Academic Journal of Science and Technology
ISSN: 2771-3032 | Vol. 15, No. 1, 2025

An Escape from Local Optima Algorithm Based on Historical Feature Distribution for Solving Continuous Distributed Constraint Optimization Problems

Meifeng Shi[1], Junnan She[2] and Tao Lo[3]

[1]Department of Computer Science and Engineering, Chongqing University of Technology, Chongqing 400054, China
[2]Department of Computer Science and Engineering, Chongqing University of Technology, Chongqing 400054, China
[3]School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu 611731, China

Abstract: To address the issue that existing solution algorithms for Continuous Distributed Constraint Optimization Problems (C-DCOPs) are prone to getting trapped in local optima and failing to escape, this paper proposes a C-DCOP algorithm based on the historical feature distribution of agents (HFDA). In HFDA, agents' historical values are utilized to detect when the algorithm falls into a local optimum. Upon detection, the historical features of neighboring agents are constructed based on constraint functions. By leveraging the distribution of these features, a probabilistic constraint selection mechanism is introduced to explore new solutions that cannot be obtained under full constraints, thereby guiding the algorithm towards optimization and overcoming local optima or stagnation. The probabilistic neglect of neighbors based on their feature distribution increases the chances of finding better solutions while effectively suppressing noise. Comparative experiments on two benchmark problems demonstrate that HFDA achieves faster convergence and higher solution quality compared to state-of-the-art C-DCOP algorithms.

Keywords: Multi-Agent system; continuous distributed constraint optimization problems; neighbor feature distribution; neglect strategy.

1. Introduction

Multi-Agent Systems (MAS)[1] represent a significant branch of distributed artificial intelligence research. As a fundamental abstraction framework for agent collaboration, Distributed Constraint Optimization Problems (DCOPs) [2] serve as a key technique for constraint modeling and cooperative optimization in multi-agent systems. By leveraging inter-agent communication to achieve global cost optimization, DCOPs exhibit strong fault tolerance and high parallelism, making them well-suited for large-scale and complex problems that require efficient distributed computing. Currently, DCOPs have been widely applied in various domains, such as task scheduling [3] and sensor networks [4].

Traditional DCOPs assume that problem variables are discrete. However, certain real-world applications, such as the sleep scheduling of wireless sensor networks [5], are better suited for continuous-variable modeling. To overcome this limitation, Continuous Distributed Constraint Optimization Problems (C-DCOPs) [6] were introduced, extending DCOPs to handle continuous domains. Several innovative algorithms have been developed to solve C-DCOPs. For instance, Continuous Max-Sum (CMS) [7] approximates complex constraint utility functions using piecewise linear functions. However, CMS suffers from practical limitations due to its restrictive applicability. To address these shortcomings, the Hybrid CMS (HCMS) [8] algorithm was proposed, integrating the discrete Max-Sum framework with continuous nonlinear optimization techniques. Additionally, algorithms such as Exact Function DPOP (EF-DPOP) and Approximate Function DPOP (AF-DPOP) [9] have been introduced for solving C-DCOPs. However, they still encounter challenges related to exponential memory and computational complexity.

To mitigate memory and computational overhead, Choudhury et al [10] proposed the F-DCOP algorithm based on Particle Swarm Optimization (PFD), a distributed variant of PSO. However, due to its lack of population diversity, PFD suffers from poor search capability and is prone to local optima. To overcome these limitations, Shi et al[11] introduced an improved version, PFD-LD, which incorporates local decision-making to reduce dependency on root agents. The Continuous Cooperative Constraint Approximation (C-CoCoA) algorithm [12] employs continuous nonlinear optimization techniques to enable rapid convergence and lower computational costs. However, C-CoCoA is highly sensitive to initialization parameters, resulting in unstable performance and difficulty in handling large-scale problems. Yu et al[13] proposed the Partial Decision Scheme (PDS), which improves local agent utility by selectively ignoring certain neighbors. Nevertheless, in large-scale and high-density settings, PDS's single-agent neglect strategy weakens its effectiveness, limiting further optimization.

Given the aforementioned challenges, getting trapped in local optima is a common and often unavoidable issue across most algorithms. Therefore, designing an algorithm that can escape local optima and explore new solution spaces remains an important research challenge.

To address the optimization difficulties of C-DCOPs in large-scale and high-density configurations, this paper proposes the Historical Feature Distribution-based C-DCOP Algorithm (HFDA). When encountering local optima or stagnation, HFDA utilizes agents' historical values and constraint relationships with neighbors to construct influence distributions. Based on these distributions, selective neighbor value neglect is applied to expand the solution space exploration, thereby enhancing the algorithm's optimization performance.

