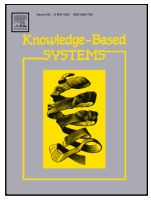
Knowledge-Based Systems 331 (2025) 114764

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys



Dynamic multi-knowledge evolutionary algorithm for sparse large-scale multi-objective optimization

Lidan Bai [a], Jun Sun [a,*], Chao Li [a], Hengyang Lu [a], Vasile Palade [b]

[a] Jiangsu Provincial Engineering Laboratory of Pattern Recognition and Computational Intelligence, Wuxi, 214122, Jiangsu, China
[b] Centre for Computational Science and Mathematical Modelling, Coventry, CV1 2TL, West Midlands, UK

ARTICLE INFO

Keywords:
Large-scale multi-objective optimization
Sparse optimization
Knowledge-guided evolutionary algorithm
Real-world sparse problems

ABSTRACT

Solving large-scale multi-objective optimization problems with sparse Pareto fronts is challenging due to the high dimensionality of the decision space and the difficulty of identifying relevant variables under sparse optimality. This work proposes a knowledge-guided evolutionary algorithm that integrates a multi-interval sampling-based initialization for estimating variable importance, two complementary binary variation strategies guided by different knowledge vectors, and an adaptive real-valued operator for refining selected variables. A probabilistic control mechanism dynamically adjusts operator usage over time, enabling a smooth balance between exploration and exploitation throughout the search process. Experimental results on eight sparse benchmark problems (SMOP1–SMOP8 [1]) show that the proposed algorithm achieves the best performance in 23 out of 24 cases under high sparsity ($\theta = 0.01$), with up to 15.5 % improvement in the inverted generational distance over the second-best algorithm. On two real-world problems—sparse neural network training and sparse signal reconstruction—it achieves the highest hypervolume score on the sparse signal reconstruction problem with statistical significance, and performs competitively on the sparse neural network training problem, outperforming eight of the eleven baselines.

1. Introduction

Large-Scale Multi-Objective Optimization Problems (LMOPs) frequently arise in domains such as transportation, finance, engineering, and also in deep neural network training [2–5]. These problems often involve 200 or more decision variables [6], which result in a high-dimensional and complex decision space, as well as multiple conflicting objectives that further complicate the search process by requiring simultaneous optimization over multiple trade-off fronts. As a result, conventional Multi-Objective Evolutionary Algorithms (MOEAs) [7,8] face difficulties in maintaining diversity and avoiding premature convergence, and also incur high computational costs in large-scale scenarios [9,10].

To address these challenges, several approaches have been developed, including divide-and-conquer techniques [11,12], decision space reduction methods [13,14], problem reformulation [15,16], and enhanced search-based evolutionary algorithms [17–19]. Despite their effectiveness in general large-scale optimization, these methods often struggle with Sparse Large-Scale Multi-Objective Optimization Problems (SLMOPs)—a specialized class of problems where Pareto optimal solutions are inherently sparse, with only a critical subset of decision variables contributing significantly to the objectives. Such sparsity commonly arises in practical applications such as feature selection [20], pattern mining [21], neural network training [22], and portfolio optimization [23]. In these sparse scenarios, effective optimization depends on the algorithm's ability to accurately identify influential variables and to focus computational resources on their exploitation. However, conventional MOEAs often lack this capability, and thus lead to suboptimal performance [1].

Recent efforts have explored sparsity-aware algorithms, including two-layer encoding [1], statistical guidance [24], and knowledge fusion strategies [25]. However, most existing methods suffer from one or more of the following limitations: (i) limited integration of real-valued information to guide the search–although Multi-Stage Knowledge-Guided Evolutionary Algorithm (MSKEA) [25] fuses binary and real-valued knowledge via a multi-stage mechanism, its stage schedule is fixed in design and lacks the flexibility to adapt over time; (ii) static control parameters for real-valued variation operators–such as fixed distribution indices in Simulated Binary Crossover (SBX) and Polynomial Mutation (PM), which determine the magnitude of perturbations. Fixed values limit the ability to apply large perturbations in early stages and smaller

* Corresponding author.
E-mail addresses: 7201905001@stu.jiangnan.edu.cn (L. Bai), junsun@jiangnan.edu.cn (J. Sun), chaoli@jiangnan.edu.cn (C. Li), luhengyang@jiangnan.edu.cn (H. Lu), ab5839@coventry.ac.uk (V. Palade).

https://doi.org/10.1016/j.knosys.2025.114764
Received 15 April 2025; Received in revised form 21 October 2025; Accepted 24 October 2025
Available online 8 November 2025

refinements later on; (iii) inefficient coordination between binary and real-valued components, leading to wasted resources on variables that are inactive (i.e., those with binary flags set to zero); (iv) limited evaluation under extreme sparsity levels (e.g., $\theta \leq 0.01$), which frequently occur in real-world applications such as gene knockout analysis [26].

In response to these limitations, we propose Dynamic Multi-Knowledge Evolutionary Algorithm (DMKEA), a novel evolutionary algorithm designed for SLMOPs. DMKEA integrates continuous probability-driven knowledge fusion, adaptive binary mutation operators, and dynamic real-valued mutation strategies. Our key contributions are as follows:

- We propose a novel probability-driven mechanism that continuously adapts the usage of multiple knowledge sources (prior vectors, filter guidance vectors, and sparsity distributions), thereby offering greater flexibility compared to fixed-stage approaches.
- We introduce an adaptive binary mutation strategy that gradually shifts from prior-based to filter-based guidance, improving convergence stability and reducing premature convergence.
- We design a real-valued mutation strategy that selectively mutates active variables and dynamically adjusts parameters to balance exploration and exploitation.
- We conduct comprehensive benchmarking on synthetic problems with extreme sparsity (as low as $\theta = 0.01$) and real-world tasks, demonstrating the robustness and scalability of DMKEA.

The remainder of this paper is organized as follows. Section 2 presents the necessary preliminaries and reviews state-of-the-art MOEAs for SLMOPs. Section 3 details the proposed DMKEA. Section 4 reports experimental results for benchmark problems and real-world case studies, followed by additional analyses including runtime evaluation, parameter sensitivity, ablation studies, and computational complexity. Finally, Section 5 offers conclusions and discusses future work.

2. Preliminaries and related work

SLMOPs present unique challenges due to their high-dimensional search spaces and the need for effective sparsity control to identify meaningful solutions among a vast number of variables. Building on these concepts, this section introduces the core mathematical background and relevant methods that provide the foundation for our approach.

2.1. Sparse multi-objective optimization

Without loss of generality, an unconstrained Multi-Objective Optimization Problem (MOP) can be mathematically formulated as follows:

$$\min_{\mathbf{x}} \quad \mathbf{f}(\mathbf{x}) = \big(f_1(\mathbf{x}), \ldots, f_M(\mathbf{x})\big), \quad \text{s.t.} \quad \mathbf{x} \in \Omega, \tag{1}$$

where $\mathbf{x} = (x_1, \ldots, x_D) \in \Omega$ is a solution consisting of D decision variables. The decision space Ω is defined as $\prod_{i=1}^{D} [\text{lower}_i, \text{upper}_i]$ with lower_i and upper_i being the lower and upper bounds for each variable x_i, respectively. The objective function vector $\mathbf{f} : \Omega \to \Lambda \subseteq \mathbb{R}^M$ maps Ω to the objective space Λ, comprising M objectives.

A solution $\mathbf{x}$ is said to dominate another solution $\mathbf{y}$, denoted as $\mathbf{x} \prec \mathbf{y}$, if:

$$(\forall i \in \{1, \ldots, M\},\ f_i(\mathbf{x}) \leq f_i(\mathbf{y})) \wedge (\exists j \in \{1, \ldots, M\},\ f_j(\mathbf{x}) < f_j(\mathbf{y})). \tag{2}$$

The objective of solving (1) is to identify a set of Pareto optimal solutions that cannot be dominated by any other solutions within the decision space Ω.

The problem is classified as a LMOP if $D > 200$ [6] and as a SLMOP when most decision variables in the Pareto optimal solutions are zero, highlighting a sparse solution space. Despite the success of MOEAs in general MOPs [7,8], their performance suffers on LMOPs due to the "curse of dimensionality," where search space complexity increases exponentially with D [9,10].

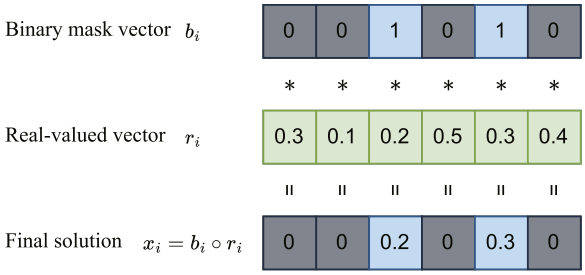


Fig. 1. Illustration of the two-layer solution representation, where $\circ$ denotes element-wise multiplication.

2.2. Related work

In SLMOPs, the coexistence of high dimensionality and sparsity complicates the optimization process, as only a limited number of decision variables contribute to the objectives. In this context, effective sparsity control is crucial for identifying meaningful solutions from a vast number of variables. Various evolutionary algorithms have been proposed to address these challenges, with a particular focus on methods for sparse multi-objective optimization. Below, we review key approaches, beginning with SparseEA [1], the first evolutionary algorithm specifically designed for SLMOPs.

SparseEA introduced a two-layer encoding scheme, where each solution $\mathbf{x}_i$ is represented by a binary mask vector $\mathbf{b}_i$, which indicates the active decision variables, and a real-valued vector $\mathbf{r}_i$, which defines the magnitude of these active variables. Specifically, each solution is constructed through element-wise multiplication of these two vectors, as illustrated in Fig. 1. This representation efficiently handles sparse solutions inherent to SLMOPs by explicitly separating sparsity patterns from magnitude values. For the entire population $\mathbf{X}$, all binary masks form the binary mask matrix $\mathbf{B}$, and all corresponding real-valued vectors form the magnitude matrix $\mathbf{R}$. Thus, the population $\mathbf{X}$ is generated by element-wise multiplication of matrices $\mathbf{B}$ and $\mathbf{R}$. These notations are consistently used throughout this paper for clarity. Although effective, SparseEA relies on static fitness scores computed only during initialization to measure the importance of decision variables, which often become inaccurate as the population evolves.

To overcome the limitation of static initialization, DSGEA [24] introduced dynamic fitness evaluations based on the evolving sparsity distributions, allowing importance measures to remain reflective of current evolutionary progress. Similarly, MGCEA [27] employed a multi-interval sampling strategy, which partitions each decision variable into multiple intervals. This approach reduces biases caused by inconsistent evaluations across intervals, thereby achieving a more robust estimation of variable importance. However, both methods primarily utilize binary mask information to update binary masks themselves, thereby neglecting potentially valuable insights available in real-valued components. MSKEA [25] partially addresses this gap by integrating both binary and real-valued information through a multi-stage knowledge fusion process. However, it suffers from two major drawbacks: its fixed-stage design reduces flexibility, and the excessive reliance on filter-based guidance during early iterations increases the risk of premature convergence.

Beyond these strategies, other methods aim to mine or exploit sparsity knowledge more explicitly. For instance, PM-MOEA [28] employs evolutionary pattern mining to extract maximum and minimum candidate sets of nonzero decision variables from the evolving Pareto front. These sets are then used to constrain the generation of offspring within relevant subspaces. Moreover, PM-MOEA incorporates customized binary crossover and mutation operators to better utilize the mined structural patterns. SGECF [29] and its extended version SCEA [30] further enhance sparsity exploitation via elitism and co-evolution. In particular, SCEA clusters non-dominated solutions to identify the current

Table 1
Comparison between DMKEA and existing methods in terms of claimed novelties, related work, limitations, and improvements.

Claimed Novelty	Related Works	Limitations of Existing Methods	How DMKEA Addresses the Limitations	Key Improvements
Continuous, probability-driven knowledge fusion	MSKEA [25]	Employs fixed-stage fusion; depends on manual switching	Uses probability-based fusion to adaptively combine prior, filter, and sparsity knowledge during search	Greater flexibility; fewer parameters to tune
Adaptive binary mutation with prior–filter guidance	MSKEA [25]	Prioritizes early filter-based guidance and late prior-based switch; may cause premature convergence	Uses an adaptive strategy that shifts guidance from prior to filter based on search progress	Better convergence stability and reduced risk of premature convergence
Dynamic, selective real-valued mutation strategy	SparseEA2 [33], TS-SparseEA [31], TS2-SparseEA [32]	Mutates inactive variables due to binary–real misalignment	Selectively mutates active variables and dynamically adjusts mutation strength during search	Improves efficiency by avoiding inactive mutations; enhances search adaptiveness
Comprehensive evaluation under extreme sparsity ($\theta = 0.01$)	SparseEA [1], MSKEA [25], DSGEA [24]	Lacks evaluation under $\theta = 0.01$; robustness in extreme sparsity remains unclear	Benchmarks DMKEA at $\theta = 0.01$ on synthetic sparse problems	Demonstrates scalability under extreme sparsity; supports practical use

optimal sparsity patterns, which guide both population initialization and offspring reproduction.

To further improve coordination between binary masks and real-valued components, two-stage matching strategies have emerged, such as TS-SparseEA [31] and TS2-SparseEA [32], which explicitly match binary masks with corresponding real-valued vectors. Despite their improved alignment, these approaches still do not fully guarantee correspondence between mutated real-valued variables and their binary masks, which may lead to computational inefficiency.

Grouping strategies have also been developed to address the interaction between binary and real-valued components. SparseEA2 [33] groups decision variables based on fitness-related criteria, synchronizing mutations across binary and real-valued layers. Other approaches, such as NUCEA [34] and DSGEA [24], use dynamic sparsity-based grouping methods to effectively manage problem dimensionality and sparsity patterns.

Additionally, machine learning techniques have been employed to further improve dimensionality reduction and the identification of influential variables. For instance, MOEA/PSL [35] leverages neural networks to reduce dimensionality and identify sparse distributions. MOEA-ADR [36] uses restricted Boltzmann machines (RBM) for clustering and guiding real-valued optimization. More recently, MOEADRL [37] incorporates deep reinforcement learning to dynamically pinpoint non-zero decision variables, significantly optimizing computational resources and improving search efficiency.

Initialization techniques such as Sparse Population Sampling (SPS) [38], which initializes individuals with varying sparsity levels, and its extensions Striped Sparse Population Sampling (SSPS) and Varied Striped Sparse Population Sampling (VSSPS), further optimize the sparsity of solutions through specialized placement of non-zero entries. These methods, combined with modifications to genetic operators (e.g., Sparse Simulated Binary Crossover (S-SBX) and Sparse Polynomial Mutation (S-PM)), help maintain sparsity throughout the evolutionary process.

Finally, operator design has also evolved, with innovations like the strongly convex sparse operator in S-ECSO [39], which generates sparse solutions based on a sparsity-oriented objective. Similarly, ST-CCPSO [40] uses a sparse truncation operator to selectively zero out variables based on cumulative gradient comparisons. These operator-level enhancements have proven effective in tackling the challenges of SLMOPs.

Nevertheless, existing methods remain limited by rigid knowledge usage schedules, inefficient real-binary coordination, and the lack of dynamic control mechanisms, thereby emphasizing the need for more flexible and sparsity-aware mechanisms. These challenges motivate the dynamic knowledge-based evolutionary approach proposed in this paper.

