International Journal of Hybrid Information Technology
Vol.8, No.12 (2015), pp. 143-152
http://dx.doi.org/10.14257/ijhit.2015.8.12.09

The Combination of Extension of Ant Colony Algorithm and Other Intelligent Algorithms

Ma Li[1], Li Qianting[1], Ma Meiqiong[1], Meng Jun[2] and Bai Jiyun[2]

[1]*Engineering College, Northeast Agricultural University, China*
[2]*College of Science, Northeast Agricultural University, Ha'erbin 150030, China*
drizzlem19@163.com

Abstract

The extension of ant colony algorithm was proposed by Dorigo, the founder of ant colony algorithm, which is the latest ant colony algorithm for solving a continuous space optimization problem. Considering the blindness of man-made choice of initial solution and initial parameters of the algorithm, and according to the algorithm converging slowly and easily falling into local optimum, this paper has provided improvement strategy for this optimization. It has introduced quantum computing and genetic algorithm, chaos optimization to carry out combination and comparison, and it has carried out improvement on the weight internally solved by memory in the algorithm. The effectiveness of various combined algorithms was determined through the optimization of numerous multi-dimensional continuous functions.

Keywords: *Extension of ant colony optimization, Optimization of continuous space, Genetic algorithm, chaos*

1. Introduction

Ant colony algorithm (ACO [1, 13]) is a bionic optimization algorithm that imitates the foraging behavior of ants, the major advantages of ACO are positive feedback of message, distributed computing, combining with other algorithms easily, *etc*. And now, It has been successfully applied to the field of discrete space optimization [2-3]. But for processing continuous space optimization problems, there are two main ways: Firstly, making the continuous space discretized, and then transform the continuous problem into discrete problem [16]; the second is combination with evolutionary algorithms [4], but the convergence was slow. Whether the first approach could be adapted to high-dimensional problems are yet inconclusive, in addition, it carry out a lot of changes on the basic structure of the ant colony optimization algorithm, which is not conducive to improving the algorithm.

In 2008, on the basis of ACO, Dorigo, *et al.*, proposed extension of ant colony algorithm (ACOR) [5-6]. ACOR was simple and easy because it was designed still in accordance with the ACO algorithm framework. It was applied to solve the classic continuous function and compared with existing algorithms to achieve higher accuracy. However, due to ACOR still using a random method to obtain the initial solutions, the initial pheromones accumulation were too slow; At the same time, the solution was influenced largely by the algorithm parameters , especially in solving the unknown optimization problems, there was a problem of algorithm parameters re-selection; The algorithm constructed pheromone only through the target function value size of the solution of memory in one search, so the direction of solution was not clear enough, convergence speed was slow , and was easy to fall into local optimum.

With the development of computing technology, there were a lot of intelligent algorithms which were used to solve the complex optimization problems, such as genetic algorithms, particle swarm optimization [7-12, 14-15], each algorithm has irreplaceable

ISSN: 1738-9968 IJHIT

advantages, the combination between individual algorithm and extensive ant colony algorithm can get more effective optimization strategy. This paper studied the improvement on extension of ant colony algorithm from the algorithm combination of view, proposed two new fusion algorithm based on literature [7-12, 14-15], and comprehensively improved the convergence speed of ACO_R algorithm to avoid falling into local optimal. Meanwhile, it used various improvement methods to do simulation for four classic multi-peak and nonlinear test functions in literature [5], and compared with extensive ant colony algorithm to determine the advantages of various methods.

2. Extension of Ant Colony Algorithm

2.1. The Process of Extension of Ant Colony Optimization

The literature [5] introduced Gauss kernel probability density function and solution memory in probability selection rule of pheromone to extend ACO algorithm [1, 13], so it could be applied to the continuous space, then the ACO_R algorithm was obtained

Gaussian kernel $G^i(x)$ is defined to the weight sum of some Gaussian functions $g_l^i(x)$ that was formula (1).

$$G^i(x) = \sum_{l=1}^{k} \omega_l g_l^i(x) = \sum_{l=1}^{k} \omega_l \frac{1}{\sigma_l^i \sqrt{2\pi}} e^{-\frac{(x-\mu_l^i)^2}{2\sigma_l^{i2}}} \tag{1}$$

Gaussian kernel $G^i(x)$ contains three parameter vectors: ω is a single Gaussian weight vector, μ^i is the mean vector, σ^i is the standard deviation of the vector. All dimensions of these vectors are equal to k , the number of Gaussian functions, which are the composition of the Gaussian kernel.

