A Multiagent Co-Evolutionary Algorithm With Penalty-Based Objective for Network-Based Distributed Optimization

Tai-You Chen, Wei-Neng Chen, *Senior Member, IEEE*, Xiao-Qi Guo, Yue-Jiao Gong, *Senior Member, IEEE*, and Jun Zhang, *Fellow, IEEE*

***Abstract*—The emergence of networked systems in various fields brings many complex distributed optimization problems, where multiple agents in the system need to optimize a global objective cooperatively when they only have local information. In this work, we take advantage of the intrinsic parallelism of evolutionary computation to address network-based distributed optimization. In the proposed multiagent co-evolutionary algorithm, each agent maintains a subpopulation in which individuals represent solutions to the problem. During optimization, agents perform local optimization on their subpopulations and negotiation through communication with their neighbors. In order to help agents optimize the global objective cooperatively, we design a penalty-based objective function for fitness evaluation, which constrains the subpopulation within a small and controllable range. Further, to make the penalty more targeted, a conflict detection method is proposed to examine whether agents are conflicting on a certain shared variable. Finally, in order to help agents negotiate a consensus solution when only the local objective function is known, we retrofit the processes of negotiating shared variables, namely, evaluation, competition, and sharing. The above approaches form a multiagent co-evolutionary framework, enabling agents to cooperatively optimize the global objective in a distributed manner. Empirical studies show that the proposed algorithm achieves comparable solution quality with the holistic algorithm and better performance than existing gradient-free distributed algorithms on gradient-uncomputable problems.**

***Index Terms*—Distributed optimization, evolutionary computation (EC), multiagent systems, penalty function.**

Manuscript received 20 December 2023; accepted 13 March 2024. Date of publication 16 April 2024; date of current version 18 June 2024. This work was supported in part by the National Natural Science Foundation of China under Grant U23B2058 and Grant 62376097; in part by the Guangdong Regional Joint Foundation Key Project under Grant 2022B1515120076; and in part by the Research Fund of Hanyang University under Grant HY-202300000003465. This article was recommended by Associate Editor P. N. Suganthan. *(Corresponding author: Wei-Neng Chen.)*

Tai-You Chen, Wei-Neng Chen, and Xiao-Qi Guo are with the School of Computer Science and Engineering and the State Key Laboratory of Subtropical Building and Urban Science, South China University of Technology, Guangzhou 510006, China (e-mail: 575583114@qq.com; cschenwn@scut.edu.cn; kallyqi@outlook.com).

Yue-Jiao Gong is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: gongyuejiao@gmail.com).

Jun Zhang is with Nankai University, Tianjin 300071, China, also with Hanyang University, ERICA, Ansan, South Korea, and also with Victoria University, Melbourne, VIC, Australia (e-mail: junzhang@ieee.org).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TSMC.2024.3380389.

Digital Object Identifier 10.1109/TSMC.2024.3380389

I. Introduction

In RECENT years, lots of networked systems have emerged with the rapid development of communication networks, like robotics [1], smart grids [2], wireless sensor networks [3] and Internet of Vehicles [4]. A networked system is composed of multiple agents connected through a network. In general, there exists many systematic problems and tasks to be solved in networked systems, where the problem information or data is distributed among multiple agents [5], [6]. Facing such problems, distributed optimization, the process of achieving the global objective of a problem through communication and cooperation between agents, has attracted a considerable amount of research attention [7], [8].

In the literature, researchers have developed various approaches for distributed optimization. There are two types of existing methods, gradient-based methods and gradient-free methods. Gradient-based methods use gradient information (or second-order Hessian information) and consensus theory to solve distributed optimization problems with known mathematical expressions. A foundational gradient-based method is distributed subgradient descent (DGD) [9] proposed by Nedic and Ozdaglar, which alternates subgradient descent locally and weighted averaging among neighbors during optimization. After that, a lot of numerical distributed optimization methods have sprung up [10], [11], [12], improving the convergence speed, stability, and accuracy of distributed optimization. For example, Shi et al. [13] proposed an exact first-order algorithm, which can converge exactly with a fixed step-size. A proportional-integral control strategy proposed by Wang and Elia [14] can be applied in practical scenarios operated in continuous-time.

Considering that gradient information may be computationally infeasible or expensive in complex scenarios, a few gradient-free methods are proposed. For example, randomized gradient-free methods (RGFs) [15], [16] use a similar idea as gradient-based methods, but estimate gradient by a random gradient-free oracle. It guarantees that each agent can achieve approximate solutions when local objective functions are convex and Lipschitz continuous. Both RGFs and gradient-based methods have specific mathematical assumptions about objective functions, such as convexity or differentiability. However, many optimization problems in real-world networked systems are black-box, nonconvex

or discrete, such as task allocation [17], [18] in multiple unmanned aerial vehicle (UAV) systems. Such problems pose a great challenge to existing distributed optimization algorithms.

Evolutionary computation (EC) is another class of gradient-free methods inspired by biological evolution and group behaviors. Compared with gradient-based methods and RGFs, EC does not rely on the mathematical characteristic and gradient computation of problems [19]. EC has strong global search ability and is often used in black-box and nonconvex optimization [20], [21]. Due to the intrinsic parallelism of EC, many parallel and distributed evolutionary algorithms have been proposed, including population-distributed models and dimension-distributed models [22]. As their names suggest, population-distributed models divide the population into several subpopulations and evolve them in different computing nodes cooperatively [23], [24]. Dimension-distributed models divide decision variables of the global problem into several subcomponents and solve them by cooperative optimizers [25], [26], [27], [28]. These algorithms mainly study the variable division, communication protocol and cooperation mechanism between multiple subpopulations, enabling multiple computing nodes to cooperate to solve a complex problem efficiently.

Although existing parallel evolutionary algorithms provide rich cooperation mechanisms for multipopulations co-evolution, they are mainly designed for accelerating centralized optimization, where the global information of problems is usually accessible directly in each computing node. In distributed optimization problems of networked systems, each agent can only access a part of global objective function, that is, the local objective function. The decentralization and conflict of local objective functions pose two challenges to the design of evolutionary algorithms: 1) How to guide subpopulations to optimize the global objective cooperatively? and 2) How to negotiate to make distributed agents reach a consensus with a reasonable communication cost? There are only a few distributed evolutionary algorithms proposed for distributed optimization. For example, facing the first challenge, distributed particle swarm optimization algorithm (DPSO) [29] uses an auxiliary variable to estimate the global fitness when evolving subpopulations. Facing the second challenge, in [30], each agent calculates the weighted average of its subpopulation and that of its neighbors per generation. However, most of them are developed for specific applications [17], [31], [32] or low-dimension problems [29], [30].

In conclusion, there are still few studies for black-box and nonconvex distributed optimization with conflicting local objectives. Although distributed evolutionary algorithms may potentially solve this problem, existing algorithms do not well tackle the above two challenges on more general and larger-scale problems. In this work, we address a common type of distributed optimization, network-based distributed optimization (NDO), which exists in traffic assignment in transportation networks [33], electric dispatching in power systems [2], distributed power system state estimation [34], etc. In these scenarios, data is usually scattered across different nodes, prompting the importance of designing distributed optimization methods. Each agent in NDO represents a vertex on the network graph and has a private local objective function. The decision variables of the local objective function consist of private variables on the vertex and shared variables on edges. The variables on an edge are shared by two agents of both vertices. Addressing the above challenges, we propose a multiagent co-evolutionary algorithm with penalty-based objective (MACPO) to solve NDO. In MACPO, each agent of the networked system controls a subpopulation, where individuals are encoded by private variables of this agent and shared variables on its edges. During optimization, an agent evolves its subpopulation at the phase of local optimization and negotiates the values of shared variables with neighbors through communication at the phase of negotiation alternately. The main contributions are shown in two aspects.

1) To help agents optimize the global objective cooperatively when each agent can only access its local objective function, we design a penalty-based objective function for fitness evaluation to guide subpopulations. Specifically, a penalty term is added to the original local objective function. The penalty-based objective constrains the subpopulation within a small and controllable range near the consensus solution. In this way, the solutions on shared variables from neighboring agents are closer, which is conducive to reaching a consensus. Further, to make the penalty more targeted, we propose a conflict detection method to detect whether two agents conflict on a certain decision variable. The penalty term about a decision variable is added to penalty-based objective functions only if there is a conflict on this variable.
2) To help agents to reach a consensus on a global solution when each agent can only access its local objective function, we retrofit the processes of negotiating shared variables, namely, evaluation, competition, and sharing. These three steps enable adjacent agents in a networked system to negotiate a solution and reach a consensus on shared variables.

Most population-based evolutionary algorithms can be embedded into this framework. MACPO is tested on a set of benchmark functions for NDO. Empirical studies show that the proposed penalty-based objective function is effective in guiding the evolution of subpopulations compared to existing divide-and-conquer algorithms. When compared with the existing holistic algorithm, our algorithm achieves competitive results in terms of solution quality, and possesses higher-computational efficiency and better scalability. On the tested gradient-computable problems, the solution quality of MACPO is close to gradient-based distributed algorithms, while gradient-based algorithms are more stable. For the tested gradient-uncomputable real-world and benchmark problems, MACPO has better-solution quality and similar computational cost when compared with gradient-free distributed algorithms.

The remainder of this article is organized as follows. Section II formulates NDO mathematically. The proposed algorithm MACPO is described in Section III. In Section IV, a series of experiments are carried out. Finally, Section V concludes this article.

II. Network-Based Distributed Optimization

In general, distributed optimization is defined as

$$\min_{\boldsymbol{x}\in\mathbb{R}^D} \; F(\boldsymbol{x}) = \sum_{i=1}^{n} f_i(\boldsymbol{x}) \tag{1}$$

where $\boldsymbol{x}$ is decision variables and f_i is the local objective function of agent i. The global objective function F is the sum of all local objective functions. The ultimate goal of the system is to minimize the global objective function, which is a single-objective problem.