To clearly position our contributions, Table 1 summarizes the key innovations of DMKEA in comparison with representative existing methods, outlining the limitations each component aims to address.

3. Proposed method

Building upon recent advances such as MSKEA and MGCEA, DMKEA introduces several dynamic mechanisms that address key limitations of prior methods. Specifically, the proposed approach integrates adaptive operator selection, dynamic knowledge fusion, a binary mutation strategy guided by prior and filter vectors, and a real-valued mutation scheme with selective application and dynamic parameter control. These components enable DMKEA to balance exploration and exploitation effectively, enhancing scalability, diversity, and efficiency in sparse high-dimensional search spaces. The following subsections detail the algorithmic structure, initialization method, and adaptive mechanisms used for generating and refining offspring.

3.1. Overview of the proposed DMKEA

The overall framework of the proposed DMKEA is illustrated in Fig. 2. Similar to MSKEA[25], the DMKEA consists of four main phases: population initialization, dynamic knowledge updating, offspring generation via adaptive genetic operators, and environmental selection. Critical enhancements introduced in DMKEA (highlighted in shaded boxes in Fig. 2) include:

- A probabilistic control mechanism for dynamically switching between complementary binary genetic operators based on evolutionary progress.
- Knowledge-guided binary variation strategies incorporating the prior vector (pv) and filter guidance vector (fv) to prioritize influential variables and promote diversity.
- A selective and adaptive real-valued mutation strategy with dynamic parameter adjustment for improving search precision and computational efficiency.

A comprehensive pseudocode of DMKEA is provided in Algorithm 1, with detailed explanations of each component presented in subsequent subsections.

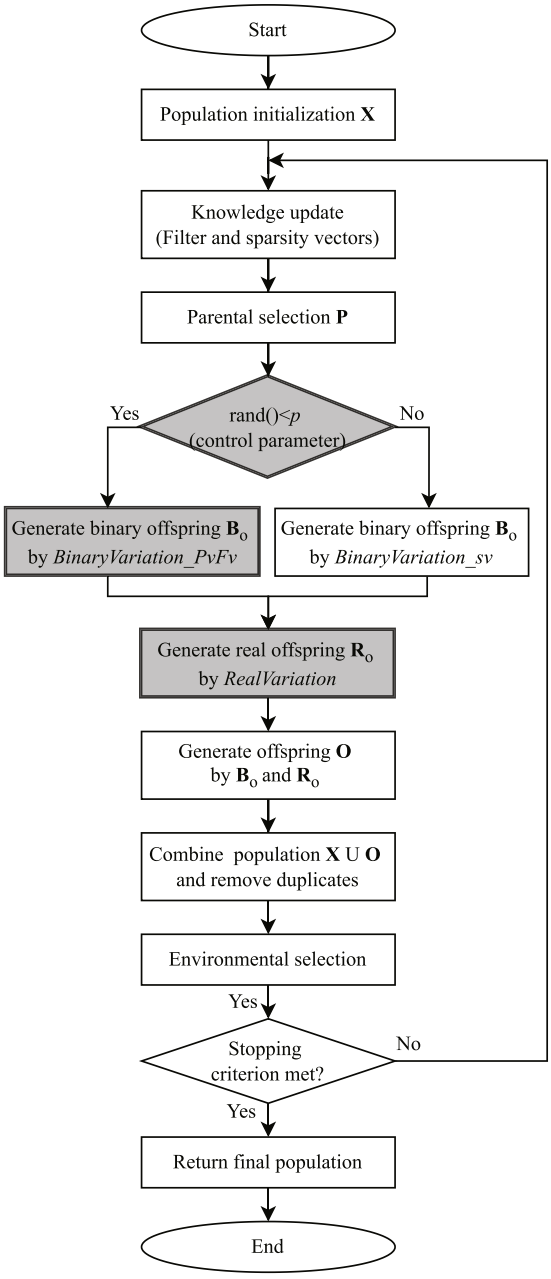


Fig. 2. Flowchart of the DMKEA framework.

3.2. Initialization via multi-interval sampling

DMKEA adopts the multi-interval sampling strategy proposed in MGCEA [27], which is briefly reviewed in Section 2.2.

Specifically, the method partitions each decision variable's domain into $nInt$ intervals and evaluates $nSamp$ solutions within each interval by using a binary mask that activates only the target variable. The resulting fitness ranks from non-dominated sorting are averaged to form the prior vector pv, where lower scores indicate higher importance. This approach mitigates sampling bias and yields a more reliable estimation of variable relevance.

To initialize the population, DMKEA uses binary tournament selection guided by pv, and explicitly restricts each individual to have at most 50 % active variables. Specifically, the number of active variables is randomly sampled from the range $[0, 0.5D]$. This promotes early-stage sparsity without assuming prior knowledge of the true sparsity level, and aligns with the general characteristics of SLMOPs. In practical settings where the sparsity ratio is known, this range can be flexibly adjusted.

Algorithm 1 Framework of DMKEA.

```
Initialize population X and prior vector pv
while termination condition not met do
    Update knowledge vectors fv, sv
    Generate parent pool P via binary tournament selection
    Calculate adaptive probability p
    if rand() < p then
        Generate binary offspring B using pv and fv
        Generate real-valued offspring R using mutation rate 1/d (d = number of active variables)
    else
        Generate binary offspring B using sv
        Generate real-valued offspring R using mutation rate 1/D
    end if
    Combine offspring: O = B∘R
    Merge and remove duplicates from population
    Environmental selection based on NSGA-II
end while
return Final population X
```

3.3. Knowledge-driven guidance vectors

DMKEA employs three knowledge-driven guidance vectors to adaptively direct the binary evolutionary process:

- **Prior vector (pv)**: It encodes variable importance, with initial estimates obtained through multi-interval sampling. It guides early-stage exploration and can be dynamically refined as the search progresses.
- **Filter guidance vector (fv)**: It captures variability among elite solutions in the non-dominated solution set (NDS). fv is used to guide a binary mutation operator (Algorithm 4), where two differing variables are randomly selected, and either the one with the higher fv value is set to 1, or the one with the lower fv value is set to 0. This promotes exploration along more variable dimensions and encourages sustained search in sparse regions:

$$fv_i = \sqrt{\frac{1}{|NDS|}\sum_{j=1}^{|NDS|}\left(\mathbf{x}_i^j - \frac{1}{|NDS|}\sum_{j=1}^{|NDS|}\mathbf{x}_i^j\right)^2}, \tag{3}$$

  where $\mathbf{x}_i^j$ is the value of the i-th variable in the j-th solution within NDS, and $|NDS|$ is the total number of solutions in NDS.
- **Statistical guidance vector (sv)**: It reflects current sparsity patterns within NDS, updated dynamically at each generation:

$$sv_i = \frac{|NDS'|}{|NDS'|+|NDS|}sv_i' + \frac{|NDS|}{|NDS'|+|NDS|}csv_i, \tag{4}$$

  where sv_i' is the previous sv value, $|NDS'|$ denotes the size of the previous non-dominated solution set NDS', and

$$csv_i = \frac{1}{|NDS|}\sum_{j=1}^{|NDS|}\mathbf{b}_i^j, \tag{5}$$

  with $\mathbf{b}_i^j$ denoting the binary state of the i-th variable in the j-th solution.

Although the prior vector (pv) is computed at initialization, DMKEA optionally refines it over time by using the update rule adapted from MSKEA [25]:

$$pv = pv \cdot (\mathbf{1} - sv) \cdot \sqrt{\delta} + pv, \tag{6}$$

where $\mathbf{1}$ denotes a vector of ones with the same dimension as sv, and δ denotes the proportion of evaluations consumed. This update rule

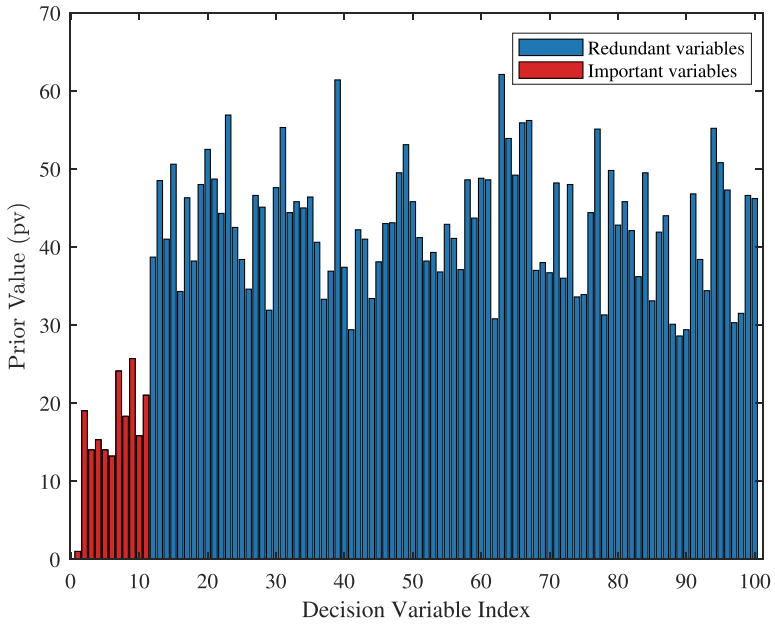


Fig. 3. Validation of initial variable importance estimates in SMOP1 using the prior vector (pv). Ground-truth important variables (2–11, in red) show significantly lower pv values than redundant ones (in blue), indicating the effectiveness of multi-interval sampling.

progressively refines the previous pv based on evolving sparsity patterns in the non-dominated set. The term $(\mathbf{1} - sv)$ ensures that variables frequently appearing as nonzero in elite solutions (higher sv) receive smaller increases, while infrequently active variables are updated more aggressively. Since lower pv values indicate higher variable importance in DMKEA, this mechanism helps prioritize relevant variables as optimization progresses. The update strength grows with $\sqrt{\delta}$, reflecting increasing confidence in the learned sparsity structure. Retaining the previous pv preserves stability and avoids abrupt changes in guidance.

In the benchmark suite (SMOP1–SMOP8), the subset of decision variables that can take non-zero values in any Pareto-optimal solution is structurally predetermined by the problem definitions (see Table 2). Although variables within this important subset may interact (e.g., due to epistasis in SMOP7 and SMOP8), the distinction between important and redundant variables remains constant throughout the optimization process. Consequently, the initial pv obtained from multi-interval sampling provides a stable and reasonably accurate estimate of variable importance, which allows us to omit dynamic updates in these synthetic settings, simplifying the algorithm and reducing computational overhead.

To validate this assumption, we compared the initial pv values assigned to the ground-truth important and redundant variables in SMOP1. For this case, the number of decision variables is 100, the sparsity level is $\theta = 0.1$, and the number of objectives is $M = 2$. According to the problem definition (see Section 4.1.2), variable 1 determines the shape of the Pareto front, while variables 2–11 are nonzero decision variables that influence the objectives. The remaining variables are redundant. Fig. 3 visualizes the initial pv: the horizontal axis denotes variable indices, and the vertical axis shows the corresponding pv values. Note that lower pv values indicate higher estimated variable importance. As expected, ground-truth important variables (in red) exhibit distinctly lower pv scores than redundant ones (in blue), confirming the ability of multi-interval sampling to identify relevant variables at initialization.

In addition, we conducted a performance comparison between the original DMKEA variant (with static pv) and a modified version incorporating dynamic pv updates under sparsity level $\theta = 0.1$ and objective number $M = 2$. Across 24 benchmark cases with varying dimensionalities ($D = 500, 1000, 2000$), each configuration was independently repeated 100 times. The original variant achieved equivalent or superior performance in 23 out of 24 cases, with only a negligible disadvantage on SMOP5 at $D = 2000$ (see Table A.1 in Appendix A). These results further support that a static pv is sufficient and more efficient in these structurally sparse synthetic problems.

In contrast, real-world sparse problems such as Sparse_NN and Sparse_SR exhibit complex, context-dependent, or dynamically evolving variable interactions. In Sparse_NN, the importance of each variable (weights and biases in a neural network) depends strongly on interactions with other parameters, making static guidance from the initial pv insufficient. Similarly, in Sparse_SR, linear interactions among variables and high precision sensitivity complicate initial importance estimation, necessitating adaptive refinement of pv.

In summary, a static pv computed at initialization suffices for synthetic benchmarks with fixed and clearly separated variable importance structures (SMOP1—SMOP8). However, dynamically updating pv using feedback from sv, as in (6), significantly improves guidance quality and search performance for complex, real-world sparse optimization scenarios.

3.4. Probabilistic operator selection

To flexibly integrate multiple knowledge sources over time, DMKEA introduces a probabilistic control parameter p to balance two binary variation strategies: *BinaryVariation_PvFv*, guided by prior and filter knowledge (pv, fv), and *BinaryVariation_sv*, driven by the evolving statistical vector sv. Early in the search, sv is typically unreliable due to limited population diversity, making prior-based guidance more trustworthy. As the optimization progresses and more non-dominated solutions accumulate, sv becomes increasingly informative.

To reflect this shift in guidance reliability, DMKEA dynamically adjusts the probability p using the logistic function:

$$p = \frac{1}{1 + \exp\left(\tau \cdot (\delta - 0.5)\right)}, \tag{7}$$

where δ denotes the proportion of consumed evaluations, and τ controls the steepness of the transition.

This formulation enables a smooth and bounded transition of p within $(0, 1)$, centered around $\delta = 0.5$. Compared to fixed or manually scheduled probabilities, the logistic formulation offers several advantages. In MSKEA [25], the operator selection probability changes

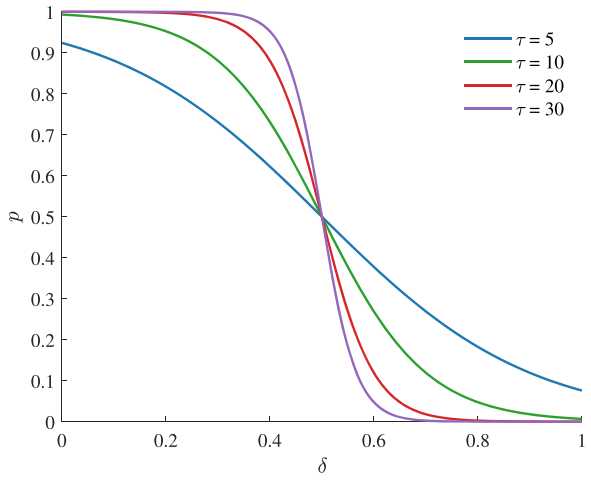


Fig. 4. Dynamic probability p curves under different values of τ.

abruptly: it remains 100 % until 38 % of evaluations are consumed, drops to 50 %, and then to 0 %. Such sharp transitions may limit early-stage diversity and make tuning inflexible. In contrast, the logistic curve offers a slow-fast-slow transition pattern: initially, p remains close to 1, ensuring the more reliable prior-filter guided operator is selected. As the statistical vector sv becomes more informative, the transition accelerates, gradually increasing the probability of statistically-guided variation. Toward the end, p stabilizes at a low value, effectively prioritizing sv. This smooth evolution avoids premature reliance on immature signals while maintaining diversity early on. Moreover, the hyperparameter τ provides an interpretable and tunable control over the steepness of this shift, offering a more flexible and principled alternative to handcrafted schedules.

