

An improved ant colony optimization algorithm with strengthened pheromone updating mechanism for constraint satisfaction problem

Qin Zhang¹ · Changsheng Zhang¹

Received: 5 November 2016 / Accepted: 13 February 2017 / Published online: 28 February 2017
© The Natural Computing Applications Forum 2017

Abstract Constraint satisfaction problem (CSP) is a fundamental problem in the field of constraint programming. To tackle this problem more efficiently, an improved ant colony optimization algorithm is proposed. In order to further improve the convergence speed under the premise of not influencing the quality of the solution, a novel strengthened pheromone updating mechanism is designed, which strengthens pheromone on the edge which had never appeared before, using the dynamic information in the process of the optimal path optimization. The improved algorithm is analyzed and tested on a set of CSP benchmark test cases. The experimental results show that the ant colony optimization algorithm with strengthened pheromone updating mechanism performs better than the compared algorithms both on the quality of solution obtained and on the convergence speed.

Keywords Ant colony optimization · Constraint satisfaction problem · Pheromone updating mechanism · Swarm intelligence

1 Introduction

In the field of artificial intelligence, CSP [1] is one of the key problems need to be tackled. Many practical problems can be modeled as it, such as machine vision [2, 3], scheduling, pattern recognition [4, 5] and resource allocation. This problem [19, 20] can be defined as a tripe $\langle X, D,$

$C \rangle$ such that X is a finite set of variables, $X = \langle X_1, X_2, \dots, X_n \rangle$; D is a function that combines each $X_i \in D_i$ with its domain, $D = \langle D_1, D_2, \dots, D_n \rangle$; and C is a set of constraints which restricts the set of values that the variables can simultaneously assume, $C = \langle C_1, C_2, \dots, C_n \rangle$. A constraint C_j is a relation–scope pair $\langle R_{S_j}, S_j \rangle$, where R_{S_j} is a relation of variable on $S_j = \text{scope}(C_j)$. In other words, R_i is a subset of the Cartesian product of a domain of variable S_i . Currently, the mainstream algorithms for solving it mainly include three categories, i.e., backtracking algorithm [6], pure random walk (PRW) algorithm [7] and swarm intelligence (SI) algorithm. In the group of SI algorithm, the commonly used algorithms for CSPs mainly include particle swarm optimization (PSO) algorithm [8–10], genetic algorithm (GA) [11–14], artificial bee colony (ABC) algorithm [15, 16], differential evolution (DE) algorithm [17] and ant colony optimization (ACO) algorithm. The scale of CSP is so large that computational efficiency of backtracking algorithm will be significantly reduced, and it will appear difficult to solve it in a reasonable time. However, PRW algorithm can improve the efficiency of solving the problem, but the quality of solution is unsatisfactory, which remains to be further improved.

Complete algorithms can theoretically solve all combinatorial optimization problems perfectly. These algorithms commonly solve CSPs in a complete and systematic way until either the CSP is shown no solution or a solution is found. Nevertheless, CSPs are generally NP hard. Although some test instances might be lightly solved, it will not be likely to find optimal solutions in a reasonable time for many complex large-scale test instances in practice. For this reason, incomplete algorithms are proposed to improve this situation. An incomplete algorithm cannot ensure to find a solution; neither can it determine when a problem

✉ Changsheng Zhang
zhangchangsheng@cse.neu.edu.cn

¹ College of Computer Science and Engineering, Northeastern University, Shenyang 110819, People's Republic of China

has no solution generally. However, it can quickly find approximately optimal solutions, although the optimal solution is not necessary in practice, that is, complete assignments that satisfy a considerable proportion of the constraints, which is quite practical in real life. Incomplete algorithms usually deploy methods stochastically, ant colony optimization algorithm being one of the most typical examples. ACO algorithm not only has the advantages of high speed and high accuracy, but also can obtain good near solutions which meet the requirements of the accuracy. ACO algorithm may be less efficient than the compared algorithms on some simple CSP test instances, but there will be a great advantage in the efficiency of solving a comparatively complex CSP test instances. Especially, when we do not know that the characteristics of the instance or we should choose what kind of algorithm in advance, this feature will become very meaningful. Furthermore, we can easily achieve it and can greatly save solution costs in solving large-scale CSPs.

Mizuno et al. [18] proposed elitist ants to solve CSPs and added the theory of partial blockage. A valid part of the solution was selected from the previous generation of candidate solutions, so that CSPs can be effectively solved. Solnon et al. [19] presented a novel incomplete approach, which follows the classic ACO meta-heuristic algorithm to solve static combinatorial optimization problems. In each cycle, every ant constructs a complete assignment and then pheromone trails are updated. Iterations continue until either predetermined maximum number of cycles has been reached or a solution is found. And this ACO algorithm combined with local search (LS) [26] and preprocessing step to improve performance of algorithm. Tarrant et al. [20] presented six ACO algorithms for solving CSPs. They also introduced compound decisions, cascaded decisions, common variable ordering heuristics and carried out experiments to compare them. Aratsu et al. [21] presented a novel algorithm, the limited memory Ant-Solver, which extends Ant-Solver. The ants in this algorithm have limited memory, enabling them to recall partial assignments from their previous iteration while building new ones. Gonzalez-Pardo et al. [22] proposed a novel efficient graph-based representation for solving CSPs by utilizing ACO algorithm. And they also presented a new heuristic, which has been designed to improve the current state of the art in the application of ACO algorithms. Michalis et al. [23] proposed a self-adaptive evaporation mechanism in which ants are responsible to select an approximate evaporation rate for CSPs. In the existing literature, while updating the pheromone, the proposed algorithm did not take the variation information into account in the process of optimal path optimization.

