Multi-Agent Evolution Strategy with Cooperative and Cumulative Step Adaptation for Black-Box Distributed Optimization

Tai-You Chen, Wei-Neng Chen, Senior Member, IEEE, Jin-Kao Hao, Yang Wang, Jun Zhang, Fellow, IEEE

Abstract—In recent years, black-box distributed optimization (DBO) has been widely studied to solve complex optimization problems in multi-agent systems, such as hyperparameter optimization of distributed machine learning. However, most existing methods use a fixed or diminishing step size to sample and search in the black box optimization space, which makes it challenging to maintain optimization efficiency on different optimization problems. In this work, we propose a multi-agent evolution strategy with cooperative and cumulative step adaptation (CCSA-DES). In CCSA-DES, each agent executes the algorithm to sample and explores its local objective function, and communicates with other agents to optimize the global objective function cooperatively, which is the sum of local objective functions. To improve the sampling adaptability, we design a cooperative and cumulative step adaptation method (CCSA) consisting of inner adaptation and outer adaptation. By detecting the evolution path of the multi-agent system, CCSA decreases the step size when the evolution directions of agents are conflicting and increases the step size when consistent. In terms of theoretical analysis, we first discuss the working principle of CCSA, and then discuss the system consensus of CCSA-DES. In terms of experimental verification, CCSA-DES achieves better consensus performance and competitive solution quality compared with state-of-the-art algorithms for DBO.

Index Terms—multi-agent systems, evolutionary computation, evolution strategies, step adaptation, black-box distributed optimization

Manuscript received August 5, 2024; revised November 14, 2024; accepted December 29, 2024. This work was supported in part by the National Key Research and Development Project No. 2023YFE0206200, National Natural Science Foundation of China under Grants U23B2058, in part by Guangdong Regional Joint Foundation Key Project 2022B1515120076, in part by the research fund of Hanyang University (HY-202300000003465), and in part by the Tianjin Top Scientist Studio Project under Grant 24JRRCRC00030. (Corresponding Author: Wei-Neng Chen, cschenwn@scut.edu.cn)

Tai-You Chen and Wei-Neng Chen are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: 575583114@qq.com; cschenwn@scut.edu.cn).

Jin-Kao Hao is with the Department of Computer Science, LERIA Laboratory, Université d'Angers, 49045 Angers, France (e-mail: jin-kao.hao@univ-angers.fr).

Yang Wang is with the School of Management, Northwestern Polytechnical University, Xi'an 710072, China (e-mail: yangw@nwpu.edu.cn).

Jun Zhang is with the Nankai University, Zhejiang Normal University and also with the Hanyang University, ERICA, South Korea. (email: junzhang@ieee.org)

I. Introduction

Multi-agent systems (MAS) widely exist in various scenarios, including wireless sensor networks [1], smart grids [2], and others [3, 4]. In the MAS, multiple physical or virtual entities with computation and communication capabilities are connected through a communication network. Compared with independent entities, the MAS is promising to achieve better global performance through the cooperation of agents. Therefore, distributed optimization for the MAS has received considerable attention from researchers in recent years, in which the global objective function is the sum of local objective functions of all agents [5].

In distributed optimization, each agent can only access its own local objective function and communicate with immediate neighbors in the network. When the local objectives conflict, it is challenging to make agents cooperate to minimize the global objective in a distributed manner. Facing this challenge, Nedic and Ozdaglar proposed a distributed subgradient descent method [6, 7], which combined the consensus theory [8] and gradient descent method to confirm the global optimality and consensus of agents. After that, distributed optimization methods under different system characteristics have been extensively studied, including time-varying networks [9], unbalanced networks [10], continuous-time systems [11], constrained problems [10], etc [12, 13].

Most above-mentioned methods rely on the first-order or second-order gradient information of objective functions. However, the gradient information of objective functions may be computationally infeasible in many complex scenarios. First, when the solution quality is evaluated by simulations, there is no explicit mathematical expression for the objective. For example, the network influence evaluation relies on Monte Carlo simulation [14], the traffic efficiency evaluation requires actual road network simulation [15], and the vehicle performance evaluation relies on physical testing [16, 17]. Second, the gradient information of hyperparameter optimization problems is also infeasible. Such scenarios exist in the network architecture search of neural networks [18], feature selection of machine learning [19], and load flow calculation of smart grids [20]. Therefore, black-box distributed optimization independent of the gradient information has great research

significance.

To address black-box distributed optimization problems, many gradient-free or zeroth-order methods have been proposed [21, 22, 23, 24, 25, 26]. The main idea of these methods is to estimate the gradient by randomized sampling around the current solution. The step size for sampling plays an important role in these gradient-free methods. Most existing methods use a fixed [23] or diminishing [27, 28, 29] step size, and the initialization step size is chosen by parameter tuning for the test problems. However, the optimization problem in real-world scenarios may vary as the environment, such as the electricity demand in the smart grid [30, 31]. In this case, an identical step size setting cannot maintain high optimization efficiency in different conditions. What's more, the optimization method is called frequently in such scenarios and cannot rely on manual parameter re-tuning.

Considering the above challenge of step adaptability, evolution strategies (ES), a class of evolutionary computation methods, have great potential [32, 33, 34]. This is because step adaptation methods have been widely studied for the mutation operation of ES, including cumulative step size adaptation (CSA) [35, 36], success-based rules [37, 38], self-adaptation [39, 40], and meta-ES [41, 42]. For example, CSA is a representative adaptation method, which adjusts the step size by analyzing the evolution path of ES. Its intuitive aim is to increase the step size in the steep region to accelerate the optimization and decrease the step size in the optimal region to gradually converge. Although the above methods have been shown to be effective for centralized optimization problems, they face challenges on distributed optimization problems because they only adapt the step according to the evolution path of a single agent. In distributed optimization problems, the local objectives usually conflict and thus the optimum region of the global objective is different from the optimum region of local objectives. When the above independent step adaptation methods are adopted for each agent, agents will choose the step size that is beneficial to its own local objective, rather than the global objective. As a result, agents will conflict to each other and cannot optimize the global objective. In conclusion, adjusting the step size independently using existing step adaptation methods without any cooperation is not beneficial for distributed optimization.

Consequently, it is worth further studying how multiple agents can cooperate to adapt the step size during distributed optimization. The cooperative step adaptation method for distributed optimization is seldom studied in existing distributed or parallel evolution strategies [21, 43, 44]. He et al. [21] pre-set a diminishing step size for the distributed evolution strategy without step size adaptation. Duan et al. [44] verified the parallel evolution strategy under the centralized problems, and did not consider the cooperation challenge in the case when the local objectives are conflicting. Considering the above research gap, we propose a multi-agent evolution strategy with cooperative and cumulative step size adaptation (CCSA-DES) to address black-box distributed optimization. CCSA-DES works in a distributed manner, where each agent executes an identical algorithm and cooperates with neighbors through peer-to-peer communication. CCSA-DES alternates two phases, local optimization and neighboring cooperation. In the first phase, agents conduct the evolution strategy for a certain number of generations independently to optimize their local objective functions. In the second phase, agents communicate with neighbors and update the local solution and step size according to neighbors' messages. The major contributions of this work are summarized as follows:

1) To improve the sampling adaptability of the algorithm, we proposed a cooperative and cumulative step size adaptation method (CCSA). CCSA consists of inner adaptation and outer adaptation, with the former working in the local optimization phase and the latter in the neighboring cooperation phase. The idea of CCSA is to identify whether the evolution direction of an agent is consistent or conflicting with that of its neighbors. Based on the identification result, CCSA will increase the step size when the evolution direction of an agent is consistent with that of its neighbors, and decrease the step size when the evolution direction is conflicting. In this work, we analyze the working principle of CCSA theoretically and verify the effectiveness of CCSA experimentally.
2) Based on CCSA, a multi-agent evolution strategy CCSA-DES is proposed. First, in terms of the consensus ability, we provide a theoretical consensus discussion for CCSA-DES based on the mechanism of CCSA. The consensus ability of CCSA-DES is also shown to be better than existing algorithms through experiments. Second, in terms of the solution quality, CCSA-DES achieves competitive solution quality on benchmark functions compared with state-of-the-art algorithms for black-box distributed optimization.

The rest of this paper is organized as follows. Section II provides the preliminaries of step adaptation methods. Section III conducts a case study of the evolution path in distributed optimization, which illustrates the limitations of existing methods. The proposed algorithm CCSA-DES is elaborated in Section IV. A series of experiments are carried out in Section V. Finally, Section VI concludes this work.

II. Preliminaries

In this section, we first introduce the problem definition and application scenarios. Then, we present the typical cumulative step adaptation method, which provide a basis for the proposed algorithm.

A. Problem Definition of Distributed Optimization

In a multi-agent system with n agents connected through a communication network, a general definition

of a distributed optimization problem is as follows:

$$\min F(x) = \sum_{i=1}^{n} f_i(x) \tag{1}$$

Here, $x \in \mathbb{R}^N$ is the N-dimensional decision variable. $f_i(*)$ denotes the local objective function of agent i, and $F(*)$ denotes the global objective function of the system. When the mathematical expression of local objective functions f_i is unknown or the gradient of f_i is uncomputable, the problem is termed black-box distributed optimization.

There are two major challenges to minimize the global objective function in distributed optimization problems. First, each agent i can only access its own local objective function f_i due to local data, limited sensing range, or privacy concerns. Thus, a single agent cannot address the global problem independently. Second, local objective functions of different agents are usually conflicting and they have distinct optimal solutions. Thus, there is no solution that can simultaneously minimize all local objectives. In order to minimize the global objective function F, it is necessary for agents to cooperate and strike a balance among multiple local objective functions. In a word, the distributed optimization problem is non-separable, where agents should cooperate instead of optimize independently.

Distributed optimization problems widely exist in realworld multi-agent systems, such as cooperative target localization in wireless sensor networks, the predictive control and economic dispatch in smart grids, the path scheduling and task dispatch for multi-UAV systems [1, 2]. Take the cooperative localization problem as an example, n sensors are connected as a multi-agent system and take the positions of targets as the optimization variable x. The local objective function $f_i(*)$ for agent i representing its own estimation error. The global objective function $F(*)$ aims to minimize the sum of estimation errors across all sensors. Similarly, considering the power allocation problem within a distributed power system containing n power stations, the optimization variable x is the power allocation strategy. In this scenario, the local objective function $f_i(*)$ is the energy loss of an energy station, while the global objective $F(*)$ is to minimize the energy loss across the entire system.

B. Cumulative step adaptation method

Cumulative step adaptation (CSA) is a representative step size control method for evolution strategies. For an optimization problem with N dimensions, CSA constructs a cumulative evolution path $p \in \mathbb{R}^N$. The initial path p^0 is set to a zero vector. With the optimization process, the evolution path is updated as follows:

$$p^{t+1} = (1 - c_s)p^t + \sqrt{c_s(2 - c_s)\mu_{\text{eff}}}\Delta x^t/\sigma^t \tag{2}$$

where t is the iteration number, σ^t is the step size, and Δx^t is the movement vector of the current generation. When the problem dimension $N < 1000$, the parameter c_s is usually set in the range $[0, 0.5]$, and μ_{eff} is set in the range $[1, 10]$.

The length of p reflects the consistency of the evolution path. When the length of p is short, it indicates that the past several steps are anti-correlated and they cancel each other out. In this case, the step size is expected to be decreased. When the length of p is long, the past several steps are considered to be correlated because they cumulate the evolution path in a similar direction. The step size is expected to be increased in this case. Based on this idea, the step size σ is adapted as follows:

$$\sigma^{t+1} = \sigma^t \exp[r(\frac{\|p^t\|}{\mathbb{E}(\|p\|)} - 1)] \tag{3}$$

where t is the iteration number. r is the adaptation rate, which is usually set in the range $[0, 0.5]$. Here, the current length of the evolution path p^t is compared with its expected length, which assumes that the path is randomly generated and uncorrelated. The step size will increase when $\|p^t\|$ is larger than its expectation, and decrease otherwise.

