International Journal of Hybrid Information Technology
Vol. 10, No.1 (2017), pp. 139-148
http://dx.doi.org/10.14257/ijhit.2017.10.1.12

Improved Differential Evolution Algorithm based on Deep Predatory Search and Secondary Gradient Acceleration

Feng Guiliang[1,2,3], CaoNing[1,2,3] and Li Zhonghua[1,2,3,*]

[1]*School of Information Science and Engineering, Hebei North University,China*,
[2] *Population Health Informazation in Hebei Province Engineering Technology Research Center, China* ,
[3] *Medical Informatics in Hebei Universities Application Technology Research and Development Center, China*
6838710@qq.com ,8018997@qq.com, lizhonghua68@126.com

Abstract

The traditional differential evolution algorithm has premature convergence problem due to the greedy strategy. To avoid this drawback, the differential evolution algorithm based on deep predation and secondary gradient acceleration is proposed. The entire search space is first used for breadth search, and a gradient acceleration trigger parameter is adopted. The local search based on gradient acceleration is then carried out for better individuals in population. The algorithm is able to converge quickly to the global optimum. Furthermore, in order to maintain the population diversity, a new differential mutation operator is designed. By comparison with existing algorithms, the proposed algorithm in this paper can effectively escape from local optimum, and avoid premature convergence. Finally, we verified our algorithm in the load allocation optimization in hot rolling mill, and the results show that the feasibility and the effectiveness of the method is promising.

Keywords: *Deep predatory, Secondary gradient acceleration, Differential evolution, Diversity variation, Load allocation*

1. Introduction

Differential Evolution (DE) is an effective optimization algorithm falls into the swam intelligence family. Unlike Genetic Algorithm, which uses binary coding, it uses real number coding. The major steps of DE are similar to that of particle swarm algorithm and artificial fish swarm intelligence [1-3]: firstly, differential mutation operation, secondly, differential crossing-over operation, thirdly, differential selection operation. The one-on-one greedy strategy is adopted for fast convergence in low dimension. However, this is not robust to local optimum and the algorithm may have the premature problem.

Many works have been done to improve DE algorithm. An automatic adjustment for searching region was proposed in reference [1], which was based on the individual evolution states and iteration number. It was similar to expert adjustment and might improve the searching of global optimum. In reference [2], the two section crossover differential evolution algorithm was proposed. The differential evolution was decomposed into two steps, and Cauchy distribution was adopted to design crossover operation and scale parameter. In reference [3], a general mutation strategy was proposed to choose new mutation operator.

Predatory Search (PS) is not a complete optimization algorithm, compared with differential evaluation and particle swarm optimization. The idea behind PS algorithm is to simulate the animal predation behavior, it gives a strategy on searching local and global optimum [4]. In this paper we propose to use PS for global search and Gradient Acceleration for local search, namely PS-GDE (Deep Predation and Secondary Gradient Acceleration based Differential Evolution) . We apply PS-GDE to the optimization of thickness allocation in hot rolling process [1], and present an effective method for on-line hot rolling load allocation.

2. PS-GDE Algorithm

2.1. Predatory Search Strategy

In PS algorithm, global search is first adopted to find an initial optimal position[5]. The local depth search is then carried out around the global initial position to find a better candidate position, otherwise the algorithm gives up the depth search and continue the global search, until the termination criteria is met.

We can see that the PS algorithm is designed to keep a balance between the global search and the local search. The timing for global breadth search and local depth search is key to the algorithm performance. A balanced solution for this problem can prevent the prematurity in breadth search and improve the effectiveness in depth search. Inspired by the idea of PS, in this paper, we use gradient based method and golden proportion to improve differential evolution.

2.2. Chaotic Initialization

Logistic model is a typical example of chaotic traversal algorithm and it is widely used for design optimization algorithms. It has the following form:

$$cx_i^{k+1} = u \cdot cx_i^k \cdot \left(1 - cx_i^k\right), i = 1, 2, \cdots, n \tag{1}$$

where cx_i^k is the value of cx_i after k-th chaotic evolution. When it satisfies the criteria : $u = 4$, $cx_i \in [0,1]$ and $cx_i \in [0.25, 0.5, 0.75]$, chaotic phenomena will merge and cx_i traverses between $[0,1]$, as shown in Figure1.

