An Integrated Evolutionary Algorithm for Expensive Global Optimization

Changtong Luo, Chun Wang, Zonglin Jiang
Institute of Mechanics
Chinese Academy of Sciences
Beijing 100190, China
luo@imech.ac.cn

Shao-Liang Zhang
Department of Computational Science and Engineering
Nagoya University
Nagoya 464-8603, Japan
zhang@na.cse.nagoya-u.ac.jp

***Abstract*—We propose an integrated algorithm named low dimensional simplex evolution extension (LDSEE) for expensive global optimization in which only a very limited number of function evaluations is allowed. The new algorithm accelerates an existing global optimization, low dimensional simplex evolution (LDSE), by using radial basis function (RBF) interpolation and tabu search. Different from other expensive global optimization methods, LDSEE integrates the RBF interpolation and tabu search with the LDSE algorithm rather than just calling existing global optimization algorithms as subroutines. As a result, it can keep a good balance between the model approximation and the global search. Meanwhile it is self contained. It does not rely on other GO algorithms and is very easy to use. Numerical results show that it is a competitive alternative for expensive global optimization.**

Keywords. expensive global optimization; evolutionary computation; low dimensional simplex evolution; radial basis function; response surface; tabu search

I. Introduction

Global optimization (GO) is to find a best solution to a given problem, or mathematically, to find a vector $\boldsymbol{x}^*$ within a feasible region $\Omega \subset \mathbb{R}^n$ such that $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \Omega$. GO is a challenging task because the gradient-based algorithms such as quasi-Newton methods and nonlinear conjugate gradient methods will get stuck at a stationary point or a local minimum. During the past decades, great efforts have been made on GO, and a number of excellent methods for global optimization have been proposed including branch-and-bound (BB), adaptive simulated annealing (ASA) [9], covariance matrix adaptation evolution strategy (CMA-ES) [7], differential evolution (DE) [18], particle swarm optimization (PSO) [13], and low dimensional simplex evolution (LDSE) [14], etc. Despite their prominent efficiency, they still require a large number of function evaluations to escape the local minima.

However, in many engineering applications, the objective function is very costly to evaluate. For example, in optimal shape design, for a given design vector ($\boldsymbol{x} = (x_1, \cdots, x_n)$), we need a computational fluid dynamics (CFD) simulation to get its corresponding performance $f(\boldsymbol{x})$, and the CFD simulation may take several minutes, several hours, or even several days. The CFD simulation can be regarded as a black-box function in the optimization problem. We want to find a reasonably good vector $\boldsymbol{x}^*$ within a very limited number of CFDs. In these cases, general GO algorithms do not work, and some problem-specific methods are needed.

Global optimization of black-box functions that are costly to evaluate (known as expensive GO) has drawn much attention in the field of engineering applications during the past decades. In 1993, Jones et al. provided a dividing rectangles (DIRECT) method [11], in which only the function value is used (derivative-free). Later, to take the advantage of the inherent smoothness of the target problems, a class of response surface based GO algorithms have also been presented. Powell used a multivariate polynomial interpolation model within a trust-region framework [16]. Jones et al. used a kriging model [12]. Ishikawa et al. used a radial basis function (RBF) model [10]. Among them the radial basis function (RBF) model is very promising due to its simplicity and stability. CORS-RBF by Regis et al. [17], RBF-G by Gutmann [6] and ARBF by Holmström [8] are all based on RBF models.

Generally, an expensive GO algorithm (e.g., CORS-RBF or ARBF) focus its attention on model approximation and call existing GO algorithms as subroutines. Thus it is difficult to keep a balance between the model approximation and the global search. In this paper, we try to overcome this shortcoming by integrating the RBF interpolation and tabu search with an existing global optimization, low dimensional simplex evolution (LDSE) [14]. Numerical results show that the resulting algorithm is a competitive alternative for expensive global optimization.

II. A brief review of low dimensional simplex evolution (LDSE)

Low dimensional simplex evolution (LDSE) [14] is a real-coded evolutionary algorithm (EA) for box-constrained global optimization of the form

$$\min_{\boldsymbol{x}\in\Omega} f(\boldsymbol{x})$$

Similar to other population-set based algorithms, LDSE maintains a population set $\vec{X}(t)$ of N individuals (points in $\mathbb{R}^n$) $X_i(t)$, $i = 1, 2, \cdots, N$, during the evolutionary progress. The evolutionary progress is to drive these points to the vicinity of the global minimizer. The drive is done by

978-1-4244-5959-9/10/$26.00 ©2010 IEEE

replacing all *bad* points in the current population with new better points from generation to generation.