2. Backgrounds

2.1. Continuous Distributed Constraint Problems

A Continuous Distributed Constraint Optimization Problems (C-DCOPs) are represented as a quintuple $< A, X, D, F, \alpha >$, where:

$A = \{a_1, \ldots, a_n\}$ is the set of agents.

$X = \{x_1, \ldots, x_n\}$ is the set of variables, each assigned by its corresponding agent. Each agent is responsible for assigning values to one or more variables.

$D = \{d_1, \ldots, d_n\}$ is the set of finite continuous domains, where d_i represents the domain of variable x_i, with its lower and upper bounds being $[LB_i, UB_i]$.

$F = \{f_1, \ldots, f_l\}$ is a set of cost functions, where each function $f \in F$ is defined over a subset of variables $X^f \subseteq X$. It specifies the cost for every possible value assignment of X^f, i.e., $f : \prod_{x_i \in X} D_i \to R$. In this paper, for simplicity, only binary constraints are considered, meaning $\left|X^f\right| = 2$.

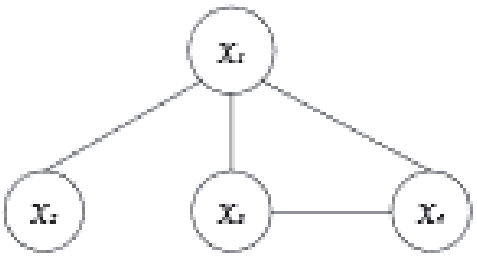
α is a mapping function that associates variables with agents, representing the relationship between a continuous variable x_i and an agent a_i. For ease of understanding, agents and variables can be considered equivalent concepts and are used interchangeably in this paper.

Let X^* be an assignment set of the C-DCOP solution that minimizes the sum of all constraint functions, which can be expressed as follows:

$$X^* = \arg\min_{d_i \in D_i, d_j \in D_j} \sum_{f_{ij} \in F} f_{ij}(x_i = d_i, x_j = d_j) \quad (1)$$

2.2. An example of C-DCOP.

Figure 1 below presents a simple example of a C-DCOP. In Figure 1(a), a constraint graph containing four variables is shown, where each variable is controlled by an agent. The edges in the graph represent constraint cost functions, which are defined as shown in Figure 1(b). Notably, the domain of variable x_i is $[-10,10]$.



a Constraint Graph

$$\forall x_1 \in X : D_i = [-10,10]$$
$$f(x_1, x_2) = \ln|x_1 x_2| + (x_1 + x_2)^2 - x_1$$
$$f(x_1, x_3) = 2x_1^3 + 2x_3 - \sin(\pi x_1) + \sin(\pi x_3)$$
$$f(x_1, x_4) = 6x_4^4 - x_1 e^{\sqrt{x_1^3 + 3x_4^4}}$$
$$f(x_3, x_4) = 7x_3^5 - 3x_4^3 + 4x_3^3 x_4^2 - x_3^2$$

b Constraint Function

Figure 1. Example of C-DCOP

3. An Escape from Local Optima Algorithm Based on Historical Feature Distribution

3.1. Motivation

HFDA divides the entire optimization process into several ignored periods, with the division based on the start and end of local optima. Considering that the values of neighbors and the constraint functions play a decisive role in the current solution space of the agent, it is decided that when an agent gets stuck in a local optimum, an integral form is used to represent the magnitude of influence based on the neighbors' values and the constraint functions. The agent can ignore the values of poorer neighbors according to a specific ignoring strategy, and further explore a new solution space. Based on this idea, the detailed content of the algorithm are introduced as the following section .

3.2. Steps of The HFDA Algorithm

Alogorithm HFDA

1. Initialization parameters $N, start, global\,\mathrm{Cost}, k, P, zThreshold$ i:
2. Initialization agents
3. Before reaching the termination ($iteration <= MaxIteration$):
3. **For** a_1 to a_n **do**
4. Agent calculates localcost:
5. **if** Neighbor's state is in the ignore state **then**
6. Reject its value message
7. Calculate local cost
8. Send value message to every neighbors $N_j \in N$
9. Update every neighbors N_j to non-ignore state
10. **If** Neighbors being ignored in current optimization and in a non-convergent phase **then**
11. Agents give up anytime property, adopt the current solution space
12. Update $global\,\mathrm{Cost}$, cache the current value of the agent x_i
13. **If** Being local optimum or non-free optimization phase $iteration < k$ **then**