The behavior of p under different values of τ is visualized in Fig. 4, which illustrates how different values of τ affect the shape of the probability curve $p(\delta)$. A larger τ results in a steeper transition, causing p to drop more abruptly, while a smaller τ produces a more gradual shift. A sensitivity analysis of τ is provided in Section 4.3.1.

In addition to guiding binary operator selection, the probability p also influences the real-valued mutation rate by coupling both mechanisms under a unified probabilistic decision (see Section 3.6.1 for details). This design promotes broader exploration in early stages (when p is high) and encourages more conservative refinement later (as p decreases), enabling coherent progression across both binary and real-valued components.

3.5. Binary genetic operators

3.5.1. Binary variation guided by prior and filter vectors

The *BinaryVariation_PvFv* algorithm generates binary offspring in two stages: a pv-guided crossover (via *Crossover_pv*) exchanges genetic material between parents, followed by a mutation operation guided by either pv or fv, depending on the evaluation proportion δ. Each solution consists of a binary vector and a real-valued vector (e.g., $\mathbf{p} = \mathbf{b}_p \circ \mathbf{r}_p$); here, we focus on the binary component. The full process is presented in Algorithm 2.

Crossover. The *Crossover_pv* operator (Algorithm 3) first copies the parent $\mathbf{b}_p$ to the offspring $\mathbf{b}_o$. With probability 0.5, two indices $m, n \in S = \{\, i \mid \mathbf{b}_p(i) = 1 \wedge \mathbf{b}_q(i) = 0 \,\}$ are sampled uniformly at random, and the one with the larger pv is flipped to 0 in $\mathbf{b}_o$. Otherwise, two indices $m, n \in T = \{\, i \mid \mathbf{b}_p(i) = 0 \wedge \mathbf{b}_q(i) = 1 \,\}$ are sampled, and the one with the smaller pv is flipped to 1 in $\mathbf{b}_o$. In our implementation, a larger pv indicates lower importance (i.e., higher pruning priority); thus the operator simultaneously prunes high-pv components and activates low-pv ones.

Mutation. After crossover, mutation is applied based on either pv or fv, depending on δ. Specifically, if a randomly generated value is less than δ, mutation follows fv-guided adaptation (*Mutation_fv*); otherwise, it follows pv-guided adaptation (*Mutation_pv*). Since δ represents the proportion of consumed evaluations, this mechanism gradually shifts the mutation guidance from fv in the early stages to pv in the later stages. This transition is the opposite of MSKEA's approach [25], where early reliance on fv-based guidance was used to accelerate convergence but led to a higher risk of premature convergence. By reversing this strategy, our method preserves early-stage diversity and improves long-term stability.

Algorithm 2 *BinaryVariation_PvFv*($\mathbf{b}_p, \mathbf{b}_q, pv, fv, \delta$).

```
b_o ← CROSSOVER_PV(b_p, b_q, pv)        ▷ Apply crossover guided by pv
if rand() < δ then
    MUTATION_FV(b_o, fv)                ▷ Apply fv-guided mutation
else
    MUTATION_PV(b_o, pv)                ▷ Apply pv-guided mutation
end if
return b_o
```

Algorithm 3 *Crossover_pv*($\mathbf{b}_p, \mathbf{b}_q, pv$).

```
b_o ← b_p                               ▷ Initialize offspring binary vector
if rand() < 0.5 then
    S ← { i | b_p(i) = 1 ∧ b_q(i) = 0 }
    Randomly select two indices m, n from S
    Compare pv(m) and pv(n) and set the one with higher pv to 0 in b_o
else
    T ← { i | b_p(i) = 0 ∧ b_q(i) = 1 }
    Randomly select two indices m, n from T
    Compare pv(m) and pv(n) and set the one with lower pv to 1 in b_o
end if
return b_o
```

The *Mutation_fv* function, outlined in Algorithm 4, modifies the offspring based on the variability captured in fv. It begins by constructing an auxiliary binary vector v, where $v(i) = 1$ if $fv(i)) > 0$. Next, it identifies var_{diff}, the set of indices where $\mathbf{b}_o$ differs from v. From this set, two indices m and n are randomly selected. If rand() < 0.5, the bit with the higher fv value is set to 1 in $\mathbf{b}_o$; otherwise, the one with the lower fv value is set to 0. In essence, this operation selectively flips bits in $\mathbf{b}_o$ where it deviates most from the auxiliary vector v, thereby promoting alignment with the information encoded in fv.

Algorithm 4 *Mutation_fv*($\mathbf{b}_o, fv$).

```
Construct auxiliary vector v, where v_i = 1 if fv_i > 0
Compute var_diff ← b_o ⊕ v (XOR operation)
Randomly select m, n from var_diff
if rand() < 0.5 then
    Set the value at the index with higher fv in b_o to 1
else
    Set the value at the index with lower fv in b_o to 0
end if
return b_o
```

The *Mutation_pv* function modifies the offspring based on the prior vector pv, which captures how frequently each variable is set to 1 in the elite solutions. This mutation strategy reinforces variables that are commonly selected in high-performing individuals while suppressing those that appear less frequently. The corresponding logic is summarized in Algorithm 5.

If a random value is less than 0.5, the algorithm selects two indices where $\mathbf{b}_o(i) = 1$ and $pv(i)$ is above the median. Among them, the one with the higher pv value is set to 0 in $\mathbf{b}_o$. Otherwise, it selects two indices

where $\mathbf{b}_o(i) = 0$ and $pv(i)$ is below the median, and sets the one with the lower pv value to 1. This encourages the retention of useful variables and the introduction of promising ones during the search.

Algorithm 5 *Mutation_pv*($\mathbf{b}_o, pv$).

if rand() < 0.5 **then**
  Select two indices m, n where $\mathbf{b}_o(i) = 1$ and $pv(i) >$ median(pv)
  Set the value at the index with higher pv to 0 in $\mathbf{b}_o$
else
  Select two indices m, n where $\mathbf{b}_o(i) = 0$ and $pv(i) <$ median(pv)
  Set the value at the index with lower pv to 1 in $\mathbf{b}_o$
end if
return $\mathbf{b}_o$

3.5.2. Binary variation guided by statistical vector (sv)

The *BinaryVariation_sv* algorithm uses the statistical guidance vector sv to determine the probability that each variable should be set to 1, thereby ensuring that offspring solutions reflect the sparsity patterns observed in elite solutions. The procedure consists of two main steps–crossover and mutation–and is summarized in Algorithm 6.

Algorithm 6 *BinaryVariation_sv*($\mathbf{P}$, sv, δ).

Input: $\mathbf{P}$ (parent population), sv (statistical guidance vector), δ (proportion of consumed evaluations)
Output: $\mathbf{B}_o$ (binary offspring population)
$\mathbf{B}_o \leftarrow \emptyset$ ▷ Initialize offspring population
while $\mathbf{P}$ is not empty **do**
  $[\mathbf{p}, \mathbf{q}] \leftarrow$ Randomly select two parents from $\mathbf{P}$
  $\mathbf{P} \leftarrow \mathbf{P} \setminus \{\mathbf{p}, \mathbf{q}\}$
  $\mathbf{b}_o \leftarrow \mathbf{b}_p$ ▷ Initialize offspring as copy of first parent's binary vector
  ▷ *Crossover Step: XOR-based flipping*
  $var_{diff} \leftarrow$ XOR($\mathbf{b}_p, \mathbf{b}_q$)
  $rand_{diff} \leftarrow$ Generate $|var_{diff}|$ random numbers in $[0, 1]$
  $rate \leftarrow$ Compute flipping probabilities using Equation (8)
  $\mathbf{b}_o(var_{diff}) \leftarrow \mathbf{b}_o(var_{diff}) \oplus (rand_{diff} < rate(var_{diff}))$
  ▷ *Mutation Step: Sparsity-aware flipping*
  $pos \leftarrow$ Randomly select indices with probability $1/D$
  $rand_{pos} \leftarrow$ Generate random numbers in $[0, 1]$ for each index in pos
  $flip_{pos} \leftarrow (rand_{pos} < rate[pos])$ ▷ Determine bits to flip
  $\mathbf{b}_o[pos] \leftarrow \mathbf{b}_o[pos] \oplus flip_{pos}$ ▷ Apply mutation flips
  $\mathbf{B}_o \leftarrow \mathbf{B}_o \cup \{\mathbf{b}_o\}$ ▷ Add offspring to population
end while
return $\mathbf{B}_o$

In the crossover phase, an XOR operation first identifies indices at which the binary vectors of two parent solutions differ. For each such index i, the flipping probability $rate(i)$ is calculated by using both $sv(i)$ and the current value of $\mathbf{b}_o(i)$:

$$rate(i) = \mathbf{b}_o(i) \cdot (1 - sv(i)) + (1 - \mathbf{b}_o(i)) \cdot sv(i). \tag{8}$$

Here, $sv(i)$ represents the likelihood that the variable at index i should be set to 1 (as observed in elite solutions), so $1 - sv(i)$ naturally corresponds to the likelihood that it should be 0. Thus, if $\mathbf{b}_o(i) = 0$, the probability of flipping it to 1 is $sv(i)$, and if $\mathbf{b}_o(i) = 1$, the probability of flipping it to 0 is $1 - sv(i)$. A random number is then generated for each differing index i; if it is smaller than $rate(i)$, the corresponding bit in $\mathbf{b}_o$ is flipped.

In the mutation phase, a subset of indices is selected for potential flipping, with each index chosen at random based on a probability of $1/D$. For every selected index i, a second random number is generated and compared to $rate(i)$; if it falls below $rate(i)$, that bit in $\mathbf{b}_o$ is flipped. Therefore, the mutation step refines the offspring and preserves consistency with the statistical guidance provided by sv.

3.6. Adaptive real-valued genetic operator

To enhance the solution quality and computational efficiency, we propose a real-valued variation strategy, summarized in Algorithm 7, which (1) restricts mutation to variables that are currently set to 1 in the associated binary mask, and (2) dynamically adjusts both mutation and crossover parameters throughout the optimization process.

Algorithm 7 *RealVariation*($\mathbf{P}, \mathbf{B}_o, \delta, disM_{\min}, disM_{\max}, p$).

Input: $\mathbf{P}$ (parent pool), $\mathbf{B}_o$ (binary offspring), δ (evaluation proportion), $disM_{\min}$, $disM_{\max}$ (distribution index bounds), p (probability used for mutation rate decision)
Output: Updated real-valued offspring population $\mathbf{R}_o$
$[disM, disC] \leftarrow$ Update parameter indices $disM$ and $disC$ via (9)(δ)
$\mathbf{R}_o \leftarrow \emptyset$
for $i = 1$ to N **do** ▷ N is the population size
  $[\mathbf{p}, \mathbf{q}] \leftarrow \mathbf{P}(i)$ ▷ Select parent pair
  $[\mathbf{r}_p, \mathbf{r}_q] \leftarrow$ Real-valued components of $\mathbf{p}$, $\mathbf{q}$
  $\mathbf{b}_o \leftarrow \mathbf{B}_o(i)$ ▷ Binary mask of offspring
  $\mathbf{r}_o \leftarrow$ SBX($\mathbf{r}_p, \mathbf{r}_q, disC$) ▷ Apply simulated binary crossover
  ▷ Determine mutation rate based on p
  if rand < p **then**
    $d \leftarrow \sum(\mathbf{b}_o)$ ▷ Use higher mutation rate $1/d$ for stronger perturbation
  else
    $d \leftarrow D$ ▷ Use lower mutation rate $1/D$ for fine-tuned adjustments
  end if
  $\mathbf{r}_o \leftarrow$ PM($\mathbf{r}_o, \mathbf{b}_o, disM, 1/d$) ▷ Apply mutation only to non-zero $\mathbf{b}_o$ entries
  $\mathbf{R}_o \leftarrow \mathbf{R}_o \cup \{\mathbf{r}_o\}$
end for
return $\mathbf{R}_o$

3.6.1. Selective mutation

Traditional mutation approaches apply perturbations to all real-valued variables, including those corresponding to zero-valued (inactive) entries in the binary mask. In sparse scenarios, these inactive variables do not contribute to the solution's quality, resulting in unnecessary computational overhead [33].

To improve efficiency, DMKEA restricts real-valued mutation to variables that are currently active (i.e., those with a binary mask value of 1). The mutation rate is determined by the same probabilistic decision used to select the binary variation strategy–based on a single comparison between *rand()* and p, performed once per generation (see Section 3.4). If the prior-filter-guided operator (*BinaryVariation_PvFv*) is selected, the mutation rate is set to $1/d$, where $d = \sum_i \mathbf{b}_o(i)$, enabling broader exploration of active variables. Otherwise, the mutation rate is set to $1/D$, supporting the statistical-guided strategy (*BinaryVariation_sv*) with more conservative adjustments.

This selective and stage-aware mutation strategy improves computational efficiency and enhances alignment between the binary and real-valued representations, leading to more effective search under sparse conditions.

3.6.2. Adaptive parameters

Genetic operators such as SBX and PM often rely on fixed distribution indices, which can be suboptimal as the algorithm progresses. Fixed small indices may cause overshooting near optima in later stages, while fixed large indices may hinder early exploration.

Table 2
Pareto front and Pareto set structures of benchmark problems.

Benchmark Problem	Pareto Front	Pareto Set	Properties of landscape	Type of Variables
SMOP1	$f_1 + f_2 = 1$	$[0,1]^1 \times \{\pi/3\}^K \times \{0\}^{D-K-1}$	Multi-modality	Real
SMOP2			Deception, multi-modality	
SMOP3			Deception	
SMOP4	$(1-f_1)^2 + (1-f_2)^2 = 1$	$[0,1]^1 \times [-1,2]^K \times \{0\}^{D-K-1}$	Low intrinsic dimensionality, multi-modality	
SMOP5		$[0,1]^1 \times \{\pi/3\}^K \times \{0\}^{D-K-1}$	Unimodality	
SMOP6		$[0,1]^1 \times \{\pi/3\}^K \times \{0,\pi/3\}^{D-K-1}$	Multi-modality	
SMOP7	$f_1^2 + f_2^2 = 1$	$[0,1]^1 \times \{\pi/3\}^K \times \{0\}^{D-K-1}$	Epistasis, multi-modality	
SMOP8		$[0,1]^1 \times \prod_{i=1}^{K} \{\mathrm{mod}(\pi(K+2-i),2)\} \times \{0\}^{D-K-1}$	Epistasis, deception	

D: the number of decision variables, $K = \lceil \theta(D-1) \rceil$, and θ: the ratio of nonzero decision variables in each Pareto optimal solution.

In DMKEA, both the crossover index (*disC*) and the mutation index (*disM*) are adaptive according to the evaluation proportion δ:

$$disM = disM_{\min} + \left(disM_{\max} - disM_{\min} \right) \times \delta, \tag{9}$$

where $disM_{\min}$ and $disM_{\max}$ are the minimum and maximum allowable values for *disM*. Initially, a low *disM* fosters broad exploration, and as δ increases, the index shifts toward $disM_{\max}$ to enable finer refinements.

By combining selective mutation with adaptive parameters, *Real-Variation* achieves both efficient resource usage and dynamic control over search granularity. This approach mitigates the overshooting issue in later stages and avoids under-exploration early on, leading to robust convergence and higher-quality solutions.