For each individual in the current population, $m+1$ individuals are randomly selected to form a m-simplex, where $m \ll n$. Each individual $X_i(t)$ tries to improve himself in a framework of try-try-struggle, which will be described as follows. In the beginning, it has two chances. The first chance is provided by the simplex reflection. If the reflection point X_r is better than $X_i(t)$, $X_i(t)$ will be replaced by X_r. As a result, the individual $X_i(t)$ is promoted. Otherwise a second chance, the simplex contraction, will be carried out. Similarly, $X_i(t)$ will get promoted if the contraction point X_c is better. However, if the individual $X_i(t)$ has lost the previous two chances and still cannot achieve the average profit (that is, its function value is greater than or equal to the average value of the current population), it will take its last struggle. The procedure of the LDSE can be outlined as follows.

Procedure LDSE algorithm:

Step 1. Initialize: Input population size N, initial bounds $\boldsymbol{l}$, $\boldsymbol{u}$, scaling factors α and β. Set the current generation $t := 0$; And initialize population $\vec{X}(0) = \{X_1(0), X_2(0), \cdots, X_N(0)\}$, where $X_i(0) \in \mathbb{R}^n$.

Step 2. Evaluate population: For each individual in the current population $\vec{X}(t)$, compute $f(X_i(t))$; Set the current position $i := 1$.

Step 3. Update population: If the current position $i \leq N$, perform the following steps.

3.1) Construct simplex: Randomly choose $m+1$ mutually different individuals X_{r_i}, $i = 1, 2, \cdots, m+1$ from current population, find their best X_b and the worst X_w, and calculate the centroid $\bar{X} = \frac{1}{m}\sum_{r_i \neq w} X_{r_i}$.

3.2) Try reflection: Compute the reflection point $X_r = \bar{X} + \alpha \cdot (\bar{X} - X_w)$.
If $f(X_r) < f(X_i(t))$, then $X_i(t+1) = X_r$,
set the current position $i := i+1$, and return to step 3.

3.3) Try contraction: Compute the contraction point $X_c = \bar{X} + \beta \cdot (X_w - \bar{X})$; If $f(X_c) < f(X_i(t))$, then $X_i(t+1) = X_c$,
set the current position $i := i+1$, and return to step 3.

3.4) Struggle: If $f(X_i(t)) \geq \frac{1}{N}\sum_i f(X_i(t))$ then compute the struggle point

$$X_s = \begin{cases} X_i(t) + 0.618 \cdot (X_b(t) - X_i(t)), \\ \qquad\qquad \text{if } f(X_{r_b}(t)) < f(X_i(t)); \\ X_i(t) + 0.382 \cdot (X_i(t) - X_w(t)), \text{ else.} \end{cases}$$

let $X_i(t+1) = X_s$, set the current position $i := i+1$, and return to step 3.

Step 4. Check point: If some stopping criterion is satisfied, output the best-so-far individual X^* and its function value $f(X^*)$; Otherwise, set the current generation $t := t+1$, set the current position $i := 1$ and return to step 3.

It can be seen from the above procedure, LDSE hybridizes EA and Nelder-Mead method with essential modifications. It generates new trial points in a Nelder-Mead way, and the individuals survive by the rule of natural selection. However, the simplex therein is low dimensional and real-time constructed; And the simplex operators are employed selectively (i.e., the expansion and reduction operators are discarded) and a new simplex operator (the last struggle) is introduced. Meanwhile, each individual is updated in a framework of try-try-struggle.

LDSE has shown its prominent performance over an improved version of differential evolution (DE) [1], [14].

III. Model approximating issues

The idea of response surface methods is to approximate the costly target function $f(\boldsymbol{x})$ with a series of model functions $m_k(\boldsymbol{x})$ $(k = p, p+1, p+2, \cdots)$ that are much cheaper to evaluate (usually it takes much less than one second). Hopefully, we have $m_k(\boldsymbol{x}) \to f(\boldsymbol{x})$ as $k \to \infty$. Then the model functions are used to identify the promising points for the target function. In other words, the expensive global optimization problem

$$\min_{\boldsymbol{x} \in \Omega} f(\boldsymbol{x}) \tag{1}$$

is reduced into a series of cheap global optimization problems

$$\min_{\boldsymbol{x} \in \Omega} m_k(\boldsymbol{x}), k = p, p+1, p+2, \cdots \tag{2}$$

The model functions (or the response surfaces) might be constructed using the multivariate polynomial interpolation model, the kriging model, or the radial basis function (RBF) model as above mentioned in section 1. Users can also choose other approximation models for response surface construction. We choose the radial basis function (RBF) model in this work because of its simplicity and stability. Suppose we have k distinct points $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_k$ with known function values $f_i = f(\boldsymbol{x}_i)$, $i = 1, \cdots, k$, then the model function $m_k(\boldsymbol{x})$ has the form

$$m_k(\boldsymbol{x}) = \sum_{i=1}^{k} \omega_i \Phi(\boldsymbol{x} - \boldsymbol{c}_i).$$

We use multiquadric type basis function $\Phi(r) = \sqrt{r^2 + \beta^2}$ in our experiments. For more information of RBF, refer to [2].