14. **For** t from $start$ to $iteration$ **do**
15. Calculate the instantaneous integral influence value $I(x_i \mid x_j)^t$ and phase variation coefficient of neighbor n_j.
16. Normalize σ_{s_k} to the final influence value e_i^j using the Softmax function.
17. Mark the state of neighbors with stronger influence as the ignore state based on the cumulative probability method.
18. $start \leftarrow iteration$
19. $iteration++$
20. $iteration++$

Due to the explosion algorithm generating multiple individuals based on a single agent, it may cause significant space overhead in the context of continuous distributed constraint optimization problems. Therefore, during the value optimization phase, HFDA only uses a simple random value generation strategy to construct new values. To more prominently display the effect of the influence strategy, the algorithm increases the number of sampling points in the value domain space during each value generation. It is worth noting that increasing the number of sampling points does not affect the final solution quality of the algorithm but only accelerates the speed at which the algorithm enters a local optimum.

In step 10, the algorithm calculates the difference between the global cost average of the most recent iterations and the most recent value. If the absolute value of this difference is smaller than a sufficiently small given number $Epsilon$, it indicates that the algorithm has entered a local optimum or stagnation state. The agent constructs the integral influence $I(x_i \mid x_j)^t$ of each iteration based on historical values and the constraint function (requiring that the constraint function is bounded, continuous, and contains only a finite number of first-class discontinuities within its value domain). The specific construction method is shown as follows:

$$I(x_i \mid x_j)^t = \int_{LB}^{UB} f(x_i^t, x_j) d(x_j) \tag{2}$$

Where t represents any iteration from $start$ to $iteration$, and this value reflects the distribution of the neighbors' control ability combined with the continuous constraint function in each iteration, After deriving the stage influence of all neighbors in that phase, the stage influence variation coefficient C_v^{sk} is then calculated, as shown in the following formula:

$$I(x_i \mid x_j)_i^{s_k} = C_v^{s_k} = \frac{\sigma_{s_k}}{u_{s_k}} \tag{3}$$

Where s_k represents the local optimal stage of the k th iteration in the global iteration process; σ_{s_k} is the variance of the neighbor influence at this stage, and μ_{s_k} is the standard deviation. This variation coefficient can measure the influence of neighbors based on their control over the agent's local values.

Since the final form of ignoring neighbors is based on the cumulative probability form, the variation coefficient is mapped to the probability space using the Softmax function as shown below:

$$e_i^j = \frac{e^{I(x_i|x_j)_i^{sk}}}{\sum_{n=1}^{N_i} e^{I(x_i|x_n)_i^{sk}}} \tag{4}$$

At this point, e_i^j represents the final influence of the neighbors.

After the spatial mapping using equation (4), the agent uses the cumulative probability-based ignoring method to consider the set of neighbors that should truly be ignored with the highest likelihood. The specific ignoring method is shown in the following figure:

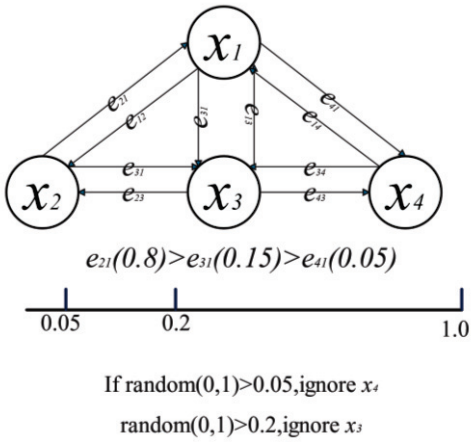


Figure 2. The probability-based ignoring strategy

In the probability state, to escape the constraints imposed by strongly controlling neighbors, the process begins with the neighbor having the smallest probability value. Select a random number r_j between 0 and 1. If $e_i^j > r_j$, the neighbor is considered ignored. Then, the next closest neighbor e_i^{j+1} is selected, and select another random number r_{j+1}. If $r_{j+1} > e_i^j + e_i^{j+1}$, the second-largest neighbor is also ignored. This process repeats iteratively until the neglect states of all neighbors are determined.

