Journal of Computing and Electronic Information Management
ISSN: 2413-1660 | Vol. 21, No. 1, 2026

Local Elite-Guided Collaborative Optimization Algorithm for Continuous Distributed Constraint Optimization Problems

Meifeng Shi, Yanmei Yu, Yuan Chen

School of Computer Science and Engineering, Chongqing University of Technology, Chongqing 400054, China

Abstract: To address the issues of premature convergence to local optima and slow convergence speed in Continuous Distributed Constraint Optimization Problems (C-DCOPs), this paper proposes a Local Elite-Guided Collaborative Optimization (EGA-LC) algorithm. Utilizing a Breadth-First Search (BFS) pseudo-tree, the algorithm constructs a distributed population structure partitioned into ten sub-populations for parallel search, each further classified into dominant, non-dominant, and local dominant sub-populations based on fitness. Subsequently, a differential evolution strategy is designed to guide individual updates through the synergistic influence of global elites and local dominant solutions, enabling deep mining of high-quality local constraint information. Furthermore, a periodic cross-population recombination and reassignment mechanism is introduced, which leverages historical global optimal solutions to accelerate convergence and prevents evolution stagnation by reallocating sub-populations. Experimental results on four benchmark problem sets demonstrate that EGA-LC significantly outperforms five state-of-the-art algorithms in terms of both solution quality and convergence speed.

Keywords: Continuous Distributed Constraint Optimization Problems; Local Elite-Guided Collaboration; Differentiated Strategy; Cross-population Elite Recombination.

1. Introduction

Multi-agent Systems (MAS)[1], as a core paradigm of distributed artificial intelligence, address complex problems through the collaborative interaction of multiple autonomous agents. Distributed Constraint Optimization Problems (DCOPs)[2] provide a formal modeling framework for multi-agent coordination, applicable to real-world scenarios such as microgrid control[3], sensor networks[4], and smart homes[5]. Based on whether they can guarantee the global optimal solution, DCOPs algorithms can be categorized into two major classes: complete and incomplete algorithms [6]. Complete algorithms ensure optimality by systematically traversing the solution space but incur significant computational overhead, with typical representatives including ADOPT [7] and DPOP [8]. In contrast, incomplete algorithms forgo optimality guarantees to provide feasible solutions within limited time; major research directions include DSA [9] and GDBA [10].

However, in scenarios such as sensor orientation for target tracking [11] and water surface scheduling for wireless sensors [12], modeling with continuous variables is often required. To overcome the limitations of discretization, Stranders [13] et al. proposed the Continuous Distributed Constraint Optimization Problem (C-DCOPs) framework, which allows variables to take values in continuous domains and represents constraint costs as functions. Since the solution space of C-DCOPs is infinite and the constraint functions often exhibit non-linear and non-convex characteristics, they pose severe challenges to algorithm design.

Regarding C-DCOPs solving algorithms, early research such as Continuous Max-Sum (CMS) extended the discrete Max-Sum to the continuous domain by approximating utility functions with piecewise linear functions, though its applicability remains limited. Hybrid Continuous Max-Sum (HCMS)[14] further integrated message passing with continuous non-linear optimization to alleviate local optima issues, but it still faces limitations when dealing with non-differentiable functions. Among the algorithms proposed by Hoang[15] et al., Exact Continuous DPOP (EC-DPOP) is only applicable to tree topologies with linear or quadratic functions. While Approximate Continuous DPOP (AC-DPOP) handles smooth differentiable functions via dynamic discretization and gradient descent with error bound guarantees, it, along with Clustered AC-DPOP (CAC-DPOP), suffers from exponential memory requirements. Furthermore, the Particle Swarm based C-DCOPs (PFD)[16] algorithm is prone to falling into local optima, and although PFD with Local Decision (PFD-LD)[17] introduces local guidance to accelerate convergence, it increases communication costs. Continuous Cooperative Constraint Approximation (C-CoCoA) [18] employs coarse search combined with gradient fine-tuning but lacks the anytime property. Additionally, the Adaptive Multi-point Crossover Genetic Based C-DCOPs (AMCGA)[19] achieves global search through adaptive crossover, while the Estimation of Distribution Based Algorithm for C-DCOPs (EDA-CD)[20] constructs probability models based on estimation of distribution theory.

Current C-DCOPs swarm intelligence algorithms mostly rely on a single-population structure, lacking parallel guidance from independent elite groups, which leads to diversity loss and local optima. This paper proposes Local Elite-Guided Collaborative Optimization (EGA-LC) algorithm with the following contributions:

1) Multi-tiered Elite Guidance: Partitioning the population into ten sub-populations (dominant, non-dominant, and local dominant). It enables efficient divide-and-conquer exploration through the collaborative guidance of global and local elites.
2) Differential Update Strategy: Implementing customized evolutionary operators for different tiers and utilizing prior solutions from local dominant sub-

populations to co-guide updates, enhancing local information mining and convergence efficiency.

3) Dynamic Recombination and Reassignment: Aggregating historical superior genes via an elite container for periodic exchange, and employing dynamic sub-population reassignment to break path dependency and prevent evolution stagnation.

