Postprint: A Multi-Population Stochastic Perturbation Ant Colony Algorithm for Distributed Constrained Optimization Problems

Authors: Shi Meifeng, Xiao Shichuan, Feng Xin

Date: 2022-05-11T10:48:42Z

Abstract

To address the issues of slow convergence and susceptibility to local optima in existing ant colony optimization-based algorithms for solving distributed constraint optimization problems, this paper proposes a Random disturbance based multi-population ant colony algorithm to solve distributed constraint optimization problems (RDMAD) to solve distributed constraint optimization problems. First, RDMAD proposes a division-of-labor cooperation mechanism that divides the population proportionally into subpopulations employing greedy search and subpopulations employing heuristic search, while constructing a hierarchical update strategy to improve algorithm convergence speed and solution quality; then, designs adaptive mutation operators and reward-penalty mechanisms for the subpopulations employing greedy search to prevent the algorithm from falling into local optima; finally, triggers a random disturbance strategy when the algorithm falls into stagnation to increase population diversity. Experimental comparisons of the optimization results between RDMAD and seven state-of-the-art incomplete algorithms on three classes of benchmark problems demonstrate that RDMAD exhibits significant advantages in solution quality and convergence speed, along with high stability.

Full Text

Preamble

Random Disturbance Based Multi-Population Ant Colony Algorithm for Distributed Constraint Optimization Problems

Shi Meifeng†, Xiao Shichuan, Feng Xin
(College of Computer Science & Engineering, Chongqing University of Technology, Chongqing 400054, China)

Abstract: The ant-based algorithm for solving distributed constraint optimization problems (ACO_DCOP) represents an excellent population-based approach for DCOPs. However, ACO_DCOP suffers from slow convergence speed and a tendency to fall into local optima. To address these limitations, this paper proposes a random disturbance based multi-population ant colony algorithm (RDMAD) for DCOPs. First, RDMAD introduces a division-of-labor mechanism that splits the population into subpopulations performing greedy search and heuristic search respectively, complemented by a hierarchical update strategy to accelerate convergence and improve solution quality. Second, an adaptive mutation operator and reward-penishment mechanism are designed for the greedy search subpopulation to prevent premature convergence. Finally, a random disturbance strategy is triggered when the algorithm stagnates, increasing population diversity. Experimental comparisons with seven state-of-the-art incomplete algorithms on three benchmark problem classes demonstrate that RDMAD achieves superior performance in solution quality and convergence speed while maintaining high stability.

Key words: distributed constraint optimization problems; ant colony algorithm; adaptive mutation operator; incomplete algorithm

0 Introduction

Multi-agent systems (MAS) constitute a fundamental area of distributed artificial intelligence [**?**]. Distributed constraint optimization problems (DCOP) serve as a core MAS framework, widely applied to model practical problems such as sensor networks [**?**] and task scheduling [**?**].

Over the past two decades, numerous algorithms have been proposed for DCOPs. Complete algorithms guarantee optimal solutions, with representative search-based methods including SyncBB [**?**], AFB [**?**], ADOPT [**?**], and BnB-ADOPT [**?**]. DPOP [**?**] exemplifies inference-based complete algorithms that employ dynamic programming, though it suffers from exponential memory consumption. MB-DPOP [**?**] was introduced to mitigate this memory issue. To further enhance MB-DPOP, Chen et al. [**?**] proposed RMB-DPOP, which reduces redundant inference and improves scalability. Rashik et al. [**?**] utilized cross-edge consistency to shorten DPOP's runtime. Since DCOPs are NP-Hard, incomplete algorithms—while not guaranteeing optimality—offer better communication and computational performance. Local search-based incomplete algorithms represent a current research focus, including DSA [**?**] and GDBA [**?**]. Frameworks such as ALS [**?**], PDS [**?**], and LSGA [**?**] have been developed to improve local search solution quality. Max-Sum [**?**] and Max-Sum_ADVP [**?**] are inference-based incomplete algorithms where agents propagate and accumulate utilities through factor graphs. Sampling-based incomplete algorithms like DUCT [**?**] solve DCOPs by sampling the search space.