4. Experimental studies

This section empirically evaluates the effectiveness of the proposed DMKEA by conducting comparative experiments on eight benchmark SLMOPs (SMOP1–SMOP8 [1]) and two real-world problems: sparse neural network training (Sparse_NN) and signal reconstruction (Sparse_SR). The characteristics of these problems are detailed in Section 4.1.2, with benchmark features summarized in Table 2. The objectives of the experiments are threefold: *(i)* To assess the competitiveness of DMKEA against state-of-the-art sparse optimization algorithms; *(ii)* To evaluate the effectiveness of the enhanced binary genetic operator, the selective real-valued mutation strategy, the adaptive parameter control mechanism, and the adaptive operator selection mechanism; *(iii)* To evaluate algorithm performance stability and efficiency across different problem dimensions and sparsity levels.

All experiments were implemented in MATLAB 2020b integrated with the PlatEMO framework [41] and conducted on a PC with an AMD Ryzen 7 5800H CPU, 16 GB RAM, and Windows 10.

4.1. Experimental setup

4.1.1. Algorithms for comparison

We selected eleven competitive algorithms specifically designed for solving SLMOPs to benchmark the performance of the proposed DMKEA. These include both classical baselines and recent state-of-the-art approaches. MSKEA [25] is chosen as the primary baseline due to its representative multi-stage knowledge fusion framework. SparseEA [1] represents one of the earliest sparse optimization algorithms, relying on simple heuristics without structural modeling. PM-MOEA [28], SparseEA2 [33], MGCEA [27], NUCEA [34], S-NSGAII [42], SGECF [29], and SCEA [30] reflect recent advancements that incorporate grouping, clustering, or co-evolutionary strategies to exploit sparsity information and variable dependencies better. MOEA/PSL [35] and SECSO [40] are also included for their robust performance in high-dimensional settings, including problems with sparse structures. Together, these methods span a diverse range of sparsity modeling strategies.

For fair comparison, all algorithms were implemented using their original or recommended default parameter settings in PlatEMO, with minor adjustments to enable parallel execution without affecting algorithmic behavior.

4.1.2. Test problems and experimental parameters

We selected eight representative SMOP benchmarks (SMOP1—SMOP8), originally proposed in [1], to offer comprehensive scenarios for evaluating sparse optimization performance. These problems have become standard benchmarks in recent SLMOP studies (e.g., [25,27–30,33,34,42]). As summarized in Table 2, they exhibit diverse Pareto structures and landscape characteristics, including multi-modality, deception, epistasis, and low intrinsic dimensionality, which pose a range of search difficulties and better reflect real-world challenges. Definitions of the objective functions f_1 and f_2 are given in [1]. We fixed the number of objectives at two ($M = 2$) and tested three problem dimensionalities ($D = 500, 1000, 2000$) to evaluate scalability. Two sparsity levels were considered: moderate ($\theta = 0.1$) and high ($\theta = 0.01$).

To further validate generalizability, we include two real-world sparse optimization problems:

- Sparse_NN: A neural network sparsification task with $D = 1601$ and $M = 2$.
- Sparse_SR: A sparse signal reconstruction task with $D = 1024$ and $M = 2$.

Their detailed characteristics and result analyses are provided in Section 4.2.3.

For all experiments, the population size was set to 100. The maximum number of fitness evaluations was set to $150 \times D$ for benchmark problems and $100 \times D$ for real-world problems. The multi-interval sampling parameters–number of intervals $nInt = 5$ and number of samples per interval $nSamp = 2$–were adopted from MGCEA [27] to ensure consistency across evaluations.

4.1.3. Performance metrics and statistical tests

The Inverted Generational Distance (IGD) metric was used to evaluate performance on benchmark problems, with smaller values indicating better results [43]. It measures the distance between the true Pareto-optimal set PF^* and the Pareto front approximation PF produced by MOEAs, defined as the average Euclidean distance from each point in PF^* to its nearest point in PF. For IGD calculations, approximately 10,000 reference points were sampled from the Pareto front PF^* of each problem.

For real-world sparse optimization problems (e.g., Sparse_NN and Sparse_SR), the true Pareto front is unknown. Therefore, the Hypervolume (HV) metric was used to assess solution quality. HV measures the volume of the objective space that is dominated by the obtained non-dominated set and bounded by a predefined reference point, set to (1, 1) in this study. A larger HV value indicates better convergence and diversity.

To ensure statistical significance, we executed 100 independent runs for all the experiments involving statistical comparisons

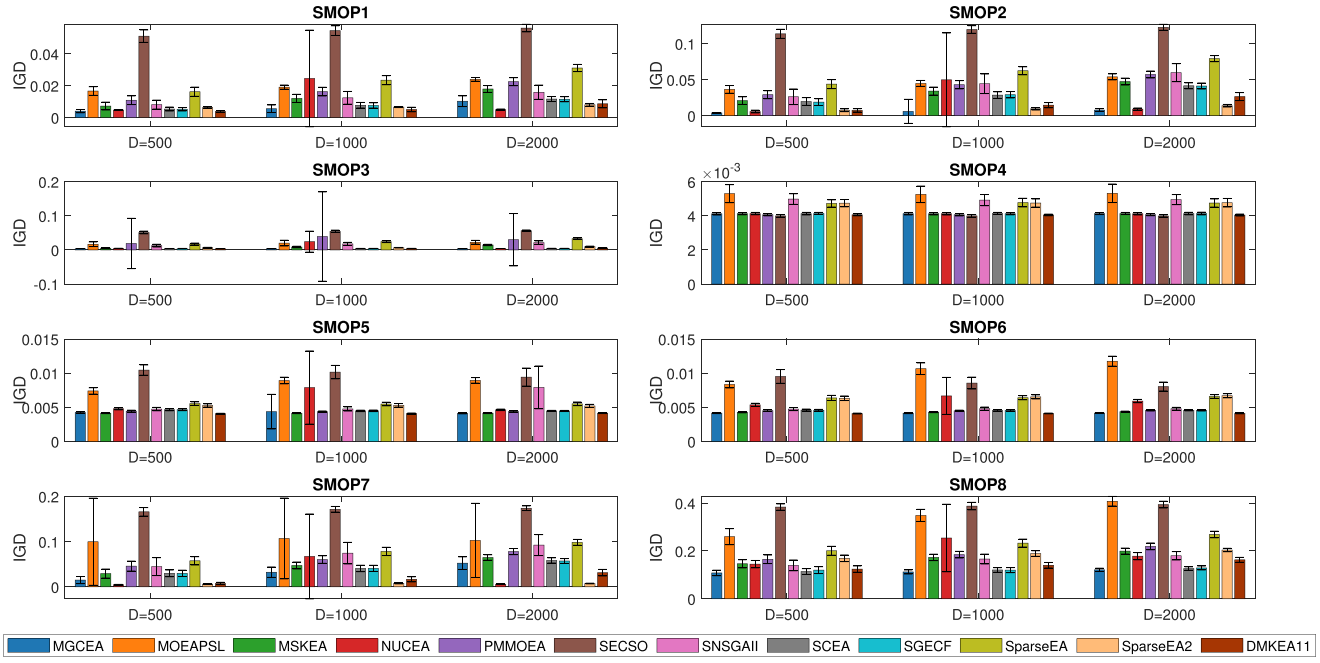


Fig. 5. Comparison of DMKEA and eleven baseline algorithms on SMOP1–SMOP8 at sparsity level $\theta = 0.1$ and dimensions $D = 500$, 1000, and 2000. Each subplot corresponds to a benchmark problem, with groups of twelve bars representing the algorithms at each dimension. Error bars show the standard deviation of IGD across 100 independent runs. Y-axis scales may differ across subplots for visibility.

(Sections 4.2.1–4.2.3, 4.3.1, and 4.4). For the runtime-only analysis in Section 4.2.4, which does not involve statistical comparisons, we conducted 30 runs, consistent with prior practice in sparse large-scale multi-objective optimization studies [25,28,29,33]. The Wilcoxon rank-sum test with a significance level of 0.05 was used for statistical analysis [43]. In the result tables, the symbols +, -, and ≈ indicate that a competing algorithm performs significantly better, significantly worse, or similarly to DMKEA, respectively.

4.2. Experimental results and analysis

4.2.1. Results for moderate sparsity ($\theta = 0.1$)

This subsection evaluates the performance of DMKEA and eleven baseline algorithms under a moderate sparsity setting ($\theta = 0.1$). To improve clarity and readability, the detailed numerical results (mean and standard deviation of IGD values) are presented in Table B.1, in Appendix B. Statistical significance is assessed with the Wilcoxon rank-sum test, and the best-performing results are shown in bold. Fig. 5 visually compares the performance of DMKEA with eleven baseline algorithms on the SMOP1–SMOP8 benchmark problems under the same sparsity level. Each subplot represents one benchmark problem, and each group of bars corresponds to a specific dimensionality ($D = 500$, 1000, 2000). Error bars represent the standard deviation over 100 independent runs.

As shown in the figure (with full results in Table B.1), DMKEA achieved the lowest IGD in 8 out of 24 test cases. Although not always the absolute best, it consistently ranked among the top performers across a variety of problem settings. Notably, five of the eleven competing algorithms failed to outperform DMKEA on any test case, further underscoring its robustness.

In SMOP4, for instance, DMKEA performed slightly worse than SECSO but still outperformed all other methods. Even in challenging high-dimensional scenarios–such as those with 2000 variables–DMKEA outperformed most competitors and remained competitive with MGCEA and NUCEA. Specifically, it achieved an equal number of wins and losses (four wins and four losses) when compared individually against each of these two algorithms. The effectiveness of MGCEA and NUCEA in such cases can be attributed to their grouping strategies, which are particularly beneficial for large-scale optimization. We anticipate that incorporating similar grouping strategies into DMKEA could further enhance its scalability, and we plan to explore this direction in future work. Overall, these results confirm that DMKEA effectively balances exploration and exploitation, making it a promising approach for sparse multi-objective optimization.

To provide additional insights into the performance of DMKEA, we present parallel coordinate plots for decision variables in SMOP1 and SMOP3 with 500 decision variables in Figs. 6 and 7, respectively. For both problems, the optimal solutions are characterized by the first 50 decision variables being non-zero with values set to $3/\pi$. DMKEA accurately identified these non-zero variables, as shown in the plots. Moreover, the values of the identified non-zero decision variables were consistently closer to the optimal value of $3/\pi$ compared to those produced by other algorithms. This demonstrates that DMKEA not only identified the active variables with high precision but also determined their values with greater accuracy. The adaptive real-valued genetic operators in DMKEA contribute to this enhanced precision. In contrast, other algorithms struggled to accurately identify and optimize non-zero variables. Therefore, DMKEA shows a clear advantage in solving SLMOPs, where both accurate identification and optimization of non-zero variables are critical.

Fig. 8 presents the runtime distribution of different algorithms across SMOP1–SMOP8 under varying dimensions at a sparsity level of $\theta = 0.1$. As shown in the box plots, DMKEA exhibits a relatively stable runtime across all settings, and in many cases, it is slightly faster than some baseline methods. While algorithms like SECSO and S-NSGAII report the shortest runtimes, they generally underperform in solution quality. On the other hand, MOEA/PSL and PM-MOEA show longer and more variable runtimes. Overall, DMKEA achieves comparable or slightly better efficiency than most competitors, without incurring excessive computational cost.

4.2.2. Results for high sparsity ($\theta = 0.01$)

This subsection evaluates the performance of DMKEA and eleven baseline algorithms under a highly sparse setting ($\theta = 0.01$). To enhance clarity, the detailed numerical results (mean and standard deviation of

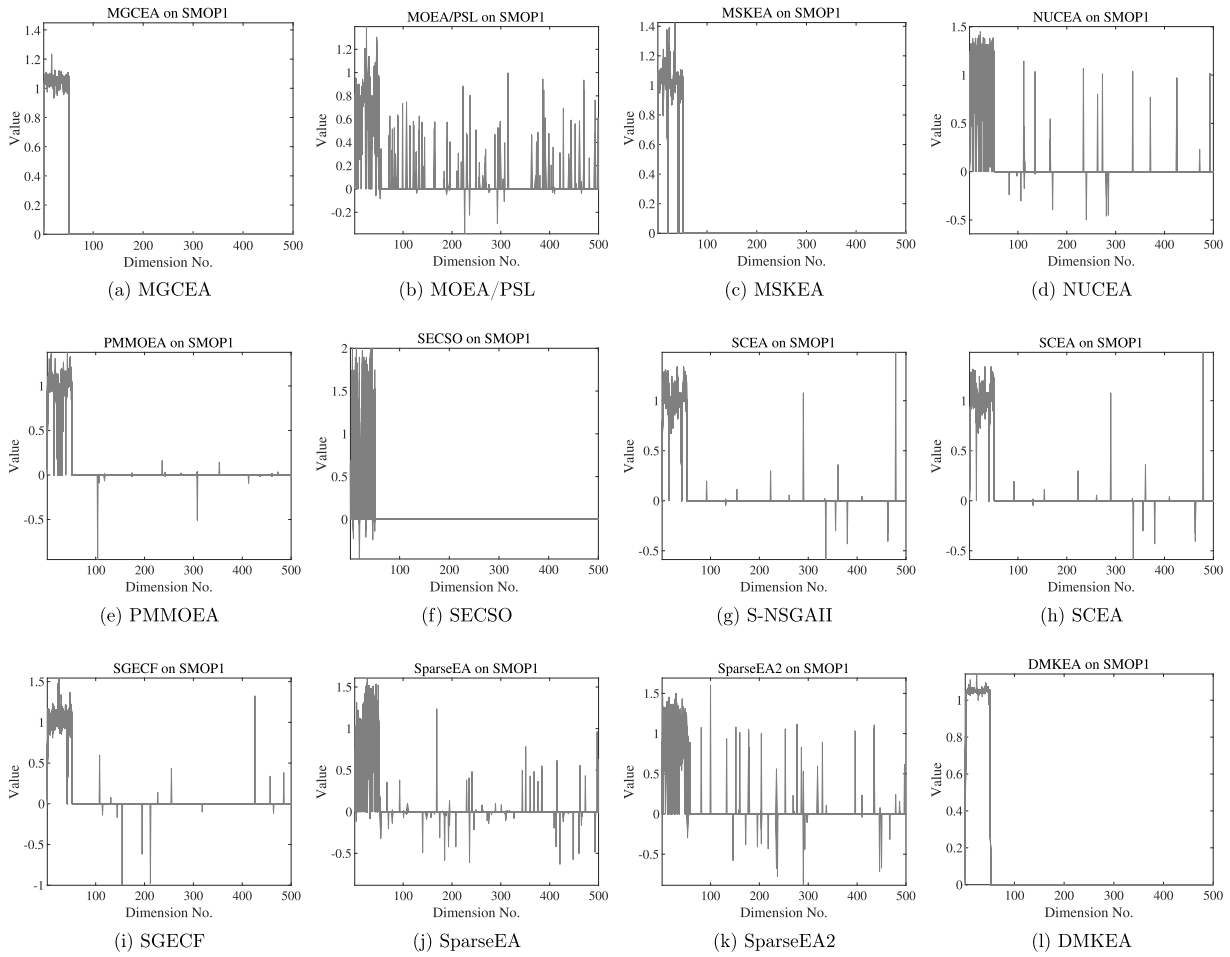


Fig. 6. Parallel coordinates of decision variables on SMOP1 ($D = 500$, $\theta = 0.1$).