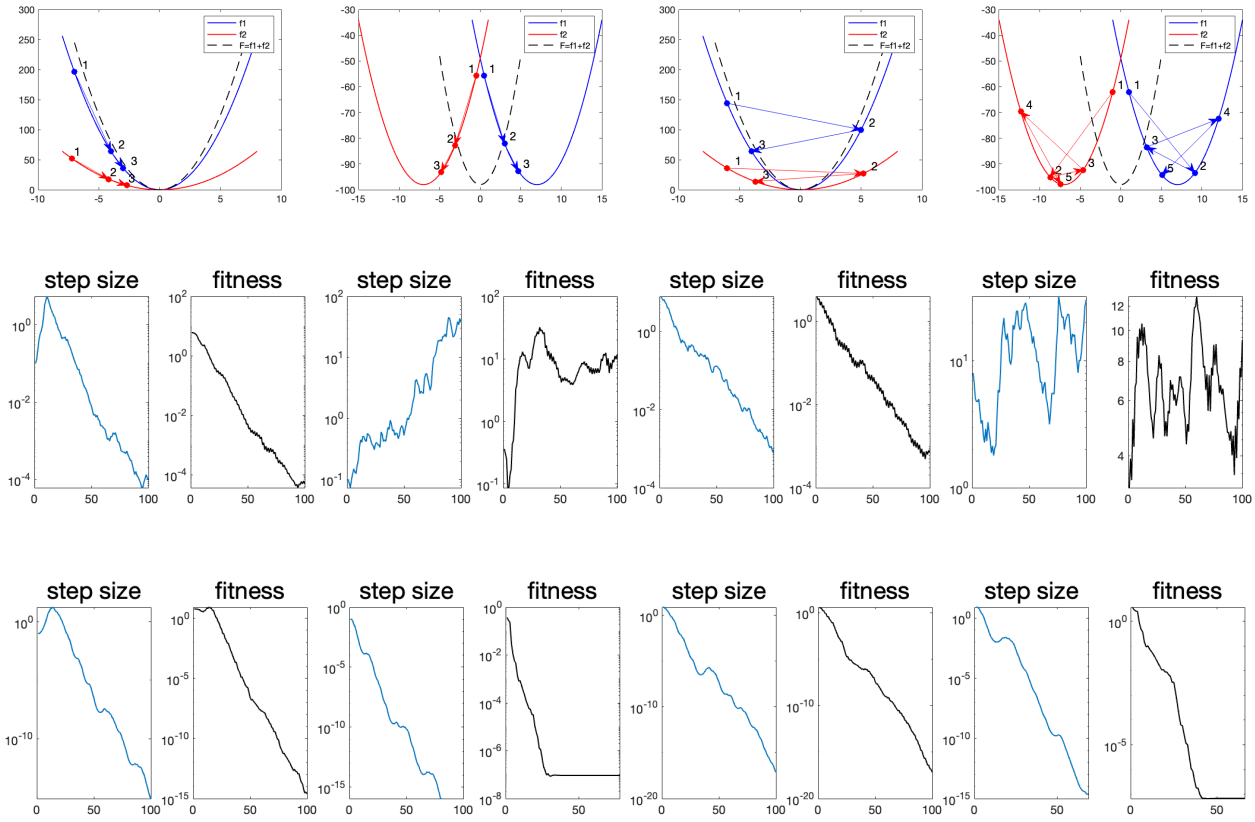
III. Case Study for Evolution Path in Distributed Optimization

The evolution path, the cumulation of multiple successive steps, is the key to adjusting step size by CSA and other variant methods. However, the evolution path analysis of distributed optimization is more complicated than centralized optimization because each agent owns an evolution path. Take a two-agent system as an example, we illustrate four typical conditions of the evolution path for distributed optimization in Fig. 1:

1) As shown in Fig. 1(a), the current evolving direction of the agent is consistent with its own historical evolution path, and also consistent with the evolution path of its neighboring agent.
2) As shown in Fig. 1(b), the current evolving direction of the agent is consistent with its own historical evolution path, but opposite to the evolution path of its neighboring agent.
3) As shown in Fig. 1(c), the current evolving direction of the agent is opposite to its own historical evolution path, but consistent with the evolution path of its neighboring agent.
4) As shown in Fig. 1(d), the current evolving direction of the agent is opposite to its own historical evolution path, and also opposite to the evolution path of its neighboring agent.

It is worth noting that, the possible evolution paths include but not limited to the above four conditions. The purpose of this section is to discuss the four typical cases in order to explain the difficulties and draw out our proposed algorithm.

The intuitive idea of the existing step size adaptation methods, like CSA, is to increase the step size when the directions of the evolution paths are consistent, and to



(a) C

(e) CSA

(i) CCSA (c

Fig. 1. Case study for evolution paths in distributed optimization. (1) Fig. (a)-(d) illustrate four toy examples of one-dimension distributed optimization problem in a two-agent system. The objective functions of the two agents are colored blue and red respectively, which is denoted by f_1 and f_2. The global objective function $F = f_1 + f_2$ is denoted by the black dotted-line. The evolution paths are marked numerically in the figures. (2) Fig. (e)-(h) record the step size and fitness of the agent when using CSA [35] for step adaptation. (3) Fig. (i)-(l) record the step size and fitness of the agent when using CCSA (proposed) for step adaptation. Note that, the horizontal coordinate of Fig. (e)-(l) is the iteration number.

decrease the step size when they are opposite. Fig. 1(e)-(h) shows the performance of CSA on the four cases, including the step size and fitness. The fitness is defined by the difference from the evaluated solution to the global optimal solution. Denote the solution as $\overline{x}$ and the global optimal solution as x^*, the fitness is computed by $f = \|\overline{x} - x^*\|$. Thus, the optimal fitness of these four cases is zero. In the following, we will analyze the behaviors of CSA:

1) For the first condition shown in Fig. 1(e), CSA increases the step size in the early stage because the evolving direction of agent 1 is consistent with its own evolution path, and decreases the step size in the late stage because it is close to the optimal region. Finally, the fitness approaches zero, which indicates that CSA works well in this condition.
2) For the second condition shown in Fig. 1(f), CSA increases the step size in general because the evolving direction of agent 1 is consistent with its own evolution path. However, the increase in the step size is not conducive to the convergence of the algorithm to the global optimum in this condition. As a result, the fitness fluctuates and remains at a high value. A more detailed explanation is provided in part I of the supplementary material.
3) For the third condition shown in Fig. 1(g), CSA decreases the step size because the evolving direction of agent 1 is opposite to its own historical evolution path. The CSA method also works in this condition because the fitness value approaches zero.
4) For the fourth condition shown in Fig. 1(h), CSA decreases the step size in the early stage because the evolving direction of agent 1 is opposite to its own historical evolution path. However, the step size fluctuates in the late stage and the fitness cannot converge to the global optimum.

In conclusion, in the first and third conditions, CSA works well because local objective functions of two agents have the same optimal solution. However, in the second and fourth conditions, CSA cannot adjust the step size properly because the optimal solutions of two local objective functions are different. In addition to CSA, other traditional adaptation methods, like success-based rules and self-adaptation, also face similar challenges. The key reason is that agents only care about their own evolution paths and ignore neighboring coordination.

Considering the above challenges of existing step adaptation methods when addressing distributed optimization

problems, we design a cooperative and cumulative step adaptation (CCSA) method, which utilizes neighboring information to assist step adaptation. The detailed procedure and principle of CCSA are elaborated in the following section.

IV. Multi-Agent Evolution Strategy with CCSA

In this work, we propose a multi-agent evolution strategy with cooperative and cumulative step adaptation (CCSA-DES). In the following, we elaborate on the algorithm framework, the detailed process of CCSA, the working principle of CCSA, and the consensus analysis in turn.

TABLE I
Notations in CCSA-DES

Notation	Meanings
T	Iteration number
M	Communication interval
λ	Number of offsprings
μ	Number of elite offsprings
w	Weights for offspring recombination
β	Weight of historical evolution direction
γ	Weight of current evolution direction
θ	Conflict angle threshold
r_1	Adaptation rate of inner step adaptation
r_2	Adaptation rate of outer step adaptation
$\boldsymbol{x}_i$	Local solution of agent i
α_i	Sampling step size of agent i
$\mathcal{N}_i$	Neighbors of agent i
$\mathcal{G}_i$	Neighboring evolution path of agent i
$\boldsymbol{p}_i$	Local evolution path of agent i
$\vec{\boldsymbol{v}_i}$	Current neighboring evolution direction around agent i
$\hat{\nabla g_i}$	estimated gradient

A. Framework

CCSA-DES is proposed to address black-box and non-convex distributed optimization problems. This framework works in multi-agent systems, such as smart grids and wireless sensor networks. An agent in the multi-agent system represents a physical entity, like a sensor or a power station. Due to the local data and limited sensing range, each agent i can only access its own local objective f_i. In order to optimize the global objective F in Eq. (1), it is necessary for agents to communicate and cooperate.

Fig. 2 shows the system structure and the flowchart of each agent. Agents have the ability of computation and communication. In terms of computation, each agent executes the same algorithm and procedure. In terms of communication, each agent can communicate with its neighbors. Agents are connected by a fixed and connected communication network. The communication network is usually a non-fully connected graph, which is determined by physical environments. Take the wireless sensor network as an example, two sensors are directly connected only when their distance is within the maximum communication range.

As shown in Algorithm 1, CCSA-DES consists of the following steps in each agent:

1) Initialization (line 2): At the beginning, each agent initializes the following variables. The first is the local solution $\boldsymbol{x}_i \in \mathbb{R}^D$, where D is the dimension of the distributed optimization problem. The second is the step size $\sigma_i \in \mathbb{R}$, which controls the scale of local sampling. Each agent maintains and adjusts σ_i independently. The third is the historical neighboring evolution direction $\mathcal{G}_i \in \mathbb{R}^D$ and the historical local evolution direction $\boldsymbol{p}_i \in \mathbb{R}^D$. Both of them are used for cooperative and cumulative step adaptation. Note that $\boldsymbol{p}_i$ is reinitialized at each iteration because it is used to track the recent local evolution path (line 5).
2) Local optimization (lines 6-16): This phase includes the basic processes of evolution strategies, mutation, selection, and recombination. First, the agent samples λ vectors by standard Gaussian distribution with the scale σ_i for mutation. Then, the λ offsprings are evaluated by the local objective function f_i. After sorting the offsprings according to their fitness, μ elite offsprings are selected and recombined as a new solution $\boldsymbol{x}_i^t$. Each agent will evolve M generations for local optimization.
3) Communication and cooperation (lines 18-20): After local optimization, each agent i communicates with its neighbors in the set $\mathcal{N}_i$. Among the communication messages, the local solutions of neighbors are recombined as a new solution $\boldsymbol{x}_i^{t+1}$, which is used as the starting point for the next iteration. Besides, the estimated gradient $\hat{\nabla g_i^t}$ and the neighboring evolution direction $\mathcal{G}_k^t$ are used to update $\mathcal{G}_k^{t+1}$.
4) Step adaptation (lines 15 and 23): The proposed cooperative and cumulative step adaptation method contains two parts, i.e., inner step adaptation and outer step adaptation. Inner step adaptation is used for local optimization when agents do not communicate with neighbors. Outer step adaptation is used for the communication phase, which utilizes the neighboring messages to control the step size cooperatively.

