Continuous Function Optimization Using Hybrid Ant Colony Approach with Orthogonal Design Scheme

Jun ZHANG♠[1], Wei-neng Chen♠, Jinghui Zhong♠, Xuan Tan♠ and Yun Li★

♠Department of Computer Science, Sun Yat-sen University, P.R.China
junzhang@ieee.org

★Department of Electronics and Electrical Engineering , University of Glasgow, UK

Abstract. A hybrid Orthogonal Scheme Ant Colony Optimization (OSACO) algorithm for continuous function optimization (CFO) is presented in this paper. The methodology integrates the advantages of Ant Colony Optimization (ACO) and Orthogonal Design Scheme (ODS). OSACO is based on the following principles: a) each independent variable space (IVS) of CFO is dispersed into a number of random and movable nodes; b) the carriers of pheromone of ACO are shifted to the nodes; c) solution path can be obtained by choosing one appropriate node from each IVS by ant; d) with the ODS, the best solved path is further improved. The proposed algorithm has been successfully applied to 10 benchmark test functions. The performance and a comparison with CACO and FEP have been studied.

1 Introduction

Ant Colony Optimization (ACO) was first proposed by Marco Dorigo in the early 1990s in the light of how ants manage to establish the shortest path from their nest to food sources [1]. By now, the idea of ACO has been used in a large number of intractable combinatorial problems and become one of the best approaches to TSP [2], quadratic assignment problem [3], data mining [4], and network routing [5].

In spite of its great success in the field of discrete space problems, the uses of ACO in continuous problems are not significant. Bilchev and Parmee [6] first introduced an ACO metaphor for continuous problems in 1995 but the mechanism of ACO was only used in the local search procedure. Later, Wodrich and Bilchev [7] introduced an effective bi-level search procedure using the idea of ants. This algorithm, which was referred to as CACO, also employed some ideas of GA. The algorithm was further extended by Mathur *et al.* [8] and the performances were significantly improved. Based on some other behaviors of ants, two algorithms called API [9] and CIAC [10] were proposed, but they did not follow the framework of ACO strictly, and poor performances are observed in high-dimension problems. Overall, the use of ACO in continuous space optimization problems is not significant.

[1] Corresponding author: junzhang@ieee.org. This work was supported in part by NSF of China Project No.60573066 and NSF of Guangdong Project No. 5003346

This paper aims at proposing an ant colony algorithm with orthogonal scheme (OSACO) for continuous function optimization problems. OSACO is based on the following principles: a) each independent variable space (IVS) of CFO is dispersed into a number of random nodes; b) the carriers of pheromone of ACO are shifted to the nodes; c) solution path can be obtained by choosing one appropriate node from each IVS by ant; d) with the ODS, the best solved path (SP) is further improved. The proposed algorithm has been successfully applied to 10 benchmark test functions. The performance and a comparison with CACO and FEP have been studied.

2 Background

2.1 ACO

The idea underlying ACO is to simulate the autocatalytic and positive feedback process of the forging behavior of real ants. Once an ant finds a path successfully, pheromone is deposited to the path. By sensing the pheromone ants can follow the path discovered by other ants. This collective pheromone-laying and pheromone-following behavior of ants has become the inspiring source of ACO.

2.2 Orthogonal Experimental Design

The goal of orthogonal design is to perform a minimum number of tests but acquire the most valuable information of the considered problem [11][12]. It performs by judiciously selecting a subset of level combinations using a particular type of array called the orthogonal array (OA). As a result, well-balanced subsets of level combinations will be chosen.

3 The Orthogonal Search ACO Algorithm (OSACO)

The characteristics of OSACO are mainly in the following aspects: a) each independent variable space (IVS) of CFO is dispersed into a number of random nodes; b) the carriers of pheromone of ACO are shifted to the nodes; c) SP can be obtained by choosing one appropriate node from each IVS by ant; d) with the ODS, the best SP is further improved.

