2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Offline Data-Driven Mixed-Variable Optimization Algorithm Using a Step-Wise Strategy

1st Yiteng Xu
School of Artificial Intelligence
Xidian University
Xi'an, China
eatenxu@foxmail.com

2nd Handing Wang
School of Artificial Intelligence
Xidian University
Xi'an, China
hdwang@xidian.edu.cn

***Abstract*—Some real-world engineering problems are offline data-driven mixed-variable optimization problems, which involve optimizing both continuous and discrete variables using only historical experimental data. The main challenges are handling mixed variables and utilizing surrogate models effectively. We propose a novel algorithm that uses a step-wise strategy to optimize the discrete and continuous variables in two stages. In the first stage, we use different radial basis function networks models as surrogates and a voting method to select a promising subspace of discrete variable values. In the second stage, we fix the discrete variable values and use a selective ensemble strategy to optimize the continuous variables. We test our algorithm on 30 test problems and compare it with two representative algorithms. The results show that our algorithm is superior and more stable on most problems, especially on complex multimodal problems. Our algorithm is an effective and flexible framework for handling mixed variables and improving search efficiency and quality.**

***Index Terms*—Offline data-drive, Mixed-variable optimization problems, Surrogate model, Selective ensemble method, Voting strategy**

I. Introduction

In the engineering field, mixed-variable optimization problems (MVOPs) are a common type of optimization problems [1]. In general, MVOPs can be formulated as follows:

$$y = \min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x}), \tag{1}$$

where $\boldsymbol{x} = (\boldsymbol{x_c}, \boldsymbol{x_d})$ is a mixed-variable vector, containing continuous vector $\boldsymbol{x_c} = (x_c^1, x_c^2, \ldots, x_c^{n_1})$ and discrete vector $\boldsymbol{x_d} = (x_d^1, x_d^2, \ldots, x_d^{n_2})$, n_1 is the number of continuous variables, n_2 is the number of discrete variables, $\mathcal{X} = \mathcal{X}_c \times \mathcal{X}_d$ is the search space, which is defined by the definition domain of continuous vector $\mathcal{X}_c \subseteq \mathbb{R}^{n_1}$ and the sets for values of the discrete vector $\mathcal{X}_d = \{S^1, S^2, \ldots, S^{n_2}\}$; S^i is the candidate set of x_d^i. Since the evaluation of $f(\boldsymbol{x})$ can only be performed based on limited data and no new data can be obtained during the optimization process, this poses a difficulty for evolutionary algorithms (EAs). Such problems are known as offline data-driven optimization problems [2]. A possible way to solve such problems is to use surrogate-assisted evolutionary algorithms (SAEAs) [3], which employ approximated models trained based on a small amount of data to replace the unavailable objective functions. Various machine learning models, such as artificial neural networks (ANNs) [4], polynomial regression (PR) [5], Kriging model [6], and radial basis function networks (RBFNs) [7], have been used as surrogates in SAEAs. Moreover, SAEAs use model management techniques to balance the exploration and exploitation during the optimization process.

Offline data-driven MVOPs face two main challenges: handling mixed variables and using surrogate models effectively with limited data. Various evolutionary operators have been proposed to handle mixed decision variables [1], such as discretization, relaxation, and two-partition. Discretization transforms continuous variables into discrete ones. Relaxation changes discrete variables into continuous ones and rounds them to the nearest feasible values. Two-partition separates the continuous and discrete variables and applies different operators to them. The two-partition method is better than the other two because it preserves the accuracy of the decision variables and avoids invalid search. However, handling mixed variables in surrogate modeling is also difficult. Some methods have been developed to deal with this problem. For example, Merch'an et al. [8] convert the discrete input variables into integers or one-hot vectors; Bartz-Beielstein et al. [9] use different distances in the kernel function; Kim et al. [10] build a tree-based model that can handle mixed variables naturally.

To manage the surrogate models effectively in offline data-driven optimization, multiple models are employed to overcome the challenges of limited data and inaccurate approximation. For example, Wang et al. [11] propose a selective surrogate ensemble method, which constructs many surrogate models before optimization and selects a small diverse subset of them adaptively during the optimization for the best local approximation accuracy. A method that uses three RBFNs as surrogate models and selects candidate solutions with high-confidence fitness predictions to enrich the training data is proposed by Huang et al. [12] to overcome the lack of training data. Similarly, Huang et al. [13] develop a stochastic ranking-based ensemble of four RBFNs with different kernels to handle high-dimensional offline data-driven optimization problems.

