Process Knowledge-guided Autonomous Evolutionary Optimization for Constrained Multi-objective Problems

Mingcheng Zuo, Dunwei Gong, *Member, IEEE*, Yan Wang, Xianming Ye, Bo Zeng and Fanlin Meng

***Abstract*—Various real-world problems can be attributed to constrained multi-objective optimization problems. Although there are various solution methods, it is still very challenging to automatically select efficient solving strategies for constrained multi-objective optimization problems. Given this, a process knowledge-guided constrained multi-objective autonomous evolutionary optimization method is proposed. Firstly, the effects of different solving strategies on population states are evaluated in the early evolutionary stage. Then, the mapping model of population states and solving strategies is established. Finally, the model recommends subsequent solving strategies based on the current population state. This method can be embedded into existing evolutionary algorithms, which can improve their performances to different degrees. The proposed method is applied to 41 benchmarks and 30 dispatch optimization problems of the integrated coal mine energy system. Experimental results verify the effectiveness and superiority of the proposed method in solving constrained multi-objective optimization problems.**

***Index Terms*—Constrained multi-objective optimization, evolutionary optimization, autonomy, process knowledge, integrated coal mine energy system**

I. Introduction

CONSTRAINED multi-objective optimization problems (CMOPs) refer to optimizing multiple conflicting objectives, with the decision variables satisfying one or more equality/inequality constraints. Many real-world problems can be attributed to CMOPs, such as integrated energy dispatch[1], multi-stage portfolio[2] and spacecraft orbit optimization[3]. Without loss of generality, a CMOP can be defined as

$$\begin{aligned} \text{Min } & F(x) = (f_1(x), f_2(x), \dots, f_m(x)) \\ \text{s.t. } & g_i(x) \le 0, i = 1,2,\dots,l \\ & h_i(x) = 0, i = l+1, l+2, \dots, k \\ & x = (x_1, x_2, \dots, x_D) \in R^D \end{aligned}$$

Since decision variables need to satisfy one or more constraints and multiple conflicting objectives are optimized simultaneously, multi-objective optimization problems are very challenging. In recent years, multi-objective evolutionary optimization has been the mainstream method to solve this problem. The key to solving CMOPs with this method is dealing with constraints to balance the feasibility, convergence, and diversity of optimization solutions in the objective space. Currently, standard constraint handling techniques in multi-objective evolutionary optimization methods include the penalty function method, objective and constraint separation method, multi-objective method, transformation method, mixed method, and multiple-operator method[4]. Among them, the multiple-operator method solves CMOPs by dynamically adjusting the distribution of a population in the decision space[5]. In general, the existing multiple-operator method usually determines the opportunity and scope of utilizing the alternative operators according to human experience. However, the limitations of human experience cause the opportunity and scope of utilizing the various operators to be subjective. In addition, the adopted operators are not adjusted timely according to the evolutionary state of a population, resulting in low efficiency when solving CMOPs.

Enhancing the multiple-operator method is to perform the most promising operator to generate a population. Dynamically selecting an efficient operator among all candidates needs continuously learning the knowledge [1] generated in the evolutionary process. Currently, various knowledge-guided evolutionary algorithms have been proposed to solve multi-

This work was sponsored by the National Key R&D Program of China with grant No. 2021YFE0199000, the National Natural Science Foundation of China with grant No. 62133015, Shandong Provincial Natural Science Foundation with grant No. ZR2022LZH017, Fundamental Research Funds for the Central Universities with grant No. JAI210003 and the Open Research Project of The Hubei Key Laboratory of Intelligent Geo-Information Processing with grant No. KLIGIP-2022-A06. *(Corresponding author: Dunwei Gong).*

Mingcheng Zuo is with the Artificial Intelligence Research Institute and School of Mathematics, China University of Mining and Technology, Xuzhou 221116, PR China (e-mail: mingcheng.zuo@cumt.edu.cn).

Mingcheng Zuo is also with the State Key Laboratory of High-end Server & Storage Technology; Hubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences, Wuhan 430078, PR China.

Dunwei Gong is with School of Information Science and Technology, Qingdao University of Science and Technology, Qingdao, Shandong, 266061, PR China (e-mail: dwgong@vip.163.com)

Yan Wang is with School of Information and Control Engineering, China University of Mining and Technology, Xuzhou, 221116, PR China (e-mail: yanwang0501@outlook.com).

Xianming Ye is with the Department of Electrical, Electronic, and Computer Engineering, University of Pretoria, Pretoria 0002, South Africa (e-mail: xianming.ye@up.ac.za).

Bo Zeng is with the State Key Laboratory of Alternate Electrical Power System with Renewable Energy Sources, North China Electric Power University, Beijing 102206, PR China (e-mail: alosecity@126.com).

Fanlin Meng is with Alliance Manchester Business School, University of Manchester, Manchester M15 6PB, UK (e-mail: fanlin.meng@manchester.ac.uk).

[1] The knowledge here refers to the useful information generated during the execution of evolutionary algorithm which can help the algorithm to make the search decision, rather than the existing human experience.

objective optimization problems. These algorithm variants include meta-knowledge-guided evolutionary algorithms[6], local search-oriented knowledge-guided evolutionary algorithms[7], multi-stage knowledge-guided evolutionary algorithms[8], reference vector-guided evolutionary algorithms[9,10], good and bad knowledge-guided evolutionary algorithms[11], individual ranking-guided evolutionary algorithms[12-15], etc. However, applying the process knowledge to multiple-operator methods for better-solving CMOPs still needs to be improved.

Given the above analysis and to fill the research gap, a process knowledge-guided autonomous evolutionary optimization method (PKAEO) for constrained multi-objective problems is proposed in this paper. Besides the operators, other solving strategies also can be embedded in PKAEO as guidance strategies for population evolution. The process knowledge indicates how to perform the guidance strategies for generating a better evolutionary population. Considering the strong ability of deep reinforcement learning to sense the effectiveness of the strategy, the representative model of deep reinforcement learning, Deep Q-learning Network (DQN), is used to reflect the process knowledge. Firstly, in the early evolutionary stage of PKAEO, the guidance strategies are randomly executed to evaluate their effects on different populations' evolutionary states where a certain number of samples are accumulated simultaneously. Then, based on these samples, the DQN is trained to establish the mapping model between population states and guidance strategies. Finally, the model recommends subsequent guidance strategies intelligently based on the current population state.

The main contributions of this paper are reflected in the following three aspects:

(1) An autonomous design pattern of guidance strategies selection in evolutionary optimization algorithms for CMOPs is proposed. Compared with the human experience-based design pattern, the proposed PKAEO needs designers to set population state, guidance strategies, and strategy evaluation method. Then, the guidance strategies can be automatically selected according to the population state to maximize the benefit of strategy execution. Theoretically, this method can provide superior strategy selection planning, and in the application, it can significantly save the time and cost of algorithm design.

(2) A simple but effective method of describing the population's evolutionary process is proposed. Currently, many evolutionary algorithms have configured dynamically adjusted strategies or parameters related to the evolutionary process. However, the relationship is usually built on the subjective manual experience. The proposed population state effectively recognizes the real-time evolutionary situation and is conducive to establishing the dynamic adjustment of strategies or parameters.

(3) A flexible method of embedding the guidance strategies into existing evolutionary algorithms is proposed. In general, the current multiple-operator method is fixed to a specific paradigm, which causes difficulty in synthesizing the capacity of multiple solving paradigms. Relatively, the guidance strategies in PKAEO can be designed by integrating various solving paradigms. More importantly, these guidance strategies are only placed between the operators without modifying the original structure of evolutionary algorithms.

The rest of this paper is organized as follows. Section II reviews the relevant research works and points out the problems existing in the current research. The proposed method is described in Section III, including the population state's characterization, the population's regulation strategies, and their performance evaluation. Section IV is the experimental results and analysis. Finally, Section V summarizes the whole paper and discusses future research directions.

II. Related Works

In this paper, we study evolutionary algorithms for constrained multi-objective optimization problems and enhance the autonomy of guidance strategy selection through process knowledge. Given this, this section mainly reviews the existing constrained multi-objective evolutionary algorithms. Considering that the proposed method involves multiple-operator methods and knowledge-guided techniques, the existing research on these two topics is also reviewed.

A. Constrained multi-objective evolutionary optimization

Constraint handling for constrained multi-objective evolutionary optimization is crucial. The penalty function method is the most commonly used in many constraint-handling methods. The idea is transforming constraints into objective functions through the penalty function so that a constrained optimization problem becomes unconstrained. It is well understood that the penalty factor dramatically affects the performance of the penalty function method, such as the adverse effects caused by too large or too small values. If the feasible region consists of several disconnected ones, then the population will be limited to a particular local search space region by a hefty penalty factor. If the penalty factor is small, many searches will be spent on infeasible regions, causing difficulty in finding feasible solutions to the problem. Given this, the penalty functions with various penalty factors have been proposed in previous works. Jan et al.[16] adjusted the value of the penalty factor according to the proportion of feasible solutions in the evolutionary population. In the dynamic penalty factor method proposed by Zapotecas et al.[17], the value of the factor is closely related to the evolutionary stage of the population. In addition, Vaz et al.[18] also proposed a three-stage penalty factor adjustment method dependent on the evolutionary stage. It can be seen that the idea of the penalty function method is simple. However, it needs to set a reasonable penalty factor, and the setting of this factor often depends on the human experience.

Contrary to the penalty function method, the objective and constraint separation method calculates the objective function values of a candidate and its degree of violating constraints, respectively, to minimize the objective function values while alleviating the degree of violating constraints through a population's evolution. Representative methods include the constraint dominance method[19], ε constraint method[20] and random ranking method[21]. In the constraint dominance method, for solutions x and y, when one of the following conditions is satisfied, x dominates y: (1)$CV(x) < CV(y)$; (2)If $CV(x) = CV(y) = 0, \forall i \in \{1,2,\dots,m\}$, there is $f_i(x) \leq f_i(y)$, and $\exists i \in$

$\{1,2,\dots,m\}$, $f_i(x) < f_i(y)$, where $CV(\cdot)$ represents the constraint violation degree of the optimization solution. Deb et al.[19] first adopted this constraint handling method in NSGA-II. Although this method can find a feasible solution quickly, it can also causes prematurity[22]. To make the population evolves toward the feasible regions, the ε constraint method deals with the constraints of optimization solutions by relaxing the constraints to a certain extent and gradually decreasing the value of ε. When $\varepsilon = 0$, the ε constraint method is equivalent to the constraint dominance method. Jiao et al.[23] used the ε constraint method to continuously reduce the constraint violation degree of infeasible solutions and obtained the optimization solution set that converged to the constraint Pareto front. Considering that the constraint dominance method may lead to the premature convergence of a population, therefore, in the random ranking method, constraint dominance, and objective dominance are selected probabilistically to generate the offspring population. Ying et al.[24] proposed an adaptive random ranking method according to the constraint violation difference of population individuals and evolutionary stages. Obviously, the objective and constraint separation method can well balance the feasibility of a candidate and its degree of optimizing objectives. However, this method involves the setting of parameters (such as ε and the probability), which still depends on human's experience.