Informally, its procedural steps are summarized as follows. ***Step 1)*** *Initialization*: nodes and the pheromone values of nodes are initialized; ***Step 2)*** *Solution Construction*: Ants follow the mechanism of ACO to select nodes separately using pheromone values and form new SPs; ***Step 3)*** *Sorting and Orthogonal Search*: SPs in this iteration are sorted and the orthogonal search procedure is applied to the global best SP; ***Step 4)*** *Pheromone Updating*: Pheromone values on all nodes of all SPs are updated using the pheromone depositing rule and the pheromone evaporating rule;

Step 5) *SP Reconstruction*: A number of the worst SPs are regenerated; ***Step 6)*** *Termination Test*: If the test is passed, stop; otherwise go to ***step 2)***.

To facilitate understanding and explanation of the proposed algorithm, we take the optimization work as minimizing a D-dimension function $f(X)$, $X=(x_1,x_2,\cdots,x_D)$. The lower and upper bounds of variable x_i are $lowBound_i$ and $upBound_i$. Nevertheless, without loss of generality, this scheme can also be applied to other continuous space optimization problems.

3.1 Definition of Data structure

We first define the structure of a Solution Path (SP):

```
structure SP
begin
    real node [D];          % the nodes of the SP;
    real r [D]              % the search radiuses for each node of the SP;
    real t [D]              % the pheromone values for each node of the SP;
    real value;             % the function value of the SP;
end
```

Fig. 1 The structure of a "SP"

In the above definition, each SP includes four attributes: the nodes of the SP in all IVS, the search radiuses for each node which are used during the orthogonal search procedure, the pheromone values for each node which are used in the solution construction procedure, and the function value of the SP.

Assume that the number of SPs (ants) in the algorithm is *SPNUM*. In the following text, we denote the four attributes of the sp_k ($1\le k\le SPNUM$) as $SP_k.NODE(SP_k.node_1, SP_k.node_2,\cdots,SP_k.node_D)$, $SP_k.R(SP_k.r_1, SP_k.r_2,\cdots, SP_k.r_D)$, $SP_k.T(SP_k.\tau_1, SP_k.\tau_2,\cdots, SP_k.\tau_D)$ and $SP_k.value$.

3.2 Initialization

In the beginning, *SPNUM* nodes are created randomly in each IVS and form *SPNUM* SPs. Pheromone values of all nodes are set to $\tau_{initial}$. ($\tau_{initial}$ is also the unitage of pheromone values and we set $\tau_{initial}=1$ for computational convenience purpose.) Function values of all SPs are calculated and sorted in ascending order. Pheromone values are updated using the following formula:

$$SP_i.\tau_j \leftarrow SP_i.\tau_j + \alpha\cdot(GOODNUM - rank_i)\cdot\tau_{initial},\ \text{if } GOODNUM - rank_i > 0 \qquad \textbf{(1)}$$

$SP_i.\tau_j$ is the pheromone value of the j^{th} node of SP_i. *GOODNUM* and α are two parameters. *GOODNUM* ($1\le GOODNUM\le SPNUM$) represents the number of SPs that can obtain additional pheromone and $\alpha\in[0,1]$ determines the amount of pheromone deposited to the SPs. $rank_i$ represents the rank of SP_i. Search radiuses of all nodes are set to $(upBound_i-lowBound_i)/SPNUM$. Moreover, the best SP will be preserved additionally and is denoted as $SP_{SPNUM+1}$.

