A computational experiment based on a nonlocal search algorithm using potential theory

S N Medvedev and O A Medvedeva

Department of Applied Mathematics, Informatics and Mechanics, Voronezh State University, 1, University Square, Voronezh, 394018, Russia

E-mail: s_n_medvedev@mail.ru

Abstract. We consider a general method to derive an optimization algorithms based on potential theory. This theoretical approach is based on the randomization of an objective function and the computation of directional derivatives of the randomized functional. Using the gradient of the potential field we consider algorithm in terms of the means of the underlying movements in the space of random vectors. Effectively, this process combines the exploration power of random-search algorithms with the exploitation power of direct-search algorithms. At the end we offer the results of a computational expert.

1. Introduction

We consider an approach to construct optimization algorithms for non-smooth functions that are given by its values (no analytical formula) and their derivatives are not known [1-4]. In this approach the original optimization problem is replaced with randomized problem, allowing the use of Monte-Carlo methods for calculating integrals. It will be shown, that the value of the gradient can be obtained using values of target function only, in the framework of Monte Carlo methods for calculating integrals. Furthermore, this value should not need to be precise, because it is recalculated at each iteration step.

2. Problem setup

Let’s consider the unconstrained optimization problem

\[ f(x_1, ..., x_n) \rightarrow \min_{x \in \mathbb{R}^n}. \]  

(1)

By randomizing we get

\[ F(X) = E[f(X)] \rightarrow \min_{X \in \{X \in \mathbb{R}^n\}} X, \]  

(2)

where \( X \) – set of random vectors with values from \( \{R^n\} \).
The solution of (2) is a random vector from $\{X\}$ that optimizes a functional $F(X)$.

Let note, that

a) (1) and (2) are equivalent in some sense;

b) (2) is the **stochastic optimization** problem of the functional $F(X)$.

The suggested scheme makes it possible to obtain optimization methods on the regular basis, similar to methodology from smooth optimization. It means that the following steps are present [1-6]:

1) randomization of the original optimization problem;
2) finding directional derivative for the randomized problem;
3) as a condition for selecting the direction $Y$ for the next iteration step we take the condition that directional derivative in the direction of $Y$ being less or equal to 0.

Because of randomization, the expression for directional derivative doesn’t contain the differential characteristics of the original function. We obtain the condition for selecting the direction of search $Y$ in terms of its characteristics – conditional expectation. Conditional expectation is a vector function (or vector field) and can be decomposed (following the theorem of decomposition of the vector field) into the sum of the gradient of scalar function $P$ and a function with $\text{div} = 0$ [2, 4]. $P$ is called a potential function.

As a result the original problem is reduced to optimization of the potential function.

Let’s note that the potential function is specific for each iteration step. Next, the equation in partial derivatives is obtained that connects $P$ and the original function.

For particular computational algorithms it is necessary to specify the measure where the dynamics of the random vectors is taking place. For example, the dynamics can be expressed in a form of densities. For the particular class of distributions, for example normal distribution, the dynamics can be written in terms of expectation and covariance matrix. It is also possible to express the dynamics in mixed characteristics.

3. **Procedure for searching in a set of random vectors. Dynamics of the expectation**

We define the motion in space of random vectors by the same linear structure

$$X^{N+1} = X^N + \alpha_{N+1} Y^{N+1}, \quad N = 0,1,...$$

(3)

here $N$ – number of iteration, $Y^{N+1}$ – random vector that define direction at the $N$-th iteration, $\alpha_{N+1}$ – step at the $N$-th iteration.

Generally, only one condition is put on the iterative process:

$$F(X^{N+1}) < F(X^N), \quad N = 0,1,...$$

(4)

We take the expectation of the (3)

$$E[X^{N+1}] = E[X^N] + \alpha_{N+1} E[Y^{N+1}]$$

and represent $E[Y]$ in the same form [2, 5, 6], because $Y$ is statistically dependent on $X$

$$E[Y] = E_x E[Y|X].$$

(5)

We get

$$E[X^{N+1}] = E[X^N] + \alpha_{N+1} E_x E[Y^{N+1}|X^N].$$

Denote [2]
\[ \overline{y}(x) = \int_{\mathbb{R}^n} y \cdot p_{Y|X=x} (y) \, dy = E[Y | X = x]. \]

Let vector \( \tilde{E}^{N+1} \) is the estimate of mathematical expectation \( E[\{X^{N+1}\}] \), then
\[ \tilde{E}^{N+1} = E^N + \alpha_{N+1} \tilde{E}_X \left[ \overline{y}(X^N) \right], \tag{6} \]
where \( E^N = E[X_N] \).

It is important to introduce the decomposition of the vector field \( p_x(x) \cdot \overline{y}(x) \) into potential field \( \nabla \varphi \) and a divergence-free field \( W \)
\[ p_x(x) \cdot \overline{y}(x) = \nabla \varphi (x) + W(x). \]

From [2] we obtain
\[ \nabla \varphi_0 (x) = -\lambda E \left[ \nabla,FSLE(x,u)(f(u) - C) \right], \]
where \( FSLE(x,\xi) = \frac{x - \xi}{\omega_n |x - \xi|^n} \) — fundamental solution of Laplace’s equation.