Like the penalty function method, the multi-objective method also transforms the constrained multi-objective optimization problem into an unconstrained multi-objective optimization problem. However, the method considers the constraints as one or more optimization objectives and optimizes the transformed objectives to reduce the constraint violation degree of the optimization solution. The number of optimized objectives after transformation differs by considering various transformation methods. Ray et al.[25], Long et al.[26] and Zhou et al.[27] transformed all constraints into one optimization objective related to the degree of constraint violation. Vieira et al.[28] transformed the constraints into two optimization objectives: the total constraint violation degree and the number of violated constraints. Obviously, this method can deal with constraints flexibly, and the number of transformed objectives can be determined according to the solving needs. However, the increase in the number of objective functions also causes difficulty in solving the problems.

The difficulty of solving a complex optimization problem can be reduced by decomposing it into several simple problems. Based on this, the transformation method divides an evolutionary algorithm into a number of components, with each being responsible for a subtask. The whole optimization problem is solved through the cooperation of these components. A typical transformation method is dividing the population into multiple subpopulations and then solving multiple subtasks through the evolution of the subpopulations. The optimization problem can be entirely solved based on the co-evolution of subpopulations. Generally, the tasks accomplished through the evolution of different subpopulations can be set flexibly. Wang et al.[29] proposed a co-evolutionary optimization method, which divided the whole population into subpopulations with a number equaling that of optimization objectives. In this method, the constrained single-objective optimization problems are solved through the evolution of each subpopulation. Liu et al.[30] divided the whole population into two subpopulations. One subpopulation searched for the optimal solutions by minimizing the optimization objectives, and the other searched for feasible solutions satisfying the constraints. Similarly, Tian et al.[31] divided the whole population into two subpopulations, which were devoted to searching for solutions on the unconstrained Pareto Front (UPF) and the constrained Pareto front (CPF), respectively. Another standard transformation method is splitting the entire evolution process into multiple stages, each of which performs different optimization subtasks. Under this circumstance, the tasks to be fulfilled can be flexibly set through a population's evolution in different stages. Fan et al.[32] proposed a search framework combining the phases of push and pull. In the push phase, the population passes through the infeasible regions and converges to the UPF. In the pull phase, the population converges to the CPF from the UPF. Tian et al.[33] also divided the whole evolution process into two stages. In the first stage, the population was guided to evolve toward the feasible regions, and in the second stage, the optimization solutions were better distributed on the CPF. Yu et al.[34] divided the whole evolution process into two stages as well. The first stage considered the balance between diversity and convergence of optimization solutions, and the second stage considered the balance between diversity and feasibility of optimization solutions. It can be seen that the transformation method reduces the difficulty of problem-solving by decomposing the optimization task into some subtasks that can be easily solved. However, how to decompose the problem reasonably depends on the human experience, and the quality of generated solutions in decomposition based methods depends strongly on the weights' setting[35].

The hybrid of the evolutionary algorithm and traditional optimization method also can improve the efficiency of solving problems. Specifically, the evolutionary algorithm guides the population to evolve toward the regions with better objective values, and the mathematical programming further searches for feasible solutions in the current region. Morovati et al.[36] combined evolutionary algorithm and Zoutendijk feasible direction method to find optimization solutions satisfying constraints. By mining the valuable information of the population in the evolutionary process, Schutze et al.[37] estimated the exploration directions for evolutionary algorithms, and predicted the exploitation direction for locating feasible regions. It can be seen that the hybrid method takes into account both global and local searches and balances the optimization performance and the degree of constraint satisfaction. However, the opportunity of performing mathematical programming during the search of evolutionary algorithms is a problem.

B. Multiple-operator method

The operators directly affect the population's distribution in the search space. The basic idea of the multiple-operator method is adaptively implementing various operators on a population according to the specific needs of problem-solving,

achieving the balance of feasibility, convergence, and diversity of optimization solutions. The customized multiple-operator methods for different optimization problems show diversiform characteristics. In the multiple-operator method proposed by Yu et al.[38], an advanced mutation operator is designed for infeasible solutions. Each mutation operator is performed on individuals with different probabilities during the evolutionary process, leading the population to evolve toward the CPF. Yu et al.[39] attempted to control the opportunity of an individual to generate offspring, where a feasible solution conducts genetic operations with a high probability. In the method proposed by Xu et al.[40], feasible solutions and infeasible solutions adopt different mutation strategies to make the population evolve toward the feasible regions as soon as possible. Liu et al.[41] divided the whole population into multiple subpopulations, where each subpopulation adopted different crossover strategies. In this way, the population achieved better global exploration ability. He et al.[42] carried out different crossover and mutation operations on feasible and sound infeasible solutions. It realized the transformation from infeasible solutions to feasible solutions. Qian et al.[43] adaptively adjusted the parameter of the mutation operator and generated the trial vectors based on the acquired knowledge in the evolution process to balance the diversity and convergence of the population. Tian et al.[44] used deep reinforcement learning to intelligently select operators at each evolutionary stage to achieve the balance of population diversity and convergence.

It can be seen that the multiple-operator method can adapt to the needs of problem-solving and balance the feasibility, convergence, and diversity of optimization solutions. However, how to configure multiple operators with complementary performance and determine the appropriate opportunity to perform the operators still need further research.

C. Knowledge-guided multi-objective evolutionary optimization

Knowledge guidance evolutionary algorithms are efficient in dealing with complex optimization problems. The knowledge is the extracted information from the data generated in the evolutionary process, which can further assist in generating a better offspring population. Ding et al.[8] used the prior knowledge to initialize the population and led the population's evolution by referring to the elite individuals. Yao et al.[45] proposed a subspace-related population initialization method and adaptively generated new promising individuals by evaluating the quality of solutions in each subspace. Guo et al.[46] proposed a knowledge-guided transfer strategy for evolutionary dynamic multi-objective optimization problems. This method extracts knowledge as a two-tuple under each historical environment, which is preserved in a knowledge pool. Redundant knowledge is recognized and adaptively removed to guarantee the pool's diversity. To promote positive knowledge transfer, a knowledge-matching strategy is developed to re-evaluate the representative of each stored knowledge under a new environment. In addition, an improved knowledge transfer mechanism based on subspace alignment is introduced. To circumvent the rapid loss of population diversity and premature convergence, Li et al.[47] proposed a knowledge-guided multi-objective particle swarm optimization using fusion learning strategies. An improved leadership selection strategy based on knowledge utilization is presented to select the appropriate global leader for improving the convergence ability of the algorithm. However, there is still no knowledge-guided algorithm for constrained multi-objective optimization problems.

C. Discussion

From the review of existing studies, it can be seen that the transformation method and the multiple-operator method both need to describe the state of the population's evolution, set the switching conditions of different strategies/stages, and evaluate the effectiveness of the strategies. These operations substantially impact the performance of constrained multi-objective evolutionary optimization algorithms. For the employed techniques in this paper, i.e., the multiple-operator method and knowledge-guided theory, there are still following bottleneck:

(1) Most existing multiple-operator methods usually determine the opportunity and scope of utilizing the various operators based on the human experience, which can be very subjective. In addition, the adopted operators are not adjusted in time according to the evolutionary state of the population, which dramatically limits the quality and efficiency of solutions. The only algorithms that can automatically select the operators, like ref.[44], develop the framework based on a specific solving paradigm, which are difficult to embed into existing algorithms to improve their performances.

(2) The current knowledge-guided multi-objective evolutionary optimization algorithms usually employ customized knowledge transfer methods, like prior knowledge, for specific problems. However, as the optimization problem becomes more large-scale, more dynamic, and more complex, the role of knowledge guidance becomes more demanding. It is imperative to propose a general knowledge guidance framework available to the constrained multi-objective optimization algorithms for different application scenarios.

To overcome the above limitations, we propose a process knowledge-guided constrained multi-objective autonomous evolutionary optimization algorithm, which is detailed in Section III. Based on the population state, the proposed algorithm can automatically recommend the strategy for the subsequent population's evolution by using the mapping model of the population state and guidance strategy, which significantly improves the autonomy of problem-solving. The proposed framework is flexible, where the population state, guidance strategies, and strategy evaluation can be easily adjusted according to the actual demands. Specifically, by setting various features in the population state, assorted information can be mined from diversified aspects; to realize the disparate guidance effects, distinctive strategies can be configured in the framework; Divergent evaluation indicators can be set to lead the algorithm to complete different solving tasks in multiple evolutionary stages.

This article has been accepted for publication in IEEE Transactions on Evolutionary Computation. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2023.3243109

III. Process Knowledge-guided Constrained Multi-objective Autonomous Evolutionary Optimization

A. Overall framework

This section proposes a constrained multi-objective autonomous evolutionary optimization method guided by process knowledge. The idea is as follows. Firstly, the regulation effects of different guidance strategies on the population states are evaluated at the early evolutionary stage. Then, based on the generated samples, the mapping model between population states and guidance strategies is established. Finally, the subsequent guidance strategy is intelligently recommended according to the established mapping model and the current population state. The advantage of this approach is that the constrained multi-objective optimization problem can be solved efficiently by adopting appropriate strategies at different stages of the population's evolution.

The overall framework of the proposed method is shown in **Algorithm 1**. In this method, process knowledge reflects the influence of guidance strategies on population state regulation, including the state s_t of the population P_t in the generation t, the adopted guidance strategy a_t and its effect evaluation r_t. Based on process knowledge, the mapping relationship between (s_t, a_t) and r_t is established and represented by Deep Q-learning Network (DQN). According to s_t, the DQN is applied to determine the guidance strategy a_t to be implemented on P_t (line 5). In addition, the evolutionary population information of different generations is used to enrich process knowledge (lines 6-9). Here, T_{max} is the maximum evolutionary generation. It is easy to see that PKAEO does not change the structure of existing evolutionary algorithms but only uses process knowledge to guide the population's evolution.

Algorithm 1 The framework of PKAEO

Input: parameters of DQN, maximum generation number T_{max}

1. $t \leftarrow 1$;
2. $P_t \leftarrow InitializePopulation()$;
3. **While** *the termination condition is not met* ***do***
4. /*Embedding PKG to EAs, referring to Algorithm 2*/
5. $V_t \leftarrow Recombination(P_t)$; $\boldsymbol{U_t \leftarrow PKG(V_t)}$; $Q_t \leftarrow Mutation(U_t)$;
6. $R_t \leftarrow Q_t \cup P_t$;
7. Generate offspring population P_{t+1} from R_t according to the evolutionary algorithm;
8. /*Archive the process data, referring to Algorithm 3*/
9. **Archive the process data into memory *M*;**
10. $t \leftarrow t + 1$;
11. **End while**
12. **Return** $P_{T_{max}}$;

In order to apply process knowledge to guide the subsequent population's evolution, the whole process of the population's evolution is divided into two stages, as shown in **Fig.1**. The first stage is the process knowledge acquisition stage, starting from the first generation to the generation T_{sam}. The second stage is the process knowledge application stage from the generation $T_{sam} + 1$ to the end of population's evolution.

More details of **Fig.1** are explained in **Algorithm 2**. In the process knowledge acquisition stage, the samples for training DQN (line 6) are generated by randomly implementing the guidance strategy a_t on population P_t(line 3), which includes the state s_{t+1} of P_{t+1}, and the evaluation value r_t of the strategy a_t. In the process knowledge application stage, according to the state s_t, the predicted evaluation value r_t' of the guidance strategy a_t is provided by DQN. Then the subsequent population's evolution is carried out by performing the a_t with maximum predicted r_t' (line 9). At the same time, the sample used to update DQN is also generated (line 8).