2. Problem Formulation and Background

2.1. Continuous Distributed Constraint Optimization Problem

A Continuous Distributed Constraint Optimization Problems (C-DCOPs) can be formally represented by a five-tuple$< A, X, D, F, \alpha >$, where:

$A = \{a_1, a_2, \ldots, a_n\}$is a set of agents;

$X = \{x_1, x_2, \ldots, x_m\}$is a set of continuous variables;

$D = \{D_1, D_2, \ldots, D_m\}$ is a set of domains,where each variable x_i takes values from a continuous range $D_i \in [UB_i, LB_i]$;

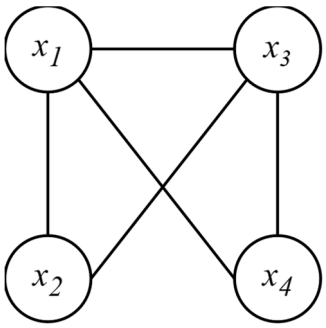
$F = \{f_1, f_2, \ldots, f_k\}$is a set of constraint functions, where each function $f : D_{i_1} \times \ldots \times D_{i_k} \to R^+, f \in F, D_{i_k} \in D$defines the cost associated with the combinations of values assigned to the involved variables.

$\alpha: X \to A$ is a mapping function that associates each variable with an agent. For simplicity, this paper assumes that each agent controls exactly one variable; thus, the terms "agent x_i " and "variable a_i " are used interchangeably hereafter.

The objective of solving a C-DCOPs is to find a complete assignment that minimizes the total global cost. Unlike discrete DCOPs, the variables in C-DCOPs are continuous, and costs are calculated through constraint functions rather than being explicitly provided in a constraint matrix. The optimal solution X^* is defined as:

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

Figure 1 illustrates an example of a C-DCOPs. Specifically, Figure 1(a) shows a constraint graph with four nodes and five edges, where each node represents an agent that independently controls a continuous variable.



(a)

$$\forall x_i \in X : D_i = [-5, 5]$$
$$f_{12} = x_1^2 - x_2^3 + sin(x_2)$$
$$f_{13} = 3x_1x_3 - x_2^2 + e^{x_3}$$
$$f_{14} = x_1x_4^2 - tan(x_4) + x_1$$
$$f_{23} = x_2^3 - 2x_2x_3 + cos(x_3)$$
$$f_{34} = ln(|x_3| + 1) + x_3 - x_4$$

(b)

Fig.1 An example of C-DCOPs;(a) Constraint graph (b) Constraint function

Figure 1(b) displays the constraint functions between nodes, describing the mutual influence between variables controlled by different agents. Based on the specific values assigned to variables within the domain $[-5, 5]$, agents calculate the corresponding constraint costs using the given functional expressions. In this instance, the agents aim to collaboratively find a set of assignments that minimizes the sum of all constraint costs

2.2. Distributed Population

In the C-DCOPs framework, the population is stored in a distributed manner among all agents. Each agent is responsible for maintaining a local population consisting of multiple individuals, where each individual corresponds to a local solution component from the perspective of that agent. Individuals with the same index across all agents collectively form a complete global solution.

Table 1 illustrates this distributed storage structure in a two- dimensional format. Assuming there are n agents and m candidate solutions (i.e., there are m individuals in a population), the local population maintained by agent a_i corresponds to the i -th column of the table, which can be expressed as $\{S_1.x_i, S_2.x_i, \cdots, S_m.x_i\}$, where $S_k.x_i$ represents the value of the k -th solution stored by agent a_i at its own dimension. Correspondingly, the k -th row represents a complete candidate solution, namely a global solution, which is collectively composed of the values of all agents at the k -th index. This is denoted as the k -th row in the table: $Solution_k = \{S_k.x_1, S_k.x_2, \cdots, S_k.x_n\}$.

Tab.1 An Example of Distributed Population

	$Agenta_1$	$Agenta_2$	…	$Agenta_n$
$Solution_1$	$S_1.x_1$	$S_1.x_2$	…	$S_1.x_n$
$Solution_2$	$S_2.x_1$	$S_2.x_2$	…	$S_2.x_n$
…	…	…	…	…
$Solution_m$	$S_m.x_1$	$S_m.x_2$	…	$S_m.x_n$

Consequently, a distributed population is essentially a collection of all agents' local populations, while each candidate solution represents a global view formed by selecting the corresponding local solution components from each local population.

2.3. Breadth First Search Pseudo-tree

In Distributed Constraint Optimization Problems and their continuous domain extensions, to efficiently organize the interaction between agents, it is usually necessary to convert the original constraint graph into a specific communication structure. Among them, the Breadth-First Search (BFS) Pseudo-tree is a structure widely used in DCOPs and C-DCOPs, as shown in Figure 2. This pseudo-tree structure not only determines the hierarchical organization of agents but also defines the basic order of message passing.

In a BFS pseudo-tree, edges are divided into two categories: solid lines represent tree edges connecting parent and child nodes, forming the hierarchical skeleton of the tree; dashed lines represent pseudo-edges connecting nodes that are not

parent-child but have constraint relationships, used to maintain the original constraint relationships of the problem. As shown in Figure 2(a), nodes connected by solid lines form the backbone of the tree, such as(x_1, x_2) and(x_1, x_3), while nodes connected by dashed lines form pseudo-parent-child relationships, such as (x_1, x_4). It should be noted that nodes pointed to by pseudo-edges, such as x_4, are called pseudo-child nodes; they, together with the child nodes connected by solid lines, constitute all lower-level constraint relationships of that node, but they do not participate in the main path of message passing as direct child nodes in the tree structure.

Based on the constructed pseudo-tree, nodes can be further ordered, as shown in Figure 2(b). The priority of nodes follows these rules: agents with a smaller depth have a higher priority; the root node is at depth 0, and the smaller the depth value, the closer to the root node; when depths are the same, the agent with a smaller variable name index has a higher priority. For example, in Figure 2, nodes x_2 and x_3 are at the same depth, but because the variable name index of x_2 is smaller than that of x_3, the priority of x_2 is higher than that of x_3.

