J Inf Process Syst, Vol.16, No.5, pp.1169~1182, October 2020
https://doi.org/10.3745/JIPS.01.0059

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

A Hybrid Genetic Ant Colony Optimization Algorithm with an Embedded Cloud Model for Continuous Optimization

Peng Wang*, Jiyun Bai*, and Jun Meng*

Abstract
The ant colony optimization (ACO) algorithm is a classical metaheuristic optimization algorithm. However, the conventional ACO was liable to trap in the local minimum and has an inherent slow rate of convergence. In this work, we propose a novel combinatorial ACO algorithm (CG-ACO) to alleviate these limitations. The genetic algorithm and the cloud model were embedded into the ACO to find better initial solutions and the optimal parameters. In the experiment section, we compared CG-ACO with the state-of-the-art methods and discussed the parameter stability of CG-ACO. The experiment results showed that the CG-ACO achieved better performance than ACO_R, simple genetic algorithm (SGA), CQPSO and CAFSA and was more likely to reach the global optimal solution.

Keywords
Ant Colony Algorithm, Cloud Model, Genetic Algorithm

1. Introduction

Ant colony optimization (ACO) algorithm is a metaheuristic optimization algorithm that is inspired by the foraging behavior of real ants. When ants walking from or to a food source it deposits the pheromone on the ground. Other ants can smell the pheromone and follow the path with a greater concentration of pheromone to find the food source. ACO algorithm borrows a similar idea, when the artificial ants find better solutions they update the pheromone to increase the probability of the search by subsequent ants in the promising regions of the search space. ACO algorithm was first introduced by Italian scholar Dorigo [1] to solve the combinatorial optimization problems in his Ph.D. thesis and has been successfully applied to many discrete optimization tasks [2,3].

In respect of updating the pheromone, Guntsch and Middendorf [4] proposed a population-based ACO (PB-ACO). They keep track of all good solutions up to a certain age in a solution archive to update the pheromone. Instead of using pheromone evaporation, the pheromone associated with the oldest solutions is removed by performing a negative update on the pheromone table.

Based on PB-ACO and Gaussian probability density function (PDF), Socha and Dorigo [5] proposed a new extension of ACO algorithm (ACO_R) in 2008. They reported a mixed Gaussian PDF named

Manuscript received August 31, 2018; first revision March 12, 2019; second revision July 10, 2019; accepted March 12, 2020.
Corresponding Author: Jiyun Bai (baijiyun79@neau.edu.cn)
* College of Arts and Sciences, Northeast Agricultural University, Harbin, China (pengwang@neau.edu.cn, baijiyun79@neau.edu.cn, 1135044376@qq.com)

Gaussian kernel PDF and a rank-based solution archive. ACO_R obtains higher precision compared with the existing metaheuristic algorithms and has been applied in many aspects such as the construction of energy-efficient networks, design of high-rise building wiring, flood forecasting and so on [6-8]. However, ACO_R also has some limitations, i.e., the random initial solutions may mislead the algorithms into converging to a local optimal and the convergence speed is slow, moreover, the parameters in ACO_R are usually determined by experience.

In this work, to overcome these limitations we present a hybrid genetic ACO algorithm with an embedded cloud model for continuous optimization. The genetic algorithm was employed to find better initial solutions and the cloud model was employed to set these ACO parameters adaptively.

This paper is organized as follows: Section 2 discusses the recent improvements and applications of the ant colony algorithm. Section 3 discusses the principle of ACO_R and gives the limitation analysis. In Section 4, we propose the genetic ACO with an embedded cloud model (CG-ACO). In Section 5, we present experiments and analysis of the performance. Section 6 concludes this paper.

2. Related Work

The improvement on the ant colony algorithm can be divided in to two categories, i.e., method improvement and method fusion.

2.1 Method Improvement

After ACO has been proposed a lot of enhanced versions have been reported, Gambardella and Dorigo [9] introduced a local pheromone update rule to help ACO escape from local minimum. Taillard [10] used a shared memory and a Queen to enable the ant to cooperate with each other and increase the convergence speed. Stutzle and Hoos [11] used bounds to prevent a premature stagnation and proposed a MAX-MIN ant system. Guntsch and Middendorf [4] proposed a PB-ACO which uses a solution archive to track all good solutions instead of pheromone evaporation.

Among these improvements, extending the ACO’s capability to tackle multiobjective problems and continuous problems are two major enhancements. For multiobjective optimization, if these objectives can be ordered with importance weight, one can combine them into one objective using a weighted sum. Doerner et al. [12,13] used two ant colonies of variable sizes with different priority rule over the objectives. If these objectives conflict with each other, one has to find Pareto-optimal solutions. Iredi et al. [14] used multiple ant colonies with multiple pheromone trail matrices for different objectives. Guntsch and Middendorf [4,15] used an external population to extend the PB-ACO to a multiobjective version.