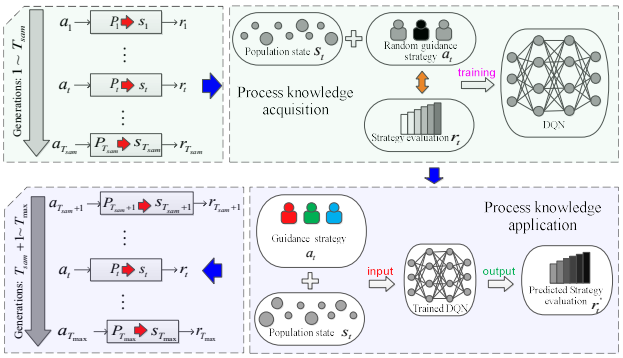


Fig.1 Process knowledge acquisition and application. s_t is the extracted population state from P_t, a_t is the guidance strategy performed on P_t, r_t is the performance evaluation of a_t.

Algorithm 2 Population knowledge guidance (PKG)

Input: recombined population V_t;

1. **If** $t \leq T_{sam}$ then
2. /* Select a random strategy */
3. Select a_t randomly;
4. **Else**
5. /*Train the DQN */
6. Train the DQN with **algorithm 4**;
7. /*Update the DQN */
8. Update the DQN with **algorithm 5**;
9. Select a_t according to equation (3);
10. **End if**
11. Conduct a_t on V_t to generate U_t;

It can be seen that the key to realizing process knowledge guidance is determining the input (s_t,a_t) and output r_t of DQN and training the DQN. Below, these fundamental techniques are detailed in Sections III-B and III-C, respectively.

B. Inputs and outputs of DQN

1) *Population state characterization based on diversity and convergence*

Diversity and convergence are two critical characteristics of the evolutionary population, which reflect the dispersion and approximation performances of the population, respectively. Given this, population states can be characterized by diversity and convergence. It is worth noting that the diversity and convergence are from the decision space. This paper adopts the following two methods to characterize population diversity. The first is the standard deviation of individual locations in the population. For P_t, the standard deviation of individual locations in this population is denoted as ξ_t[48,49]. The second is the number of individual clusters in the population. For P_t, the

best number of individual clusters in this population is denoted as κ_t[50,51], which can be estimated by the Silhouette Coefficient method[52]. It is easy to see that ξ_t and κ_t reflect the intra-generational characteristics of the population, and describe the diversity of the population from different aspects.

The convergence characteristics of the population are described in the following two ways. One is the distance between the center points of populations in two successive generations. Considering the population of generations $t-1$ and t, the distance between the centers of P_{t-1} and P_t is λ_t [53]. The other is the average distance between centroids of individual clusters in two successive generations. For P_{t-1} and P_t, the average movement distance of clustering centroids is denoted as τ_t, then, τ_t can be presented as

$$\tau_t = \frac{1}{\kappa_t}\sum_{i=1}^{\kappa_t} |NC_{i,t} - C_{i,t}| \quad (1)$$

where $C_{i,t}$ is the i^{th} centroid of P_t. Please note that P_{t-1} is divided into κ_{t-1} clusters here, and $NC_{i,t}$ means the nearest cluster centroid in C_{t-1} to $C_{i,t}$. It can be seen that λ_t and τ_t reflect the population's inter-generational characteristics and depict the population's convergence from different angles.

For constrained optimization problems, in addition to using the above methods to characterize the whole population, it is also necessary to consider the characterization of feasible solutions in the population. For the feasible solution sets of the population in the generation t, the above characteristics are denoted as λ_t^f, τ_t^f, ξ_t^f and κ_t^f. In particular, when all the individuals in P_t are infeasible solutions, let $\lambda_t^f = \tau_t^f = \xi_t^f = \kappa_t^f = 0$; when all the individuals in P_t are feasible solutions, $\lambda_t = \lambda_t^f, \tau_t = \tau_t^f, \xi_t = \xi_t^f, \kappa_t = \kappa_t^f$.

In addition, the evolution process of the population, denoted as φ_t, is characterized as follows:

$$\varphi_t = \frac{t}{T_{max}}$$

In this way, the population P_t can be characterized by the above nine features to form the population state, denoted as s_t. Therefore, s_t can be represented as

$$s_t = (\lambda_t, \tau_t, \xi_t, \kappa_t; \lambda_t^f, \tau_t^f, \xi_t^f, \kappa_t^f; \varphi_t) \quad (2)$$

2) *Guidance strategies for population state regulation*

In evolutionary algorithms, the regulation of the population state is achieved through various guidance strategies[54]. It is easy to understand that disparate evolutionary algorithm paradigms usually adopt distinctive guidance strategies in preference. Among many paradigms of evolutionary algorithms, the differential evolution algorithm is very typical and widely used. In addition to the original guidance strategy, numerous enhanced guidance strategies have been proposed for the differential evolution algorithms, like the DE/current-to-pbest/1-X[55]. Existing differential guidance strategies can be divided into the following three categories: i) DE/best/1 and DE/best/2; ii) DE/rand/1 and DE/rand/2; and iii) DE/current-to-pbest/1. For DE/best/1 and DE/best/2, they have better convergence performance, among which DE/best/1 has a faster convergence speed and DE/best/2 has better local convergence performance. For DE/rand/1 and DE/rand/2, they can well maintain population diversity[56]. Compared with DE/rand/1, DE/rand/2 can generate diversified evolutionary directions in the search space and has a more vital ability to maintain population diversity. For DE/current-to-pbest/1, they take into account the convergence and diversity of the population[57]. In addition, they can effectively protect individual genes that satisfy all or part of the constraints.

For other paradigms of evolutionary algorithms, there is a variety of guidance strategies with an extraordinary performance that also can be used but these are beyond the scope of this paper.

In summary, there are five guidance strategies for regulating population state here. If the number of guidance strategies performed on the population P_t is denoted as a_t, then, the value of a_t and its meaning are as follows: 0 indicates that no guidance strategy is implemented; 1 to 5 indicates the implementation of guidance strategies DE/best/1, DE/best/2, DE/rand/1, DE/rand/2, and DE/current-to-pbest/1, respectively.

3) *Phased evaluation of guidance strategies*

When using evolutionary algorithms to solve constrained multi-objective optimization problems, the evolution of the population can be roughly divided into two stages. The first stage is when the evolutionary population does not contain any feasible solution to the problem; another stage is when the evolutionary population contains one or more feasible solutions to the problem. For these stages, different indicators are used to evaluate the performance of guidance strategies.

Suppose the evolutionary population does not contain any feasible solution to the problem. In this case, the performance of the guidance strategy can be evaluated by reducing the overall constraint violation degree of the population. Here, the constraint dominance method is employed. Consider the population P_t, and perform the strategy a_t on the P_t. In order to evaluate the performance of a_t, the overall constraint violation degree of P_t is calculated, denoted as ϕ_t. The constraint violation degree of an individual is obtained by summing these violation degrees. Then, the performance of the guidance strategy a_t, denoted as r_t, can be presented by

$$r_t = \frac{\phi_t - \phi_{t+1}}{\phi_t} \quad (3)$$

Obviously, the value of r_t is between 0 and 1. In particular, if $r_t = 0$, it means the constraint satisfaction does not improve after performing an guidance strategy; If $r_t = 1$, it indicates after performing a_t, all the individuals of P_t satisfy the constraints.

When the evolutionary population contains one or more feasible solutions to the problem, the performance of the guidance strategy is evaluated by the improvement degree of PD[58] and HV[59]. Here, PD is used to reflect the diversity of feasible solutions, and HV reflects not only the diversity, but also the convergence of feasible solutions. For the population P_t, the PD and HV of feasible solutions are denoted as PD_t and HV_t, respectively. Then, the performance r_t of the guidance strategy a_t can be expressed as

$$r_t = \varphi_t * \frac{PD_{t+1} - PD_t}{PD_t} + (1 - \varphi_t) * \frac{HV_{t+1} - HV_t}{HV_t} \quad (4)$$

In this way, the performance r_t of the guidance strategy a_t can be expressed as

$$r_t = \begin{cases} \frac{\phi_t - \phi_{t+1}}{\phi_t}, & if\ \ \phi_t > 0 \\ \varphi_t * \frac{PD_{t+1} - PD_t}{PD_t} + (1 - \varphi_t) * \frac{HV_{t+1} - H_{\ t}}{HV_t}, & otherwise \end{cases} \quad (5)$$

Note that due to different search preferences, this evaluation method may generate conflicts with the embedded algorithms. In specific applications, diversified evaluation

methods can be flexibly set according to the guiding need of the population's evolution, such as replacing the constraint dominance method with the ε constraint method.

C. Training and application of DQN

1) *Sample acquisition and augmentation*

When the proposed method is used to solve the constrained multi-objective optimization problems, a sample is generated in each generation of the population's evolution, and the t^{th} sample is denoted as $e_t = (s_t, a_t, r_t, s_{t+1})$. Different samples have different functions. Specifically, the collected samples in the process knowledge acquisition and application are used to train and update the mapping model DQN, respectively. The sample acquisition method is shown in **Algorithm 3**. First, initialize the memory M to hold the samples and set the size of M to N (line 2). Then, after one evolution generation of population P_t, the performance r_t of the guidance strategy a_t is evaluated, and the state s_{t+1} of the offspring population P_{t+1} is obtained and recorded (line 4). Finally, the generated sample $e_t = (s_t, a_t, r_t, s_{t+1})$ is saved in M. If the number of samples exceeds N, then the sample with the smallest generation number in M is replaced (line 5).

Algorithm 3 Acquisition of samples

1. **If** $t = 1$ then
2. Initialize replay memory M to capacity N;
3. **End if**
4. Observe r_t and the state s_{t+1} of P_{t+1};
5. Archive transition $e_t = (s_t, a_t, r_t, s_{t+1})$ into M.

Since the number of obtained samples is difficult to meet the requirements of DQN training, this paper generates the augmented samples by randomly perturbing the existing samples. For the sample $e_t = (s_t, a_t, r_t, s_{t+1})$, different augmented samples can be obtained by augmenting them once or more. It is easy to understand that increasing the augmentation produces more augmented samples, which helps to improve the accuracy of DQN, but also leads to high computational burden. Denote the times of augmenting sample e_t as α, then, the augmented samples of this sample can be expressed as

$$\begin{aligned} e_t^1 &= (s_t^1, a_t, r_t^1, s_{t+1}) \\ e_t^2 &= (s_t^2, a_t, r_t^2, s_{t+1}) \\ &\cdots \\ e_t^\alpha &= (s_t^\alpha, a_t, r_t^\alpha, s_{t+1}) \end{aligned} \tag{6}$$

where $s_t^i = s_t * norm(1, \sigma)$, $r_t^i = r_t * norm(1, \sigma)$, $norm(1, \sigma)$ is the normal distribution function, i =1, 2, …, α.

It is worth noting that the component s_{t+1} is not augmented because it does not participate in the training process. To keep the sample balance with different strategies, the augmentation of e_t is conducted only when the proportion of a_t among all the samples is lower than $\frac{1}{NS}$, where NS is the overall number of guidance strategies.