The ACO algorithm generates the iterative optimal solution so far after several iterations, and then a better

iterative optimal solution was compared with it to determine the dynamic changes of information, that is, which edges are newly explored. Furthermore, the variation information can contribute to narrow the scope of the search area and improve the search efficiency. Therefore, a novel ACO algorithm with strengthened pheromone updating mechanism is proposed, which makes full use of dynamic change information in the process of the optimal path optimization and strengthens pheromone on the edges which had never appeared before. By adding such mechanism, the convergence speed of the improved algorithm can be speeded up under the premise of not influencing the quality of the solution in solving large-scale CSPs.

This paper is organized as follows. Section 2 shows the basic ACO algorithm for solving CSPs. Section 3 describes an improved ACO algorithm with strengthened pheromone updating mechanism for CSP. Section 4 reports experimental results on CSP benchmark test cases with the other three compared algorithms. Section 5 draws conclusions and discusses future work.

2 The basic ACO algorithm for CSPs

When solving CSPs [19, 20], we commonly model them as searching for a minimum cost path in an undirected constraint graph. The graph associates each variable–value pair $\langle X_i, v \rangle$ of the CSP with a vertex. There is an edge between any pair of vertices corresponding to two different variables, which represents constraint relation between variables. All of the edges are not needed because a path through the graph that contains $\langle X_i, v \rangle$ cannot contain another label for variable x_i as this would violate the definition of an assignment. More specifically speaking, the construction graph related to a CSP (X, D, C) is the undirected graph $G = (V, E)$ and therefore as follows:

$$V = \{ \langle X_i, v \rangle \mid X_i \in X, v \in D(X_i) \} \quad (1)$$

$$E = \{ (\langle X_i, v \rangle, \langle X_j, w \rangle) \in V^2 \mid X_i \neq X_j \} \quad (2)$$

This method is consistent with the idea of using ACO algorithm to search for the shortest path. Therefore, we take advantage of this idea to find the solution path that meets all constraints in the constraint graph. And this path is the final minimum cost solution of CSP.

Ant-Solver based on Max–Min ant system (MMAS) [24, 25] has been applied to CSPs, which includes five steps: initializations, construction of assignment, updating pheromone trails, evaluation of the solution, and determining whether the termination condition is satisfied. The research of this paper is mainly focused on the section of updating pheromone trails. If you want to know other concrete operation steps, please refer to the literature [19].

Next, we will introduce the section of updating pheromone trails in detail. After each ant has constructed a complete assignment, updating pheromone trails according to pheromone updating formula: all pheromone trails are decreased uniformly, so as to simulate evaporation and allow all of the ants to forget unsatisfactory assignments, and then the best ants of the iteration deposit pheromone. Generally speaking, at the end of iteration, the quantity of pheromone on each vertex is updated as follows:

$$\tau(i) \leftarrow (1 - \rho) \cdot \tau(i) + \sum_{A_k \in bestA} \Delta\tau(A, i) \tag{3}$$

where ρ is the evaporation parameter such that $0 \leq \rho \leq 1$. $\tau(i)$ is the total amount of pheromone, and $bestA$ is the set of the best assignments constructed during the iteration. If it is not on the optimal path, only do pheromone evaporation, the increment is zero according to Formula (4). Otherwise, if it is on the optimal path, the cost function stores a certain amount of pheromone.

$$\Delta\tau(A_k, \langle x_i, v \rangle) = \begin{cases} \frac{1}{Cost(A_k)} & \text{if } \langle x_i, v \rangle \in A_k \\ 0 & \text{Otherwise} \end{cases} \tag{4}$$

The basic process of ACO algorithm for CSPs is shown in Fig. 1.

3 ACO algorithm with strengthened pheromone updating mechanism for CSPs

Through the analysis of pheromone updating mechanism of Ant-Solver, it can be found that the pheromone updating process is executed after each ant has constructed a complete assignment, which includes

Algorithm Ant-Solver

1. **input** X, D, C
 2. **begin**
 3. Set parameters and Initialize pheromone trails
 4. **repeat**
 5. **for** each ant K **do**
 6. Construct a complete assignment A_k
 7. **if** $Cost(A_k) < Cost(bestA)$ **then**
 8. $bestA \leftarrow A_k$
 9. **end if**
 10. **end for**
 11. Update pheromone on each component referenced by Eq.3 and Eq.4
 12. **until** $Cost(bestA) = 0 \vee$ maximum number of fitness evaluations reached
 13. **return** $bestA$
 14. **end**
-

Fig. 1 Pseudo-code of Ant-Solver scheme

pheromone evaporation and pheromone enhancement. Pheromone evaporation is the process that pheromone trails on each vertex or edge weakened automatically, gradually. This evaporation process is mainly used to avoid algorithm to fall into local optimum region quickly, which contributes to enlarging the search space. Pheromone enhancement process is an optional part of ACO algorithm, which is called offline updating. This updating way updates the remaining information informally after all of the m ants have visited n vertices.

ACO algorithm in the existing literature on updating pheromone just simply evaporates and enhances pheromone and do not analyze the evolution information of iterative optimal path optimization. In addition, algorithm would appear slow convergence speed in solving large-scale CSPs. Furthermore, the search space of the basic ant colony algorithm is too large, resulting in wasting of resources. Therefore, we reduce the search space through strengthened pheromone updating mechanism to improve the search efficiency. This mechanism makes full use of dynamic change information in the process of the optimal path optimization and strengthens pheromone on the edge which had never appeared before.