3.3 Solution Construction

All ants build their SPs to the problem incrementally in this phase. The new SPs created in this phase are denoted as *antSP*. The procedure for ant k ($1 \le k \le SPNUM$) to build its solution $antSP_k$ is as follows:

$$\begin{cases} antSP_k \leftarrow SP_k, & \text{if } q > q0 \\ antSP_k \text{ is created using pheromone information}, & \text{otherwise} \end{cases}, \quad 1 \le k \le SPNUM; \tag{2}$$

At first, a random number $q \in (0,1)$ is created and is compared with a parameter $q0$ ($0 \le q0 \le 1$). If $q > q0$, the solution created by ant k is the same as SP_k. All attributes of SP_k are reserved by $antSP_k$. Otherwise, a new solution is built by ant k in terms of the pheromone information using the roulette wheel scheme given by equation (4). The node in each IVS has to be selected separately based on the pheromone values of all nodes in that IVS. The probability of selecting $SP_j.node_i$ ($1 \le j \le SPNUM+1$) as the ith node of SP constructed by ant k is in direct proportion to the pheromone value of the ith node in SP_j. It is important to note that the attributes of the best SP $SP_{SPNUM+1}$ can also be selected by ants.

$$p(antSP_k.node_i = SP_j.node_i) = \frac{SP_j.\tau_i}{\sum_{l=1}^{SPNUM+1} SP_l.\tau_i}, \tag{3}$$
$$(1 \le j \le SPNUM+1, 1 \le k \le SPNUM, 1 \le i \le D)$$

Suppose $SP_j.node_i$ is selected to be the ith node of the SP constructed by ant k, the pheromone value of the ith node on SP_j will also be inherited to $antSP_k$, that is, $antSP_k.\tau_i = SP_j.\tau_i$. After all nodes have been selected by ant k, the complete SP is evaluated, that is, $antSP_k.value = f(antSP_k.NODE)$. Search radiuses of all nodes on $antSP_k$ will also be regenerated, that is, $antSP_k.r_i = (upBound_i - lowBound_i)/SPNUM$ ($1 \le i \le D$).

3.4 Sorting and Orthogonal Search

All new SPs created by ants ($antSP_k$ ($1 \le k \le SPNUM$)) are sorted in ascending order based on their function values. If the function value of the best SP created by ants is smaller than $SP_{SPNUM+1}.value$, the best SP is preserved in the place of $antSP_{SPNUM+1}$, otherwise $antSP_{SPNUM+1} = SP_{SPNUM+1}$. Then, the orthogonal search procedure will be implemented to the global best SP $antSP_{SPNUM+1}$.

1) Orthogonal Search

In order to introduce the orthogonal design technique to this case, we assume that each IVS corresponds to a single factor of the experiment. Also, we divide the search range in each IVS of a SP into l-1 segments to obtain l dividing values. These l dividing values are accordingly considered as the l levels of the experiment. Hence, if we are optimizing an D-dimension object function $f(x_1, x_2, \cdots x_D)$, we can simply take the n variables $(x_1, x_2, \cdots x_D)$ as the D factors. When acquiring the l dividing values of the ith IVS from a SP S located at $S.NODE(S.node_1, S.node_2, \cdots, S.node_D)$, we first compute the search range of each IVS r_i=$S.r_i$·*ran*, where $S.r_i$ is the search radius in the i^{th} IVS of that SP and *ran* is a random number distributed in [0,1]. Then we could get $S.node_i$-r_i as the lower bound of this IVS and $2r_i$ as its search diameter. Thus, $S.node_i-r_i+2jr_i/(l-1)$, $(0\le j\le l-1)$ are just the l dividing values we need. With that, the SP search problem is formulated as a D-factor experiment with all factors having l levels and an OA can be applied to search the SP. We call the combinations of factor levels generated by OA as the orthogonal points. In a 3-dimension SP, we obtain two levels of each IVS simply by using the two ends of each edge. Then the OA $L_4(2^3)$ is applied and the four points are finally selected.

2) SP Moving

Soon after all orthogonal points have been selected, they are evaluated in the function $f(x_1,x_2,\cdots x_D)$. Once the old SP is worse than the best one of all these orthogonal points, it will be replaced by the best one. That is, all nodes of the old SP are replaced by the nodes of the best orthogonal point.