This priority ordering determines the direction of message passing, i.e., high-priority nodes pass information to low-priority nodes. During the execution of the algorithm, after each agent completes its local assignment, it passes its value information downward through the pseudo-tree structure to all lower-priority neighbors, including child nodes and pseudo-child nodes. For example, the root node x_1 sends assignment messages to $\{x_2, x_3, x_4\}$, while x_2 sends assignment messages to x_3. Through this hierarchical message-passing mechanism, various DCOPs and C-DCOPs algorithms can efficiently perform distributed solving.

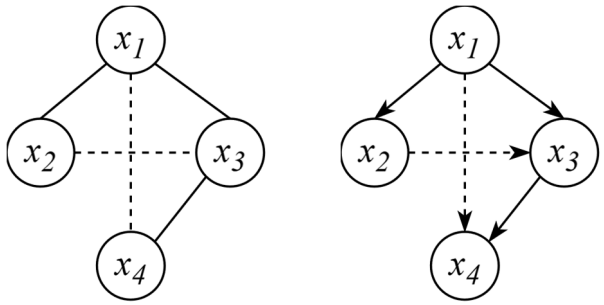


Fig.2 An Example of BFS Pseudo-tree;(a) BFS Pseudo-tree (b) Ordered Arrangement

3. The EDA-CD Algorithm

EGA-LC is an iterative algorithm based on population evolution, consisting of four core phases: initialization, local elite selection, hierarchical elite co-guided phase, and cross-population recombination. During the initialization phase, a BFS pseudo-tree communication structure is constructed to generate the initial population and initialize parameters. In the local elite selection phase, agents compute and share the local and global fitness of individuals in a distributed manner, completing the population division and the construction of local superior populations. During the hierarchical elite co-guided phase, the algorithm applies differentiated update strategies to the superior and inferior populations based on global and local elite information to generate new individuals. The cross- population recombination phase is responsible for periodically gathering historical elite information to achieve global information sharing through cross-population recombination, effectively avoiding premature convergence and maintaining population diversity. Detailed algorithmic procedures are presented in Algorithm 1.

Algorithm 1: EGA-LC	
	Construct BFS pseudo-tree
	Initialize parameters: K、$Nnum$、α、β、δ、T_{recom}
	for each agent a_i do
	Initialize $S.x_i \in [LB, UB]$ via Eq. (2).
	send $S.x_i$ to agents in L_i
	end for
	$P_N \leftarrow$Partition P into $Nnum$ sub-populations
	$Q \leftarrow \emptyset$
	While $t < maxIteration$ do
	for each agent a_i do
	Send $S.x_i$ to all neighbors
	Receive values and calculate local fitness $fit_{local}(S.x_i)$
	if $fit_{local}(S_{LC}.x_i) = min(fit_{local}(P))$ then
	Send $S_{LC}.x_i$ to all neighbors
	end if
	if $fit_{local}(S_l.x_i) = min(fit_{local}(P_N))$ then
	$P_n^L \leftarrow_{S_l.x_i}$
	end if
	receive$S_{LC}.x_i$and calculate decision fitness$fit_{local}^{LC}(S.x_i)$
	send $\{fit_{local}(S.x_i), fit_{local}^{LC}(S.x_i)\}$ to H_i
	if is root agent then
	Calculate $fit(S.x_i)$ and $fit^{LC}(S.x_i)$
	if $fit^{LC}(S.x_i)(t) < fit^{LC}(S.x_i)(t-1)$ then
	return $fit^{LC}(S.x_i)(t)$
	else
	return $fit^{LC}(S.x_i)(t-1)$
	$fit^{LC}(S.x_i)(t) \leftarrow fit^{LC}(S.x_i)(t-1)$
	end if
	end if
	if t =0 then
	$sort\_cost \leftarrow Rank(P_n, fit)$
	else
	$sort\_cost \leftarrow Rank(Top50\%, fit)$ after $Merge(P, P_{new})$
	Re-partition $sort\_cost$ into $Nnum$ sub-populations
	end if
	Partition into P_N^G and P_N^W based on $sort\_cost$ and Eq. (4)
	if $t \ mod T_{recom} \neq 0$ then
	for each$S_k.x_i \in P_N^G$ do
	$\eta \leftarrow$random value in [0,1]
	if $\eta < \alpha$ then
	Updata $S_k.x_i$ var Eq. (5)
	else if $\eta < \varphi$ then
	Updata $S_k.x_i$ var Eq. (7)
	else
	Retain $S_k.x_i$ var Eq. (8)
	end if
	end for
	for $S_k.x_i \in P_N^W$ do
	$\mu \leftarrow$random value in [0,1]
	if $\mu < \beta$ then
	Updata $S_k.x_i$ var Eq. (10)
	else
	Updata $S_k.x_i$ var Eq. (11)

	end if
	end for
	$Q \leftarrow Q \cup Best(P_N)$
	else
	Updata $S_k.x_i$ var Eq. (13)
	$Q \leftarrow \emptyset$
	end if
	end for
	$t \rightarrow t+1$
	end while

The initialization phase begins with the ordered arrangement of agents in a Breadth First Search (BFS) pseudo-tree and the initialization of parameters, where the pseudo-tree defines the message passing order for subsequent fitness evaluation and update phases. Within the upper and lower bounds of the continuous domain, each agent randomly selects initial values for its controlled decision variables according to Equation 2.

