Group Correction-based Local Disturbance Particle Swarm Optimization algorithm for solving Continuous Distributed Constraint Optimization Problems

1st Meifeng Shi
College of Computer Science and Engineering
Chongqing University of Technology
Chongqing, China
Faculty of Information Science and Electrical Engineering
Kyushu University
Fukuoka, Japan
shimf@cqut.edu.cn

2nd Haitao Xin
College of Computer Science and Engineering
Chongqing University of Technology
Chongqing, China
XinHaitao163@163.com

3rd Makoto Yokoo
Faculty of Information Science and Electrical Engineering
Kyushu University
Fukuoka, Japan
yokoo@inf.kyushu-u.ac.jp

***Abstract*—Continuous Distributed Constraint Optimization Problems (C-DCOPs) are a significant constraint handling framework to model continuous variable problems of multi-agent systems. Many excellent algorithms have been designed to solve C-DCOPs in recent decades. However, these algorithms are prone to falling into local optimum, which is a major challenge in solving C-DCOPs. This paper proposes a Group Correction-based Local Disturbance Particle Swarm Optimization algorithm named GC-LDP to improve its solution quality. In GC-LDP, we introduce two items, the average of the personal best positions and the average of the personal current positions, into the velocity update formula of traditional Particle Swarm Optimization to utilize the group knowledge to correct the exploitation direction. In addition, a local disturbance strategy is designed in GC-LDP to increase the swarm diversity by searching the nearest particle group in the solution space to enhance the algorithm's exploration ability. GC-LDP has been theoretically proven to be an anytime algorithm. Furthermore, based on the extensive experiments on four types of benchmark problems, we demonstrate that GC-LDP outperforms state-of-the-art C-DCOP algorithms in terms of convergence speed and solution quality.**

***Index Terms*—Group Knowledge, Group Correction, Local Disturbance, Particle Swarm Optimization, Continuous Distributed Constraint Optimization Problems**

I. Introduction

Distributed Constraint Optimization Problems (DCOPs) [1] are a fundamental framework for modeling cooperative multi-agent systems. Researchers have explored various DCOPs solving algorithms in the past decade, leading to the classification of these algorithms into two categories: complete algorithms and incomplete algorithms. Complete algorithms like ADOPT [2], DPOP [3], and PT-FB [4] are designed to find the global optimal solution for the given DCOPs. However, with the increase of the problem scale, complete algorithms face challenges related to exponential memory requirements and computational overhead since DCOPs are NPHard. Different from complete algorithms, incomplete algorithms such as DSA [5], MGM & MGM2 [6], Max-Sum [7], CoCoA [8], AED [9], MIF-DCOP [10], and ACO_DCOP [11], seek to quickly approximate the global optimal solution by prioritizing lower memory requirements and shorter execution time.

However, several real-world applications, such as target tracking sensor orientation [12] and sleep scheduling of wireless sensors [13], are more appropriate to be modeled with continuous variables. In order to solve the limitations, Continuous Distributed Constraint Optimization Problems (C-DCOPs) [14] were proposed to model continuous variable problems to solve the limitations of discrete variable space.

To cope with C-DCOPs, some innovative algorithms have emerged recently. Continuous Max-Sum (CMS) [7] is introduced to approximate complex constraint utility functions using piece-wise linear functions. However, CMS is limited in its real-world application due to the scarcity of scenarios where piece-wise linear functions are relevant. To overcome these limitations, a Hybrid CMS (HCMS) [15] methodology is introduced, combining the discrete Max-Sum framework with continuous non-linear optimization techniques.

This work is supported by Graduate Education High-Quality Development Action Plan of Chongqing University of Technology (No. gzlcx20233215), JST ERATO JPMJER2301, and JSPS JP21H04979.

979-8-3503-5409-6/24/$31.00 ©2024 IEEE
DOI 10.1109/CAI59869.2024.00128

Furthermore, a suite of algorithms [16], Exact Functional DPOP (EF-DPOP), Approximate Functional DPOP (AF-DPOP), and Clustering Approximate Functional DPOP (CAF-DPOP), are simultaneously proposed to solve C-DCOPs, where EF-DPOP can solve C-DCOPs by performing addition and projection operations, and AF-DPOP and CAF-DPOP will only produce approximate solutions. However, they still face challenges related to exponential memory and computation overhead.