Keep in mind that the assumption $k \to \infty$ is impractical because we want to optimize the costly target function within a very limited number of function evaluations, i.e., k is upper bounded, $k < K$. Usually the upper bound K is limited to thousands, hundreds, or even dozens. We need to find a reasonably good vector $\boldsymbol{x}^*$ within K costly evaluations.

In fact, we can get a reasonably good vector $\boldsymbol{x}^*$ provided that

(1) the response surface $m_k(\boldsymbol{x})$ is good enough, and

(2) the global optimization search (over $m_k(\boldsymbol{x})$) is reliable.

Usually, the first assumption (1) is more critical since it is difficult to get a good response surface with a very limited number of function evaluations. This makes it necessary to keep a good balance between the model approximation (i.e.,

the construction of the response surface) and the global search during the optimization process.

IV. LDSE Algorithm with RBF and Tabu Techniques

Our new algorithm is referred to as low dimensional simplex evolution extension (LDSEE, pronounced LDC). It is based on the previously described low dimensional simplex evolution (LDSE), and uses radial basis function (RBF) interpolation as the response surface model. To get a good response surface for the approximation of the costly target function, the idea from tabu search [5] is also applied.

LDSEE starts from an initial RBF response surface determined by a set of initial points. The LDSE algorithm with tabu search is applied to the RBF model

$$\min_{\boldsymbol{x}\in\Omega} m_p(\boldsymbol{x})$$

to find a promising point. This step is referred to as global search. By promising we mean the point can hopefully improve the response surface or help to locate a better point for the target function. Then the new got promising point will be added to the tabu list and used to update the RBF response surface. This step is referred to as surface construction. Repeat the above global search and surface construction steps until the limited number of number of costly function evaluation is used up. Meanwhile, LDSEE puts emphasis on the surface construction at the beginning and the global search in the end. The procedure of the LDSEE can be outlined as follows.

Procedure LDSEE:

Step 1. (Initialization) Input the number of initial points p, the maximum number of expensive function evaluations K, number of nodes at the j-th direction N_j. Select a set of initial vectors $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_p$ and calculate their target function values $f_i = f(\boldsymbol{x}_i)$, $i = 1, 2, \cdots, p$. Find the best vector $\boldsymbol{x}^*$ and its target value f^*. Let the tabu list $T = \{\boldsymbol{x}_i | i = 1, 2, \cdots, p\}$, set $k := p$ and the initial tabu radius $\delta := (\delta_1, \cdots, \delta_n)$, where $\delta_j = \frac{1}{2(N_j+1)} \cdot 0.9, j = 1, 2, \cdots, n$.

Step 2.(Surface construction) Construct a RBF response surface $m_k(\boldsymbol{x})$ with known data $(\boldsymbol{x}_i, f_i)$, $i = 1, 2, \cdots, k$.

Step 3. (Global search) Use the LDSE algorithm (see section II) for solving the cheap problem

$$\min_{\boldsymbol{x}\in\Omega} m_k(\boldsymbol{x})$$

with the tabu radius δ to get a new promising point $\boldsymbol{x}_{k+1}$.

Step 4. (Target evaluation) Calculate the target function value at the promising point, i.e., $f_{k+1} = f(\boldsymbol{x}_{k+1})$, and update the best vector and its target value $(\boldsymbol{x}^*, f^*)$ if $f_{k+1} < f^*$.

Step 5. (Stop Checking) If $k \geq K$, stop and output the best vector $\boldsymbol{x}^*$ and its target value f^*. Otherwise, set $k := k + 1$, update the tabu list $T = \{\boldsymbol{x}_i | i = 1, 2, \cdots, k\}$, update the tabu radius δ, where $\delta_j = \frac{1}{2(N_j+1)} \cdot 0.9 \cdot (1 - \frac{k-p}{K-p}), j = 1, 2, \cdots, n$, and return to step 2.

Different from the original LDSE reviewed previously in section II, LDSEE uses the RBF response surfaces to locate the minimum of the target function and requires that every new generated individual must keep a dynamic distance away from the points in the tabu list.

