Open Access

Combinatorial Optimization of Multi-agent Differential Evolution Algorithm

Fahui Gu[1,2,*], Kangshun Li[1], Lei Yang[1] and Yan Chen[1]

[1]*School of Information, South China Agricultural University, Guangzhou, Guangdong 510006, China;* [2]*Department of Electronic Information Engineering, Jiangxi Applied Technology Vocational College, Ganzhou, Jiangxi 341000, China*

Abstract: Combinatorial optimization is often with the local extreme point in large numbers. It is usually discontinuous, multidimensional, non-differentiable, constraint conditions, highly nonlinear NP problem. In this paper, according to the characteristics of combinatorial optimization problem, we put forward the combination optimization of multi-agent differential evolution algorithm (COMADE) through combining the multi-agent and differential evolution algorithm, in which we designed the competition behavior and self-learning behavior of agent. Through performance testing of strong connected, weak connected and overlap connected deceptive function on the COMADE algorithm, the results show that the COMADE algorithm is effective and practical value.

Keywords: Combinatorial optimization, competition behavior, differential evolution algortithm, mulit-agent, self- learning behavior.

1. INTRODUCTION

Combinatorial optimization is often with the local extreme point in large numbers. It is usually discontinuous, multidimensional, non-differentiable, constraint conditions, highly nonlinear NP problems. The combinatorial optimization is always hot subject in the fields of science and engineering. The traveling salesman problem (TSP) is a famous problem in the combinatorial optimization. The combinatorial optimization's solution has not only great academic value but important practical value. It has many solutions [1-3], such as Ant colony Algorithm (ACA), Particle Swarm Optimization (PSO) and so on, and the method of intelligent optimization is quite efficient. In this paper, a kind of combination optimization of multi-agent differential evolution algorithm (COMADE) is proposed. Through performance testing of strong connected, weak connected and overlap connected deception function on the COMADE algorithm, the results show that the COMADE algorithm is effective and practical value.

2. DESIGN OF AGENT FOR COMBINATORIAL OPTIMIZATION

Combinatorial optimization can be described as [4]: $\left(S, f\right)$, where S is the search space, and f is the objective function: $f: S \to R$. The purpose of solving is to find $x^* \in S$ for $f(x^*) \geq f(x)$ and $\forall x \in S$. Therefore, we can use an agent to represent a state of search space.

Now we define that an agent is a candidate solution of problems to be optimized, which is expressed as a model one:

$$a = (a_1, a_2, \cdots, a_n) \in S, a_i = 0 or 1 (1 \leq i \leq n) \quad (1)$$

where n is the scale of the problem, the energy of agent is equal to the value of the objective function, that is $Energy(a) = f(a)$.

In order to calculate the energy of each agent, we put the agents into a fixed grid L expressed Fig. (**1**).

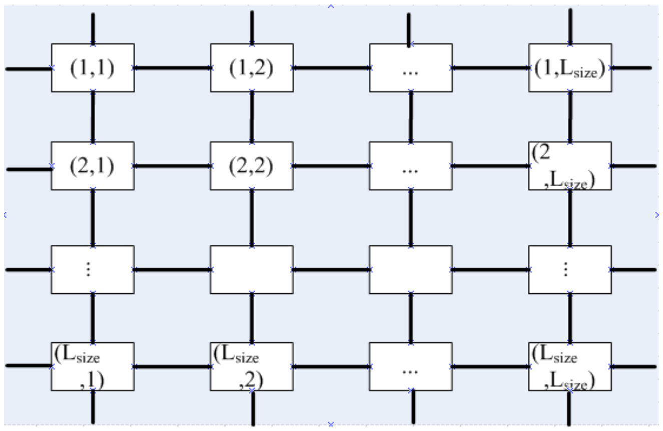

Fig. (1). Agent grid model.

The agents which can interact with $L_{i,j}$ are determined by the parameter of perception range, which can be denoted as R_s. Thus, the agents which can interact with $L_{i,j}$ are expressed as the follow model tow.

$$L_{k,l}, (i - R_s) \leq k \leq (i + R_s), (j - R_s) \leq (j + R_s) \quad (2)$$

where k and l can be denoted as follows:

$$k = \{^{k+L_{size},k<1}_{k-L_{size},k>L_{size}}, l = \{^{l+L_{size},l<1}_{l-L_{size},l>L_{size}}$$

The neighborhood of $L_{i,j}$, which is the range of interaction with $L_{i,j}$, is denote as $N_{i,j}$.