Once the neighbors are selected in the current local optimum phase, the start time of the local optimum phase is updated as $iteration$ to be used for calculating the influence in the next phase. Under the influence of the probability-based ignoring mechanism based on influence, the algorithm is highly likely to remain in a poor solution space even after entering a new solution space. Although the ignored current solution space has great potential to find better solutions, due to the anytime nature and the high complexity of the solution space in continuous distributed constraint optimization problems, the algorithm may abandon

the newly found solution space after a comparison of solution quality, thus preventing it from escaping the local optimum. Therefore, before the algorithm enters the convergence stage, it abandons the anytime property for some iterations and develops new solution spaces. In the experiments attached later in this paper, although the algorithm's curve may stay in a poor state after certain ignoring phases, it still finds better solution spaces in subsequent optimization stages than when it was trapped in the local optimum. Before the final convergence, the algorithm successfully escapes from the local optimum each time it reaches a new local optimum and ultimately achieves better solutions.

3.3. Complexity Analysis

Given that the problem topology of the algorithm is a fully connected graph with N agents, the algorithm's complexity primarily consists of space complexity and time complexity. In HFDA, agents must store historical values for each iteration before reaching a local optimum, resulting in a space complexity of $O(N)$. The total storage overhead for the entire iterative process is given by: $O(N*maxIterations)$.

During the initialization and non-neglect phases, only population and numerical information need to be transmitted in each iteration, resulting in a message size of $O(N)$. When computing local costs, an agent must iterate over all its neighbors, leading to a complexity of $O(N)$. Consequently, in a fully connected graph, the total complexity for all agents across all iterations is :

$$O(N*N*maxIterations) \tag{5}$$

The time complexity of HFDA primarily depends on the construction of influence during the neglect phase. Since agents iterate at each step when in a local optimum or stagnation state, the time cost can be expressed as:

$$O(|N|*maxIteration + |N|*|N|*maxIteration) \tag{6}$$

4. Experimental Result and Analysis

To verify the effectiveness of the HFDA algorithm, this paper conducts an experimental evaluation of PFD, C-COCOA, PFD-LD, and AMCGA against HFDA on two different benchmark problems(Random Dense Graph and Scale Free Network). Since an algorithm may exhibit varying random performance across multiple runs, this experiment analyzes the statistical characteristics of 30 runs for each algorithm on each problem model. To ensure fairness, all algorithms adopt a maximum of 500 iterations as the termination condition. The experimental configuration for these two types of problems is as follows, random graphs are classified into dense and sparse graphs, with problem densities of 0.6 and 0.1, respectively. The number of agents ranges from 50 to 100. In the scale-free network structure, m_1 =10 and m_2 =7, with the number of agents also ranging from 50 to 100. This paper uniformly uses the form of the constraint function $ax^2+bx+cxy+dy+ey^2+f$, where all coefficients of the polynomial are randomly generated from the interval $[-5,5]$. For the non-iterative algorithm C-CoCoA, according to the settings from the original paper, the number of discretization points for the value range is set to 3, the number of nonlinear optimization iterations is set to 100, and the learning rate is set to 0.01.

4.1. Convergence Experiment

To directly compare the performance differences between HFDA and the other four algorithms, Fig.1 and Fig.2 present the convergence diagrams of HFDA in Random Graph(with 100 agents, density=0.6/0.1).