Extending ACOs algorithm to continuous domain is another important enhancement. One way to accomplish this goal is to discretize the search space. Hu et al. [16,17] sampled the search space into keys and proposed the SamACO. The other way is shifting the discrete PDF to a continuous one (Gaussian PDF in most cases). Pourtakdoust and Nobahari [18] designed a continuous pheromone model and a new pheromone update mechanism. Socha and Dorigo [5] used a mixed Gaussian PDF and a ranked solution archive and proposed an ACO_R. De Franca et al. [19] used a multivariate Gaussian PDF instead of Gaussian PDF to increase the independence of each dimension of the search space. Liao et al. [20] further presented an ACOMV which includes a continuous optimization mechanism, a continuous relaxation mechanism and a categorical optimization mechanism to extend ACO's capability to tackle mixed-variable.

2.2 Method Fusion

Method fusion is another important branch of ACO improvement. Ant-Q may be the first hybrid ant colony algorithm to solve the Traveling Salesman Problem (TSP) problem proposed by Gambardella and Dorigo [21]. They embedded the ant colony algorithm with Q-learning to describe if the move to some city is useful. Mahi et al. [22] combined the particle swarm optimization (PSO) and 3-opt algorithm with ant colony algorithm to solve the TSP. They used PSO to optimize the parameters of ant colony algorithm and used 3-opt algorithm to improve city selection operations. Nemati et al. [23] combined a genetic algorithm with ant colony algorithms to search in parallel for a better solution. Alsaeedan et al. [24] combined a genetic algorithm with MAX-MIN ant system and used genetic algorithm to optimize the parameters of ant colony algorithm. Goel and Maini [25] used a similar idea as [22] they combined firefly algorithm with ant colony system, and used firefly algorithm to search for the unexplored solution space. Karakonstantis and Vlachos [26] embedded the pattern search into ACO_R and proposed a PSACO to solve emission and economic dispatch problems.

To sum up, the improvements generally aim to alleviate the limitations of ACO algorithm, such as slow convergence speed, the probability of falling into the local minimum, lack of support in multiobjective optimization or continuous domain optimization. The improvements include modifying the searching framework, introducing new rules to ACO and borrowing good properties from another algorithm. However, except for the algorithm in [26], most of the hybrid ant colony algorithms were based on the traditional discrete ant colony algorithm which aims at various combinatorial optimization problems like the TSP. In this work, we proposed a novel hybrid ant colony algorithm based on the continuous ant colony algorithm.

3. Ant Colony Algorithm for Continuous Optimization

3.1 The ACO_R Algorithm

Typically, the optimization problem is to find the minimum value of function $f(\boldsymbol{x})$ in a search space **S** subject to the constraints Ω. ACO_R algorithm generates new solutions s_l from **S** by sampling Gaussian kernel PDF constantly and stores these solutions in a solution archive. It uses two strategies, which have an effect similar to pheromone deposition and pheromone evaporation to find a better solution.

The solution archive is constructed as Table 1, where s_{il} is the i-th variable of the l-th solution and K is the size of the archive T.

Table 1. The archive of solutions kept by ACO_R

s_1	s_{11}	s_{12}	·	s_{1i}	·	s_{1n}	$f(s_1)$	ω_1
·	·	·	·	·	·	·	·	·
s_l	s_{l1}	s_{l2}	·	s_{li}	·	$s_{\mathrm{l}n}$	$f(s_l)$	ω_l
·	·	·	·	·	·	·	·	·
s_k	s_{k1}	s_{k2}	·	s_{ki}	·	s_{kn}	$f(s_k)$	ω_k
	G_1	G_2		G_i		G_n		

The Gaussian kernel PDF used in ACO_R is a weighted combination of Gaussian functions defined as follows:

$$G_i(x) = \sum_{l=1}^{k} \omega_l g_{li}(x) = \sum_{l=1}^{k} \omega_l \frac{1}{\sigma_{li}\sqrt{2\pi}} e^{-\frac{(x-\mu_{li})^2}{2\sigma^2_{li}}} \tag{1}$$

where $g_{li}(x)$ is a Gaussian function, μ is the vector of means, σ is the vector of standard deviations and ω is the vector of weights. Each solution s_{li} have an associated Gaussian function $g_{li}(x)$, σ_{li} can be calculated by:

$$\sigma_{li} = \xi \sum_{e=1}^{k} \frac{|s_{ei} - s_{li}|}{k-1} \tag{2}$$

where ξ is a parameter of ACO_R algorithm which needs to be set by experience, ξ is a positive number, the larger value of ξ, the lower the convergence speed of the algorithm. In the solution archive, the solutions s_l were sorted according to the value of $f(s_l)$. ACO_R gives good solutions a relatively larger weight to increase its pheromone and thus the next solutions are more likely to be generated near the pervious good solution. Thus, the weight ω_l can be defined by the following equation:

$$\omega_l = \frac{1}{qk\sqrt{2\pi}} e^{-\frac{(l-1)^2}{2q^2k^2}} \tag{3}$$

where q is another parameter of weight and usually set by experience. When q is small, the best-ranked solutions are strongly preferred, and when it is large, the probability becomes more uniform. A more detailed analysis of the parameters ξ and q are presented in Section 2.2.