NDO is a special form of distributed optimization. In some multiagent systems, decision variables exist not only on agents but also on connections between agents. For example, in network flow problems in transportation network [33] and electric power systems [2], the flow between two agents is the shared variable that affects both of them, while the storage capacity or consumption speed inside an agent is the private variable. This kind of problem takes into account the heterogeneity and correlation of agents, which can be used to model many practical scenarios. More examples exist in Web page categorization [35], Web search ranking [36], disease progression prediction [37], node-specific active noise control [38], distributed power system state estimation [34], etc. In many cases, solving such problems requires massive data at each node, such as planning power supply through historical travel demand and vehicle information of electric vehicles in each city. Distributed optimization does not need to aggregate data from nodes and is promising to provide better-communication efficiency and data security.

The mathematical definition of NDO is given as follows. Consider a network of n agents denoted by an undirected graph $G = (V, E)$, where vertices $i \in V$ represent agents and edges $(i, j) \in E$ represent connections between agents. Let $X = [x^1, x^2, \ldots, x^D]$ denotes the global variables of NDO. For agent i, its local objective is determined by variables on vertex i and edges connected to vertex i. Therefore, local variables of agent i are defined as

$$X_i = \mathrm{col}\left\{x^d\right\}_{d\in\Theta_i} \text{ where } \Theta_i = \left\{\mathcal{I}_i, \bigcup_{j\in\mathcal{N}_i} \mathcal{I}_{i,j}\right\}. \tag{2}$$

Here, $\mathcal{I}_i \subseteq \{1, 2, \ldots, D\}$ denotes the index set of variables on vertex i, which is also called private variables of agent i. And $\mathcal{I}_{i,j} \subseteq \{1, 2, \ldots, D\}$ denotes the index set of variables on edge (i, j), which is also called shared variables between agent i and j. Θ_i is the index set of local variables of agent i. Take the system in Fig. 1 as an example, the decision variable set of agent 1 is the union of variables on node 1, edge $(1, 2)$, edge $(1, 3)$ and edge $(1, 4)$. That is, $\Theta_1 = \mathcal{I}_1 \cup \mathcal{I}_{1,2} \cup \mathcal{I}_{1,3} \cup \mathcal{I}_{1,4}$.

Then, the global objective function of NDO is defined as

$$\min_{X\in\mathbb{R}^D} \; F(X) = \sum_{i=1}^{n} f_i(X_i) \tag{3}$$

where function $f_i : \mathbb{R}^{\|X_i\|} \to \mathbb{R}$ is the local objective function of agent i and the global objective function is the sum of all local objective functions.

In a distributed manner, NDO defined in (3) is equivalent to

$$\begin{aligned} \min_{X_1, X_2, \ldots, X_n} \; & \sum_{i=1}^{n} f_i(X_i) \\ \text{s.t.} \; & X_i = \mathrm{col}\left\{x_i^d\right\}_{d\in\Theta_i} \\ & \Theta_i = \left\{\mathcal{I}_i, \bigcup_{j\in\mathcal{N}_i} \mathcal{I}_{i,j}\right\} \\ & x_i^d = x_j^d \quad \forall d \in \mathcal{I}_{i,j} \; \forall (i, j) \in E. \end{aligned} \tag{4}$$

Here, each variable x^d on edge (i, j) is denoted as two distinct variables, x_i^d and x_j^d. This formulation is more suitable for explaining the algorithm in the distributed environment.

In complex scenarios, private variables and shared variables in the local objective function are coupled, so it is impossible to optimize shared variables and private variables separately. Moreover, due to the transitivity of the network, all agents in the whole network need to cooperate to solve the optimization problem, instead of solving the problem individually.

III. Multiagent Co-Evolutionary Algorithm With Penalty-Based Objective

In this section, the proposed multiagent co-evolutionary algorithm framework is first introduced. Then, we introduce three parts of this framework in the following order, penalty-based local optimization, negotiation and conflict detection.

A. Multiagent Co-Evolutionary Algorithm Framework

In our work, we propose a multiagent co-evolutionary algorithm framework. In this framework, each agent represents a node of the networked system, so that it can access the local objective of this node. Each agent maintains a population and coevolve with its neighbors. The flowchart of each agent and the communication between agents are shown in Fig. 1. The communication topology of the multiagent system is generated according to the solved problem. A communication relationship could be established between two discrete agents i and j if they are connected in the network G. The neighbor set of agent i is defined as follows:

$$\mathcal{N}_i := \{j \in V \backslash \{i\} : (i, j) \in E\}. \tag{5}$$

At each iteration, the steps of agents consist of two parts, local optimization and negotiation. As the name suggests, local optimization refers to the internal computation phase of each agent. Negotiation refers to the process that neighboring agents reach a consensus on their shared variables and guide the population to learn from the consensus solution.

In the local optimization phase, each agent optimizes its decision variables X_i toward the penalty-based objective h_i. For agent i, the optimization objective h_i is initially set to its original local objective f_i and changes after each time of negotiation. It is worth noting that the optimization algorithm of the local optimization phase can be any population-based evolutionary algorithm. This is because this framework only

embedded EC		LLSO						CSO					
Functions		MACPO	MACPO-NC	MAC-CC	MAC-FEA	DPSO	GFPDO	MACPO	MACPO-NC	MAC-CC	MAC-FEA	DPSO	GFPDO
F1	mean	1.75E+08	2.14E+08	4.41E+08	1.16E+09	2.25E+10	8.46E+08	3.42E+08	3.87E+08	1.11E+09	1.83E+09	3.65E+10	2.06E+10
	median	1.77E+08	2.05E+08	4.01E+08	1.02E+09	2.23E+10	8.29E+08	3.27E+08	3.76E+08	1.05E+09	1.70E+09	3.72E+10	2.06E+10
	std	1.62E+07	3.62E+07	1.74E+08	3.10E+08	2.66E+09	8.46E+07	6.52E+07	9.29E+07	2.53E+08	6.81E+08	4.15E+09	3.15E+09
	p-value	-	5.66e-03#	1.24e-05#	2.73e-06#	9.13e-05#	5.46e-06#	-	2.60E-01	9.13e-05#	9.13e-05#	9.13e-05#	5.46e-06#
F2	mean	7.65E+05	9.00E+05	1.25E+07	8.07E+07	5.61E+10	9.43E+06	1.41E+06	1.46E+06	1.91E+07	1.43E+08	2.69E+10	2.37E+09
	median	7.64E+05	9.01E+05	1.65E+06	3.88E+07	1.34E+10	9.26E+06	1.46E+06	1.47E+06	2.17E+06	6.63E+07	5.56E+09	7.07E+08
	std	6.64E+04	4.72E+04	1.68E+07	1.15E+08	9.39E+10	4.13E+06	1.11E+05	1.41E+05	4.99E+07	2.13E+08	6.51E+10	4.34E+09
	p-value	-	3.84e-04#	2.73e-06#	2.73e-06#	9.13e-05#	5.46e-06#	-	2.10E-01	6.57e-04#	9.13e-05#	9.13e-05#	5.46e-06#
F3	mean	2.13E+10	2.31E+10	7.11E+10	1.18E+11	3.86E+10	5.86E+10	2.08E+10	2.22E+10	6.60E+10	1.55E+11	3.99E+10	1.04E+11
	median	2.13E+10	2.26E+10	7.21E+10	1.14E+11	3.87E+10	5.83E+10	2.07E+10	2.22E+10	6.54E+10	1.53E+11	3.97E+10	1.03E+11
	std	8.53E+08	1.52E+09	4.29E+09	1.36E+10	5.62E+08	4.22E+09	6.64E+08	1.07E+09	8.87E+09	2.88E+10	1.12E+09	3.92E+09
	p-value	-	2.29e-03#	2.73e-06#	2.73e-06#	9.13e-05#	5.46e-06#	-	8.53e-04#	9.13e-05#	9.13e-05#	9.13e-05#	5.46e-06#
F4	mean	2.70E+07	4.01E+07	8.91E+07	8.43E+09	1.68E+10	4.53E+08	1.02E+08	1.24E+08	4.98E+08	1.45E+09	1.89E+10	8.62E+09
	median	2.54E+07	3.62E+07	8.78E+07	2.93E+09	1.67E+10	4.67E+08	8.95E+07	1.35E+08	4.46E+08	1.49E+09	1.82E+10	7.43E+09
	std	5.78E+06	1.34E+07	2.58E+07	1.94E+10	2.33E+09	6.74E+07	2.42E+07	2.37E+07	2.80E+08	4.29E+08	4.13E+09	3.16E+09
	p-value	-	7.01e-03#	2.73e-06#	2.73e-06#	9.13e-05#	5.46e-06#	-	3.20e-02#	9.13e-05#	9.13e-05#	9.13e-05#	5.46e-06#
F5	mean	6.58E+08	5.42E+08	2.17E+09	1.02E+10	2.65E+10	2.11E+10	3.31E+08	3.97E+08	5.45E+09	2.53E+10	4.30E+10	5.60E+10
	median	5.34E+08	4.86E+08	1.99E+09	9.83E+09	2.68E+10	2.05E+10	3.27E+08	3.85E+08	5.81E+09	2.41E+10	4.31E+10	5.57E+10
	std	3.20E+08	2.15E+08	7.41E+08	1.90E+09	1.87E+09	2.24E+09	3.51E+07	6.25E+07	1.30E+09	6.79E+09	4.11E+09	3.77E+09
	p-value	-	2.40E-01	3.86e-06#	2.73e-06#	9.13e-05#	5.46e-06#	-	4.55e-03#	9.13e-05#	9.13e-05#	9.13e-05#	5.46e-06#
F6	mean	2.39E+08	2.12E+08	5.91E+09	7.82E+09	2.72E+10	4.00E+10	1.61E+08	2.69E+08	4.82E+10	1.23E+11	4.00E+10	5.23E+10
	median	2.17E+08	2.04E+08	4.17E+09	5.42E+09	2.71E+10	3.93E+10	1.52E+08	2.41E+08	4.31E+10	1.20E+11	3.27E+10	5.22E+10
	std	7.20E+07	4.75E+07	4.21E+09	7.01E+09	2.87E+09	4.31E+09	3.04E+07	1.06E+08	2.35E+10	3.93E+10	2.24E+10	4.16E+09
	p-value	-	1.90E-01	2.73e-06#	2.73e-06#	9.13e-05#	5.46e-06#	-	5.66e-03#	9.13e-05#	9.13e-05#	9.13e-05#	5.46e-06#
w/t/l		-	4/2/0	6/0/0	6/0/0	6/0/0	6/0/0	-	4/2/0	6/0/0	6/0/0	6/0/0	6/0/0

[1] In this table, MACPO-NC and MACPO are algorithms proposed in this work. MAC-CC is the algorithm generated by adopting CCEA in our framework, and MAC-FEA is the algorithm generated by adopting FEA in our framework.
[#] The experiment results of the compared algorithm are significantly worse than MACPO according to the Wilcoxon rank sum test of level 0.05.
[*] The experiment results of the compared algorithm are significantly better than MACPO according to the Wilcoxon rank sum test of level 0.05.