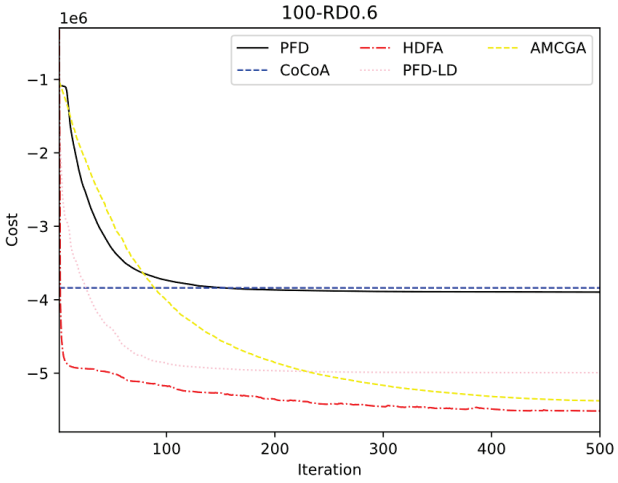


Figure 3. Convergence curve of HFDA and other algorithms on a random dense graph

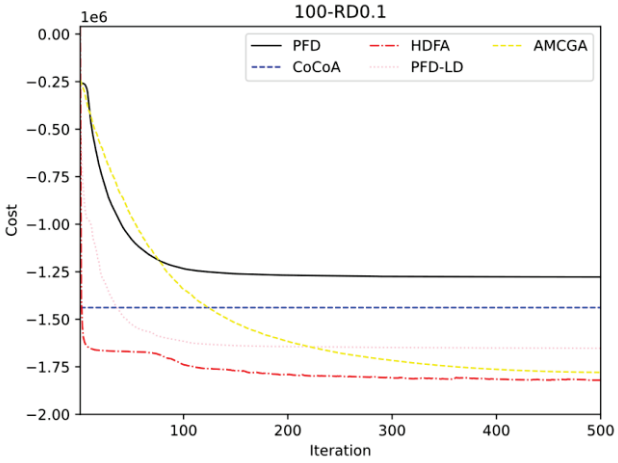


Figure 4. Convergence curve of HFDA and other algorithms on a random sparse graph

According to Fig.3 and Fig.4, as the number of iterations increases, HFDA initially falls into a local optimum. However, due to the activation of influence computation and the ignoring of neighbor values, HFDA continues exploring new solution spaces and outperforms other algorithms. AMCGA also demonstrates good convergence quality as iterations progress, but since it lacks a mechanism to escape local optima, its final convergence result two figures remain inferior to that of HFDA.

The following presents the performance comparison of HFDA in scale-free networks.

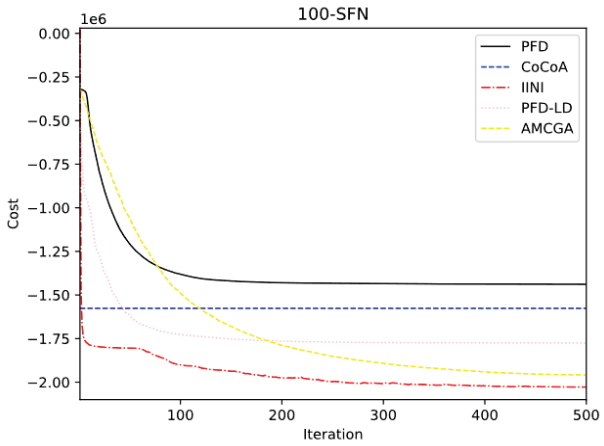


Figure 5. Convergence curve of HFDA and other algorithms on a random sparse graph

Fig.5 shows that the HFDA algorithm demonstrates superior performance compared to other algorithms when solving problems with a high degree of topological complexity. Compared with the benchmark algorithms, the HFDA algorithm, which utilizes historical local solution information, performs better. In scale-free networks, most nodes have few connections, while a few nodes have extremely high connectivity. This distribution characteristic makes scale-free networks highly resistant to random disturbances but vulnerable to targeted attacks. HFDA can significantly enhance local search capabilities for solving structured problems.

4.2. The Friedman Test

To provide a more intuitive comparison of HDFA's effectiveness, Table 2 presents the pairwise Friedman test results of HDFA compared to other algorithms. ① represents the problem type as a random dense graph. ② represents the problem type as a scall free network. $\chi^2(a)$ represents the results of the standard statistical test, the rank test, in pairwise comparison experiments. $P(a)$ represents the statistical significance level of the pairwise comparison of algorithms.

Table 1. The Friedman test on three benchmark problems

Problems	HDFA	AMCGA	PFD-LD	PFD	C-COCOA
①	1	2	3.5	3.5	5
$\chi^2(a)$	\	0.917	2.49	3.97	3.624
$P(a)$	\	0.174	0.04	0	0
②	1	2	3.82	5	5.8
$\chi^2(a)$	\	0.881	2.426	3.64	4.821
$P(a)$	\	0.215	0.005	0	0

The rank and ranking of HDFA relative to AMCGA are not significantly different from other algorithms. This is because HDFA is a phase-based algorithm that starts with a local optimum stage as its optimization cycle, rather than being an iterative optimization algorithm like others. HDFA shows good performance in average rankings across five problem sizes compared to other algorithms, indicating that it achieves results surpassing iterative optimization algorithms through phase-based optimization. Furthermore, the Friedman statistical results show that all p-values are less than 0.05, indicating statistical significance between HDFA and the four comparison algorithms, which suggests that HDFA performs well across various problem contexts in general.