To decrease the memory requirements and excessive computational overhead, a Particle Swarm Optimization based F-DCOP algorithm (PFD) [17], a distributed version of PSO, is proposed. However, the search ability of PFD is poor, and it is easy to fall into the local optimum due to the lack of swarm diversity. To deal with these issues, an improved PFD algorithm with Local Decision (PFD-LD) [18] is proposed to reduce the dependence of PFD on the root agent through local decisions. The Continuous Cooperative Constraint Approximation (C-CoCoA) algorithm [19] is a non-iterative algorithm and employs continuous nonlinear optimization techniques to achieve quick convergence and reduce computational overhead. However, C-CoCoA is easily affected by initialization parameters and is very difficult to solve large-scale problems. The Dual population Search Diferential Evolution Algorithm for F-DCOP (DSDE-FD) [20] is designed to balance the exploration and exploitation by dividing the population into the local search population and global search population. The solution quality of DSDE-FD is good, but the convergence speed is poor. Recently, the Distributed Bayesian (D-Bay) algorithm [21] is proposed to address C-DCOPs by employing Bayesian optimization for the adaptive sampling of variables. However, D-Bay can only solve problem instances with less than ten agents.

Based on the above analysis, falling into local optima is a common inevitable issue for most algorithms. Therefore, how to design an algorithm to avoid falling into local optimum as much as possible is a worthwhile topic and challenge. This paper proposes a Group Correction-based Local Disturbance Particle Swarm Optimization algorithm named GC-LDP to utilize the group knowledge to correct the exploitation direction. In addition, a local disturbance strategy is designed in GC-LDP to increase the swarm diversity by searching the nearest particle group in the solution space to enhance its exploration ability. Unlike the traditional swarm optimization algorithm based on purely elite strategy, GC-LDP provides a novel idea of using group information to correct the evolutionary direction, which may be a new perspective for the research of optimization algorithms.

II. Background

This section presents the definition of Continuous Distributed Constraint Optimization Problem and Particle Swarm Optimization, respectively.

A. Continuous Distributed Constraint Optimization Problem

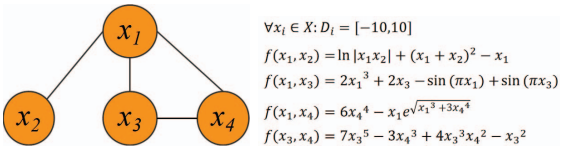
A C-DCOP can be defined as a tuple $\langle A, X, D, F\rangle$ where,

- $A = \{a_1, \ldots, a_n\}$ is a set of n agents and $X = \{x_1, \ldots, x_n\}$ is a set of n variables. We assume each agent a_i can control exactly one variable x_i. Thus, we use the terms "agent" and "variable" interchangeably.
- D is a set of continuous domains $\{D_1, D_2, \ldots, D_n\}$, where each D_i is given as a range $[LB_i, UB_i]$, with LB_i and UB_i representing the lower and upper bounds of the variable x_i.
- F is a set of cost functions $\{f_1, f_2, \ldots, f_l\}$, where each $f \in F$ is defined over a subset of variables $X^f \subseteq X$. It is defined for every possible value assignment of X^f, that is, $f : \Pi_{x_i \in X^f} D_i \rightarrow R$. In this paper, for simplicity, we only consider binary constraints, i.e., $|X^f| = 2$.

Let $\theta = (\theta_1, \ldots, \theta_n)$ denote one possible assignment of all variables, where $LB_i \leq \theta_i \leq UB_i$ holds for all $\theta_i \in \theta$. Let Θ denote the set of all possible assignments. For θ, let θ^f denote the projection of θ on the set of variables X^f. Let $\widehat{f}(\theta)$ denote $\sum_{f \in F} f(\theta^f)$. The solution of a C-DCOP is an assignment θ^* that minimizes the sum of cost functions as shown in Equation (1).

$$\theta^* = \arg\min_{\theta \in \Theta} \widehat{f}(\theta) \tag{1}$$

Figure 1 presents a simple example of a C-DCOP, where Figure 1 (a) showcases a constraint graph involving four variables, with each variable x_i under the control of agent a_i. The edges in the graph represent constraint cost functions, and their definitions are presented in Figure 1 (b). Notably, the domain D_i for variable x_i is given as [-10, 10].



(a) Constraint Graph (b) Cost Functions

Fig. 1. Example of a C-DCOP.