To tackle the challenges posed by offline data-driven MVOPs, we propose a novel offline data-driven mixed-variable optimization algorithm using a step-wise strategy (DDEA-

This work was supported in part by the National Natural Science Foundation of China (No. 61976165)

SW), which optimizes the discrete and continuous variables in two stages. The main contributions of this paper can be summarized as follows:

1) We use a step-wise strategy to perform a global search for discrete variables and a local search for continuous variables sequentially. In the global search stage, a voting method is used to determine the best discrete variable values. In the local search stage, the best discrete variable values are fixed and a selective ensemble strategy based on ranking and grouping is used to optimize the continuous variables. This strategy can avoid being trapped in local optimal in the discrete variable space and balance the exploration and exploitation of the search space.
2) We propose a voting method to select a consensus subspace of discrete variable values from multiple candidate solutions generated by different RBFN models, which are surrogate models trained with different center vectors. This method can help us find the promising regions of interest in the discrete variable space.

The remainder of this paper is organized as follows. In section II, the related techniques, such as the radial basis function network based on Gower-distance and two-partition evolutionary algorithm are introduced. A comprehensive description of the proposed algorithm framework is provided in Section III. Section IV will present the experiment settings, as well as the design of comparative experiments studies and result analysis. At last, Section V concludes the study and discusses future work.

II. Radial basis function network based on Gower-distance

RBFN based on the Gower Distance is a method to approximate functions with mixed variables. Given the initial offline data consists of N sample points $\boldsymbol{X} = (\boldsymbol{x^1}, \boldsymbol{x^2}, \ldots, \boldsymbol{x^N})^T$, whose true value $\boldsymbol{f} = \left(f\left(\boldsymbol{x^1}\right), f\left(\boldsymbol{x^2}\right), \ldots, f\left(\boldsymbol{x^N}\right)\right)^T$; and the center vector $\boldsymbol{c} = (\boldsymbol{c^1}, \boldsymbol{c^2}, ..., \boldsymbol{c^m})$ can be obtained by the K-means clustering. RBFN can be expressed as $\hat{f}(\boldsymbol{x}) = w_0 + \sum_{i=1}^{m} w_i \varphi(\boldsymbol{x}; \boldsymbol{c}^i)$, where $\varphi(\boldsymbol{x}; \boldsymbol{c}^i)$ is kernel function to map the distance between x and i-th center vector $\boldsymbol{c}^i$; each $\boldsymbol{c^i} = (\boldsymbol{c^i_{con}}, \boldsymbol{c^i_{dis}})$ is a mixed-variable vector that contains continuous and discrete components; and $\boldsymbol{W} = (w_0, w_1, \ldots, w_m)^T$ is the weight vector and can be calculated by pseudo-inverse method [14].

When the input variables are mixed, we can use the Gower distance as the kernel function of the RBFN. The Gower distance between the decision vector $\boldsymbol{x}$ and i-th center $\boldsymbol{c^i}$ could be expressed [15]:

$$d\left(\boldsymbol{x}, \boldsymbol{c^i}\right) = \frac{1}{n_1 + n_2}\left(\sum_{j=1}^{n_1} \frac{\left|x_{con_j} - c^i_{con_j}\right|}{\Delta x_{con_j}} + \sum_{j=1}^{n_2} S\left(x_{dis_j}, c^i_{dis_j}\right)\right), \quad (2)$$

where n_1 is the number of continuous variables; n_2 is the number of discrete variables; Δx_{con_j} is range of the j-th continuous variable; and the $S(x_{dis_j}, c^i_{dis_j})$ is defined as:

$$S\left(x_{dis_j}, c^i_{dis_j}\right) = \begin{cases} 0, & \text{if } x_{dis_j} = c^i_{dis_j} \\ 1, & \text{otherwise} \end{cases}.$$

Furthermore, we use the Gaussian kernel as the basis function for the RBFN, whose hyper-parameter β is determined by several factors, such as n, the maximal distance between sampling points $D_{\max}$, the dimension of the decision vector d, and the number of training data N [16]. The Gaussian Kernel is expressed as $\varphi(\boldsymbol{x}; \boldsymbol{c}^i) = -\exp\left(\frac{d(\boldsymbol{x};\boldsymbol{c}^i)}{\beta}\right)$, where $\beta = 2^{4\cdot(n-1)} \cdot D_{\max} \cdot (dN)^{-\frac{1}{d}}$. In our work, we use β with $n = 4$ for the Gaussian kernel, and RBFN based on the Gower distance as the surrogate model.