When the basic ant colony algorithm was used for the continuous optimization, choice of each ant was no longer confined to a finite set. Each solution vector s_l, the objective function value $f(s_l)$ and weight of a n-dimensional problem were stored in one solution memory T. Therefore, the ith variable of the lth solution was s_l^i, Which structure was illustrated in Figure 1.

s_1	s_1^1	·	s_1^i	·	s_1^n	$f(s_1)$	ω_1
	·	·	·	·	·	·	·
s_l	s_l^1	·	s_l^i	·	s_l^n	$f(s_l)$	ω_l
	·	·	·	·	·	·	·
s_k	s_k^1		s_k^i		s_k^n	$f(s_k)$	ω_k
	G^1		G^i		G^n		

Figure 1. Structure of Solution Memory of ACO_R

On the Figure 1, the feasible solutions was ordered according to their function values (or fitness) $f(s_l)$. For example, the problem of finding the minimum value was sorted as $f(s_1) \le f(s_2) \le \ldots \le f(s_k)$. Each function value had a corresponding weight ω , and

the order was $\omega_1 \geq \omega_2 \geq \ldots \geq \omega_k$. Structuring probability density function Gi only required the i-dimensional coordinates of all k(k=K)solution. Gauss kernel function was composed by the k independent Gauss functions. For each G^i , all solutions value of i-variable in solution memory were changed into the elements of vector μ^i : $\mu^i = \{\mu_1^i, \ldots, \mu_k^i\} = \{s_1^i, \ldots, s_k^i\}$. the formula(2) was used to Calculate the weight ω_l of the solution vector s_l :

$$\omega_l = \frac{1}{qk\sqrt{2\pi}} e^{-\frac{(l-1)^2}{2q^2k^2}} \tag{2}$$

In which qk is the standard deviation, q is a parameter of the algorithm.

The Process of ACO_R algorithm mainly included three steps: solution memory initialization, construction of feasible solutions through the Gauss kernel probability density function and the pheromone updating.

(1) Solution Memory Initialization

Assumed that there were m ants in the ant colony, the length of solution memory T is K, the variable of continuous optimization problem was n-dimensional, solutions of memory T was initialized to K-dimensional solution vector randomly, and the length of each solution vector was n. corresponding objective function values could be calculated according to the K solution vector, and the weights of each solution vector could be calculated by the formula (2).

(2) Constructing feasible solution by sampling from Probability Density Function of the Gaussian kernel

Sampling process involved two steps. The first step was select one Gaussian Function from the compositions of the Gaussian Kernel Function, and according to the follows formula (3) to calculate selection probability.

$$p_l = \frac{\omega_l}{\sum_{r=1}^{k} \omega_r} \tag{3}$$

Second step was to sample for the Gaussian Function selected in the first step. This sampling procedure could be completed by using a parametric normal random number generator to generate a random number. The formula as follows,

$$\sigma_l^i = \xi \sum_{e=1}^{k} \frac{\left| s_e^i - s_l^i \right|}{k-1} \tag{4}$$

The parameter ξ for all variables was the same regardless of whether variables' dimensions and $\xi > 0$.

(3) The pheromone updating

A temporary solution vector could be formed of the solution vectors from above *m* ants sampling and original solution in memory T, and it was ordered by the objective function. In order to keep the length of *K*, the former *K* solution vectors were added in memory T. which ensures only the optimal solution could be stored in solution memory T, then the solution in memory would be able to guide the ants search better

2.1. Algorithm Analysis

Literature [5] had verified the effectiveness of the ACOR algorithm by experiment. However, a large number of instances showed that ACOR algorithm also had some shortcomings:

(1) For continuous space optimization, feasible solution shows density, while only the initial solutions in ACO_R were determined by random method, so the quality of individual could not be guaranteed. In addition, the initial information was blind, and accumulation of information is slowly.