The algorithm terminates when the maximum iteration number is reached or the consensus condition is satisfied. The consensus condition for the distributed evolution strategy is as follows:

$$\begin{aligned} &\frac{1}{n}\sum_{i=1}^{n}\|\boldsymbol{x}_i-\overline{\boldsymbol{x}}\|_2^2<\epsilon \\ &\text{where}\quad \overline{\boldsymbol{x}}=\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i \end{aligned} \tag{4}$$

Here, ϵ is a small threshold close to zero, and $\overline{\boldsymbol{x}}$ is the mean vector of all local solutions. When the above condition is satisfied, it is considered that the distributed system reaches a consensus and the algorithm can terminate.

B. Cooperative and Cumulative Step Adaptation (CCSA)

Considering the limitations of CSA in the case study in Sec. III, the intuitive idea of CCSA is to increase the step

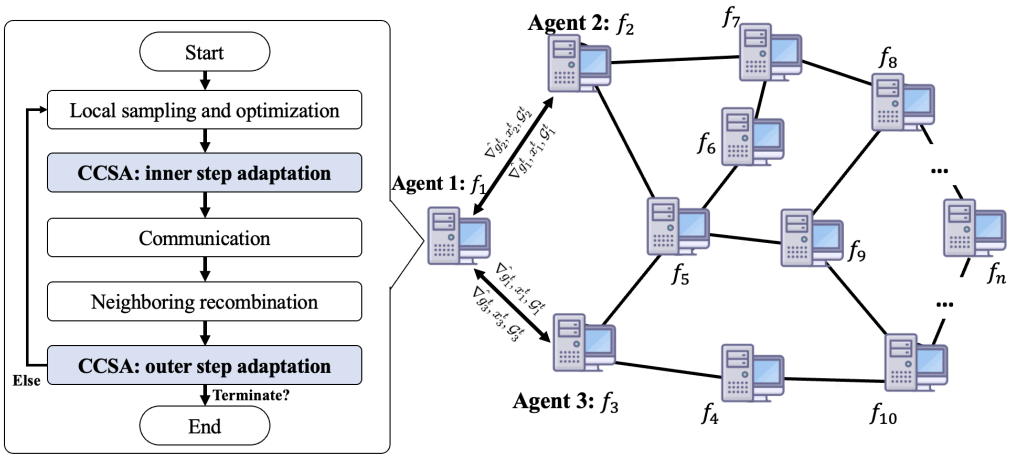


Fig. 2. System structure and procedure of CCSA-DES. CCSA-DES works in the multi-agent system, where each agent executes the same procedure. Agents may represent sensors in the wireless sensor network, or power stations in the distributed grid system, etc.

Algorithm 1 CCSA-DES in each agent

Input: Local objective function f_i

/* Initialization */
let $\boldsymbol{x}_i^0 = \mathbf{0}, \sigma_i = \sigma_0, \mathcal{G}_i = \mathbf{0}, t = 0$
while $t < T$ and consensus condition is not satisfied do
  /* Local Optimization */
  let $\boldsymbol{p}_i = \mathbf{0}$
  for $m = 0, ..., M-1$ do
    for $j = 1, ..., \lambda$ do
      Sample $\boldsymbol{z}_j \sim \mathcal{N}(\mathbf{0}, I)$
      Evaluate $y_j = f_i(\boldsymbol{x}_i + \sigma_i \boldsymbol{z}_j)$
    end for
    sort $\{z_j\}_{j=1,2,...,\lambda}$ according to $\{y_j\}_{j=1,2,...,\lambda}$
    $\boldsymbol{x}_i^t = \boldsymbol{x}_i^t + \sum_{j=1}^{\mu} w_j * \sigma_i \boldsymbol{z}_j$
    /* Inner step adaptation of CCSA */
    Update $\boldsymbol{p}_i$ according to Eq. (2)
    $\sigma_i = \sigma_i \cdot \exp(\tau \cdot r_1(\frac{\|\boldsymbol{p}_i\|}{\mathbb{E}(\|\boldsymbol{p}_i\|)} - 1))$
  end for
  /* Communication and cooperation*/
  Send $\boldsymbol{x}_i^t, \hat{\nabla g_i^t}, \mathcal{G}_i^t$ to neighbors $k \in \mathcal{N}_i$
  Receive $\boldsymbol{x}_k^t, \hat{\nabla g_k^t}, \mathcal{G}_k^t$ from neighbors $k \in \mathcal{N}_i$
  $\boldsymbol{x}_i^{t+1} = \sum_{k \in \mathcal{N}_i \cup \{i\}} W_{i,k} \boldsymbol{x}_k^t$
  /* Outer step adaptation of CCSA */
  Update $\mathcal{G}_i^{t+1}$ according to Eq. (5) - (7)
  $\sigma_i = \sigma_i \cdot \exp(r_2(\|\mathcal{G}_i\| - 1))$
  $t = t + 1$
end while

Output: x

size when the evolution direction of an agent is consistent with that of its neighbors, and decrease the step size otherwise. For this design goal, the following two problems are addressed in CCSA, i.e., how to construct an evolution path for distributed optimization, and how to adapt the step size based on the evolution path.

1) How to construct a neighboring evolution path: An evolution path is cumulated by multiple successive steps. Therefore, we define the single step for distributed optimization by the neighboring direction vector $\vec{v_i}$ of the current iteration as follows:

$$\begin{aligned} \vec{v_i} &= \frac{\sum_{k \in \mathcal{N}_i \cup \{i\}} \hat{\nabla g_k}(t)}{\| \sum_{k \in \mathcal{N}_i \cup \{i\}} \hat{\nabla g_k}(t) \|} \\ \text{s.t.} \quad \hat{\nabla g_k}(t) &= \frac{[f_k(\boldsymbol{x}_k^{t+1}) - f_k(\boldsymbol{x}_k^t)] \Delta \boldsymbol{x}_k}{\|\Delta \boldsymbol{x}_k\|_2^2} \end{aligned} \tag{5}$$

where $\Delta \boldsymbol{x}_i$ is the moving vector of the agent i at the t-th iteration, which is defined by $\Delta \boldsymbol{x}_i = \boldsymbol{x}_i^{t+1} - \boldsymbol{x}_i^t$. Because CCSA-DES is proposed for black-box optimization, we use $\hat{\nabla g_i}$ to estimate the gradient of local objective functions according to the change of fitness and positions. The neighboring direction vector $\vec{v_i}$ reflects the integrated movement direction of neighboring agents, which provides the basis for the following path analysis.

To adapt the step size properly, the length of the evolution path is expected to reflect the consistency of neighboring agents. Therefore, we design the cumulation method of the evolution path $\mathcal{G}_i$ as follows:

$$\begin{aligned} & \mathcal{G}_i^{t+1} = \beta \mathcal{G}_i^t + \gamma \vec{v_i} \\ \text{s.t.} \quad & \beta^2 + \gamma^2 + 2\beta\gamma cos(\theta) = 1 \\ & \beta, \gamma > 0 \quad \text{and} \quad \theta = (1 - \frac{t}{T}) 90^\circ \end{aligned} \tag{6}$$

where β and γ are weights of the historical path and the current evolving direction respectively. θ is the conflict angle threshold determining whether the conflict between the current direction and the historical direction is within an acceptable range. At the beginning, the threshold θ is set to the orthogonal angle 90°. As the iteration number increases, θ decays to zero gradually. This indicates that the conflict detection of CCSA is getting stricter. The working principle is explained in Sec. IV-C.

Because the neighboring direction vector $\vec{v_i}$ can only combine the path information of one-hop neighbors, we

make the information diffuse in the multi-agent system to help the whole system coordinate:

$$\mathcal{G}_i^t = \sum_{k \in \mathcal{N}_i \cup \{i\}} W_{i,k} \mathcal{G}_k^t \tag{7}$$

where W is a stochastic matrix, representing the adjacent matrix of the multi-agent system.

2) How to adapt the step size: According to Eq. (6), the evolution path is long when the evolution directions of neighboring agents are consistent, and short otherwise. Therefore, we adapt the step size based on the length of the evolution path. Specifically, CCSA consists of two parts, outer adaptation for the phase of neighboring cooperation, and inner adaptation for the local optimization.

First, at the phase of neighboring cooperation, the step size is adjusted according to the neighboring evolution path $\mathcal{G}_i$. The outer adaptation is designed as follows:

$$\sigma_i = \sigma_i \cdot \exp(r_2(\|\mathcal{G}_i^t\| - 1)) \tag{8}$$

where r_2 is the outer adaptation rate. The step size will increase when the length of $\mathcal{G}_i$ is larger than one, and decrease when the length of $\mathcal{G}_i$ is smaller than one.

Second, at the phase of local optimization, each agent can only access its local evolution directions. Thus, a local evolution path $\boldsymbol{p}_i$ is constructed based on Eq. (2) for each agent i. However, according to the discussion in Sec. III, an agent focusing only on its local evolution path is not conducive to the optimization of the global objective function. Therefore, we use the neighboring evolution path at the previous iteration $\mathcal{G}_i^{t-1}$ to assist the inner adaptation:

$$\begin{aligned} \sigma_i &= \sigma_i \cdot \exp[\tau \cdot r_1(\frac{\|\boldsymbol{p}_i\|}{\mathbb{E}(\|\boldsymbol{p}_i\|)} - 1)] \\ \text{where } \tau &= \begin{cases} 1 & (\frac{\|\boldsymbol{p}_i\|}{\mathbb{E}(\|\boldsymbol{p}_i\|)} - 1)(\|\mathcal{G}_i^{t-1}\| - 1) > 0 \\ 0 & else \end{cases} \end{aligned} \tag{9}$$

where r_1 is the inner adaptation rate and τ is a binary variable that controls whether the step size should be adjusted. Specifically, τ is activated when the local evolution path and the neighboring evolution path have the same trend of step size adaptation, such as the first or fourth condition of Sec. III. Otherwise, τ is deactivated and the step size is not adjusted.

C. The working principle of CCSA

In this subsection, we analyze the working principle of CCSA based on the following four conditions:

	$\|\mathcal{G}_i^t\| \geq 1$	$\|\mathcal{G}_i^t\| < 1$
$\angle(\mathcal{G}_i^t, \vec{v_i}) > \theta$	condition 1	condition 4
$\angle(\mathcal{G}_i^t, \vec{v_i}) \leq \theta$	condition 2	condition 3

(1) When $\|\mathcal{G}_i^t\| \geq 1$, the step size is in an increasing trend originally. If $\angle(\mathcal{G}_i^t, \vec{v_i}) > \theta$, we have the following derivation:

$$\begin{aligned} &\|\mathcal{G}_i^{t+1}\|^2 - \|\mathcal{G}_i^t\|^2 \\ =&\beta^2\|\mathcal{G}_i^t\|^2 + \gamma^2 + 2\beta\gamma\|\mathcal{G}_i^t\| cos(\angle(\mathcal{G}_i^t, \vec{v_i})) - \|\mathcal{G}_i^t\|^2 \\ <&(\beta^2 - 1)\|\mathcal{G}_i^t\|^2 + \gamma^2 + 2\beta\gamma\|\mathcal{G}_i^t\| cos(\theta) \\ =&(\beta^2 - 1)\|\mathcal{G}_i^t\|^2 + \gamma^2 + (1 - \beta^2 - \gamma^2)\|\mathcal{G}_i^t\| \\ =&(\beta^2 - 1)(\|\mathcal{G}_i^t\|^2 - \|\mathcal{G}_i^t\|) - \gamma^2(\|\mathcal{G}_i^t\| - 1) \\ =&(\|\mathcal{G}_i^t\| - 1)(\|\mathcal{G}_i^t\|(\beta^2 - 1) - \gamma^2) \leq 0 \end{aligned} \tag{10}$$

Here, the first inequality sign is obtained due to $cos(\angle(\mathcal{G}_i^t, \vec{v_i})) < cos(\theta)$, and the second inequality sign is due to $\|\mathcal{G}_i^t\| \geq 1$ and $\beta < 1$. According to Eq. (10), the length of $\mathcal{G}_i$ will decrease based on the proposed strategy. This indicates that the trend of step size growth will slow down when the current direction is inconsistent with the historical neighboring evolution path.