III. Proposed Algorithm

We propose an offline data-driven mixed-variable optimization algorithm using a step-wise strategy to optimize the mixed variables. Our method consists of three modules: initialization module, global search module and local search module. The initialization module trains different RBFN models for the given offline data. The global search module produces a number of candidate solutions and selects a promising subspace of discrete variables from the candidate solutions by a majority voting method. The local search module refines the continuous variable with a selective ensemble surrogate. The overall framework of the algorithm is shown in Fig. 1.

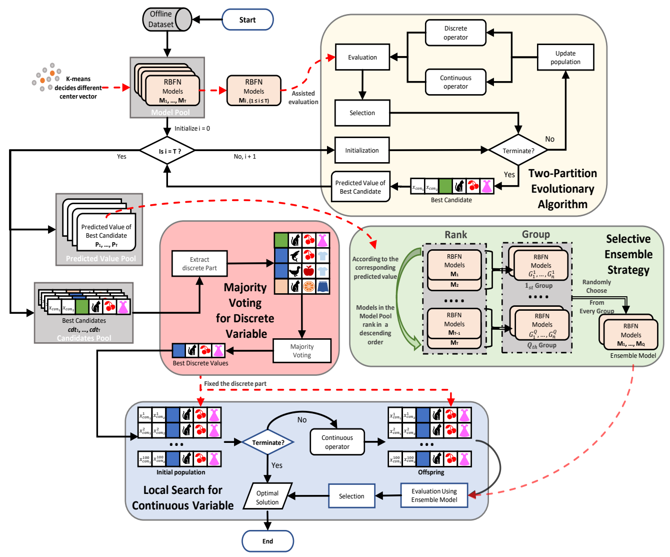
Offline Dataset
Start
K-means decides different center vector
RBFN Models M1, ..., MT
Model Pool
RBFN Models Mi (1≤i≤T)
Assisted evaluation
Evaluation
Discrete operator
Continuous operator
Update population
Selection
Initialization
Terminate?
No
Yes
Two-Partition Evolutionary Algorithm
Predicted Value of Best Candidate
Best Candidate
Initialize i = 0
Is i = T ?
No, i + 1
Predicted Value of Best Candidate P1, ..., PT
Predicted Value Pool
Best Candidates cdt1, ..., cdtT
Candidates Pool
Extract discrete Part
Majority Voting for Discrete Variable
Majority Voting
Best Discrete Values
Rank
Group
Selective Ensemble Strategy
According to the corresponding predicted value
Models in the Model Pool rank in a descending order
Randomly Choose From Every Group
Ensemble Model
Fixed the discrete part
Initial population
Offspring
Optimal Solution
Evaluation Using Ensemble Model
Local Search for Continuous Variable
End

Fig. 1: The Framework of the Proposed Algorithm

The main steps of our algorithm are as follows:

1) **Initialization:** Given the offline dataset, the proposed algorithm builds a set of T RBFN models (denoted by $M_1, ..., M_T$). Each RBFN model is trained on the whole offline dataset, but their diversity is ensured by using different random RBFN clustering centers.

2) **Global search module:** This module consists of two sub-modules, namely candidate solution generation and majority voting. In the candidate solution generation sub-module, a single RBFN model assisted two-partition evolutionary algorithm is used to obtain the best candidate solution (denoted by $C_1, ..., C_T$) for each base learner. These candidate solutions are stored in a candidate pool and their predicted values (denoted by $P_1, ..., P_T$) are also recorded. In the majority voting sub-module, the proposed algorithm extracts the best discrete variable values (denoted by $\boldsymbol{B} = \{b_1, b_2, ..., b_{n_2}\}$) for the mixed-variable solutions in the candidate pool by a majority voting method.
3) **Local search module:** The proposed algorithm ranks and divides all the base learners into Q groups according to their predicted values in the predicted value pool. Then, it randomly chooses one base learner from each group to form an ensemble model (denoted by $M_1, ..., M_Q$). Then, the ensemble model assisted EA is run again in the continuous subspace with fixed discrete part obtained in the previous step. The EA stops when the termination condition is reached and outputs the optimal solution.

A. Majority Voting Strategy for Promising Subspace

The goal of the majority voting strategy is to find a promising subspace of discrete variable that can improve the algorithm performance. A promising subspace is a subset of discrete values that are most common among the best candidate solutions. By selecting a promising subspace, we can simplify the search space and focus on the optimal regions [17].