In the initialization steps, the solution archive is filled with solutions generated by uniform random sampling and sorted according to $f(s_l)$, ξ and q are set by experience, μ_{li} in Eq. (1) are initialized to s_{li}. In the optimization steps, ACO_R generates new solutions by sampling the Gaussian kernel PDF and adding them to the archive. In the Gaussian kernel PDF good solutions have larger weights, thus the new solutions are likely to follow around the good solutions. This strategy has similar effects on pheromone deposition. When adding new solutions to the solution archive, the worst solutions were removed from the archive. This strategy has similar effects to pheromone evaporation. In this way, the solution archive can always get better solutions, and the optimization steps will run constantly until the termination conditions are met.

3.2 Limitations of ACO_R

By shifting the discrete PDF to Gaussian kernel function, ACO_R extends ACO algorithm to continuous domain. However, initialize the solution archive by uniform random sampling may not be the best option, because bad initialization may mislead the algorithms into converging to a local optimal and the convergence speed is slow, moreover, the optimization result is influenced by the parameters. In this section, we will use an example to show these limitations. The testing function f is defined as flows:

$$f(x,y) = 0.5 - \frac{\sin^2\sqrt{x^2+y^2} - 0.5}{(1+0.001(x^2+y^2))^2} \qquad x,y \in [-100,100] \tag{4}$$

The extreme point of function f is (0,0) and $f(0,0) = 1$. Beyond the extreme point there is a ring-shaped ridge with a height of 0.9903 (local maximum). Fig. 1 shows the optimization results of ACO_R. We ran ACO_R 10 times and 400 generations each time, from the result we found ACO_R only converged to global optimal twice and in most of the time it was converged to 0.9903. The reason might be that uniform random sampling contains less information about the optimal solution. When the search cycle reaches a certain number, the pheromone concentrates around the local optimal and ACO_R can hardly find a new path.

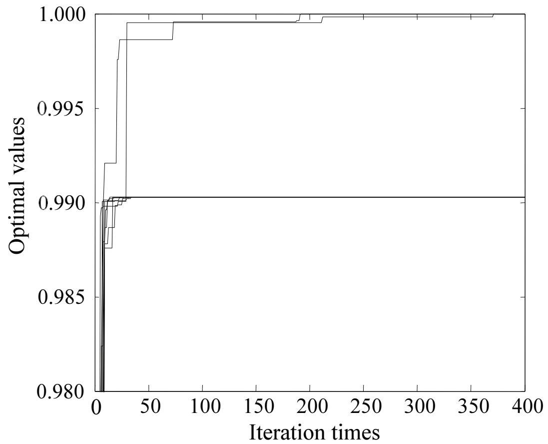


Fig. 1. The adaptation curve of $f(x,y)$ in Eq. (4).

The parameters ξ and q will influence the optimization result, and it is difficult to set these parameters manually. ξ is the weight of σ, when ξ is small the Gaussian function will have a small standard deviation, and results in the newly generated solutions are close to μ. This will weaken the ability to explore new feasible solutions and make the algorithm tend to converge to a local optimal. When ξ is large, newly generated solutions can be far from μ and will enhance the ability to explore new feasible solutions but also will slow down the convergence speed. Parameter q is defined the weights of the Gaussian function. When q approaches 0, only the Gaussian function associated with the current best solution is used for generating new solutions. For instance, the length of the solution archive is 50 and q=0.01 new solutions have the nearly 90% probability to be generated using only the Gaussian function associated with the current best solution. Thus, when q is large, the algorithm samples the search space using a larger number of good solutions in the archive. When q is small the algorithm tends to sample the search space only using the best solution in the archive.

To evaluate the influence of the parameters, we run a test on f with different parameters (ξ is in {0.0001, 0.05, 0.5} and q is in {0.5, 1, 1.4}). In each test, we ran ACO_R 10 times and 400 generations each time, we found the result was not stable. The experiment using parameter ξ of 0.0001 and q of 1 had the best performance and the experiment using parameter ξ of 1.4 and q of 0.5 had the worst performance.

4. Genetic ACO with an Embedded Cloud Model

To overcome the limitations of ACO_R, in this section we present a CG-ACO. In CG-ACO we use a genetic algorithm to generate the initial solutions, we also embedded a cloud model in the updating loop to adaptively set the parameters ξ and q.

4.1 Initialize the Solution Archives using Genetic Algorithm

Similar to ACO, genetic algorithm (GA) is also a metaheuristic optimization algorithm. GA was inspired by the process of natural selection [27]. GA has the ability of rapid global convergence, recently, some of the work attempting to combine GA and ACO has been presented [22,23,28-30]. Xiong et al. [29] compared the convergence speed between ACO and GA and presented a convergence curve. Generally, the convergence speed between GA and ACO is shown in Fig. 2.

From Fig. 2 we can see that at the beginning of GA the convergence speed was fast but it became slower and slower over time. That is because GA doesn't have a positive feedback mechanism. However, ACO shows the opposite trend, the convergence speed was slow at first, but it grew faster and faster over time. The reason is that the pheromone was not enough at the beginning, thus the convergence speed was slow at first, with the positive feedback mechanism (the cumulate of the pheromone) the convergence speed grows faster and faster.