Fig. 1. Framework of MACPO. This figure shows an example of a networked system with six agents. Each green circle represents an agent. Because agent 1 and agent 2 are connected, the flowchart of these two agents is presented to show how the two connected agents communicate and negotiate in MACPO. Green lines between two flowcharts represent the communication between agent 1 and agent 2. *Symbol note:* x^i is the ith variable of the global variables of NDO. $\mathcal{I}_i$ denotes the index set of private variables on vertex i, while $\mathcal{I}_{i,j}$ denotes the index set of shared variables on edge (i, j). The detailed definition is shown in Section II. h_i is the penalty-based objective function proposed in Section III-B.

modifies the objective function and assigns values to the population when negotiating.

In the negotiation phase, there are four steps for agents to negotiate with neighbors, that are evaluation, competition, sharing, and conflict detection. It costs three times of communication for agent i to negotiate the solution of shared variables $\mathcal{I}_{i,j}$ with neighbor j per iteration. For the first time, agents send their best solutions, i.e., candidate solutions, to their neighbors at the evaluation step. At the second time, agents will send two evaluation results to neighbors at the competition step. The third time of communication occurs at the step of conflict detection when agents send results of gradient estimation to neighbors.

B. Penalty-Based Local Optimization

As we have introduced before, the algorithm alternates between the local optimization phase and the negotiation phase. According to Fig. 1, the penalty-based objective function h_i is initialized as the original local objective function f_i at the beginning. It will be updated at each iteration after the consensus solution $\boldsymbol{x}_{i,\text{con}}$ is obtained. The design of penalty-based objective function h_i related to $\boldsymbol{x}_{i,\text{con}}$ will be introduced in this section.

The penalty-based objective function is designed by adding a penalty function to the original local objective. The formulation of penalty-based objective is

$$h_i(x) = f_i(x) + w * \sum_{j\in\mathcal{N}_i}\sum_{d\in\mathcal{I}_{i,j}}\left(t_{j,d}\left|\boldsymbol{x}^d - \boldsymbol{x}^d_{i,\text{con}}\right|\right) \tag{6}$$

where $w \in \mathbb{R}$ is a penalty weight, $t_{j,d} \in \{0, 1\}$ is a penalty switch, and $\boldsymbol{x}_{i,\text{con}}$ is the consensus solution updated at each iteration. The penalty item in the penalty-based objective reflects the distance between the evaluated individual x and the consensus solution $\boldsymbol{x}_{i,\text{con}}$.

The penalty-based objective can constrain the population within a small range near the consensus solution. Therefore, the conflict between the candidate solutions from different agents is small in the next negotiation phase, which helps the negotiation and cooperation of neighbors. On one hand, the solutions on shared variables are close, enabling neighboring agents to reach a consensus more easily. On the other hand, the private variables can match shared variables better when the solution of shared variables is replaced by the consensus solution.

The penalty-based objective has two important parameters, penalty weight w and penalty switch $t_{j,d}$. The effect and setting of these two parameters are introduced as follows.

1) Penalty Weight: The penalty weight has an important influence on the performance of MACPO. If the penalty weight is too large, the fitness will be dominated by the penalty item and the original objective will be ignored. On the contrary, if it is too small, the penalty item will be ignored and lose its due effect. Since the original fitness of the problem will decrease with the optimization process, it is inappropriate to use a fixed penalty weight. In this work, we set the penalty weight adaptively, which changes with the original fitness of the problem

$$w = \lambda \sum_{i=1}^{n} f_i\left(\boldsymbol{x}_{i,\text{con}}\right) \tag{7}$$

here, λ is a static parameter, which is analyzed in parameter experiments. The item $\sum_{i=1}^{n} f_i(\boldsymbol{x}_{i,\text{con}})$ is the sum of agents' local fitness of current consensus solutions. As the optimization goes on, the fitness of consensus solutions will decrease, and the penalty weight w will decrease accordingly.

This adaptive penalty weight can balance the relative size of the penalty item and original objective value, making the penalty-based objective better guide the search of the population. The penalty weight is updated periodically by a broadcast approach.

2) Penalty Switch: In (6), $t_{j,d}$ is the penalty switch, which is set to 0 or 1. It determines whether to constrain the evolution of the dth shared variable with neighbor j in the current population. When the penalty switch is set to 1, the penalty-based objective will be relative to the distance between x and $\boldsymbol{x}_{i,\text{con}}$ on the dth shared variable. Therefore, the solution far from the consensus solution on that variable is not preferred because its fitness will increase. In this case, the evolution of the agent on that variable is constrained. This variable is dynamically adjusted during the optimization, which will be introduced in Section III-C3.

C. Negotiation

In the proposed framework, we use the first three steps of the negotiation phase, which are evaluation, competition, and sharing, to achieve consensus among different agents. The negotiation methods in this work can achieve the consensus on shared variables in a distributed scenario, where each agent can only conduct local fitness evaluations.

1) Evaluation: At the beginning of the evaluation step, each agent will send its candidate solution, the best solution in the subpopulation, to its neighbors, as shown in Fig. 1. After receiving neighbors' candidate solutions, agents will evaluate them. The idea of evaluation is to compute the local fitness when a certain dimension of the current solution is replaced by the neighbor's solution. Algorithm 1 returns two vectors, $\text{fit}_{i,i}$ and $\text{fit}_{i,j}$. The value $\text{fit}^d_{i,i}$ in the first vector refers to the fitness of subproblem f_i if the consensus solution adopts $\boldsymbol{x}_i$'s value on the dth shared variable. Similarly, the value $\text{fit}^d_{i,j}$ in the second vector refers to the fitness of subproblem f_i if the consensus solution adopts $\boldsymbol{x}_{j,\text{best}}$'s value on the dth shared variable.

Considering the interaction of shared variables, the value of shared variables in consensus solution $\boldsymbol{x}_{i,test}$ is initialized as the candidate solution from the smaller-index agent (lines 3 and 4). This operation guarantees that $\text{fit}_{i,j}$ and $\text{fit}_{j,j}$ are evaluating the same solution for shared variables. Then, for each shared variable, its value will be replaced by the solution from the higher-index agent (a1), and the fitness of $\boldsymbol{x}_{i,test}$ is computed as $\text{fit}^d_{i,b}$ (a2). After that, the value of the dth shared variable is replaced back. The fitness $\text{fit}^d_{i,a}$ does not need to be computed repeatedly for the reason that it equals the initial fitness fit_0.

2) Competition: In the competition step, adjacent agents exchange the evaluation results through communication first. And then they will conduct the competition operation in Algorithm 2. For shared variables in $\mathcal{I}_{i,j}$, agent i competes with agent j through four vectors, $\text{fit}_{i,i}$, $\text{fit}_{i,j}$, $\text{fit}_{j,i}$, and $\text{fit}_{j,j}$. The idea of competition is to determine the consensus solution according to global fitness. The global fitness of shared variables in $\mathcal{I}_{i,j}$ can be represented as the sum of f_i and f_j because they only exist in these two agents. Thus, the global fitness for $\boldsymbol{x}_{i,\text{best}}$ on dth shared variable is $\text{fit}^d_{i,i} + \text{fit}^d_{j,i}$, and the global fitness for $\boldsymbol{x}_{j,\text{best}}$ on dth shared variable is $\text{fit}^d_{i,j} + \text{fit}^d_{j,j}$.

Algorithm 1 Evaluation in MACPO

Input: local solution $\boldsymbol{x}_{i,best}$, indices of decision variables Θ_i, indices of shared variables $\mathcal{I}_{i,j}$

1: send $\boldsymbol{x}_{i,best}$ to each agent $j \in \mathcal{N}_i$
2: receive $\boldsymbol{x}_{j,best}$ from each agent $j \in \mathcal{N}_i$
3: let $a = min(i,j)$, $b = max(i,j)$
4: let $\boldsymbol{x}_{i,test} = \boldsymbol{x}^{\Theta_i \setminus \mathcal{I}_{i,j}}_{i,best} \cup \boldsymbol{x}^{\mathcal{I}_{i,j}}_{a,best}$
5: $fit_0 = f_i(\boldsymbol{x}_{i,test})$
6: let $fit_{i,i} = fit_{i,j} = 0^{1\times\|\mathcal{I}_{i,j}\|}$
7: **for** each d in $\mathcal{I}_{i,j}$ **do**
8: $\quad \boldsymbol{x}^d_{i,test} = \boldsymbol{x}^d_{b,best}$ (a1)
9: $\quad fit^d_{i,b} = f_i(\boldsymbol{x}_{i,test})$ (a2)
10: $\quad \boldsymbol{x}^d_{i,test} = \boldsymbol{x}^d_{a,best}$ (a3)
11: $\quad fit^d_{i,a} = fit_0$ (a4)
12: **end for**

Output: $fit_{i,i}, fit_{i,j}$

Algorithm 2 Competition in MACPO

Input: local solution $\boldsymbol{x}_{i,best}$, neighbor j's solution $\boldsymbol{x}_{j,best}$, indices of shared variables $\mathcal{I}_{i,j}$

1: send $fit_{i,i}, fit_{i,j}$ to each agent $j \in \mathcal{N}_i$
2: receive $fit_{j,j}, fit_{j,i}$ from each agent $j \in \mathcal{N}_i$
3: $\boldsymbol{x}_{i,con} = \boldsymbol{x}_{i,best}$
4: $res_j = 0^{1\times\|\mathcal{I}_{i,j}\|}$
5: **for** each d in $\mathcal{I}_{i,j}$ **do**
6: $\quad$ **if** $fit^d_{i,i} + fit^d_{j,i} > fit^d_{i,j} + fit^d_{j,j}$ **then**
7: $\quad\quad \boldsymbol{x}^d_{i,con} = \boldsymbol{x}^d_{j,best}$ (a5)
8: $\quad\quad res^d_j = 0$ (a6)
9: $\quad$ **else**
10: $\quad\quad res^d_j = 1$ (a7)
11: $\quad$ **end if**
12: **end for**

Output: $\boldsymbol{x}_{i,con}$, res_j

Also, except for the selection of consensus solution (a5), the competition result is also recorded in res_j (a6, a7), which is useful in the sharing operation.