When $x_i \in [a_i, b_i] \neq [0,1]$, we have the following transformation [6]:

$$cx_i = (x_i - a_i)/(b_i - a_i) \tag{2}$$

$$x_i = a_i + cx_i (b_i - a_i) \tag{3}$$

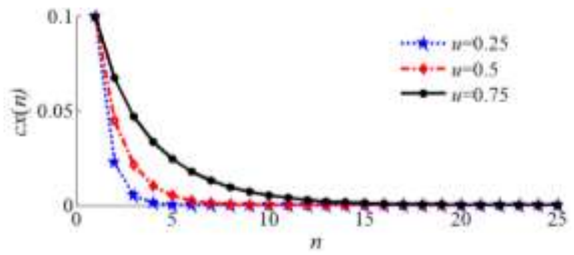

Figure 1. Particle Distribution of Logistic Model

2.3. Improvement on Mutation

Mutation is a key step to prevent premature phenomenon, in this paper we improved the differential mutation step:

$$x_i^{t+1} = x_i^t + F\left(x_{r1}^t - x_i^t + x_{r2}^t - x_{r3}^t\right) \tag{4}$$

The mutation in (t+1)-th generation is based on the individuals surrounding x_i^t . The local mutation operation may keep the population diversity and keep the global evolution direction unchanged. The last component $x_{r2}^t - x_{r3}^t$ in Equation (4) may introduce turbulence to randomization and increase the diversity after mutation.

We consider the direction of population evolution when selecting parameter $r1, r2, r3$. For minimization problem, first generate three different random number $c1 \neq c2 \neq c3 \in Z[0,1]$, then $r1$ is set to the number with smallest target function value and $r3$ is set to the number with largest target function value, as shown in the following steps:

Step 1. $r1 = find\{c1, c2, c3\}$ $s.t.\ min\left\{val\left(x_{c1}^t\right), val\left(x_{c2}^t\right), val\left(x_{c3}^t\right)\right\}$	Step 2. $r2 = find\{c1, c2, c3\}$ $s.t.\ mid\left\{val\left(x_{c1}^t\right), val\left(x_{c2}^t\right), val\left(x_{c3}^t\right)\right\}$	Step3. $r3 = find\{c1, c2, c3\}$ $s.t.\ max\left\{val\left(x_{c1}^t\right), val\left(x_{c2}^t\right), val\left(x_{c3}^t\right)\right\}$

The above ranking method may balance the individual diversity and the global evolution direction. It may also compensate the over randomized mutation and result in an effective searching algorithm.

2.4. Local Search Based on Gradient.

Given $f(\vec{x})$, $\vec{x} = (x_1, x_2, \cdots, x_n)$, the gradient may be represented as [7]:

$$\Delta f(\vec{x}) = \left[\frac{\partial f(\vec{x})}{\partial x_1}, \frac{\partial f(\vec{x})}{\partial x_2}, \cdots, \frac{\partial f(\vec{x})}{\partial x_n}\right]^T \qquad \text{NNNN} \qquad (5)$$

Using the straight displacement in the negative gradient direction in local depth search will improve the algorithm efficiency. The straight searching is based on the golden proportion method, the pseudo code is shown in Figure2.

```
Alg. 1: Pseudo code for gradient based local search
if $rand \le p$
 $x_i^{t+1} = x_i^t + F\left(x_{r1}^t - x_i^t + x_{r2}^t - x_{r3}^t\right)$;
else
 $\vec{t}_2 = \vec{a} + \beta\left(\vec{b} - \vec{a}\right)$; $f_2 = f\left(\vec{t}_2\right)$; $\vec{t}_1 = \vec{a} + \vec{b} - \vec{t}_2$; $f_1 = f\left(\vec{t}_1\right)$;
  while $\left|\vec{t}_1 - \vec{t}_2\right| \ge \varepsilon$
    if $f_1 \le f_2$
      $\vec{b} = \vec{t}_2$; $\vec{t}_2 = \vec{t}_1$; $f_2 = f_1$;
    else
      $\vec{a} = \vec{t}_1$; $\vec{t}_1 = \vec{t}_2$; $f_1 = f_2$;  $\vec{t}_2 = \vec{a} + \beta\left(\vec{b} - \vec{a}\right)$; $f_2 = f\left(\vec{t}_2\right)$;
    end
  end
 $x_i^{t+1} = \left(\vec{t}_1 + \vec{t}_2\right)/2$;
end
```

Figure 2. Pseudo Code for Gradient Based Local Search

In this algorithm, $[\vec{a}, \vec{b}]$ is the initial gradient searching region, and ε is the threshold, empirically set to 0.1. Acceleration along the negative gradient direction makes the evolution more effective. The algorithm is easier to converge and computation complexity is not significantly increased.