3) Radius Adapting

The characteristic of a SP is adaptive, that is, the radiuses of all nodes ($S.r_1$, $S.r_2$,⋯, $S.r_D$) would adjust themselves during the algorithm by applying (5), where $\theta(0\leqslant\theta\leqslant 1)$ is a parameter.

$$S.r_i \leftarrow \begin{cases} S.r_i/\theta, & \text{if SP } S \text{ is replaced by a new one} \\ S.r_i*\theta, & \text{otherwise} \end{cases} \tag{4}$$

This can effectively help us to improve the SP by deciding whether to move it faster or to let it shrink. If the SP is not substituted, it is probably that the best solution area is inside the search range of the SP so that we decrease its radius to obtain a solution in higher precision. Otherwise, the best solution of the test function may not lie in the search range of this SP. In this case, we enlarge the search range of the SP to make it move faster to a better area.

3.5 Pheromone Updating

GOODNUM (0≤*GOODNUM*≤*SPNUM*) is a parameter which represents the number of SPs that can obtain additional pheromone, that is, pheromone will only be deposited to the best *GOODNUM* SPs. Additionally, pheromone on all nodes of all SPs will be evaporated. The pheromone updating procedure is executed as follows:

$$\begin{cases} antSP_k.\tau_i \leftarrow \rho\cdot antSP_k.\tau_i + \alpha\cdot(GOODNUM - rank_k)\cdot\tau_{initial}, & \text{if } rank_k < GOODNUM \\ antSP_k.\tau_i \leftarrow \rho\cdot antSP_k.\tau_i, & \text{otherwise} \end{cases}, \quad (1\le k\le SPNUM, 1\le i\le D) \tag{5}$$

$\alpha\in(0,1)$ and $\rho\in(0,1)$ are two parameters. α determines the amount of pheromone that is deposited to the SPs. ρ determines the evaporating rate of the pheromone. $rank_i$ represents the rank of $antSP_i$.

3.6 SP Reconstruction

At the end of each iteration, a number of the worst SPs will be forgotten by ants and will be regenerated randomly. (The number of the deserted SPs is denoted as *DUMPNUM*, which is a parameter ($0 \le DUMPNUM \le SPNUM$).) Then, all old SPs are replaced by new generation of SPs constructed by ants, that is, $SP_k = antSP_k$ ($1 \le k \le SPNUM+1$).

4 Computational Results and Discussing

To demonstrate the effectiveness of the proposed algorithm, 10 test functions in table 2 are selected from [13] where we can obtain more information about these functions. f_1-f_5 are unimodal functions used to test the convergence rate of an algorithm and to evaluate how much precise an algorithm can obtain. f_6-f_{10} are multimodal functions with local optima. Moreover, f_1-f_9 are 30-dimension functions. A good performance on such functions is always taken as a proof of an algorithm's effectiveness.

Table 1. List of 10 Test Functions

Test functions	Search Domain	Test functions	Search Domain
$f_1(x)=\sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$	$f_6(x)=\sum_{i=1}^{D} -x_i \sin(\sqrt{x_i})$	$[-500, 500]^D$
$f_2(x)=\sum_{i=1}^{D}(\sum_{j=1}^{i} x_j)^2$	$[-100, 100]^D$	$f_7(x)=\sum_{i=1}^{D}[x_i^2-10\cos(2\pi x_i)+10]$	$[-5.12, 5.12]^D$
$f_3(x)=\max_i(\lvert x_i\rvert, 1\le i\le D)$	$[-100, 100]^D$	$f_8(x)=\frac{1}{4000}\sum_{i=1}^{D} x_i^2-\prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}})+1$	$[-600, 600]^D$
$f_4(x)=$ $\sum_{i=1}^{D-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	$[-30, 30]^D$	$f_9(x)=\frac{\pi}{D}\{10\sin^2(3\pi x_1)+\sum_{i=1}^{D-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]$ $+(x_n-1)^2[1+\sin^2(2\pi x_n)]\}+\sum_{i=1}^{D}u(x_i,10,100,4)$	$[-50, 50]^D$
$f_5(x)=\sum_{i=1}^{D}(\lfloor x_i+0.5\rfloor)^2$	$[-100, 100]^D$	$f_{10}(x)=\sum_{i=1}^{11}[a_i-\frac{x_1(b_i^2+b_i x_2)}{b_i^2+b_i x_3+x_4}]^2$	$[-5,5]^D$