Recently, a new class of incomplete algorithms utilizing populations has emerged.

Mahmud et al. [?] proposed an evolutionary optimization approach for DCOPs. Chen et al. [?] introduced ACO_DCOP, the first algorithm applying ant colony optimization to DCOPs, evolved from traditional ACO. However, ACO_DCOP employs a single population, resulting in slow convergence and susceptibility to local optima due to pheromone influence. Multi-population strategies have proven effective in other domains: Xue et al. [?] solved job-shop scheduling through cooperation between core and search ant colonies; Zhu et al. [?] applied multi-population ACO to manipulator path planning; Chen et al. [?] effectively solved TSP using a master-slave ant colony mechanism. To address ACO_DCOP's limitations, this paper proposes RDMAD, a random disturbance based multi-population ant colony algorithm for DCOPs. The main contributions are fourfold:

a) A division-of-labor mechanism where subpopulations perform greedy search and heuristic search respectively, enabling coordinated local and global exploration.

b) A random perturbation strategy triggered during stagnation that repartitions the population into three subpopulations, adding a random-value subpopulation to increase diversity and escape local optima.

c) A hierarchical update strategy with different update methods for subpopulations based on their guiding roles, improving convergence speed and solution quality.

d) Theoretical complexity analysis of RDMAD.

1.1 Distributed Constraint Optimization Problem

In this work, each agent controls one variable, making "agent" and "variable" interchangeable. Figure 1(a) shows a DCOP constraint graph where nodes represent agents and edges represent constraints. Figure 1(b) depicts the constraint matrix, where 0 and 1 are variable values and other entries indicate constraint costs for corresponding variable assignments. For example, when $x_1 = 0$ and $x_4 = 0$, the cost is 5.

A DCOP is defined as a tuple $\langle X, D, F, A, \alpha \rangle$ where $X = \{x_1, x_2, ..., x_n\}$ is a set of discrete variables, $D = \{D_1, D_2, ..., D_n\}$ is the set of domains with D_i being the domain of variable x_i, $F = \{f_1, f_2, ..., f_q\}$ is a set of constraint functions defining relationships between variables in X, and $A = \{a_1, a_2, ..., a_m\}$ is the set of agents where each agent controls one or more variables. The goal of DCOP algorithms is to find an assignment combination X^* that minimizes the global constraint cost shown in Equation (1):

$$X^* = \arg \min_{X_i \subseteq X} \sum_{i=1}^{q} f_i(X_i)$$

The constraint graph is first converted to a breadth-first search pseudo-tree [**?**] as shown in Figure 2(a). Agent message-passing order (ant traversal direction) is then constructed, where upper-layer agents have higher priority than lower-layer agents. Among same-layer neighbors, agents with more neighbors and larger domains receive higher priority; if these are equal, smaller agent IDs receive higher priority. Consequently, each agent' s neighbors are divided into high-priority neighbors H_i and low-priority neighbors L_i, with messages flowing from high-priority to low-priority agents.

Based on different agent values, multiple pheromone paths exist between each pair of agents. As shown in Figure 2(c), when agent a_i takes value d_i and agent a_j takes value d_j, the pheromone concentration on that path is $\tau_{ij}^{d_i d_j}$. In each iteration, ants depart from the highest-priority agent. After each ant obtains values, the agent sends the ant' s assignment to its low-priority neighbors. Upon receiving assignments from high-priority neighbors, agent a_i first merges the received solution sets. When a_i has received assignments from all high-priority neighbors, it selects values from each ant's domain using transition probabilities; otherwise, it waits. After assigning values to all ants, a_i sends the assignments to its low- or lowest-priority neighbors. When the lowest-priority agent receives all ant assignments, each ant has completed solution construction. The cost of each solution is calculated, with lower costs indicating better quality. The global best solution is updated accordingly, and pheromone increments for each ant are computed and sent to all agents. Upon receiving pheromone increments, agents update and evaporate pheromone concentrations on paths to their high-priority neighbors, concluding one iteration.

