International Journal of Advances in Intelligent Informatics
Vol 1, No 2, July 2015, pp. 90-97
ISSN: 2442-6571

Automatic differentiation based for particle swarm optimization steepest descent direction

Aris Thobirin[a,1], Iwan Tri Riyadi Yanto[b,2,*]

[a] *Department of Mathematics, University of Ahmad Dahlan,Yogyakarta*
[b] *Department of Information Systems, University of Ahmad Dahlan,Yogyakarta*
[1]*aris.thobi@math.uad.ac.id;* [2] *yanto.itr@is.uad.ac.id**

ARTICLE INFO

Article history:
Received July 2, 2015
Revised July 25, 2015
Accepted July 30, 2015

Keywords:
Particle swam optimization
Descent direction,
Automatic Differentiation.

ABSTRACT

Particle swam optimization (PSO) is one of the most effective optimization methods to find the global optimum point. In other hand, the descent direction (DD) is the gradient based method that has the local search capability. The combination of both methods is promising and interesting to get the method with effective global search capability and efficient local search capability. However, In many application, it is difficult or impossible to obtain the gradient exactly of an objective function. In this paper, we propose Automatic differentiation (AD) based for PSODD. We compare our methods on benchmark function. The results shown that the combination methods give us a powerful tool to find the solution.

I. Introduction

The Particle swarm Optimization (PSO) [1] is a population-based, self adaptive search optimization method motivated by the observation of simplified animal social behavior. It is becoming very popular due to its simplicity of implementation and ability to quickly converge to reasonably good solution [2]–[4]. Especially, global search capability of the method is very powerful. The particle swam optimization utilizes common knowledge of the group and individual experience effectively. That is, direction for the best estimator that a particle has ever reached, direction for the best one that all particle have ever found and momentum are successfully combined to determine the next iteration. Unfortunately, PSO show some weakness in term of balance between exploitation and exploration during the search [5]. For example in multi-objective problems, the search is not concentrated on the visited areas effectively, and often it shows a premature convergence and lack of diversification during moving from position to another. In order to solve this problem, various techniques have been proposed can be found in the literature [6], [7]. In most of the introduced techniques, extensive and intensive search are controlled by using the parameters setting. However this has an influence on the search for new solutions in case of multi-objective problems [7]. There is a popular technique which is used for evolutionary approaches, it is based on starting the search by an intensive search and then gradually explore other locations until all the search space is covered [8], [9]. However, such techniques make solving of multi-objective problems complicated especially in some situations where the search space contains many local optima.

The particle swam optimization itself does not have a capability searching the neighbor of the position and it may miss the optimal point near the present position because the method does not use local information of the function. Even if a particle is close to a global optimal, the particle moves based on the three factors described above. As a result, efficiency of the particle swam optimization may be limited in some cases. It seems better to search neighbor area carefully. To do so, local information such as gradient is necessary. On other hand, The descent direction search local area [10][11], [12] based on the gradient of the function. If the local search capability of the descent direction can be added to global search one of the particle swam optimization, we have a useful optimization method with global search capability and efficient local search ability at the same time. Therefore, combination of the particle swam optimization and the descent direction method is

promising and interesting approach. However, The descent direction method requires the gradient that is the first derivative of the function. In many application, it is difficult or impossible to obtain the gradient exactly of an objective function. In order to overcome the drawback, the Automatic Differentiation [13] is introduced. It is upcoming technology which provides software for automatic computation of derivatives of a general function provided by the user. Automatic Differentiation is also called computation differentiation or algorithmic differentiation [13], [14]. The idea is that basic derivative rules from calculus, such as the chain rule, can be implemented in a numerical environment.

II. Related Works

A. Descent Direction

The optimization problem is to find a optimum point of an objective function $f(x) \in R^n$ with an adjustable n dimension parameter vector $x \in R^n$. The steepest descent method (also called Cauchy's method or gradient method) is one of the oldest and simplest procedures for minimization of a real function defined on R^n [10][11], [12], [15]. It is also the departure point for many other more sophisticated optimization procedures. The iteration is given as in (1).

$$x(t+1) = x(t) + a(t)d(t), \quad t = 0,1, \ldots \tag{1}$$

where $d(t)$ is a descent direction $a(t)$ is a scalar of step length. The gradient vector of a function points towards the direction in which locally the function is increasing the most rapidly. A natural choice for the descent direction is to use the negative gradient direction. The function is changing most rapidly in this direction, it is known as the steepest descent direction [10], the direction is defined as in (2).