V. Numerical results

The new algorithm LDSEE is implemented in C++. As an integrated evolutionary algorithm, LDSEE is self contained. It does not rely on other GO algorithms and is very easy to use.

To test the performance of LDSEE, we choose a set of box-constrained multimodal functions (see table I) to do our numerical experiments. Most of these functions (the last five) are from Dixon-Szegö [3], and frequently cited to test the performance of expensive GO algorithms [11], [17], [8]. The first one dimensional function (1D) is defined as $f(x) = -(((3 * x - 1) * \sin x - 2) * \cos x - 1)$. The second function (Peaks) is the logo function of Matlab (a popular numerical computing programming language, developed by the MathWorks). The third function (2-D six-hump camel back) is a multimodal function frequently used to test GO algorithms [18]. In table I, test function name, its abbreviation, dimension, domain, and the number of local minima are denoted as func.name, abbr, dim, domain, no.min respectively. As described in [17], these functions are not really costly to evaluate, but their multimodal property is similar to that of the real world costly functions. Therefore, the performance on these test functions is expected to mimic the performance on the real world costly functions.

TABLE I
Characteristics of test functions

func.name	abbr	dim	domain	no.min
1D	1D	1	[-4,4]	4
Peaks	PK	2	$[-4,4]^2$	2
Six-hump camel back	CB	2	$[-4,4]^2$	6
Branin	BR	2	$[-4,4]^2$	6
Goldstein-Price	GP	2	$[-2,2]^2$	5
Hartman-3	H3	3	$[0,1]^3$	5
Sheckel-10	S10	4	$[0,10]^4$	10
Hartman6	H6	6	$[0,1]^6$	5

In our numerical experiments, uniform inner grid points are used as initial points. For example, if we consider a function within $[0, 1] \times [-1, 0]$ and we want to seed two sample points at each direction, there will be four initial points $(0.333, -0.333)$, $(0.333, -0.666)$, $(0.666, -0.333)$ and $(0.666, -0.666)$. Note that the number of inner grid points will be quite large for high dimensional problems. To avoid unnecessary sampling, experimental design method such as Latin Hypercube [15] and Uniform Design [4] might be applied. However, according to our experience, too few initial points might result in unreliable results.

The control parameters of LDSEE and the performance are listed in table II, where the abbreviation of function name, the number of initial points, the number of search points, the actual global minimum, the obtained minimum, and the relative residue are denoted as func, no.ini, no.sch, act.min, obt.min and rel.res respectively. The control parameters of

the global search part are set as follows. The scaling factors $\alpha = 1.0$, $\beta = 0.333$. The population size $N = 50$ and the maximum generations $M = 200$. They are large enough to make sure that the global search is reliable. It can be seen from table II, the results are very encouraging.

TABLE II
SETTINGS AND PERFORMANCE OF LDSEE ALGORITHM

func	**no.ini**	**no.sch**	**act.min**	**obt.min**	**rel.res**
1D	4	10	-6.80484	-6.80484	0.00002%
PK	4	10	-0.6551	-0.6542	0.14%
CB	4	10	-1.031	-1.027	0.39%
BR	4	10	0.398	0.399	0.16%
GP	9	20	3.0	3.0	0.0001%
H3	8	20	-3.862782	-3.82629	0.94%
S10	16	40	-10.5364	-10.4514	0.81%
H6	64	30	-3.322368	-3.32143	0.028%

To compare with other existing expensive GO algorithms, we cite the results from Regis et al. [17]. The comparison results are listed in table III, where the DIRECT is presented by Jones et al. [11], and the RBF-G is presented by Gutmann [6]. The best result of each test function is marked in bold. It shows that LDSEE performs the best on 3 of 5 functions. For the other two functions (H3 and S10), performance of LDSEE is very close to the best. Note that we use a preset number of search points for LDSEE, thus the obtained relative error might be much less than 1% (e.g., GP and H6). If we stop at 1%, the required number of function evaluations might be smaller. We can conclude that LDSEE is a competitive alternative for expensive global optimization.

TABLE III
NUMBER OF FUNCTION EVALUATIONS NEEDED TO ACHIEVE A FUNCTION VALUE WITH RELATIVE ERROR LESS THAN 1% FOR DIFFERENT EXPENSIVE GO ALGORITHMS

func	DIRECT	RBF-G	CORS-RBF(SP1)	CORS-RBF(SP2)	LDSEE
BR	63	44	34	40	**14**
GP	101	63	49	64	**29**
H3	83	43	**25**	61	28
S10	97	**51**	**51**	64	56
H6	213	112	108	104	**94**

To help the readers a better understand on the difference between the expensive GO algorithms and the general GO algorithms (for cheap function optimization), we give the number of function evaluations needed to achieve a function value with relative error less than 0.1% for different algorithms in table IV. This does NOT mean LDSEE is better than LDSE, DE or PSO because they are designed for the different kind of problems. In fact, LDSEE is designed for expensive GO while the others are for general GO. In case the objective is cheap to evaluate, LDSEE might consume more CPU time and memory.