2.1. Competition Behavior

In competition behavior, the perception scope of each agent is set 1, thus, there are 8 agents in the neighborhood, which can be denoted as N^c. When the energy of $L_{i,j}$ is not less than the others of neighborhood, $L_{i,j}$ will continue to survive, otherwise it will die. The procedure of competition behavior can be described as following [4, 5]:

Where $L_{i,j} = (l_1, l_2, \cdots, l_n), a_{\max} = (a_1, a_2, \cdots, a_n) \in N^c_{i,j}$ *and* $\forall a \in N^c_{i,j}$, $Energy(a) \le Energy(a_{\max})$, if $Energy(L_{i,j}) \le Energy(a_{\max})$, the offspring $c = (c_1, c_2, \cdots, c_n)$ can be created by $a_{\max}$ in the 2 ways.

First way: when the set D represents the differential bit number of $L_{i,j}$ and $a_{\max}$, we establish the model three:

$$c_i = \{^{a_i,(i \notin D) or ((i \in D) and (Random(2)=0))}_{1-a_i, otherwise}, (1 \le i \le n) \quad (3)$$

where $Random(2)$ is used to randomly generate 0 or 1.

Second way: we build the model four:

$$c_i = \{^{a_i, Random > \frac{1}{n}}_{1-a_i, otherwise}, (1 \le i \le n) \quad (4)$$

where $Random$ is used to randomly generate the real number between 0 and 1.

The number of set D is equal to Hamming Distance between $L_{i,j}$ and $a_{\max}$. When the number is small, it shows that $L_{i,j}$ and $a_{\max}$ are similar, it is hard to generate the offspring by the first way. Therefore, we choose the way to generate the offspring c by the parameter of $D^h \in (0,1)$, if $\frac{|D|}{n} > D^h$, the first way is used, otherwise the second way.

2.2. Self-learning Behavior

Each agent can increase energy by self-learning behavior. But only when energy of an agent is not less than the any one of the learning scope, can the agent can get a chance to learn. We define the learning table as matrix $(LL)_{p\times 2}$, which has p rows and 2 columns, therefore the learning table meets the following conditions [4-6]:

$$1 \le LL_{i,j} \le n(2) \text{ and } LL_{i,1} \le LL_{i,2}, 1 \le i \le p, j = 1 or 2$$

$$\forall i \ne j, (LL_{i,1} \ne LL_{j,1}) or (LL_{i,2} \ne LL_{j,2})$$

$$p \le \frac{n(n+1)}{2}$$

Therefore, the learning table is $(LL)_{\frac{n(n+1)}{2}\times 2}$, the several lines of the table $(LL)_{\frac{n(n+1)}{2}\times 2}$ can be composed of a learning sub meter.

The learning procedure of the agent $(L_{i,j} = (l_1, l_2, ..., l_n))$ is described as the following:

First learning way:

Step 1: $q \leftarrow 1$;

Step 2: generate $(LL)^q$;

Step 3: choose 1 row from $(LL)^q$, suppose no. j row, generate a new agent $(a = (a_1, a_2, ..., a_n))$ from the following the model five:

$$a_i = \{^{l_i, (i < LL^q_{j,1}) or (i > LL^q_{j,2})}_{1-l_i, otherwise}, (1 \le i \le n) \quad (5)$$

Step 4: if $Energy(a) > Energy(L_{i,j})$, then $Learning(a) \leftarrow False$, $L_{i,j} \leftarrow a$, stop;

Step 5: delete the j row from $(LL)^q$, if $(LL)^q$ is null, then $q \leftarrow q+1$;

Step 6: if $q \le LL_w$, turn to step 2, otherwise $Learning(L_{i,j}) \leftarrow True$, stop, where LL_w is expressed the learning sub meter number of the total learning table, every sub meter table has $\frac{n(n+1)}{2} / LL_w$, denote the table as $LL^1, LL^2 \cdots, LL^{LL_w}$;

Second learning way:

Step 1: randomly generate a sequence of integers from 1 to n, denote it as $(p_1, p_2, \cdots, p_n)$, set $q \leftarrow 1$;

Step 2: generate $(LL)^q$;

Step 3: choose 1 row from $(LL)^q$, suppose no. j row, generate a new agent $(a = (a_1, a_2, ..., a_n))$ from the following the model five:

$$a_{pi} = \{^{l_{pi}, (i < LL^q_{j,1}) or (i > LL^q_{j,2})}_{1-l_{pi}, otherwise}, (1 \le i \le n) \quad (6)$$

Step 4: if $Energy(a) > Energy(L_{i,j})$, then $Learning(a) \leftarrow False$, $L_{i,j} \leftarrow a$, stop;

Step 5: delete the j row from $(LL)^q$, if $(LL)^q$ is null, then $q \leftarrow q+1$;

Table 1. Comparison of the average function evaluation times of the COMADE test on strong connected function for 50 times independent and other algorithm.