$$d(t) = -\nabla f(x(t)) \tag{2}$$

B. Particle Swam Optimization

Particle Swarm Optimization (PSO) [1] is a population-based, self adaptation search optimization method motivated by the observation of simplified animal behaviors. PSO searches for optimal solution via collaborating with individuals within a swarm of population. Each individual, called Particle or Agent is made of two parts, the position and velocity. For an n_dimensional problem and a swarm of m particles, the i^{th} particle's position and velocity, in general, are denoted as $x_i[x_{i1}, x_{i2}, \ldots, x_{in}]^T$ and $V_i = [V_{i1}, V_{i2}, \ldots, V_{in}]^T$ for $i = 1,2, \ldots, m$, respectively. Consider on the inertia weight PSO, the algorithm of the particle swarm optimization is described as in (3).

$$x_i(t+1) = x_i(t) + V_i(t+1) \tag{3}$$

where

$$V_i(t+1) = \chi\left(\omega V_i(t) + \phi_1\big(p(t) - x_i(t)\big) + \phi_2\big(g(t) - x_i(t)\big)\right) \tag{4}$$

The parameter vector $x_i(t)$ denote an estimator of the optimum point at the t^{th} iteration. $V_i(t)$ is called a velocity vector, that is, a modifying vector for the parameter vector (momentum for the next iteration). $p(t)$ is the best estimator that this particle has ever reached. $g(t)$ is the best one that all the particles have ever found until the t^{th} iteration. $p(t)$ and $g(t)$ are called personal best and global best, respectively. The coefficient ϕ_1 and ϕ_2 are two positive random number in a certain range using uniform distribution with upper limitation to decide a balance between the individual best estimator and the swarm best one. ω denotes a coefficient to adjust the effect of the inertia and χ is a gain coefficient for the update.

III. Basic of Automatic differentiation

Automatic differentiation (AD) [13] is a set of techniques for transforming a program that calculates numerical values of a function, into a program which calculates numerical values for derivatives of that function with about the same accuracy and efficiency as the function values themselves [14]. The basic process of AD is to take the text of a program (called the underlying program) which calculates a numerical value, and to transform it into the text of program (called the transformed program) which calculates the desired derivative values. The transformed program carries out these derivative calculations by repeated use of the chain rule from elementary calculus, but applied to floating point numerical values rather than to symbolic expressions [14]. AD is a chain-rule-based technique for evaluating the derivatives with respect to the input variables of functions defined by a high-level language computer program. AD has two basic modes of operations, the forward mode and the reverse mode. Suppose that f as in [14] is an underlying program which takes n independent variables x_i as inputs, and produce m dependent variables y_i as outputs, and the Jacobean $J = f^{'} = \partial y_i / \partial x_j$ given particular values for x_i, want to be obtained. The forward mode associates with each floating point program variable v a vector $\dot{v}$ of floating point derivatives value. Conceptually, the case is when each dot vector $\dot{v}$ contains one component for each independent variable x_i and component i contains the corresponding derivatives $\partial v / \partial x_i$, so that $\dot{v} = \nabla_x v$. The reverse mode associates with each floating point program variable v a vector $\bar{v}$ of floating point derivatives values. Conceptually, the case is when each of these bar vector contains one component for each dependent variable, and component i contains the corresponding derivative $\partial y_i / \partial v$, so that $\bar{v} = D_v y$[14].

IV. Proposed approach

Global optimal can be obtained using the particle swarm optimization. However, since the particle swarm optimization itself does not have a capability searching the neighbor of the position and it may miss the optimal point near the present position. Thus, efficiency of the particle swarm optimization may be limited in some cases. The Steepest descent direction method search only local area rapidly. If the local method of steepest descent direction can be added to the global search of the particle swarm optimization, a useful optimization method with good global search capability and efficient local search ability at the same time will be gotten. In this paper, we propose some schemes of combinations the particle swarm optimization with the steepest descent direction by automatic differentiation consider to the minimization problem.

A. *Case 1*

The particle swarm optimization with the the steepest descent direction is combined directly that is the velocity (momentum) of PSO use the steepest descent direction term. The steepest descent direction is used to change the direction of modification. The equation is defined as in (5).

$$V_i(t+1) = \chi\left(-\alpha \nabla f\big(x(t)\big) + \phi_1\big(p(t) - x_{i(t)}\big) + \phi_2\big(g(t) - x_{i(t)}\big)\right) \quad (5)$$

In this scheme, all individuals have the same characteristics since the gradient is applied for all particles.