2) *Training of DQN*

The training process of DQN is shown in **Algorithm 4**. First, b samples are randomly selected and normalized from M (line 2). The training parameters are initialized, including the weight θ of DQN, the maximum training times g, the number of hidden layers and neurons, and the learning rate lr of the network. The hidden layers are connected through the Sigmoid function, and the activation function of the output layer is linear (line 3). Then, for a given sample $e_t = (s_t, a_t, r_t, s_{t+1})$, the predicted value r_t' of the sample is obtained by forward propagation, thus the prediction error is calculated, and the network weights θ are adjusted along the direction of the negative gradient of the prediction error. Finally, when the fitting precision is met, or the number g of training epochs is reached, DQN training is stopped (line 4).

Algorithm 4 Training of the DQN

1. **If** $t = T_{sam} + 1$ then
2. Sample and normalize random b transitions from M;
3. Initialize the DQN parameters;
4. Train the DQN to get $Q(s_t, a; \theta)$;
5. **End If**

3) *Application of DQN*

After training DQN, for the population P_t in the generation t, the population state s_t is taken as the input of DQN, and the performance prediction values of different guidance strategies are obtained. Denoting the effect of guidance strategy, after performing guidance strategy a_t on P_t, is obtained as r_t'. For all the guidance strategies, the guidance strategy with the maximum performance prediction value is denoted as $a_t^{max} = arg\ \max\{r_t'\}$. Therefore, a_t^{max} is selected as the guidance strategy for the subsequent population.

In addition, in order to reduce the negative impact caused by overfitting DQN and improve the exploration ability in population state space, this paper draws on Epsilon's greedy strategy. It randomly selects a strategy from all guidance strategies with a probability ϵ to replace the strategy recommended by DQN for the subsequent population's evolution. Specifically, Epsilon's greedy strategy is as follows:

$$a_t = \begin{cases} rand\_strategy & if\ rand < \epsilon \\ a_t^{max} & otherwise \end{cases} \tag{7}$$

Where $rand\_strategy$ means selecting a random strategy, function $rand$ is a random float number belonging to [0, 1]. Considering that PKAEO needs to get more random samples at the early stages, ϵ is set to 0.3^{φ_t}, which is a dynamically decreased value, as shown in **Fig.2**.

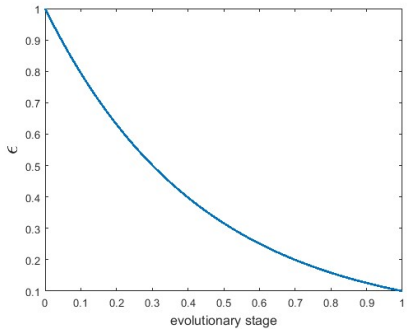


Fig.2 The value of ϵ in PKAEO

4) *Updating of DQN*

As the population continues to evolve, more samples are generated, and the prediction accuracy of DQN may also decrease. Given this, it is necessary to update DQN in the process of population evolution to predict the performance of

guidance strategies accurately. For convenience, we update DQN regularly. As shown in **Algorithm 5**, denote the updating period of DQN as T_{upd}, DQN is updated when $\mathrm{t} \geq T_{sam} \wedge (\mathrm{t} - T_{sam})\%T_{upd} = 0$. For the k^{th} update of DQN, the sample set generated from the last DQN update to the current generation, denoted as M_k, is used to update the parameters of DQN. The updating sets a goal y_j for each $e_j = (s_j, a_j, r_j, s_{j+1})$, and construct a loss function

$$L = (y_j - Q(s_j, a_j; \theta))^2 \tag{8}$$

The loss function L is the optimization objective to update the weight parameter θ of the neural network. $Q(.)$ is the predicted r_j' of a_j for s_j. If the stop condition of the evolutionary algorithm is not reached, $y_j = r_j + \gamma max_a Q(s_{j+1}, a; \theta)$, where $\gamma \in [0,1]$ is the discount on the predicted strategy evaluation. $max_a Q(s_{j+1}, a; \theta)$ is the maximum predicted r'_{j+1} of all possible a_{j+1} for s_{j+1}. The purpose of this operation is to accelerate the convergence of DQN. When the stop condition of the evolutionary algorithm is reached, $y_j = r_j$. Overall,

$$y_j = \begin{cases} r_j & if\ \ terminal\ s_{j+1} \\ r_j + \gamma max_a Q(s_{j+1}, a; \theta) & otherwise \end{cases} \tag{9}$$

Algorithm 5 Updating of the DQN

1. **If** $\mathrm{t} \geq T_{sam} \wedge (\mathrm{t} - T_{sam})\%T_{upd} = 0$
2. Sample and normalize random mini-batch of b transitions $e_j = (s_j, a_j, r_j, s_{j+1})$ from M;
3. Update $Q(s_t, a; \theta)$ according to equation (9);
4. **End If**

D. Complexity analysis

It can be seen from the above description that the time complexity of the proposed method is mainly determined by the extraction of the population state, the execution of the guidance strategy, the evaluation of the guidance strategy, and the training of the DQN.

When the dimension of decision space for solving the optimization problem is d, for the evolutionary algorithm whose maximum evolutionary generation number is T_{max} and population size is n, to extract the population state, it is necessary to calculate the distance between the population individuals. Executing guidance strategies involves vector computation of all population individuals. The evaluation of the guidance strategy is related to the number of feasible solutions and the number m of optimization objectives. For DQN, assume the neural network contains h layers, and the average number of nodes in the hidden layer is p, the number of nodes in the input and output layers is 10 and 1, respectively. Each training or update needs g iterations, and a total of b samples are selected for training. The time complexity of these four parts is as follows:

(1) The time complexity of extracting population state is $O(T_{max}nd)$;

(2) The time complexity of executing the guidance strategy in the worst case is $O(T_{max}nd)$;

(3) The time complexity of guidance strategy evaluation in the worst case is $O(T_{max}n^2m + T_{max}n^{m-2}\ log\ n)$[60];

(4) The time complexity of training and updating DQN is $O(\lfloor\frac{T_{max}-T_{sam}}{T_{upd}}\rfloor((h-3)p^2 + 11p)bg)$.

It can be seen that the time complexity of the proposed method is mainly determined by T_{max}, n, d, m and the parameters of DQN. For large-scale constrained multi-objective optimization problems, the above time complexity is mainly determined by the extraction of population state and the execution of guidance strategies, about $O(T_{max}nd)$; When the number of objectives increases, the time complexity is mainly determined by the evaluation of guidance strategy, about $O(T_{max}n^2m + T_{max}n^{m-2}\ log\ n)$; When the dimensions of the decision space and objective space of the optimization problem are small, the time complexity is mainly determined by the training and update of DQN, about $O(\lfloor\frac{T_{max}-T_{sam}}{T_{upd}}\rfloor((h-3)p^2 + 11p)bg)$.

E. Further elucidation

The core idea of PKAEO is to select appropriate strategies to improve the performance of evolutionary algorithms by tracking the changes in the population state. Previously, Sharma et al.[61] proposed a description method for population states based on the ranking of population individuals and the locations of the population. In their method, DDQN is used to select the search strategies automatically. In terms of the solved problems, their method is devoted to the single-objective optimization problems and has the following flaws: (1) the population state description is redundant; (2) the strategy evaluation method is simplified; (3) the offline training of samples is required. In addition, Tian et al.[44] proposed an automatic operator selection framework guided by decision variables and weight vectors for multi-objective optimization problems. Unlike our method, their method evaluates the performance of guidance strategies by comparing their improvements in fitness value and does not involve the state description of the evolutionary population, therefore provides no straightforward way to mine the population information. More importantly, PKAEO can easily embed existing evolutionary algorithms. By setting preferential evaluation methods of guidance strategies, the most beneficial strategies are intelligently selected to improve the performance of algorithms.

IV. EXPERIMENT

Eight experiments are conducted in this section to evaluate the proposed method's performance. The first group verifies the effectiveness of the proposed method on benchmark problems by comparing it with five constrained multi-objective optimization algorithms. The second group verifies the superiority of the proposed method on benchmark problems by comparing it with four multi-strategy optimization frameworks. The third to fifth groups study the influence of parameter α on the algorithm stability. The fifth group studies the comparison of intelligent recommendation of strategies in PKAEO with empirical use of strategies. The sixth group studies the suitable number of configured guidance strategies. The last group studies the time consumption of PKAEO. The operating environment of the experiments is as follows: Intel(R) Core(TM) i5-9500 CPU @ 3.00GHz, 32GB RAM, Windows 10, and PlatEMO[62].

A. Benchmark test suites

In order to evaluate the performance of the proposed method, four benchmark suites are selected: LIR-CMOP[63], MW[64], DTLZ[65], and VNT[66]. These test suites cover the four common types of constrained multi-objective optimization problems[4], namely (1) the UPF and the CPF completely coincide; (2) Parts of UPF are feasible, and CPF is parts of UPF; (3) Some regions of UPF are feasible, and CPF and UPF partly coincide; (3) UPF is located in the infeasible region, and CPF is wholly separated from UPF. The above indicate that the selected test suites are representative and helpful in testing the proposed method.

The number of optimization objectives and the dimension of decision variables are set as follows. For LIR-CMOP1~LIR-CMOP12, $m = 2$, $d = 30$; for LIR-CMOP13~LIR-CMOP14, $m = 3$, $d = 30$; for MW4, MW8 and MW14, $m = 3$, $d = 15$; for other MW test instances, $m = 2$, $d = 15$; for DTLZ8, $m = 3$, $d = 30$; for DTLZ9, $m = 2$, $d = 20$; for VNT4, $m = 3$, $d = 2$.

B. Comparison algorithms and parameters setting

In this section, the state-of-the-art algorithms for solving constrained multi-objective optimization problems, including CMOEA-MS[36], AGEMOEA[67], AGEMOEAII[68], ARMOEA[69] and the typical algorithm NSGA-II[70], are selected as the embedded objects of the process knowledge guidance proposed in this paper. The variants with process knowledge guidance are denoted as "PK-*". The effectiveness of PKAEO is verified by comparing the performance of "PK-*" and original algorithms. To illustrate the superiority of PKAEO, four multi-strategy frameworks for solving constrained multi-objective optimization problems, including C-TAEA[71], MOCell[72], MaOEAIT[73], and Top[74], are selected as the comparison algorithms. The parameter values of the above algorithms are consistent with those in the original articles.

In "PK-*", the parameters of DQN are set as $\gamma = 0.9$, $N = 500$, $g = 100$, $T_{upd} = 0.25 * T_{max}$, $T_{sam} = 0.3 * T_{max}$, $\alpha = 5$; the number of nodes in the hidden layer of DNN is 32, 64, and 32 respectively, and the learning rate of network training $lr = 0.001$. The parameters of guidance strategies are set to $F \in [0,0.5]$, $CR \in [0,1]$. Considering that if all the guidance strategies are adopted, the ample strategy space will have a high requirement on the number of samples. Therefore, from the perspective of maintaining population diversity and convergence, DE/rand/1 and DE/current-to-pbest/1 are selected in the experiment, i.e. $a_t = 1$ and $a_t = 2$, respectively.

The population size is 100 for all test instances. The function evaluation times of each algorithm are 100000 and 10000 for the LIR-CMOP test suite and the other test suites, respectively.