3) Sharing: After the competition step, the new consensus solution will be used to guide the evolution of the population in the next iteration. In Algorithm 3, the guidance has two aspects, updating population and objective function. Generally speaking, we expect the winning side in the competition to guide the evolution of shared variables. Therefore, in our algorithm, the winning side in the competition will not be constrained in the next local optimization phase and the penalty switch $t_{j,d}$ of the penalty-based objective function is set to 0 (a8). Instead, the losing side in the competition will be constrained, and the penalty switch is set to 1 (a9). What is more, the population's value on the variable will be replaced by the consensus solution if the current agent fails (a10).

Remark 1 (Consensus Discussion): The above three steps of the negotiation phase confirm that neighbor agents can reach a consensus on their shared decision variables, that is

$$\boldsymbol{x}^d_{i,\text{con}} = \boldsymbol{x}^d_{j,\text{con}} \quad \forall d \in \mathcal{I}_{i,j}\ \forall (i,j) \in E.$$

Algorithm 3 Sharing in MACPO

Input: consensus solution $\boldsymbol{x}_{i,con}$, population $\mathcal{P}_i$, competition result res

for each j in $\mathcal{N}_i$ **do**
  for each d in $\mathcal{I}_{i,j}$ **do**
    if $res_j^d == 1$ **then**
      $t_{j,d} = 0$ (a8)
    else
      $t_{j,d} = 1$ (a9)
      for each $\boldsymbol{x}$ in $\mathcal{P}_i$ **do**
        $\boldsymbol{x}^d = \boldsymbol{x}_{i,con}^d$ (a10)
      end for
    end if
  end for
end for
update h_i

Output: $\mathcal{P}_i$, h_i

In NDO, shared decision variables are defined on the edge of two agents, shared by these two agents. At each iteration, there are two candidate solutions for a shared variable. The two agents sharing this variable need to choose the better one to reach a consensus. In the proposed algorithm, two neighbor agents can make the same decision at the competition step by exchanging evaluation results. According to Algorithm 2, agents choose the solution for shared variables based on the size relationship between $\text{fit}_{i,i}^d + \text{fit}_{j,i}^d$ and $\text{fit}_{i,j}^d + \text{fit}_{j,j}^d$. Therefore, MACPO enables to reach consensus in a distributed environment.

D. Conflict Detection

Generally speaking, the penalty item in the penalty-based objective is designed to help neighboring agents optimize in the same direction, reducing the difference of shared variables due to the conflict of original local objectives. However, two subproblems do not always conflict or never conflict at different positions in the solution space. Their conflict condition is dynamic and changes with the optimization process. Thus, if there is little or no conflict between two neighboring agents at some point, then the penalty item in the penalty-based objective can be removed.

Therefore, we design an additional step of the negotiation phase, conflict detection, to adaptively adjust the penalty switch by gradient estimation. To be specific, if the estimated gradients of two agents on a shared variable are in the same direction, it is considered that they will optimize toward the same direction autonomously. Therefore, the penalty item will be removed in this case.

Fig. 2 shows an example of a one-dimension distributed problem. According to the figure, f_1 and f_2 are two conflicting subproblems in the global problem F. When the consensus solution is located at $x = -150$, the gradient direction of f_1 and f_2 are the same. Thus, the penalty item will be removed. On contrast, when the consensus solution is located at $x = -100$, their gradient direction is opposite, then the penalty item will be reserved in the penalty-based objective.

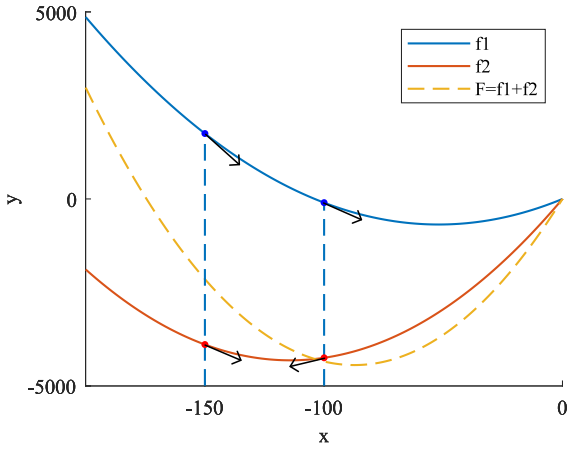


Fig. 2. Illustrate of the conflict detection.

Algorithm 4 Conflict Detection in MACPO

Input: $\boldsymbol{x}_{i,con}$, indices of shared variables $\mathcal{I}_{i,j}$

for each j in $\mathcal{N}_i$ **do**
  for each d in $\mathcal{I}_{i,j}$ **do**
    $p_{i,j}^d = \triangle_{\delta,\boldsymbol{x}_d}[f_i](\boldsymbol{x}_{i,con})$ (a11)
    $n_{i,j}^d = \triangle_{-\delta,\boldsymbol{x}_d}[f_i](\boldsymbol{x}_{i,con})$ (a12)
  end for
  send $p_{i,j}$, $n_{i,j}$ to agent j
end for
for each j in $\mathcal{N}_i$ **do**
  receive $p_{j,i}$, $n_{j,i}$ from agent j
  for each d in $\mathcal{I}_{i,j}$ **do**
    if $p_{j,i}^d * p_{i,j}^d > 0$ and $n_{j,i}^d * n_{i,j}^d > 0$ **then**
      $t_{j,d} = 0$ (a13)
    end if
  end for
end for
update h_i

Output: h_i

The estimate of the gradient for black-box functions is designed as

$$\triangle_{\delta,\boldsymbol{x}_d}[f_i](\boldsymbol{x}) = f_i(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d + \delta, \ldots, \boldsymbol{x}_D) - f_i(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d, \ldots, \boldsymbol{x}_D) \quad (8)$$

where δ is the disturbance for a variable, which is set based on the variable scope. For a variable with the scope $[a, b]$, δ is set to $\theta(b - a)$, where θ is a weight parameter.

According to Algorithm 4, each agent will perform gradient estimation for each shared variable (a11, a12) and then send the estimation result to the corresponding neighbor. After receiving estimation information from neighbors, agents will compare it with its estimated gradient. If they are in the same direction, the penalty item will be removed (a13).

IV. Experiment

In this section, we compare the proposed algorithm with two popular divide-and-conquer algorithms, cooperative co-evolution evolutionary algorithm (CCEA) and factored evolutionary algorithm (FEA), in Section IV-C. In Section IV-D,

a holistic optimization algorithm implemented on the master-slave model is compared with our algorithm. For a fair comparison, each experiment was repeated 25 times.

A. Experiment Settings

In general, a problem with a larger dimension size requires more generations for the population to search the dimension space. Therefore, the number of generations is set to $0.4D$, where D is the dimension size of the agent. We conduct the parameter investigation experiment on the parameter λ in (7) and θ in (8), which is shown in Part III of the supplementary material. According to experiment results, the parameter λ of penalty weight is set to $1/512$, and θ is set to 0.05%.

For the following experiments, we design 18 benchmark functions F1-F18 from the perspective of problem scale, homogeneity and heterogeneity, elementary function type, and network topology, which is shown in Part II of the supplementary material.[1]

Both MACPO and the compared algorithms are general algorithm frameworks, which can be embedded by different population-based evolutionary algorithms. In the experiment, level-based learning swarm optimizer (LLSO) [39] and competitive swarm optimizer (CSO) [40] are adopted as the unified optimizer, because both of them perform well in large-scale optimization.

The algorithms are implemented by message passing interface (MPI) in C++ language. The experiments are conducted on a cluster composed of Intel Xeon CPU E5-2699 v3@2.30-GHz 36 cores and CentOS 7.5 x64 system.

B. Effectiveness of Conflict Detection

In order to verify the effectiveness of conflict detection, we compare the performance of MACPO and its variant, MACPO with no conflict detection (MACPO-NC). The experiment results on F1-F6 are shown in Table I. The only difference between MACPO-NC and MACPO is that MACPO-NC does not contain the "conflict detection" step. According to the experiment result, when the embedded optimizer is LLSO, MACPO performs better than MACPO-NC on 4 of 6 functions and performs similarly to MACPO-NC on 2 of 6 functions. When the embedded optimizer is CSO, the performance is also the same. The experiment results show that conflict detection is effective.

MACPO performs better because the penalty item is removed when a shared variable does not cause a conflict. As mentioned earlier, the conflict condition of two subproblems is dynamic during the optimization process. Therefore, MACPO uses conflict detection to detect the conflict on each variable and better helps agents to cooperate. According to the experiment results, conflict detection is effective. Conflict detection can relax guidance by the penalty item when there is no conflict between subproblems, making agents optimize more freely. Therefore, agents are more likely to find good solutions.

[1]The implementation of benchmark functions is shown in the link: *https://github.com/iamrice/Supplementary-materials-of-Multi-Agent-Co-evolutionary-Algorithm-with-Penalty-Based-Objective-MACPO-*.

Fig. S2 in the supplementary material shows the conflict probability of shared variables during the optimization, which is detected by the proposed method. According to the figures, the conflict probabilities in all functions show a roughly ascending trend. It indicates that the conflict degree of local objective functions differs in different positions of the domain. In the early stage, the conflict probability is relatively low. Then, as the population approaches the optimal region in the latter stage, the conflict probability will gradually increase.