1.2 ACO_DCOP

Ant Colony Optimization (ACO) is a population-based metaheuristic for combinatorial optimization problems, successfully applied to traveling salesman problems and constraint satisfaction problems. Since DCOPs lack physical paths, traditional ACO cannot be directly applied. ACO_DCOP [**?**] is the only algorithm that successfully adapts ACO principles to DCOPs by using agent message-passing to simulate ant movement, pioneering swarm intelligence for DCOPs.

Using Figure 1 as an example, the pheromone path construction graph is shown in Figure 2(c), where nodes represent agents. The process involves: (1) converting the constraint graph to a BFS pseudo-tree (Figure 2(a)), (2) constructing message-passing order (Figure 2(b)), and (3) establishing pheromone paths (Figure 2(c)). Pheromone information is stored by low-priority agents.

---

2 RDMAD Algorithm

RDMAD employs ant movement between agents to construct solutions, utilizing a division-of-labor mechanism, hierarchical updates, and random perturbation strategies. By leveraging different guiding roles of subpopulations during optimization, RDMAD effectively improves convergence and solution performance.

2.1 Initialization Phase

RDMAD first converts the agent constraint graph into a BFS pseudo-tree structure, then constructs agent message-passing order to generate the final construction graph (as shown in Section 1.2, Figure 2). Pheromone information is stored by low-priority agents. After construction graph completion, parameters are initialized and each agent initializes its ant solution set as empty. The highest-priority agent then randomly assigns values to all ants and sends these assignments to its low-priority neighbors. Using the example from Section 1.2 Figure 2, where a_1 is the highest-priority node with population size 2, a_1 assigns value 1 to ant 1 and value 0 to ant 2, then sends its own value and ant assignments $\{1, 0\}$ to low-priority neighbors a_2, a_3, and a_4.

2.2 Division of Labor Mechanism

Population-based DCOP solvers have emerged only recently, evolving directly from traditional swarm intelligence algorithms that utilize single-population optimization. RDMAD enhances the existing ant transition probability approach by adding a greedy search subpopulation, employing multi-population cooperation to better balance exploitation and exploration.

When agent a_i receives assignments from high-priority neighbors, it merges all received values. Upon receiving assignments from all high-priority neighbors, a_i begins value selection for ants using different strategies per subpopulation:

1) **Subpopulation 1** performs greedy search, enhancing local exploration and convergence speed. Equation (2) selects for ant k the value that minimizes the sum of constraint costs between a_i and its neighbors:

$$d_i = \arg\min_{d \in D_i} \sum_{a_j \in H_i} \text{cost}(d, V_j^k)$$

where V_j^k is the value assigned to ant k by high-priority neighbor a_j, and $\text{cost}(d, V_j^k)$ is the constraint cost when a_i takes value d and a_j takes value V_j^k for ant k.

To prevent rapid entrapment in local optima due to greedy search, subpopulation 1 employs an adaptive mutation operator defined in Equation (4):

$$p_m = m \times \frac{\text{total\_cycle} - \text{cur\_cycle}}{\text{total\_cycle}}$$

where m is a weight controlling mutation magnitude, total_cycle is the total iteration count, and cur_cycle is the current iteration. When a random number $q \in [0,1]$ is less than p_m, ants k and k' are randomly selected and their values are swapped to generate new individuals.