In this paper, we add a pheromone increment matrix on the basis of the original pheromone matrix. The improved algorithm maintains a pheromone increment matrix, that is, all of the ants in the group share the pheromone increment matrix. *IOSolution* is used to represent the iterative optimal solution set. *BIOSolution* is used to represent the better iterative optimal solution set. *VInformation* is used to represent the variation information. Moreover, we add a dynamic strengthened pheromone updating mechanism on the basis of the existing pheromone updating mechanism. After several iterations the algorithm generates the iterative optimal solution so far; after this then a better iterative optimal solution was compared with it to determine the dynamic changes of information, that is, which edges are newly explored. Next, we provide the information with extra dynamic pheromone enhancement in the condition of original pheromone updating and take the increment into the pheromone increment matrix. The quantity of pheromone increment laid is inversely proportional to the better optimal path cost; therefore, the more constraints are violated, and the less pheromone is stored. So it will increase the probability subsequent ants to access the edges, to some extent, improving the search efficiency and speeding up the convergence speed of algorithm.

$$\begin{aligned} IOSolution &= \{A_k \in \{A_1, \dots, A_{nbAnts}\} \\ &= \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, \dots\} \\ &|\text{Cost}_{\min}(A_k)\} \end{aligned} \tag{5}$$

$$\begin{aligned}
 BIOSolution &= \{A_k \in \{A_1, \dots, A_{nbAnts}\} \\
 &= \{e_1, e_2, e_3, e_4, e_5, e_6, e_9, e_7, e_8, e_{10}, \dots\} \\
 &\quad |Cost_{\min}(A_k)\} \tag{6}
 \end{aligned}$$

$$VInformation = \{E_k = \{e_9, e_{10}, \dots\}\} \tag{7}$$

where A_k is the optimal solution set. e_i represents an edge of the optimal solution set. E_k is the variation edges set. For example, e_9, e_{10} may be the variation edges by comparison, which are newly explored.

The innovation of this paper is embodied in stage of pheromone updating, improving original pheromone updating formula (Eq. 3), as follows:

$$\tau(i) \leftarrow (1 - \rho)\tau(i) + \sum_{A_k \in BestOfCycle} \Delta\tau(A, i) + \Delta\tau_{ib}(A, i) \tag{8}$$

where ρ is the evaporation parameter such that $0 \leq \rho \leq 1$. $\tau(i)$ is the total amount of pheromone, and $BestOfCycle$ is the set of the best assignments constructed during the iteration. It is formally defined as shown in Eq. (9). $\Delta\tau_{ib}(A, i)$ is the additional enhanced pheromone increment for newly explored edges. If the path is better than the optimal path in the current iteration, add $\Delta\tau_{ib}(A, i)$ to pheromone updating formula; otherwise, update pheromone according to the original formula. The specific value of $\Delta\tau_{ib}(A, i)$ is shown in the following formula:

$$\Delta\tau(A_k^{ib}, \langle x_i, v \rangle) = \begin{cases} \frac{1}{Cost(A_k^{ib})} & \text{if } \langle x_i, v \rangle \in A_k^{ib} \\ 0 & \text{Otherwise} \end{cases} \tag{9}$$

$$BestOfCycle = \{A_k \in \{A_1, \dots, A_{nbAnts}\} | Cost_{\min}(A_k)\} \tag{10}$$

The concrete process of ACOU algorithm for CSP is shown in Fig. 2.

A CSP (X, D, C) , where X is the set of variables, n is the number of variables in X , and q is the number of vertices in construction graph.

$$q = \sum_{x_i \in X} |D(x_i)| \tag{11}$$

When constructing a complete assignment, calculations of pheromone values in all the transition probability require $O(n * q)$ operations. After each cycle, pheromone updating needs $O(q^2)$ pheromone evaporation and $O(n^2)$ to add pheromone concentration of the visited edge or vertex. Difference of time complexity is mainly based on that whether the constraint is global constraint or binary constraint for the operation that has nothing to do with pheromone, such as the selection of variables, the calculation of heuristic factor. The total time complexity is

$$T(n) = O(N_c * q^2 * m) \tag{12}$$

Algorithm ACOU

```

1. input  $X, D, C$ 
2. begin
3. Set parameters and initialize pheromone trails
4. repeat
5.   for each ant  $K$  do
6.     Construct a complete assignment  $A_k$ 
7.     if  $Cost(A_k) < Cost(BestOfCycle)$  then
8.        $BestOfCycle \leftarrow A_k$ 
9.     end if
10.  end for
11.  for each component  $i$  in graph do
12.     $\tau(i) \leftarrow$  update pheromone generated by Eq.8 and Eq.9
13.    If  $\tau(i) < \tau_{\min}$  then  $\tau(i) \leftarrow \tau_{\min}$ 
14.    If  $\tau(i) > \tau_{\max}$  then  $\tau(i) \leftarrow \tau_{\max}$ 
15.  end for
16. until  $Cost(BestOfCycle) = 0 \vee$  maximum number of fitness evaluations reached
17. return  $BestOfCycle$ 
18. end

```

Fig. 2 Pseudo-code of ACOU

where N_c is the number of iterations; m is the number of ant in ant colony. Thus, the time complexity will increase with increase in the size of CSP. The total space complexity is

$$S(n) = O(q^2) + O(q * m) \tag{13}$$

Time complexity and space complexity did not significantly changing, when the improved ACO algorithm is compared with the basic ACO algorithm.

4 Experimental results

4.1 Datasets of test case

In this paper, we choose Model A in random constraint network (random instance, RAND) classic model to generate test cases. The generator of Model A [27] randomly generates four groups of binary CSP test case dataset, i.e., Case1, Case2, Case3 and Case4. Each dataset contains six specific test cases, which covers the constraint of low complexity, high complexity or no solution. Any random CSP instance can usually be defined as five components (m, n, d, p_1, p_2) , where m represents the number of constraints, n represents the number of variables, d represents the domain size of each variable, p_1 is used to measure the connectivity of constraint graph, and p_2 is used to measure the tightness of constraints. For binary CSPs, m can usually be omitted; therefore, random CSP instance can usually be defined as four components $(n, d, p_1$ and $p_2)$.

The specific data of four datasets of test cases, Case1, Case2, Case3 and Case4, are shown in Table 1.