Fig. 2 has a crucial point that is the junction point of the two convergence curves (point a). Before t_a GA was more efficient and after t_a ACO was more efficient. Therefore, if we combine the two methods, t_a is the best time to shift the algorithm. In CG-ACO the iteration number t of GA satisfied $t_b < t < t_c$ where t_b and t_c are the minimum and maximum iterations repetitively. To achieve this, after each iteration we calculate the evolution rate R by:

$$R = \overline{f}_i(s) - \overline{f}_{i+1}(s) \tag{5}$$

where $\overline{f}_i(s)$ is the average of fitness function. If R is small enough ($R < R_{\min}$, $R_{\min}$ is the shifting threshold) for three successive iterations, we stop GA and use the result of GA as the initial solutions of ACO.

From the discussion above we can see that combine GA with ACO can generate several better initial solutions and increase the convergence speed. The global convergence ability of GA is also helpful in avoiding the local optimal solution.

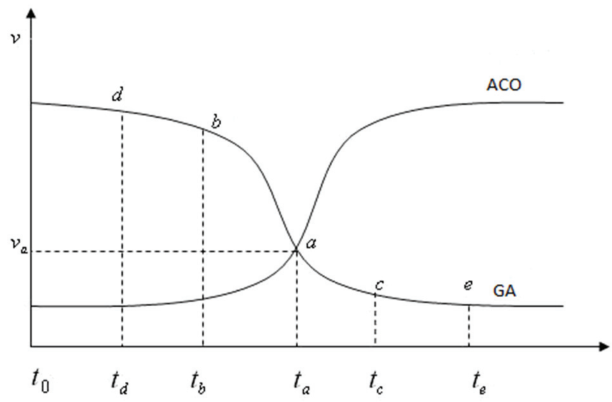


Fig. 2. The convergence curve of ACO and GA.

4.2 Embedding Cloud Model for Adaptive Parameter Selection

Cloud model is a cognitive model proposed by Li et al [31]. It can synthetically describe the randomness and fuzziness of concepts and implement the uncertain transformation between a qualitative concept and its quantitative instantiations. Cloud model has been successfully applied in artificial intelligent control system, data miming and other fields [32-38]. Qi and Yang [32] embedded the cloud model into particle swarm optimization algorithm by introducing the two conditional cloud generators to the hybrid operation and a basic cloud generator to the mutation operation. Wei et al. [33] embedded the cloud model into artificial fish swarm algorithm by using the stable tendency of a basic cloud generator to modify the behavior of the fishes. In this section, we will briefly introduce the cloud model and use it for adaptive parameter selection.

X={x} is a universe of discourse defined by a qualitative concept $\tilde{A}$, $x \in X$ is a random instantiation of concept X, $\mu_{\tilde{A}}(x)$ is the certainty degree of x belonging to X, which corresponds to a random number with a steady tendency. Then, the distribution of x in the universe X can be defined as a cloud and x can be called a cloud drop.

The cloud model can effectively integrate the randomness and fuzziness of concepts and describe the overall quantitative property of a concept using its expectation (E_x), entropy (E_n), and hyper entropy (H_e). There are two kinds of cloud generators, i.e., the forward generator and the backward generator. The forward generator generates cloud drops according to its numerical characteristics and the backward generator calculates the numerical characteristics from the existing cloud drops.

The most commonly used cloud model is the normal cloud model. In a normal cloud model $x \sim N(E_x, \sigma^2)$, $\sigma \sim N(E_n, H_e^2)$, and $\mu_{\tilde{A}}(x)$ can be calculated as:

$$\mu_{\tilde{A}}(x) = e^{-\frac{(x-E_x)^2}{2\sigma^2}} \tag{6}$$

Fig. 3 illustrates a typical normal cloud model. From Fig. 3, we can see that the entropy represents the uncertainty measurement of a qualitative concept and reflects the dispersing extent of the cloud drops. 99.7% of cloud drops are in the range of [E_x-3E_n, E_x+3E_n] and the hyper entropy represents the uncertain degree of entropy.

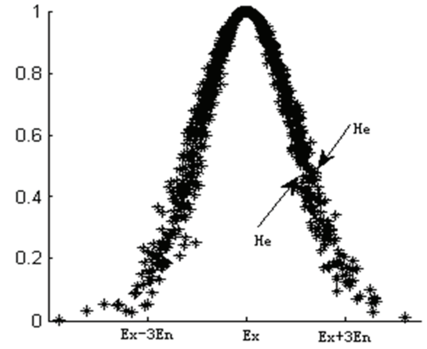


Fig. 3. A typical normal cloud model.

In this work, the solutions in the solution archive are regarded as cloud drops and the solution archive is regarded as a cloud. If the cloud is random and fuzzy, it means we still lack the pheromone thus, we should increase ξ and q to search more solutions on a wider scale; and if the cloud is no longer fuzzy, we should decrease ξ and q to narrow the searching scale and increase the convergence speed.