2) **Subpopulation 2** retains the original transition probability approach, employing heuristic search from ACO principles for global exploration. This probability depends on pheromone and heuristic factors, combined with roulette wheel selection. Equation (5) gives the probability of assigning value d to ant k:

$$p_i^k(d) = \frac{[\theta_i^k(d)]^\alpha \cdot [\eta_i^k(d)]^\beta}{\sum_{d' \in D_i} [\theta_i^k(d')]^\alpha \cdot [\eta_i^k(d')]^\beta}$$

where α and β are weights for pheromone and heuristic factors respectively. The pheromone factor $\theta_i^k(d)$, defined in Equation (6), represents the sum of pheromone concentrations between a_i and all high-priority neighbors when a_i takes value d for ant k:

$$\theta_i^k(d) = \sum_{a_j \in H_i} \tau_{ij}^{dV_j^k}$$

The heuristic factor $\eta_i^k(d)$, defined in Equation (7), influences solution exploitation: larger constraint costs between a_i and its neighbors yield smaller heuristic values, reducing exploitation probability:

$$\eta_i^k(d) = \frac{1}{\mathrm{LC}_i^k(d)}$$

where $\mathrm{LC}_i^k(d)$ is the estimated minimum sum of constraint costs between a_i and its low-priority neighbors L_i, defined in Equation (8):

$$\mathrm{LC}_i^k(d) = \min_{d' \in D_j, a_j \in L_i} \sum_{a_j \in L_i} \mathrm{cost}(d, d')$$

This pre-estimation helps evaluate the exploitability of value d.

2.3 Random Perturbation Strategy

Population diversity affects the search space coverage. Since ACO_DCOP relies on pheromone concentrations, diversity decreases over iterations, causing stagnation in local optima. RDMAD introduces a random perturbation strategy to increase diversity. When stagnation count reaches threshold count, the

strategy triggers, repartitioning the population into three subpopulations. Subpopulations 1 and 2 maintain their original tasks, while subpopulation 3 disrupts pheromone accumulation by using completely random value selection according to Equation (9), breaking existing pheromone patterns:

$$d_i = \text{random}(D_i)$$

2.4 Hierarchical Update Strategy

A hierarchical update strategy is constructed based on each subpopulation' s distinct guiding role. When the lowest-priority agent receives assignments from all high-priority neighbors, it calculates each ant' s pheromone increment $\Delta\tau^k$ using Equation (10):

$$\Delta\tau^k = \frac{1}{\text{cost}_k - \text{best\_cost} + 1}$$

where cost_k is the cost of the complete assignment constructed by ant k, and best_cost is the global best cost value. The lowest-priority agent sends $\Delta\tau^k$, the population solution set, and the best solution to all other agents. Agents then update pheromone concentrations on paths to high-priority neighbors using Equation (11):

$$\tau_{ij}^{d_i d_j} \leftarrow \tau_{ij}^{d_i d_j} + \Delta\tau^k \quad \forall j \in H_i$$

Equation (12) defines pheromone increments for each subpopulation. Since subpopulation 1 guides the population's exploration direction, its update follows the reward-punishment mechanism in Equation (13):

$$\Delta\tau_{ij}^{d_i d_j} = \begin{cases} \Delta\tau^k / n_1 & \text{if } \Delta\tau^k \geq 0 \\ \Delta\tau^k \cdot n_1 & \text{if } \Delta\tau^k < 0 \end{cases}$$

where n_1 is subpopulation 1' s size. Positive $\Delta\tau^k$ rewards corresponding paths while negative values punish them, reducing pheromone concentration differences and preventing premature convergence. This hierarchical update strategy reflects different guidance roles, improving convergence speed and solution quality.

2.5 Pheromone Evaporation

Pheromone evaporation allows ants to forget poor paths. After updating pheromones, agents evaporate according to Equation (14):

$$\tau_{ij}^{d_i d_j} \leftarrow (1-\rho) \cdot \tau_{ij}^{d_i d_j} + \rho \cdot \tau_0$$

where ρ is the evaporation rate and τ_0 is the initial concentration, with pheromone range $[\tau_{\min}, \tau_{\max}]$. Parameters r_1 and r_2 control evaporation magnitude (default $r_1 = 1$, $r_2 = 1$). When random perturbation triggers, $r_1 = 2$ and $r_2 = 0.5$ enhance evaporation, helping escape local optima.

2.6 RDMAD Algorithm Steps

Algorithm 1 details the RDMAD implementation for each agent.