Table 1 Test cases generated by Model A

	Dataset of test case	Test case	P_2	k
Case1	(100, 4, 0.14, p_2)	Test1	0.14	0.754
		Test2	0.16	0.869
		Test3	0.18	0.992
		Test4	0.20	1.115
		Test5	0.22	1.242
		Test6	0.24	1.372
Case2	(100, 8, 0.14, p_2)	Test7	0.20	0.743
		Test8	0.22	0.828
		Test9	0.24	0.914
		Test10	0.26	1.003
		Test12	0.28	1.094
		Test11	0.30	1.188
Case3	(150, 4, 0.14, p_2)	Test13	0.10	0.792
		Test14	0.12	0.961
		Test15	0.14	1.134
		Test16	0.16	1.311
		Test17	0.18	1.493
		Test18	0.20	1.679
Case4	(150, 8, 0.14, p_2)	Test19	0.14	0.754
		Test20	0.16	0.874
		Test21	0.18	0.995
		Test22	0.20	1.119
		Test23	0.22	1.246
		Test24	0.24	1.376

The existing literature [20] provides another method to test the tightness of constraints, optimizes parameters for each random binary test case and then computes the k value [28]. Among them, the range of the k value is $[0, \infty]$. When $k < 1$, test case is in the state of less constraints, which can be solved; when $k > 1$, test case is in the state of too much constraints, which is often difficult to be solved.

4.2 Comparison with the related algorithms

This section mainly analyzes the advantages and disadvantages of four algorithms under various k values, from three aspects about comparison based on convergence analysis, the comparison of algorithms based on descriptive statistics and the comparison of algorithms based on hypothesis testing. The four algorithms are ACOU algorithm, ACOS algorithm, ACOD algorithm and PRW algorithm. The comparison based on convergence analysis shows the convergence of different algorithms with the increase in the number of iterations.

The comparison of algorithms based on descriptive statistics clearly shows the running results of each algorithm that includes the average cost, running time and success rate, which can reflect the characteristics of the algorithm intuitively. The comparison of the algorithms based on hypothesis testing more accurately analyzes the advantages and disadvantages of the four algorithms by statistical analysis on the same test case. All the experiments are carried out on the same computer (3.40 GHz CPU and 16 GB RAM), and the programming language is java. All the statistics are based on 30 independent runs.

4.2.1 Comparison based on convergence analysis

Algorithm performance can be evaluated only in convergence. Therefore, ACO algorithm with different vertex selection strategy and pheromone updating strategy is firstly analyzed and compared in convergence. The convergence curve and convergence period of the algorithm are observed, which can provide reference for the later evaluation of algorithm performance. In the two-dimensional coordinate system of the convergence curve, the abscissa is the number of fitness evaluation times, and the ordinate is the average cost of the algorithm to search the global optimal solution under the fitness evaluation times. According to the convergence speed of ACO algorithm with different pheromone update strategies, the intervals with different precision are set to maximize match the convergence characteristics of algorithm. In every periodic interval, the global optimal solution of the algorithm is recorded in real time, and the corresponding average cost value is calculated according to the cost function. Then, the convergence curve is drawn with the updated change point in the value. The variation range of the average cost value is less than 1 within 50 fitness evaluation times, which is regarded as the state of convergence.

As shown in Fig. 3, four test cases, Test4, Test9, Test15 and Test19, which are representative of CSP benchmark test cases, are selected to compare and analyze four algorithms, respectively. From the above figure, we can see that the convergence speed of ACO algorithm and PRW algorithm is different for the same test case, but all of them achieve the convergence in about 900 time's number of fitness evaluations. For the small-scale problems, Test4, Test9, the four algorithms achieve convergence around 600 times number of fitness evaluations generally. For the large-scale problems, Test15, Test19, three ACO algorithms achieve the convergence mostly about 200 times number of fitness evaluations.