In MACPO, the penalty item of many shared variables is removed in the early stage because there is no conflict on these variables. In contrast, each shared variable always has a penalty item in MACPO-NC. This is the only difference between them. The optimization curve of MACPO and MACPO-NC is shown in Fig. S3 of the supplementary material. MACPO has an advantage over MACPO-NC from the early stage.

C. Comparison With Divide-and-Conquer Algorithms

In this section, we compare MACPO with two popular divide-and-conquer algorithms, CCEA and FEA. They divide the decision variables of the problem into multiple subcomponents and solve them separately, which is developed for centralized optimization. Most divide-and-conquer algorithms represented by CCEA and FEA do not consider the conflict of original local objectives, so they lack guidance at the phase of local optimization. Therefore, we propose the penalty-based objective function to guide the local optimization and make agents more globally conscious. The purpose of comparing MACPO with divide-and-conquer evolutionary algorithms is to show the effectiveness of the proposed penalty-based objective function.

For the reason that CCEA and FEA cannot handle NDO directly, we embedded the idea of CCEA and FEA into our algorithm framework and named them MAC-CC and MAC-FEA, respectively. In MAC-CC, a shared variable is optimized by only one agent at each iteration, and another agent fixes the values of shared variables. The allocation of shared variables is decided by competition results, that is, the winner of the competition can evolve the shared variable at the next iteration. Agents will still negotiate at each iteration, but the candidate solution provided by the loser is the consensus solution of the previous iteration. Except for the allocation of shared variables, other implementations of MAC-CC are the same as MACPO. In MAC-FEA, both agents will optimize the shared variables toward their original local objectives. That is, the penalty weight w is set to 0 in MAC-FEA. Except for the objective function, other implementations of MAC-FEA are the same as MACPO.

According to Table I, MACPO performs better than MAC-CC and MAC-FEA on F1-F6. In the following, we further analyze the performance difference among these three algorithms and explain the advantage of the penalty-based objective of MACPO.

1) Analysis of MAC-FEA: To explore the reasons for the performance difference between MAC-FEA and MACPO, we calculate the average distance between the local population

TABLE I
COMPARISON WITH VARIANT ALGORITHMS AND EXISTING ALGORITHMS [1]

embedded EC		LLSO						CSO					
Functions		MACPO	MACPO-NC	MAC-CC	MAC-FEA	DPSO	GFPDO	MACPO	MACPO-NC	MAC-CC	MAC-FEA	DPSO	GFPDO
F1	mean	1.75E+08	2.14E+08	4.41E+08	1.16E+09	2.25E+10	8.46E+08	3.42E+08	3.87E+08	1.11E+09	1.83E+09	3.65E+10	2.06E+10
	median	1.77E+08	2.05E+08	4.01E+08	1.02E+09	2.23E+10	8.29E+08	3.27E+08	3.76E+08	1.05E+09	1.70E+09	3.72E+10	2.06E+10
	std	1.62E+07	3.62E+07	1.74E+08	3.10E+08	2.66E+09	8.46E+07	6.52E+07	9.29E+07	2.53E+08	6.81E+08	4.15E+09	3.15E+09
	p-value	-	5.66e-03#	1.24e-05#	2.73e-06#	9.13e-05#	5.46e-06#	-	2.60E-01	9.13e-05#	9.13e-05#	9.13e-05#	5.46e-06#
F2	mean	7.65E+05	9.00E+05	1.25E+07	8.07E+07	5.61E+10	9.43E+06	1.41E+06	1.46E+06	1.91E+07	1.43E+08	2.69E+10	2.37E+09
	median	7.64E+05	9.01E+05	1.65E+06	3.88E+07	1.34E+10	9.26E+06	1.46E+06	1.47E+06	2.17E+06	6.63E+07	5.56E+09	7.07E+08
	std	6.64E+04	4.72E+04	1.68E+07	1.15E+08	9.39E+10	4.13E+06	1.11E+05	1.41E+05	4.99E+07	2.13E+08	6.51E+10	4.34E+09
	p-value	-	3.84e-04#	2.73e-06#	2.73e-06#	9.13e-05#	5.46e-06#	-	2.10E-01	6.57e-04#	9.13e-05#	9.13e-05#	5.46e-06#
F3	mean	2.13E+10	2.31E+10	7.11E+10	1.18E+11	3.86E+10	5.86E+10	2.08E+10	2.22E+10	6.60E+10	1.55E+11	3.99E+10	1.04E+11
	median	2.13E+10	2.26E+10	7.21E+10	1.14E+11	3.87E+10	5.83E+10	2.07E+10	2.22E+10	6.54E+10	1.53E+11	3.97E+10	1.03E+11
	std	8.53E+08	1.52E+09	4.29E+09	1.36E+10	5.62E+08	4.22E+09	6.64E+08	1.07E+09	8.87E+09	2.88E+10	1.12E+09	3.92E+09
	p-value	-	2.29e-03#	2.73e-06#	2.73e-06#	9.13e-05#	5.46e-06#	-	8.53e-04#	9.13e-05#	9.13e-05#	9.13e-05#	5.46e-06#
F4	mean	2.70E+07	4.01E+07	8.91E+07	8.43E+09	1.68E+10	4.53E+08	1.02E+08	1.24E+08	4.98E+08	1.45E+09	1.89E+10	8.62E+09
	median	2.54E+07	3.62E+07	8.78E+07	2.93E+09	1.67E+10	4.67E+08	8.95E+07	1.35E+08	4.46E+08	1.49E+09	1.82E+10	7.43E+09
	std	5.78E+06	1.34E+07	2.58E+07	1.94E+10	2.33E+09	6.74E+07	2.42E+07	2.37E+07	2.80E+08	4.29E+08	4.13E+09	3.16E+09
	p-value	-	7.01e-03#	2.73e-06#	2.73e-06#	9.13e-05#	5.46e-06#	-	3.20e-02#	9.13e-05#	9.13e-05#	9.13e-05#	5.46e-06#
F5	mean	6.58E+08	5.42E+08	2.17E+09	1.02E+10	2.65E+10	2.11E+10	3.31E+08	3.97E+08	5.45E+09	2.53E+10	4.30E+10	5.60E+10
	median	5.34E+08	4.86E+08	1.99E+09	9.83E+09	2.68E+10	2.05E+10	3.27E+08	3.85E+08	5.81E+09	2.41E+10	4.31E+10	5.57E+10
	std	3.20E+08	2.15E+08	7.41E+08	1.90E+09	1.87E+09	2.24E+09	3.51E+07	6.25E+07	1.30E+09	6.79E+09	4.11E+09	3.77E+09
	p-value	-	2.40E-01	3.86e-06#	2.73e-06#	9.13e-05#	5.46e-06#	-	4.55e-03#	9.13e-05#	9.13e-05#	9.13e-05#	5.46e-06#
F6	mean	2.39E+08	2.12E+08	5.91E+09	7.82E+09	2.72E+10	4.00E+10	1.61E+08	2.69E+08	4.82E+10	1.23E+11	4.00E+10	5.23E+10
	median	2.17E+08	2.04E+08	4.17E+09	5.42E+09	2.71E+10	3.93E+10	1.52E+08	2.41E+08	4.31E+10	1.20E+11	3.27E+10	5.22E+10
	std	7.20E+07	4.75E+07	4.21E+09	7.01E+09	2.87E+09	4.31E+09	3.04E+07	1.06E+08	2.35E+10	3.93E+10	2.24E+10	4.16E+09
	p-value	-	1.90E-01	2.73e-06#	2.73e-06#	9.13e-05#	5.46e-06#	-	5.66e-03#	9.13e-05#	9.13e-05#	9.13e-05#	5.46e-06#
w/t/l		-	4/2/0	6/0/0	6/0/0	6/0/0	6/0/0	-	4/2/0	6/0/0	6/0/0	6/0/0	6/0/0

[1] In this table, MACPO-NC and MACPO are algorithms proposed in this work. MAC-CC is the algorithm generated by adopting CCEA in our framework, and MAC-FEA is the algorithm generated by adopting FEA in our framework.
[#] The experiment results of the compared algorithm are significantly worse than MACPO according to the Wilcoxon rank sum test of level 0.05.
[*] The experiment results of the compared algorithm are significantly better than MACPO according to the Wilcoxon rank sum test of level 0.05.

and the consensus solution in the optimization process. The distance is computed as follows:

$$D(X) = \frac{1}{\mathrm{NP}} \sum_{i=1}^{\mathrm{NP}} \sqrt{\sum_{d \in \Theta_i \backslash \mathcal{I}_i} \left(\boldsymbol{x}_i^d - \boldsymbol{x}_{i,\mathrm{con}}^d\right)} \tag{9}$$

where $\boldsymbol{x}_{i,\mathrm{con}}$ is the consensus solution, and NP is the population size of an agent.

Fig. S4 in the supplementary material shows the trend of distance during 90 generations of the local optimization phase in F1-F6. In detail, "MACPO: win" and "MAC-FEA: win" show the distance of agents that win in the previous competition phase, and "MACPO: fail" and "MAC-FEA: fail" show the distance of the failed agents. According to the figures, the trends of distance are similar in MACPO and MAC-FEA when an agent wins in the competition. On the contrary, when an agent fails in the competition, the trends of distance differ greatly in these two algorithms. The distance between the population and consensus solution in MAC-FEA increases rapidly, while the distance in MACPO stays low.

When an agent wins in the competition, the trends of distance are similar in MACPO and MAC-FEA because their optimization objectives are both the original local objectives. Compared to the failed agents, the distance between the population and consensus solution remains within a small range during the evolution of local optimization, which indicates that successful agents have high acceptance of the consensus solution.

When an agent fails in the competition, the trends of distance differ greatly in these two algorithms, which shows the effectiveness of the penalty function in MACPO. The rapid increase in MAC-FEA indicates that the failed agents have low acceptance of the consensus solution. This is because the solution of private variables in the population does not match the new solution of shared variables well when the shared variables in the failed agents' population are replaced by the consensus solution,

TABLE II
AVERAGE DISTANCE BETWEEN TWO CANDIDATE SOLUTIONS

algo. / func.	MACPO	MAC-FEA
F1	9.8225	22.4365
F2	6.6016	13.0673
F3	145.2344	341.5268
F4	6.2481	7.0815
F5	71.117	99.9663
F6	39.8489	116.8364

For MAC-FEA, when the original objective is used as the optimization objective, the population will move away from the consensus solution quickly. In this case, the candidate solution from the successful agent and the failed agent will differ greatly. The large conflict of candidate solutions is not conducive to the negotiation and cooperation of neighbors. That is why MAC-FEA has a poor performance in the experiment.