Function	Dimension	COMADE	Paper [4]	Paper [5]
f_1	n=30	796	842	8500
	n=60	3679	3817	27300
	n=90	9023	9790	57000
f_2	n=30	805	869	14300
	n=60	3681	4088	41250
	n=90	8367	8956	75450

Step 6: if $q \le LL_w$, turn to step 2, otherwise $Learning(L_{i,j}) \leftarrow True$, stop.

Generally speaking, it is good to take the first learning way. When it is failure to learn in the first way and $Learning(L_{i,j}) \leftarrow True$, we will take the second way to learn.

3. DESIGN OF COMADE

The algorithm of COMADE is described as the following [4-7]:

Step 1: initialize population L^0, randomly generate $L_{size} \times L_{size}$ agents, set $Learning(L_{i,j}) \leftarrow False$, where $i, j = 1, 2, \cdots, L_{size}$;

Step 2: evaluate Energy of every agent in L^t by the competition behavior method, if $\forall a \in N^c_{i,j}, Energy(a) \le Energy(L^{'}_{i,j})$, $L^{t+1/2}_{i,j} \leftarrow L^{'}_{i,j}$, otherwise, generate a new agent c by D^h, $Learning(c) \leftarrow False$, $L^{t+1/2}_{i,j} \leftarrow c$;

Step 3: evaluate Energy of every agent in $L^{t+1/2}$ by the self-learning behavior method, if $\forall a \in N^1_{i,j}, Energy(a) \le Energy(L^{t+1/2}_{i,j})$ and $Learning(L^{t+1/2}_{i,j}) = False$, take the first self-learning behavior method for $L^{t+1/2}_{i,j}$ to learn, if $\forall a \in N^1_{i,j}, Energy(a) \le Energy(L^{t+1/2}_{i,j})$ and $Learning(L^{t+1/2}_{i,j}) = True$, take the second self-learning behavior method for $L^{t+1/2}_{i,j}$ to learn, then $L^{t+1}_{i,j} \leftarrow L^{t+1/2}_{i,j}$;

Step 4: differential and crossover operation for each agent of $L^{t+1}_{i,j}$, generate L^{t+1};

Step 5: evaluate Energy of every agent in L^{t+1}, if $Energy(L^{t+1}_{i,j}) > Energy(a^t_{\max})$, then $a^{t+1}_{\max} = L^{t+1}_{i.j}$, otherwise, $a^{t+1}_{\max} = a^t_{\max}$;

Step 6: if meet the termination condition, exit, otherwise, t=t+1, turn to step 2.

4. SIMULATION EXPERIMENT

There are many practical problems of combinatorial optimization. In order to test the algorithm performance of COMADE, we choose the deception function to test [4, 7-12].

4.1. Experiment of Strong Connected Function

We use the following tow strong connected functions to test COMADE:

$$f_1(a) = \sum_{i=1}^{n/3} f_{deceptive3}(a_{3i-2}, a_{3i-1}, a_{3i})$$

$$f_2(a) = \sum_{i=1}^{n/5} f_{trap5}(a_{5i-4}, a_{5i-3}, a_{5i-2}, a_{5i-1}, a_{5i})$$

From the above Table **1**, we can know that the calculation amount of COMADE is about 10% of the others, the performance of COMADE is very good.

4.2. Experiment of Weak Connected Function

We use the following tow weak connected functions to test COMADE:

$$f_3(a) = \sum_{i=1}^{n/3} f_{deceptive3}(a_i, a_{i+n/2}, a_{i+2n/3})$$

$$f_4(a) = \sum_{i=1}^{n/6} f_{bipolar6}(a_i, a_{i+n/6}, a_{i+2n/6}, a_{i+3n/6}, a_{i+4n/6}, a_{i+5n/6})$$

From the above Table **2**, we can know that it is harder to solve the weak connected deceptive function than to solve the strong connected deceptive function, but we can see that it is only millions of evaluation to solve the 90 dimensional weak connected cheat function through the COMADE algorithm

4.3. Experiment of Overlap Connected Function

We use the following tow overlap connected functions to test COMADE:

Table 2. Comparison of performance based on COMADE through strong connected and weak connected function testing.

Function	Average Times of Function Evaluation		Ratio	Index Number
f_1	n=30	796	0.33	$O(n^{2.26})$
	n=60	3679		
	n=90	9023		
	n=210	65301		
f_2	n=30	805	0.40	$O(n^{2.25})$
	n=60	3681		
	n=90	8367		
	n=210	71642		
f_3	n=30	65312	6.52	$O(n^{2.94})$
	n=60	900218		
	n=90	2325892		
	n=210	42865421		
f_4	n=30	59315	0.028	$O(n^{4.06})$
	n=60	1136828		
	n=90	7569132		
	n=210	254689537		

Table 3. Comparison of performance based on COMADE through overlap connected and weak connected function testing.