$$S_k.x_i = LB_i + rand(0,1)^*(UB_i - LB_i) \quad (2)$$

Subsequently, high-priority agents pass their assignment information down the pseudo-tree structure to low-priority neighbors, collaboratively constructing an initial population of K complete feasible solutions. This population is randomly and equally divided into $Nnum$ parallel sub-populations $P_N(N = 1,2,\dots,Nnum)$,and an elite container for storing historical high-quality solutions Q is initialized. The local elite selection phase starts after agents complete local assignments and information interaction, where each agent sends and receives solution assignments to compute the fitness of solutions in a distributed manner as shown in Equation 3.

$$fitness(S_k) = \sum_{f_i \in F} f_i(S_k.x_i) \quad (3)$$

After receiving assignment information from all neighbors, agents compute local fitness $fit_{local}(S.x_i)$ and compare it with neighbors to select the individual $S_{LC}.x_i$ with the minimum local fitness in the entire population to broadcast to all neighbors. Simultaneously, the individual $S_l.x_i$ with the minimum local fitness within each sub-population is identified and stored in the local superior population P_N^L. Agents then receive local optimal solutions $S_{LC}.x_i$ from all neighbors, calculate decision local fitness $fit_{local}^{LC}(S.x_i)$, and send $fit_{local}(S.x_i)$ and $fit_{local}^{LC}(S.x_i)$ to all parent nodes. The root node aggregates fitness information from all child nodes to calculate global fitness $fit(S.x_i)$ and decision global fitness values $fit^{LC}(S.x_i)$, updating and outputting the latter (Algorithm 1: lines 19~29). Based on the ranking of global fitness, each sub-population strictly divides its internal individuals into a superior population and an inferior population according to Equation 4 to prepare for the next stage (Algorithm 1: line 36).

$$(P_N^G, P_N^W) = \begin{pmatrix} \{S_k.x_i \in P_N | rank(S_k.x_i) \le \lfloor 0.5 \cdot |P_N| \rfloor\} \\ \{S_k.x_i \in P_N | rank(S_k.x_i) > \lfloor 0.5 \cdot |P_N| \rfloor\} \end{pmatrix} \quad (4)$$

In the hierarchical elite co-guided phase, the algorithm executes differentiated update mechanisms for these two populations. For the superior population, agents generate new individuals under the joint guidance of global and local elites using the difference vector exploration strategy defined in Equation 5 and the cosine oscillation exploitation strategy defined in Equation 7, supplemented by the elite retention mechanism of Equation 8 to maintain population diversity (Algorithm 1: lines 38~47). For the inferior population, agents perform directional exploration guided by the sub-population's global optimal solution according to Equation 10, or Gaussian random gradient perturbation updates according to Equation 11(Algorithm 1: lines 48~55).

$$S_{new}.x_i = S_k.x_i + r_1 * (S_{k1}.x_i - S_{k2}.x_i) + r_2 * R * (S_{best}.x_i - S_k.x_i) + r_3 * \left(\overline{S}_L.x_i - S_k.x_i\right) \quad (5)$$

$$\overline{S}_L.x_i = \frac{1}{|P_N^L|}\sum_{m=1}^{|P_N^L|} S_{m,L}.x_i \quad (6)$$

$$S_{new}.x_i = cos(\pi * r_4 * 0.5) * (S_{best}.x_i - S_k.x_i) + sin(\pi * r_5 * 0.5) * (S_{Lbest}.x_i - S_k.x_i) \quad (7)$$

$$S_{new}.x_i = S_k.x_i \quad (8)$$

$$\varphi = 1 - t^{-1/6}/T^{-1/6} \quad (9)$$

$$S_{new}.x_i = S_{best}.x_i + r_6 * (S_{k3}.x_i - S_{k4}.x_i) \quad (10)$$

$$S_{new}.x_i = \left(S_{k5}.x_i + |RG| * (S_{k6}.x_i - S_{k7}.x_i)\right) * U + (1 - U) * S_k.x_i \quad (11)$$

In Equations 5 through 11, r_1 to r_6 are uniform random numbers in the [0,1] interval, $S_{k1}.x_i$ and $S_{k2}.x_i$ are two individuals randomly selected from the superior population, $S_{k3}.x_i$ and $S_{k4}.x_i$ are two individuals randomly selected from the inferior population, $S_{best}.x_i$ is the global optimal individual of the sub-population, and $S_{Lbest}.x_i$ is the local optimal individual of the sub-population. $cos(\pi * r_4 * 0.5)$ and $sin(\pi * r_3 * 0.5)$ control the guidance strength of the global and local elites, respectively, with values in [0,1] to ensure the search direction always converges toward high-quality regions. $|RG|$ is a random gradient perturbation factor, $RG \sim N(0, \sigma^2)$ following a standard Gaussian distribution, $|\cdot|$ is the absolute value operator, $S_{k5}.x_i$, $S_{k6}.x_i$, and $S_{k7}.x_i$ are three individuals randomly selected from the sub-population, and U take values of 0 or 1.