C. Performance indicator

Since the Inverted Generational Distance (IGD)[75] and hyper-volume (HV)[59] can comprehensively evaluate the performance of intelligent optimization algorithms, they are selected as performance indicators in this paper. In order to evaluate the significant difference of compared methods on performance indicators, the Wilcoxon Rank-sum test is used for the hypothesis test, and the significance level is set at 0.05. The symbols "+", "-" and "=" are used to indicate that "PK-*" are significantly better than, worse than and not different from the original algorithms, respectively. In addition, the multi-problem Wilcoxon Rank-sum test with a significance level of 0.05 is used to evaluate the significance of the performance difference between the compared algorithms.

D. Effectiveness of the proposed method

For all problems, each algorithm is run 30 times independently to obtain the HV and IGD performance indicators. Due to the space limitation, their mean values and standard deviations are listed in Tables S-1 to S-6 in the Appendix, in which the highlighted gray background data are the best values. **Table 1** lists the HV and IGD performance indicators of "PK-*" and each original algorithm. For IGD, "PK-*" significantly outperform the original algorithms on at most 20 and at least 17 test problems. For HV, "PK-*" significantly outperform the original algorithms on at most 19 and at least 16 test problem. Considering the R^+ value of "PK-*" is higher than the R^- value, it can be seen that process knowledge guidance can improve the performance of the embedded algorithms.

Table 1 IGD and HV performance indicators of "PK-*" and embedded algorithms

IGD	**+/-/=**	R^+	R^-	α**=0.05**
PK-NSGAII vs NSGAII	20/2/9	473.5	22.5	YES
PK-AGEMOEA vs AGEMOEA	19/2/10	251.5	213.5	YES
PK-AGEMOEAII vs AGEMOEAII	19/4/8	454.5	41.5	YES
PK-ARMOEA vs ARMOEA	20/3/8	291.5	204.5	YES
PK-CMOEA-MS vs CMOEA-MS	17/6/8	343.0	153.0	YES
HV	**+/-/=**	R^+	R^-	α**=0.05**
PK-NSGAII vs NSGAII	16/0/15	465.0	0.0	YES
PK-AGEMOEA vs AGEMOEA	19/1/11	494.5	1.5	YES
PK-AGEMOEAII vs AGEMOEAII	18/3/10	426.5	38.5	YES
PK-ARMOEA vs ARMOEA	17/3/11	494.5	1.5	YES
PK-CMOEA-MS vs CMOEA-MS	17/6/8	496.0	0.0	YES

For the MW test instances, the constraints have various styles. Specifically, the feasible regions are tiny and divided by a vast infeasible region, causing the CPF contains multiple isolated solutions. According to Tables S-1 and S-4, process knowledge guidance is practical for most problems. In order to visually demonstrate the performance effect of process knowledge guidance, the distribution of the non-dominated solutions provided by "PK-*" and the original algorithms on the MW3 problem is shown in Fig.S-1 and Fig.S-2, respectively. It can be seen that process knowledge guidance can effectively guide the algorithm to obtain more widely distributed non-dominated solutions. More specifically, according to the searching process of PK-ARMOEA on MW3 and MW5 in Fig.S-3 and Fig.S-4, the process knowledge guidance first guides the population to locate feasible solutions quickly. It then guides the population to approach the CPF quickly once a feasible solution is found. Finally, process knowledge guidance improves the diversity distribution of the non-dominated solutions.

The feasible regions for the LIR-CMOP test instances are tiny, and even some problems contain only one curve. In addition, the CPF is divided by a vast infeasible region, forming several disjoint segments or sparse points. According to Tables S-2 and S-5, process knowledge guidance can improve the comprehensive performance of all embedded algorithms on LIR-CMOP. However, the performance on different test instances is slightly different, detailed as follows.

This article has been accepted for publication in IEEE Transactions on Evolutionary Computation. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2023.3243109

(1) For LIR-CMOP1~LIR-CMOP4 with small feasible regions, process knowledge guidance can help the embedded algorithm to obtain better non-dominated solutions, which is attributed to the fast localization of the feasible regions by DE/rand/1 in global exploration and the fast convergence of DE/current-to-pbest/1 in local exploitation.

(2) For LIR-CMOP5~LIR-CMOP6, which need to go through the infeasible regions to locate the CPF, the process knowledge guidance is limited by the constraint handling ability of the embedded algorithm to a certain extent, which makes it challenging to improve the performance of AGEMOEA and ARMOEA significantly. Nevertheless, it can help NSGAII find feasible solutions for LIR-CMOP6.

(3) For LIR-CMOP7~LIR-CMOP14, process knowledge guidance has significant advantages, which can improve the performance of all algorithms on most test instances.

For the DTLZ and VNT test instances, the constrained multi-objective optimization problems DTLZ8~DTLZ9 and VNT4 are selected. According to Tables S-3 and S-6, except for the CMOEA-MS algorithm, process knowledge guidance can improve the comprehensive performance of all other algorithms. According to the solving process of ARMOEA and PK-ARMOEA on the DTLZ8 problem in Fig.S-5 and Fig.S-6, PK-ARMOEA maintains a high population diversity in the evolution process, showing obvious advantages at the 50th generation and converging to the CPF at the 75th generation.

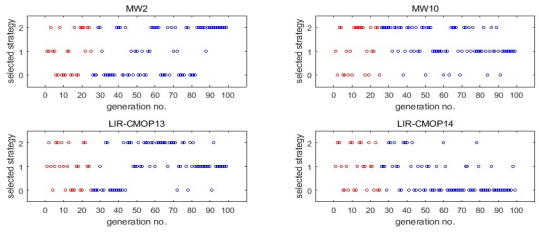


Fig.3 Strategy selection results of PK-ARMOEA in the process of solving MW2, MW10, LIR-CMOP13, and LIR-CMOP14, where red dots represent the random strategy selection in generations 1~T_{sam}, and the blue dots represent the DQN strategy recommendation in the process knowledge application.

It is worth noting that process knowledge guidance is not effective in some test instances. According to the distribution of non-dominated solutions to these test instances shown in Fig.S-7 and the strategy selection of PK-ARMOEA given in **Fig.3**, we can see that

(1) The non-dominated solutions of MW2 and MW10 distribute centrally on some dimensions. Thus, DE/current-to-pbest/1 with fast convergence gets a higher evaluation value than DE/rand/1. For MW2, to locate the promising regions, no guidance is performed in the early stage; the DE/current-to-pbest/1 is used to accelerate the convergence in the late stage. Although the algorithm has performed the DE/current-to-pbest/1 in the generations T_{sam+1}~T_{max}, the strategy sampling in generations 1~T_{sam}, specifically the random execution of DE/rand/1, still reduces the convergence speed. For MW10, the population is premature. Though the DE/rand/1 is employed to search for new promising regions, it is still challenging to significantly improve the solutions further.

(2) The non-dominated solutions of LIR-CMOP13 and LIR-CMOP14 are dispersed and located in different local regions. For LIR-CMOP13, at the early stage of process knowledge application (generations T_{sam+1}~T_{max}), the continuous employment of DE/current-to-pbest/1 focuses on local exploitation and neglects global exploration. Even if the DE/rand/1 is performed at the late stage of process knowledge application, the quality of the non-dominated solutions is still affected. For LIR-CMOP14, the mutation operators of DE are not effective, thus no guidance is conducted in process knowledge application.

In summary, process knowledge guidance has a significant effect on improving the performance of evolutionary algorithms. However, in the application process, the immediate reward of fast convergence strategies and the random sampling for exploring the effect of strategies will lead to the wrong guidance of the population's evolution. Therefore, the design of a more scientific strategy evaluation system is the critical factor affecting the effect of process knowledge guidance.

E. Further research

1) Comparison with other multi-strategy frameworks

At present, the multi-strategy frameworks for solving constrained multi-objective optimization problems include C-TAEA, MOCell, MaOEAIT, and Top. This section compares the performance of PKAEO with these frameworks. In addition to the above test suites, the CF test suite[76] is also used here. Taking PK-CMOEA-MS as an example, each algorithm is independently run 30 times for all optimization problems to calculate the mean and standard deviation of HV and IGD indicators, as listed in Tables S-7 to S-12. **Table 2** lists the HV and IGD performance statistics of PK-CMOEA-MS and other algorithms. For IGD, PK-CMOEA-MS significantly outperforms the original algorithms on at most 35 and at least 25 test problems. For HV, PK-CMOEA-MS outperforms the original algorithms on at most 35 and at least 22 test problems. The overall advantage of PKAEO is verified according to the higher R^+ value than the R^- value of PK-CMOEA-MS. In addition, the Friedman test is used to compare the IGD and HV of all algorithms, and the results are listed in **Tables 3** and **4**. It is easy to see that PK-CMOEA-MS ranks best among all the comparison algorithms. The detailed result analysis is given as follows:

(1) For the MW test instances, MOCell and MaOEAIT have similar performances. On HV and IGD indicators, PK-CMOEA-MS achieves both 12+/0-/2= in comparison with these two algorithms. Compared with CTAEA, the advantage of PK-CMOEA-MS is slight, only 7+/6-/1= is achieved on both HV and IGD indicators. Top performs worst on the MW test instances, which only finds the feasible solutions of two problems.

(2) For the LIRCMOP test instances, the advantage of PK-CMOEA-MS is significant. Specifically, PK-CMOEA-MS performs better than MOCell and MaOEAIT on all the problems. Compared with ToP, PK-CMOEA-MS shows superiority on 13 ones among all the problems in terms of HV indicator; as for the IGD indicator, PK-CMOEA-MS achieves 12+/0-/2=. CTAEA performs better than PK-CMOEA-MS on 4 and 3 problems in terms of HV and IGD indicators, respectively.

(3) For other problems, PK-CMOEA-MS shows the most obvious advantage in comparison with ToP, where it achieves 11+/1-/1= on both HV and IGD indicators. Compared with MOCell, PK-CMOEA-MS achieves 9+/2-/1= on both HV and IGD indicators. Similarly, PK-CMOEA-MS achieves 9+/2-/2=

and 9+/1-/3= on HV and IGD indicators, respectively. Compared with CTAEA, PK-CMOEA-MS achieves 8+/0-/5= on both HV and IGD indicators.

Table 2 IGD and HV indicator statistics of PK-CMOEA-MS and other algorithms

IGD	+/-/=	R^+	R^-
PK-CMOEA-MS vs C-TAEA	23/9/9	597.0	264.0
PK-CMOEA-MS vs MOCell	35/2/3	796.0	65.0
PK-CMOEA-MS vs MaOEAIT	26/1/4	769.0	92.0
PK-CMOEA-MS vs ToP	25/1/5	740.0	121.0
HV	+/-/=	R^+	R^-
PK-CMOEA-MS vs C-TAEA	22/10/9	567.0	294.0
PK-CMOEA-MS vs MOCell	35/2/3	807.5	54.0
PK-CMOEA-MS vs MaOEAIT	26/2/3	841.0	20.0
PK-CMOEA-MS vs ToP	26/1/4	858.0	3.0

Table 3 IGD ranking of PK-AGEMOEA and comparison algorithms

Algorithm	Ranking
PK-CMOEA-MS	1.88
C-TAEA	2.34
ToP	3.20
MOCell	3.54
MaOEAIT	4.05

Table 4 HV ranking of PK-AGEMOEA and comparison algorithms

Algorithm	Ranking
PK-CMOEA-MS	1.56
C-TAEA	1.99
ToP	3.11
MOCell	3.24
MaOEAIT	4.10

2) Sensitivity analysis of α

Since the purpose of this paper is to propose a process knowledge guidance framework for evolutionary algorithms, we do not focus on fine-tuning the framework parameters, such as the scaling factor F of search strategy, crossover probability CR, parameters of DQN including lr, T_{sam} and T_{upd}. However, we should pay attention to the stability of PKAEO, which is closely related to the times of data augmentation (α) for DQN.

