Therefore, an algorithm could generate a sequence converging towards a stationary point $\bar{x}$, with $f\left(\bar{x}\right) > f^*$, and could never satisfy the criterion (2). To tackle this possible occurrence, Lucidi [4] have also introduced the following stopping criterion

$$\max_{i=1,2,\cdots,n} \left\{ \alpha_k^i \right\} \le 10^{-5}.$$

Finally, an algorithm is terminated when it has performed a number $N_{\max} = 1000$ of function evaluations which is not accordance with any of the two the stopping criterions.

We use the same numbering system as that in [5]. And we have tested our algorithm on the set of problem with these different values of ε, namely $\varepsilon = 10^{-1}, \varepsilon = 10^{-3}, \varepsilon = 10^{-6}$. Next we give the results of compared different value in table 1.

Where p represents the problem, ni represents the number of iterations. nf and f represent the numbers of function calculations and function values, respectively. Additionally, the symbols "×" means that the algorithm terminates because the number of function evaluation exceeds 1000.

Table-1: The computational results with different value

	$\varepsilon = 10^{-1}$		$\varepsilon = 10^{-3}$		$\varepsilon = 10^{-6}$	
P	ni/nf	f	ni/nf	f	ni/nf	f
1	3(16)	226.000000000000000	8(50)	1.744249056521678	27(140)	9.904221353475555E-001
2	10(87)	8.069403313633597E-002	×	3.185557319949477E-002	×	3.185557319949477E-002
3	2(11)	1.872979626928002E-008	7(36)	1.229773350044900E-008	15(75)	1.128199030219998E-008
4	1(33)	3.332684828862538E-002	2(48)	8.992110460296558E-004	×	1.885276454801245E-006
5	0(104)	87.924516358860330	0(144)	9.045313242136058E-001	25(249)	5.501810695295482E-001
6	0(8)	28873.250000000020000	0(9)	531.411600000000600	×	7087.800000000000000
7	0(7)	1.039771219166782	×	4.393713051614986E-002	×	4.393713051614986E-002
8	0(6)	1.842048983570117	×	8.061138392749635E-002	×	8.061138392749635E-002
9	×	126558.065240000000000	×	127270.565230000000000	×	127270.565230000000000
10	0(5)	4.040932327162389E-002	7(55)	2.044972255680543E-003	19(131)	1.048460241963999E-005

11	2(13)	8.205588889092597	3(20)	4.079890300419146E-002	21(246)	4.523609729460724E-004
12	1(29)	783.999999999355500	1(29)	783.999999999355500	1(29)	783.999999999355500
13	×	5155325.263149790000000	×	5155325.263149790000000	×	5155325.263149790000000
14	25(153)	5.121685717389848	25(153)	5.121685717389848	25(153)	5.121685717389848
15	27(255)	4.303011410533279E-003	27(255)	4.303011410533279E-003	27(255)	4.303011410533279E-003
16	8(53)	1.557738104462623	29(140)	2.500000011141310E-001	29(140)	2.500000011141310E-001
17	1(10)	20.937500000000000	11(87)	1.100158691406250E-001	15(125)	1.999666072661058E-004
18	0(8)	8.876953125000000E-001	4(16)	0.000000000000000E+000	4(16)	0.000000000000000E+000
19	0(12)	1548.775000000000000	9(83)	10.361914062500000	52(454)	7.876417263871907
20	18(159)	3.114016303330327E-003	×	1.677046996345801E-003	×	1.677046996345801E-003
21	26(213)	7.013697702361832E-003	×	4.157200717201540E-003	×	4.157200717201540E-003
22	31(243)	2.754576149125480E-003	×	1.947492266966226E-004	×	1.947492266966226E-004
23	19(180)	9.041782069466963E-003	72(679)	6.531053039722382E-003	72(719)	6.503013632708789E-003

From the table 1, We can clearly see that, for $\varepsilon = 10^{-1}$, most of the test problems can be solved by our method. But with the increasing of the precision, the number of function evaluations is also increasing. However, with the degree of required precision becoming higher, our function evaluations are smaller than precious.

Then, we compare our algorithm both with Li [5] and Lucidi [4]. The results are listed in Table 2 and Table 3. Furthermore, for depicting the performance of different algorithms, we say that an algorithm wins only if the number of function calculations required to solve a test problem is smaller than or equal to 95% of the one required by another algorithm. Particularly, the modified new algorithm is defined by "newalg".

Table-2 The statistic results of experiments

	Algorithms	No. of wins in terms of n_f	No. of balances
$\varepsilon = 10^{-1}$	newalg	13	1
$\varepsilon = 10^{-1}$	Lucidi [4]	8	1
$\varepsilon = 10^{-3}$	newalg	10	3
$\varepsilon = 10^{-3}$	Lucidi [4]	9	3
$\varepsilon = 10^{-6}$	newalg	15	2
$\varepsilon = 10^{-6}$	Lucidi [4]	6	2

Table-3: The statistic results of experiments

	Algorithms	No. of wins in terms of n_f	No. of balances
$\varepsilon = 10^{-1}$	newalg	12	5
$\varepsilon = 10^{-1}$	Li [5]	6	5
$\varepsilon = 10^{-3}$	newalg	9	7
$\varepsilon = 10^{-3}$	Li [5]	7	7
$\varepsilon = 10^{-6}$	newalg	8	8
$\varepsilon = 10^{-6}$	Li [5]	7	8

Clearly, Table 2 shows that, on the whole, the result of our algorithm is better than the one in Lucidi [4], especially when there is a high precision. That is to say, higher the degree of required precision, more efficient of our algorithm. Furthermore, For the comparative results with Li [5], our strategy is more competitive. In general, our algorithm is

efficient in reducing the number of function evaluations then the exist ones, which is the most often used indicator in derivative free optimization problems.

CONCLUSIONS AND FUTURE WORKS

In this paper, we investigate the performance of a new derivative-free algorithm for solving the bound constrained optimization problems. The algorithm is based on the coordinate search method. After we find two suitable descent directions, we construct and execute the socalled composite expansion step. By introducing this simple strategy, the number of function calculations is improved significantly for most test problems. In general, we think the new method should be more effective in practice. In the near future, we will still devote ourselves to studying the effect with different choices of the descent directions and construct more efficient search strategies.

Acknowledgements

This work is supported by the National Nature Science Foundation of China (Grant No. 11101115) and the Natural Science Foundation of Hebei Province (Grant No. A2014201003, A2014201100).

REFERENCES

1. Torczon V; On the convergence of pattern search algorithms. SIAM J. Optim, 1997; 7:1-25.
2. Lewis RM, Torczon V; Pattern search methods for bound constrained minimization. SIAM Journal on Optimization, 1999; 9: 1082–1099.
3. Hooke R, Jeeves TA; Direct search" solution of numerical and statistical problems. J.ACM, 1961; 8: 212-229.
4. Lucidi S, Sciandrone M; A Derivative-Free Algorithm for Bound Constrained Optimization. Computational Optimization and Applications , 2002; 21: 119-142 .
5. Yan Li, Qinghua Zhou; A modified derivative-free algorithm for bound constrained optimization. Machine Learning and Cybernetics, 2006 International Conference on. pp. 2242 – 2245.
6. Elster C, Neumaier A; A grid algorithm for bound constrained optimization of noisy functions. IMA Journal of Numerical Analysis, 1995; 15:585–608.
7. Mor´e JJ, Garbow BS, Hillstrom KE; Testing unconstrained optimization software, ACM Trans. On Math. Software, 1981; 7:17–41.