For MACPO, the penalty-based objective can constrain the population within a small range near the consensus solution. During the evolution of local optimization, the private and shared variables will match better and better. In this case, the conflict between the candidate solutions from the successful and failed agents is small in the next negotiation phase, which helps the negotiation and cooperation of neighbors. Experiment results in Table I also show that it is effective in improving the optimization performance.

We compute the average distance between candidate solutions at each negotiation phase in Table II, which can reflect the conflict between neighbor agents in the optimization process. This distance is computed as

$$D(X) = \sqrt{\sum_{d \in \mathcal{I}_{i,j}} \left(\boldsymbol{x}_{i,\mathrm{best}}^d - \boldsymbol{x}_{j,\mathrm{best}}^d\right)} \tag{10}$$

where $\boldsymbol{x}_{i,\text{best}}$ and $\boldsymbol{x}_{j,\text{best}}$ are the candidate solution from agent i and j, respectively. According to the table, the average distance in MAC-FEA is much larger than that in MACPO. Thus, the cooperation of agents in MAC-FEA is more difficult than MACPO due to the conflict when negotiating.

2) Analysis of MAC-CC: In the previous analysis, we emphasized the importance of constraining the population around the consensus solution. In this part, the influence of constraint strength on the optimization performance will be discussed. The constraint of MAC-CC on the population is stricter than MACPO. In agents that lose in the previous competition, the values of shared variables are fixed as the consensus solution, and only the private variables are evolved.

According to the experiment result, MAC-CC performs better than MAC-FEA in F1-F6, which shows the role of constraint on populations. However, on the other hand, MAC-CC performs worse than MACPO in F1-F6, which indicates that the constraint strength also has an impact on the algorithm performance. When the shared variables can only be optimized by one of the agents, the evolution result is unfair and biased toward that agent. Thus, this approach is not conducive to cooperative co-evolution among agents.

In conclusion, the proposed penalty-based objective function in MACPO has the effect of constraining the population around the consensus solution when optimizing NDO. Experiment results show that appropriate constraints are beneficial for multiple subpopulations to coevolve to optimize the global objective. In contrast, a lack of constraints can lead to conflict between populations, and excessive constraints can lead to results favoring one side over the other.

D. Comparison With Holistic Algorithms

Holistic algorithms are also popular for large-scale black-box optimization. These algorithms optimize the problem as a whole and have a great optimization effect. However, in a distributed optimization problem, such as NDO, the computational cost is large, and the scalability is limited. In this section, we compare MACPO with a holistic algorithm to show that MACPO can still be competitive in terms of solution quality at a much lower-computational cost than holistic algorithms.

In the following experiments, we compare MACPO with LLSO in the master-slave model, named MS-LLSO. Also, MACPO is embedded by LLSO, making the comparison fair. The reason we embed LLSO into a master-slave model is that holistic algorithms like LLSO cannot handle NDO directly. In this model, the holistic algorithm runs in the master node. When conducting the global fitness evaluation, the master node will send the solution to each slave node and wait for slave nodes to return the local fitness. The sum of all the local fitness from slave nodes is calculated as the global fitness of the solution. MS-LLSO uses $(n+1)$ computing nodes for a NDO with n agents.

The conflict degree of NDO affects the performance of MACPO because it solves NDO in a distributed way. Therefore, we design an indicator to quantify the degree of conflict in the problem, which can help to interpret the performance of our algorithm. For agent i and its neighbor j whose shared variables are $\{d1, d2, \ldots, dm\} \in \mathcal{I}_{i,j}$, the conflict indicator, named CI, is defined by

$$CI = -\left\langle \frac{\widetilde{\nabla} f_i}{\|\widetilde{\nabla} f_i\|}, \frac{\widetilde{\nabla} f_j}{\|\widetilde{\nabla} f_j\|} \right\rangle \tag{11}$$

where

$$\begin{aligned} \widetilde{\nabla} f_i &= \left(\triangle_{\delta,d1}[f_i](\boldsymbol{x}), \triangle_{\delta,d2}[f_i](\boldsymbol{x}), \ldots, \triangle_{\delta,dm}[f_i](\boldsymbol{x})\right) \\ \widetilde{\nabla} f_j &= \left(\triangle_{\delta,d1}[f_j](\boldsymbol{x}), \triangle_{\delta,d2}[f_j](\boldsymbol{x}), \ldots, \triangle_{\delta,dm}[f_j](\boldsymbol{x})\right). \end{aligned} \tag{12}$$

This indicator is the negative inner product of the estimated gradients of f_i and f_j after normalization. It can reflect the included angle of the two gradient directions. The range of conflict index is from -1 to 1. The larger CI, the greater the conflict between two agents. The gradients of f_i and f_j are in the opposite direction when $CI = 1$, and in the same direction when $CI = -1$.

In the following, MACPO and MS-LLSO will be compared from two aspects: 1) solution quality and 2) computational cost.

1) Solution Quality: The experiment results on F1-F18 are shown in Table III, which records the fitness of these two algorithms with $3000 \times D$ times of evaluations and $30000 \times D$ times of evaluations. The experiment result can be summarized as follows.

1) MACPO has better-exploration ability than MS-LLSO in the early stage. When the number of fitness evaluations is $3000 \times D$, MACPO performs better than MS-LLSO on 16 of 18 functions, except F3 and F6. This indicates that MACPO is efficient in exploration.
2) In some functions, the exploitation ability of MACPO is worse than MS-LLSO. When the number of fitness evaluations is $30\,000 \times D$, MACPO performs worse than MS-LLSO on F3, F5, F6, F9, F11, F12, F15, F17, and F18. These functions have a common feature that their elementary functions contain Rosenbrock. Among them, F3, F9, and F15 are homogeneous functions of Rosenbrock, and the others are heterogeneous functions composed of Rosenbrock and other elementary functions.

The conflict between agents is common in distributed optimization problems, including NDO due to the shared variables. This conflicting feature poses a great challenge to distributed optimization algorithms. Holistic algorithms, like LLSO, and divide-and-conquer algorithms, like DCC and FEA, do not suffer from this difficulty because they always conduct global fitness evaluations in the process of optimization. This is why MS-LLSO performs better than MACPO in the above functions.

We compute the conflicting index in F1-F6 in the optimization process of MACPO. In each negotiation phase, the conflicting index of each pair of adjacent agents is computed, and the average value is recorded in Table III. We can summarize the experiment result as follows.

1) There are two classes of functions that have a high-conflict index. The first class is a homogeneous function

TABLE III
COMPARISON WITH HOLISTIC ALGORITHM [1]

E_{max}		$3000 \times D$		$30000 \times D$	
Func.	CI	MACPO	MS-LLSO	MACPO	MS-LLSO
F1	0.228	3.66E+08	5.22E+09	1.75E+08	4.55E+08
F2	0.049	1.47E+06	5.59E+07	7.65E+05	1.81E+07
F3	0.846	2.29E+10	2.25E+10	2.13E+10	1.84E+10
F4	0.051	9.77E+07	1.77E+09	2.70E+07	2.21E+08
F5	0.497	8.19E+08	3.54E+09	6.58E+08	3.73E+08
F6	0.268	2.61E+08	2.51E+08(=)	2.39E+08	1.05E+08
F7	0.173	2.23E+09	3.68E+10	1.06E+09	3.89E+09
F8	-0.110	1.24E+07	5.66E+08	6.74E+06	4.52E+07
F9	0.634	3.25E+11	4.15E+11	2.74E+11	1.70E+11
F10	0.090	3.23E+01	1.99E+10	3.64E+08	2.12E+09
F11	0.381	5.47E+10	9.95E+10	4.26E+10	3.27E+10
F12	0.169	6.23E+10	1.23E+11	4.97E+10	3.06E+10
F13	0.280	6.75E+09	1.59E+11	2.54E+09	2.17E+10
F14	-0.180	6.19E+07	7.92E+10	2.29E+07	1.99E+08
F15	0.590	1.24E+12	3.48E+12	8.47E+11	5.85E+11
F16	0.105	2.92E+09	1.45E+11	5.84E+08	1.30E+10
F17	0.471	2.15E+11	6.71E+11	1.21E+11	1.10E+11
F18	0.192	2.15E+11	2.50E+12	8.15E+10	6.57E+10
w/t/l		16/1/1		9/0/9	

[1] The experiment result of MACPO and MS-LLSO on functions F1-F18. This table records the results when the number of fitness evaluations for each agent is $3000 \times D$ and when that is $30000 \times D$, where D is the dimension of the agent.

of Rosenbrock, containing F3, F9, and F15. The second class is the heterogeneous functions composed of Rosenbrock and Elliptic, containing F5, F11, and F17. When optimizing these functions, MACPO will waver around the optimal solution and cannot go any further. Thus, its solution quality is worse than the holistic algorithm MS-LLSO. This is a common challenge of distributed optimization problems, especially for nonconvex and black-box problems.

2) There are two classes of functions that have a low-conflict index. The first class is a homogeneous function of Schwefel, containing F2, F8, and F14. The second class is the heterogeneous functions composed of Schwefel and Elliptic, containing F4, F10, and F16. Although the agents of these problems also have conflicts, they are not serious, which makes the cooperative strategy of the proposed algorithm can effectively optimize the problem. On these functions, MACPO performs better than MS-LLSO.

In conclusion, the higher the conflict index of the problem, the greater the challenge to the distributed optimization algorithm MACPO. The holistic algorithm MS-LLSO is not affected by the level of conflict because it treats NDO as a centralized problem.

2) Computational Cost Comparison: In this part, we will prove that the multiagent co-evolutionary model in MACPO has a lower-computational cost and better scalability than MS-LLSO. The computational cost consists of two parts: 1) the evaluation cost T_e of the problem and 2) the communication cost T_c of the distributed model.