Our method is to use majority voting to identify the promising subspace from candidate solutions generated by different RBFN models. The majority voting counts the frequency of each discrete value for every position of the discrete variable among all candidate solutions ($\boldsymbol{cdt_1}, \ldots, \boldsymbol{cdt_T}$). Then, the most frequent value ($b_1, \ldots, b_{n_2}$) is selected for that position. This approach uses the diversity and quality of the candidate solutions to find the common features of the best solutions and guide the search process.

To implement the majority voting strategy, we use RBFN models solved by independently run EA to obtain the best candidate solutions. For each position of the discrete variable, we extract and count the corresponding discrete values of all the best candidate solutions. Then, we select the most frequent value as the value of that position. We repeat this for all positions of the discrete variable, resulting in a promising subspace of discrete variable values. The details of the majority voting strategy are in Algorithm 1.

B. Local Search for Continuous Variables

The local search for continuous variables aims to refine the solutions obtained by the global search by further optimizing the continuous variables.

Algorithm 1 Majority Voting Strategy for Discrete Variable

```
Input: A set of candidate solutions $\boldsymbol{cdt} = \{\boldsymbol{cdt_1}, \boldsymbol{cdt_2}, ..., \boldsymbol{cdt_T}\}$, where each candidate solution $\boldsymbol{cdt_i}$ has a discrete variable part $\boldsymbol{D_i} = \{d_{i1}, d_{i2}, ..., d_{in_2}\}$, $1 \le i \le T$ and $1 \le j \le n_2$.
Output: The best discrete values $\boldsymbol{B} = \{b_1, b_2, ..., b_{n_2}\}$, where $b_j$ is the value of the $j$-th position of the discrete variable that has the highest frequency in $\boldsymbol{D_{1:n_2}}$.
1: $\boldsymbol{B} \leftarrow$ an empty list of size $n_2$;
2: for $j \leftarrow 1$ to $n_2$ do
3:     $F \leftarrow$ a frequency list for the $j$-th position of the discrete variable;
4:     for $i \leftarrow 1$ to $T$ do
5:         $F_{d_{ij}} \leftarrow F_{d_{ij}} + 1$;
6:     end for
7:     $b_j \leftarrow$ argmax$(F_{d_{ij}})$;
8:     $B_j \leftarrow b_j$;
9: end for
10: return $\boldsymbol{B}$;
```

To perform the local search, we build an ensemble model that can provide more accurate and robust predictions than a single RBFN model [18]. The ensemble model is constructed by a selective ensemble strategy that can balance the global ensemble accuracy and the fitting accuracy of each RBFN model. Given the offline dataset, we use T RBFN models independently with different clustering centers. These RBFN models (base learners) are fixed and form the model pool. However, not all base learners are equally useful for the ensemble model. Some base learners may have similar or average performance, which may reduce the diversity and effectiveness of the ensemble. Therefore, we select Q models from T models by ranking and grouping them (denoted by $M_1, \ldots, M_T$) according to their predicted values ($P_1, \ldots, P_T$) of the best candidates, and randomly selecting one model from each group. This way, we can ensure that the selected models have different fitting accuracy and capture different aspects of the data. To elaborate on the selective ensemble strategy, we first sort the T RBFN models in descending order based on their predicted values. Then, we divide the sorted models into Q groups, each containing n models. For example, the first group contains the first to the n_{th} models in the sorted order (denoted by $G_1^1, \ldots, G_1^n$), and the last group contains the last n models (denoted by $G_Q^1, \ldots, G_Q^n$). Then, we randomly choose one model from each group and repeat this process Q times. This way, we obtain Q models that form the ensemble model ($M_1, \ldots, M_Q$).

After building the ensemble model, the proposed algorithm runs the EA to optimize the continuous variables. The discrete part is fixed by the best discrete variable values obtained by the majority voting strategy, while the continuous part is manipulated by simulated binary crossover and polynomial mutation. The fitness value of each candidate solution is evaluated by using the ensemble model. The EA stops when the maximum number of iterations is exceeded and outputs the optimal solution.

IV. Experimental Studies

A. Test Problems and Parameters Settings

1) Test Problems: To comprehensively evaluate the performance of our proposed algorithm, we adopted 30 artificial test problems F1-F30 that were proposed by Liu et al. [19].

In our experiments, the Latin-hypercube sampling (LHS) method is modified to encode the discrete variables as integers and generates initial offline data for both continuous and discrete variables, and all algorithms start with 100 initial samples generated by LHS [20].