Furthermore, to precipitate high-quality historical experience during iterations, the algorithm introduces a global elite container, which is an ordered queue for storing the historical optimal solutions of each sub-population, accumulating high-quality elite information across different iterations to provide data support for cross-population interaction. After each iteration, the current sub-population optimal individual $S_{best}.x_i$ is stored in the elite container according to iteration order, following the storage rules in Equation 12.

$$Q(t) = Q(t-1) \cup \{S_{best}.x_i | N = 1,2,\dots,K\} \quad (12)$$

To break local optima, the cross-population recombination phase is periodically triggered when the iteration count reaches $tmodT_{recom} = 0$. Agents extract historical optimal individuals from the elite container to construct a global historical elite pool and calculate the mean of all elite individuals via Equation 13 as the baseline direction for cross-population search. Combining an adaptive search radius from Equation 14 and a direction perturbation factor from Equation 15, the algorithm utilizes Equation 16 to guide the generation of new cross-population individuals.

$$Q_{mean}.x_i = \frac{1}{M}\sum_{i=1}^{N} S_{m,elite}.x_i \quad (13)$$

$$R = \frac{30}{1+(S_i.x_i - Q_{mean}.x_i)^*\xi} \quad (14)$$

$$RA = R * (-1) + (2 * R * r_4) \quad (15)$$

$$S_{new}.x_i = Q_{mean}.x_i + RA \quad (16)$$

Finally, the updated sub-populations retain the top 50% of individuals with the highest fitness, after which the original order is completely disrupted and the individuals are re-divided into $Nnum$ sub-populations. This global recombination mechanism ensures the redistribution of high-quality solutions across different sub-populations, guiding agents to continuously and collaboratively optimize across the entire continuous search space until the maximum number

of iterations is reached (Algorithm 1: lines 32~35).

4. Algorithm Theoretical Analysis

4.1. Theoretical Proof

Theorem (Anytime Property of the EGA-LC Algorithm): The EGA-LC algorithm possesses the anytime property. This means that the algorithm can be interrupted at any moment to return a feasible solution, and the quality of the solution is monotonically non-decreasing over iterative time.

Proof: The algorithm maintains the global optimal solution and updates it after each iteration: $S_{best}.x_i = argmin_{S_k.x_i \in P} F(S_k.x_i)$,since $S_{best}.x_i$ is always selected as the best solution among all current individuals, the sequence of objective function values $\{F(S_{best}.x_i)\}$ constitutes a non-increasing sequence: $F(S^1_{best}.x_i) \geq \ldots \geq F(S^t_{best}.x_i) \geq F(S^{t+1}_{best}.x_i)$ At any moment τ when the algorithm is interrupted, the current optimal solution $S^\tau_{best}.x_i$ can be returned. As the number of iterations increases, $S_{best}.x_i$ gradually converges toward the global optimum. Therefore, the EGA algorithm satisfies the definition of an anytime algorithm

4.2. Complexity Analysis

Suppose the constraint graph is a complete graph. The population size is K, and the number of agents is $|A| = n$. Let denote H_i the set of higher-priority agents and L_i denote the set of lower-priority agents. The number of neighbors for agent x_i is $|N_i| = |H_i| + |L_i|$.

During the initialization and update phases, agent x_i sends value information to its lower-priority neighbors $x_i \in L_i$, with a complexity of $O(2 \cdot |L_i|)$. During the evaluation phase, agent x_i sends cost information to its higher-priority neighbors $x_i \in H_i$, with a complexity of $O(|H_i|)$. Throughout the iteration, the total number of messages sent by an agent is $O(2 \cdot |L_i| + |H_i|)$. In the worst-case scenario, where all neighbors are lower-priority, the number of messages sent per iteration remains $O(n)$.

During the iterative process, an agent only needs to calculate the fitness for each individual after updating them. In the worst-case scenario, this part of the time complexity is $O(K \cdot n)$. Therefore, the total time complexity of the EGA-LC algorithm is $O(K \cdot n)$.

5. Experimental Results

To comprehensively evaluate the performance of the EGA-LC algorithm, this section provides a comparative analysis of EGA-LC and several baseline algorithms across various benchmark problems, including sparse random graphs, dense random graphs, random trees, small-world networks, and scale-free networks. All experiments were conducted on a workstation equipped with an Intel(R) Core(TM) i3-10100F CPU at 3.60 GHz and 16GB of RAM. Following the methodology in[21], the constraint cost functions are defined in the quadratic form $ax^2 + bx + cxy + dy + ey^2 + f$, where the coefficients a, b, c, d, e, f are randomly generated within the interval $[-5, 5]$, and the domain of variables is set to $[-50,50]$. To ensure that the algorithms achieve stable convergence while avoiding unnecessary computational overhead, the maximum number of iterations is uniformly set to 500. To mitigate the impact of stochastic variations on experimental outcomes, each algorithm was executed independently 30 times for every experimental configuration, with the average values serving as the final performance metrics. The parameters for the comparative algorithms were configured in accordance with [22], as summarized in Table 2.

Tab.2 Parameter configuration of comparison algorithms for C-DCOPs

Algorithms	Parameter
PFD	$K = 2000, \omega = 0.9, c_1 = 0.9, c_2 = 0.1, max_{sc} = 15, max_{fc} = 5$
PFD-LD	$K = 2000, \omega = 0.9, c_1 = 0.9, c_2 = 0.1, max_{sc} = 15, max_{fc} = 5, \alpha = 0.1, \beta = 0.1$
C-CoCoA	$\alpha = 3, \Delta t = 0.01, T = 100$
AMCGA	$K = 2000, G = 5 * n, P_{c_1} = 0.9, P_{c_2} = 0.05, S_a = 0.3 * n, P_m = 0.02$
EDA-CD	$K = 8 * n, \beta = 0.01, G = 0.28 * n$
EGA-LC	$K = 2000, Nnum = 10, \alpha = 0.8, \beta = 0.8$

In the sparse random graph scenarios, EGA-LC exhibits superior rapid-search capabilities during the early iterations, with a convergence gradient significantly steeper than that of the comparative algorithms. As the iterations progress, EGA-LC consistently maintains a distinct advantage in solution quality, driven by the continuous guidance of local elite information. In contrast, AMCGA experiences apparent premature convergence in the later stages of the search, while EDA-CD, despite its ability to gradually narrow the performance gap with PFD-LD, remains unable to surpass the performance benchmark set by EGA-LC. Fig. 4 illustrates that as the agent population scales from 50 to 100, the performance advantage of EGA-LC in solution quality amplifies in proportion to the increasing problem complexity.