Thus, we have
\[ \tilde{E}^{N+1} = E^N + \alpha_{N+1} \tilde{E}_X \left[ \frac{f(u) - C(X^N - u)}{\omega_n |X^N - u|^n p_N(X^N)} \right], \tag{7} \]
where \( \omega_n = \frac{\pi^{n/2}}{\Gamma(n/2)} \) — the area of unit sphere in \( \mathbb{R}^n \).

Let \( \hat{f}(x) = f(x) - C \).

We take the realization of random vector \( u \) and select the expectation \( E^N \) as realization of random vector \( X^N \) and as a result obtain
\[ \tilde{E}^{N+1} = E^N + \alpha_{N+1} \frac{\hat{f}(u)(E^N - u)}{\omega_n |E^N - u|^n p_N(E^N)} \tag{8} \]

If the realization of \( u \) is a “perspective” \( \hat{f}(u) < 0 \) then the movement is carried in the direction \( (E^N - u) \), in other case \( \hat{f}(u) \geq 0 \) — in the direction that opposite \( (E^N - u) \). \( f(E^N) \) can be selected as \( C^N \). If \( \hat{f}(\tilde{E}^{N+1}) < 0 \), then assume \( \tilde{E}^{N+1} = E^N \). On other hand, when \( \hat{f}(\tilde{E}^{N+1}) \geq 0 \), then necessary to reduce step \( \alpha_{N+1} \). If we have not received satisfactory \( \tilde{E}^{N+1} \) after several attempts, then it is better to take the new realization of vector \( u \).

So, next algorithm can be offered.
4. Algorithm

Consider the computational scheme of the algorithm.

1. Generate $K$ realization random vector $u^N: u^{N,i} \ (i = 1, K)$. 
2. Calculate objective function in these realizations $f(u^{N,i})$. 
3. Renumber points in increasing order of objective function values. 
4. Calculate average level $C^N = \frac{1}{K} \sum_{i=1}^{K} f(u^{N,i})$. 
5. Note $L$ as number of «perspective» realizations: 
   \[ \tilde{f} (u^{N,i}) < 0 \quad (i = 1, L) \; ; \; \tilde{f} (u^{N,j}) \geq 0 \quad (j = L+1, K). \] 
6. Calculate $E^N_{\alpha} = \frac{1}{L} \sum_{i=1}^{L} u^{N,i}$ by «perspective» realizations $u^{N,i}$. 
7. Apply (27) to each point $u^{N,i}$ 
   \[ \hat{E}^{N+1} = E^N + \alpha \frac{\tilde{f}(u)(E^N - u)}{\omega_{\text{nu}} (E^N - u)^{\text{nu}} p_n(E^N)}. \]
   If $\tilde{f}(\hat{E}^{N+1}) \geq 0$, then reduce step $\alpha$. 
8. Check stop-criterion. If failure, then assume $u^{N+1,i} = E^{N+1,i}$ and go to step 3.

We obtain derivative free algorithm.

it is expedient to use standard deviation of the function $f$ from average level $C$ as stop-criterion 

\[ \tilde{D}^N = \frac{1}{K} \sum_{i=1}^{K} (f(u^{N,i}) - C^N)^2. \]  

Algorithm stops if $\tilde{D}^N < \varepsilon$, where $\varepsilon$ – small value.

We can define «coefficients of perspective» $\gamma_i$ to enhance the local properties of the algorithm 

\[ \gamma_i = \frac{\tilde{f}(u^{N,i})}{\sum_{i=1}^{K} \tilde{f}(u^{N,i})}. \]  

and then 6-th step can be rewritten in the following way 

\[ E^N = \sum_{i=1}^{L} \gamma_i f(u^{N,i}). \]  

We can assume $C = f(u^{N,K-1})$ to enhance the nonlocal properties of the algorithm.
5. Computational experiment

The proposed algorithm was tested on various test data. The most successful results are shown in the following tables.

The first one (Table 1) shows the results for a function of one variable.

| Functions                             | Algorithm          | Exact solution |
|---------------------------------------|--------------------|----------------|
| $f(x) = x^2 + 1$                      | $x_{\text{min}} = 0.00000143$ | $x^* = 0$, $f(x^*) = 1$ |
| $f(x) = \sin(x)$, $x \in [0, 2\pi]$ | $x_{\text{min}} = 4.7465294$ | $x^* = \frac{3\pi}{2}$, $f(x^*) = 4.71238898$ |
| $f(x) = 0.1x^4 - 5x^2 + x - 1$       | $x_{\text{min}} = -5.084387$ | $x^* = -5.04926936$ |

Let note that results depend on first $K$ realization of random vector $u^N$. So at some starts we receive for the function 3 wrong results: $x_{\text{min}} = 4.94423$, $f(x_{\text{min}}) = -58.525013$. This is also related to special kind of function 3. Thus, it is necessary to do several starts of the algorithm.