Function	Average Times of Function Evaluation		Ratio	Index Number
f_1	n=30	796	0.33	$O(n^{2.26})$
	n=60	3679		
	n=90	9023		
	n=510	478941		
	n=810	1253853		
	n=990	2917483		
f_2	n=30	805	0.40	$O(n^{2.25})$
	n=60	3681		
	n=90	8367		
	n=510	510348		
	n=810	1349204		
	n=990	2184380		
f_5	n=30	783	0.25	$O(n^{2.26})$
	n=60	3587		
	n=90	7123		
	n=510	319318		
	n=810	1019287		
	n=990	1457625		

Table 3. Contd…….

Function	Average Times of Function Evaluation		Ratio	Index Number
f_6	n=30	978	0.25	$O(n^{2.29})$
	n=60	3754		
	n=90	10356		
	n=510	456872		
	n=810	1236863		
	n=990	2001358		

$$f_5(a)=\sum_{i=1}^{n-2} f_{deceptive3}(a_i, a_{i+1}, a_{i+2})$$

$$f_6(a)=\sum_{i=1}^{\frac{n-1}{4}} f_{trap5}(a_{4i-3}, a_{4i-2}, a_{4i-1}, a_{4i}, a_{4i+1})$$

From the above Table **3**, we can know that it is only millions of evaluation to solve the 900 overlap connected cheat function through the COMADE algorithm.

CONCLUSION

From the Tables **1-3**, we can see that the COMADE algorithm has good performance, especially for solving large-scale complex combinatorial optimization problem.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

This work is supported by the Education the Education Science and technology fund of the Education Department of Jiangxi, China (No. GJJ14807), and the Fund of Natural Science Foundation of Guangdong Province, China (No. 2014A030313454).

REFERENCES

[1] C. Lin, A. Qing, and Q. Feng, "Synthesis of unequally spaced antenna arrays by using differential evolution," *IEEE Transactions on Antennas and Propagation*, vol. 58, no. 8, pp. 2553-2561, 2010.

[2] F. Peng, K. Tang, G. Chen, and X. Yao, "Population-based algorithm portfolios for numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 5, pp. 782-800, 2010.

[3] K. Tang, Y. Mei, and X. Yao, "Memetic algorithm with extended neighborhood search for capacitated arc routing problems," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1151-1166, 2009.

[4] Z. Wei-Cai, J. Liu, F. Liu, and L. Jiao, "Combinatorial optimization using multi-agent evolutionary algorithm", *Chinese Journal of Computers* vol. 26, pp. 1341-1353, 2004.

[5] M. Pelikan, D. E. Goldberg, "*BOA*: the Bayesian optimization algorithm," IIIiGAL Report No. 98013, Urbana, IL:University of IIIinois at Urbana-Champaign, IIIinois Genetic Algorithms Laboratory, 1998.

[6] K. Tang, Y. Mei, and X. Yao, "Memetic algorithm with extended neighborhood search for capacitated arc routing problems," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1151-1166, 2009.

[7] K. L. Brown, E. Nudelman, and Y. Shoham, "Empirical hardness models: Methodology and a case study on combinatorial auctions," *Journal of ACM*, vol. 56, no. 4, pp. 1-52, 2009.

[8] K. D. Jong. *Evolutionary Computation: a unified Approach*, MIT Press, Cambridge, MA 2006, pp. 85-140.

[9] M. Gagliolo and J. Schmidhuber, "Learning dynamic algorithm portfolios," *Annals of Mathematics and Artificial Intelligence*, vol. 47, no. 3-4, pp. 295-328, 2007.

[10] F. Peng, K. Tang, G. Chen, and X. Yao, "Population-based algorithm portfolios fo rnumerical optimization," *IEEE Transactions on Evolutionary Computation,* vol. 14, no. 5, pp. 782-800, 2010.

[11] K. H. Han, K. H. Park, C. H. Lee, and J. H. Kim, "Parallel quantum inspired genetic algorithm for combinatorial optimization problem," In: *Proceedings 2001 Congress on Evolutionary Computation*, Piscataway, NJ:IEEE Press, vol. 2, 2001, pp. 1422-1429.

[12] S. Deluccia, and D. H. Werner, "Nature-based design of aperiodic linear arrays with broadband elements using a combination of rapid neural-network estimation techniques and genetic algorithms," *IEEE Antennas and Propagation Magazine*, vol. 49, no. 5, pp. 13-23, 2007.

Received: September 16, 2014 Revised: December 23, 2014 Accepted: December 31, 2014