In order to investigate the influence of α on PKAEO, PK-AGEMOEA is taken as a study example. For each optimization problem, PK-AGEMOEA is independently run 30 times to calculate the mean and standard deviation of HV and IGD indicators, as listed in Tables S-13 and S-14. It can be seen that the value of α has a significant impact on the algorithm's stability. In addition, the Friedman test is used to compare the standard deviation of HV under different settings of α, and the analysis results are listed in **Table 5**. It is easy to find that as the value of α increases, PK-AGEMOEA becomes more and more stable. Thus, according to algorithms' stability requirement and the limitation of computing resources, an appropriate value can be set in real-world applications.

Table 5 Stability ranking of PK-AGEMOEA with different α values

Algorithm	Ranking
PK-AGEMOEA ($\alpha = 5$)	2.18
PK-AGEMOEA ($\alpha = 10$)	1.92
PK-AGEMOEA ($\alpha = 15$)	1.90

3) Sensitivity analysis of T_{sam}

T_{sam} determines the size of collected samples for training the DQN. A smaller T_{sam} will cause the insufficiency of samples, reducing the accuracy of trained DQN, while a larger T_{sam} pays too much attention to sample collection, shortening the scope of evolutionary guidance. To study the influence of T_{sam} on the algorithm, PK-NSGAII with T_{sam} = 0.1*T_{max}, 0.2*T_{max}, 0.3*T_{max} and 0.4*T_{max} are tested and compared. They are independently run 30 times to calculate the mean and standard deviation of the HV indicator, as listed in Table S-15. The Friedman test is used to compare the HV of all algorithm variants, and the results are listed in **Table 6**. Overall, PK-NSGAII is not sensitive to the $T_{sam} \in [0.1 * T_{max}, 0.4 * T_{max}]$. It achieves the best and worst performances at $T_{sam} = 0.3 * T_{max}$ and $T_{sam} = 0.2 * T_{max}$, respectively. Therefore, $T_{sam} = 0.3 * T_{max}$ is a promising setting.

Table 6 HV ranking of PK-NSGAII with different T_{sam} values

Algorithm	Ranking
PK-NSGAII ($T_{sam} = 0.3 * T_{max}$)	2.40
PK-NSGAII ($T_{sam} = 0.4 * T_{max}$)	2.46
PK-NSGAII ($T_{sam} = 0.1 * T_{max}$)	2.45
PK-NSGAII ($T_{sam} = 0.2 * T_{max}$)	2.71

4) Sensitivity analysis of T_{upd}

T_{upd} determines the frequency of updating the DQN. A smaller T_{upd} will cause instability of the DQN, and increase the time consumption, while a larger T_{upd} is unable to timely perceive the dynamic changes of strategy evaluation. To study the influence of T_{upd} on the algorithm, PK-NSGAII with T_{upd} = 0.1* T_{max}, 0.2* T_{max}, 0.3* T_{max} and 0.4* T_{max} are tested and compared. They are independently run 30 times to calculate the mean and standard deviation of the HV indicator, as listed in Table S-16. The Friedman test is used to compare the HV of all algorithm variants, and the results are listed in **Table 7**. Overall, PK-NSGAII is not sensitive to the $T_{upd} \in [0.1 * T_{max}, 0.4 * T_{max}]$. It achieves the best and worst performances at $T_{upd} = 0.3 * T_{max}$ and $T_{upd} = 0.4 * T_{max}$, respectively. Therefore, $T_{upd} = 0.25 * T_{max}$ is a promising setting.

Table 7 HV ranking of PK-NSGAII with different T_{upd} values

Algorithm	Ranking
PK-NSGAII ($T_{upd} = 0.3 * T_{max}$)	2.36
PK-NSGAII ($T_{upd} = 0.2 * T_{max}$)	2.47
PK-NSGAII ($T_{upd} = 0.1 * T_{max}$)	2.47
PK-NSGAII ($T_{upd} = 0.4 * T_{max}$)	2.71

5) Comparison with the empirical use of strategies

In PKAEO, the guidance strategies are selectively recommended by the DQN. However, there is already successful experience to better arrange the execution of these strategies. For example, the DE/rand/1 and DE/current-to-pbest/1 are tendentiously employed in the early and late stages respectively, which is beneficial in balancing the global search and local exploitation. To realize this empirical use of strategies, the DE/rand/1 and DE/current-to-pbest/1 are fixedly executed at $t < \frac{T_{max}}{2}$ and $t \geq \frac{T_{max}}{2}$, respectively. This variant deployed in PK-NSGAII is named PK-NSGAII-rand. PK-NSGAII-rand is independently run 30 times to calculate the mean and standard deviation of HV and IGD indicators, as listed in Tables S-17 and S-18. For HV, PK-NSGAII performs better on 9 test problems. For IGD, PK-NSGAII outperforms the PK-NSGAII-rand on 10 test problems. It is worth noting that the empirical strategy configuration has also achieved good results. To further strengthen the PKAEO, offline training can help DQN to acquire the ability of configuring different strategies for the entire evolution process and automatically switch strategies at different stages.

6) The influence of the number of guidance strategies

This article has been accepted for publication in IEEE Transactions on Evolutionary Computation. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2023.3243109

As the above-mentioned reasons, only two guidance strategies are configured for PKAEO. To verify the rationality of this choice, PK-NSGAII is tested with different number of guidance strategies. The variants of PK-NSGAII with DE/rand/1 and DE/current-to-pbest/1 are named PK-NSGAII-T1 and PK-NSGAII-T2, respectively; the variant of PK-NSGAII with DE/rand/1, DE/current-to-pbest/1 and DE/best/1 is named PK-NSGA-II-T3. PK-NSGAII is still configured with DE/rand/1 and DE/current-to-pbest/1. According to the HV indicator shown in Table S-19, the Friedman test is used to compare the HV of all algorithm variants, and the results are listed in **Table 8**. Some interesting phenomenon can be concluded as follows:

(1) DE/rand/1 is more effective than DE/current-to-pbest/1 on most problems. For HV, PK-NSGAII performs better than PK-NSGAII-T2 on at most 8 and at least 1 test problems; in comparison with PK-NSGAII-T1, PK-NSGAII performs better on at most 3 and at least 1 test problems.

(2) Configuring with two guidance strategies is a promising selection than considering more candidates. For HV, PK-NSGAII performs better than PK-NSGAII-T3 on at most 6 and at least 3 test problems.

Overall, the current PKAEO is just capable of dispatching two guidance strategies. To further strengthen the PKAEO, offline training can help DQN to identify the matching relationship between other guidance strategies and population states.

Table 8 HV ranking of PK-NSGAII with different strategies

Algorithm	Ranking
PK-NSGAII	2.26
PK-NSGAII-T1	2.35
PK-NSGAII-T3	2.40
PK-NSGAII-T2	2.98

7) The time consumption of PKAEO

The process knowledge guidance module brings extra time consumption. For MW, LIR-CMOP, DTLZ and VNT, the comparison of PK-NSGAII and NSGAII in terms of time consumption is shown in **Fig.4**. Obviously, the extra time consumption caused by the process knowledge guidance is significant. According to the operation of PKAEO, the time consumption is also partially caused by the high feasible rate of population, and therefore the subsequent feature extraction of feasible solutions. To reduce the time consumption, the feature extraction should be simplified, especially for the computation related to the distances between population individuals.

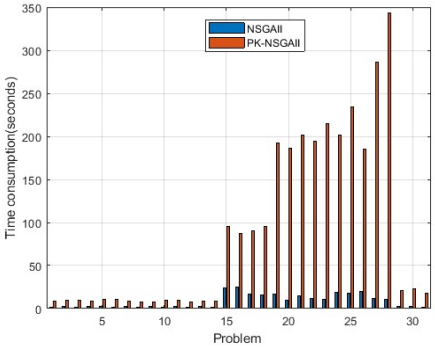


Fig.4 Time consumption of PK-NSGAII and NSGAII

V. Application To the Dispatch Of Integrated Coal Mine Energy System

This section discusses the application of process knowledge guidance to the optimal dispatch of integrated coal mine energy system (ICMES). Based on the dispatch optimization model of the integrated coal mine energy systems in ref.[1], the objectives of minimizing economic cost and carbon emission allowance are selected, and 30 strongly constrained multi-objective optimization problems ICMES1~ICMES30 are designed by setting different cold, heat, electrical load, light, and wind intensity data. More details about the optimization model are provided in the supplementary material. For all ICMES problems, m=2, d=384. Considering that it is challenging to find feasible solutions for strongly constrained multi-objective optimization problems, this section also uses the feasible rate (FR) to evaluate the performance of the algorithms. It sets the function evaluation times to be 50000. For each problem, all algorithms are independently run 30 times to obtain the mean value and standard deviation of HV and IGD. The results are listed in Tables S-17 and S-18. Since CMOEA-MS cannot solve the ICMES problems, the results of PK-CMOEA-MS and CMOEA-MS are not given here. **Table 9** lists the HV and IGD statistics of "PK-*" and each original algorithm. For HV, "PK-*" are significantly better than the original algorithms on at most 11 and at least 9 test instances. For FR, "PK-*" are significantly better than the original algorithms on all test instances. As R^+ value of "PK-*" is higher than R^- value, it can be seen that process knowledge guidance can improve the performance of the embedded algorithms.

The distribution of non-dominated solutions on ICMES23 is presented in **Fig.5**, showing that the ICMES problems' feasible regions are tiny. Taking the labeled non-dominated solution in **Fig.5** as a study example, **Fig.6** shows the energy dispatch solution, including the consumption and generation of energy during different periods. The dispatch satisfies the cold, electricity, and heat load constraints through multi-energy complementation and minimizes the economic cost and carbon emission allowance.

In the guidance strategies of current PKAEO, the scaling factor $F \in [0,0.5]$. Considering the purpose of this paper is to develop a framework, therefore the parameters are not fine-tuned. However, in the above experiments, we found when setting $F \in [0,1]$, the performance of PKAEO can be further enhanced. The reasons of this result can be deduced as follows:

(1) For DE/rand/1, a larger F is more beneficial for the global exploration, especially for the strongly constrained optimization problems. For DE/current-to-pbest/1, the larger F motivates the population move faster to the better individuals, which accelerates the convergence.

(2) Limited by the feasibility requirements of engineering application, the upper and lower boundaries of decision variables are not given in large ranges. Hence, on some variables, the feasible solutions are situated on the boundaries. When generating mutated solutions, the larger F brings about more elements exceeding the boundaries. In this case, the boundary check technique focusing on the boundary search is very effective.