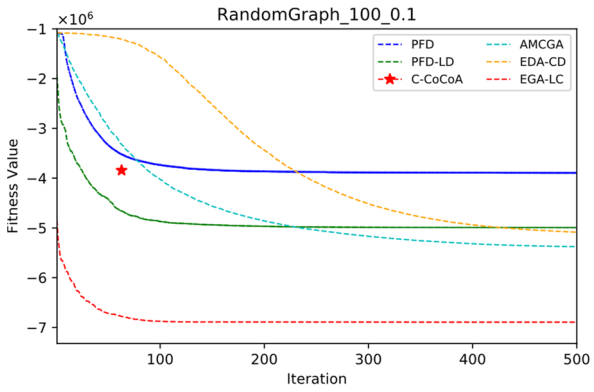


Fig.3 Convergence curves of EGA-LC and comparison algorithms on sparse random graphs (number of agents=100)

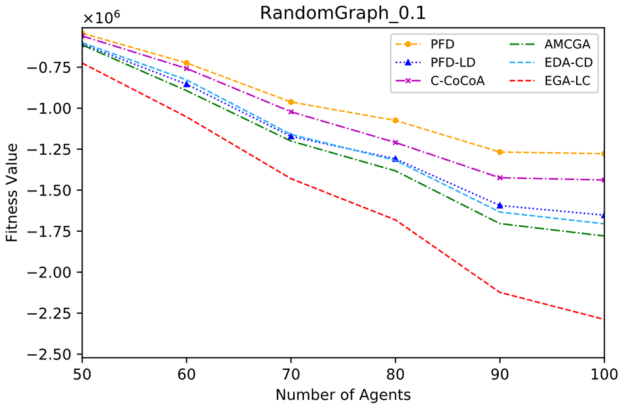


Fig.4 Solution Quality of EGA-LC and Comparison Algorithms on Sparse Random Graphs

For dense random graphs, which represent high-complexity scenarios, Fig. 5 demonstrates that EGA-LC possesses exceptional search performance; its initial solution quality significantly outperforms all baselines, and it reaches stability within approximately 50 iterations, reflecting high computational efficiency. The scalability analysis in Fig. 6 indicates that while performance disparities among algorithms are relatively minor in small-scale instances, the robustness of EGA-LC becomes increasingly prominent as the number of agents grows. The experimental results verify that the local elite co-guided mechanism effectively mitigates the high-coupling constraints inherent in dense topologies, thereby enhancing convergence efficiency.

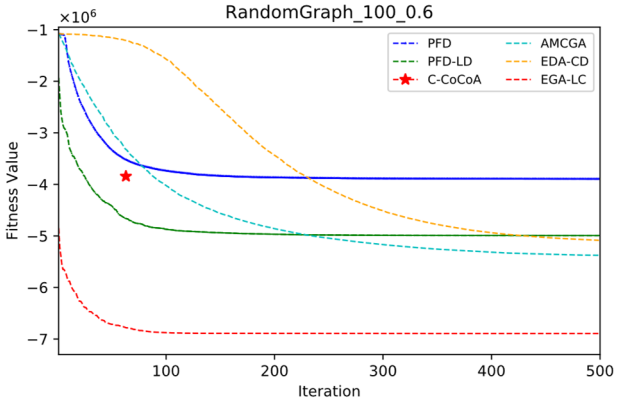


Fig.5 Convergence Curves of EGA-LC and Comparison Algorithms on Dense Random Graphs (Number of Agents=100)

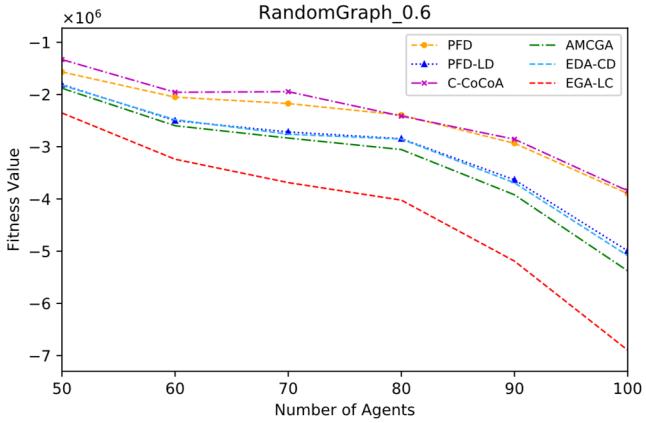


Fig.6 Solution Quality of EGA-LC and Comparison Algorithms on Dense Random Graphs

In the context of random trees, characterized by their hierarchical topological constraints, EGA-LC maintains excellent search speed, reaching convergence at approximately the 100th iteration. Fig. 8 presents the solution quality across various agent scales, where EGA-LC consistently achieves the optimal objective value. Notably, the performance advantage of EGA-LC in this structure shows a degree of scale-dependency, where its local decision-making mechanism demonstrates superior stability when navigating hierarchical constraint structures.