2) Parameter settings: In our experiments, RBFN consists of a number of centers, which is set to half of the training sample size $[N/2]$, where N represents the total number of training samples. And we use 100 generations for each base learner to generate candidate solutions. The optimization process for all the compared algorithms involves both continuous and discrete operators. The continuous operator employs the simulated binary crossover (SBX) operator with a distribution index $\eta = 1$, and the polynomial mutation operator with $\eta = 1$. On the other hand, the discrete operator utilizes the two-point crossover and random mutation techniques. The population size is set to 100, and the termination condition is met after 100 generations. The probabilities for crossover and mutation are set to 0.9 and 0.1, respectively. These parameters are fixed for all the algorithms in the following experiments.

To assess the performance and stability, each problem is tested through 20 independent runs and the best objective values obtained in each run are recorded. These values are evaluated by the true function $f(\boldsymbol{x})$, but the evaluation of true function is not used in the optimization process. The performance indicators are the mean and standard deviation of these values. To compare the algorithms in a statistically meaningful way, Wilcoxon's rank-sum tests are conducted with a significance level of $\alpha = 0.05$ [21]. The symbols $+$, $=$, and $-$ indicate that the proposed algorithm has a significantly better, similar, or substantially worse performance than the algorithm being compared, respectively.

B. Parameter Analysis

The number of base learners T affects the efficiency and accuracy of the algorithm. The number of base learners determines how many surrogate models can be used in the optimization process. A larger T may increase the diversity and reliability of the surrogate models, but also increase the computational cost and complexity of the algorithm.

We conduct this experiment to analyze the sensitivity of this parameter and find a suitable value, as it affects both the global search stage and local search stage of the algorithm. We test the algorithm with different numbers of base learners ($T \in 10, 20, 30, 50, 70, 100, 150, 200$) on two test functions (F27 and F29), which both contain 5 continuous variables and 5 discrete variables. F27 and F29 represent complex multimodal and unimodal problems, respectively. We run each T for 20 independent experiments and record the best objective values obtained in each experiment. The experimental results are shown in Fig. 2, which shows the mean and standard deviation of the best objective values under different numbers of base learners. From Fig. 2, we can see that when $T = 50$, the algorithm achieves optimal performance and stability on both F27 and F29, with the lowest mean objective values and relatively small standard deviations. When $T < 50$, the algorithm lacks global search ability and may not find a promising subspace of discrete variable values. When $T > 50$, the selective ensemble surrogate is more likely to randomly choose inferior base learners from each group, which can impact the quality of the ensemble model and the optimization of continuous variables. Therefore, $T = 50$ is a suitable value that can balance the global search and local search effects.

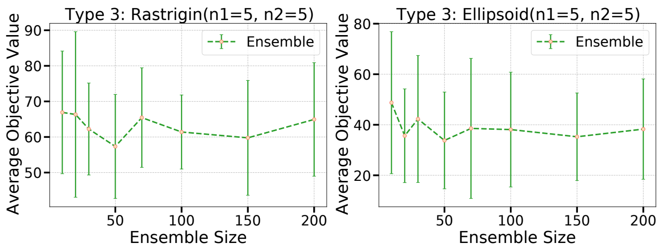

Fig. 2: Average best objective value obtained by DDEA-SW with different numbers of base learners on F27 and F29

C. Effects of Global Search Module

To investigate the effect of the global search module, we design a variant algorithm DDEA-SW(no GS) by removing the global search module from DDEA-SW. The global search module uses a voting method to select a promising subspace of discrete variables from the candidate solutions. The local search module uses a selective ensemble strategy based on ranking and grouping to optimize the continuous variables. DDEA-SW(no GS) only retains the local search module and does not fix the discrete variables. The DDEA-SW(no GS) will utilize the selective ensemble surrogate-assisted two-partition EA to evaluate the population.

We select eight test functions with different proportions of continuous and discrete variables, as well as different modality properties. These test functions are F16, F17, F19, F20, F26, F27, F29, and F30, which are derived from four continuous functions. F16, F19, F26 and F29 are unimodal functions, while F17, F20, F27 and F30 are multimodal functions. The purpose of this selection is to test the performance of the voting method in different situations.