B. Particle Swarm Optimization

PSO [22] is a computational algorithm that optimizes a problem by having a swarm of candidate particles. In our DCOP context, each swam has its current assignment θ^j as its position. Then, at each iteration, it updates the velocity and position according to Formula (2) and (3), where v^j is the velocity, c_1 and c_2 are positive constants, r_1 and r_2 are random parameters, and w is the inertia weight. $pBest^j$ is j's personal best position, and $gBest$ is the global best position.

$$v^j \leftarrow w \cdot v^j + c_1 \cdot r_1 \cdot (pBest^j - \theta^j) + c_2 \cdot r_2 \cdot (gBest - \theta^j) \tag{2}$$

$$\theta^j \leftarrow \theta^j + v^j \tag{3}$$

Algorithm 1 gives the sketch of PSO.

Algorithm 1: The Particle Swarm Optimization

for *each Particle* $j \in K$ **do**
  Initialize velocity v^j and position θ^j
  Evaluate Particle j and set $pBest^j = \theta^j$
$gBest = \arg\min_{j \in K} \widehat{f}(pBest^j)$
while *Termination condition not met* **do**
  for $j = \in K$ **do**
    Update the velocity and position of Particle j according to Formula (2) and (3)
    Evaluate Particle j
    if $\widehat{f}(\theta^j) < \widehat{f}(pBest^j)$ **then**
      $pBest^j = \theta^j$
    if $\widehat{f}(pBest^j) < \widehat{f}(gBest)$ **then**
      $gBest = pBest^j$
  Output $gBest$

III. The Proposed GC-LDP Algorithm

The details of the proposed GC-LDP algorithm are described in this section.

In GC-LDP, two items are added to the traditional PSO velocity update formula to utilize the group knowledge to correct the exploitation direction. The innovation idea of GC-LDP is exhibited in Figure 2, where **Personal knowledge** and **Social guidance** represent the $pBest$ and $gBest$ in Equation (2), respectively. In GC-LDP, we use the average personal best position of all particles to express the **Group knowledge**, and the average current position of all particles as **Group correction** information to correct the exploitation direction.

Fig. 2. Innovation idea of GC-LDP.

It is worth mentioning that an agent only holds one dimension of information for all the particles, a particle represents a solution, as shown in Figure 3.

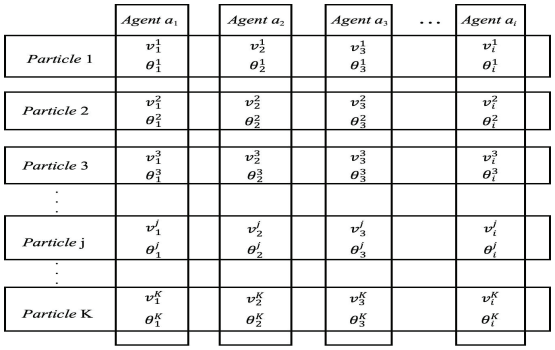


Fig. 3. Distributed population in GC-LDP.

The GC-LDP algorithm consists of five phases: **Initialization, Evaluation, Local Disturbance, Output, and Update**. The details of GC-LDP are exhibited in **Algorithm 2**.

In the **Initialization** phase, as shown in Figure 4, GC-LDP constructs a Breadth-First Search (BFS) pseudo-tree and initializes parameters and the population. Subsequently, each agent, denoted as a_i, transmits the message θ_i^j to its neighbors within N_i. Each agent initializes the velocity to 0 and the position to a random value in the domain D_i (Algorithm 2: Lines 4–6). Further, agent a_i sends θ_i^j to lower priority neighbors L_i (Algorithm 2: Line 7).

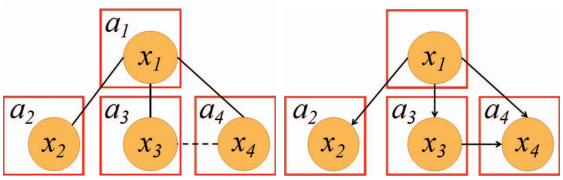


Fig. 4. Pseudo-tree construction and ordered arrangement.

In the **Evaluation** phase, agent a_i undertakes the computation of its partial fitness by processing the θ_i^j message dispatched by its higher-priority neighbors in H_i. Following this, agent a_i then forwards its calculated costs with neighbors to the corresponding higher-priority neighbors in H_i (Algorithm 2: Lines 15 and 21). Furthermore, excluding the leaf agents, each agent sends the partial fitness received from the lower-priority neighbors upwards along the BFS pseudo-tree. Finally, the partial fitness is passed to the root agent (Algorithm 2: Lines 23–28).