(2) When $\|\mathcal{G}_i^t\| \geq 1$, the step size is in an increasing trend originally. If $\angle(\mathcal{G}_i^t, \vec{v_i}) \leq \theta$, we have the following derivation:

$$\begin{aligned} \|\mathcal{G}_i^{t+1}\|^2 =&\beta^2\|\mathcal{G}_i^t\|^2 + \gamma^2 + 2\beta\gamma\|\mathcal{G}_i^t\| cos(\angle(\mathcal{G}_i^t, \vec{v_i})) \\ \geq&\beta^2 + \gamma^2 + 2\beta\gamma cos(\angle(\mathcal{G}_i^t, \vec{v_i})) \\ \geq&\beta^2 + \gamma^2 + 2\beta\gamma cos(\theta) = 1 \end{aligned} \tag{11}$$

Here, the first inequality sign is obtained due to $\|\mathcal{G}_i^t\| \geq 1$ and the second one is due to $cos(\angle(\mathcal{G}_i^t, \vec{v_i})) \geq cos(\theta)$. According to Eq. (11), the length of $\mathcal{G}_i$ will remain larger than one based on the proposed strategy. This indicates that the step size will keep increasing when the current direction is consistent with the historical neighboring evolution path.

(3) When $\|\mathcal{G}_i^t\| < 1$, the step size is in an decreasing trend originally. Similar to the derivation in the above conditions, according to Eq. (12), the length of $\mathcal{G}_i$ will increase if $\angle(\mathcal{G}_i^t, \vec{v_i}) \leq \theta$. This indicates that the trend of decreasing step size will slow down when the current direction is consistent with the historical neighboring evolution path.

$$\begin{aligned} &\|\mathcal{G}_i^{t+1}\|^2 - \|\mathcal{G}_i^t\|^2 \\ =&\beta^2\|\mathcal{G}_i^t\|^2 + \gamma^2 + 2\beta\gamma\|\mathcal{G}_i^t\| cos(\angle(\mathcal{G}_i^t, \vec{v_i})) - \|\mathcal{G}_i^t\|^2 \\ \geq&(\beta^2 - 1)\|\mathcal{G}_i^t\|^2 + \gamma^2 + 2\beta\gamma\|\mathcal{G}_i^t\| cos(\theta) \\ =&(\beta^2 - 1)\|\mathcal{G}_i^t\|^2 + \gamma^2 + (1 - \beta^2 - \gamma^2)\|\mathcal{G}_i^t\| \\ =&(\beta^2 - 1)(\|\mathcal{G}_i^t\|^2 - \|\mathcal{G}_i^t\|) - \gamma^2(\|\mathcal{G}_i^t\| - 1) \\ =&(\|\mathcal{G}_i^t\| - 1)(\|\mathcal{G}_i^t\|(\beta^2 - 1) - \gamma^2) > 0 \end{aligned} \tag{12}$$

(4) When $\|\mathcal{G}_i^t\| < 1$, the step size is in an decreasing trend originally. Similar to the derivation in the above conditions, according to Eq. (13), the length of $\mathcal{G}_i$ will remain smaller than one if $\angle(\mathcal{G}_i^t, \vec{v_i}) > \theta$. This indicates that the step size will keep decreasing when the current direction is inconsistent with the historical neighboring evolution path.

$$\begin{aligned}\|\mathcal{G}_i^{t+1}\|^2 =&\beta^2\|\mathcal{G}_i^t\|^2+\gamma^2+2\beta\gamma\|\mathcal{G}_i^t\|cos(\angle(\mathcal{G}_i^t,\vec{v_i}))\\ <&\beta^2+\gamma^2+2\beta\gamma cos(\angle(\mathcal{G}_i^t,\vec{v_i}))\\ <&\beta^2+\gamma^2+2\beta\gamma cos(\theta)=1\end{aligned} \tag{13}$$

As a result, the working principles of CCSA are summarized as follows:

$$\|\mathcal{G}_i^{t+1}\|\begin{cases}<\|\mathcal{G}_i^t\| & \text{if } \|\mathcal{G}_i^t\|\geq 1 \text{ and } \angle(\mathcal{G}_i^t,\vec{v_i})>\theta\\ \geq 1 & \text{if } \|\mathcal{G}_i^t\|\geq 1 \text{ and } \angle(\mathcal{G}_i^t,\vec{v_i})\leq\theta\\ >\|\mathcal{G}_i^t\| & \text{if } \|\mathcal{G}_i^t\|<1 \text{ and } \angle(\mathcal{G}_i^t,\vec{v_i})\leq\theta\\ <1 & \text{if } \|\mathcal{G}_i^t\|<1 \text{ and } \angle(\mathcal{G}_i^t,\vec{v_i})>\theta\end{cases} \tag{14}$$

We also test the behavior of the evolution strategy with CCSA on the four typical conditions of distributed optimization. Fig. 1(i)-(l) record the step size and fitness when CCSA is used.

1) CCSA has a step adaptation behavior similar to CSA in the first and third conditions. Both of them work well in these two conditions. This is because the two local objective functions have the same optimal solution in these two conditions, and the centralized optimization technique CSA can address it well.
2) The second condition shows the advantage of CCSA over CSA for distributed optimization. Although the evolving direction of agent 1 is consistent with its own evolution path, the evolution directions of the two agents are in conflict. In this case, CCSA can detect the conflict and decrease the step size according to Fig. 1(j). As a result, the final fitness is lower than 10^{-6}, which is much better than that of CSA.
3) The fourth condition also shows the effect of CCSA. Although both CSA and CCSA decrease the step size at the beginning, the step size of CSA fluctuates in the late stage because it cannot detect the conflict between the two agents. Differently, CCSA keeps decreasing the step size and the final fitness is lower than 10^{-5}.

In conclusion, CCSA can better address typical conditions of distributed optimization through neighboring evolution path analysis. In Sec. V, the effectiveness of CCSA on more complicated problems is verified.

D. Consensus Discussion

In this part, we discuss the system consensus of CCSA-DES. In the proposed algorithm, the conflict angle threshold θ is set to decrease from an initial value to zero according to Eq. (6). In the following, we analyze the system after θ becomes zero, which means that $\beta+\gamma=1$. In order to avoid confusion between the time symbol t and the exponent calculation, $\mathcal{G}_i^t$ is denoted by $\mathcal{G}_i(t)$ in the following, and so are other variables.

First, we describe the neighboring evolution path of all agents together by a dynamical equation according to Eq. (6) and (7) in Sec. IV.B:

$$\begin{aligned}&\mathbb{G}(t+1)=\beta W\mathbb{G}(t)+\gamma\boldsymbol{V}(t)\\ &\text{where } \mathbb{G}(t)=\begin{pmatrix}\mathcal{G}_1(t)\\ \mathcal{G}_2(t)\\ \cdots\\ \mathcal{G}_n(t)\end{pmatrix} \text{ and } \boldsymbol{V}(t)=\begin{pmatrix}\vec{v_1}(t)\\ \vec{v_2}(t)\\ \cdots\\ \vec{v_n}(t)\end{pmatrix}\end{aligned} \tag{15}$$

Then, the state of the system at the t-th iteration can be derived:

$$\begin{aligned}\mathbb{G}(t)&=\beta W\mathbb{G}(t-1)+\gamma\boldsymbol{V}(t-1)\\ &=\beta W(\beta W\mathbb{G}(t-2)+\gamma\boldsymbol{V}(t-2))+\gamma\boldsymbol{V}(t-1)\\ &=\beta^2W^2\mathbb{G}(t-2)+\beta\gamma W\boldsymbol{V}(t-2)+\gamma\boldsymbol{V}(t-1)\\ &=\cdots\\ &=\beta^tW^t\mathbb{G}^0+\gamma\sum_{k=0}^{t-1}\beta^kW^k\boldsymbol{V}(t-k-1)\end{aligned} \tag{16}$$

When the iteration number $t\to\infty$, the first term approaches zero because $0<\beta<1$. In this case, consider the length of the i-th row of the matrix $\mathcal{G}_i(t)$, we have:

$$\|\mathcal{G}_i(t)\|_{t\to\infty}=\|\gamma\sum_{k=0}^{t}\beta^k[W^k]_{i*}\boldsymbol{V}(t-k-1)\| \tag{17}$$

where $[W^k]_{i*}$ is the i-th row of the matrix W^k. In the following, we discuss the length of the evolution path $\mathcal{G}_i(t)$ in two conditions.

1) First, suppose the evolution directions of agents are not consistent all the time, i.e., $\vec{v_i}(t)\neq\vec{v_j}(t)(\exists 1\leq i,j\leq n,\ \exists t=1,2,...,\infty)$. In this case, we have the following derivation:

$$\begin{aligned}\|\mathcal{G}_i(t)\|_{t\to\infty}&=\|\gamma\sum_{k=0}^{\infty}\beta^k[W^k]_{i*}\boldsymbol{V}(t-k)\| &(18)\\ &\leq\gamma\sum_{k=0}^{\infty}\beta^k\|[W^k]_{i*}\boldsymbol{V}(t-k)\| &(19)\\ &=\gamma\sum_{k=0}^{\infty}\beta^k\|\sum_{j=1}^{n}[W^k]_{i,j}\vec{v_j}(t-k)\| &(20)\\ &<\gamma\sum_{k=0}^{\infty}\beta^k\sum_{j=1}^{n}[W^k]_{i,j}\|\vec{v_j}(t-k)\| &(21)\\ &=\gamma\sum_{k=0}^{\infty}\beta^k\sum_{j=1}^{n}[W^k]_{i,j} &(22)\\ &=\gamma\sum_{k=0}^{\infty}\beta^k &(23)\end{aligned}$$

Here, Ineqs. (19) and (21) are derived from the triangle inequality, while the equation sign does not hold in (21). The equal sign only holds if the evolution directions of agents are in the same direction, i.e., $\vec{v_i}(t)=\vec{v_j}(t)\ \forall 1\leq i,j\leq n$ and $t=1,2,...,\infty$. This does not satisfy the above-mentioned condition.

For the left derivations, Eq. (22) holds because $\vec{v_j}$ is a unit vector and its length equals one. Eq. (23) holds due to the properties of the stochastic matrix that

each row of a stochastic matrix sums to one. Because W is a stochastic matrix, W^k is still a stochastic matrix for $k \in \mathbb{N}$.

2) Second, suppose the evolution directions of all agents are consistent without any conflict, i.e., $\vec{v_1}(t) = \vec{v_2}(t) = \cdots = \vec{v_n}(t) = \vec{v_*}(t), \forall t = 1, 2, ..., \infty$. In this case, we have:

$$\|\mathcal{G}_i(t)\|_{t\to\infty} = \|\gamma \sum_{k=0}^{\infty} \beta^k [W^k]_{i*} \boldsymbol{V}(t-k)\| \tag{24}$$

$$= \|\gamma \sum_{k=0}^{\infty} \beta^k \sum_{j=1}^{n} [W^k]_{i,j} \vec{v_*}(t-k)\| \tag{25}$$

$$= \|\gamma \sum_{k=0}^{\infty} \beta^k \vec{v_*}(t-k)\| \tag{26}$$

$$< \gamma \sum_{k=0}^{\infty} \beta^k \|\vec{v_*}(t-k)\| \tag{27}$$

$$= \gamma \sum_{k=0}^{\infty} \beta^k \tag{28}$$

Here, Ineq. (27) is derived from the triangle inequality. Suppose the optimization problem has a minimization solution, the equation sign does not hold. The equal sign holds only when the evolution directions at all the iterations are in the same direction, i.e., $\vec{v_*}(t_1) = \vec{v_*}(t_2)\ \forall t_1, t_2 \in \{1, 2, ..., \infty\}$. In this case, according to the working principle of CCSA, the step size will keep increasing, so the solution will become infinite. However, this does not occur in a problem with a minimization solution, because the optimization algorithm moves in the direction of a decline in the fitness value.