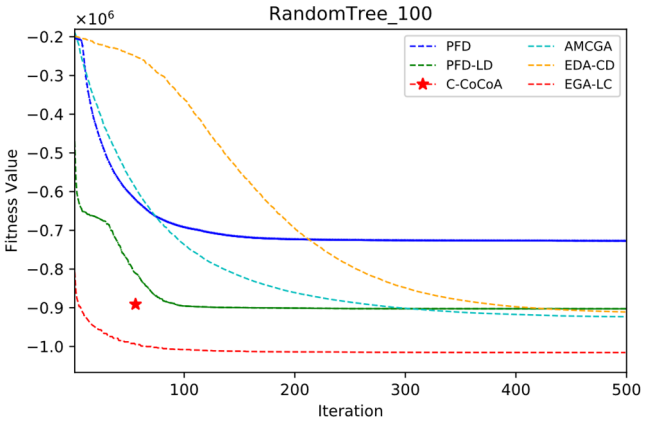


Fig.7 Convergence Curves of EGA-LC and Comparison Algorithms on Random Trees (Number of Agents=100)

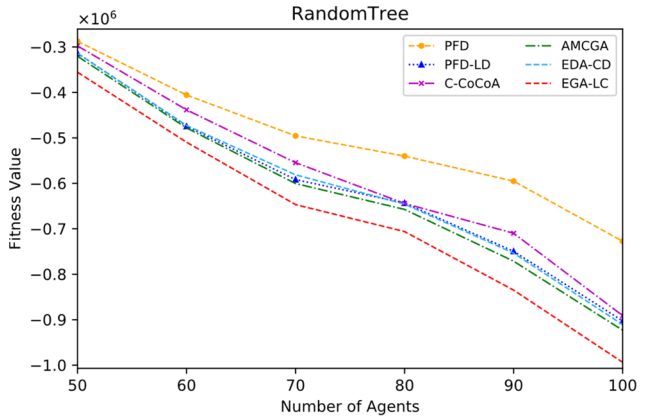


Fig.8 Solution Quality of EGA-LC and Comparison Algorithms on Random Trees

Small-world networks, featuring high clustering coefficients and short average path lengths, impose significant demands on the parallel search capabilities of the algorithm. As shown in the convergence curves in Fig. 9, EGA-LC outperforms all comparative algorithms with a rapid descent trend in the early stages of iteration, stabilizing within approximately 80 cycles. Fig. 10 further reveals the exceptional performance of EGA-LC in small-world topologies: the performance gap between EGA-LC and the baselines widens significantly as the agent population increases from 50 to 100. This confirms a highly favorable alignment between the multi-subpopulation parallel search strategy of EGA-LC and the efficient connectivity inherent in small-world networks.

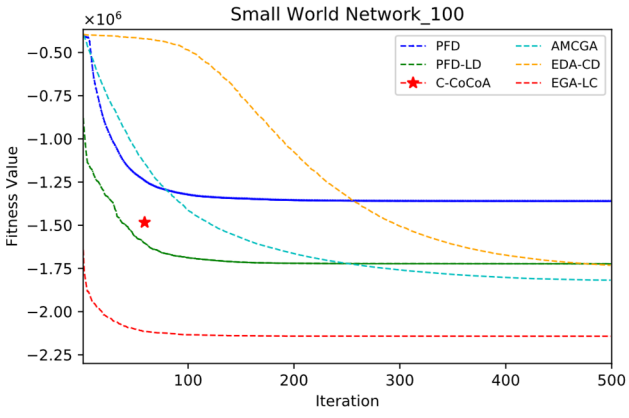


Fig.9 Convergence Curves of EGA-LC and Comparison Algorithms on Small-World Networks (Number of Agents=100)

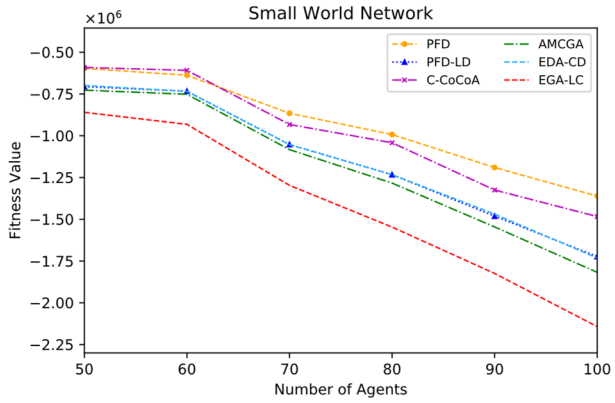


Fig.10 Solution Quality of EGA-LC and Comparison Algorithms on Small-World Networks

Scale-free networks are dominated by a small number of high-degree nodes, and this non-uniform topology necessitates a high level of balance in the algorithm's search capability. The convergence curves in Fig. 11 demonstrate that EGA-LC consistently maintains a performance lead throughout the iterative process, achieving the lowest global constraint cost. The comparative data in Fig. 12 indicates that while PFD-LD and EDA-CD exhibit relatively similar performance across different scales, EGA-LC achieves the optimal solution quality in all scenarios due to its dynamic population recombination strategy. Furthermore, this advantage broadens steadily as the problem scale expands.