Table 9 HV and FR indicator statistics of PK-CMOEA-MS and other algorithms

HV	**+/-/=**	R^+	R^-	α=0.05
PK-NSGAII vs NSGAII	9/0/17	460.0	5.0	YES
PK-AGEMOEA vs AGEMOEA	6/0/17	465.0	0.0	YES
PK-AGEMOEAII vs AGEMOEAII	6/0/17	456.0	9.0	YES
PK-ARMOEA vs ARMOEA	11/0/13	435.0	0.0	YES
PK-CMOEA-MS vs CMOEA-MS	\	\	\	\
FR	**+/-/=**	R^+	R^-	α=0.05
PK-NSGAII vs NSGAII	30/0/0	465.0	0.0	YES
PK-AGEMOEA vs AGEMOEA	30/0/0	465.0	0.0	YES
PK-AGEMOEAII vs AGEMOEAII	30/0/0	465.0	0.0	YES
PK-ARMOEA vs ARMOEA	30/0/0	465.0	0.0	YES
PK-CMOEA-MS vs CMOEA-MS	\	\	\	\

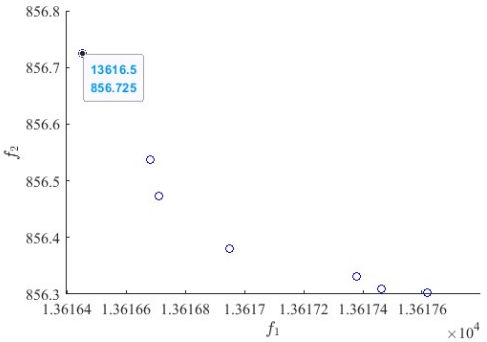
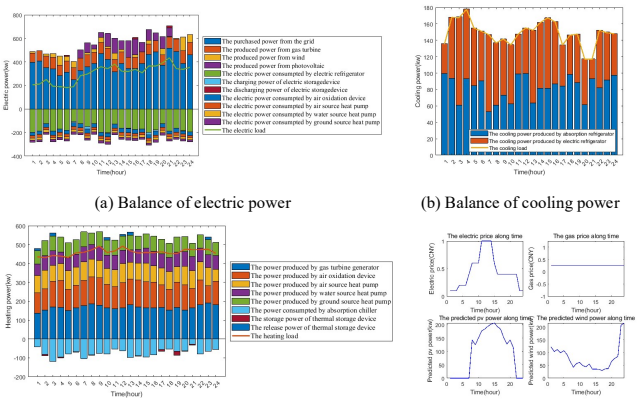


Fig.5 Non-dominated solutions to ICMES23 problem found by PK-NSGAII



(a) Balance of electric power (b) Balance of cooling power

(c) Balance of heating power (d) Key parameters

Fig.6 Dispatch solution and key parameters in the optimization model

VI. Conclusion

For constrained multi-objective optimization problems, we propose an evolutionary optimization framework based on population convergence and diversity regulation, namely PKAEO. Firstly, PKAEO accumulates a certain amount of samples by randomly executing guidance strategies at the early evolutionary stage and evaluating the regulatory effects of different guidance strategies on the population. Then, the samples are used to train DQN, and the mapping model between population state and guidance strategy is established. According to the established mapping model and population state, the subsequent guidance strategy is recommended. Finally, with the continuous enrichment of sampled data in the evolution process, DQN is periodically updated to improve the accuracy of guidance strategy recommendations.

In order to evaluate the performance of PKAEO, we apply it to 41 benchmark test problems. Experimental results show that PKAEO can effectively improve the performance of embedded algorithms, especially in locating small and decentralized feasible regions. In addition, the application effectiveness of PKAEO on the dispatch optimization problem of integrated coal mine energy systems proves that PKAEO is practical and scalable.

To further improve the performance of PKAEO, we will conduct future research in following directions: (1) the undifferentiated execution of guidance strategies at the sampling stage caused the evolution direction misguidance. The offline deep reinforcement learning method can be used to assist decision-making and avoid sampling during the execution process of the evolutionary algorithm; (2) aiming at the premature population convergence caused by the immediate reward of guidance strategy, an evolutionary strategy selection method combining manual intervention and autonomous selection could be adopted to limit the freedom of DQN strategy selection.

References

[1] Hu H, Sun X, Zeng B, Gong D, Zhang Y. Enhanced evolutionary multi-objective optimization-based dispatch of coal mine integrated energy system with flexible load[J]. Applied Energy, 2021: 118130.

[2] Gong D, Xu B, Zhang Y, Guo Y, Yang S. A similarity-based cooperative co-evolutionary algorithm for dynamic interval multiobjective optimization problems[J]. IEEE Transactions on Evolutionary Computation, 2019, 24 (1): 142-156.

[3] Zuo M, Dai G, Peng L, Wang M, Liu Z, Chen C. A case learning-based differential evolution algorithm for global optimization of interplanetary trajectory design[J]. Applied Soft Computing, 2020, 94: 106451.

[4] Liang J, Ban X, Yu K, Qu B, Qiao K, Yue C, Chen K, Tan K C. A survey on evolutionary constrained multi-objective optimization[J]. IEEE Transactions on Evolutionary Computation, 2022.

[5] Liang Z, Liang W, Xu X, Liu L, Zhu Z. A two stage adaptive knowledge transfer evolutionary multi-tasking based on population distribution for multi/many-objective optimization[J]. arXiv preprint arXiv:.00810, 2020.

[6] Li J-Y, Zhan Z-H, Tan K C, Zhang J. A meta-knowledge transfer-based differential evolution for multitask optimization[J]. IEEE Transactions on Evolutionary Computation, 2021, 26 (4): 719-734.

[7] Zhang X, Zhan Z-H, Fang W, Qian P, Zhang J. Multipopulation ant colony system with knowledge-based local searches for multiobjective supply chain configuration[J]. IEEE Transactions on Evolutionary Computation, 2021, 26 (3): 512-526.

[8] Ding Z, Chen L, Sun D, Zhang X. A multi-stage knowledge-guided evolutionary algorithm for large-scale sparse multi-objective optimization problems[J]. Swarm and Evolutionary Computation, 2022, 73: 101119.

[9] Cheng R, Jin Y, Olhofer M, Sendhoff B. A reference vector guided evolutionary algorithm for many-objective optimization[J]. IEEE Transactions on Evolutionary Computation, 2016, 20 (5): 773-791.

[10] Liu Q, Jin Y, Heiderich M, Rodemann T, Yu G. An adaptive reference vector-guided evolutionary algorithm using growing neural gas for many-objective optimization of irregular problems[J]. IEEE Transactions on Cybernetics, 2020.

[11] Wimmer H, Rada R. Good versus bad knowledge: Ontology guided evolutionary algorithms[J]. Expert systems with applications, 2015, 42 (21): 8039-8051.

[12] Chen G, Li J. A diversity ranking based evolutionary algorithm for multi-objective and many-objective optimization[J]. Swarm and Evolutionary Computation, 2019, 48: 274-287.

[13] Cheng J, Pan Z, Liang H, Gao Z, Gao J. Differential evolution algorithm with fitness and diversity ranking-based mutation operator[J]. Swarm and Evolutionary Computation, 2021, 61: 100816.

[14] Zeng S, Jiao R, Li C, Li X, Alkasassbeh J S. A general framework of dynamic constrained multiobjective evolutionary algorithms for constrained optimization[J]. IEEE transactions on Cybernetics, 2017, 47 (9): 2678-2688.

[15] Zuo M, Dai G, Peng L, Tang Z, Gong D, Wang Q. A differential evolution algorithm with the guided movement for population and its application to