(2) In the ACO_R, constructing and updating the solution memory only depended on the size of the objective function values. The weight of each solution was only determined by the formula(2).Therefore, the selection probability of solution (equation(3)) was only related to the objective function value quality, without considering the contribution extent of the current solution to explore the optimal value, so search direction was not clear enough, the speed of convergence was slowly

(3) When the search cycles rose to a certain number, it was easily to be precocious phenomenon and lead to a local optimum optimal solution due to the concentration of pheromone produced by local optimal solution is too high. The reason was the ACO_R uses solution memory as storage of pheromone to construct next solution.

3. Improvement on ACO_R Algorithm Integrated with Intelligent Algorithm

In order to overcome these shortcomings of ACO_R algorithm, this paper introduced two optimization algorithms to integrate with ACO_R algorithm, making use of the advantages of each algorithm fully to accelerate the convergence speed of ACO_R algorithm, while avoiding of falling into local optima.

3.1. Improvement on Extension of Ant Colony Algorithm Integrated with Genetic Algorithm

3.1.1. The Idea of Algorithm

The main features of Genetic Algorithm (GA) are the search strategy in the population and the exchange of information between individuals and groups. Genetic Algorithm has fast and stochastic global search capability, but it is powerless for the use of the system feedback. When solving a certain range, it tends to do a lot of redundant iterations, which lead to low efficiency when the exact solution to be demanded [7-10, 15].

This paper integrated Genetic Algorithm and ACOR, using genetic algorithm to generate the initial solution and the initial pheromone, and using ACO_R algorithm for exact solution. While after each iteration, it carried out mutation to avoid falling into local optima. In addition, the paper also had following improvements: using the change rate of solution to reflect contribution extent of current solution for finding the optimal solution. Assuming,

$$\nabla f_{lj} = \frac{\nabla f_j(x_j^l)}{\nabla f_j \max}$$

Weight calculation of solution in the solution storage is as follows:

$$\omega_l' = \omega_l - \sum_{j=1}^{n} \nabla f_{lj} \quad (5)$$

In which, the ω_l was calculated by formula (2), the $\nabla f_j(x_j^l)$ was the gradient of evaluation function $f(x)$ in the point of x_j^l, $\nabla f_{l\max}$ was the maximum of gradient of evaluation function $f(x)$ for the *j*th variable, $j = 1,2,\cdots,n$. If it needed minimum of objective function, ∇f_{lj} would be negative, and the absolute value be larger, then it Indicated that the objective function was to be better, the corresponding weights should be larger, whereas weights should be smaller. From formula (5), the first part was equivalent of the 'use' of solution that has been used, the second part was equivalent of 'explore' for unused solution, the size determined the ability of ants finding new good solution.

3.1.2. Process of Improvement on Genetic ACO_R

The main process of Genetic algorithm-extension of ant colony optimization (GAACOR) was as follows:

(1) Producing a more optimal solution by using genetic algorithm, it was constructed the initial solution of the ACO_R algorithm.

Set the iteration times of GA were $t(t_b < t < t_c)$, t_b and t_c respectively represented the minimum and maximum genetic iterations. If three consecutive generation evolution rate R meet the request $0 < R < R_{\min}$ in the process of calculating each iteration evolution rate $R = \bar{f}_i(s) - \bar{f}_{i+1}(s)$, then genetic algorithm was terminated and turned into the ant colony algorithm. Among them $\bar{f}_i(s)$ represents the average value fitness function of GA after the *i*th times iteration, $R_{\min}$ was the given minimum threshold. Matrix S_1 was formed by K/2 individual of the highest fitness function by choose from the last generation population of GA. In order to prevent falling into local optimum, it randomly formed the matrix S_2 composed by K/2 individuals to form a $K \times N$ -dimensional new matrix S together with matrix S_1, in which N was dimensions of decision variables.