In the **Local Disturbance** phase, the local disturbance is carried out after agent a_i receives θ_i^j sent by its neighbors in H_i. Each agent a_i calculates the local fitness of each particle to find the particle with the smallest fitness, denoted as s. Then, as shown in Figure 5, construct the nearest particle group represented as $Group(\theta^s)$ by comparing half of the particles closest to particle s in the current particle swarm, and send the related position of particles in $Group(\theta^s)$ to the lower-priority neighbor (Algorithm 2: Lines 16-18). Finally, find the particle m that with the smallest fitness within $Group(\theta^s)$, and each

agent a_i sends the minimum cost $cost^m_{i,u}$ with its neighbor $u \in N_i$ to the higher-priority neighbor (Algorithm 2: Lines 19-21).

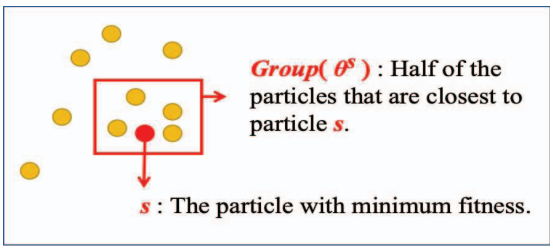


Fig. 5. An example of $Group(\theta^s)$.

In the **Output** phase, the root agent a_{root} aggregates all the fitness received from its lower-priority neighbors L_{root}. Given the hierarchical propagation of all fitness, the complete fitness $\widehat{f}(\theta)$ contains the constraint cost function of each $f \in F$. To ensure the solution quality, a comparative evaluation of the global optimal solution is undertaken with each iteration. If the current global solution outperforms the previous one, GC-LDP will output the current global solution. Otherwise, output the previous one and update the global best solution (Algorithm 2: Lines 29-41).

In the **Update** phase, after receiving $pBest^j$ and $gBest$ of the previous iteration, agent a_i updates the θ^j_i and v^j_i of each particle j.

Before updating the particles, we referred to PFD [17] to calculate the number of consecutive successes and failures according to Equations (4) and (5), denoted as s_c and f_c respectively (Algorithm 2: Line 45). Then, we can calculate the exploration diameter ρ according to Equations (6), which represents the search area of the particle.

$$s_c(t) = \begin{cases} s_c(t-1)+1 & \text{if } PBest^j(t) < gBest(t-1) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$f_c(t) = \begin{cases} f_c(t-1)+1 & \text{if } PBest^j(t) = gBest(t-1) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$\rho(t) = \begin{cases} 1 & t = 0 \\ 2 \cdot \rho(t-1) & s_c(t-1) > max_{s_c} \\ 0.5 \cdot \rho(t-1) & f_c(t-1) > max_{f_c} \\ \rho(t-1) & otherwise \end{cases} \tag{6}$$

Finally, the global optimal particle is updated by Equations (8) and (9), and other particles are updated by Equations (7) and (9) (Algorithm 2: Lines 47-50). Different from the velocity update formula of traditional PSO in Equation (2), in Equation (7), the item with parameter α is used to describe the average current position of neighboring particles of the particle j and the item with parameter β is the average personal best position of neighboring particles of the particle j. α and β are two random values in [0, 1].

$$\begin{aligned} v^j_i(t) = & w \cdot v^j_i(t-1) \\ & + r_1 \cdot c_1 \cdot (pBest^j(t) - \theta^j_i(t-1)) \\ & + r_2 \cdot c_2 \cdot (gBest^j(t-1) - \theta^j_i(t-1)) \\ & + \alpha \cdot (Average(\theta^j_i(t-1)) - \theta^j_i(t-1)) \\ & + \beta \cdot (Average(pBest^j(t-1)) - \theta^j_i(t-1)) \end{aligned} \tag{7}$$

$$v^j_i(t) = -\theta^j_i(t-1) + gBest^j(t-1) + \rho * (1 - 2r_2) \tag{8}$$

$$\theta^j_i(t) = \theta^j_i(t-1) + v^j_i(t) \tag{9}$$

IV. Theoretical Analysis

The theoretical analysis of GC-LDP is presented in this section. Suppose **Messages** are steps for communication, which is the time needed for neighboring agents to receive messages sent by the current agent. h is the depth of the BFS pseudo-tree.

Lemma 1: At Messages $t+h+4$, root agent a_{root} acquires knowledge of the $\widehat{f}(\theta^m)$ up to Messages t.

Proof: Each agent a_i requires three Messages steps to calculate their $Group(\theta^s)$. Subsequently, lower-priority agents require one Messages step to acquire $Group(\theta^s)$ for computing $\widehat{f}(\theta^m)$. Simultaneously, the root agent $root$ must await at most h iterations since the depth of the BFS pseudo-tree is h. Hence, it can be inferred that at iteration $t+h+4$, the root agent $root$ can calculate the fitness for each particle up to iteration t.