Further the results for the function of two variables are presented.

Input for the experiment:
1. Three functions:
   - Rosenbrock function
     
     $$f(x, y) = (1-x)^2 + 100(y-x^2)^2;$$

   - Paraboloid
     
     $$f(x, y) = x^2 + y^2;$$

   - Schaffer function № 4
     
     $$f(x, y) = 0.5 + \frac{\cos^2\left(\sin\left(\|x^2 - y^2\|\right)\right) - 0.5}{\left(1 + 0.001\left(x^2 + y^2\right)\right)^2}.$$
Figure 1. a) Rosenbrock function; b) Paraboloid; c) Schaffer function № 4.

2. Number of initial random vectors: 10, 50, 100, 500.
3. Range of random values \([L, R]: [-2, 2], [-10, 10], [-100, 100]\).

The Table 2 shows the results for the considered functions with fixed range \([-10, 10]\).

| Algorithm                | Opt. value | Opt. point | 10       | 50       | 100      | 500      |
|--------------------------|------------|------------|----------|----------|----------|----------|
|                          |            |            | Value    | Point    | Value    | Point    | Value    | Point    | Value    | Point    |
| Rosenbrock function      | 0          | (1, 1)     | 0.5692   | (0.25356, 0.07522) | 0.161    | (0.61796, 0.36947) | 0.0150    | (1.08313, 1.18217) | 0.494    | (0.72816, 0.46537) |
| Paraboloid of revolution  | 0          | (0, 0)     | 0.0009   | (-0.02892, -0.01101) | 0.000    | (0.009616, -0.027238) | 0.0015    | (-0.03523, -0.01889) | 0.000    | (0.00161, 0.00058)  |
| Schaffer function № 4    | 0.29257    | (0, 1.25313) | 0.2960   | (-1.62975, -2.28981) | 0.292    | (1.24522, -0.13423) | 0.2944    | (1.27839, 0.00557) | 0.292    | (-0.25967, -1.22475) |

The best results are obtained with the number of vectors equal to 100.

The Table 3 shows the results with fixed number of random vectors – 100 and various range.

| Algorithm                | Opt. value | Opt. point | \([-2, 2]\) | \([-10, 10]\) | \([-100, 100]\) |
|--------------------------|------------|------------|--------------|----------------|----------------|
|                          |            |            | Value        | Point          | Value        | Point          | Value        | Point          |
| Rosenbrock function      | 0          | (1, 1)     | 0.0148       | (1.11298782, 1.23419407) | 0.216225   | (0.891699, 0.840348) | 0.6713139   | (0.489331, 0.303499) |
| Paraboloid of revolution  | 0          | (0, 0)     | 0.0002       | (0.0002725, -0.003473) | 0.00013   | (-0.010759, 0.004885) | 0.0069     | (0.037637, -0.074355) |
| Schaffer function № 4    | 0.292579   | (0, 1.25313) | 0.2926     | (-0.4703473, -1.1573851) | 0.2925776 | (0.724128, -1.022993) | 0.2925788   | (1.278391, 0.005570) |
Generally, with increasing interval, the accuracy falls. This is especially well seen for the simple function – paraboloid.

The running time of the algorithm is shown in the table 4.

| Number of initial random vectors | Running time |
|---------------------------------|-------------|
| 10                              | 0.1175      |
| 100                             | 1.4126      |
| 1000                            | 12.1006     |

Conclusions from the computational experiment
1. The algorithm always converges.
2. The algorithm is retrained with a large number of initial vectors. However small values do not provide enough information. Thus optimal value of the number of initial vectors is in the interval [50, 200].
3. Expanding the range for some functions leads to better solution. However simple functions are retrained.
4. The stop-criterion (9) guarantees convergence but don’t guarantees convergence to the optimal point.

6. Conclusions
Structural components of the methods of a nonlocal search were identified and analyzed using the first variation of the functional. They form the basis for algorithmic schemes for the development of optimization methods, which use only objective function values.

References
[1] Kaplinskii A I, Limarev A E and Chernyshova G D 1980 Building randomized optimization algorithms R nd S u (Riga, Zinatne) 8 pp 63-91 (in Russian)
[2] Kaplinskii A I and Propoi A I 1990 S u u p n n f n n d u in Preprint Inst. of Syst. Res. p 30 (in Russian)
[3] Kaplinskii A I and Propoi A I 1993 Nonlocal optimization methods that use potential theory u i n nd R C n 7 pp. 55-66
[4] Kaplinskii A I and Propoi A I 1994 First-order nonlocal optimization methods that use potential theory u i n nd R C n 7 pp. 94-103
[5] Tregubov A G and Medvedev S N 2015 u i n f n S n i v i y S i Sy n y i nd inf i n n i 4 pp 31-37 (in Russian)
[6] Tregubov A G and Medvedev S N 2018 u i n f n S n i v i y S i Sy n y i nd inf i n n i 4 pp 35-43 (in Russian)