B. *Case 2*

If the best particle is close to the global minimum, and this is likely, the best particle had better search neighbor of the present point carefully. Then, modification based on the original particle swarm optimization is not suitable for this particle. The gradient type of method is suitable. Therefore, The steepest descent direction are applied only to the best particle. All the other individuals are updated by the ordinary particle swarm optimization.

C. *Case 3*

The particle swarm optimization and the steepest descent direction are mixed. That is, in every iteration, half of individuals in the population are updated by the particle swarm optimization, left half particles are modified only by the steepest descent direction. All the individuals select the

particle swarm optimization or the steepest descent direction randomly with probability of 0.5 in every iteration.

D. Case 4

Basically, we use the scheme 3. However, the best individual is updated only by the steepest descent direction since the best particle has a good chance to be a neighbor of a global minimum.

V. Comparison

In order to evaluate performance of these algorithms, The benchmark functions are used. These functions have their inherent characteristics about local minimum or slope. The algorithm are implemented in MATLAB version 7.6.0.324 (R2008a). They are executed sequentially on a processor Intel Core 2 Duo CPUs. The total main memory is 1G and the operating system is Windows XP Professional SP3.Comparisons are carried out for ten-dimensional case, that is, $n = 10$ for all test functions. 30 particles are included in the population. Change of average means that an average of the best particle in 30 particles at the iteration for 20 trials are shown.

Table 1. The parameter setting for experimental of bencmarks function

φ_1	φ_2	χ	ω	a off all case
2	1	1	0.9	0.00003

A. Rastrigin function

The Rastrigin function is described as in (6).

$$f(x) = 10n + \sum_{i=1}^{n} x_i^2 - 10\cos 2\pi\, x_i \tag{6}$$

Fig. 1 is the shape of this function for 2 demission case. As shown in the figure 1, this function contains many local minimum points. It is generally difficult to find a global minimum using the gradient type of the method. It is difficult also for the particle swarm optimization to cope with the function. The value of the global minimum of the function is 0. Using the setting are given in the Table 1. we compare these four methods and the ordinary particle swarm optimization. The results of the change of the best particle for Rastrigin function are given in Fig. 2. Fig. 2 shows that the scheme 1 and 3 have better performance.

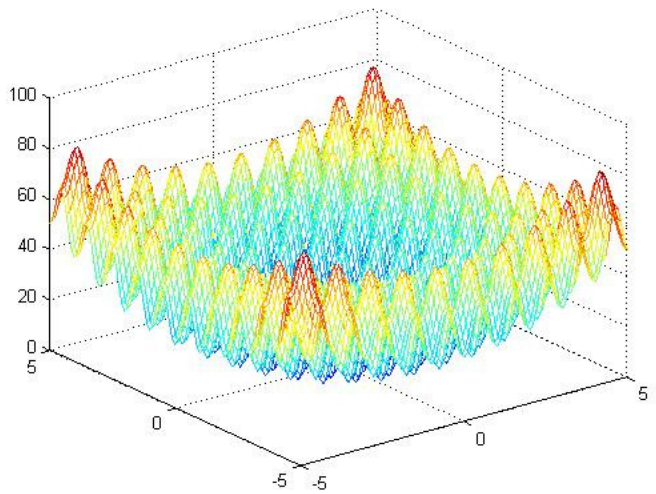

Fig. 1. Rastrigin function

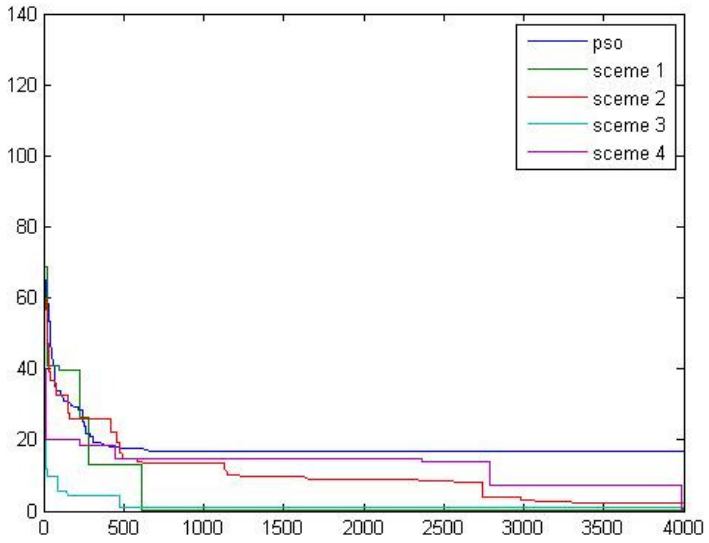

Fig. 2. Change of the best particle for rastrigin function

B. Rosenbrock's valley function

The Rosenbrock's valley is a classic optimization problem,also known as banana function. The function is defined in (7).

$$f(x) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \quad (7)$$

Fig. 3 is the shape of this function for 2 demission case. Using the setting are given in the Table 1, we can see that scheme 2, 3 and 4 match for this function as in Fig. 4.