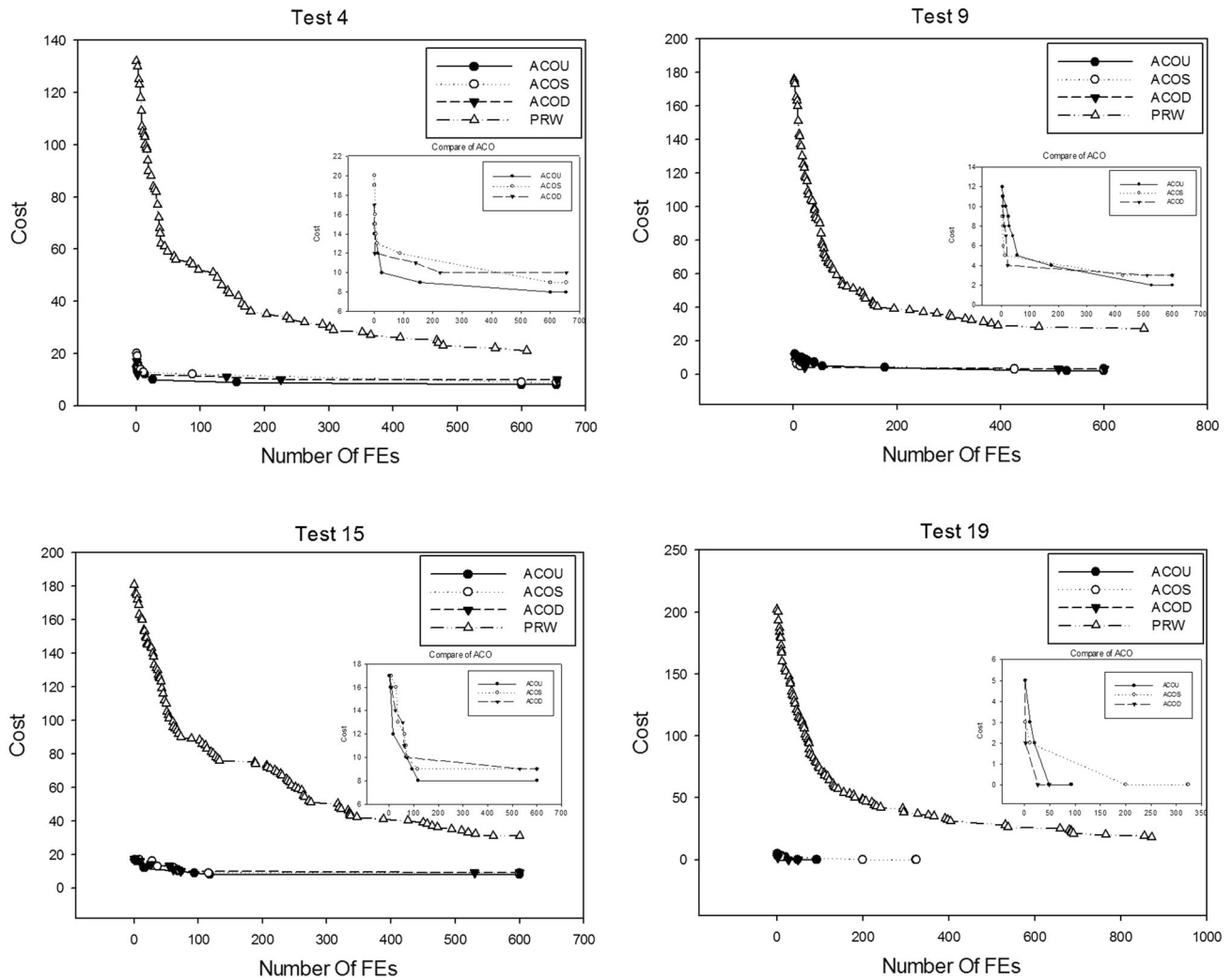


Fig. 3 Convergence graphs of four algorithms

It is not difficult to see from the figure that the ACOU algorithm with dynamic pheromone updating mechanism performs better in almost all kinds of test cases, and the convergence position is better than other algorithms. For the small-scale CSPs, Test4, Test9, the basic algorithm MMAS by limiting the upper and lower bounds, while avoiding local convergence, can quickly lock the optimal solution range. For large-scale or no solution CSPs, Test15, Test19, the ACO algorithm can rapidly find the optimal solution with its unique pheromone updating strategy and heuristic strategy, as well as the local search [26]. The convergence speed is also obviously better than the PRW algorithm. The updating strategy of ACOD algorithm and ACOS algorithm shows a good

convergence speed and is similar to ACOU algorithm in the early process. However, the convergence effect of the two algorithms is not better than ACOU algorithm in the later process.

4.2.2 Comparison of algorithms based on descriptive statistics

The four algorithms of ACOU, ACOS, ACOD and PRW run 30 times independently under 24 test cases, respectively. This paper mainly describes the advantages and disadvantages of the algorithm from the five statistic angles: the average maximum cost, the average minimum cost, the running time, the average fitness evaluation

Table 2 Comparison of algorithms based on descriptive statistics

Test cases	Test	P_2	Average maximum cost\average minimum cost\running time (ms)\average fitness evaluation number\success rate			
			ACOU	ACOS	ACOD	PRW
(100, 4, 0.14, p_2)	1	0.14	2\0\126\22\100%	3\0\293\22\100%	2\0\622\32\100%	66\5\201\486\0%
	2	0.16	2\0\189\30\100%	3\0\327\40\100%	3\0\658\36\100%	72\5\223\467\0%
	3	0.18	3\0\223\35\100%	3\0\389\45\100%	3\0\699\38\100%	75\6\219\492\0%
	4	0.20	5\0\285\225\0%	6\0\338\135\0%	8\1\458\186\70%	82\29\219\752\0%
	5	0.22	39\25\387\456\0%	42\29\446\198\0%	39\29\559\243\0%	169\28\308\668\0%
	6	0.24	39\25\476\552\0%	42\30\523\212\0%	39\27\627\286\0%	165\26\286\451\0%
(100, 8, 0.14, p_2)	7	0.20	8\0\257\56\90%	10\1\341\65\70%	8\0\322\60\80%	139\14\265\669\0%
	8	0.22	12\1\329\113\80%	10\2\450\224\0%	12\2\612\514\0%	145\18\289\623\0%
	9	0.24	15\4\556\452\0%	18\5\858\645\0%	16\4\858\685\0%	147\20\306\675\0%
	10	0.26	25\5\626\668\0%	17\5\1120\852\0%	17\5\925\814\0%	159\25\312\623\0%
	11	0.28	25\11\430\425\0%	25\14\458\157\0%	25\12\665\162\0%	185\30\349\518\0%
	12	0.30	27\13\585\553\0%	28\15\656\332\0%	29\13\1120\886\0%	189\37\329\612\0%
(150, 4, 0.14, p_2)	13	0.10	14\6\253\55\100%	15\7\256\68\60%	15\6\256\69\0%	184\23\314\683\0%
	14	0.12	15\7\282\98\90%	16\8\345\112\50%	16\7\325\156\0%	175\18\309\629\0%
	15	0.14	16\6\214\123\70%	16\6\288\156\40%	16\5\282\223\0%	176\20\311\635\0%
	16	0.16	42\27\223\188\0%	45\28\298\223\0%	45\27\311\252\0%	152\19\363\595\0%
	17	0.18	61\56\382\41\0%	64\57\466\75\0%	64\59\422\85\0%	268\75\404\610\0%
	18	0.20	62\51\387\56\0%	65\52\456\118\0%	68\57\466\96\0%	285\77\461\612\0%
(150, 8, 0.14, p_2)	19	0.14	4\1\384\126\100%	6\1\453\128\90%	5\1\453\128\90%	192\13\500\700\0%
	20	0.16	7\1\624\523\90%	8\1\695\174\380%	10\2\678\919\80%	185\14\525\722\0%
	21	0.18	10\1\689\486\40%	10\2\785\1125\45%	10\2\1125\919\40%	189\13\625\752\0%
	22	0.20	34\24\376\428\0%	34\25\485\252\0%	32\24\406\100\0%	256\45\689\859\0%
	23	0.22	45\30\262\417\0%	46\33\312\65\0%	44\31\423\228\0%	319\67\765\957\0%
	24	0.24	48\35\292\435\0%	50\40\386\285\0%	46\38\455\235\0%	322\70\768\989\0%

The bold numbers are used to highlight the difference of the experimental results in the corresponding position

number and the success rate. Specific experimental statistics are shown in Table 2.