(2) $t \leftarrow 0$ Based on matrix S as initial solution, to construct the initial solution memory;

(3) used formula (5) and formula (2) to calculate the weight of each solution in memory, used formula (3)to select Gaussian kernel probability density function and determine the new solution;

(4) According to the mutation probability judged whether variation was carried out. If there were variations, then new solution was gotten, otherwise rejected. According to the process to get a new memory;

(5) $t \leftarrow t + 1$, Go to step 3 to continue to iteration, If $t = \max(t)$, output the global optimal solution.

3.2. Improvement on ACO_R Integrated with Chaos Optimization

3.2.1. Algorithm Ideal

Chaotic motion has some characteristics like randomness, ergodic, and sensitive to initial value, it can be not repeated through all states in a certain range according to their own laws [12]. Chaos optimization, which is a relatively new optimization algorithm using these properties of chaotic motion to do optimization search. Logistic chaotic sequence is a general sequence that is used by this algorithm commonly. It can be described by the following formula:

$z_{j+1} = \mu z_j(1 - z_j), \quad j = 0,1,2,\cdots,$

In which μ is control parameter, when $\mu = 4$, $0 \le z_0 \le 1$, Logistic sequence is in the chaotic station.

According to the shortcomings of ACO_R, in the base of literature [9], this paper constructed the initial solution by introduction of chaos optimization, designing search radius of variable scale, using Chaos search, accelerating convergence speed, and avoiding local optima. According to the importance of solutions in memory to improve weight value of each solution so that increase directionality and quickly access to optimal solution.

3.2.2. The Process of Improved Chaos-extended ant Colony Algorithm

Improved chaos-extended ant colony algorithm ($CACO_R$) was achieved by two nested searching. First, ACO_R to do global search with ACO_R, the next, carry out mutative scale Chaos search near to optimal solution of per iteration and achieve local fine search.

(1) Search 1: process of global search

1) Constructing initial solution of Chaotic;

$t \leftarrow 1$,*m* initial populations that was n-dimensional variables generated by Logistic mapping in [0,1] ,which was denoted as $z_{ij}(t)$, $i = 1,2,\cdots,n$, $j = 1,2,\cdots,m$, $\mu = 4$, $z_{i1}(1) = 0.9$.

2) The independent variable produced by the first step were mapped to optimize space, $x_{ij}(t) = a_i + z_{ij}(t)(b_i - a_i)$, $i = 1,2,\cdots,n$, $j = 1,2,\cdots,m$.

3) Calculating the fitness of each individual to construct memory of solutions of figure (1);

4) to do the chaos search by using this improved mutative scale chaos fine search in the near global optimal ant （ $x_{ik}^*(t)$, $i = 1,2,\cdots,n$ ） ,if it could find the better than the global optimal ants, set it to the global optimal ants;

5) Using formula (5) and formula (2) to calculate the weight of each solution in memory and using formula (3) to select Gaussian kernel probability density function, determine the data solution;

6) Repeating step3~step5 until termination condition was met.

(2)Search 2: The process of fining mutative scale chaos search

1) to carry out initialization by using Logistic map to produce *p* initial solutions $z_{ij}(t)$ within [0, 1] , in which $i = 1,2,\cdots,n$, $j = 1,2,\cdots,p$.

2) to compute fine search radius, do a scale transform for the independent variables and map to optimization space$[x^*_{ik}(t) \pm r_{ti}]$。

The purpose of using mutative scale chaos search was to make the early search range of algorithm larger to avoid early falling into local optimum, and let the search range was smaller in the later in order to improve search precision. According to the requirement, this paper used Gaussian probability density function to determine the search radius, let,

$$r_t = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(t-1)^2}{2\sigma^2}}$$

In which, *t* is the number of iterations. According to the principle of the Gaussian kernel probability density function 3σ , let $\sigma = \max(t)$, therefore, search radius of the *i*-dimensional variable of the *i*-times iteration is as follows:

$$r_{ti} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(t-1)^2}{2\sigma^2}} (\frac{b_i - a_i}{2}) \quad (6)$$

3) Calculating the function values of each individual and updating the current optimal function value;

4. Solving the Multi-dimensional Continuous Function by Improved ACO_R

For Contrast, the two test functions of literature [6] were applied in this paper to make a comparison analysis between the improved algorithms of ACO_R ($GAACO_R$ and $CACO_R$)and ACO_R to determine the effectiveness of the improved algorithms.