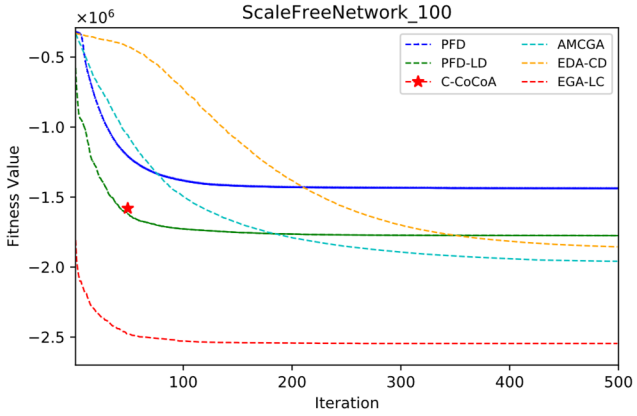


Fig.11 Convergence Curves of EGA-LC and Comparison Algorithms on Scale-Free Networks (Number of Agents=100)

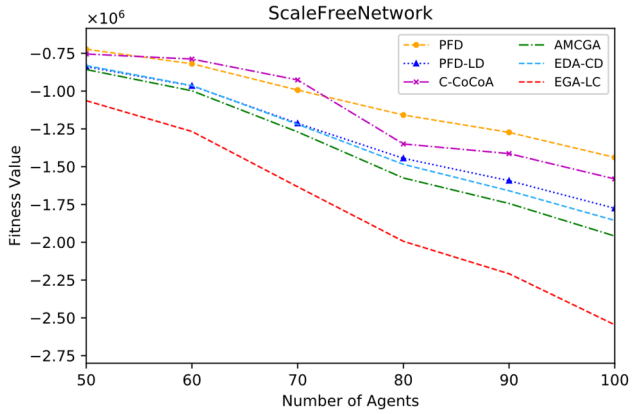


Fig.12 Solution Quality of EGA-LC and Comparison Algorithms on Scale-Free Networks

6. Conclusion and Future Work

To address the challenges of low search efficiency and susceptibility to local optima in large-scale C-DCOPs, this paper proposes an Local Elite-Guided Collaborative Optimization (EGA-LC) algorithm based on local elite collaborative guidance. By constructing a communication topology using a BFS pseudo-tree, the algorithm enables efficient information transfer between higher-priority nodes and children nodes, significantly reducing global communication overhead. At its core, EGA-LC employs a hierarchical evolutionary strategy that partitions the population into dominant and non-dominant groups based on global fitness feedback, applying differentiated evolutionary operators to balance exploration breadth and exploitation depth. Furthermore, a dynamic re-partitioning mechanism is introduced to prevent evolutionary stagnation, which, combined with an elite container and Anytime mechanism, ensures the consistent output of high-quality optimal solutions. Experimental results demonstrate that EGA-LC excels in convergence speed and solution stability. In future work, we plan to incorporate adaptive probability operators and local perturbation mechanisms to dynamically adjust parameters based on evolutionary states, while extending this hierarchical coordination framework to solve multi-objective and dynamic C-DCOPs.

Acknowledgements

This work was supported by the Graduate Innovation Project of Chongqing University of Technology (No. gzlcx20253208).

References

[1] Pujol-onzalez M. Multi-agent coordination: DCOP and beyond[C]//IJCAI Proceedings Internation-al Joint Conference on Artificial Intelligence.2011,22(3):2838.

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

[3] Kumar A, Faltings B, Petcu A. Distributed constraint optimization with structured resource constraints[C]//Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS).2009:923-930.

[4] Muldoon C, O'Hare G M P, O'Grady M J, et al. Distributed constraint optimisation for resource limited sensor networks[J]. Science of Computer Programming, 2013,78(5):583-593.

[5] Fioretto F, Yeoh W, Pontelli E. A multiagent system approach to scheduling devices in smart homes[C]//Proceedings of the 16th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS).2017:981-989.

[6] Wang Jie. Research on Complete Synchronous Search Algorithms for Distributed Constraint Optimization Problems Based on Pseudotrees [D]. Chongqing University, 2023.

[7] Modi P J, Shen W M, Tambe M, et al. ADOPT: Asynchronous distributed constraint optimization with quality guarantees[J]. Artificial Intelligence, 2005, 161(1-2): 149-180.

[8] Rashik M, Rahman M M, han M M, et al. Speeding up distributed pseudo-tree optimization procedures with cross edge consistency to solve DCOP[J]. Applied Intelligence, 2021, 51: 1733- 1746.

[9] 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.

[10] Okamoto S, Zivan R, Nahon A. Distributed Breakout: Beyond Satisfaction[C]// Proc. of the 25th International Joint Conference on Artifical Intelligence,2016,447-453.

[11] 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.

[12] Hsin C, Liu M. Network coverage using low duty-cycled sensors: random & coordinated sleep algorithms[C]//Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN).2004:433-442.

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

[14] Voice T, Stranders R, Rogers A, et al. A hybrid continuous max-sum algorithm for decentralised coordination[C]//Proceedings of the 19th European Conference on Artificial Intelligence. 2010:61-66.

[15] 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.

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

[17] 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.

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

[19] Liao Xin, Shi Meifeng, Chen Yuan. An adaptive multi-point crossover genetic algorithm for solving continuous distributed constraint optimization problems [J]. CAAI Transactions on Intelligent Systems, 2023, 18(04): 793-802.

[20] Shi M, Zhang P, Liao X, et al. An estimation of distribution based algorithm for continuous distributed constraint optimization problems[J]. Information Technology and Control, 2024,53(1):80-97

[21] 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.

[22] 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.