The experimental results are shown in Table I, which presents the mean and standard deviation of the best objective values obtained by DDEA-SW and DDEA-SW(no GS) on different test functions. From Table I, we can see that DDEA-SW outperforms DDEA-SW(no GS) on four test functions (F16, F19, F20, and F30), especially on multimodal functions. On the other four test functions (F17, F26, F27, and F29), there is no significant difference between DDEA-SW and

TABLE I: Average best results (shown as mean±std) obtained by DDEA-SW and DDEA-SW (no GS) on 8 test problems, and the best results are highlighted in bold

Test Function	DDEA-SW	DDEA-SW (no GS)
F16	**1692.25±1203.18**	3624.13±2441.89 (+)
F17	30.67±10.35	34.96±15.53 (=)
F19	**28.08±21.59**	83.17±59.35 (+)
F20	**22.68±19.26**	41.11±27.43 (+)
F26	2406.36±1144.00	2678.96±855.05 (=)
F27	58.95±17.06	67.60±13.22 (=)
F29	38.97±17.41	51.93±29.61 (=)
F30	**14.23±7.88**	27.89±12.74 (+)
$+/=/$		4/4/0

DDEA-SW(no GS). This result indicates that the global search module is an effective component to improve the performance of the algorithm on some problems, especially those with complex multimodal landscapes. The global search module can help the algorithm find promising subspaces of discrete variable values by using the voting method. Without this module, the algorithm may fall into local optima or miss some potential solutions. The global search module is more useful for problems with more discrete variables or more multimodalities. For problems with fewer discrete variables or unimodality, the global search module may not have much effect.

D. *Comparative Experiment*

To further examine the performance of DDEA-SW, we compare it with SADEmv [1] and DDEA-SEmv [11] on 30 test functions (F1-F30) with different types and proportions of mixed-variables. Both algorithms use the same stopping criterion as the original papers. The main characteristics of the compared algorithms are shown below:

1) SADEmv: A single surrogate model (RBFN based on the Gower distance) assisted two-partition evolutionary algorithm for optimizing mixed-variables.
2) DDEA-SEmv: A selective surrogate ensemble method that uses RBFN based on the Gower distance as the base learners and two-partition evolutionary algorithm as the search method for handling mixed variables. It uses bootstrap sampling [22] for model training, and the number of base learners is fixed at 50 in the optimization process.

We use the same general settings of initialization and stop after 100 generations. We use the default values of the specific parameters for the compared algorithms as given in above. The average best objective value obtained by DDEA-SW and two compared algorithms over 20 independent runs on 30 test functions are shown in Table II. From the results, we can see that DDEA-SW performs better on most of problems, except F3, F8 and F23. From the results in Table II, it can be seen that DDEA-SW obtains the best performance with 22 best results on 30 test functions, followed by DDEA-SEmv with 3 best results. It is important to optimize discrete variables as well as continuous variables on these test functions. This is why DDEA-SW has superior performance to other compared mixed-variable optimization algorithms on many of these test functions, especially on multimodal problems.

We also plot the convergence curves of DDEA-SW, SADEmv and DDEA-SEmv on four test functions (F21, F25, F26 and F29) in Fig. 3. The convergence curves are the average of 20 independent runs for each algorithm. We can see that DDEA-SW converges faster and better than SADEmv and DDEA-SEmv on these test functions. This result shows that DDEA-SW can find promising solutions in the early stage of optimization by using the voting method to determine the best subspace of discrete variable values. This method can reduce the search space and improve the efficiency of optimization. Moreover, DDEA-SW can further improve the solutions in the later stage of optimization by using the selective ensemble strategy to optimize the continuous variables. This strategy can balance the exploration and exploitation of the search space, and avoid falling into local optima.

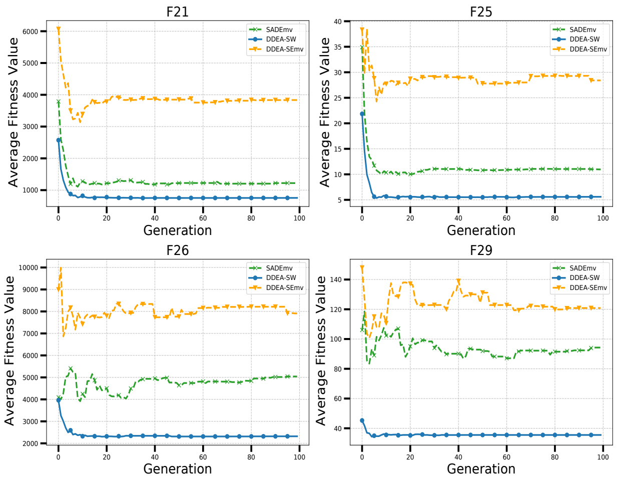

Fig. 3: Convergence profiles of offline DDEAs on F21, F25, F26 and F29

Therefore, we can conclude that DDEA-SW is a superior algorithm for optimizing mixed variables, and can achieve better performance than SADEmv and DDEA-SEmv on most test functions. The stepwise optimization algorithm framework is an effective and flexible framework for handling mixed variables.