For the other derivations, Eq. (26) is due to the properties of the stochastic matrix that each row of a stochastic matrix sums to one. Eq. (28) is because the length of the unit vector $\vec{v_*}(t-k)$ is one.

For the above two conditions, we have the same result that $\|\mathcal{G}_i(t)\|_{t\to\infty} < \gamma \sum_{k=0}^{\infty} \beta^k$. Considering the exponential power series of β is $\frac{1}{1-\beta}$ due to $|\beta| < 1$, we have:

$$\|\mathcal{G}_i(t)\|_{t\to\infty} < \gamma \sum_{k=0}^{\infty} \beta^k = \frac{\gamma}{1-\beta} = 1 \tag{29}$$

According to Sec. IV-B2, when $\|\mathcal{G}_i(t)\| < 1$, the step size will be decreased in outer adaptation, and be decreased or remain unchanged in inner adaptation. Taken together, the step size will keep decreasing and approach zero when the iteration number $t \to \infty$. In this case, the phase of local optimization does not change the solutions of agents because the sampling step size approaches zero (see line 9 in Algorithm 1). Thus, we can describe the solutions of all agents by a dynamical equation as follows:

$$\mathbb{X}(t+1) = W\mathbb{X}(t)$$
$$\text{where } \mathbb{X}(t) = \begin{pmatrix} \boldsymbol{x}_1(t) \\ \boldsymbol{x}_2(t) \\ \cdots \\ \boldsymbol{x}_n(t) \end{pmatrix} \tag{30}$$

According to the consensus theory [8], when $t \to \infty$, we have:

$$\boldsymbol{x}_i^* = \boldsymbol{x}_j^* \quad \forall 1 \le i, j \le n \tag{31}$$

As a result, the multi-agent system can finally reach a consensus.

V. Experiments

In this section, we conduct the following experiments, i.e., parameter investigation, ablation experiments, and comparison with state-of-the-art algorithms.

A. Experiment Settings and Benchmarks

To verify the performance of the proposed algorithm, we adopt the black-box and non-convex distributed optimization benchmark functions proposed in [45]. The general definition of the benchmark function is as follows:

$$F(\boldsymbol{x}) = \sum_{i=1}^{n} f_i(\boldsymbol{x}) \tag{32}$$

$$\text{where} \quad f_i(\boldsymbol{x}) = f^i_{elementary}(\boldsymbol{z}) + c\sum_{j=1}^{D} [A]_{ij} \boldsymbol{z}_j \tag{33}$$

$$\text{and} \quad \boldsymbol{z} = T^{0.2}_{asy}(T_{osz}(R(\boldsymbol{x} - \boldsymbol{x}'))) \tag{34}$$

where $f_i(*)$ is the local objective function of the i-th agent, and $F(*)$ is the global objective function. As shown in Eq. (33), the local objective function consists of a non-convex elementary function and a linear term. In Eq. (34), the solution x is processed by four operations: (1) Shift: x is shifted by a randomly generated vector x'; (2) Rotation: The solution is rotated by a randomly generated orthogonal matrix R; (3) Unsmooth: The solution is put into a transformation function T_{osz} to create smooth local irregularities. (4) Asymmetric: The vector is put into a transformation function T_{asy} to break the symmetry of the symmetric functions.

Table II shows the function configuration of the benchmark set. Elementary functions include Elliptic, Schwefel, Rosenbrock, and Griewank. The elementary function $f^i_{elementary}(*)$ is the same for each agent in F1-F3, but different for agents in F4-F9. Referring to the settings in [45], the number of agents is set to 20, the problem dimension is set to 100, and the maximum number of objective function evaluations is set to 1.5E+6 for each agent.

The parameters of CCSA-DES are partly set according to existing studies and partly determined by parameter investigation. According to [35], the offspring number λ is set to 34 and the number of elite offspring μ is set to 17. The weight of recombination is set to $w_j = log(\mu +$

TABLE II
Configuration of benchmark functions

Benchmark functions	Elementary function
F1	Elliptic
F2	Schwefel
F3	Rosenbrock
F4	Elliptic & Schwefel
F5	Elliptic & Rosenbrock
F6	Schwefel & Rosenbrock
F7	Elliptic & Griewank
F8	Schwefel & Griewank
F9	Rosenbrock & Griewank

$0.5) - log(j)$ for $j = 1, 2, ..., \mu$. The adaptation weight r_1 and r_2 is set to 0.01 and 0.001 respectively. The initial step size σ_0 is set to 1E-6. Other parameters, including the historical weight β and the communication interval M, are investigated through experiments, which are discussed in Sec. V-B.

The following experiments are conducted on a cluster composed of Intel(R) Xeon(R) CPU E5-2699 v3@2.30GHz 36 cores and CentOS 7.5 x64 system. Each experiment is repeated for 25 times independently.

B. Parameter Investigation

In this subsection, we investigate three important parameters in CCSA-DES. In brief, the historical weight affects the sensitivity of CCSA, the communication interval affects the frequency of CCSA, and the conflict angle threshold affects the tendency of CCSA. Specific experiments and analyses are as follows.

1) Historical Weight: The historical weight determines the importance of the historical path in the process of evolution path accumulation, as shown in Eq. (6). In order to investigate the impact of historical weight on the algorithm, we test the performance of CCSA-DES when the historical weight β is set to 0.6, 0.7, 0.8, 0.9, 0.95, 0.97, 0.99, and 0.999, respectively. According to the results in Fig. S1 of the supplementary material, the fitness value shows a V-shaped curve as the historical weight changes in most tested functions. This indicates that too large or too small values are not suitable for the historical weight.

When the historical weight is too large, especially when the weight is set to 0.999, the fitness value is much larger than others in Fig. S1. This is because the step size adaptation becomes sluggish and it cannot catch the conflict of evolution paths of different agents in time. Consider the following condition, assuming that the direction of evolution has remained consistent over the past period of time, then the length of the evolution path $\mathcal{G}_i$ is large according to the working principle in Sec. IV-C. In this state, when the optimized region moves from the consistent region to the conflict region, the step size is expected reduced. However, due to the long evolution path and large historical weight, the step size will still maintain the trend of growth. As a result, the algorithm cannot be stabilized in the global optimal region, so the optimization result is poor.

When the historical weight is too small, especially when the weight is set to 0.6, the algorithm performance is quite worse than others from functions F4 to F9. According to Eq. (6), the current direction weight γ is 0.8 when β is 0.6 and θ is 90°. In this case, the step size adaptation becomes shortsighted, focusing only on the near-term evolution path. As a result, the trend of step size adaptation will change frequently, resulting in the optimization becoming unstable.

Considering the overall performance, 0.97 is an appropriate parameter setting for the historical weight. When β is set to 0.97, the algorithm achieves the best performance in five of nine functions. Besides in F5 and F6, the difference between the fitness of 0.97 and the best fitness is not significant according to the Wilcoxon rank sum test. To be specific, the performance of the two algorithms is considered to be significantly different only when the p-value is smaller than 0.05 in the Wilcoxon rank sum test. The p-value between 0.97 and 0.99 on F5 is 0.202, and the p-value between 0.97 and 0.95 on F6 is 0.148. Therefore, we set the historical weight to 0.97 in the following experiments.

2) Communication Interval: The communication interval of CCSA-DES determines the number of generations at the phase of local optimization. In this part, we aim to investigate the impact of the communication interval on CCSA-DES and find an appropriate setting. We test the performance of the algorithm when the communication interval is set to 1, 2, 5, 10, and 20, respectively. Fig. S2 of the supplementary material shows how the fitness curve changes with communication bits. The experiment findings are as follows.

In functions F1-F5, the larger the communication interval is, the faster the fitness curve drops under the same communication bits. This is because the agents explore more at the phase of local optimization when the communication interval is larger. However, this phenomenon does not apply to all benchmark functions. In functions F7-F9, the fitness curve shows a sharp rebound when the interval is set to 20 and 10. This is due to the lack of necessary cooperation among agents. To be specific, at the phase of local optimization, an agent can only adjust the step size according to its own local evolution path. In general, the optimal region of the local objective function is different from that of the global objective function. In this case, when the communication interval is large, agents tend to increase the step size and search for their own optimal regions. As a result, agents will jump out of the global optimal region and cause the global fitness value to rebound.

In conclusion, a large communication interval may weaken the cooperation of agents, and a small communication interval will have a large communication cost. To balance these two constraints, we set the default communication interval to five in the following experiments. When the interval is set to five, CCSA-DES always has a good performance in the tested nine functions.

3) Conflict Angle Threshold: The conflict angle threshold affects the evolution path and further affects the step size adaptation. In CCSA-DES, the conflict angle threshold decays gradually with optimization as shown in Eq. (6). In this part, we conduct experiments to verify the advantage of the decaying angle threshold over the static angle threshold. In this experiment, the static angle threshold is set to 90°, 70°, and 50° respectively. From experiment results in Fig. S3 of the supplementary material, we can conclude the following phenomenons.

First, when the static angle threshold is set to 90°, the fitness curve of F1 shows a sharp rebound. According to the working principle of CCSA in Sec. IV-C, a fixed and large angle will make the conflicting judgment more relaxed. So that the algorithm tends to increase the step size. As a result, the sampling step size of the algorithm may become too large, causing the solution to jump out of the original convergence region. This is the reason that the fitness value rebounds on F1. Although the algorithm performs well on the other eight functions, it is still not an ideal setting because it cannot adapt to different problems.

Second, when the static angle threshold is set to 50°, the fitness curves on all functions are almost stagnant from the beginning. This is because a fixed and small angle will make the conflicting judgment strict. According to the working principle of CCSA, the sampling step size tends to decrease once the algorithm starts. In this case, the local optimization of agents is limited to a small region around the initial position, so the algorithm cannot fully explore the optimization space.

Third, when the static angle threshold is set to 70°, the performance of the algorithm is as well as that of the decaying angle threshold from functions F1 to F3. However, it has a stagnant fitness curve from functions F4 to F9.

The above three cases show that it is difficult for the static angle threshold to adapt to different optimization problems well. In contrast, the performance of CCSA-DES with decaying angle threshold performs well in all of the nine functions. This is because the decaying angle threshold considers both the early exploration and later exploitation of the algorithm. In conclusion, this experiment verifies the effectiveness of the decaying angle threshold.

C. Ablation Experiment for CCSA

In the literature of black-box distributed optimization, the step size control methods include fixed step size [23], diminishing step size [21, 24, 25, 26, 27, 28], and adaptive step size [46, 35]. Therefore, we conduct an ablation experiment to verify the advantage of the proposed CCSA over existing methods. We select CSA [35] as the adaptive step method because it is a representative method of evolution strategies. We implement the diminishing method according to a recently proposed distributed evolution strategy [21]. To test the adaptability of the step size control methods, we design nine sets of twin functions based on the existing nine benchmark functions. To be specific, each set contains two functions named Fi-L and Fi-S, where i means the index of the benchmark function. Fi-S is contracted by Fi-L 10000 times over the domain. For example, F1-L and F1-S are defined as follows:

$$\begin{aligned} \text{F1-L}(\boldsymbol{x}) &= \text{F1}(\boldsymbol{x}) \\ \text{F1-S}(\boldsymbol{x}) &= \text{F1}(10000 * \boldsymbol{x}) \end{aligned} \tag{35}$$

Table III shows the fitness of solutions on both the twin functions. The overall fitness is the mean of the above two fitness, and the p-value is computed based on the overall fitness according to the Wilcoxon rank sum test. The experiment findings are as follows.