From the above Table 2, we can intuitively see that the ACO algorithm performs better than the PRW algorithm in terms of average cost, average fitness evaluation number and success rate. With the increase of the k value, average cost and average fitness evaluation number are gradually increased; however, the success rate is gradually reduced. Although ACOU algorithm is better than ACOS algorithm in solving the problem small-scale CSPs, such as Test1, Test2, the advantage is not obvious when compared with the ACOD algorithm. For Test4 and Test9, the scatter plot for convergence analysis shows that the ACOU algorithm is not obviously different from the ACOS algorithm in distribution of average cost; however, the average maximum cost is improved in Table 2. Therefore, it is shown that ACOU algorithm performs not outstanding in solving small-scale CSPs. For large-scale or no solution CSPs, ACOU algorithm shows more convincing advantages than the other three

algorithms, ACOS algorithm, ACOD algorithm and PRW algorithm, which not only narrows the distribution range of cost value, but also reduces the average fitness evaluation number accordingly. At the same time, the running time is also improved, and to some extent, it promotes the convergence speed of the algorithm.

In summary, the solving quality and convergence speed of the three kinds of ACO algorithms are better than PRW algorithm in all the 24 test cases, whether it is a small-scale CSP or a large-scale or no solution CSP. What’s more, the advantages of ACOU algorithm are not so prominent compared with ACOS algorithm and ACOD algorithm for the small-scale CSPs; however, for large-scale CSPs, advantages of ACOU algorithm are more outstanding.

4.2.3 Comparison of algorithms based on nonparametric statistical analysis

To demonstrate whether the algorithm is applicable to all samples with a k value in the sample space, we adopt the

Table 3 Results of unilateral hypothesis testing

	ACOU	ACOS	ACOD	PRW
Test1				
ACOU	–	0.482	0.387	2.785e–12
ACOS	0.518	–	0.566	4.115e–12
ACOD	0.613	0.434	–	2.535e–12
PRW	>0.05	>0.05	>0.05	–
ACOU	–	0.503	0.497	3.685e–12
Test2				
ACOS	0.497	–	0.612	3.825e–12
ACOD	0.503	0.388	–	2.522e–12
PRW	>0.05	>0.05	>0.05	–
ACOU	–	0.456	0.567	2.835e–12
Test3				
ACOS	0.544	–	>0.05	2.421e–12
ACOD	0.433	5.413e–12	–	2.335e–12
PRW	>0.05	>0.05	>0.05	–
ACOU	–	0.523	4.456e–12	5.625e–12
Test4				
ACOS	0.476	–	7.257e–12	4.521e–12
ACOD	>0.05	>0.05	–	3.332e–12
PRW	>0.05	>0.05	>0.05	–
ACOU	–	4.715e–12	5.631e–12	5.725e–12
Test5				
ACOS	>0.05	–	0.985	4.856e–12
ACOD	>0.05	0.015	–	3.547e–12
PRW	>0.05	>0.05	>0.05	–
ACOU	–	4.625e–12	5.232e–12	3.453e–12
Test6				
ACOS	>0.05	–	>0.05	5.125e–12
ACOD	>0.05	3.143e–12	–	6.425e–12
PRW	>0.05	>0.05	>0.05	–
ACOU	–	7.254e–12	0.016	5.254e–12
Test7				
ACOS	>0.05	–	>0.05	6.223e–12
ACOD	0.84	4.453e–12	–	7.155e–12
PRW	>0.05	>0.05	>0.05	–
ACOU	–	5.114e–12	7.224e–12	4.253e–12
Test8				
ACOS	>0.05	–	0.018	4.241e–12
ACOD	>0.05	0.982	–	6.223e–12
PRW	>0.05	>0.05	>0.05	–
ACOU	–	1.224e–12	0.023	5.254e–12
Test9				
ACOS	>0.05	–	>0.05	6.223e–12
ACOD	0.977	4.651e–12	–	7.155e–12
PRW	>0.05	>0.05	>0.05	–
ACOU	–	0.982	0.983	5.254e–12
Test10				
ACOS	0.018	–	0.985	6.321e–12