Lemma 2: At Messages $t+2h+4$, each agent becomes aware of the $\widehat{f}(\theta^m)$ up to Messages t.

Proof: After the root agent $root$ obtained the $\widehat{f}(\theta^m)$, it takes at most h Messages for this information to propagate from the root agent to all other agents because the depth of the BFS pseudo-tree is h. Hence, by combining this with **Lemma 1**, at Messages $t+2h+4$, each agent can receive the $\widehat{f}(\theta^m)$ up to Messages t.

Theorem 1: GC-LDP is an anytime algorithm.

Proof: According to **Lemma 2**, at Messages $t+2h+4+\delta$, where $\delta > 0$, each agent can receive the $\widehat{f}(\theta^m)$ up to Messages $t+\delta$. Since the $\widehat{f}(\theta^m)$ will be updated when a better one is found, the $\widehat{f}(\theta^m)$ at Messages $t+2h+4+\delta$ is at least not worse than that at Messages $t+2h+4$. That means, the quality of solutions does not decline over the course of iteration. Consequently, GC-LDP is proved to be an anytime algorithm.

V. Complexity Analysis

In this section, we conduct a complexity analysis of GC-LDP. We assume the total number of agents to be denoted as $|A| = n$, and for each agent $a_i \in A$, the number of its neighbors is represented as $|N_i| = |H_i| + |L_i|$. Considering that the graph forms a complete structure, implying $|N_i| = n$.

In the **Initialization** phase of GC-LDP algorithm, one agent a_i transmits $|N_i|$ messages. In the **Update** phase, it

Algorithm 2: The GC-LDP Algorithm

Construct BFS pseudo-tree
Initialize parameters: K, ω, c_1, c_2, max_{s_c}, max_{f_c}, α, β
for *each agent* a_i **do**
  for *each particle* $j \in K$ **do**
    $v_i^j \leftarrow 0$
    $\theta_i^j \leftarrow$ a random value from D_i
  Send θ_i to agent in L_i
while *Termination condition not met each agent* a_i **do**
  for θ_u *received from neighbor* $u \in H_i$ **do**
    $costSet \leftarrow \{\}$
    for *each particle* $j \in K$ **do**
      $cost_{i,u}^j \leftarrow f(\theta_i^j, \theta_u^j)$
      $costSet \leftarrow cost_{i,u}^j \cup costSet$
      $localCost^j \leftarrow \sum_{u \in N_i} cost_{i,u}^j$
    Send $costSet$ and $localCost$ to agents in H_i
    if $localCost^s$ *is the smallest in* $localCost$ **then**
      Find s and $Group(\theta^s)$
      Send $Group(\theta^s)$ to agents in L_i
  for *particles* $j, l \in Group(\theta^s)$ *(l can be equal to j) received from* H_i **do**
    $cost_{i,u}^m \leftarrow \min f(\theta_i^j, \theta_u^l)$
  Send $cost_{i,u}^m$ to agents in H_i
  Wait until $cost_{i,u}^m$ and $costSet$ received from all agent in L_i
  if $\mid L_i \mid \neq 0$ **then**
    $\widehat{f}(\theta^m) \leftarrow \sum_{u \in L_i} cost_{i,u}^m$
    for *each particle* $j \in K$ **do**
      $\widehat{f}(\theta^j) \leftarrow \sum costSet$
    if $a_i \neq$ *root* **then**
      Send $\widehat{f}(\theta^m)$ and $\widehat{f}(\theta)$ to agents in H_i
  if $a_i =$ *root* **then**
    if $\widehat{f}(\theta^m)(t) < \widehat{f}(\theta^m)(t-1)$ **then**
      output $\widehat{f}(\theta^m)(t)$
    else
      output $\widehat{f}(\theta^m)(t-1)$
      $\widehat{f}(\theta^m)(t) \leftarrow \widehat{f}(\theta^m)(t-1)$
    $pBestSet \leftarrow \{\}$
    for *each particle* $j \in K$ **do**
      if $\widehat{f}(\theta^j) < \widehat{f}(pBest^j)$ **then**
        $pBest^j \leftarrow j$
        $pBestSet \leftarrow pBest^j \cup pBestSet$
      if $\widehat{f}(\theta^j) < \widehat{f}(gBest)$ **then**
        $gBest \leftarrow j$
    Send $pBestSet$ and $gBest$ to agents in L_i
  Wait until $pBestSet$ and $gBest$ are received from H_i
  if $pBestSet$ *and* $gBest$ *are received from* H_i *then* **then**
    Calculate s_c and f_c according to Equation (4) and (5)
    for *each particle* $j \in K$ **do**
      if $gBest = j$ **then**
        Calculate θ_i^j and v_i^j according to Equation (8) and (9)
      else
        Calculate θ_i^j and v_i^j according to Equation (7) and (9)
if $\mid L_i \mid \neq 0$ **then**
  for *each particle* $j \in K$ **do**
    Send θ_i^j, $pBest^j$ and $gBest$ to agents in L_i