First, observing the adaptive step method, the performance of CSA is better than CCSA on F1, and slightly worse than CCSA on F3 and F5. However, CSA has a big gap compared with CCSA in the other functions. This is because CSA adapts the step size according to the agent's own evolution path only, and cooperation among agents is lacking. CSA works well only when the optimal region of local objectives is close to that of the global objective, like F1 and F3. In contrast, when the optimal region of local objectives is different from that of the global objective, CSA may increase the step size at the optimal region of the global objective function. As a result, the algorithm may jump out of the optimal region and the global fitness increases sharply.

Second, observing the performance of the fixed step size, there is no suitable step size that can both get good performance in twin functions. Before the ablation experiment for fixed step size, we selected the best step size for the twin functions by tuning parameters respectively. The best step size is 1E-1 for Fi-L and 1E-5 for Fi-S. The performance of the algorithm with these two step sizes is shown in Table III. When the fixed step is set to a larger value 1E-1, it has a good performance from F1-L to F9-L. However, it has a poor performance from F1-S to F9-S, because a large step cannot exploit the optimal region of Fi-S. Similarly, when the fixed step is set to a smaller value 1E-5, it has a good performance from F1-S to F9-S, but poor performance from F1-L to F9-L. This is because a small step size cannot well explore Fi-L and the solution only stays near the initial position.

Third, observing the performance of the diminishing step size, the experiment finding is similar to the fixed step. There is no initial step size that can get good performance in both the twin functions. Before the ablation experiment, we also selected the best step size for the twin functions by parameter tuning. The best step size is 1.0 for Fi-L and 1E-3 for Fi-S. When the initial step is set to a larger value 1, it has a poor performance from F1-S to F9-S. When the initial step is set to a smaller value 1E-3, it has a poor performance from F1-L to F9-L.

In conclusion, the overall performance of the proposed CCSA is better than the other three types of existing step control methods.

TABLE III
Ablation experiment for the proposed step adaptation method CCSA

benchmark		proposed: CCSA	adaptive step: CSA	fixed step: large value	fixed step: small value	decaying step: large initial value	decaying step: small initial value
F1	fitness on F1-L	1.51E+06	5.21E+05	5.21E+05	9.87E+10	4.06E+06	1.89E+11
	fitness on F1-S	1.36E+06	6.18E+05	3.74E+11	4.31E+05	1.53E+10	2.38E+07
	overall fitness	1.44E+06	5.70E+05	1.87E+11	4.93E+10	7.66E+09	9.46E+10
	p-value	-	2.80e-04*	2.80e-04#	2.80e-04#	8.74e-08#	8.74e-08#
F2	fitness on F2-L	5.92E+05	1.76E+287	5.82E+05	2.60E+17	1.02E+06	4.09E+23
	fitness on F2-S	4.25E+05	9.89E+16	3.33E+11	6.21E+05	9.53E+10	3.64E+05
	overall fitness	5.08E+05	8.79E+286	1.66E+11	1.30E+17	4.77E+10	2.05E+23
	p-value	-	2.80e-04#	2.80e-04#	2.80e-04#	8.74e-08#	8.74e-08#
F3	fitness on F3-L	2.80E+02	1.22E+03	3.62E+02	3.56E+12	5.23E+02	1.74E+13
	fitness on F3-S	3.12E+02	1.22E+03	4.18E+13	4.02E+02	3.06E+12	8.04E+02
	overall fitness	2.96E+02	1.22E+03	2.09E+13	1.78E+12	1.53E+12	8.68E+12
	p-value	-	2.80e-04#	2.80e-04#	2.80e-04#	8.74e-08#	8.74e-08#
F4	fitness on F4-L	1.35E+06	1.01E+305	9.50E+05	2.40E+17	1.22E+06	5.68E+23
	fitness on F4-S	1.21E+06	6.77E+305	2.61E+25	9.79E+05	2.72E+10	6.75E+06
	overall fitness	1.28E+06	3.89E+305	1.31E+25	1.20E+17	1.36E+10	2.84E+23
	p-value	-	2.80e-04#	2.80e-04#	2.80e-04#	8.74e-08#	8.74e-08#
F5	fitness on F5-L	2.97E+03	1.29E+05	1.52E+04	2.13E+12	1.73E+04	8.15E+12
	fitness on F5-S	2.99E+03	1.37E+05	7.80E+13	1.52E+04	1.27E+12	1.27E+05
	overall fitness	2.98E+03	1.33E+05	3.90E+13	1.06E+12	6.36E+11	4.07E+12
	p-value	-	2.80e-04#	2.80e-04#	2.80e-04#	8.74e-08#	8.74e-08#
F6	fitness on F6-L	1.49E+03	6.67E+294	3.38E+02	1.69E+17	4.71E+02	3.45E+21
	fitness on F6-S	1.28E+03	2.62E+304	7.32E+17	3.66E+02	1.43E+14	5.49E+03
	overall fitness	1.39E+03	1.31E+304	3.66E+17	8.47E+16	7.13E+13	1.72E+21
	p-value	-	2.80e-04#	2.80e-04#	2.80e-04#	8.74e-08#	8.74e-08#
F7	fitness on F7-L	8.18E+06	1.73E+18	4.54E+06	5.79E+10	6.26E+06	9.44E+10
	fitness on F7-S	2.05E+07	1.68E+18	4.48E+13	4.48E+06	7.01E+13	1.81E+08
	overall fitness	1.44E+07	1.70E+18	2.24E+13	2.89E+10	3.51E+13	4.73E+10
	p-value	-	2.80e-04#	2.80e-04#	2.80e-04#	8.74e-08#	8.74e-08#
F8	fitness on F8-L	1.77E+06	9.48E+217	1.41E+06	3.41E+16	2.59E+06	1.38E+24
	fitness on F8-S	2.58E+06	2.91E+124	1.58E+15	1.46E+06	1.29E+37	7.94E+07
	overall fitness	2.18E+06	4.74E+217	7.89E+14	1.71E+16	6.47E+36	6.89E+23
	p-value	-	2.80e-04#	2.80e-04#	2.80e-04#	8.74e-08#	8.74e-08#
F9	fitness on F9-L	1.53E+03	9.63E+27	3.73E+02	1.90E+12	1.11E+03	8.40E+12
	fitness on F9-S	1.49E+03	1.05E+28	2.33E+14	4.05E+02	5.23E+15	2.85E+03
	overall fitness	1.51E+03	1.00E+28	1.16E+14	9.51E+11	2.62E+15	4.20E+12
	p-value	-	2.80e-04#	2.80e-04#	2.80e-04#	8.74e-08#	8.74e-08#

* The performance of the compared algorithm are significantly better than the proposed CCSA method based to the Wilcoxon rank sum test of level 0.05.
# The performance of the compared algorithm are significantly worse than the proposed CCSA method based to the Wilcoxon rank sum test of level 0.05.

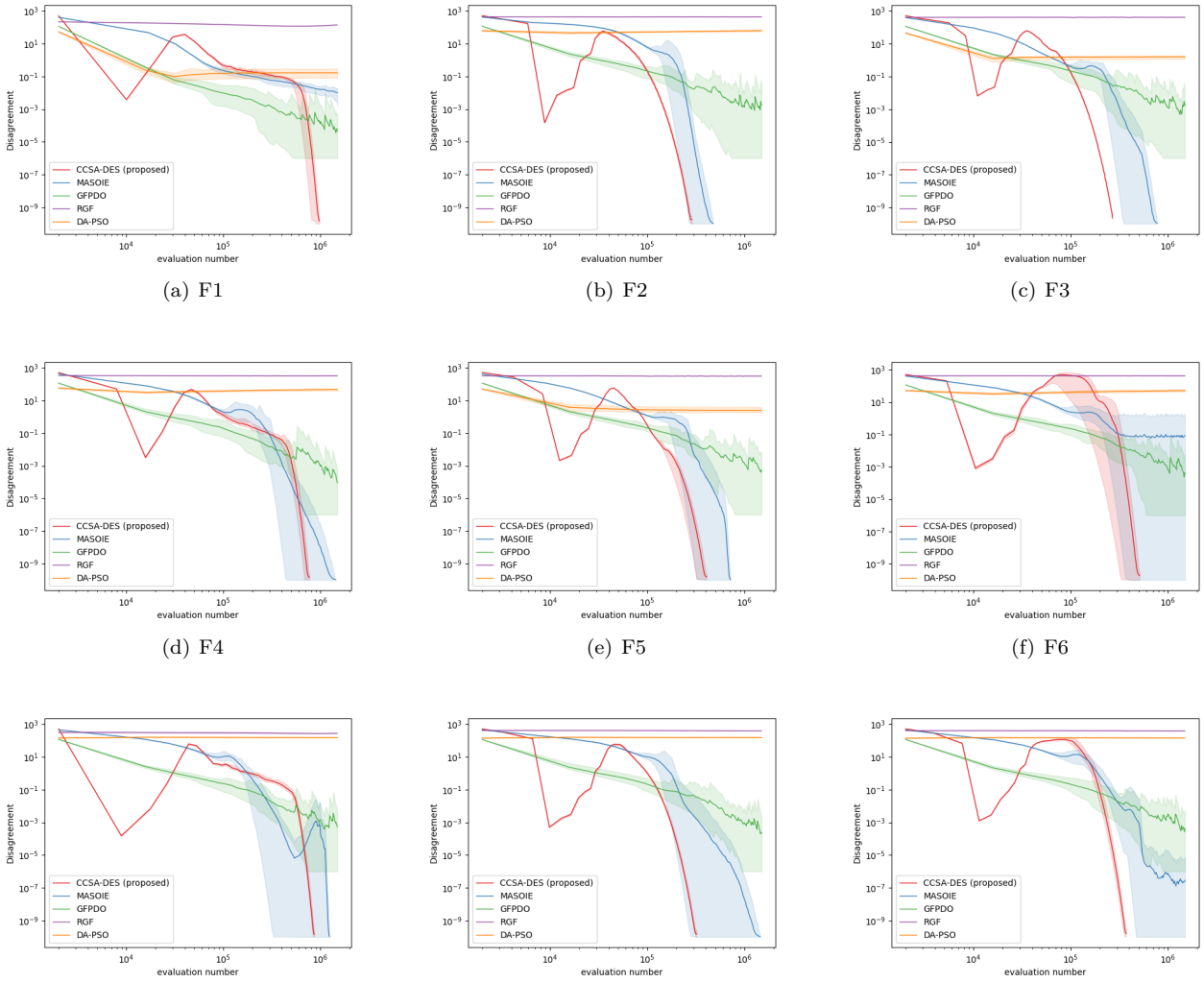
D. Comparison with State-of-the-art Algorithms

In this experiment, we compare CCSA-DES with four existing algorithms for black-box and non-convex distributed optimization. MASOIE is a multi-agent swarm optimization with adaptive internal and external learning, which is newly released [45]. RGF is a classical randomized gradient-free method [23]. GFPDO is a general framework for population-based distributed optimization [47]. DA-PSO is a distributed particle swarm optimization algorithm using the strategies of diffusion adaptation [48]. Algorithms are compared based on the same maximum evaluation number of objective functions.

The control parameters of compared algorithms are set as follows. First, the swarm size of population-based algorithms, including MASOIE, GFPDO, and DA-PSO, is set to 300 [45]. Besides, the initial communication interval of MASOIE is set to 4 [45]. For RGF, the initial step size is set to 1E-5 and the magnitude of Gaussian perturbation is set to 1E-3. For DA-PSO, the weight of inertial velocity is set to 0.5. Finally, the consensus accuracy level of GFPDO is set to 1E-3 [47].

Fig. S4 shows the solution quality of CCSA-DES and compared algorithms. According to the fitness curves in the figure, the optimization process of the algorithms can be divided into three stages, early stage, middle stage and late stage. In the early stage, the fitness value of CCSA-DES is higher than other algorithms because the initial step size is very small. MASOIE, GFPDO, and DA-PSO are population-based algorithms, which do not contain the sampling step. Their initial populations are randomly spread across the search domain, so they have smaller fitness values at the early stage. In the middle stage, the curve of CCSA-DES shows a rapid decrease trend in all the nine functions. This is because CCSA-DES increases the step size adaptively and thus the exploration speeds up. In this stage, CCSA-DES outperforms other algorithms in six functions. In the late stage, the fitness curves of the algorithms gradually flatten and converge.