We used the backward cloud generator to calculate the numerical characteristics of the cloud (the solution archive) and used the entropy to measure its randomness and fuzziness. The entropy E_n of the initial solution archive was used as the standard. We compared the entropy of the i-th iteration E_{ni} with E_n and defined a fuzzy ratio r as follows:

$$r_i = \frac{E_{ni} - E_n}{E_{ni}} \tag{7}$$

In the i+1-th iteration ξ and q were set by the following equation adaptively.

$$\xi_{i+1} = \begin{cases} \xi_0 & r_i > 0.6 \\ 2\xi_0 & 0.2 < r_i < 0.6 \\ 5\xi_0 & r_i < 0.2 \end{cases} \tag{8}$$

$$q_{i+1} = \begin{cases} q_0 & r_i > 0.6 \\ 10q_0 & 0.2 < r_i < 0.6 \\ 100q_0 & r_i < 0.2 \end{cases} \tag{9}$$

4.3 The Steps of CG-ACO

Algorithm 1. CG-ACO metaheuristic

```
GAInitialization()
while shifting conditions not met do
   GA Optimization
end while
ACOInitialization()
while termination conditions not met do
   ACO Optimization
      SolutionConstruction()
      PheromoneUpdate()
      ACOParameterUpdate()
   End ACO Optimization
end while
```

The metaheuristic framework is presented in Algorithm 1 and in the following we will explain these steps in detail.

GAInitialization(): generate M initial population randomly and initialize the crossover rate P_c, mutation rate P_m, maximum iterations I_{max}, minimum iterations I_{min} and minimum evolution rate R_{min}.

After the parameter initialization, GA was used to generate initial solutions until the shifting conditions were met. The shifting conditions are: the number of iterations is greater than I_{min} and the evolution rate is smaller than R_{min} for three successive iterations, or the number of iteration reaches I_{max}.

ACOInitialization(): Initialize the size of solution archive K, the initial values of ξ and q, fill half of the solution archive with K/2 best solutions found by GA and fill the other half with K/2 solutions generated by uniform sampling.

The ACO optimization contains three parts, solution construction, pheromone update and the updating of ξ and q.

SolutionConstruction(): similar as the ACO_R, a solution with n variables needs to be constructed with n steps. In each step we constructed one variable, e.g., for s_{li}, μ_i was initialized to s_i, σ_{li}, and ω_l was calculated according to Eqs. (2) and (3). Then choose one of the Gaussian functions consisting the Gaussian kernel by the probability p_l, and p_l is calculated by:

$$p_l = \frac{\omega_l}{\sum_{r=1}^{k} \omega_r} \tag{10}$$

Generate the solution variable s_{li} by sampling the Gaussian function $g_{li}(x)$ in Eq. (1) and use the same way to generate other solution variables.

PheromoneUpdate(): since the pheromone is stored as a solution archive, the pheromone update is accomplished by adding the set of newly generated solutions to the archive, sorting the solution archive, and removing the same number of worst solutions.

ACOParameterUpdate(): the fuzzy ratio r is calculated according to Eq. (7) and parameters ξ and q are updated according to Eqs. (8) and (9).

5. Experiments and Results

5.1 Comparison with ACO_R and SGA

In this experiment the CG-ACO was compared with its parents, i.e. ACO_R and simple genetic algorithm (SGA). To facilitate the comparison, the testing functions in [5] were used in this experiment:

$$f_1 = \sum_{i-1}^{4} [100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2], \quad x_i \in [-10,10] \tag{11}$$

$$f_2 = \sin(\sum_{i=1}^{5} |x_i - 5|) \Big/ \sum_{i=1}^{5} |x_i - 5| \quad , \quad x_i \in [1,10] \tag{12}$$

The global minimum point of f_1 is x_i=0, and the minimum value is 0. Since it is a recursive function, in the experiment, the algorithm has successfully found the global minimum of f_1 which means that the output is smaller than 0.0001. The global maximum point of f_2 is x_i=5, and the maximum value is 1, in the experiment, the algorithm has successfully found the global maximum of f_2 which means that the output is greater than 0.999 [5].

The parameter configuration of ACO_R is the same with the best parameters used in [5], where the ant number m=70, the size of the solution archive K=45, the parameters ξ=1 and q=0.0001. The parameters configuration of SGA is that: the size of the population M=50, one-point crossover, one-point mutation and the roulette wheel selection strategies were used, the crossover rate P_c=0.9, the mutation rate P_m=0.1. The parameter configuration of CG-ACO shares most of the values in the ACO_R and SGA, i.e., m=70, K=45, the initial value of ξ=1, the initial value of q=0.0001, the crossover rate P_c=0.9, the mutation rate P_m=0.1. Other configurations of CG-ACO's parameters are maximum iterations I_{max}=400, minimum iterations I_{min}=100, and minimum evolution rate R_{min}=10. Each optimization algorithm was tested ten times, and the results are presented in Table 2. From Table 2, we can see that the CG-ACO can achieve better performance than ACO_R and SGA, the average output of CG-ACO is closest to the global optimal. Moreover, we found the hybrid of GA and ACO is also helpful in avoiding the local optimal solution.