sends $|L_i|$ messages, in the **Evaluation** phase, the number of messages is $|H_i|$, and in the **Local Disturbance** phase, the agent sends $|H_i| + |L_i|$ messages. Consequently, the total number of messages sent by one agent a_i is bounded by $O(|N_i|+2|H_i|+2|L_i|)$. Given the complete graph assumption, it is $O(3n)$ in the worst case.

The size of messages sent by each agent a_i are as follows: θ_i is $O(K*n)$, $Group(\theta^s)$ is $O(K/2*n)$, $\widehat{f}(\theta^j)$ is $O(n)$, $\widehat{f}(\theta)$ is $O(K*n)$, $pBest^j$ is $O(K*n)$, and $gBest$ is $O(n)$. Hence, the total message size sent by each agent a_i is $O(4.5*K*n+2*n)$, which can be simplified to $O(K*n)$ per iteration.

The computational complexity for calculating each message during one iteration is as follows: θ_i^j is $O(K)$, v_i^j is $O(K)$, $\widehat{f}(\theta^j)$ is $O(K*n)$, and $cost_{i,u}^m$ is $O(n)$. Consequently, in the worst case, the total computational complexity per agent for a single iteration is $O(2*K+2*K*n+n)$, further simplified to $O(K*n)$.

The complexity analysis demonstrates that GC-LDP is superior to gradient-based optimization algorithms in terms of computational overhead and memory requirements.

VI. Experiments and Analysis

A. Experimental configuration

To verify the performance of the proposed GC-LDP, we compare it with the competing algorithms such as C-CoCoA, PFD, PFD-LD and DSDE-CD on four benchmark problems: **Random Graphs**, **Scale-free Networks**, **Random Trees**, and **Small-world Networks**. Since D-Bay can only solve the problem instances with less than 10 agents, this paper does not exhibit the comparison results with the D-Bay. According to the form of constraint function used in the compared references, we employ constraints expressed in the form: $ax^2 + bx + cxy + dy + ey^2 + f$ to ensure fair and unbiased comparison. It is noteworthy that GC-LDP can solve C-DCOPs with any constraint forms. The coefficients, a, b, c, d, e, and f, are obtained randomly from the interval [-5, 5]. The variable domain D_i, controlled by agent a_i, is set to [-50, 50]. The experimental configuration to evaluate the solution quality is exhibited in Table I.

We set the population size of GC-LDP to 2000, and other parameters to $w = 0.9$, $c_1 = 0.9$, $c_2 = 0.1$, $\alpha = 0.1$, $\beta = 0.1$, $max_{s_c} = 15$, and $max_{f_c} = 5$, respectively.

For non-iterative algorithm C-CoCoA, according to the original reference, the discrete point is set to 3, the number of nonlinear optimizations is set to 100, and the learning rate ac is set to 0.01.

In addition, to ensure the statistical characteristic of the experimental results, we randomly repeat each instance 30 times independently and use the average of the 30 repetitions as results for a question instance.

B. Comparisons with the state-of-the-art algorithms

This subsection presents the comparison results of GC-LDP and other competing algorithms on the four types of benchmark problems. The solution quality on these benchmark problems is exhibited in Figure 6 to 10, respectively.

TABLE I
EXPERIMENTAL CONFIGURATION

Problem type	Density	The number of agents
Random Graphs	0.1 (sparse)	from 10 to 100
Random Graphs	0.6 (dense)	from 10 to 100
Scale-free Networks	$m_1 = 10$, $m_2 = 7$	from 50 to 100
Random Trees	-	from 50 to 100
Small-world Networks	$K = 6$, $P = 0.5$	from 50 to 100

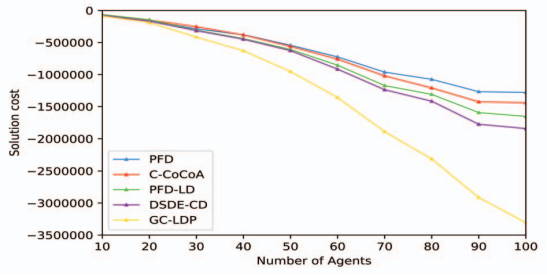


Fig. 6. Solution quality on Sparse Random Graphs.