We also show the consensus performance of CCSA-DES and compared algorithms in Fig. 3. The vertical axis in the figures is disagreement of the distributed system, which is usually used to evaluate the consensus



(a) F1 (b) F2 (c) F3

(d) F4 (e) F5 (f) F6

(g) F7 (h) F8 (i) F9

Fig. 3. Consensus performance of the proposed algorithm and state-of-art algorithms. The vertical axis is the disagreement of the distributed system, i.e., the difference of local solutions. The disagreement is computed according to Eq. (36).

performance of distributed algorithms. The disagreement is defined by the sum of the distance between local solutions and the average solution:

$$\mathcal{D} = \frac{1}{n}\sum_{i=1}^{n} \|\boldsymbol{x}_i - \overline{\boldsymbol{x}}\|_2^2 \tag{36}$$

where $\boldsymbol{x}_i$ is the local solution of i-th agent, and $\overline{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i$ is the average solution of all local solutions. When the local solutions are consistent, the disagreement value approaches zero, which indicates that the system reach a consensus.

According to the disagreement curves in Fig. 3, the curve of CCSA-DES also exhibits three stages during the optimization process. In the early stage, the disagreement of CCSA-DES decreases because the initial step size is very small and thus there is less local exploration. In the middle stage, the disagreement of CCSA-DES increases because CCSA-DES increases the step size adaptively and thus the exploration is enhanced. In the late stage, CCSA-DES decreases the step size gradually and the disagreement converges to 1E-10. The multi-agent system reaches a consensus finally in all the nine functions. According to the figure, CCSA-DES has the best consensus performance among the algorithms. The consensus performance of MASOIE and GFPDO is second. In particular, the disagreement of MASOIE converges to 1E-10 in six functions. The disagreement of GFPDO shows a relatively slow decreasing trend and does not reach 1E-10. Unlike the above algorithms, RGF and DA-PSO do not show a decreasing trend in the disagreement. The consensus performance of RGF is confirmed on convex functions only, so it cannot address the non-convex benchmark well. The disagreement curve of DA-PSO remains at its initial value because its algorithm design does not consider the system consensus.

Table IV shows the solution quality and communication amount of these algorithms. The communication amount records the number of times that an agent communicates

TABLE IV
Comparison with state-of-the-art algorithms

		CCSA-DES (proposed)	MASOIE[45]	GFPDO[47]	RGF[23]	DA-PSO[48]
F1	mean	1.51E+06	7.69E+04	5.85E+06	5.02E+03	4.23E+05
	median	1.49E+06	7.60E+04	5.64E+06	5.01E+03	3.77E+05
	std	1.41E+05	1.48E+04	9.08E+05	4.48E+01	2.02E+05
	p-value	-	7.08e-10*	7.08e-10#	6.88e-11*	7.98e-10*
	comm. amount	6.35E+07	6.01E+08	4.85E+09	1.50E+09	9.96E+06
F2	mean	5.92E+05	6.43E+04	8.87E+08	1.03E+11	7.80E+07
	median	5.57E+05	6.36E+04	8.98E+08	1.03E+11	7.72E+07
	std	1.11E+05	7.64E+03	1.21E+08	0.00E+00	1.09E+07
	p-value	-	7.07e-10*	7.07e-10#	4.85e-11#	7.07e-10#
	comm. amount	1.98E+07	2.11E+08	1.48E+10	1.50E+09	9.96E+06
F3	mean	2.80E+02	1.09E+04	9.58E+09	3.72E+10	1.86E+04
	median	2.75E+02	1.04E+04	9.76E+09	3.72E+10	8.25E+03
	std	2.00E+01	2.93E+03	4.17E+08	0.00E+00	2.28E+04
	p-value	-	7.08e-10#	7.08e-10#	4.86e-11#	7.08e-10#
	comm. amount	1.92E+07	2.56E+08	1.53E+10	1.50E+09	9.96E+06
F4	mean	1.35E+06	2.06E+07	1.82E+09	2.52E+09	4.23E+08
	median	1.38E+06	2.07E+07	1.82E+09	2.52E+09	4.08E+08
	std	1.86E+05	1.86E+06	1.38E+08	9.90E+05	1.07E+08
	p-value	-	7.07e-10#	7.07e-10#	1.56e-10#	7.07e-10#
	comm. amount	5.08E+07	3.25E+08	1.29E+10	1.50E+09	9.96E+06
F5	mean	2.97E+03	2.41E+06	6.95E+09	3.84E+09	6.62E+06
	median	2.93E+03	2.56E+06	6.88E+09	3.84E+09	5.74E+06
	std	2.03E+02	7.77E+05	2.83E+08	0.00E+00	2.89E+06
	p-value	-	7.08e-10#	7.08e-10#	4.86e-11#	7.08e-10#
	comm. amount	2.67E+07	2.59E+08	1.46E+10	1.50E+09	9.96E+06
F6	mean	1.49E+03	1.50E+05	6.66E+09	4.50E+10	1.12E+09
	median	1.33E+03	1.45E+05	6.66E+09	4.50E+10	9.83E+08
	std	6.20E+02	6.03E+04	3.68E+08	0.00E+00	3.75E+08
	p-value	-	7.07e-10#	7.07e-10#	4.86e-11#	7.07e-10#
	comm. amount	2.87E+07	3.28E+08	1.46E+10	1.50E+09	9.96E+06
F7	mean	8.18E+06	4.54E+07	8.99E+08	1.32E+09	1.43E+09
	median	7.85E+06	4.55E+07	9.04E+08	1.22E+09	1.35E+09
	std	1.17E+06	2.97E+06	5.10E+07	4.31E+08	3.44E+08
	p-value	-	7.08e-10#	7.08e-10#	7.08e-10#	7.08e-10#
	comm. amount	5.93E+07	3.35E+08	1.44E+10	1.50E+09	9.96E+06
F8	mean	1.77E+06	1.98E+05	5.44E+08	3.43E+10	8.58E+07
	median	1.76E+06	1.92E+05	5.55E+08	2.99E+10	5.64E+07
	std	1.59E+05	2.49E+04	8.90E+07	1.96E+10	9.63E+07
	p-value	-	7.08e-10*	7.08e-10#	7.08e-10#	7.08e-10#
	comm. amount	2.21E+07	2.55E+08	1.46E+10	1.50E+09	9.96E+06
F9	mean	1.53E+03	1.08E+07	5.69E+09	2.31E+10	2.67E+09
	median	1.56E+03	1.02E+07	5.74E+09	2.38E+10	2.44E+09
	std	2.34E+02	2.26E+06	2.89E+08	4.10E+09	6.84E+08
	p-value	-	7.07e-10#	7.07e-10#	7.07e-10#	7.07e-10#
	comm. amount	2.57E+07	3.93E+08	1.41E+10	1.50E+09	9.96E+06

* The performance of the compared algorithm are significantly better than the proposed CCSA-DES based to the Wilcoxon rank sum test of level 0.05.
# The performance of the compared algorithm are significantly worse than the proposed CCSA-DES based to the Wilcoxon rank sum test of level 0.05.

with its neighbors during the whole optimization process. Besides, p-value is the result of Wilcoxon rank sum test. It indicates whether there is a significant difference in the performance of two algorithms. According to the table, the solution quality of CCSA-DES is significantly better than other algorithms on six of nine functions. In addition, the performance of RGF on F1 is better than CCSA-DES, and the performance of MASOIE on F2 and F8 is better than CCSA-DES. However, the communication amount of RGF and MASOIE is 1 to 2 orders of magnitude higher than CCSA-DES. In the scenarios where the communication amount is limited, CCSA-DES has the advantage over MASOIE, GFPDO, and RGF.

In conclusion, the proposed CCSA-DES has better consensus performance and competitive solution quality when compared with state-of-the-art algorithms for black-box distributed optimization.

E. Validation on Realworld Problems

In this subsection, we verify the performance of the proposed algorithm on a realworld problem, multi-target localization problems in wireless sensor networks. Consider a wireless sensor network with n sensors, whose positions are denoted by $y_1, y_2, ..., y_n$. Suppose there are N_t targets appear concurrently in the detection area. Each sensor can receive the signal strength from multiple targets. For sensor i, the measurement data is denoted by $\phi_{i,1}, \phi_{i,2}, ..., \phi_{i,N_t}$.

$$\phi_{i,t} = P_0 - 10n_p \log_{10}(\frac{\|p_t - y_i\|}{d_0}) + \epsilon \tag{37}$$

where $\epsilon \sim \mathcal{N}(0, v)$ is the measurement noise. P_0 is a known reference power value in dB milliwatts at a reference distance d_0. In this experiment, v is set to 0.1, P_0 is set to 100, d_0 is set to 1, and n_p is set to 2 [49].

A sensor can estimate the distance from targets to itself according to the signal strength. However, due to the lack of direction information, it is hard for a single sensor to locate targets. Therefore, it is necessary for multiple sensors to cooperate and locate targets. As a result, the multi-target localization problem is defined as follows:

$$F(p_1, p_2, ..., p_{N_t}) = \frac{1}{n}\sum_{i=1}^{n} f_i(p_1, p_2, ..., p_{N_t}) \tag{38}$$

$$\text{s.t.} \quad f_i(*) = \sum_{t=1}^{N_t}[\phi_{it} - (P_0 - 10n_p \log_{10}(\frac{\|p_t - y_i\|}{d_0}))]^2$$

Here, $p_1, p_2, ..., p_{N_t}$ are the optimization variables, where $p_t \in \mathbb{R}^3$ represents the estimated position of target t. $f_i(*)$ is the local objective function of sensor i, which represents the estimation error based on the measurement data of sensor i. $F(*)$ is the global objective function, which is the total estimation error of all sensors.

We test the performance of CCSA-DES and compared algorithms on this problem with different number of targets. According to the experiment result in Fig. 4, we can find that the overall performance of CCSA-DES is better than compared algorithms. Although MASOIE achieve the close performance with CCSA-DES in the case of 10 targets. As the number of targets increases, the estimation error of MASOIE increases rapidly. This is because the problem dimension increases with the number of targets. Differently, the performance of our method CCSA-DES is slightly affected by the increase of the number of targets, and it can still work well in the case of 50 targets.

VI. Conclusion

In this work, we propose a multi-agent evolution strategy with cooperative and cumulative step adaptation for black-box distributed optimization (CCSA-DES). We design a cumulative method for the neighboring evolution path in distributed optimization. Based on the neighboring evolution path, an inner adaptation method and an outer adaptation method are designed for the local

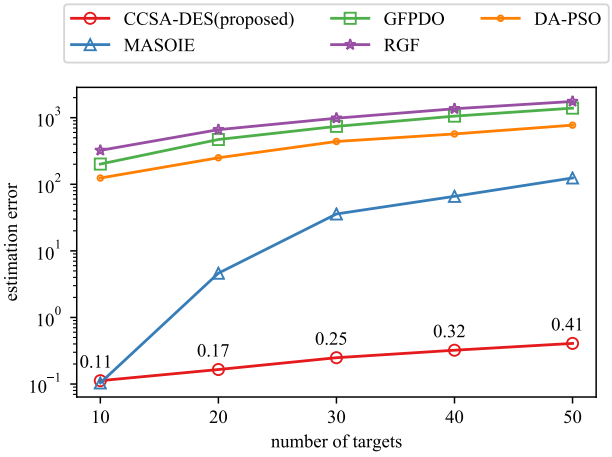


Fig. 4. Performance of CCSA-DES and compared algorithms on the multi-target localization problem with different number of targets.

optimization phase and communication phase, respectively. In terms of theoretical analysis, we first discuss the working principle of the proposed step adaptation method, and then discuss the system consensus of the proposed distributed optimization algorithm. In terms of experimental verification, CCSA-DES achieves competitive solution quality and better consensus performance compared with state-of-the-art algorithms for black-box distributed optimization.