2.5. PS-GDE Algorithm Description.

In predatory searching strategy, the global optimal search is fist carried out and the local search is carried out afterwards. Here, we give a simplified strategy, in which we use a default period to trigger the local search. When the evolving population reaches an integral multiple of the default period, the local search is triggered multiple times for the best individuals. Compared with a single search, predatory strategy is applied multiple times in order to make use of the global searching ability and local convergence ability in differential evolution algorithm. This will prevent the algorithm from premature phenomenon. The PS-GDE algorithm steps are listed as follows:

Step 1: Set the PS-GDE population size NP , dimension D, and the termination generation G according to the actual problem. Initialize the search region $[l^0, u^0]$, and set $s = 1$. The default period NC that triggers the local predatory search is set to 20. Sample selection percentage is set to: $pr = 5\%$.

Step 2: Initialize population p^1 with size NP using chaotic method within region $[l^0, u^0]$, and calculate the fitness.

Step 3: If s is an integer multiple of NC, go to Step 4 to perform local search based on gradient, otherwise go to Step 5 to perform global search based on the improved differential evolution algorithm.

Step 4: Carry out global search, perform mutation operation according to Equation (4), perform crossover operation and calculate the fitness value of new individual. Go to Step 6.

Step 5: Perform local search according to the pseudo code in Figure 2.

Step 6: If the fitness value of the current best individual meets the termination criteria or the iteration number reaches the maximum number, stop PS-GDE algorithm and output the optimized solution, otherwise, go to Step 3 and update evolution generation: $s = s + 1$.

The parameter NC and pr in predatory strategy will affect the algorithm performance, we will demonstrate the related experimental study in Section 3.1.

3. Performance Test

Three basic testing functions are selected according to reference [8], including unimodal independence, unimodal non-independence, multimodal independence, and multimodal non-independence:

(1) $f1 = \sum_{i=1}^{30} x_i^2$ (2) $f2 = \dfrac{\sum_{i=1}^{30} x_i^2 - \prod_{i=1}^{30} \cos\left(x_i / \sqrt{i}\right) + 1}{4000}$ (3) $f3 = \sum_{i=1}^{29} \left|100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2\right|$

3.1. Parameter Analysis in Predatory Strategy

Using the above described testing functions, we carry out experimental study on parameters NC and pr. Parameter pr is the percentage of the chosen best individuals among the entire population, and it should not be too large. When parameter NC becomes too big, the algorithm will converge at a slower rate. When parameter NC becomes too small, the computational complexity will increase. Two experiments are carried out in this section.

Experiment 1: Set the period trigger, $NC = 30$, in the predatory search, and set $pr = [0.1, 0.3, 0.5, 0.7, 0.9]$. Use the algorithm termination generation zd and the running time t as the indications of the performance. Experimental results are shown in Table 1.

We can see from Table 1 that parameter pr affects function f3 most significantly. When pr is set to 0.5, it has the worst performance. When pr is set to 0.1, it has the best performance.

Table 1. Enumeration of pr on Algorithm Performance

Mean of 10 running times		pr (NC = 30)				
		0.1	0.3	0.5	0.7	0.9
f1	zd	386	385	386	383	381
	t(s)	5.1	5.4	5.9	5.9	5.9
f2	zd	341	343	344	335	341
	t(s)	5.7	5.8	6.0	6.3	6.5
f3	zd	1594	1803	3346	2104	1621
	t(s)	21.6	27.8	49.8	34.2	23.3

Experiment 2: Set sample selection percentage $pr = 0.1$, and set parameter $NC = [10,30,50,70,90]$. We also choose the algorithm termination generation zd and the running time t as the indications of the performance. The experimental results are shown in Table 2. We can see that when pr equals to 0.1, NC equals to 50, the algorithm reaches the best performance.