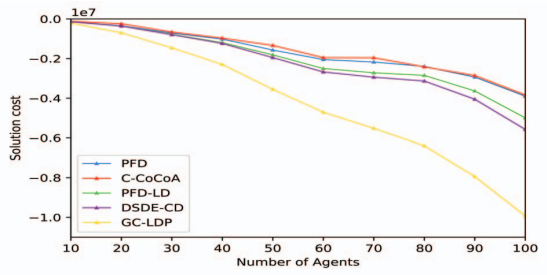


Fig. 7. Solution quality on Dense Random Graphs.

The comparison results demonstrate that the solution quality of GC-LDP significantly outperforms all of the competing state-of-the-art algorithms. Furthermore, the larger the scale of the problem, the better the performance of GC-LDP. The reason is that the local disturbance strategy can quickly find better solutions when solving complex problems.

As seen from Figure 8 and 10, it is worth noting that GC-LDP has the most obvious superiority in solving Scale-free networks and small-world networks, which have a similar topology. Therefore, It can be reasonably inferred that the proposed GC-LDP is very suitable for solving problems with such topology.

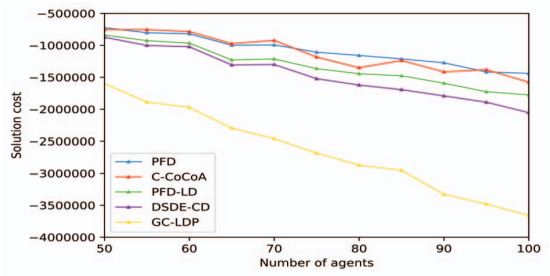


Fig. 8. Solution quality on Scale-free Networks.

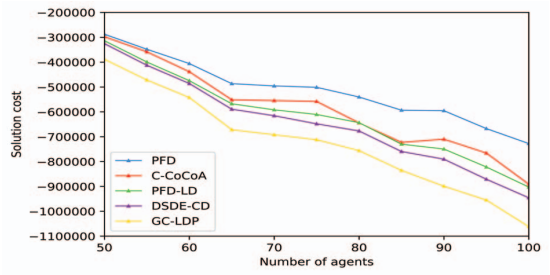


Fig. 9. Solution quality on Random Trees.

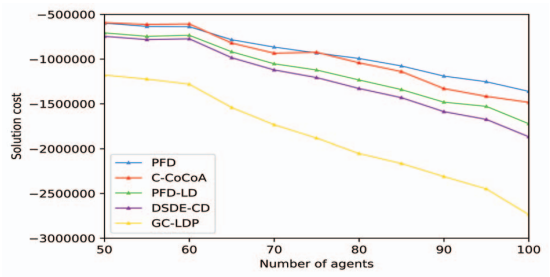


Fig. 10. Solution quality on Small-world Networks.

Table II exhibits the average improvement rates of GC-LDP compared to four competing algorithms on different benchmark problems. The average improvement rates are calculated according to the improvement rate of the configuration of different numbers of agents, and the results are rounded to two decimal places.

TABLE II
THE AVERAGE IMPROVEMENT RATES OF GC-LDP COMPARED TO FOUR COMPETING ALGORITHMS

Benchmark Problems	PFD	C-CoCoA	PFD-LD	DSDE-CD
Sparse Random Graphs	91.38%	73.35%	58.56%	46.67%
Dense Random Graphs	129.50%	148.24%	95.09%	86.32%
Scale-free Networks	142.32%	138.56%	99.80%	82.95%
Random Trees	40.07%	25.04%	17.13%	17.84%
Small-world Networks	99.46%	91.91%	65.23%	54.21%

VII. Conclusion

In this paper, we propose a Group Correction-based Local Disturbance PSO algorithm named GC-LDP to enhance the exploitation and exploration ability of PSO. Firstly, we introduced the average of the personal best positions and the average of the personal current positions into the PSO velocity update formula to utilize the group knowledge to correct the exploitation direction. Then, we design a local disturbance strategy to increase the swarm diversity by searching the nearest particle group in the solution space to enhance its exploration ability. We theoretically proved that GC-LDP is an anytime algorithm. Extensive experiment results also demonstrate that utilizing group knowledge can correct the evolutionary direction, which may be a new perspective for the research of optimization algorithms.