Table 3 continued

	ACOU	ACOS	ACOD	PRW
ACOD	0.017	0.015	–	4.152e–12
PRW	>0.05	>0.05	>0.05	–
ACOU	–	7.254e–12	6.223e–12	7.114e–12
Test11				
ACOS	>0.05	–	>0.05	3.553e–12
ACOD	>0.05	4.453e–12	–	6.124e–12
PRW	>0.05	>0.05	>0.05	–
ACOU	–	7.254e–12	0.022	6.878e–12
Test12				
ACOS	>0.05	–	>0.05	5.665e–12
ACOD	0.878	3.323e–12	–	4.535e–12
PRW	>0.05	>0.05	>0.05	–
ACOU	–	6.985e–12	6.657e–12	7.254e–12
Test13				
ACOS	>0.05	–	5.223e–12	5.314e–12
ACOD	>0.05	>0.05	–	6.223e–12
PRW	>0.05	>0.05	>0.05	–
ACOU	–	4.525e–12	6.113e–12	6.221e–12
Test14				
ACOS	>0.05	–	5.875e–12	5.992e–12
ACOD	>0.05	>0.05	–	4.772e–12
PRW	>0.05	>0.05	>0.05	–
ACOU	–	7.125e–12	5.823e–12	4.231e–12
Test15				
ACOS	>0.05	–	4.135e–12	6.952e–12
ACOD	>0.05	>0.05	–	2.742e–12
PRW	>0.05	>0.05	>0.05	–
ACOU	–	6.124e–12	0.035	5.211e–12
Test16				
ACOS	>0.05	–	>0.05	7.052e–12
ACOD	0.965	4.441e–12	–	4.542e–12
PRW	>0.05	>0.05	>0.05	–
ACOU	–	4.234e–12	6.254e–12	5.741e–12
Test17				
ACOS	>0.05	–	6.532e–12	4.112e–12
ACOD	>0.05	>0.05	–	4.482e–12
PRW	>0.05	>0.05	>0.05	–
ACOU	–	5.233e–12	4.214e–12	5.981e–12
Test18				
ACOS	>0.05	–	3.222e–12	6.322e–12
ACOD	>0.05	>0.05	–	5.511e–12
PRW	>0.05	>0.05	>0.05	–
ACOU	–	0.025	0.752	6.514e–12
Test19				
ACOS	0.975	–	0.285	4.372e–12
ACOD	0.248	0.715	–	5.344e–12
PRW	>0.05	>0.05	>0.05	–
ACOU	–	5.223e–12	4.231e–12	6.451e–12

Table 3 continued

	ACOU	ACOS	ACOD	PRW
Test20				
ACOS	>0.05	–	2.223e–12	4.334e–12
ACOD	>0.05	>0.05	–	4.895e–12
PRW	>0.05	>0.05	>0.05	–
ACOU	–	5.113e–12	0.016	3.129e–12
Test21				
ACOS	>0.05	–	4.433e–12	6.098e–12
ACOD	0.984	>0.05	–	5.174e–12
PRW	>0.05	>0.05	>0.05	–
ACOU	–	3.263e–12	0.957	4.122e–12
Test22				
ACOS	>0.05	–	>0.05	5.198e–12
ACOD	0.043	5.114e–12	–	5.104e–12
PRW	>0.05	>0.05	>0.05	–
ACOU	–	4.223e–12	5.343e–12	5.132e–12
Test23				
ACOS	>0.05	–	>0.05	4.238e–12
ACOD	>0.05	3.223e–12	–	6.113e–12
PRW	>0.05	>0.05	>0.05	–
ACOU	–	5.211e–12	4.114e–12	2.152e–12
Test24				
ACOS	>0.05	–	>0.05	4.524e–12
ACOD	>0.05	4.243e–12	–	7.254e–12
PRW	>0.05	>0.05	>0.05	–

Fisher-independent unilateral hypothesis testing. Next, we choose the average minimum cost that four algorithms run 30 times in 24 test cases. And then the hypothesis test was performed at the confidence level of 0.05. If the result of comparison between algorithms 1 and 2 is less than 0.05, it means that the solution of algorithm 1 is better than the solution of algorithm 2. Furthermore, algorithm 1 is superior to algorithm 2 in solving this problem to some extent. Specific experimental results are shown in Table 3.

From Table 3, we can clearly see that unilateral hypothesis testing result is less than 0.05 when ACOU, ACOD and ACOS are compared with PRW algorithm in all 24 test cases, which indicates that the ACO algorithm is better than PRW algorithm in solving the CSPs. The effect of solving small-scale CSPs, ACO algorithm with strengthened pheromone updating mechanism, is not prominent. For example, ACOU algorithm is not apparently better than ACOS algorithm and ACOD algorithm in Test1 and Test2 and Test3. However, ACOU algorithm is superior to ACOS algorithm and ACOD algorithm in Test5, Test6, which is mainly because the small scale of CSPs can quickly be resolved by the processing step and local search. ACOU algorithm is significantly better than ACOU algorithm and ACOD algorithm in solving large-

scale CSPs, such as Test23 and Test24, which are shown that it is feasible to combine dynamic change information of the iterative optimal path to enhance the extra pheromone of the newly explored edge.

In summary, in most cases, ACOU algorithm has shown a more predominant performance, especially in solving large-scale CSPs. However, it may not be optimal in some extremely individual test cases. For example, in Test22, ACOU algorithm is better than ACOS algorithm and is inferior to ACOD algorithm yet. In Test10, ACOU algorithm is inferior to ACOS algorithm and ACOD algorithm. This may be caused by the stochastic characteristics of random test cases.

Combined with the above convergence analysis and comparisons of algorithms based on descriptive statistics, we can conclude that in most cases, ACOU algorithm performs better than compared algorithms.

5 Conclusion

In this paper, we firstly introduce constraint satisfaction problem (CSP) and ant colony optimization (ACO) algorithm. Next, four kinds of algorithms, ACOU algorithm,

ACOS algorithm, ACOD algorithm and PRW algorithm, are proposed to solve the CSPs. And then the Ant-Solver based on Max–Min ant system (MMAS) is described and combined with local search to solve the CSPs. We subsequently discuss the improved ACOU algorithm, ACO algorithm with strengthened pheromone updating mechanism, which is applied to solving CSP on a set of benchmark test cases. Finally, it is verified that the ACO algorithm is superior to PRW algorithm in solving the CSPs. Among the three ACO algorithms, ACOU algorithm with strengthened pheromone updating mechanism performs better than ACOS algorithm and ACOD algorithm. In the following study, when the searching of ACO algorithm is near to stagnation, the algorithm is easy to fall into the state of local optimum. Accordingly, we can reinitialize the pheromone matrix by using iterative period after stagnation to continue the search, which can enhance the algorithm's capabilities of global search.

Acknowledgements This work was supported by the National Natural Science Foundation Program of China (61572116, 61572117, 61602105), CERNET Innovation Project (NGII20160126), and the Special Fund for Fundamental Research of Central Universities of Northeastern University (N150408001, N150404009).