The CG-ACO reaches the global optimal value for 19 of 20 times, which are much more than other methods. In the experiment, the CG-ACO usually shifts to ACO part after around 100 GA iterations, and for most of the time the ACO part can reach the global optimal within 50 iterations due to the adaptive parameter configuration strategy. In the experiment, CG-ACO used no more than 150 iterations (total iterations in GA part and ACO part) on average to reach the global optimal, which is faster than the ACO_R (more than 200 iterations).

Table 2. Optimization result of f_1 and f_2

Test function	Algorithm	Best output	Average output	Average iterations to reach the global optimal	Successful rate
f_1	SGA	0.002	0.0895	-	0/10
	ACO_R	0	0.0267	267	3/10
	CG-ACO	0	0.0051	131	9/10
f_2	SGA	1	0.9621	354	2/10
	ACO_R	1	0.9865	211	6/10
	CG-ACO	1	1	147	10/10

5.2 Comparison with Other Fusion Methods

In the second experiment, we did some comparison between CG-ACO and some optimization algorithms also embedded with cloud model, i.e., the CQPSO [32] and the CAFSA [33]. The function reported in [20] was tested in this section:

$$f_3(x,y)=0.5+\frac{\sin^2\sqrt{x^2+y^2}-0.5}{(1+0.001(x^2+y^2))^2},\quad x,y\in[-100,100] \tag{13}$$

The global minimum point of f_3 is at (0,0) and the minimum value is 0. f_3 has an infinite number of local minimums. In the experiment the algorithm has successfully found the global minimum of f_3 only if the output is 0.

The parameters of CG-ACO are the same as the previous experiment and the parameters of CQPSO and CAFSA are configured as [32] and [33], with the maximum iterations of 400. Each optimization algorithm was tested 10 times and the results are presented in Table 3. From Table 3, we can see that the CG-ACO also has the advantages in reaching the global optimal than CQPSO and CAFSA.

Table 3. Optimization result of f_3

Test function	Algorithm	Best output	Average output
f_3	CQPSO [32]	8.1531×10^{-8}	3.9251×10^{-7}
	CAFSA [33]	8.1531×10^{-8}	3.9254×10^{-7}
	CG-ACO	0	0

5.3 Parameter Stability

5.3.1 ACO parameters

In this section, the parameter stability was evaluated by testing the proposed method with different parameters. To evaluate the influence of the parameters effectively, we designed the experiment according to the uniform design method [39]. In the experiment the ant number m and size of the solution archive K varied from 40 to 60, the parameters ξ varied from 0.4 to 1.5 and q varied from 0.0001 to 0.5. These parameters were mixed according to uniform design table $U^{*}(6^{4})$ and listed in Table 4. From Table 4, we can see the parameters are almost randomly distributed.

Table 4. Experiment design according to uniform design table

Parameter set	m	k	ξ	q
1	40	40	0.8	0.5
2	40	50	1.5	0.2
3	50	60	0.6	0.1
4	50	40	1.2	0.01
5	60	50	0.4	0.001
6	60	60	1	0.0001

In this section we also used function f_3 since it is more complicated than f_1 and f_2. Each parameter set was tested 10 times and the maximum iteration number is 400. The experiment results are listed in Table 5. From Table 5, we can see that the algorithm not only has good performance but also is robust to ACO parameter variations.

Table 5. Result of parameter stability experiment

Test	Best output	Average output	Successful rate
1	0	0.0044	8/10
2	0	0.0077	8/10
3	0	0	10/10
4	0	0	10/10
5	0	0	10/10
6	0	0	10/10

5.3.2 GA parameters

Combining GA with ACO can increase the convergence speed and the possibility to reach the global optimal solution; however, it also makes the algorithm have more parameters to optimize, i.e., the population size M, the crossover rate P_c and the mutation rate P_m. In this work to facilitate the comparison the population size M was set to 50, the crossover rate P_c was set to 0.9 and the mutation rate P_m was set to 0.1 which are the same as the optimal GA parameters in the compared literature. To evaluate the parameter stability, we did experiments to evaluate the sensitivity of GA parameters in the hybrid algorithm.

In the experiments the M was set between 40 to 60 with the step of 1, P_c was set between 0.8 to 0.9 with the step of 0.01 and P_m was set between 0.05 to 0.15 with the step of 0.01, thus, we got 1,000 sets of GA parameters. We reran each pervious test with 1,000 different parameter sets, from the experiment we found the results were the same as that in Table 5. One reason for this may be that introducing GA is to get some better initial solutions for ACO but not the best solution, thus the performance of GA doesn't need to be tuned very high, some typical parameters would be sufficient and the ACO algorithm with Cloud model will further improve these solutions.

From the parameter stability experiment we can see the performance of CG-ACO is relatively stable to some changes in both ACO and GA parameters. First, by introducing the cloud model the parameters of ant colony can be set adaptively. Second, after we used a genetic algorithm to initialize the solution archive with some good solutions, these solutions would be updated to better solutions by the ant colony algorithm. Thus, we can get good performance without taking too much effort to tune these parameters.