We will compare the computational cost of MACPO and MS-LLSO with the same number of evaluations. Suppose the maximum number of fitness evaluations for each agent is the same, which is denoted as $E_{\max}$. In addition, we assume that the time cost at each time of communication is the same, and the time cost of each fitness evaluation of subproblems is the same. They are denoted as t_c and t_e, respectively.

For MACPO, the computational cost of the ith agent is

$$\begin{aligned} T^i_{\text{MACPO}} &= T^i_c + T^i_e \\ &= 3I\|\mathcal{N}_i\|t_c + It_e\left(k\|\mathcal{P}_i\| + \sum_{j\in\mathcal{N}_i}\left(\|\mathcal{I}_{i,j}\|+1\right)\right) \\ &= \frac{3\|\mathcal{N}_i\|E_{\max}}{k\|\mathcal{P}_i\| + \sum_{j\in\mathcal{N}_i}\left(\|\mathcal{I}_{i,j}\|+1\right)}t_c + E_{\max}t_e \\ &\text{where}\quad I = \frac{E_{\max}}{k\|\mathcal{P}_i\| + \sum_{j\in\mathcal{N}_i}\left(\|\mathcal{I}_{i,j}\|+1\right)} \end{aligned} \tag{13}$$

where n is the number of agents in the problem, I is the number of iterations, and k is the number of generations in each iteration. In each iteration, each agent will communicate with each neighbor three times, so the communication cost is $3\|\mathcal{N}_i\|t_c$. In each iteration, the fitness evaluations are used for population evolution and negotiation with neighbors, which is $k\|\mathcal{P}_i\|$ and $\sum_{j\in\mathcal{N}_i}(\|\mathcal{I}_{i,j}\|+1)$, respectively.

For MS-LLSO, each local fitness evaluation will cost two times of communication, with the master node sending the solution to slave nodes and receiving fitness from slave nodes. Thus, the computational cost of the master agent is

$$T_{\text{master}} = E_{\max}(2nt_c + t_e). \tag{14}$$

For each time, the evaluation cost is counted as t_e because the slave nodes can conduct local evaluations in parallel.

The computational cost of MACPO is determined by the agent with the largest computational cost. The computational cost of MS-LLSO is determined by the master node. Therefore, the computational cost ratio of these two algorithms is

$$\begin{aligned} \frac{T_{\text{MS-LLSO}}}{T_{\text{MACPO}}} &= \frac{T_{\text{master}}}{\max\left(T^i_{\text{MACPO}}\right)} \\ &= \frac{2nt_c + t_e}{qt_c + t_e} \\ &= \frac{2n(t_c/t_e)+1}{q(t_c/t_e)+1} \end{aligned} \tag{15}$$

where

$$q = \max_{i\in\{1,\ldots,n\}} \frac{3\|\mathcal{N}_i\|}{k\|\mathcal{P}_i\| + \sum_{j\in\mathcal{N}_i}\left(\|\mathcal{I}_{i,j}\|+1\right)}. \tag{16}$$

It can be seen that the time ratio of algorithms is related to q, n and (t_c/t_e). Therefore, we can draw the following conclusions.

1) The parameter q is decided by the benchmark functions and the parameter setting of MACPO. In most cases, q is less than 1. For example, in F1-F6, each agent has one or two neighbors, and the shared size $\|\mathcal{I}_{i,j}\|$ is 5 for each pair of neighbors. The population size $\|\mathcal{P}_i\|$ is 300 and the number of generations k is $0.4 \times D$ in MACPO. Thus, $q = 1/502$ in F1-F6. Similarly, q is $1/1337$ in F7-F12 and $1/2005$ in F13-F18.

2) The value of t_c/t_e varies in different application scenarios, depending on the quality of the communication network, the topology of agents, and the evaluation method of problems. According to Fig. S6 in the supplementary material, when the communication cost is 0, that is $t_c/t_e = 0$, the cost of the two algorithms is equal. With the increase of t_c/t_e, the cost ratio of these two algorithms will gradually increase and approach $2n/q$. The maximum ratio $2n/q$ is 20 080 for F1-F6, 106 960 for F7-F12, and 240 600 for F13-F18.
3) According to (13), the computation cost of MACPO is influenced by the number of shared variables, the swarm size, and the number of generations per iteration.

In addition to theoretical analysis, we tested the time cost of MACPO and MS-LLSO by experiments, shown in Fig. S5 of the supplementary material. According to the experiment results, as the network scale increases, the time cost of MACPO does not change much, while that of MS-LLSO increases rapidly.

In conclusion, when optimizing the same problem, MACPO has a lower-computational cost than MS-LLSO. The lower the cost ratio of communication and fitness evaluation in the scenario, the more significant the gap between them. The computational cost ratio eventually approaches a value, which is determined by the characteristics of problems.

E. Comparison With Existing Distributed Algorithms

We first test a resource allocation problem on networks, which is introduced in Part V of the supplementary material in detail. The proposed MACPO is compared with consensus-based alternating direction method of multipliers (ADMM) [41], EXTRA (an exact first-order algorithm for decentralized consensus optimization) [13], DGD (distributed subgradient methods for multiagent optimization) [9]. The experiment result is shown in Fig. S9 of the supplementary material. According to the figure, all the algorithms converge below 10^{-5}. ADMM converges earliest, while DGD converges last. Our algorithm MACPO has a relatively larger variance than compared algorithms. The best case of MACPO reaches 10^{-26}, which is better than comparison algorithms. And the worst case reaches 10^{-5}. In conclusion, the effect of our algorithm is close to that of the comparison algorithms, and the comparison algorithms are more stable.

In addition to the above problem, our algorithm can also handle complex distributed problems whose gradient is hard to compute. Therefore, we test our algorithm on an electric dispatching problem on a network of power stations, which is introduced in Part VI of the supplementary material. We compare our algorithm with two distributed black-box algorithms, DPSO [29] and general framework for population-based distributed optimization (GFPDO) [30]. The implementation of comparison algorithms is mentioned in Part VI of the supplementary material. Fig. S10(a) in the supplementary material shows the fitness curve of them. According to the result, our algorithm performs better than comparison algorithms. In addition, we set the number of nodes from 10 to 60, and test the time cost of three algorithms. Experiment results are shown in Fig. S10(b) of the supplementary material. According to the curve, the time cost and trend of MACPO and comparison algorithms are similar. We also test the performance of distributed black-box algorithms on benchmark functions. According to the results in Table I, the proposed MACPO performs better than DPSO and GFPDO in benchmark functions F1-F6, no matter when embedded by LLSO or CSO.

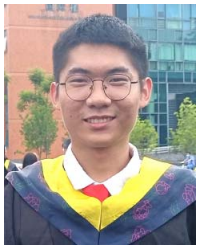
V. Conclusion

In this work, we propose a multiagent co-evolutionary algorithm to solve NDO problems. According to experiment results and analysis, the performance of the proposed algorithm is not as good as the holistic algorithm on benchmark problems whose conflict index *CI* is relatively high. Therefore, there is still a lot of potential to study the algorithm in high-conflict problems or scenarios. What is more, various advanced optimization algorithms have been proposed and shown outstanding performance on complex optimization problems, such as online learning, scheduling, multiobjective optimization, transportation, medicine, and data classification [42], [43], [44]. Therefore, it is worth further studying the effect of more advanced optimization algorithms for NDO, such as new types of hybrid heuristics, metaheuristics, adaptive algorithms, self-adaptive algorithms, island algorithms, polyploid algorithms, etc. [45], [46].

References

[1] S. Vorotnikov, K. Ermishin, A. Nazarova, and A. Yuschenko, "Multi-agent robotic systems in collaborative robotics," in *Interactive Collaborative Robotics*, A. Ronzhin, G. Rigoll, and R. Meshcheryakov, Eds. Cham, Switzerland: Springer, 2018, pp. 270–279.
[2] D. Molzahn, F. Dörfler, H. Sandberg, S. Low, S. Chakrabarti, R. Baldick, and J. Lavaei, "A survey of distributed optimization and control algorithms for electric power systems," *IEEE Trans. Smart Grid*, vol. 8, no. 6, pp. 2941–2962, Nov. 2017.
[3] P. Wan and M. D. Lemmon, "Event-triggered distributed optimization in sensor networks," in *Proc. Int. Conf. Inf. Process. Sens. Netw.*, Apr. 2009, pp. 49–60.
[4] R. Bähnemann, D. Schindler, M. Kamel, R. Siegwart, and J. Nieto, "A decentralized multi-agent unmanned aerial system to search, pick up, and relocate objects," in *Proc. IEEE Int. Symp. Safety, Security Rescue Robot. (SSRR)*, 2017, pp. 123–128.
[5] N. Patari, V. Venkataramanan, A. Srivastava, D. K. Molzahn, N. Li, and A. Annaswamy, "Distributed optimization in distribution systems: Use cases, limitations, and research needs," *IEEE Trans. Power Syst.*, vol. 37, no. 5, pp. 3469–3481, Sep. 2022.
[6] J. Martinez-Piazuelo, G. Diaz-Garcia, N. Quijano, and L. F. Giraldo, "Discrete-time distributed population dynamics for optimization and control," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 11, pp. 7112–7122, Nov. 2022.
[7] D. Ye, M. Zhang, and A. V. Vasilakos, "A survey of self-organization mechanisms in multiagent systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 3, pp. 441–461, Mar. 2017.
[8] T. Yang et al., "A survey of distributed optimization," *Annu. Rev. Control*, vol. 47, pp. 278–305, Jun. 2019.
[9] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.
[10] Q. Ma, Q. Meng, and S. Xu, "Distributed optimization for uncertain high-order nonlinear multiagent systems via dynamic gain approach," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 53, no. 7, pp. 4351–4357, Jul. 2023.
[11] H. Wang, X. Liao, T. Huang, and C. Li, "Cooperative distributed optimization in multiagent networks with delays," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 2, pp. 363–369, Feb. 2015.