Algorithm 1: RDMAD Algorithm (for agent a_i)
Input: Initialized parameters $\alpha, \beta, \rho, \tau_0, K$, count
Output: Assignment combination X^* minimizing global constraint cost

1. For each $d_i \in D_i$, initialize $\text{est}_i(d_i)$
2. If a_i is the highest-priority node, randomly assign values to all ants and send to low-priority neighbors
3. Upon receiving value messages, merge into solution set V_{recv}
4. If a_i has received all high-priority neighbor assignments:
   - For subpopulation 3 (if active), assign random values using Equation (9)
   - For subpopulations 1-2, assign values using Equations (2), (4), and (5)
   - Send updated assignments to low- or lowest-priority neighbors
5. If a_i is lowest-priority and has received all assignments:
   - Update best_cost and corresponding best assignment X^*
   - Update stagnation count
   - Calculate each ant' s pheromone increment $\Delta\tau^k$ using Equation (10)
   - Send $\Delta\tau^k$, solution set, and best solution to all agents
6. Upon receiving pheromone information:
   - Update and evaporate pheromones using Equations (11) and (14)
   - Update pre-estimations $\text{est}_i(d_i)$ using Algorithm 2
7. If termination condition not met, begin next iteration

Algorithm 2: Update Pre-estimations (for agent a_i)
Input: Population solution set V

Output: Updated pre-estimations $\text{est}_i(d_i)$

1. For each $d_i \in D_i$, count occurrences $\text{num}(d_i)$ in V
2. Update $\text{est}_i(d_i) \leftarrow (\text{est}_i(d_i) + \text{sum\_cost}(d_i))/2$

In distributed scenarios, agents only know neighbor information and high-priority neighbor assignments based on message-passing order. These pre-estimations help a_i approximate the optimal cost sum with all neighbors, enabling heuristic factor evaluation of current values.

2.7 Complexity Analysis

RDMAD' s complexity analysis covers message count, space complexity, and time complexity, compared against ACO_DCOP. In each iteration, every agent except the lowest-priority sends value messages to low-priority neighbors, while the lowest-priority agent sends pheromone messages to all agents. Value messages contain ant solution sets with size $O(nK)$ where n is the number of agents, matching ACO_DCOP. Pheromone messages containing solution sets, increments, and best solution have size $O((K+1)n + K)$, also consistent with ACO_DCOP.

Each agent stores pheromone paths to high-priority neighbors requiring $O(|H_i| \cdot |D_i| \cdot |D_j|)$ space, identical to ACO_DCOP. Time complexity primarily involves value computation: in the worst case, when a_i assigns values to ant k, it traverses all high-priority neighbor assignments for each domain value $d_i \in D_i$, requiring $O(K \cdot |H_i| \cdot |D_i|)$ operations per value message, matching ACO_DCOP' s time complexity.

---

3 Experimental Evaluation

Experiments employ three benchmark problem classes: random DCOPs [**?**] (EXP-1, EXP-2), scale-free network problems [**?**] (EXP-3, EXP-4), and weighted graph coloring problems (EXP-5). Table 1 summarizes the configurations.

Table 1: Problem Configuration

Problem	Agent Count	Domain Size	Cost Range	Density
EXP-1	70	[1,10]	[1,100]	0.3
EXP-2	70	[1,10]	[1,100]	0.7
EXP-3	70	[1,10]	[1,100]	Scale-free
EXP-4	120	[1,10]	[1,100]	Scale-free
EXP-5	70	[1,3]	[1,100]	0.5

3.2 Parameter Analysis

Using random DCOPs (EXP-1), we analyze key parameters: pheromone factor α, heuristic factor β, evaporation rate ρ, and stagnation threshold count. The control variable method ensures fairness, with each parameter value tested over 30 independent runs. Results are shown in Figure 3.