In the future, we will study the robustness and dynamic characteristics of black-box distributed optimization problems, which exist in complex multi-agent systems, such as time-varying networks, node failures, etc. Besides, the applications of the multi-agent evolution strategy on wireless sensor networks and distributed machine learning systems are also worth studying.

References

[1] 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 TRANSACTIONS ON POWER SYSTEMS, vol. 37, no. 5, p. 13, 2022.

[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 Transactions on Smart Grid, vol. PP, pp. 1–1, Jul. 2017.

[3] Y. Wang, S. Wang, and L. Wu, "Distributed optimization approaches for emerging power systems operation: A review," Electric Power Systems Research, vol. 144, pp. 127–135, Mar. 2017.

[4] O. Van Cutsem, D. Ho Dac, P. Boudou, and M. Kayal, "Cooperative energy management of a community of smart-buildings: A Blockchain approach," International Journal of Electrical Power & Energy Systems, vol. 117, p. 105643, May 2020.

[5] T. Yang, X. Yi, J. Wu, Y. Yuan, D. Wu, Z. Meng, Y. Hong, H. Wang, Z. Lin, and K. H. Johansson, "A survey of distributed optimization," Annual Reviews in Control, vol. 47, pp. 278–305, 2019.

[6] A. Nedic and A. Ozdaglar, "Distributed Subgradient Methods for Multi-Agent Optimization," IEEE Transactions on Automatic Control, vol. 54, no. 1, pp. 48–61, Jan. 2009.

[7] A. Nedić, A. Olshevsky, W. Shi, and C. A. Uribe, "Geometrically convergent distributed optimization with uncoordinated step-sizes," in 2017 American Control Conference (ACC), 2017, pp. 3950–3955.

[8] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," IEEE Transactions on Automatic Control, vol. 49, no. 9, pp. 1520–1533, Sep. 2004.

[9] A. Nedic, A. Olshevsky, and W. Shi, "Achieving Geometric Convergence for Distributed Optimization Over Time-Varying Graphs," SIAM Journal on Optimization, vol. 27, Jul. 2016.

[10] H. Li, Q. Lu, and T. Huang, "Convergence Analysis of a Distributed Optimization Algorithm with a General Unbalanced Directed Communication Network," IEEE Transactions on Network Science and Engineering, vol. 6, no. 3, pp. 237–248, Jul. 2019.

[11] B. Gharesifard and J. Cortés, "Distributed Continuous-Time Convex Optimization on Weight-Balanced Digraphs," IEEE Transactions on Automatic Control, vol. 59, no. 3, pp. 781–786, Mar. 2014.

[12] S. A. Alghunaim and A. H. Sayed, "Distributed Coupled Multiagent Stochastic Optimization," IEEE Transactions on Automatic Control, vol. 65, no. 1, pp. 175–190, Jan. 2020.

[13] A. S. Berahas, R. Bollapragada, N. S. Keskar, and E. Wei, "Balancing Communication and Computation in Distributed Optimization," IEEE Transactions on Automatic Control, vol. 64, no. 8, pp. 3141–3155, Aug. 2019.

[14] Y. Li, J. Fan, Y. Wang, and K.-L. Tan, "Influence maximization on social graphs: A survey," IEEE Transactions on Knowledge and Data Engineering, vol. 30, no. 10, pp. 1852–1872, 2018.

[15] X.-C. Liao, W.-N. Chen, Y.-H. Jia, and W.-J. Qiu, "Towards scalable dynamic traffic assignment with streaming agents: A decentralized control approach using genetic programming," IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 8, no. 1, pp. 942–955, 2024.

[16] M.-H. Tayarani-N, X. Yao, and H. Xu, "Meta-heuristic algorithms in car engine design: A literature survey," IEEE Transactions on Evolutionary Computation, vol. 19, no. 5, pp. 609–629, 2014.

[17] F.-F. Wei, W.-N. Chen, Q. Li, S.-W. Jeon, and J. Zhang, "Distributed and expensive evolutionary constrained optimization with on-demand evaluation," IEEE Transactions on Evolutionary Computation, 2022.

[18] H. Zhu and Y. Jin, "Real-time federated evolutionary neural architecture search," IEEE Transactions on Evolutionary Computation, vol. 26, no. 2, pp. 364–378, 2021.

[19] X.-Q. Guo, F.-F. Wei, J. Zhang, and W.-N. Chen, "A classifier-ensemble-based surrogate-assisted evolutionary algorithm for distributed data-driven optimization," IEEE Transactions on Evolutionary Computation, 2024.

[20] A. Kargarian, J. Mohammadi, J. Guo, S. Chakrabarti, M. Barati, G. Hug, S. Kar, and R. Baldick, "Toward distributed/decentralized dc optimal power flow implementation in future electric power systems," IEEE Transactions on Smart Grid, vol. 9, no. 4, pp. 2574–2594, 2016.

[21] X. He, Z. Zheng, C. Chen, Y. Zhou, C. Luo, and Q. Lin, "Distributed Evolution Strategies for Black-Box Stochastic Optimization," IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 12, pp. 3718–3731, 2022.

[22] T.-Y. Chen, W.-N. Chen, X.-Q. Guo, Y.-J. Gong, and J. Zhang, "A multiagent co-evolutionary algorithm with penalty-based objective for network-based distributed optimization," IEEE Transactions on Systems, Man, and

Cybernetics: Systems, vol. 54, no. 7, pp. 4358–4370, 2024.
[23] D. Yuan and D. W. C. Ho, "Randomized gradient-free method for multiagent optimization over time-varying networks," IEEE Transactions on Neural Networks and Learning Systems, vol. 26, no. 6, pp. 1342–1347, Jun. 2015.
[24] Y. Pang and G. Hu, "Randomized Gradient-Free Distributed Optimization Methods for a Multiagent System with Unknown Cost Function," IEEE Transactions on Automatic Control, vol. 65, no. 1, pp. 333–340, 2020.
[25] D. Wang, J. Yin, and W. Wang, "Distributed Randomized Gradient-Free Optimization Protocol of Multiagent Systems over Weight-Unbalanced Digraphs," IEEE Transactions on Cybernetics, vol. 51, no. 1, pp. 473–482, 2021.
[26] D. Yuan, D. W. Ho, and S. Xu, "Zeroth-Order Method for Distributed Optimization with Approximate Projections," IEEE Transactions on Neural Networks and Learning Systems, vol. 27, no. 2, pp. 284–294, 2016.
[27] A. Das and F. Lewis, "Cooperative adaptive control for synchronization of second-order systems," International Journal of Robust and Nonlinear Control, vol. 18, no. March 2014, pp. 557–569, 2010.
[28] J. Li, C. Wu, Z. Wu, and Q. Long, "Gradient-free method for nonsmooth distributed optimization," Journal of Global Optimization, vol. 61, no. 2, pp. 325–340, 2015.
[29] Y. Tang, J. Zhang, and N. Li, "Distributed Zero-Order Algorithms for Nonconvex Multiagent Optimization," IEEE Transactions on Control of Network Systems, vol. 8, no. 1, pp. 269–281, Mar. 2021.
[30] M. Ayar, S. Obuz, R. D. Trevizan, A. S. Bretas, and H. A. Latchman, "A distributed control approach for enhancing smart grid transient stability and resilience," IEEE Transactions on Smart Grid, vol. 8, no. 6, pp. 3035–3044, 2017.
[31] D. K. Mishra, M. J. Ghadi, A. Azizivahed, L. Li, and J. Zhang, "A review on resilience studies in active distribution systems," Renewable and Sustainable Energy Reviews, vol. 135, p. 110201, 2021.
[32] Z. Li, X. Lin, Q. Zhang, and H. Liu, "Evolution strategies for continuous optimization: A survey of the state-of-the-art," Swarm and Evolutionary Computation, vol. 56, no. July 2019, 2020.
[33] Y. Jin, M. Olhofer, and B. Sendhoff, "A framework for evolutionary optimization with approximate fitness functions," IEEE Transactions on evolutionary computation, vol. 6, no. 5, pp. 481–494, 2002.
[34] I. Loshchilov, T. Glasmachers, and H.-G. Beyer, "Large scale black-box optimization by limited-memory matrix adaptation," IEEE Transactions on Evolutionary Computation, vol. 23, no. 2, pp. 353–358, 2018.
[35] N. Hansen, "The CMA Evolution Strategy: A Tutorial," pp. 1–39, 2016. [Online]. Available: http://arxiv.org/abs/1604.00772
[36] H.-G. Beyer and B. Sendhoff, "Simplify your covariance matrix adaptation evolution strategy," IEEE Transactions on Evolutionary Computation, vol. 21, no. 5, pp. 746–759, 2017.
[37] C. Igel, T. Suttorp, and N. Hansen, "A computational efficient covariance matrix update and a (1+ 1)-cma for evolution strategies," in Proceedings of the 8th annual conference on Genetic and evolutionary computation, 2006, pp. 453–460.
[38] O. Ait Elhara, A. Auger, and N. Hansen, "A median success rule for non-elitist evolution strategies: Study of feasibility," in Proceedings of the 15th annual conference on Genetic and evolutionary computation, 2013, pp. 415–422.
[39] N. Hansen, A. Atamna, and A. Auger, "How to assess step-size adaptation mechanisms in randomised search," in Parallel Problem Solving from Nature–PPSN XIII: 13th International Conference, Ljubljana, Slovenia, September 13-17, 2014. Proceedings 13. Springer, 2014, pp. 60–69.
[40] S. Meyer-Nieberg and H.-G. Beyer, "Self-adaptation in evolutionary algorithms," in Parameter setting in evolutionary algorithms. Springer, 2007, pp. 47–75.
[41] H.-G. Beyer, M. Dobler, C. Hämmerle, and P. Masser, "On strategy parameter control by meta-es," in Proceedings of the 11th Annual conference on Genetic and evolutionary computation, 2009, pp. 499–506.
[42] M. Hellwig and H.-G. Beyer, "Analysis of a meta-es on a conically constrained problem," in Proceedings of the Genetic and Evolutionary Computation Conference, 2019, pp. 673–681.
[43] D. Wilson, K. Veeramachaneni, and U.-M. O'Reilly, "Cloud scale distributed evolutionary strategies for high dimensional problems," in European Conference on the Applications of Evolutionary Computation. Springer, 2013, pp. 519–528.
[44] Q. Duan, G. Zhou, C. Shao, Y. Yang, and Y. Shi, "Distributed evolution strategies for large-scale optimization," in Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2022, pp. 395–398.
[45] T.-Y. Chen, W.-N. Chen, F.-F. Wei, X.-M. Hu, and J. Zhang, "Multi-agent swarm optimization with adaptive internal and external learning for complex consensus-based distributed optimization," IEEE Transactions on Evolutionary Computation, pp. 1–1, 2024.
[46] E. Warchulski and J. Arabas, "A New Step-Size Adaptation Rule for CMA-ES Based on the Population Midpoint Fitness," 2021 IEEE Congress on Evolutionary Computation, CEC 2021 - Proceedings, pp. 825–831, 2021.
[47] W. Ai, W. Chen, and J. Xie, "A general framework for population-based distributed optimization over networks," Information Sciences, vol. 418–419, pp. 136–152, Dec. 2017.
[48] 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 2015 International Symposium on Signals, Circuits and Systems (ISSCS), Jul. 2015, pp. 1–4.
[49] G. Mao, B. Fidan, and B. D. Anderson, "Wireless sensor network localization techniques," Computer Networks, vol. 51, no. 10, pp. 2529–2553, 2007.