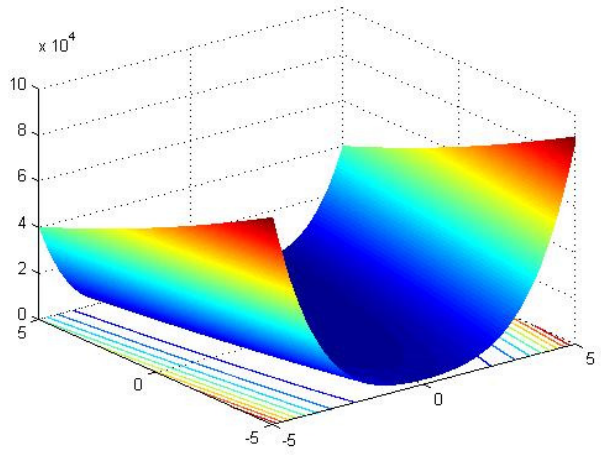

Fig. 3. Rosenbrock's valley function

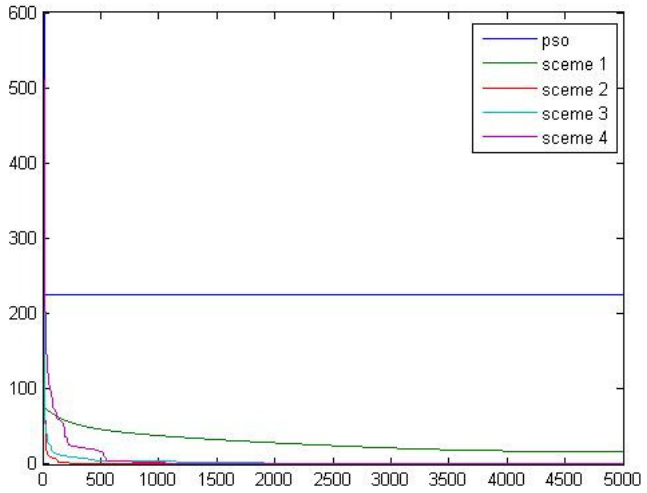

Fig. 4. Change of the best particle for Rosenbrock's valley function

C. 2^n *minima function*

The 2^n minima function is described as in (8).

$$f(x) = \sum_{i=1}^{n} x_i^4 - 16x_i^2 + 5x_i \quad (8)$$

Fig. 5 is the shape of this function for 2 demission case. The function has some local minimum points and relatively flat bottom. The value of the global minimum of the function is -783.32. Using the setting are given in the Table 1, The comparison results for 2^n minima function are given in Fig. 6. It seems that scheme 3 and 4 have good performance for this function.