[12] S. Yang, Q. Liu, and J. Wang, "Distributed optimization based on a multiagent system in the presence of communication delays," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 5, pp. 717–728, May 2017.
[13] W. Shi, Q. Ling, G. Wu, and W. Yin, "EXTRA: An exact first-order algorithm for decentralized consensus optimization," Nov. 2014, *arXiv:1404.6264*.
[14] J. Wang and N. Elia, "Control approach to distributed optimization," in *Proc. 48th Annu. Allerton Conf. Commun. Control, Comput.*, Dec. 2010, pp. 557–561.
[15] D. Yuan and D. W. C. Ho, "Randomized gradient-free method for multiagent optimization over time-varying networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 6, pp. 1342–1347, Jun. 2015.
[16] Y. Pang and G. Hu, "Randomized gradient-free distributed optimization methods for a multiagent system with unknown cost function," *IEEE Trans. Autom. Control*, vol. 65, no. 1, pp. 333–340, Jan. 2020.
[17] R. Patel, E. Rudnick-Cohen, S. Azarm, M. Otte, H. Xu, and J. W. Herrmann, "Decentralized task allocation in multi-agent systems using a decentralized genetic algorithm," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 3770–3776.
[18] H. Choi, Y. Kim, and H. J. Kim, "Genetic algorithm based decentralized task assignment for multiple unmanned aerial vehicles in dynamic environments," *Int. J. Aeronaut. Space Sci.*, vol. 12, pp. 163–174, Jun. 2011.
[19] I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Inf. Sci.*, vol. 237, pp. 82–117, Jul. 2013.
[20] A. Song, W.-N. Chen, T. Gu, H. Yuan, S. Kwong, and J. Zhang, "Distributed virtual network embedding system with historical archives and set-based particle swarm optimization," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 51, no. 2, pp. 927–942, Feb. 2021.
[21] K. Hussain, M. N. M. Salleh, S. Cheng, and Y. Shi, "Metaheuristic research: A comprehensive survey," *Artif. Intell. Rev.*, vol. 52, pp. 2191–2233, Dec. 2019.
[22] Y.-J. Gong et al., "Distributed evolutionary algorithms and their models: A survey of the state-of-the-art," *Appl. Soft Comput.*, vol. 34, pp. 286–300, Sep. 2015.
[23] Z.-H. Zhan, Z.-J. Wang, H. Jin, and J. Zhang, "Adaptive distributed differential evolution," *IEEE Trans. Cybern.*, vol. 50, no. 11, pp. 4633–4647, Nov. 2020.
[24] Q. Yang et al., "A distributed swarm optimizer with adaptive communication for large-scale optimization," *IEEE Trans. Cybern.*, vol. 50, no. 7, pp. 3393–3408, Jul. 2020.
[25] Y.-H. Jia et al., "Distributed cooperative co-evolution with adaptive computing resource allocation for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 2, pp. 188–202, Apr. 2019.
[26] M. N. Omidvar, X. Li, and X. Yao, "Cooperative co-evolution with delta grouping for large scale non-separable function optimization," in *Proc. IEEE Congr. Evol. Comput.*, Barcelona, Spain, Jul. 2010, pp. 1–8.
[27] S. Strasser, J. Sheppard, N. Fortier, and R. Goodman, "Factored evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 21, no. 2, pp. 281–293, Apr. 2017.
[28] Y.-H. Jia, Y. Mei, and M. Zhang, "Contribution-based cooperative co-evolution for nonseparable large-scale problems with overlapping subcomponents," *IEEE Trans. Cybern.*, vol. 52, no. 6, pp. 4246–4259, Jun. 2022.
[29] Y. Wakasa and S. Nakaya, "Distributed particle swarm optimization using an average consensus algorithm," in *Proc. 54th IEEE Conf. Decis. Control (CDC)*, 2015, pp. 2661–2666.
[30] W. Ai, W. Chen, and J. Xie, "A general framework for population-based distributed optimization over networks," *Inf. Sci.*, vol. 418–419, pp. 136–152, Dec. 2017.
[31] K. Utkarsh, A. Trivedi, D. Srinivasan, and T. Reindl, "A consensus-based distributed computational intelligence technique for real-time optimal control in smart distribution grids," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 1, no. 1, pp. 51–60, Feb. 2017.
[32] M. K. Jalloul and M. A. Al-Alaoui, "A distributed particle swarm optimization algorithm for block motion estimation using the strategies of diffusion adaptation," in *Proc. Int. Symp. Signals, Circuits Syst. (ISSCS)*, Jul. 2015, pp. 1–4.
[33] L. Tong, X. Zhou, and H. J. Miller, "Transportation network design for maximizing space–time accessibility," *Transp. Res. B, Methodol.*, vol. 81, pp. 555–576, Nov. 2015.
[34] V. Kekatos and G. B. Giannakis, "Distributed robust power system state estimation," *IEEE Trans. Power Syst.*, vol. 28, no. 2, pp. 1617–1626, May 2013.
[35] J. Chen, L. Tang, J. Liu, and J. Ye, "A convex formulation for learning shared structures from multiple tasks," in *Proc. 26th Annu.Int. Conf. Mach. Learn.*, Jun. 2009, pp. 137–144.
[36] O. Chapelle, P. Shivaswamy, S. Vadrevu, K. Weinberger, Y. Zhang, and B. Tseng, "Multi-task learning for boosting with application to Web search ranking," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, Jul. 2010, pp. 1189–1198.
[37] J. Zhou, L. Yuan, J. Liu, and J. Ye, "A multi-task learning formulation for predicting disease progression," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, Aug. 2011, pp. 814–822.
[38] J. Plata-Chaves, A. Bertrand, and M. Moonen, "Incremental multiple error filtered-X LMS for node-specific active noise control over wireless acoustic sensor networks," in *Proc. IEEE Sens. Array Multichannel Signal Process. Workshop (SAM)*, Jul. 2016, pp. 1–5.
[39] Q. Yang, W.-N. Chen, J. D. Deng, Y. Li, T. Gu, and J. Zhang, "A level-based learning swarm optimizer for large-scale optimization," *IEEE Trans. Evol. Comput.*, vol. 22, no. 4, pp. 578–594, Aug. 2018.
[40] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 191–204, Feb. 2015.
[41] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2010.
[42] M. Heider, D. Pätzel, H. Stegherr, and J. Hähner, "A metaheuristic perspective on learning classifier systems," in *Metaheuristics for Machine Learning: New Advances and Tools*, M. Eddaly, B. Jarboui, and P. Siarry, Eds. Singapore: Springer Nat., 2023, pp. 73–98.
[43] J.-A. Mejía-de Dios, A. Rodríguez-Molina, and E. Mezura-Montes, "Multiobjective bilevel optimization: A survey of the state-of-the-art," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 53, no. 9, pp. 5478–5490, Sep. 2023.
[44] R. Elshaer and H. Awad, "A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants," *Comput. Ind. Eng.*, vol. 140, Feb. 2020, Art. no. 106242.
[45] T. Dökeroglu, E. Sevinç, T. Kucukyilmaz, and A. Cosar, "A survey on new generation metaheuristic algorithms," *Comput. Ind. Eng.*, vol. 137, Nov. 2019.
[46] S. Meyer-Nieberg and H.-G. Beyer, "Self-adaptation in evolutionary algorithms," in *Parameter Setting in Evolutionary Algorithms*, F. G. Lobo, C. F. Lima, and Z. Michalewicz, Eds. Heidelberg, Germany: Springer, 2007, vol. 54, pp. 47–75.



Tai-You Chen received the bachelor's degree in computer science and technology from the South China University of Technology, Guangzhou, China, in 2022, where he is currently pursuing the Ph.D. degree in computer science and technology with the School of Computer Science and Engineering.

His current research interests include swarm intelligence, evolutionary computation, consensus-based distributed optimization, multiagent systems, and their applications in real-world problems.

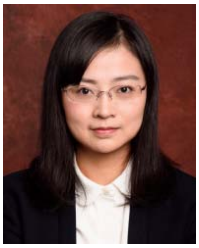
Wei-Neng Chen (Senior Member, IEEE) received the bachelor's and Ph.D. degrees in computer science from Sun Yat-sen University, Guangzhou, China, in 2006 and 2012, respectively.

Since 2016, he has been a Full Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. He has coauthored over 100 international journal and conference papers, including more than 70 papers published in the IEEE TRANSACTIONS journals. His current research interests include computational intelligence, swarm intelligence, network science, and their applications.

Dr. Chen was a recipient of the IEEE Computational Intelligence Society (CIS) Outstanding Dissertation Award in 2016, and the National Science Fund for Excellent Young Scholars in 2016. He was also a Principle Investigator of the National Science and Technology Innovation 2030—the Next Generation Artificial Intelligence Key Project. He is currently the Vice-Chair of the IEEE Guangzhou Section and the Chair of the IEEE SMC Society Guangzhou Chapter. He is also a Committee Member of the IEEE CIS Emerging Topics Task Force. He serves as an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS and the *Complex and Intelligent Systems*.

Xiao-Qi Guo received the bachelor's and PhD. degrees in computer science and technology from the South China University of Technology, Guangzhou, China, in 2018 and 2023, respectively.

Her current research interests include evolutionary computation, distributed optimization, edge–cloud computing, and their applications on data-driven optimization in real-world problems.



Yue-Jiao Gong (Senior Member, IEEE) received the B.S. and Ph.D. degrees in computer science from Sun Yat-sen University, Guangzhou, China, in 2010 and 2014, respectively.

She is currently a Full Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. Her research interests include evolutionary computation, swarm intelligence, and their applications to intelligent transportation and smart city scheduling. She has published over 100 papers, including more than 40 IEEE TRANSACTIONS papers in her research area.

Dr. Gong was a recipient of the Pearl River Young Scholar from the Guangdong Education Department in 2017 and the Guangdong Natural Science Funds for Distinguished Young Scholars in 2022.

Jun Zhang (Fellow, IEEE) received the Ph.D. degree in electrical engineering from the City University of Hong Kong, Hong Kong, in 2002.

His research contributions span over 500 peer-reviewed publications of which more than 220 appear in IEEE TRANSACTIONS. His research interests include computational intelligence and evolutionary computation.

Prof. Zhang currently serves as an Associate Editor for the IEEE TRANSACTIONS ON ARTIFICIAL INTELLIGENCE and the IEEE TRANSACTIONS ON CYBERNETICS.