6. Conclusion

In this paper, we proposed a new scheme of ACO_R, the genetic algorithm and cloud model were introduced to the ACO_R scheme. Generally, GA converges to the global optimal faster than ACO_R thus for the initialization of the solution archive, half of the solutions were filled using the best solutions founded by genetic algorithm and half were filled using uniform sampling. This strategy is helpful to increase the convergence speed and decrease the probability of falling into the local minimums. We also embedded a cloud model to configure the parameters of ACO according to the fuzziness of the current solution archive adaptively.

The experiments show that CG-ACO can achieve better performance than ACO_R, SGA, CQPSA and CAFSA. Moreover, CG-ACO reached the global optimal more often than other algorithms. The parameter stability test showed that the CG-ACO was also robust to the parameter variations.

Acknowledgement

This paper is supported by the Heilongjiang philosophy and social science project (No. 17GLB015).

References

[1] M. Dorigo, “Optimization, learning and natural algorithms,” Ph.D. dissertation, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, 1992.

[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, pp. 53-66, 1997.

[3] M. Dorigo and T. Stutzle, *Ant Colony Optimization*. Cambridge, MA: MIT Press, 2004.

[4] M. Guntsch and M. Middendorf, “A population based approach for ACO,” in *Applications of Evolutionary Computing*. Heidelberg, Germany: Springer, 2002, pp. 72-81.

[5] K. Socha and M. Dorigo, “Ant colony optimization for continuous domains,” *European Journal of Operational Research*, vol. 185, no. 3, pp. 1155-1173, 2008.

[6] M. de Paula Marques, F. R. Durand, and T. Abrao, “WDM/OCDM energy-efficient networks based on heuristic ant colony optimization,” *IEEE Systems Journal*, vol. 10, no. 4, pp. 1482-1493, 2016.

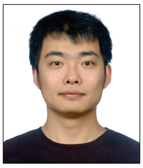
[7] C. Liu, “Optimal design of high-rise building wiring based on ant colony optimization,” *Cluster Computing*, vol. 22, pp. 3479-3486, 2018.

[8] S. N. Sabri and R. Saian, “Predicting flood in Perlis using ant colony optimization,” *Journal of Physics: Conference Series*, vol. 855, article no. 012040, 2017.

[9] L. M. Gambardella and M. Dorigo, “Solving symmetric and asymmetric TSPs by ant colonies,” in *Proceedings of IEEE International Conference on Evolutionary Computation*, Nagoya, Japan, 1996, pp. 622-627.

[10] E. D. Taillard, “FANT: fast ant system,” Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Manno, Switzerland, *Technical Report IDSIA-46-98*, 1998.
[11] T. Stutzle and H. Hoos, “MAX-MIN Ant System and local search for the traveling salesman problem,” in *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC)*, Indianapolis, IN, 1997, pp. 309-314.
[12] K. Doerner, R. F. Hartl, and M. Reimann, “Are COMPETants more competent for problem solving? The case of a multiple objective transportation problem,” *Central European Journal of Operations Research*, vol. 11, no. 2, pp. 115-141, 2003.
[13] K. Doerner, W. J. Gutjahr, R. F. Hartl, C. Strauss, and C. Stummer, “Pareto ant colony optimization: a metaheuristic approach to multiobjective portfolio selection,” *Annals of Operations Research*, vol. 131, no. 1-4, pp. 79-99, 2004.
[14] S. Iredi, D. Merkle, and M. Middendorf, “Bi-criterion optimization with multi colony ant algorithm,” in *Evolutionary Multi-Criterion Optimization*. Heidelberg, Germany: Springer, 2001, pp. 359-372.
[15] M. Guntsch and M. Middendorf, “Solving multi-criteria optimization problems with population-based ACO,” in *Evolutionary Multi-Criterion Optimization*. Heidelberg, Germany: Springer, 2003, pp. 464-478.
[16] X. M. Hu, J. Zhang, and Y. Li, “Orthogonal methods based ant colony search for solving continuous optimization problems,” *Journal of Computer Science and Technology*, vol. 23, no. 1, pp. 2-18, 2008.
[17] X. M. Hu, J. Zhang, H. S. H. Chung, Y. Li, and O. Liu, “SamACO: variable sampling ant colony optimization algorithm for continuous optimization,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 40, no. 6, pp. 1555-1566, 2010.
[18] S. H. Pourtakdoust and H. Nobahari, “An extension of ant colony system to continuous optimization problems,” in *Ant Colony Optimization and Swarm Intelligence*. Heidelberg, Germany: Springer, 2004, pp. 294-301.
[19] F. O. de Franca, G. P. Coelho, F. J. Von Zuben, and R. R. de Faissol Attux, “Multivariate ant colony optimization in continuous search spaces,” in *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, Atlanta, GA, 2008, pp. 9-16.
[20] T. Liao, K. Socha, M. A. Montes de Oca, T. Stutzle, and M. Dorigo, “Ant colony optimization for mixed-variable optimization problems,” *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 503-518, 2014.
[21] L. M. Gambardella and M. Dorigo, “Ant-Q: a reinforcement learning approach to the traveling salesman problem,” in *Machine Learning Proceedings 1995*. San Francisco, CA: Morgan Kaufmann Publishers, 1995, pp. 252-260.
[22] M. Mahi, O. K. Baykan, and H. Kodaz, “A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem,” *Applied Soft Computing*, vol. 30, pp. 484-490, 2015.
[23] S. Nemati, M. E. Basiri, N. Ghasem-Aghaee, and M. H. Aghdam, “A novel ACO–GA hybrid algorithm for feature selection in protein function prediction,” *Expert Systems with Applications*, vol. 36, no. 10, pp. 12086-12094, 2009.
[24] W. Alsaeedan, M. E. B. Menai, and S. Al-Ahmadi, “A hybrid genetic-ant colony optimization algorithm for the word sense disambiguation problem,” *Information Sciences*, vol. 417, pp. 20-38, 2017.
[25] R. Goel and R. Maini, “A hybrid of ant colony and firefly algorithms (HAFA) for solving vehicle routing problems,” *Journal of Computational Science*, vol. 25, pp. 28-37, 2018.
[26] I. Karakonstantis and A. Vlachos, “Hybrid ant colony optimization for continuous domains for solving emission and economic dispatch problems,” *Journal of Information and Optimization Sciences*, vol. 39, no. 3, pp. 651-671, 2018.
[27] M. Mitchell, *An Introduction to Genetic Algorithms*, Cambridge, MA: MIT Press, 1996.