Compliance with ethical standards

Conflict of interest We declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work, and there is no professional or other personal interest of any nature or kind in any product, service and/or company that could be construed as influencing the position presented in or the review of the manuscript entitled, “An improved ant colony optimization algorithm with strengthened pheromone updating mechanism for constraint satisfaction problem.”

References

- Rouahi A, Salah KB, Ghédira K (2015) Belief constraint satisfaction problems. In: IEEE/ACS international conference of computer systems and applications. IEEE
- Ranft B, Stiller C (2016) The role of machine vision for intelligent vehicles. *IEEE Trans Intell Vehi* 1(1):8–19
- Ekwongmunkong W, Mittrapiyanuruk P, Kaewtrakulpong P (2016) Automated machine vision system for inspecting cutting quality of cubic zirconia. *IEEE Trans Inst Meas* 65(9):2078–2087
- Vidovic M, Hwang HJ, Amsuss S et al (2015) Improving the robustness of myoelectric pattern recognition for upper limb prostheses by covariate shift adaptation. *IEEE Trans Neural Syst Rehabil Eng* 24(9):961–970
- Adewuyi AA, Hargrove LJ, Kuiken TA (2016) An analysis of intrinsic and extrinsic hand muscle emg for improved pattern recognition control. *IEEE Trans Neural Syst Rehabil Eng Publ IEEE Eng Med Biol Soc* 24(4):1
- Zhang C, Lin Q, Gao L et al (2015) Backtracking search algorithm with three constraint handling methods for constrained optimization problems. *Expert Syst Appl* 42(21):112–116
- Xu W, Gong F (2016) Performances of pure random walk algorithms on constraint satisfaction problems with growing domains. *J Comb Optim* 32(1):51–66
- Narjess D, Sadok BA (2016) New hybrid GPU-PSO approach for solving Max-CSPs. In: Proceedings of the genetic and evolutionary computation conference companion. ACM
- Dali N, Bouamama S (2015) GPU-PSO: parallel particle swarm optimization approaches on graphical processing unit for constraint reasoning: case of Max-CSPs. *Proc Comput Sci* 60(1):1070–1080
- Breaban M, Ionita M, Croitoru C (2007) A new PSO approach to constraint satisfaction. In: IEEE congress on evolutionary computation, 2007. CEC 2007. IEEE, pp 1948–1954
- Hemert JIV (2015) Evolutionary computation and constraint satisfaction, springer handbook of computational intelligence. Springer, Berlin, pp 1271–1288
- Sharma A (2015) Analysis of evolutionary operators for ICHEA in solving constraint optimization problems. In: IEEE congress on evolutionary computation, CEC 2015. IEEE, Sendai, pp 46–53. doi:10.1109/CEC.2015.7256873
- Karim MR, Mouhoub M (2014) Coevolutionary genetic algorithm for variable ordering in CSPs. In: IEEE congress on evolutionary computation. pp 2716–2723
- Craenen BGW, Eiben AE, van Hemert JI (2003) Comparing evolutionary algorithms on binary constraint satisfaction problems. *IEEE Trans Evol Comput* 7(5):424–444
- Aratsu Y, Mizuno K, Sasaki H et al (2013) Experimental evaluation of artificial bee colony with greedy scouts for constraint satisfaction problems. In: Conference on technologies and applications of artificial intelligence. IEEE Computer Society, pp 134–139
- Aratsu Y, Mizuno K, Sasaki H et al (2013) Solving constraint satisfaction problems by artificial bee colony with greedy scouts. *Proc World Congr Eng Comput Sci* 1(1):1–6
- Yang Q (2008) A comparative study of discrete differential evolution on binary constraint satisfaction problems. In: IEEE congress on evolutionary computation, CEC 2008. IEEE, Hong Kong, pp 330–335. doi:10.1109/CEC.2008.4630818
- Mizuno K, Hayakawa D, Sasaki H et al (2011) Solving constraint satisfaction problems by ACO with cunning ants. In: International conference on technologies and applications of artificial intelligence. IEEE Computer Society, pp 155–160
- Solnon C (2002) Ants can solve constraint satisfaction problems. *IEEE Trans Evol Comput* 6(4):347–357
- Tarrant F, Bridge D (2005) When ants attack: ant algorithms for constraint satisfaction problems. *Artif Intell Rev* 24(3–4):455–476
- Goradia HJ (2013) Ants with limited memory for solving constraint satisfaction problems. In: IEEE congress on evolutionary computation, CEC 2013. IEEE, Cancun, pp 1884–1891. doi:10.1109/CEC.2013.6557789
- Gonzalez-Pardo A, Camacho D (2013) A new CSP graph-based representation for ant colony optimization. In: IEEE congress on evolutionary computation, 2013. CEC 2013. IEEE, Cancun, pp 689–696. doi:10.1109/CEC.2013.6557635
- Mavrouniotis M, Yang S (2014) Ant colony optimization with self-adaptive evaporation rate in dynamic environments. In: IEEE symposium on computational intelligence in dynamic and uncertain environments (CIDUE). pp 47–54
- Stützle T, Hoos HH (2000) MAX–MIN ant system. *Future Gener Comput Syst* 16:889–914
- Zhang Z, Feng Z (2009) A novel Max–Min ant system algorithm for traveling salesman problem. In: IEEE international conference on intelligent computing and intelligent systems. IEEE, pp 508–511

26. Lin JY, Chen YP (2011) Analysis on the collaboration between global search and local search in memetic computation. *IEEE Trans Evol Comput* 15(5):608–623
27. Macintyre E, Prosser P, Smith B et al (1998) *Random constraint satisfaction: theory meets practice*. Springer, Berlin
28. Fan Y, Shen J (2011) On the phase transitions of random k -constraint satisfaction problems. *Artif Intell* 175(3–4): 914–927