We first set the parameters of the proposed algorithm. Parameters are configured carefully and the ones we use are given below. It is important to note that different test functions may result in different good parameter settings, but the settings we use here are able to balance the performances in all test functions.

Parameters are set as follows: the number of SPs or ants *SPNUM*=20, pheromone depositing rate α=0.9, pheromone evaporating rate ρ=0.8, the probability of selecting a new SP $q0$=0.9, the changing rate of the radiuses θ=0.8, the number of SPs that receive additional pheromone $GOODNUM=SPNUM \times 0.1=2$, and the number of deserted SPs $DUMPNUM=SPNUM \times 0.05=1$. Also, we use the OA $L_{81}(3^{40})$ to satisfy the need of optimizing 30-dimension functions in f_1-f_9, and use $L_9(3^4)$ in f_{10}, which is only a 4-dimension test function.

The proposed algorithm is compared with two other algorithms – CACO [8] and FEP [13]. CACO is by now one of the top algorithms that use the idea of ACO for continuous optimization problems. FEP is one of the state-of-the-art approaches to continuous function optimization problems. Parameters of these two algorithms are set in terms of paper [8] and [13] respectively.

Table 2. Comparison of optimization results and computational effort between OSACO, CACO, and FEP (All results are averaged over 100 runs.)

	OSACO		CACO		FEP	
	Computational Effort	Mean best (Variance)	Computational Effort	Mean best (Variance)	Computational Effort	Mean best (Variance)
f_1	150000	**1.669e-34** 1.553e-33	150000	3.30e-21 1.21e-20	150000	5.7e-4 1.3e-4
f_2	500000	**1.31e-71** 5.21e-71	500000	0.1173 0.0819	500000	0.016 0.014
f_3	500000	**1.30e-37** 3.89e-37	500000	0.365 0.702	500000	0.30 0.50
f_4	2000000	**0.3596** 1.0443	2000000	38.003 25.715	2000000	5.06 5.87
f_5	50000	0 0	50000	0 0	150000	0 0
f_6	500000	**-12569.49** 1.28e-11	900000	-12446.71 133.928	900000	-12554.5 52.6
f_7	500000	**7.71e-10** 7.71e-9	500000	2.577 1.715	500000	0.046 0.012
f_8	200000	0.01078 0.01136	200000	**0.00826** 0.01391	200000	0.016 0.022
f_9	800000	**1.570e-32** 2.751e-47	1000000	0.00311 0.01778	150000	9.2e-6 3.6e-6
f_{10}	400000	**4.294e-4** 3.460e-4	400000	5.873e-4 1.150e-4	400000	5.0e-4 3.2e-4

The computational results are shown in Table 2. Obviously, OSACO performs better than CACO and FEP in most cases. In unimodal functions, OSCAO obtains higher precision than CACO and FEP in f_1-f_4, and gets the global best solutions of f_5 much faster than FEP. These prove that the use of the orthogonal search scheme can significantly improve the search precision of the algorithm. In multimodal functions, OSACO obtains the best final results of f_6, f_7, f_9, and f_{10}, and the result of f_8 obtained by OSACO is only slightly worse than CACO, but better than FEP. In f_9, though OSACO seems slower than FEP, but manages to get much higher precision than FEP.