IGD values) are provided in Table C.1 in the Appendix C. Statistical significance is assessed with the Wilcoxon rank-sum test, with the best-performing results highlighted in bold.

To provide a clearer overview of algorithmic performance, Fig. 9 summarizes the results across all eight benchmark problems (SMOP1–SMOP8) at $\theta = 0.01$ and $D = 500, 1000, 2000$. Each subplot corresponds to one problem, and each group of bars compares twelve algorithms at a given dimensionality. As shown in the figure (with full numerical results in Table C.1), the proposed DMKEA achieved the best results in 23 out of 24 test cases, demonstrating its effectiveness in solving SLMOPs. The only exception was SMOP4 at $D = 2000$, where SECSO slightly outperformed DMKEA, although DMKEA still outperformed all other algorithms.

Moreover, DMKEA exhibited consistently low standard deviations across all benchmark problems, indicating stable and reliable performance. These results suggest that DMKEA not only obtains optimal or near-optimal solutions but also does so with a high degree of consistency, which makes it a robust approach for tackling SLMOPs across diverse problem settings.

To explain why DMKEA outperforms other algorithms under extreme sparsity, we briefly examine the limitations of existing approaches in such scenarios. Under extreme sparsity (e.g., $\theta = 0.01$), the performance of many existing algorithms tends to degrade due to two main limitations. First, most approaches (except SparseEA2) do not ensure alignment between the binary vector and the real-valued vector, leading to unnecessary updates on variables whose binary entries are zero–thereby wasting computational resources. Although SparseEA2 attempts to mitigate this with a grouping strategy, it offers only coarse control. Second, existing methods commonly rely on fixed-parameter operators such as SBX and PM for real-valued variation. These fixed parameters may cause overly aggressive perturbations in the late stages of the search, especially under high sparsity where fine-tuning becomes critical. In contrast, DMKEA addresses both issues by selectively mutating active variables and adaptively reducing mutation strength, leading to more stable convergence in highly constrained sparse settings.

The comparison of computational time is shown in Fig. 10, where box plots illustrate the runtime distribution of each algorithm across SMOP1–SMOP8 at different dimensions. Compared to most baselines, DMKEA consistently exhibits lower runtime, especially in high-dimensional settings. It matches the efficiency of NUCEA, SECSO, and S-NSGAII–the latter two are explicitly designed for computational speed. Despite its comparable or shorter runtime, DMKEA significantly outperforms these algorithms in solution quality, highlighting its superior balance between optimization performance and computational cost.

4.2.3. *Results for practical applications*

To evaluate DMKEA in practical settings, we applied it to two real-world sparse optimization problems: Sparse_NN and Sparse_SR. In these experiments, DMKEA employs the dynamic refinement mechanism for the prior vector pv, which integrates feedback from the statistical vector sv based on the formula (6).

Sparse_NN involves training a single-hidden-layer feedforward neural network on a classification task. In our experiments, we used 80 hidden units to construct the network, resulting in $D = 1601$ decision variables, each corresponding to a weight or bias. The optimization simultaneously minimizes the training error and the number of non-zero

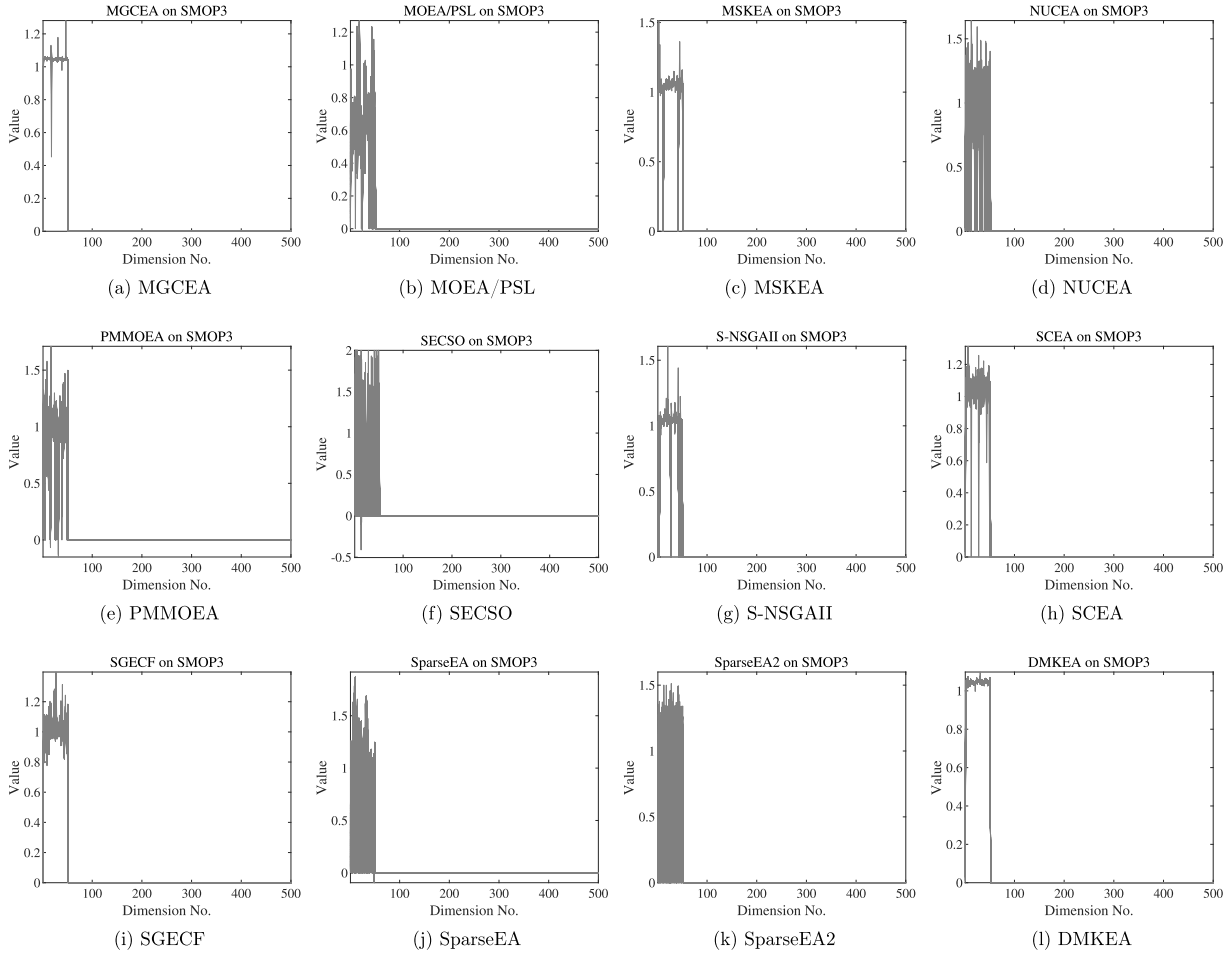


Fig. 7. Parallel coordinates of decision variables on SMOP3 ($D = 500$, $\theta = 0.1$).

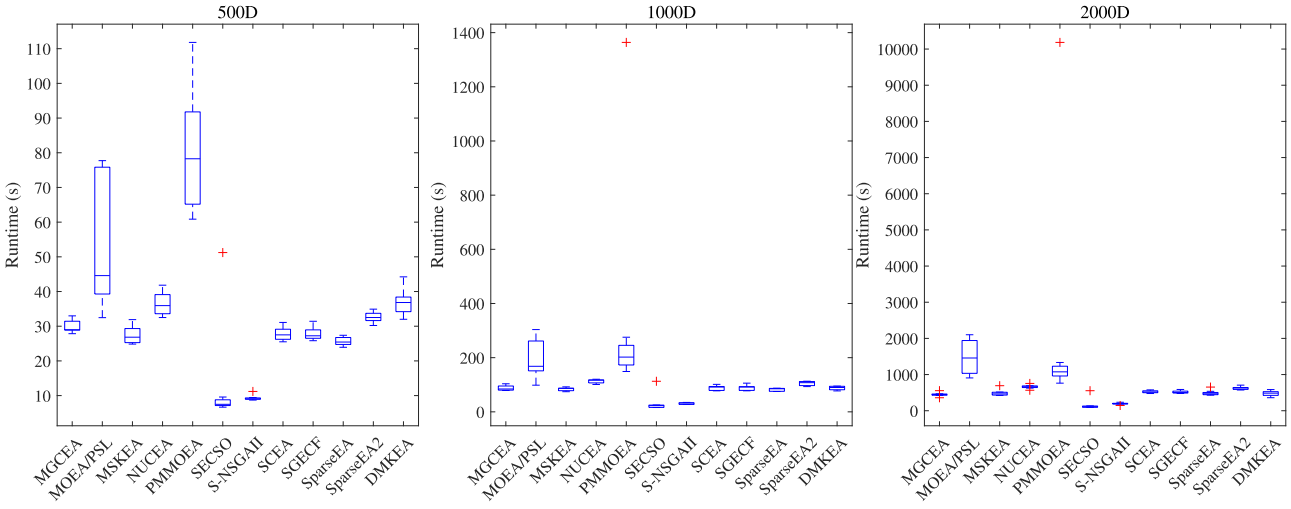


Fig. 8. Runtime comparison across dimensions at sparsity level $\theta = 0.1$. Each box plot shows the median (central line), interquartile range (box), whiskers (non-outlier range), and outliers (red crosses).

parameters to promote model sparsity. Due to the strong nonlinear dependencies among weights, the importance of each variable is highly context-dependent and can change dynamically during optimization. This makes fixed guidance less effective.

Sparse_SR is a sparse signal reconstruction problem where the goal is to recover a high-dimensional sparse signal from a limited number of linear measurements. In our setting, the signal length is $D = 1024$, and the number of observations is 480, with a known ground-truth sparsity level of 20 non-zero components. The optimization minimizes both the reconstruction error and the proportion of non-zero variables to promote sparsity. Although the non-zero variable set is fixed, accurate recovery demands high numerical precision due to the linear mixing of variable contributions by the measurement matrix. This makes it challenging to estimate variable importance during early search stages

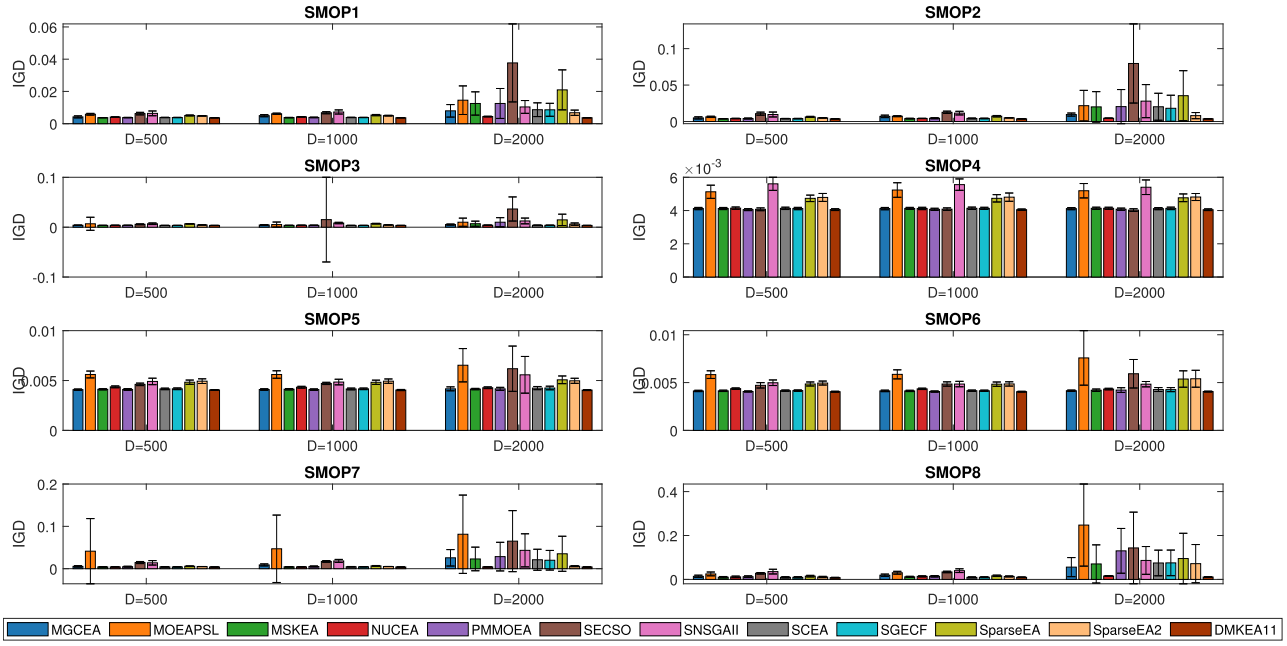


Fig. 9. Performance comparison of DMKEA and eleven baseline algorithms on SMOP1–SMOP8 at sparsity level $\theta = 0.01$ and different dimensions $D = 500, 1000, 2000$. Each subplot represents a benchmark problem. Each group of bars corresponds to one dimension, with twelve bars per group representing the twelve algorithms. Error bars indicate the standard deviation of IGD across 100 independent runs. Y-axis scales may differ across subplots for visibility.

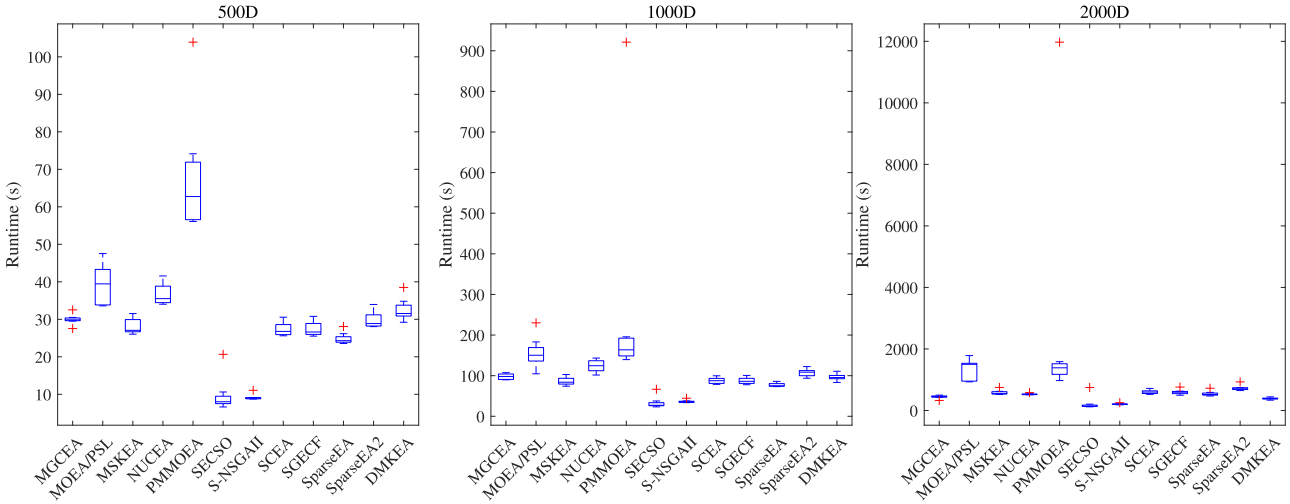


Fig. 10. Runtime comparison across dimensions at sparsity level $\theta = 0.01$. Each box plot shows the median (central line), interquartile range (box), whiskers (non-outlier range), and outliers (red crosses).

and increases the difficulty of effective guidance from initialization alone.