3.2. Comparison of Algorithm Performance

We set the parameters of PS-GDE algorithm as follows. Dimension D is set to 30, which is a commonly used setting in evolutionary computing. The population size should be 5 to 10 times of the dimension [4], and we set $NP = 200$.

Table 2. Enumeration of NC on Algorithm Performance

Mean of 10 running times		NC (pr = 0.1)				
		10	30	50	70	90
f1	zd	416	386	383	380	388
	t(s)	5.7	5.1	4.8	4.7	4.8
f2	zd	378	341	334	343	362
	t(s)	6.3	5.7	4.9	5.7	5.7
f3	zd	2214	1594	1573	1614	1626
	t(s)	31.2	21.6	21.1	22.6	23.7

A bigger population size will benefit the diversity but increase the computational complexity and a smaller one will do the opposite. The maximum allowed iteration number is set to 8000. The scale factor F is set according to the experimental study in [4]. F should satisfy: $F \in [0.4, 0.8]$, and in this paper we set: $F = 0.6$. The crossover factor should satisfy: $CR \in [0.3, 0.9]$. According to section 2.1 we set: $pr = 0.1$、 $NC = 50$.

We choose SACPMDE, ASMDE and DERL algorithms for comparison. The parameters setting for SACPMDE and ASMDE can be found in [8]. The parameters setting for DERL can be found in [4]. The fitness accuracy requirement is: VTR=10-6. The simulation results are shown in Table 3, Figure 3, Figure 4 and Figure5. The log value is used for a better demonstration.

Table 3. Comparison of Algorithm Performance

		Optimal	Averaged	Iterati	Variance	Time/s
f1	PS-GDE	7.34×10-7	8.18×10-7	381	3.35×10-18	5.4
	SACPMDE	6.13×10-6	8.85×10-6	297	9.77×10-18	8.6
	ASMDE	6.47×10-6	9.01×10-6	1087	8.21×10-18	9.3
	DERL	7.54×10-6	9.13×10-6	314	4.89×10-13	3.6
f2	PS-GDE	6.76×10-7	7.23×10-7	337	2.17×10-15	4.9
	SACPMDE	6.98×10-6	9.00×10-6	301	6.16×10-13	9.3
	ASMDE	6.85×10-6	9.04×10-6	1159	7.19×10-13	18.8
	DERL	7.50×10-6	1.29×10-2	5813	2.07×10-4	85.3
f3	PS-GDE	4.38×10-6	5.14×10-6	1418	3.12×10-13	20.0
	SACPMDE	7.62×10-6	1.35	4413	1.81×101	51.1
	ASMDE	3.92×10-3	7.31×10-3	8000	9.51×10-6	94.3
	DERL	9.90×10-6	1.41×102	6457	3.42×105	92.9

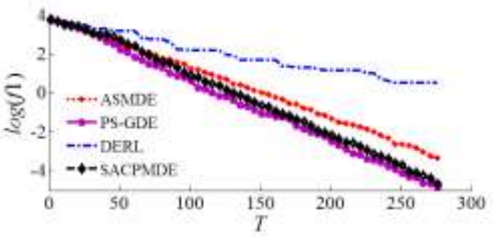

Figure 3. Convergence Curve of Function f1

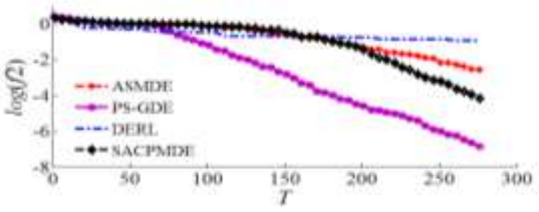

Figure 4. Convergence Curve of Function f2

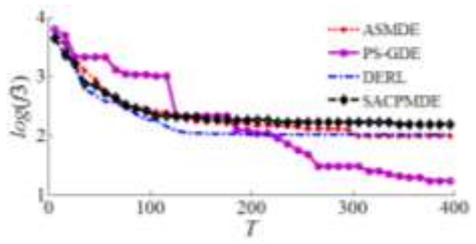

Figure 5. Convergence Curve of Target Function f3

We can see from the above results that PS-GDE algorithm shows a constant improvement on test functions compared with other algorithms. Especially on test function f3, ASMDE, SACPMDE and DERL algorithms all show premature convergence. PS-GDE algorithm has an obvious advantage in keeping the population diversity and preventing algorithm from premature convergence.