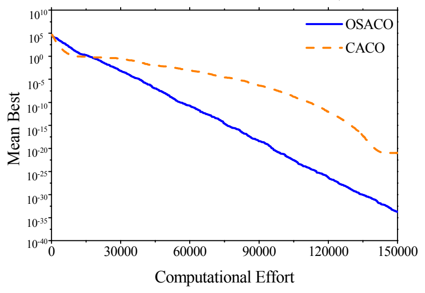

Fig. 2 Convergent speed of OSACO and CACO in f_1

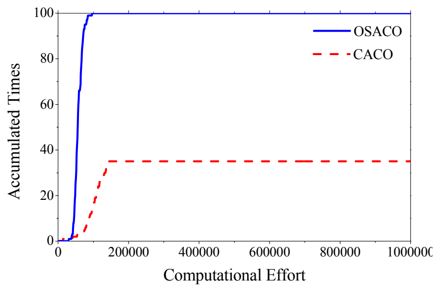

Fig. 3 Accumulative times of errors smaller than 1.0 in f_6

Additionally, Fig. 2 illustrates the comparison of the convergent speed between OSACO and CACO in unimodal function f_1. It is apparent that OSACO is able to obtain higher precision with fewer computational efforts. Fig. 3 reveals the accumulative times of errors smaller than 1.0 in multimodal function f_6 within 100 runs. OSACO successes in all times with at most 100,000 times of function evaluations, while CACO only successes for 35 times after 1,000,000 times of function evaluations. These demonstrated the effectiveness of OSACO.

5 Conclusion

The hybrid orthogonal scheme ant colony optimization (OSACO) algorithm has been proposed. The general idea underlying this algorithm is to use the orthogonal design scheme to improve the performance of ACO in the filed of continuous space optimization problems. Experiments on 10 diverse test functions presented the effectiveness of the algorithm compared with CACO and FEP.

References

[1] M. Dorigo, V. Maniezzo and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Trans. on systems man, and cybernetics - part B: cybernetics,* vol. 26, 1996, pp. 29-41.

[2] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to TSP", *IEEE Trans. Evol. Comput.*, vol. 1, 1997, pp. 53–66.

[3] M. Dorigo, G. Di Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization," *Artificial Life*, 1999,5(2), pp. 137-172.

[4] R. S. Parpinelli, H. S. Lopes, and A. A. Freitas, "Data mining with an ant colony optimization algorithm," *IEEE Transactions on Evolutionary Computation,* vol. 4, 2002, pp. 321-332.

[5] K. M. Sim and W. H. Sun, "Ant colony optimization for routing and load-balancing: survey and new directions," *IEEE Trans. on systems man, and cybernetics - part A: system and humans,* vol. 33, 2003, pp. 560-572.

[6] G. Bilchev and I. C. Parmee, "The ant colony metaphor for searching continuous design spaces," *AISB Workshop on evolutionary computation*, 1995.

[7] M. Wodrich and G. Bilchev, "Corporative distributed search: the ants' way", *Control Cybernetics*, 1997, 26, 413

[8] M. Mathur, S. B. Karale, S. Priye, V. K. Jayaraman, and B. D. Kulkarni, "Ant colony approach to continuous function optimization," *Ind. Eng. Chem. Res.* 2000, pp3814-3822.

[9] N. Monmarché, G. Venturini, and M. Slimane, "On how *Pachycondyla apicalis* ants suggest a new search algorithm," *Future Generation Computer Systems*, 16:937-946, 2000.

[10] J. Dréo and P. Siarry, "Continues interacting ant colony algorithm based on based on dense heterarchy," *Future Generation Computer Systems*, 20(5):841-856, 2004.

[11] K. T. Fang and Y. Wang, *Number-Theoretic Methods in Statistics*, New York: Chapman & Hall, 1994.

[12] A. S. Hedayat, N. J. A. Solane, and John Stufken, *Orthogonal Arrays: Theory and Applications*, New York: Springer-Verlag, 1999.

[13] X. Yao, Y. Liu and G. Lin, "Evolutionary programming made faster", *IEEE Transactions on Evolutionary Computation,* vol. 8, 2004, pp. 456-470.