References

[1] Z. Chen, Y. Deng, and T. Wu, "An iterative refined Maxsum_ad algorithm via single-side value propagation and local search," Proceedings of the 16th conference on autonomous agents and multiagent systems, 2017, pp. 195-202.

[2] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo, "ADOPT: Asynchronous distributed constraint optimization with quality guarantees," Artificial Intelligence, vol. 161, No. 1-2, 2005, pp. 149-180.

[3] A. Petcu, and B. Faltings, "DPOP: A scalable method for multiagent constraint optimization," IJCAI 05. No. CONF. 2005, pp. 266-271.

[4] O. Litov, and A. Meisels, "Forward bounding on pseudo-trees for DCOPs and ADCOPs," Artificial Intelligence, vol. 252, 2017, pp. 83-99.

[5] W. Zhang, G. Wang, Z. Xing, and L. Wittenburg, "Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks," Artificial Intelligence, vol.161, 2005, pp. 55-87.

[6] R. T. Maheswaran, J. P. Pearce, and T. Milind, "Distributed Algorithms for DCOP: A Graphical-Game-Based Approach," PDCS, 2004, pp. 432-439.

[7] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings,"Decentralised coordination of low-power embedded devices using the max-sum algorithm," 2008, pp. 639-646.

[8] F. Fioretto, W. Yeoh, and E. Pontelli, "A multiagent system approach to scheduling devices in smart homes," Workshops at the Thirty-First AAAI Conference on Artificial Intelligence, 2017.

[9] S. Mahmud, M. Choudhury, M. M. Khan, T. Long and N. R. Jennings, "AED: An anytime evolutionary dcop algorithm," arXiv preprint arXiv:1909.06254, 2019.

[10] S. Mahmud, M. M. Khan, M. Choudhury, T. Long and N. R. Jennings,"Learning optimal temperature region for solving mixed integer functional DCOPs," arxiv preprint arxiv:2002.12001, 2020.

[11] Z. Chen, and T. Wu, Y. Deng, and C. Zhang, "An ant-based algorithm to solve distributed constraint optimization problems," Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32. No. 1, 2018.

[12] V. Lesser, CL. Ortiz Jr, and T. Milind Tambe, "Distributed sensor networks: introduction to a multiagent perspective," Distributed Sensor Networks: A Multiagent Perspective, Boston, MA: Springer US, 2003, pp. 1-8.

[13] C. Hsin, and M. Liu, "Network coverage using low duty-cycled sensors: random & coordinated sleep algorithms," Proceedings of the 3rd international symposium on Information processing in sensor networks, 2004.

[14] R. Stranders, A. Farinelli, A. Rogers, and N. R. Jennings, "Decentralised control of continuously valued control parameters using the max-sum algorithm," Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems, 2009, pp. 601-608.

[15] T. Voice, R. Stranders, A. Rogers, and N. R. Jennings, "A Hybrid Continuous Max-Sum Algorithm for Decentralised Coordination," ECAI 2010, IOS Press, 2010, pp. 61-66.

[16] K. D. Hoang, W. Yeoh, M. Yokoo, and Z. Rabinovich, "New algorithms for continuous distributed constraint optimization problems," Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, 2020, pp. 502-510.

[17] M. Choudhury, S. Mahmud, and M. M. Khan, "A particle swarm based algorithm for functional distributed constraint optimization problems," Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34. No. 05, 2020, pp. 7111-7118.

[18] M. Shi, X. Liao, and Y. Chen, "A Particle Swarm with Local Decision Algorithm for Functional Distributed Constraint Optimization Problems," International Journal of Pattern Recognition and Artificial Intelligence, Vol. 36 No. 12, 2022.

[19] A. Sarker, M. Choudhury, and M. M. Khan, "A local search based approach to solve Continuous DCOPs," Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems, 2021, pp. 1127-1135.

[20] M. Shi, X. Liao, and Y. Chen, "A dual-population search differential evolution algorithm for functional distributed constraint optimization problems," Annals of Mathematics and Artificial Intelligence, Vol. 90 No. 10, 2022, pp. 1055-1078.

[21] J. Fransman, J. Sijs, H. Dol, E. Theunissen, and B. D. Schutter, "Distributed bayesian: a continuous distributed constraint optimization problem solver," Journal of Artificial Intelligence Research 76, 2003, pp. 393-433.

[22] J. Kennedy, and R. Eberhart, "Particle swarm optimization," Proceedings of ICNN'95 - International Conference on Neural Networks, Vol. 4, 1995, pp. 1942-1948.