V. Conclusions

In this paper, we address the problem of offline data-driven mixed-variable optimization, which is a common and challenging problem in engineering fields. The problem involves optimizing both continuous and discrete variables with limited data, which poses difficulties for handling mixed variables and building reliable surrogate models. We propose a novel algorithm that uses a step-wise strategy to optimize the discrete and continuous variables separately. Our algorithm consists of two stages: a global search stage and a local search stage. The global search stage aims to find a promising subspace of discrete variable values by using multiple RBFN models and a voting method. The local search stage aims to optimize the continuous variables by using a selective ensemble strategy that combines different base learners. We evaluate our algorithm on 30 artificial test problems with different types and proportions of mixed variables, as well as different modality properties. We compare our algorithm with two representative algorithms, namely SADEmv and DDEA-SEmv. The experimental results demonstrate that our algorithm outperforms the compared algorithms on most test problems, especially on complex multimodal problems. The convergence curves also indicate that our algorithm can converge faster and better than the compared algorithms.

TABLE II: Average (shown as mean±std) obtained optimum by DDEA-SW and two compared algorithms on all 30 test problems, and the best results are highlighted in bold.

Test Function	DDEA-SW	SADEmv	DDEA-SEmv
F1	**1010±286**	1675±573 (+)	2526±859 (+)
F2	85.88±24.65	90.90±18.98 (=)	90.18±17.49 (=)
F3	16.57±1.40	17.19±1.79 (=)	**14.26±1.82** (-)
F4	**19.65±6.51**	26.78±7.95 (+)	55.11±31.05 (+)
F5	**8.28±3.08**	15.82±7.54 (+)	30.32±10.43 (+)
F6	**1408±508**	2401±832 (+)	3331±1045 (+)
F7	88.94±19.26	101.44±22.81 (=)	93.12±21.84 (=)
F8	16.81±1.25	16.81±1.73 (=)	**13.56±1.63** (-)
F9	**31.52±10.40**	57.27±22.19 (+)	83.57±39.45 (+)
F10	**13.52±5.82**	24.79±12.74 (+)	34.88±9.59 (+)
F11	**83±68**	292±233 (+)	904±540 (+)
F12	**13.26±9.67**	18.05±7.18 (+)	32.27±17.53 (+)
F13	10.11±2.17	10.66±2.15 (=)	9.74±2.33 (=)
F14	**0.99±0.82**	3.11±1.93 (+)	39.88±35.69 (+)
F15	**2.06±0.78**	4.61±1.63 (+)	34.08±18.34 (+)
F16	**1710±1258**	4837±3929 (+)	5230±3135 (+)
F17	**25.31±17.10**	48.22±18.49 (+)	50.90±23.57 (+)
F18	**7.91±3.29**	10.97±2.86 (+)	11.47±3.48 (+)
F19	**35.93±29.95**	133.28±100.38 (+)	131.54±111.27 (+)
F20	**23.79±21.35**	67.18±54.09 (+)	78.58±50.48 (+)
F21	**755±378**	1220±565 (+)	3834±1499 (+)
F22	54.50±12.66	56.34±15.98 (=)	66.94±15.70 (+)
F23	13.50±2.08	14.27±2.20 (=)	**11.79±2.29** (-)
F24	**8.18±2.57**	13.25±5.13 (+)	39.77±20.60 (+)
F25	**5.60±2.09**	10.97±6.25 (+)	28.40±11.49 (+)
F26	**2319±1036**	5040±2858 (+)	7910±3026 (+)
F27	66.27±20.22	71.19±16.09 (=)	76.59±23.66 (=)
F28	11.45±2.30	13.63±1.87 (+)	11.90±2.68 (=)
F29	**35.51±20.37**	94.27±47.38 (+)	120.83±76.04 (+)
F30	**18.54±7.50**	42.19±19.56 (+)	51.98±32.78 (+)
$+/=/$		22/8/0	22/5/3

For the future work, we plan to extend our algorithm in several aspects. For example, we can explore other machine learning methods to further refine the continuous variable search part, such as using multiple surrogate models and selecting candidate solutions with high-confidence fitness predictions, which can enrich the training data and improve the approximation accuracy. Some adaptive methods also can be used to adjust the number or the type of base learners according to the problem characteristics, which can enhance the diversity and reliability of the surrogate models. We can also test our algorithm on some engineering applications to demonstrate its applicability.