Since the true Pareto front is unknown for these real-world problems, we adopted the HV metric to assess solution quality. HV measures the volume of the objective space dominated by the obtained non-dominated set, bounded by a reference point (1, 1). A larger HV indicates better convergence and diversity. Each experiment was independently repeated 100 times to ensure statistical reliability.

The results, presented in Table 3, show the mean and standard deviation of the HV values achieved by each algorithm. The table also includes the Wilcoxon rank-sum test results, with the best-performing results highlighted in bold. The dynamic update of the prior vector pv enables DMKEA to better adapt to evolving variable importance, contributing to its superior performance on these challenging practical problems.

As shown in Table 3, DMKEA achieves competitive or superior performance on both real-world problems. In Sparse_SR, it outperforms all competing methods and achieves the best HV value, 2.8619e-1, with statistical significance. In Sparse_NN, it performs comparably to the best algorithms (NUCEA, SparseEA, SparseEA2), with no significant difference. This demonstrates the practical effectiveness of the proposed DMKEA on real-world sparse optimization problems.

To assess the computational efficiency of the proposed algorithm on real-world applications, we compared the runtime of all competing methods on the Sparse_NN and Sparse_SR test cases. As shown in Fig. 11, each bar reported the average runtime with standard deviation. DMKEA exhibited stable and moderate runtime in both cases. For Sparse_NN, PM-MOEA and MOEA/PSL required considerably higher runtime, with PM-MOEA also showing the largest variability. In contrast, SECSO and S-NSGAII achieved the lowest runtime. The remaining algorithms, including DMKEA, fell within a moderate range (500–600s)

Table 3
Comparison of HV values on Sparse_NN and Sparse_SR. Best values are in bold. Symbols '+', '-', and '≈' indicate significantly better, worse, or similar performance compared to DMKEA.

Problem	D	MGCEA	MOEAPSL	MSKEA	NUCEA	PMMOEA	SECSO	SNSGAII	SCEA	SGECF	SparseEA	SparseEA2	DMKEA
NN	1601	9.7594e-1 (3.34e-3) -	9.6013e-1 (1.79e-2) -	9.8127e-1 (4.67e-3) -	**9.8415e-1** **(3.36e-3)** ≈	9.4636e-1 (1.61e-2) -	9.5111e-1 (1.77e-2) -	9.6861e-1 (7.11e-3) -	9.7669e-1 (3.40e-3) -	9.7688e-1 (3.26e-3) -	9.8352e-1 (3.80e-3) ≈	9.8406e-1 (2.47e-3) ≈	9.8367e-1 (3.41e-3)
SR	1024	2.3319e-1 (2.67e-3) -	2.7630e-1 (4.52e-3) -	2.7153e-1 (4.39e-3) -	2.8157e-1 (4.78e-3) -	2.6520e-1 (4.66e-3) -	2.4427e-1 (1.16e-2) -	1.0210e-1 (1.34e-2) -	2.6284e-1 (5.76e-3) -	2.5027e-1 (2.52e-3) -	2.8346e-1 (5.26e-3) -	2.8420e-1 (4.57e-3) -	**2.8619e-1** **(4.23e-3)**
+/ − / ≈		0/2/0	0/2/0	0/2/0	0/1/1	0/2/0	0/2/0	0/2/0	0/2/0	0/2/0	0/1/1	0/1/1	

and demonstrated stable execution. For Sparse_SR, SECSO and S-NSGAII again ran the fastest, while PM-MOEA incurred the highest cost and MOEA/PSL was also relatively slow. Most other algorithms, including DMKEA, achieved runtimes around 80–100s. Overall, DMKEA delivered stable and competitive efficiency. It avoided the large overhead of PM-MOEA and MOEA/PSL while maintaining runtime comparable to most peers. At the same time, it offered a better trade-off between performance and computational cost than SECSO and S-NSGAII, which showed inferior solution quality in both cases (see Table 3).

In summary, these results demonstrate that DMKEA not only generalizes well across synthetic benchmarks but also adapts effectively to complex real-world scenarios, particularly when variable importance is context-dependent or evolves during the search.

4.2.4. Effect of dimension D and sparsity θ on runtime

To investigate how the runtime of DMKEA is affected by problem dimensionality D and sparsity level θ, we conducted experiments on SMOP1–SMOP8 under combinations of $D \in \{500, 1000, 2000\}$ and $\theta \in \{0.01, 0.02, 0.04, 0.06, 0.08, 0.1\}$. Each experiment was repeated 30 times, and the average runtime (in seconds), aggregated across all problems, is illustrated in Fig. 12.

As shown, the effect of sparsity θ on runtime is generally limited. While minor variations are observable–particularly at $D = 1000$–the overall runtime trend remains relatively stable across different sparsity levels. No consistent monotonic trend with respect to θ is observed, indicating the algorithm's stability under various sparsity regimes. These fluctuations are likely attributable to runtime-level factors such as operating system scheduling, background processes, or variations in resource allocation during parallel execution, rather than intrinsic properties of the algorithm.

In contrast, the impact of problem dimensionality D is substantially more pronounced. As D increases from 500 to 2000, the runtime grows superlinearly, reflecting a trend broadly consistent with the algorithm's theoretical time complexity of $\mathcal{O}(D^2)$ (see Section 4.5). This quadratic relationship arises from key components such as mask operations, variation operators, and environmental selection, each involving pairwise comparisons or matrix-level computations that scale with the square of the decision space size. The empirical results thus support the theoretical runtime analysis.

4.3. Parameter sensitivity analysis

4.3.1. Parameter sensitivity (τ)

This section presents a sensitivity analysis of the parameter τ, which controls the adaptive operator selection probability p. Four different values of τ were tested: 5, 10, 20, and 30. The experiments were conducted on SMOP1—SMOP8 at a sparsity level $\theta = 0.01$ with 1000 decision variables, and each setup was independently run 100 times. Statistical analysis was performed by using the Wilcoxon rank-sum test at a significance level of 0.05. Results are reported in the format $+/-/\approx$, denoting the number of problems where the setting performs significantly better/worse/similar compared to the baseline($\tau = 20$).

The results, summarized in Table 4, show that varying τ produced only minor differences in performance across the tested problems (e.g., $\tau = 10$ resulted in $0/2/6$ wins/losses/ties relative to $\tau = 20$, while the other settings remained $0/0/8$). This indicates that the algorithm is largely insensitive to the choice of τ. Based on this observation, $\tau = 20$ was selected as the default value.

4.4. Ablation studies

This section evaluates the contribution of individual components in DMKEA, specifically enhanced binary operators, selective real-valued mutation techniques, and adaptive operator selection.

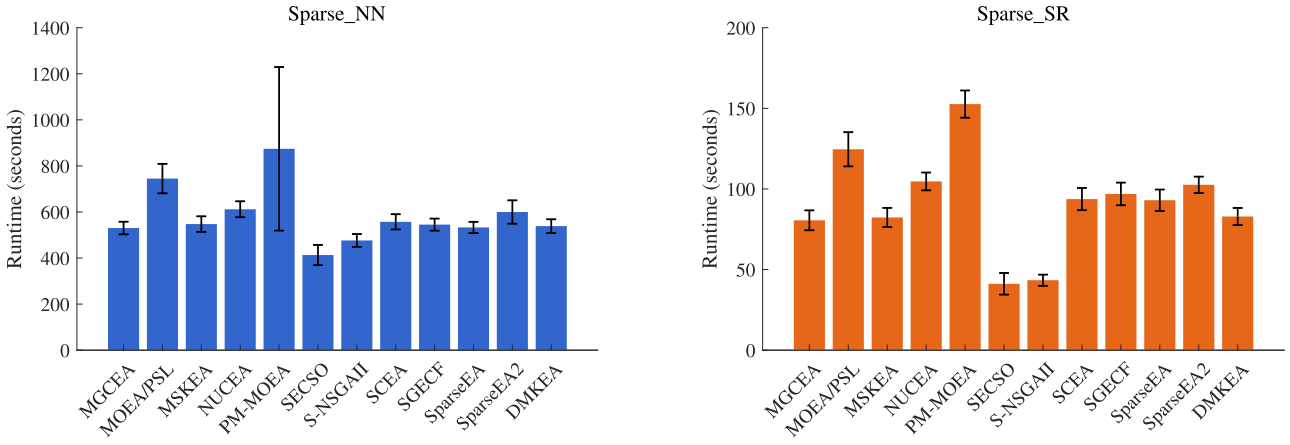


Fig. 11. Runtime comparison across real-world test cases. The left panel reports results on Sparse_NN and the right panel on Sparse_SR. Each bar represents the average runtime (seconds) over 100 independent runs, with error bars denoting standard deviation.

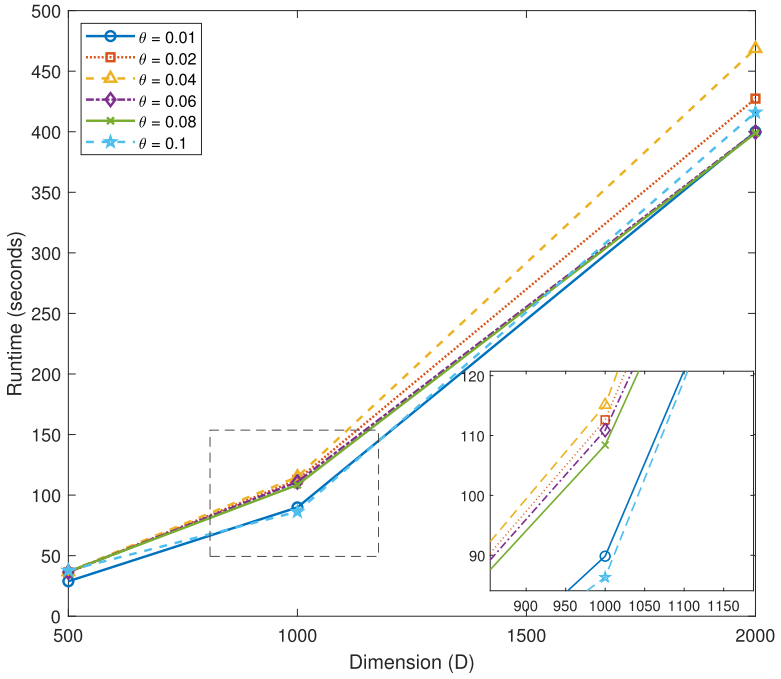


Fig. 12. Average runtime (in seconds) of DMKEA across different problem sizes and sparsity levels, aggregated over SMOP1–SMOP8.

4.4.1. Effectiveness of the enhanced binary genetic operator

To isolate and evaluate the contribution of the enhanced binary genetic operator in DMKEA, we developed a variant named DMKEA-MS. This version replaces Algorithm 2 with the original binary operator from MSKEA [25], with the proposed real-valued variation intact. This allows a controlled comparison between the new binary operator and its predecessor.

Experiments were conducted on SMOP1—SMOP8 at a sparsity level $\theta = 0.1$ with 1000 decision variables, with each configuration run independently 100 times. Statistical analysis was performed by using the Wilcoxon rank-sum test at a significance level of 0.05. The results, presented in Table 5, demonstrate that DMKEA outperformed DMKEA-MS in 4 out of 8 cases and showed statistically comparable results in the remaining 4 cases. This consistent advantage highlights the effectiveness of the enhanced binary operator to guide binary evolution more effectively under sparse constraints.

Unlike MSKEA, which applies filter-based guidance in early iterations, DMKEA reverses this order by prioritizing prior vector guidance initially and gradually incorporating filter guidance as the search progresses. This reversed strategy enables broader exploration in the early phase and preserves diversity, thus reducing the risk of premature convergence.

To complement the performance results, Fig. 13 offers a visual comparison of the structural properties of solutions produced by DMKEA and DMKEA-MS. In both problems, the Pareto-optimal solutions follow the structure $[0,1]^1 \times \{\pi/3\}^K \times \{0\}^{D-K-1}$, where $K = 0.1 \times D$. That is, only the first variable varies continuously, the next K variables are fixed at $\pi/3$, and the remaining variables are zero.

Compared to DMKEA-MS, DMKEA produces decision vectors with non-zero values more accurately concentrated within the first 100 dimensions (i.e., the sparse region), and those values are tightly clustered around $\pi/3$. In contrast, DMKEA-MS exhibits greater dispersion in both the location and magnitude of non-zero values. These observations confirm that the enhanced binary operator improves both the identification of sparse regions and the consistency of real-valued estimates through its reversed guidance schedule. This validates its design intent of promoting early exploration while maintaining convergence precision.

Table 4
Performance of DMKEA with different values of τ on SMOP1–SMOP8 ($D = 1000, \theta = 0.01$). Best values are in bold. Symbols ' + ', ' - ', and '≈' indicate significantly better, worse, or similar performance compared to DMKEA.

Problem	$\tau = 5$	$\tau = 10$	$\tau = 30$	$\tau = 20$
SMOP1	3.6444e-3 (2.54e-5) ≈	**3.6424e-3 (2.04e-5)** ≈	3.6448e-3 (2.61e-5) ≈	3.6518e-3 (7.15e-5)
SMOP2	3.6503e-3 (7.24e-5) ≈	3.6539e-3 (8.25e-5) ≈	3.6561e-3 (9.04e-5) ≈	**3.6448e-3 (5.24e-5)**
SMOP3	3.6411e-3 (1.37e-5) ≈	3.6465e-3 (2.09e-5) -	3.6412e-3 (1.49e-5) ≈	**3.6397e-3 (1.43e-5)**
SMOP4	4.0550e-3 (5.38e-5) ≈	4.0567e-3 (5.56e-5) ≈	4.0565e-3 (5.15e-5) ≈	**4.0505e-3 (4.18e-5)**
SMOP5	**4.0471e-3 (4.91e-5)** ≈	4.0619e-3 (5.50e-5) -	4.0565e-3 (4.98e-5) ≈	4.0482e-3 (5.87e-5)
SMOP6	4.0492e-3 (4.53e-5) ≈	4.0491e-3 (4.37e-5) ≈	**4.0395e-3 (4.52e-5)** ≈	4.0476e-3 (4.29e-5)
SMOP7	4.0289e-3 (8.39e-5) ≈	4.0361e-3 (8.34e-5) ≈	4.0276e-3 (8.24e-5) ≈	**4.0234e-3 (2.63e-5)**
SMOP8	9.4280e-3 (1.69e-3) ≈	1.0152e-2 (2.11e-3) ≈	**9.3786e-3 (1.97e-3)** ≈	9.6018e-3 (1.67e-3)
+/ − / ≈	0/0/8	0/2/6	0/0/8	

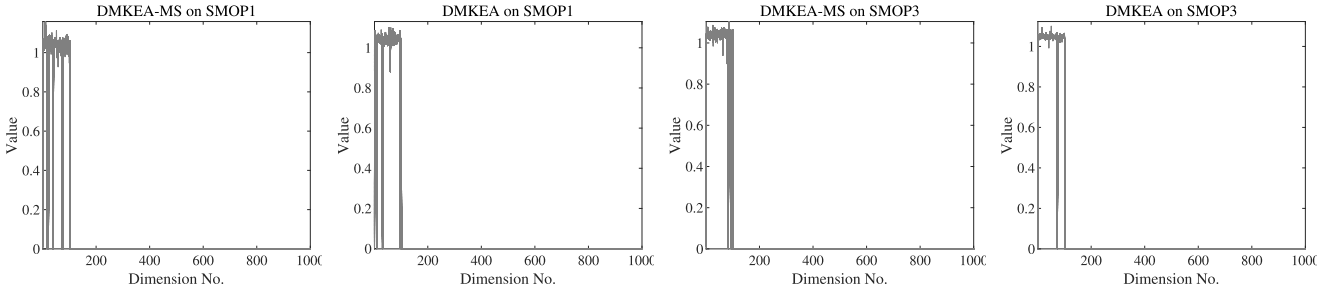


Fig. 13. Parallel coordinates of the decision variables for DMKEA-MS and DMKEA on SMOP1 (13a, 13b) and SMOP3 (13c, 13d) with 1000 variables and 0.1 sparsity level.