The main title (on the first page) should begin 1 3/16 inches (7 picas) from the top edge of the page, centered, and in Times New Roman 14-point, boldface type. Capitalize the first letter of nouns, pronouns, verbs, adjectives, and adverbs; do not capitalize articles, coordinate conjunctions, or prepositions (unless the title begins with such a word). Please initially capitalize only the first word in other titles, including section titles and first, second, and third-order headings (for example, “Titles and headings” — as in these guidelines). Leave two blank lines after the title.

4. Load Allocation Algorithm in Hot Rolling Mill Based on PS-GDE

The load allocation optimization in hot rolling mill is a key process for steel plate production. An optimized solution may help to decrease the energy cost and improve the steel plate quality. The problem can be modeled as a multi-target parameter optimization. In the past references, the traditional method fix the weighted coefficients and convert it into a single target optimization problem [9]. However, in actual industry applications such weighted coefficients are very hard to estimate. To solve this problem, we introduce the weighted coefficients adaptation based on the improved differential evolution algorithm, which may improve the optimization model for load allocation in hot rolling mill.

4.1. Target Function and Constraints.

During the hot rolling process, since the steel plate is usually very thin , and the strip threading speed is slow, there is no constraint on the biting condition. The constraints are given as follows [9]:

$$\begin{cases} 0 \le P_i \le P_{\max} \\ 0 \le I_i \le I_{\max} \\ h_{i+1} < h_i \end{cases} \tag{6}$$

where P is the rolling power, I is the roll torque and h is the thickness of the strip plate export.

The principle in load allocation is that: the front stander is used for balancing load and save energy cost; the back stander takes care of the plate thickness and quality requirements. The target function can be represented in the following form [9].

$$J = min\{z_1, z_2, z_3\}\text{ , } z_1 = (P_1 - KP_2)^2\text{ , } z_2 = (P_2 - P_3)^2\text{ , } z_3 = \sum_{i=4}^{7}(CR_i/h_i - CR_7/h_7 \pm \Delta)^2 \tag{7}$$

where K_1 and K_2 are the weighted coefficients, Δ is the adjustment buffer, P_i is the actual roll power of the rolling mill frame, CR_i is the actual convexity of the plate in hot rolling mill, and h_i is the thickness of the plate.

4.2. Optimization Steps

The thickness allocation plan in hot rolling mill based on PS-GDE algorithm is listed below:

Step 1: Read the parameters of the equipment, rolled piece, initial state and steel strip requirements.

Step 2: Calculate the basic target thickness $\overline{h_i}$ of each stander in the hot rolling mill using the traditional empirical method.

Step 3: Calculate the parameters of each stander.

Step 4:Search for the thickness value h_i that gives the optimized target value J in Equation (7) using PS-GDE algorithm.

Step 5: If the termination criteria is met, continue to next step, otherwise go to Step 3.

Step 6:If the constraints in Eq. (6) is satisfied, continue to the next step, otherwise go to Step 3.

Step 7: Output the optimized load allocation value.

4.3. Simulation Study

According to reference [1], we use steel type Q235 to perform simulation with the related parameters. The width of plate is set to $B_c = 1535\text{mm}$, the initial thickness $H_0 = 36.7\text{mm}$, the final thickness $h_7 = 5.7\text{mm}$, the actual temperature of coarse roll export is $T_{RC} = 1340\text{K}$, the final requirements of convexity is set to, $R_7 = 0.016\text{mm}$, the population size $m = 50$,crossover probability factor $CR = 0.6$,and the dimension is set to $n = 7$. The simulation results are shown through Table 4 to Table 6.