[28] B. Ge, J. H. Han, Z. Wei, L. Cheng, and Y. Han, "Dynamic hybrid ant colony optimization algorithm for solving the vehicle routing problem with time windows," *Pattern Recognition And Artificial Intelligence*, vol. 28, no. 7, pp. 641-650, 2015.
[29] Z. H. Xiong, S. K. Li, and J. H. Chen, "Hardware/Software partitioning based on dynamic combination of genetic algorithm and ant algorithm," *Journal of Software*, vol. 16, no. 4, pp. 503-512, 2005.
[30] J. Li, "Combination of genetic & ant colony algorithms for multi-project resource leveling problem," *Computer Integrated Manufacturing Systems*, vol. 16, no. 3, pp. 643-649, 2010.
[31] D. Li, C. Liu, and W. Gan, "A new cognitive model: cloud model," *International Journal of Intelligent Systems*, vol. 24, no. 3, pp. 357-375, 2009.
[32] M. Qi and A. Yang, "Quantum particle swarm optimization based on cloud model cloud droplet strategy," *Computer Engineering and Applications*, vol. 48, no. 24, pp. 49-52, 2012.
[33] X. Wei, H. Zeng, and Y. Zhou, "Cloud theory-based artificial fish swarm algorithm," *Computer Engineering and Applications*, vol. 46, no. 22, pp. 26-29, 2010.
[34] P. Wang, X. Xu, S. Huang, and C. Cai, "A linguistic large group decision making method based on the cloud model," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 6, pp. 3314-3326, 2018.
[35] X. Shang, P. Ma, and T. Chao, "Performance evaluation of electromagnetic railgun exterior ballistics based on cloud model," *IEEE Transactions on Plasma Science*, vol. 45, no. 7, pp. 1614-1621, 2017.
[36] J. Cui, Q. Zheng, Y. Xin, C. Zhou, Q. Wang, and N. Zhou, "Feature extraction and classification method for switchgear faults based on sample entropy and cloud model," *IET Generation, Transmission & Distribution*, vol. 11, no. 11, pp. 2938-2946, 2017.
[37] S. Yang, R. Jiang, H. Wang, and S. S. Ge, "Road constrained monocular visual localization using Gaussian-Gaussian Cloud model," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 12, pp. 3449-3456, 2017.
[38] Y. Lin, L. Zhao, H. Li, and Y. Sun, "Air quality forecasting based on cloud model granulation," *EURASIP Journal on Wireless Communications and Networking*, vol. 2018, no. 1, article no. 106, 2018.
[39] K. T. Fang, D. K. J. Lin, P. Winker, and Y. Zhang, "Uniform design: theory and application," *Technometrics*, vol. 42, no. 3, pp. 237-248, 2000.



Peng Wang https://orcid.org/0000-0001-9436-2499

He received Ph.D. degree in School of Computer Science and Technology from Harbin Institute of Technology in 2014. Since 2014, he is with the College of Arts and Sciences from Northeast Agricultural University. His current research interests include machine learning and computer vision.

Jiyun Bai https://orcid.org/0000-0001-7009-3052

She received Ph.D. degree in School of Astronautics from Harbin Institute of Technology in 2013. Since 2001, she is with the College of Arts and Sciences from Northeast Agricultural University. Her current research interests include optimization algorithms and intelligent agriculture.

Jun Meng https://orcid.org/0000-0002-3620-0202

He received Ph.D. degree in School of Engineering from Northeast Agricultural University in 2002. Since 1990, he is with the College of Arts and Sciences from Northeast Agricultural University. His current research interests include agricultural system engineering and agricultural data analysis.