interplanetary transfer trajectory design[J]. Engineering Applications of Artificial Intelligence, 2022, 110: 104727.
[16] Jan M A, Zhang Q. MOEA/D for constrained multiobjective optimization: Some preliminary experimental results[C].2010 UK Workshop on computational intelligence (UKCI),2010: 1-6.
[17] Maldonado H M, Zapotecas-Martínez S. A Dynamic Penalty Function within MOEA/D for Constrained Multi-objective Optimization Problems[C].2021 IEEE Congress on Evolutionary Computation (CEC),2021: 1470-1477.
[18] Vaz F, Lavinas Y, Aranha C, Ladeira M. Exploring Constraint Handling Techniques in Real-world Problems on MOEA/D with Limited Budget of Evaluations[C].International Conference on Evolutionary Multi-Criterion Optimization,2021: 555-566.
[19] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. IEEE Transactions on Evolutionary Computation, 2002, 6 (2): 182-197.
[20] Takahama T, Sakai S. Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites[C].2006 IEEE International Conference on Evolutionary Computation,2006: 1-8.
[21] Runarsson T P, Yao X. Stochastic ranking for constrained evolutionary optimization[J]. IEEE Transactions on evolutionary computation, 2000, 4 (3): 284-294.
[22] Jiao R, Zeng S, Li C. A feasible-ratio control technique for constrained optimization[J]. Information Sciences, 2019, 502: 201-217.
[23] Jiao R, Zeng S, Li C, Yang S, Ong Y-S. Handling constrained many-objective optimization problems via problem transformation[J]. IEEE Transactions on Cybernetics, 2020, 51 (10): 4834-4847.
[24] Ying W-Q, He W-P, Huang Y-X, Li D-T, Wu Y. An adaptive stochastic ranking mechanism in MOEA/D for constrained multi-objective optimization[C].2016 International Conference on Information System and Artificial Intelligence (ISAI),2016: 514-518.
[25] Ray T, Singh H K, Isaacs A, Smith W: Infeasibility driven evolutionary algorithm for constrained optimization,Constraint-handling in evolutionary optimization: Springer,2009: 145-165.
[26] Long Q. A constraint handling technique for constrained multi-objective genetic algorithm[J]. Swarm Evolutionary Computation, 2014, 15: 66-79.
[27] Zhou Y, Zhu M, Wang J, Zhang Z, Xiang Y, Zhang J. Tri-goal evolution framework for constrained many-objective optimization[J]. IEEE Transactions on Systems, Man, Cybernetics: Systems, 2018, 50 (8): 3086-3099.
[28] Vieira D A, Adriano R L, Vasconcelos J A, Krahenbuhl L. Treating constraints as objectives in multiobjective optimization problems using niched Pareto genetic algorithm[J]. IEEE Transactions on Magnetics, 2004, 40 (2): 1188-1191.
[29] Wang J, Liang G, Zhang J. Cooperative differential evolution framework for constrained multiobjective optimization[J]. IEEE transactions on cybernetics, 2018, 49 (6): 2060-2072.
[30] Liu B, Ma H, Zhang X, Zhou Y. A memetic co-evolutionary differential evolution algorithm for constrained optimization[C].2007 IEEE Congress on Evolutionary Computation,2007: 2996-3002.
[31] Tian Y, Zhang T, Xiao J, Zhang X, Jin Y. A coevolutionary framework for constrained multiobjective optimization problems[J]. IEEE Transactions on Evolutionary Computation, 2020, 25 (1): 102-116.
[32] Fan Z, Li W, Cai X, Li H, Wei C, Zhang Q, Deb K, Goodman E. Push and pull search for solving constrained multi-objective optimization problems[J]. Swarm and evolutionary computation, 2019, 44: 665-679.
[33] Tian Y, Zhang Y, Su Y, Zhang X, Tan K C, Jin Y. Balancing objective optimization and constraint satisfaction in constrained evolutionary multiobjective optimization[J]. IEEE Transactions on Cybernetics, 2021.
[34] Yu K, Liang J, Qu B, Yue C. Purpose-directed two-phase multiobjective differential evolution for constrained multiobjective optimization[J]. Swarm Evolutionary Computation, 2021, 60: 100799.
[35] Jiao R, Zeng S, Li C, Ong Y-S. Two-type weight adjustments in MOEA/D for highly constrained many-objective optimization[J]. Information Sciences, 2021, 578: 592-614.
[36] Morovati V, Pourkarimi L. Extension of Zoutendijk method for solving constrained multiobjective optimization problems[J]. European Journal of Operational Research, 2019, 273 (1): 44-57.
[37] Schütze O, Alvarado S, Segura C, Landa R. Gradient subspace approximation: a direct search method for memetic computing[J]. Soft Computing, 2017, 21 (21): 6331-6350.
[38] Yu X, Yu X, Lu Y, Yen G G, Cai M. Differential evolution mutation operators for constrained multi-objective optimization[J]. Applied Soft Computing, 2018, 67: 452-466.
[39] Yu X, Luo W, Xu W, Li C. Constrained multi-objective differential evolution algorithm with ranking mutation operator[J]. Expert Systems with Applications, 2022, 208: 118055.
[40] Xu B, Duan W, Zhang H, Li Z. Differential evolution with infeasible-guiding mutation operators for constrained multi-objective optimization[J]. Applied Intelligence, 2020, 50 (12): 4459-4481.
[41] Liu Y, Li X, Hao Q. A new constrained multi-objective optimization problems algorithm based on group-sorting[C].Proceedings of the Genetic and Evolutionary Computation Conference Companion,2019: 221-222.
[42] He C, Cheng R, Tian Y, Zhang X, Tan K C, Jin Y. Paired offspring generation for constrained large-scale multiobjective optimization[J]. IEEE Transactions on Evolutionary Computation, 2020, 25 (3): 448-462.
[43] Qian F, Xu B, Qi R, Tianfield H. Self-adaptive differential evolution algorithm with α-constrained-domination principle for constrained multi-objective optimization[J]. Soft Computing, 2012, 16 (8): 1353-1372.
[44] Tian Y, Li X, Ma H, Zhang X, Tan K C, Jin Y. Deep reinforcement learning based adaptive operator selection for evolutionary multi-objective optimization[J]. IEEE Transactions on Emerging Topics in Computational Intelligence, 2022.
[45] Yao X, Zhao Q, Gong D, Zhu S. Solution of large-scale many-objective optimization problems based on dimension reduction and solving knowledge guided evolutionary algorithm[J]. IEEE Transactions on Evolutionary Computation, 2021.
[46] Guo Y, Chen G, Jiang M, Gong D, Liang J. A Knowledge guided Transfer Strategy for Evolutionary Dynamic Multiobjective Optimization[J]. IEEE Transactions on Evolutionary Computation, 2022.
[47] Li W, Meng X, Huang Y, Mahmoodi S. Knowledge-guided multiobjective particle swarm optimization with fusion learning strategies[J]. Complex & Intelligent Systems, 2021, 7 (3): 1223-1239.
[48] Poláková R, Tvrdík J, Bujok P. Differential evolution with adaptive mechanism of population size according to current population diversity[J]. Swarm and Evolutionary Computation, 2019, 50: 100519.
[49] Yang M, Li C, Cai Z, Guan J. Differential evolution with auto-enhanced population diversity[J]. IEEE transactions on cybernetics, 2014, 45 (2): 302-315.
[50] Zuo M, Dai G, Peng L, Chen L, Chen X, Song Z. Global optimisation of multiple gravity assist spacecraft trajectories based on search space exploring and PCA[C].2016 IEEE Congress on Evolutionary Computation,2016: 2655-2660.
[51] Liang J, Qiao K, Yue C, Yu K, Qu B, Xu R, Li Z, Hu Y. A clustering-based differential evolution algorithm for solving multimodal multi-objective optimization problems[J]. Swarm and Evolutionary Computation, 2021, 60: 100788.
[52] Rousseeuw P J, Mathematics A. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis[J]. Journal of computational, 1987, 20: 53-65.
[53] Auger A, Hansen N. A restart CMA evolution strategy with increasing population size[C].2005 IEEE congress on evolutionary computation,2005: 1769-1776.
[54] Mcginley B, Maher J, O'riordan C, Morgan F. Maintaining healthy population diversity using adaptive crossover, mutation, and selection[J]. IEEE Transactions on Evolutionary Computation, 2011, 15 (5): 692-714.
[55] Zuo M, Dai G, Peng L. A new mutation operator for differential evolution algorithm[J]. Soft Computing, 2021, 25 (21): 13595-13615.
[56] Storn R, Price K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces[J]. Journal of global optimization, 1997, 11 (4): 341-359.
[57] Zhang J, Sanderson A C. JADE: adaptive differential evolution with optional external archive[J]. IEEE Transactions on evolutionary computation, 2009, 13 (5): 945-958.
[58] Wang H, Jin Y, Yao X. Diversity assessment in many-objective optimization[J]. IEEE transactions on cybernetics, 2016, 47 (6): 1510-1522.
[59] Bader J, Zitzler E. HypE: An algorithm for fast hypervolume-based many-objective optimization[J]. Evolutionary computation, 2011, 19 (1): 45-76.
[60] Fonseca C M, Paquete L, López-Ibánez M. An improved dimension-sweep algorithm for the hypervolume indicator[C].2006 IEEE international conference on evolutionary computation,2006: 1157-1163.
[61] Sharma M, Komninos A, López-Ibáñez M, Kazakov D. Deep reinforcement learning based parameter control in differential evolution[C].Proceedings of the Genetic and Evolutionary Computation Conference,2019: 709-717.
[62] Tian Y, Cheng R, Zhang X, Jin Y. PlatEMO: A MATLAB platform for evolutionary multi-objective optimization [educational forum][J]. IEEE Computational Intelligence Magazine, 2017, 12 (4): 73-87.

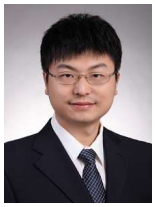
[63] Fan Z, Li W, Cai X, Huang H, Fang Y, You Y, Mo J, Wei C, Goodman E. An improved epsilon constraint-handling method in MOEA/D for CMOPs with large infeasible regions[J]. Soft Computing, 2019, 23 (23): 12491-12510.
[64] Ma Z, Wang Y. Evolutionary constrained multiobjective optimization: Test suite construction and performance comparisons[J]. IEEE Transactions on Evolutionary Computation, 2019, 23 (6): 972-986.
[65] Deb K, Thiele L, Laumanns M, Zitzler E: Scalable test problems for evolutionary multiobjective optimization,Evolutionary multiobjective optimization: Springer,2005: 105-145.
[66] Vlennet R, Fonteix C, Marc I. Multicriteria optimization using a genetic algorithm for determining a Pareto set[J]. International Journal of Systems Science, 1996, 27 (2): 255-260.
[67] Panichella A. An adaptive evolutionary algorithm based on non-Euclidean geometry for many-objective optimization[C].Proceedings of the Genetic and Evolutionary Computation Conference,2019: 595-603.
[68] Panichella A. An Improved Pareto Front Modeling Algorithm for Large-scale Many-Objective Optimization[C].The Genetic and Evolutionary Computation Conference,2022.
[69] Tian Y, Cheng R, Zhang X, Cheng F, Jin Y. An indicator-based multiobjective evolutionary algorithm with reference point adaptation for better versatility[J]. IEEE Transactions on Evolutionary Computation, 2017, 22 (4): 609-622.
[70] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. IEEE transactions on evolutionary computation, 2002, 6 (2): 182-197.
[71] Li K, Chen R, Fu G, Yao X. Two-archive evolutionary algorithm for constrained multiobjective optimization[J]. IEEE Transactions on Evolutionary Computation, 2018, 23 (2): 303-315.
[72] Nebro A J, Durillo J J, Luna F, Dorronsoro B, Alba E. Mocell: A cellular genetic algorithm for multiobjective optimization[J]. International Journal of Intelligent Systems, 2009, 24 (7): 726-746.
[73] Sun Y, Xue B, Zhang M, Yen G G. A new two-stage evolutionary algorithm for many-objective optimization[J]. IEEE Transactions on Evolutionary Computation, 2018, 23 (5): 748-761.
[74] Liu Z-Z, Wang Y. Handling constrained multiobjective optimization problems with constraints in both the decision and objective spaces[J]. IEEE Transactions on Evolutionary Computation, 2019, 23 (5): 870-884.
[75] Coello C a C, Cortés N C. Solving multiobjective optimization problems using an artificial immune system[J]. Genetic programming evolvable machines, 2005, 6 (2): 163-190.
[76] Zhang Q, Zhou A, Zhao S, Suganthan P N, Liu W, Tiwari S. Multiobjective optimization test instances for the CEC 2009 special session and competition[J]. University of Essex, Colchester, UK; Nanyang technological University, Singapore, special session on performance assessment of multi-objective optimization algorithms, technical report, 2008, 264: 1-30.

Mingcheng Zuo received the Ph.D. degree in the School of Computer Science, China University of Geosciences (Wuhan). He is currently a Lecturer in the Artificial Intelligence Research Institute, China University of Mining and Technology. His research interests include computational intelligence and its application to integrated energy system.

Dunwei Gong (SM'22) received the B.Sc. degree in mathematics from China University of Mining and Technology, Xuzhou, China, in 1992, the M.Sc. degree in automatic control theory and applications from Beihang University, Beijing, China, in 1995, and the Ph.D. degree in control theory and control engineering from China University of Mining and Technology, Xuzhou, China, in 1999, respectively. He is currently a Professor in Computational Intelligence and the Director of the Centre for Intelligent Software and Systems, School of Information Science and Technology, Qingdao University of Science and Technology, Qingdao, China. He has published over 200 journal papers, and obtained three awards issued by Ministry of Education and Jiangsu Province, China, respectively, in recent years. His research interests include computation intelligence in multi-objective optimization, dynamic and uncertain optimization, as well as applications in software engineering, integrated energy systems, big data processing and parsing.

Yan Wang received the M.Sc. degree in computer science and technology from the Anhui University, Hefei, China, in 2020. She is currently pursuing the Ph.D. degree with the School of Information and Control Engineering, China University of Mining and Technology, Xuzhou, China. Her research interests include intelligence optimization and integrated energy system operation optimization.

Xianming Ye received his BEng and MEng degrees in the Department of Automation, Wuhan University, China in 2008 and 2010, respectively. He completed his PhD degree in Electrical Engineering from the University of Pretoria in 2015. He is currently an Associate Professor and the Junior Exxaro Chair in Energy Efficiency in the Department of Electrical, Electronic and Computer Engineering, University of Pretoria, South Africa. He is also a Certified Measurement and Verification Professional. His research expertise lies in the areas of building and industry energy system modelling and optimisation, renewable energy, microgrids, P2P energy sharing, battery management systems, explainable AI, and electric vehicles. He is a Guess Editor for IEEE Transactions on Consumer Electronics, and an Associate Editor for IET Renewable Power Generation.



Bo Zeng (S'10-M'14) received the Ph.D. degree in Electrical Engineering from North China Electric Power University (NCEPU), Beijing, in 2014. He is currently an Associate Professor in the Department of Electrical and Electronic Engineering of NCEPU, Beijing. His research interests include integrated energy system optimization and power distribution system planning.

Fanlin Meng received the Ph.D. degree in the School of Computer Science, University of Manchester. He is currently a Lecturer in the Alliance Manchester Business School, University of Manchester. His research interests include computational intelligence, game theory, smart energy and mobility.