Table 4. Roll Power Allocation Comparison (kn)

Method	P1	P2	P3	P4	P5	P6	P7
Empirical	19.42	17.64	21.91	16.81	11.97	11.64	8.58
PS-GDE	17.81	19.8	19.8	13.58	12.82	10.81	9.42

Table 5. Relative Convexity of Hot Rolling Mill (10-4)

Method	LR1	LR2	LR3	LR4	LR5	LR6	LR7
Empirical	11	12	21	21	18	21	18
PS-GDE	8	14	19	17	17	17	17

Table 6. Thickness Allocation of Each Stander Export (mm)

Method	h1	h2	h3	h4	h5	h6	h7
Empirical	25.49	18.53	12.65	9.54	7.84	6.52	5.70
PS-GDE	26.24	18.20	12.75	10.03	8.04	6.68	5.70

According to Table 4, Table 5, and Table 6, we can see that the load allocation optimization based on PS-GDE is better than the empirical methods. Generally, the roll power P1 of the first stander is set to 0.9 times of the roll power P2 of the second stander. The roll powers of the second and the third stander are approximately set to the same value. The roll power drops from the third stander to the last stander. The last four stander should keep the same relative convexity as the plate shape is an important measure for steel quality. Experimental results in Table 5 show that the requirements on relative convexity can be satisfied using PS-GDE algorithm in load allocation. The running time of load allocation optimization using PS-GDE is smaller than 4s, and the searching iteration is smaller than 70. From the view of algorithm efficiency, the convergence is significantly better than Immune genetic algorithm reported in [9]. Future online real-time application is therefore possible.

5. Conclusions.

We propose an improved differential evolution algorithm based on predatory strategy and gradient acceleration, with a relatively low enhancement in computational complexity.

Using repeated predatory strategy we may avoid the algorithm from premature and keep the fast convergence at the meantime. The simulation results on test functions shown he effectiveness of the proposed algorithm. Finally we applied it to the optimization problem in load allocation in hot rolling mill and the simulation results show a promising performance.

References

[1] F. Yao, W. Yang, M. Zhang and Z. Li, "Improved space-adaptive-based differential evolution algorithm", Control Theory & Applications, vol. 27, no. 1, **(2010)**, pp.32-35.
[2] R. Liu, B. Wang and J. Zheng, "Two Section Crossover Differential Evolution Algorithm and Its Application", Journal of System Simulation, vol. 25, no. 7, **(2013)**, pp.1549-1553.
[3] V. Feoktistov and S. Janaqi, "Generalization of the strategies in differential evolution", Proc of the 18th Int Parallel and Distributed Processing Symposium, Santa Fe, **(2004)**, pp.165-170.
[4] P. Kaelo and M. Ali, "A numerical study of some modified differential evolutional algorithms", European Journal of Operational Research, vol. 169, **(2006)**, pp.1176-1184.
[5] Y. P. Wang and C. Y. Dang, "An evolutionary algorithm for global optimization based on level-set evolution and Latin squares", IEEE Transactions on Evolutionary Computation, vol. 11, no. 5, **(2007)**, pp.579 - 595.
[6] A. Linhares, "Preyng on optima: A Predatory search strategy for combinatorial problems", Proc. IEEE International Conference on System, Man, and Cybernetics, vol. 78, **(1998)**, pp.167-173.
[7] D. Wang, J. Wang and H. Wang, "Intelligent Optimization Methods", Beijing: Higher Education Press, **(2007)**.
[8] L. Wu, "Study on differential evolution algorithm and its application", Master's Degree Thesis, Hunan: Hunan University, **(2007)**, pp.42-66.
[9] Y. Wang, J. Liu and Y. Sun, "Immune Genetic Algorithms Based Scheduling Optimization for Finisher", Journal of University of Science & Technology Beijing, vol. 24, no. 3, **(2002)**, pp.339.
[10] R. Cheng, M. Gen and S. S. Oren, "An adaptive Hyperplane approach for multiple object optimization problems with complex constraints", Ashikaga City, Tochigi, Japan: Ashikaga Institute of Technology, **(1998)**.

Authors

Feng Guiliang, he is a lecturer of department of information science and engineering, Hebei North University, He is good at the field of software engineering and multimedia development.

Cao Ning, he is a lecturer of department of information science and engineering,Hebei North University,He is good at the field of software engineering and multimedia development.

Li Zhonghua, her primary research interests are Data Mining, Big Data Computing and Algorithms along with their applications in several domains..