4.1. Test Function

(1) Shaffer's F6 function

$$f_1(x, y) = 0.5 - \frac{\sin^2\sqrt{x^2 + y^2} - 0.5}{(1 + 0.001(x^2 + y^2))^2}$$

In which $x, y \in [-100,100]$.

This function has infinite numbers of local maximum points, only one point (0, 0) is the global maximum point, the maximum is 1.

(2) Rosenbrock function

$$f_2 = \sum_{i=1}^{N-1} [100\ (x_i^2 - x_{i+1})^2 + (x_i - 1)^2]$$

N=5, $-10 \le x_i \le 10$, When x_i=1 ,it can obtain the global minimum value 0 and it is success when f_2=0.0001.

4.2. Parameter Selection

Referencing literature [5], initial parameter selection of $GAACO_R$ was as follow: genetic parts, population size M = 35, the largest number of iterations was 100, the minimum number of iterations was 50, minimum evolution rate R=10, crossover probability $p_c = 0.90$ and mutation probability $p_m = 0.10 - [1:1:M] \cdot (0.01)/M$; Extended ant colony parts, ants number m=70 , solution storage capacity $K = 45$, parameter $\xi = 1$, $q = 0.0001$.

Initial parameters of $CACO_R$, ants number m=70, capacity of solution storage $K = 45$, and parameter $\xi = 1$, $q = 0.0001$.

4.3. Simulation Result

10 times optimization was done with each algorithm of $GAACO_R$, $CACO_R$, and ACO_R on the function f_1 and f_2, the simulation results were shown in Tables 1 to 2, Figure 2 to Figure 3.

Table 1. The Optimization Results of Extremum Problems of $f_1(x, y)$ (10 Times Experiment)

Algorithms	$f_1(x, y)$				
	Optimum	Average	The first optimal number of iterations	Numbers of success	corresponding curve
$GAACO_R$	1	1	180	10	1
$CACO_R$	1	1	120	10	2
ACO_R	1	0.9925	261	2	3

Table 2. The Optimization Results of Extremum Problems of f_2 (10 Times Experiment)

Algorithms	f_2				
	Optimum	Average	The first optimal number of iterations	Numbers of success	corresponding curve
$GAACO_R$	0.0001	0.0011	210	7	1
$CACO_R$	0.0001	0.0002	176	9	2
ACO_R	0.0001	0.0761	407	1	3

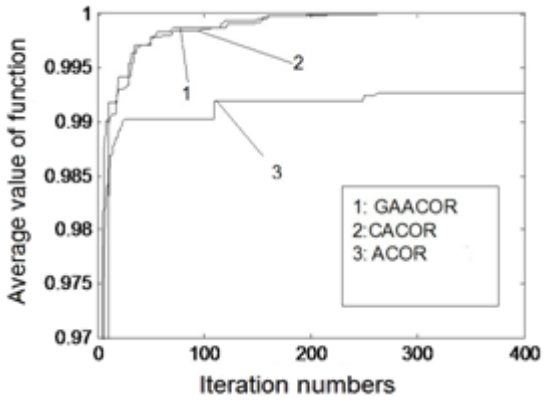

Figure 2. $f_1(x, y)$ Function Optimization Results

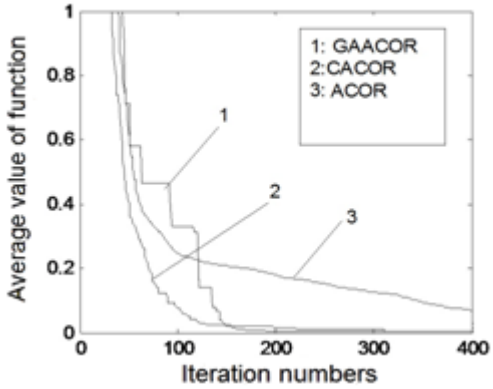

Figure 3. f_2 Function Optimization Results

It can be seen from Table 1 to 2, Figures. 2 to 3 that the three improved algorithms of this paper not only have a faster search speed, but also have a strong global search capability on the continuous space optimization problems. The two

functions that were used to simulate have more local extremums and complex forms. ACO_R algorithm was often difficult to jump out local extremum, $GAACO_R$ and $CACO_R$ algorithm almost jumped out of local minimum entirely.