Figure 3(a) shows α' s impact: solution quality improves as α increases, but performance degrades when $\alpha > 1$, making $\alpha = 1$ optimal. Figure 3(b) demonstrates β' s significant influence: larger β values improve solution quality and convergence until $\beta > 3$, after which quality declines. Figure 3(c) reveals that evaporation rate ρ affects performance modestly compared to α and β, with $\rho = 0.0025$ yielding the best results. Figure 3(d) shows that appropriate stagnation intervals effectively improve solution quality when triggering random perturbation.

Based on these results, RDMAD parameters are set as: $\alpha = 1$, $\beta = 3$, $\rho = 0.0025$, $\tau_0 = 3$, count = 80. Subpopulation sizes are 0.5K each before perturbation; after perturbation, they become 0.5K, 0.3K, and 0.2K for subpopulations 1, 2, and 3 respectively.

3.3 Experimental Results and Analysis

RDMAD' s robustness and optimization performance are evaluated against six state-of-the-art incomplete DCOP algorithms: PDS-DSA [**?**], GDBA [**?**], ACO_DCOP [**?**], DSAN [**?**], DSA [**?**], LSGA_DSA [**?**], and AED [**?**]. Each instance is run independently 30 times, with results averaged over 20 randomly generated instances per problem.

Table 2 presents mean costs and standard deviations across 20 instances, with Wilcoxon signed-rank test results. The "+" column indicates how many of the 20 instances RDMAD outperformed each competitor, while "-" shows the opposite. RDMAD achieves lower mean costs than all competitors on all problems except EXP-2, where it is only slightly worse than AED. Statistically, RDMAD consistently outperforms ACO_DCOP, DSA, DSAN, and GDBA across all instances, demonstrating clear advantages and high stability on other problems. P-values from the Wilcoxon test confirm RDMAD' s significant superiority in solution quality.

Table 2: Statistical Results on 20 Instances per Problem

Algorithm	EXP-1 Mean±Std	EXP-2 Mean±Std	EXP-3 Mean±Std	EXP-4 Mean±Std	EXP-5 Mean±Std
RDMAD	5340±79.7	55439±210.6	3648±81.9	12334±92.0	279±29.3
ACO_DCOP	5575±82.0	56212±205.8	3816±69.3	12684±126.5	342±36.4
GDBA	5463±72.6	55400±225.0	3853±61.8	12424±101.9	343±31.9
AED	5991±98.4	56524±188.3	4360±76.9	13160±139.2	747±45.7

Algorithm	EXP-1 Mean±Std	EXP-2 Mean±Std	EXP-3 Mean±Std	EXP-4 Mean±Std	EXP-5 Mean±Std
LSGA-DSA	5431±71.0	55456±211.5	3845±77.1	12413±90.3	396±31.5
PDS-DSA	5411±82.2	56057±199.7	3804±80.4	12589±106.3	372±31.7
DSA	5797±86.4	56129±149.9	4137±75.6	12935±101.1	476±41.1
DSAN	6127±81.7	57002±187.6	4408±61.9	13449±73.2	693±44.7

Figure 4 shows convergence curves across all test problems. RDMAD demonstrates excellent convergence and optimization performance on all five benchmarks. On EXP-1, RDMAD improves upon ACO_DCOP by approximately 4.2% and outperforms other algorithms by 1.3%–12.9%. DSA and DSAN show poor optimization capability due to lack of global information. While LSGA-DSA enhances DSA' s local search, its improvement is limited compared to RDMAD' s superior convergence. Both ACO_DCOP and AED use population-based optimization, but RDMAD exhibits better convergence and quality.

On EXP-2, high problem density increases inter-agent constraints, affecting all algorithms' performance, yet RDMAD maintains strong optimization, improving upon ACO_DCOP by 1.4% and achieving comparable quality to LSGA-DSA and AED with faster convergence (1.1%–2.7% improvement over others). On EXP-3 and EXP-4, RDMAD outperforms ACO_DCOP by 4.4% and 2.8% respectively, and surpasses others by 4.1%–17.2% and 0.6%–8.3%. RDMAD shows particularly strong performance on scale-free networks (EXP-3, EXP-4), indicating significant advantages for structured problems. On EXP-5, RDMAD improves upon ACO_DCOP by 18.4% and other algorithms by 18.7%–62.7%.