TABLE IV
NUMBER OF FUNCTION EVALUATIONS NEEDED TO ACHIEVE A FUNCTION VALUE WITH RELATIVE ERROR LESS THAN 0.1% FOR PSO, DE, LDSE AND LDSEE ON THE SIX-HUMP CAMEL BACK FUNCTION

Algorithm	**PSO**	**DE**	**LDSE**	**LDSEE**
no.eval	639	210	187	14

VI. CONCLUSION

We have presented a new algorithm named low dimensional simplex evolution extension (LDSEE) for expensive global optimization. It integrates the radial basis function (RBF) interpolation and tabu search with an existing algorithm, low dimensional simplex evolution (LDSE). As an integrated algorithm, it can keep a good balance between the model approximation and the global search. LDSEE is self contained. It does not rely on other GO algorithms and is very easy to use. Numerical results indicate that it is a very promising algorithm.

In practical applications of LDSEE, parallelization design is sometimes necessary for both the global search and the costly function evaluations. Meanwhile, nonlinear constraint handling is another problem to be concerned. These topics are left for our future studies.

ACKNOWLEDGEMENT

This work was partially supported by the National Natural Science Foundation of China (10632090, 90916028).

REFERENCES

[1] Ali, M.M., Khompatraporn, C. and Zabinsky, Z.B.: A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. J. Glob. Optim. 31, 635-672 (2005)

[2] Buhmann, M.D.: Radial Basis Functions, Theory and Implementations. Cambridge University Press, Cambridge (2003)

[3] Dixon, L.C.W. and Szegö G.P.: The global optimisation problem: an introduction. In: Dixon, L., Szego G. (eds.) Toward Global Optimization, vol. 2, pp. 1-15. New York (1978)

[4] Fang, K.T. and Wang, Y.: Number-Theoretic Methods in Statistics. Chapman & Hall, London (1994)

[5] Glover, F. and Laguna M.: Tabu Search. Kluwer Academic Publishers, Boston (1997)

[6] Gutmann, H.-M.: A radial basis function method for global optimization. J. Global Optim. 19, 201-227 (2001)

[7] Hansenand, N. and Ostermeier, A.: Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. in: Proc. IEEE Conf. Evol Comput., 312-317 (1996)

[8] Holmström, K.: An adaptive radial basis algorithm (ARBF) for expensive black-box global optimization, J. Global Optim. 41, 447-464 (2008)

[9] Ingber, L.: Simulated Annealing: Practice Versus Theory. J. Math. Comput. Modelling 18, 29-57 (1993)

[10] Ishikawa, T. and Matsunami, M.: An optimization method based on radial basis functions. IEEE Trans. Magn. 33(2), 1868-1871 (1997)

[11] Jones, D.R., Perttunen, C.D. and Stuckman, B.E.: Lipschitzian optimization without the Lipschitz constant. J. Opt. Theory and Appl. 79, 157-181 (1993)

[12] Jones, D.R., Schonlau, M. and Welch, W.J.: Efficient global optimization of expensive black-box functions. J. Global Optim. 13(4), 455-492 (1998)

[13] Kennedy, J. and Eberhart, R.: Particle Swarm Optimization. in: IEEE Int. Conf. Neural Networks Conf. Proc., Piscataway, NJ, vol.4, pp. 1942-1948 (1995)

[14] Luo, C.T. and Yu, B.: Low dimensional simplex evolution - A hybrid heuristic for global optimization. in: Proc. Eighth ACIS Int. Conf. Softw. Eng. Artif. Intell. Netw. Parallel Distrib. Comput. 2, 470-474 (2007)

[15] McKay, M., Beckman, R. and Conover, W.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 21, 239C246 (1979)

[16] Powell M.J.D., UOBYQA: unconstrained optimization by quadratic approximation. Math. Program. 92, 555-582 (2000)

[17] Regis, R.G. and Shoemaker, C.A.: Constrained global optimization of expensive black box functions using radial basis functions. J. Global Optim. 31(1), 153-171 (2005)

[18] Storn, R. and Price, K.: DE - A simple and efficient heuristic for global optimization over continuous space. J. Glob. Optim. 11, 341-359 (1997)