Table 5
Performance comparison of DMKEA-MS and DMKEA($D = 1000, \theta = 0.1$). DMKEA-MS is a variant of DMKEA where the enhanced binary operator is replaced by the original operator from MSKEA [25], with the proposed real-valued variation preserved. Best values are in bold. Symbols ' + ', ' - ', and '≈' indicate significantly better, worse, or similar performance compared to DMKEA.

Problem	DMKEA-MS	DMKEA
SMOP1	6.2041e-3 (1.80e-3) -	**5.0740e-3 (1.40e-3)**
SMOP2	1.8085e-2 (5.79e-3) -	**1.4648e-2 (3.60e-3)**
SMOP3	4.3049e-3 (7.34e-4) -	**4.0333e-3 (6.22e-4)**
SMOP4	**4.0512e-3 (4.86e-5)** ≈	4.0527e-3 (4.61e-5)
SMOP5	**4.0769e-3 (5.34e-5)** ≈	4.0907e-3 (6.96e-5)
SMOP6	4.1022e-3 (5.15e-5) ≈	**4.1010e-3 (5.00e-5)**
SMOP7	1.8097e-2 (6.69e-3) ≈	**1.6701e-2 (5.64e-3)**
SMOP8	1.4919e-1 (1.15e-2) -	**1.4032e-1 (1.16e-2)**
+/ − / ≈	0/4/4	

Table 6
Performance comparison of DMKEA with modified versions DMKEA′ and DMKEA″ on SMOP1–SMOP8 ($D = 1000, \theta = 0.01$). DMKEA′ applies PM with a fixed mutation rate of $1/D$ uniformly to all decision variables. DMKEA″ applies PM with a fixed mutation rate of $1/d$ to variables with mask value 1, where d is the number of such variables. Best values are in bold. Symbols ' + ', ' - ', and '≈' indicate significantly better, worse, or similar performance compared to DMKEA.

Problem	DMKEA′	DMKEA″	DMKEA
SMOP1	**3.6416e-3 (1.51e-5)** ≈	3.6649e-3 (2.64e-5) -	3.6518e-3 (7.15e-5)
SMOP2	3.7003e-3 (6.24e-5) -	3.6690e-3 (5.48e-5) -	**3.6448e-3 (5.24e-5)**
SMOP3	3.6405e-3 (1.46e-5) ≈	3.6613e-3 (1.76e-5) -	**3.6397e-3 (1.43e-5)**
SMOP4	4.0585e-3 (5.13e-5) ≈	4.0842e-3 (4.99e-5) -	**4.0505e-3 (4.18e-5)**
SMOP5	4.0539e-3 (4.98e-5) ≈	4.0837e-3 (5.81e-5) -	**4.0482e-3 (5.87e-5)**
SMOP6	4.0479e-3 (5.25e-5) ≈	4.0778e-3 (5.36e-5) -	**4.0476e-3 (4.29e-5)**
SMOP7	4.0958e-3 (1.20e-4) -	4.0582e-3 (8.69e-5) -	**4.0234e-3 (2.63e-5)**
SMOP8	1.0745e-2 (2.15e-3) -	9.8023e-3 (1.87e-3) ≈	**9.6018e-3 (1.67e-3)**
+/ − / ≈	0/3/5	0/7/1	

4.4.2. *Effectiveness of selective real-valued mutation strategy*

To evaluate the contribution of the selective real-valued mutation design in DMKEA, we implemented two variants: DMKEA′ and DMKEA″, each isolating a specific component of the proposed strategy.

DMKEA′ applies polynomial mutation uniformly across all decision variables, ignoring the binary mask, with a fixed mutation rate of $1/D$. This variant is designed to assess whether restricting mutation to variables with binary mask value 1 yields any performance benefit in sparse optimization tasks.

DMKEA″ improves upon DMKEA′ by applying mutation exclusively to variables with mask value 1. However, it retains a fixed mutation rate of $1/d$, where d is the number of such variables. This allows us to isolate the effect of selective mutation without incorporating adaptive rate control.

In contrast, the full DMKEA not only restricts mutation to mask-activated variables but also applies a stage-based mutation rate schedule. Specifically, the mutation rate is determined by a probability p derived from evaluation progress: when rand $< p$, the rate is set to $1/d$ (favoring stronger perturbations); otherwise, it is set to $1/D$ (enabling more conservative refinements). This design encourages broader exploration in the early stages and stable convergence later on.

Experiments were conducted on SMOP1—SMOP8 at a sparsity level $\theta = 0.01$ with 1000 decision variables, with each configuration independently repeated 100 times. Statistical significance was assessed using the Wilcoxon rank-sum test at a 0.05 level. The results in Table 6 show that DMKEA outperformed DMKEA′ in 3 out of 8 cases and matched it in the rest. This demonstrates that selective mutation–restricting perturbations to only relevant variables–provides a performance advantage over uniform mutation, even when the mutation rate remains fixed. When compared to DMKEA″, DMKEA demonstrated superior performance in 7 out of 8 cases, with statistically similar results in the remaining one. This highlights the added value of adaptively switching the mutation rate based on evaluation progress. Together, these results confirm that the synergy of selective mutation and adaptive rate scheduling plays a critical role in the success of DMKEA for sparse optimization.

Table 7
Performance of DMKEA modifications with fixed $disM$ values on SMOP1—SMOP8 ($D = 1000, \theta = 0.01$). Best values are in bold. Symbols ' + ', ' - ', and '≈' indicate significantly better, worse, or similar performance compared to DMKEA.

Problem	DMKEA-20	DMKEA-50	DMKEA-100	DMKEA
SMOP1	3.7090e-3 (2.87e-5) -	3.6885e-3 (2.60e-5) -	3.6650e-3 (1.87e-5) -	**3.6518e-3 (7.15e-5)**
SMOP2	3.8015e-3 (1.29e-4) -	3.7135e-3 (6.07e-5) -	3.7774e-3 (1.72e-4) -	**3.6448e-3 (5.24e-5)**
SMOP3	3.7051e-3 (2.62e-5) -	3.6859e-3 (2.11e-5) -	3.6605e-3 (1.82e-5) -	**3.6397e-3 (1.43e-5)**
SMOP4	4.1472e-3 (7.18e-5) -	4.1227e-3 (6.01e-5) -	4.0804e-3 (5.88e-5) -	**4.0505e-3 (4.18e-5)**
SMOP5	4.1381e-3 (7.09e-5) -	4.1243e-3 (5.84e-5) -	4.0808e-3 (6.23e-5) -	**4.0482e-3 (5.87e-5)**
SMOP6	4.1340e-3 (7.02e-5) -	4.1147e-3 (6.28e-5) -	4.0693e-3 (4.38e-5) -	**4.0476e-3 (4.29e-5)**
SMOP7	4.1699e-3 (4.48e-5) -	4.1038e-3 (9.28e-5) -	4.1683e-3 (1.64e-4) -	**4.0234e-3 (2.63e-5)**
SMOP8	**9.4565e-3 (1.87e-3)** ≈	1.0554e-2 (1.83e-3) -	1.0985e-2 (1.93e-3) -	9.6018e-3 (1.67e-3)
+/ − / ≈	0/7/1	0/8/0	0/7/1	

4.4.3. Effectiveness of adaptive parameter control

We aim to investigate whether adaptively tuning the distribution indices for crossover and mutation–rather than relying on fixed settings–can improve performance in sparse optimization. To this end, we compare the full DMKEA, which applies adaptive control to both parameters, with three modified variants that use static values.

We selected three representative values–20, 50, and 100–to explore the effect of low, medium, and high distribution pressure, corresponding to increasingly conservative variation behaviors. These values were used to construct three fixed-parameter variants: DMKEA-20, DMKEA-50, and DMKEA-100, in which both the crossover index ($disC$) and mutation index ($disM$) were set to the same fixed value.

Experiments were conducted on SMOP1—SMOP8 at a sparsity level of $\theta = 0.01$, using 1000 decision variables. Each configuration was independently repeated 100 times, and statistical significance was assessed by using the Wilcoxon rank-sum test at a 0.05 significance level. The results, summarized in Table 7, show that the fully adaptive DMKEA consistently outperformed its fixed-parameter counterparts. In each comparison, the adaptive variant achieved statistically better results in 7 or 8 out of 8 benchmark problems, with no statistically worse results.

This consistent advantage across all eight problems highlights the effectiveness of dynamically adjusting variation strength, which not only improves convergence quality but also enhances robustness across diverse problem scenarios. Moreover, the adaptive mechanism is modular and independent of the main DMKEA framework, and can be seamlessly integrated into other MOEAs to boost their performance in large-scale sparse optimization tasks.

4.4.4. Effectiveness of the adaptive operator selection mechanism

We further evaluated whether adaptively selecting between binary genetic operators improves performance over a fixed scheduling strategy. To this end, we compared DMKEA with a modified variant, DMKEA-fixed, which adopts a predefined three-phase schedule inspired by MSKEA [25]. Specifically, the fixed-phase strategy proceeds as follows: (1) in the early phase ($\delta < 0.382$), Algorithm 2 is exclusively used; (2) in the intermediate phase ($0.382 \le \delta < 0.618$), Algorithms 2 and 6 are selected with equal probability; and (3) in the late phase ($\delta \ge 0.618$), only Algorithm 6 is applied.

Experiments were conducted on SMOP1—SMOP8 at a sparsity level of $\theta = 0.01$, with 1000 decision variables. Each configuration was repeated independently 100 times. Statistical significance was assessed by using the Wilcoxon rank-sum test at a 0.05 level.

As shown in Table 8, DMKEA showed performance on par with DMKEA-fixed across the eight benchmark problems (0/0/8). This demonstrates that removing manually designed phase schedules does not compromise solution quality. At the same time, the adaptive operator selection mechanism eliminates the need for handcrafted transitions and responds to search dynamics in real time, thereby improving interpretability and design simplicity.

Importantly, the adaptive mechanism is lightweight and fully modular, integrating seamlessly into the DMKEA framework with minimal parameter overhead. The only additional parameter, τ, has negligible impact on performance (see Section 4.3.1), reducing the burden of tuning. Its generality also facilitates easy integration into other multi-operator evolutionary algorithms. Overall, DMKEA is comparable in performance to carefully designed fixed-phase schedules and also offers a more flexible, robust, and extensible solution.

Table 8
Comparison of DMKEA and DMKEA-fixed on SMOP1—SMOP8 ($D = 1000, \theta = 0.01$). Best values are in bold. Symbols ' + ', ' - ', and '≈' indicate significantly better, worse, or similar performance compared to DMKEA.

Problem	DMKEA-fixed	DMKEA
SMOP1	**3.6445e-3 (2.18e-5)** ≈	3.6518e-3 (7.15e-5)
SMOP2	3.6514e-3 (7.46e-5) ≈	**3.6448e-3 (5.24e-5)**
SMOP3	3.6401e-3 (1.48e-5) ≈	**3.6397e-3 (1.43e-5)**
SMOP4	4.0589e-3 (5.14e-5) ≈	**4.0505e-3 (4.18e-5)**
SMOP5	4.0498e-3 (4.94e-5) ≈	**4.0482e-3 (5.87e-5)**
SMOP6	4.0520e-3 (4.02e-5) ≈	**4.0476e-3 (4.29e-5)**
SMOP7	4.0236e-3 (2.46e-5) ≈	**4.0234e-3 (2.63e-5)**
SMOP8	**9.2767e-3 (1.72e-3)** ≈	9.6018e-3 (1.67e-3)
+/ − / ≈	0/0/8	

4.5. Computational complexity analysis

The computational complexity of DMKEA depends on the number of decision variables D, population size N, and multi-interval sampling parameters $nInt$ and $nSamp$. The analysis is performed under a fixed fitness evaluation budget of $150D$.

Initialization. For each of the D decision variables, multi-interval sampling constructs $nInt \cdot nSamp$ binary vectors. Each vector involves $O(D)$ operations for vector construction and an additional $O(D)$ for fitness evaluation. Thus, the total initialization complexity is:

$$O\left(D^2 \cdot nInt \cdot nSamp\right).$$

Additionally, forming the initial population requires $O(ND)$, resulting in a combined initialization complexity of:

$$O(D^2 \cdot nInt \cdot nSamp + ND).$$

Main evolutionary loop. Each generation involves knowledge update and parent selection ($O(N)$), binary and real-valued variation operators ($O(ND)$), and environmental selection via NSGA-II ($O(N^2)$). Therefore, the complexity per generation is:

$$O(N^2 + ND).$$

Given that initialization consumes $D \cdot nInt \cdot nSamp$ evaluations, the remaining generations allowed under the fixed evaluation budget ($150D$) are:

$$T = \frac{150D - D \cdot nInt \cdot nSamp}{N}.$$

Thus, the evolutionary loop's total complexity is:

$$O\left(\frac{150D - D \cdot nInt \cdot nSamp}{N} \cdot (N^2 + ND)\right)$$
$$= O\big(D \cdot (150 - nInt \cdot nSamp) \cdot (N + D)\big).$$

Overall complexity. Combining initialization and main loop complexities yields the overall complexity as:

$$O\big(D^2 \cdot nInt \cdot nSamp + ND + D \cdot (150 - nInt \cdot nSamp) \cdot (N + D)\big).$$

This expression captures how computational expense scales with dimensionality, population size, and initialization parameters under the fixed evaluation limit.

Dominant term analysis. Depending on the relative magnitudes of D and N, the dominant complexity terms are:

- When $D \gg N$ (typical for large-scale sparse problems): The complexity simplifies to $O(D^2 \cdot nInt \cdot nSamp + D^2)$, dominated by $O(D^2)$.
- When $N \approx D$: Both quadratic terms are significant, again yielding complexity $O(D^2)$.
- When $N \gg D$: Environmental selection or population-wide variation dominate, leading to a complexity of $O(N^2)$.

Considering our problem context (large-scale sparse optimization with typically $D \gg N$), the overall computational complexity is effectively dominated by $O(D^2)$.