5. Conclusions

This paper analyzed advantages and weaknesses of ACO_R, so far as to analyze several common intelligent algorithms. It proposed combination algorithm of ACO_R and other intelligent algorithms, which overcame the shortcomings of ACO_R, and improved searching efficiency of it. Meanwhile the study was a comprehensive attempt on the integration of ACO_R and other intelligent algorithms. In the end, it got two new fusion algorithms, through applied to optimize several representative multidimensional continuous functions to determine the advantages of them.

Acknowledgments

The research is funded by Heilongjiang Youth Fund (QC2011C045) , Doctoral Fund of Northeast agricultural University.

References

[1] M. Dorigo, V. Maniezzo and A. Colorni, "Ant System: Optimization by a colony of cooperating agents", IEEE Trans Syst Man Cybernetics, vol. 26, no. 1, **(1996)**, pp. 29-41.
[2] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem", IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, **(1997)**, pp. 53-66.
[3] A. Colorni, M. Dorigo and V. Manizezzo, "Ant colony system for job-shop scheduling", Belgian Journal of Operations Research Statistics and Computer Science, vol. 34, no. 1, **(1994)**, pp. 39-53.
[4] V. K. Jayaraman, B. D. Kulkarni, K. Sachin, *et al.*, "Ant colony framework for optimal design and scheduling of batch plants", Computer and Chemical Engineering, vol. 24, no. 8, **(2000)**, pp. 1901-1912.
[5] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains", European Journal of Operational Research, vol. 185, no. 3, **(2008)**, pp. 1155~1173.
[6] L. Shiyong and B. Jiyun, "Extended quantum ant colony algorithm for continuous function optimization", Journal of Harbin Engineering University, vol. 33, no. 1, **(2012)**, pp. 80-84.
[7] H. Ming, W. Cong and L. Xu, "Improved Gene Volume Control Hybrid Ant Colony Genetic Algorithm for Traveling Salesman Problem", Journal Of Dalian Jiaotong University, vol. 32, no. 2, **(2011)**, pp. 86-88.
[8] X. J. Rong, L. Yun and L. H. Tao, "Hybrid genetic ant colony algorithm for traveling salesman problem", Computer Applications, vol. 28, no. 8, **(2008)**, pp. 2084-2087.
[9] W. Bin-Xiao, H. Yan-Quan, S. Ting-Zhen and D. Jia-Du, "Load model parameters identification based on chaos ant colony optimization", Power System Protection and Control, vol. 39, no. 14, **(2011)**, pp. 47-51.
[10] L. S. Yong and W. Qing, "Extensive Particle Swarm Ant Colony Algorithm for Continuous Space Optimization", Journal of Test and Measurement Technology, vol. 23, no. 4, **(2009)**, pp. 319-325.
[11] P. S. Shelokar, P. Siarry, V. K. Jayaraman and B. D. Kulkarni, "Particle swarm and ant colony algorithms hybridized forimproved continuous optimization", Appl Math Comput., vol. 188, **(2007)**, pp. 129-42.
[12] Y. Xuecai and Z. Tianwen, "Multiple colony ant algorithm based on particle swarm optimization", Journal of Harbin Institute of Technology, vol. 42, no. 5, **(2010)**, pp. 766-769.
[13] M. Dorigo and T. Stützle, "Ant Colony Optimization", MIT Press, Cambridge, **(2004)**.
[14] L. S. Yong, "Nonlinear Science and Complexity Science", Harbin Harbin Institute of Technology Press, **(2006)**.
[15] S. W. Mahfoud and D. E. Goldgerg, "A genetic algorithm for parallel simulated annealing", Parallel Problem Solving from Nature, North Holland, vol. 2, **(1992)**, pp. 301-310.
[16] W. Lei and W. Qidi, "Ant system algorithm for optimization in continuous space", Proceedings of the IEEE Inter, Conference on Control Applications. Mexico City, Mexico: IEEE Press, **(2001)**, pp. 395-400.