4 Conclusion

Effectively utilizing swarm intelligence for DCOPs represents a novel approach to enhancing algorithm performance. This paper proposes RDMAD, a random disturbance based multi-population ant colony algorithm that leverages population characteristics through division-of-labor cooperation and hierarchical updates to balance exploration and exploitation, improving convergence speed and solution quality. The random perturbation strategy increases population diversity to avoid local optima. Comparisons with ACO_DCOP and six other state-of-the-art incomplete algorithms on three benchmark classes demonstrate RDMAD' s significant advantages in solution quality, convergence speed, and stability. Future work will incorporate agent local convergence state evaluation, such as information entropy metrics, to complement the random perturbation mechanism.

References

[14] Zivan R. Anytime local search for distributed constraint optimization [C]. International Joint Conference on Autonomous Agents and Multiagent Systems. DBLP, 2008: 1449-1452.

[15] Chen Z, Yu Z, He J, et al. A partial decision scheme for local search algorithms for distributed constraint optimization problems [C]. In Proceedings of the 16th international conference on autonomous agents and multiagent systems, 2017: 187-194.

[16] Chen Z, Liu L, He J. et al. A genetic algorithm based framework for local search algorithms for distributed constraint optimization problems [J]. Auton Agent Multi-Agent Syst, 2020, 34: 41.

[17] Farinelli A, Rogers A, Petcu A, et al. Decentralised coordination of low-power embedded devices using the max-sum algorithm [C]. In Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Estoril, Portugal, 2008: 639-646.

[18] Zivan R, Peled H. Max/min-sum distributed constraint optimization through value propagation on an alternating DAG [C]. International Conference on Autonomous Agents and Multiagent Systems. 2012: 265-272.

[19] Ottens B, Dimitrakakis C, Faltings B. Duct: An upper confidence bound approach to distributed constraint optimization problems [C]. In Proceedings of the 26th conference on Artificial Intelligence (AAAI), Toronto, Canada, 2012: 528-533.

[20] Mahmud S, Choudhury M, Khan M M, et al. AED: An anytime evolutionary DCOP algorithm [C]. In Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, 2020.

[21] Chen Z, Wu T, Deng Y, et al. An ant-based algorithm to solve distributed constraint optimization problems [C]. In Proc. of the 32th AAAI conference on Artificial Intelligence, 2018: 4654-4661.

[22] Xue Hongquan, Wei Shengmin, Zhang Peng, et al. Flexible job-shop scheduling based on multiple ant colony algorithm [J]. Computer Engineering and Applications, 2013, 49 (24): 243-248.

[23] Zhu Youtao, He Zhiqin, Shi Wenye. Design and Application of Multi-colony Ant Algorithm in Path Planning of Manipulator [J]. Journal of Mechanical Transmission, 2021, 4: 160-165.

[24] Chen Jia, You Xiaoming, Liu Sheng, et al. Entropy-game based multi-population ant colony optimization [J]. Computer Engineering and Applications, 2019, 55 (16): 170-178.

[25] Zivan R, Okamoto S, Peled H. Explorative anytime local search for distributed constraint optimization [J]. Artificial Intelligence, 2014, 212: 1-26.

[26] Ziyu Chen, Zhen He, Chen He. An improved DPOP algorithm based on breadth first search pseudo-tree for distributed constraint optimization [J]. Applied Intelligence, 2017, 47: 607-623.

[27] Zivan R, Okamoto S, Peled H. Explorative anytime local search for distributed constraint optimization [J]. Artificial Intelligence, 2014, 212: 1-26.

[28] Barabási, A-L, Albert R. Emergence of scaling in random networks [J]. Science, 1999, 286: 509-512.

[29] Arshad M, Silaghi M C. Distributed simulated annealing [J]. Distributed Constraint Problem Solving and Reasoning in Multi-Agent Systems, 2004, 112.

Note: Figure translations are in progress. See original paper for figures.