References

[1] Y. Liu and H. Wang, "Surrogate-assisted hybrid evolutionary algorithm with local estimation of distribution for expensive mixed-variable optimization problems," *Applied Soft Computing*, vol. 133, p. 109957, 2023.

[2] Y. Jin, H. Wang, T. Chugh, D. Guo, and K. Miettinen, "Data-driven evolutionary optimization: An overview and case studies," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 3, pp. 442–458, 2018.

[3] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 61–70, 2011.

[4] Y. Jin, M. Olhofer, and B. Sendhoff, "A framework for evolutionary optimization with approximate fitness functions," *IEEE Transactions on evolutionary computation*, vol. 6, no. 5, pp. 481–494, 2002.

[5] Z. Zhou, Y. S. Ong, M. H. Nguyen, and D. Lim, "A study on polynomial regression and gaussian process global surrogate model in hierarchical surrogate-assisted evolutionary algorithm," in *2005 IEEE congress on evolutionary computation*, vol. 3, pp. 2832–2839, IEEE, 2005.

[6] T. Chugh, Y. Jin, K. Miettinen, J. Hakanen, and K. Sindhya, "A surrogate-assisted reference vector guided evolutionary algorithm for computationally expensive many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 129–142, 2016.

[7] C. Sun, Y. Jin, R. Cheng, J. Ding, and J. Zeng, "Surrogate-assisted cooperative swarm optimization of high-dimensional expensive problems," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 4, pp. 644–660, 2017.

[8] E. C. Garrido-Merchán and D. Hernández-Lobato, "Dealing with categorical and integer-valued variables in bayesian optimization with gaussian processes," *Neurocomputing*, vol. 380, pp. 20–35, 2020.

[9] T. Bartz-Beielstein and M. Zaefferer, "Model-based methods for continuous and discrete global optimization," *Applied Soft Computing*, vol. 55, pp. 154–167, 2017.

[10] K. Kim and J.-s. Hong, "A hybrid decision tree algorithm for mixed numeric and categorical data in regression analysis," *Pattern Recognition Letters*, vol. 98, pp. 39–45, 2017.

[11] H. Wang, Y. Jin, C. Sun, and J. Doherty, "Offline data-driven evolutionary optimization using selective surrogate ensembles," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 2, pp. 203–216, 2018.

[12] P. Huang, H. Wang, and Y. Jin, "Offline data-driven evolutionary optimization based on tri-training," *Swarm and evolutionary computation*, vol. 60, p. 100800, 2021.

[13] P. Huang, H. Wang, and W. Ma, "Stochastic ranking for offline data-driven evolutionary optimization using radial basis function networks with multiple kernels," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 2050–2057, IEEE, 2019.

[14] G. Li, Q. Zhang, Q. Lin, and W. Gao, "A three-level radial basis function method for expensive optimization," *IEEE Transactions on Cybernetics*, vol. 52, no. 7, pp. 5720–5731, 2021.

[15] M. D'Orazio, "Distances with mixed type variables some modified gower's coefficients," *arXiv preprint arXiv:2101.02481*, 2021.

[16] H. Yu, Y. Tan, J. Zeng, C. Sun, and Y. Jin, "Surrogate-assisted hierarchical particle swarm optimization," *Information Sciences*, vol. 454, pp. 59–72, 2018.

[17] S. Mahdavi and S. Rahnamayan, "Enhancing discrete differential evolution by conducting election," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–7, IEEE, 2017.

[18] N. García-Pedrajas, C. Hervás-Martínez, and D. Ortiz-Boyer, "Cooperative coevolution of artificial neural network ensembles for pattern classification," *IEEE transactions on evolutionary computation*, vol. 9, no. 3, pp. 271–302, 2005.

[19] J. Liu, Y. Wang, G. Sun, and T. Pang, "Multisurrogate-assisted ant colony optimization for expensive optimization problems with continuous and categorical variables," *IEEE Transactions on Cybernetics*, vol. 52, no. 11, pp. 11348–11361, 2021.

[20] A. Afzal, K.-Y. Kim, and J.-w. Seo, "Effects of latin hypercube sampling on surrogate modeling and optimization," *International Journal of Fluid Machinery and Systems*, vol. 10, no. 3, pp. 240–253, 2017.

[21] J. Carrasco, S. García, M. Rueda, S. Das, and F. Herrera, "Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review," *Swarm and Evolutionary Computation*, vol. 54, p. 100665, 2020.

[22] B. Efron and R. J. Tibshirani, *An introduction to the bootstrap*. CRC press, 1994.