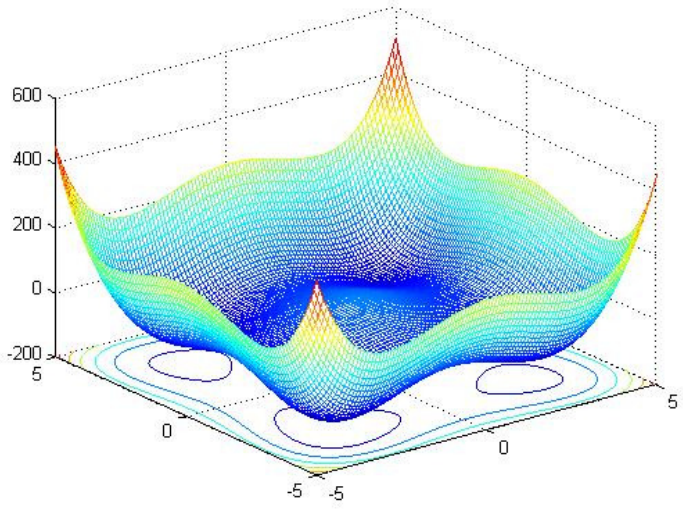

Fig. 5. 2^n minima function

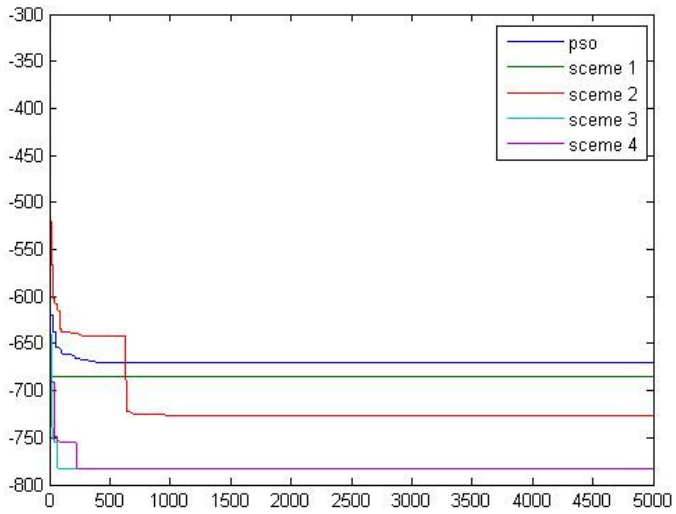

Fig. 6. Change of the best particle for 2^n minima function function

VI. Conclusion

In this paper, we have proposed an combination of particle swarm optimization method and gradient methods to solve global optimization problem. The both methods are combined directly in forth's schemes. Further, we compare our methods on benchmark function. The results shown that the combination methods give us a powerful tool to find the solution.

References

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proceedings of ICNN'95 - International Conference on Neural Networks, vol. 4, pp. 1942–1948.

[2] H. Shen, X. Peng, J. Wang, and Z. Hu, "A Mountain Clustering Based on Improved PSO Algorithm," in Advances in Natural Computation SE - 58, vol. 3612, L. Wang, K. Chen, and Y. Ong, Eds. Springer Berlin Heidelberg, 2005, pp. 477–481.

[3] Q. Li, Z. Shi, J. Shi, and Z. Shi, "Swarm Intelligence Clustering Algorithm Based on Attractor," Intelligence, no. 60435010, pp. 496–504, 2005.

[4] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," MHS'95. Proc. Sixth Int. Symp. Micro Mach. Hum. Sci., pp. 39–43, 1995.

[5] E. Cuevas, A. Echavarría, and M. Ramírez-Ortegón, "An optimization algorithm inspired by the States of Matter that improves the balance between exploration and exploitation," Appl. Intell., vol. 40, no. 2, pp. 256–272, 2014.

[6] B. Ostadmohammadi Arani, P. Mirzabeygi, and M. Shariat Panahi, "An improved PSO algorithm with a territorial diversity-preserving scheme and enhanced exploration–exploitation balance," Swarm Evol. Comput., vol. 11, no. 0, pp. 1–15, Aug. 2013.

[7] S. F. Adra and P. J. Fleming, "Diversity Management in Evolutionary Many-Objective Optimization," IEEE Trans. Evol. Comput., vol. 15, no. 2, pp. 183–195, Apr. 2011.

[8] M. Črepinšek, S.-H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms," ACM Comput. Surv., vol. 45, no. 3, pp. 1–33, Jun. 2013.

[9] I. Paenke and J. Branke, "Balancing Population- and Individual-Level Adaptation in Changing Environments," Adapt. Behav., vol. 17, no. 2, pp. 153–174, Mar. 2009.

[10] M. A. Bhatti, Practical Optimization Methods. New York, NY: Springer New York, 2000.

[11] L. M. Graña Drummond and B. F. Svaiter, "A steepest descent method for vector optimization," J. Comput. Appl. Math., vol. 175, no. 2, pp. 395–414, 2005.

[12] R. Burachik, L. M. Graña Drummond, a. N. Iusem, and B. F. Svaiter, "Full convergence of the steepest descent method with inexact line searches," Optimization, vol. 32, no. 2, pp. 137–146, 1995.

[13] L. B. Rall, Ed., Automatic Differentiation: Techniques and Applications, vol. 120. Berlin, Heidelberg: Springer Berlin Heidelberg, 1981.

[14] M. Bartholomew-biggs, S. Brown, B. Christianson, and L. Dixon, "Automatic di erentiation of algorithms," J. Comput. Appl. Math., vol. 124, pp. 171–190, 2000.

[15] J. Tang and L. Dong, "A new descent algorithm with curve search rule for unconstrained minimization," in 2010 Second International Conference on Computational Intelligence and Natural Computing, 2010, vol. 1, pp. 89–92.