At the end of the experiment, the improvement rate of HDFA compared to other algorithms is presented:

Table 2. The improvement rate of HDFA compared to other algorithms

Benchmark problems	PFD	PFD-LD	AMCGA	C-CoCoA
Random Sparse Graph	31.60%	8.06%	2.19%	20.98%
Random Dense Graph	32.44%	8.32%	2.34%	40.25%
Scale Free Networks	32.18%	8.48%	3.16%	29.35%

The experimental results show that HDFA outperforms other algorithms across different problem types. In random sparse graphs, HDFA achieves a 31.60% improvement over PFD, significantly surpassing PFD-LD (8.06%), AMCGA (2.19%), and C-CoCoA (20.98%), demonstrating its effectiveness in low-connectivity environments. In random dense graphs, HDFA surpasses PFD by 32.44% and outperforms C-CoCoA (40.25%), indicating its stability in high-connectivity graphs. In scale-free networks, HDFA shows a 32.18% improvement over PFD and a 29.35% improvement over C-CoCoA, proving its adaptability in complex topological structures. Overall, HDFA performs particularly well in dense graphs and scale-free networks, demonstrating superior optimization capabilities.

5. Conclusion

This paper provides a detailed analysis of the current research status of incomplete solving algorithms for C-DCOP. Addressing the issues of existing algorithms that often get trapped in local optima and cannot escape, this paper proposes an escape from local optima algorithm based on historical feature distribution for solving continuous distributed constraint optimization problems (HFDA). The algorithm caches the agent's historical values during the process of falling into local optima, and after completely becoming stuck in a local optimum, constructs the stage influence using the historical values and the integral form of the neighbor's constraint functions. It then uses a probabilistic ignoring mechanism to disregard neighbors, helping the agent escape from local optima dominated by strong neighbor control. This approach enables the exploration of new solution spaces. The designed probabilistic ignoring mechanism takes into account the strength of the neighbors' influence and the potential noise impact after ignoring, expanding the exploration of the solution space while minimizing algorithm degradation. Furthermore, the algorithm innovatively abandons the anytime property during the early iterations, allowing the newly generated solution space to be effectively explored, significantly liberating the solution development capability of each agent. In the theoretical analysis, this paper analyzes the complexity of each step and focuses on the time complexity during the ignoring phase.

Comprehensive experimental results show that the algorithm achieves high solution quality across various problem models, particularly for large-scale and high-density

problems. In these cases, the algorithm leverages more historical solution information, yielding better results.

Acknowledgements

Chongqing Municipal Education Commission Science and Technology Research Program Youth Project (NO. KJQN202401101), Chongqing University of Technology Graduate Innovation Engineering Project (gzlcx20233196), and the Scientific Research Foundation of Chongqing University of Technology. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

References

[1] Pujol G. Multi-agent coordination Dcops and beyond [J]. Proceedings International Joint Conference on Artificial Intelligence, 2011, 22 (3): 28-38.

[2] Leite A R, Enembreck F, Barthes J P A. Distributed constraint optimization problems: Review and perspectives[J]. Expert Systems with Applications, 2014, 41(11): 5139-5157.

[3] Enembreck F, Barthes J P A. Distributed constraint optimization with MULBS: A case study on collaborative meeting scheduling[J]. Journal of Network and Computer Applications, 2012, 35(1): 164-175.

[4] Zhang W, Wang G, Xing Z, et al. Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks[J]. Artificial Intelligence, 2005, 161(1-2): 55-87.

[5] Lesser V, Ortiz Jr C L, Tambe M. Distributed sensor networks: introduction to a multiagent perspective[M]//Distributed sensor networks: A multiagent perspective. Boston, MA: Springer US, 2003: 1-8.

[6] Stranders R, Farinelli A, Rogers A, et al. Decentralised control of continuously valued control parameters using the max-sum algorithm[J]. 2009.

[7] Farinelli A, Rogers A, Petcu A, et al. Decentralised coordination of low-power embedded devices using the max-sum algorithm[J]. 2008.

[8] Voice T, Stranders R, Rogers A, et al. A hybrid continuous max-sum algorithm for decentralised coordination[M]//ECAI 2010. IOS Press, 2010: 61-66.

[9] Hoang K D, Yeoh W, Yokoo M, et al. New algorithms for continuous distributed constraint optimization problems [C]// Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems. 2020: 502-510.

[10] Choudhury M, Mahmud S, Khan M M. A particle swarm based algorithm for functional distributed constraint optimization problems[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2020, 34(05): 7111-7118.

[11] Shi M, Liao X, Chen Y. A particle swarm with local decision algorithm for functional distributed constraint optimization problems[J]. International Journal of Pattern Recognition and Artificial Intelligence, 2022, 36(12): 2259025.

[12] Sarker A, Choudhury M, Khan M M. A local search based approach to solve continuous DCOPs[C]//Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems. 2021: 1127-1135.

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