5. Conclusions

This paper proposed DMKEA, a knowledge-guided evolutionary algorithm tailored for solving large-scale multi-objective optimization problems with sparse Pareto-optimal solutions. DMKEA integrates three key components: a multi-interval sampling-based initialization strategy for estimating variable importance, a set of binary and real-valued variation operators guided by knowledge vectors, and a probabilistic mechanism for operator selection that adapts over the course of evolution. These elements collectively enable DMKEA to effectively identify relevant variables, exploit sparsity, and maintain a balance between exploration and exploitation in high-dimensional search spaces.

Extensive experiments were conducted on eight benchmark problems (SMOP1–SMOP8) across multiple dimensionalities and sparsity levels. The results demonstrated that DMKEA consistently outperformed eleven state-of-the-art algorithms, particularly in scenarios with extreme sparsity ($\theta = 0.01$). Its superior performance was reflected not only in mean IGD values but also in the robustness and stability of results across multiple runs. For example, DMKEA achieved up to 15.5 % improvement over the second-best algorithm on SMOP8 (D = 500), with other notable gains including 17.3 % on SMOP1 (D = 2000), 17.0 % on SMOP2 (D = 2000), and over 10 % on SMOP3 (D = 2000), highlighting its effectiveness in highly sparse settings. The ablation study further confirmed the contribution of each algorithmic component, including the multi-interval sampling strategy and the adaptive use of guidance vectors. To evaluate practical applicability, DMKEA was also tested on two real-world sparse optimization problems: Sparse_NN and Sparse_SR. DMKEA demonstrated strong performance on both tasks, significantly outperforming all competitors on Sparse_SR and remaining highly competitive on Sparse_NN, underscoring its adaptability in practical scenarios. In summary, DMKEA generalizes well across both synthetic benchmarks and real-world sparse problems. Its ability to dynamically guide the search process using data-driven knowledge makes it a promising approach for complex large-scale optimization scenarios.

5.1. Limitations and future work

While DMKEA demonstrates strong performance on sparse problems with two objectives, its scalability to many-objective problems ($M > 2$) remains an open question. The current guidance mechanisms and selection pressure may become less effective as the number of objectives increases, due to the increasing difficulty of preserving diversity and convergence simultaneously. Furthermore, although the algorithm is designed for sparse settings, its behavior under dense or mixed-sparsity scenarios has not been thoroughly investigated. Future work could explore extending DMKEA to handle many-objective problems by incorporating decomposition strategies or reference vector adaptation. Additionally, integrating variable grouping or structure learning could improve performance on problems with strong variable dependencies. An additional direction worth exploring is the adaptation of the framework to dynamic or streaming optimization contexts.

CRediT authorship contribution statement

Lidan Bai: Writing – original draft, Software, Methodology, Conceptualization; **Jun Sun:** Writing – review & editing, Validation, Supervision, Funding acquisition; **Chao Li:** Writing – review & editing; **Hengyang Lu:** Methodology, Funding acquisition; **Vasile Palade:** Writing – review & editing.

Data availability

The source code is available at https://github.com/xiaobai3/DMKEA.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by theNational Key Research and Development Program of China (Grant no. 2020YFA0908303), the Natural Science Foundation of Jiangsu Province (Grant no. BK20221068), and the National Natural Science Foundation of China (Grants no. 62272202, 61672263).

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.knosys.2025.114764

Appendix A. Prior vector strategy comparison

Table A.1 compares DMKEA with static and dynamic prior vector (pv) strategies across all SMOP benchmark problems under different dimensionalities. The results show that static pv generally performs comparably or better than the dynamic version on synthetic problems, supporting the decision to omit pv updates in these settings.

Table A.1

Performance comparison between DMKEA+ (dynamic pv) and DMKEA (static pv) on SMOP1–SMOP8 under sparsity level $\theta = 0.1$ and three dimensions $D = 500, 1000,$ and 2000. Values are mean IGD with standard deviation in parentheses, averaged over 100 independent runs. " + ", " - ", and "≈" indicate that DMKEA+ performs significantly better, worse, or equivalent to DMKEA according to the Wilcoxon signed-rank test at the 0.05 significance level. The best result in each row is highlighted in bold.

Problem	D	DMKEA+	DMKEA
SMOP1	500	3.9177e-3 (6.52e-4) ≈	**3.8702e-3 (5.14e-4)**
SMOP2		7.6469e-3 (2.71e-3) ≈	**7.4385e-3 (2.31e-3)**
SMOP3		3.6812e-3 (1.30e-4) ≈	**3.6803e-3 (1.32e-4)**
SMOP4		**4.0545e-3 (5.41e-5)** ≈	4.0602e-3 (5.51e-5)
SMOP5		4.0688e-3 (5.14e-5) ≈	**4.0681e-3 (5.57e-5)**
SMOP6		4.0924e-3 (5.47e-5) ≈	**4.0889e-3 (5.80e-5)**
SMOP7		8.6613e-3 (3.44e-3) -	**7.4730e-3 (2.47e-3)**
SMOP8		**1.2390e-1 (1.40e-2)** ≈	1.2424e-1 (1.33e-2)
SMOP1	1000	5.3158e-3 (1.27e-3) -	**5.0740e-3 (1.40e-3)**
SMOP2		1.6411e-2 (3.84e-3) -	**1.4648e-2 (3.60e-3)**
SMOP3		4.0997e-3 (6.54e-4) ≈	**4.0333e-3 (6.22e-4)**
SMOP4		4.0533e-3 (4.84e-5) ≈	**4.0527e-3 (4.61e-5)**
SMOP5		4.0950e-3 (6.50e-5) ≈	**4.0907e-3 (6.96e-5)**
SMOP6		4.1208e-3 (5.89e-5) -	**4.1010e-3 (5.00e-5)**
SMOP7		1.8425e-2 (6.58e-3) ≈	**1.6701e-2 (5.64e-3)**
SMOP8		1.4332e-1 (9.88e-3) -	**1.4032e-1 (1.16e-2)**
SMOP1	2000	9.5997e-3 (2.72e-3) -	**8.6455e-3 (2.52e-3)**
SMOP2		2.8709e-2 (5.94e-3) -	**2.6579e-2 (5.53e-3)**
SMOP3		6.3682e-3 (1.83e-3) -	**5.5794e-3 (1.38e-3)**
SMOP4		4.0491e-3 (5.50e-5) ≈	**4.0476e-3 (5.53e-5)**
SMOP5		**4.1319e-3 (1.13e-4)** +	4.1469e-3 (5.81e-5)
SMOP6		4.1822e-3 (9.66e-5) -	**4.1137e-3 (5.88e-5)**
SMOP7		3.2679e-2 (6.95e-3) ≈	**3.1329e-2 (7.20e-3)**
SMOP8		1.6656e-1 (1.11e-2) -	**1.6295e-1 (1.02e-2)**
+/-/≈		1/10/13	

Appendix B. Detailed results under moderate sparsity ($\theta = 0.1$)

Appendix C. Detailed results under high sparsity ($\theta = 0.01$)

Table B.1

Comparison of IGD values on SMOP benchmarks at sparsity level $\theta = 0.1$. Best values are in bold. Symbols ' + ', ' - ', and '≈' indicate significantly better, worse, or similar performance compared to DMKEA.

SMOPsuite	D	MGCEA	MOEA/PSL	MSKEA	NUCEA	PM-MOEA	SECSO	S-NSGAII	SCEA	SGECF	SparseEA	SparseEA2	DMKEA
1	500	4.0621e-3 (1.02e-3) -	1.6672e-2 (2.73e-3) -	7.2304e-3 (2.44e-3) -	4.7532e-3 (3.11e-4) -	1.0910e-2 (2.73e-3) -	5.1115e-2 (3.92e-3) -	8.1379e-3 (2.80e-3) -	5.3765e-3 (1.22e-3) -	5.2083e-3 (1.03e-3) -	1.6217e-2 (2.88e-3) -	6.3292e-3 (3.82e-4) -	**3.8702e-3 (5.14e-4)**
	1000	5.5553e-3 (2.35e-3) ≈	1.9112e-2 (1.30e-3) -	1.2045e-2 (2.55e-3) -	2.4569e-2 (3.03e-2) -	1.6310e-2 (2.70e-3) -	5.4595e-2 (3.05e-3) -	1.2396e-2 (4.01e-3) -	7.7490e-3 (1.75e-3) -	7.6458e-3 (1.66e-3) -	2.3495e-2 (2.84e-3) -	6.6165e-3 (3.13e-4) -	**5.0740e-3 (1.40e-3)**
	2000	1.0292e-2 (3.35e-3) -	2.3997e-2 (1.17e-3) -	1.7951e-2 (2.15e-3) -	**4.9741e-3 (3.94e-4)** +	2.2550e-2 (2.50e-3) -	5.6178e-2 (2.34e-3) -	1.5906e-2 (4.47e-3) -	1.1643e-2 (1.56e-3) -	1.1542e-2 (1.59e-3) -	3.1126e-2 (2.24e-3) -	8.0524e-3 (8.12e-4) ≈	8.6455e-3 (2.52e-3)
2	500	**3.9002e-3 (5.14e-4)** +	3.6534e-2 (5.38e-3) -	2.0966e-2 (5.31e-3) -	5.7827e-3 (1.88e-3) +	2.9151e-2 (5.55e-3) -	1.1366e-1 (6.29e-3) -	2.6274e-2 (1.08e-2) -	1.9697e-2 (5.42e-3) -	1.8874e-2 (4.69e-3) -	4.3871e-2 (6.24e-3) -	7.5696e-3 (1.92e-3) ≈	7.4385e-3 (2.31e-3)
	1000	**5.9805e-3 (1.67e-2)** +	4.4818e-2 (4.11e-3) -	3.4184e-2 (5.51e-3) -	4.9796e-2 (6.51e-2) -	4.3025e-2 (5.70e-3) -	1.1972e-1 (5.30e-3) -	4.4365e-2 (1.36e-2) -	2.8601e-2 (4.61e-3) -	2.9127e-2 (4.41e-3) -	6.2394e-2 (5.54e-3) -	9.6435e-3 (1.72e-3) +	1.4648e-2 (3.60e-3)
	2000	**7.7787e-3 (1.94e-3)** +	5.3919e-2 (4.13e-3) -	4.7390e-2 (4.75e-3) -	9.1489e-3 (1.52e-3) +	5.7212e-2 (4.79e-3) -	1.2271e-1 (4.12e-3) -	5.9799e-2 (1.25e-2) -	4.1502e-2 (4.13e-3) -	4.1221e-2 (4.13e-3) -	7.9275e-2 (4.23e-3) -	1.3875e-2 (1.74e-3) +	2.6579e-2 (5.53e-3)
3	500	3.7059e-3 (2.16e-5) -	1.6833e-2 (6.80e-3) -	5.2335e-3 (1.58e-3) -	4.3831e-3 (5.18e-4) -	1.9003e-2 (7.36e-2) -	5.1263e-2 (3.84e-3) -	1.2701e-2 (3.93e-3) -	3.9678e-3 (3.71e-4) -	3.9751e-3 (4.95e-4) -	1.7273e-2 (2.80e-3) -	6.6834e-3 (9.70e-4) -	**3.6803e-3 (1.32e-4)**
	1000	**3.7011e-3 (1.91e-5)** +	2.0113e-2 (8.33e-3) -	8.6648e-3 (2.16e-3) -	2.4305e-2 (3.07e-2) -	3.9305e-2 (1.31e-1) -	5.4644e-2 (3.39e-3) -	1.7788e-2 (4.46e-3) -	4.0593e-3 (4.71e-4) -	4.0886e-3 (4.97e-4) -	2.4469e-2 (2.77e-3) -	6.6516e-3 (7.26e-4) -	4.0333e-3 (6.22e-4)
	2000	**3.7020e-3 (2.10e-5)** +	2.2630e-2 (5.44e-3) -	1.4595e-2 (1.94e-3) -	4.2048e-3 (8.67e-5) +	3.0109e-2 (7.67e-2) -	5.6393e-2 (2.12e-3) -	2.1532e-2 (4.93e-3) -	4.3245e-3 (4.99e-4) +	4.2969e-3 (4.22e-4) +	3.2925e-2 (2.29e-3) -	9.8843e-3 (1.25e-3) -	5.5794e-3 (1.38e-3)
4	500	4.1207e-3 (7.02e-5) -	5.3022e-3 (5.28e-4) -	4.1280e-3 (6.46e-5) -	4.1319e-3 (6.77e-5) -	4.0664e-3 (6.36e-5) ≈	**3.9962e-3 (7.03e-5)** +	4.9838e-3 (3.13e-4) -	4.1287e-3 (7.23e-5) -	4.1398e-3 (6.04e-5) -	4.7273e-3 (2.13e-4) -	4.7443e-3 (2.13e-4) -	4.0602e-3 (5.51e-5)
	1000	4.1270e-3 (7.10e-5) -	5.2611e-3 (4.82e-4) -	4.1252e-3 (6.88e-5) -	4.1244e-3 (7.20e-5) -	4.0639e-3 (6.57e-5) ≈	**3.9981e-3 (7.40e-5)** +	4.9237e-3 (3.27e-4) -	4.1401e-3 (6.05e-5) -	4.1214e-3 (5.91e-5) -	4.7767e-3 (2.51e-4) -	4.7513e-3 (2.39e-4) -	4.0527e-3 (4.61e-5)
	2000	4.1325e-3 (5.76e-5) -	5.3216e-3 (5.41e-4) -	4.1370e-3 (6.16e-5) -	4.1228e-3 (6.86e-5) -	4.0688e-3 (6.09e-5) -	**3.9911e-3 (6.42e-5)** +	4.9538e-3 (2.95e-4) -	4.1335e-3 (7.18e-5) -	4.1357e-3 (7.74e-5) -	4.7518e-3 (2.40e-4) -	4.7694e-3 (2.49e-4) -	4.0476e-3 (5.53e-5)
5	500	4.2545e-3 (9.76e-5) -	7.4027e-3 (4.67e-4) -	4.1467e-3 (6.57e-5) -	4.7907e-3 (1.56e-4) -	4.4414e-3 (1.36e-4) -	1.0458e-2 (7.59e-4) -	4.7441e-3 (2.44e-4) -	4.6543e-3 (1.52e-4) -	4.6548e-3 (1.41e-4) -	5.5508e-3 (2.56e-4) -	5.2769e-3 (2.33e-4) -	**4.0681e-3 (5.57e-5)**
	1000	4.3671e-3 (2.53e-3) -	8.9306e-3 (4.79e-4) -	4.1406e-3 (6.15e-5) -	7.8951e-3 (5.36e-3) -	4.3548e-3 (1.06e-4) -	1.0162e-2 (9.79e-4) -	4.7630e-3 (3.00e-4) -	4.4768e-3 (1.02e-4) -	4.4872e-3 (1.11e-4) -	5.4870e-3 (2.58e-4) -	5.2974e-3 (2.57e-4) -	**4.0907e-3 (6.96e-5)**
	2000	**4.1168e-3 (6.50e-5)** +	8.9494e-3 (4.11e-4) -	4.1498e-3 (6.45e-5) ≈	4.6317e-3 (1.06e-4) -	4.3851e-3 (8.44e-5) -	9.4265e-3 (1.32e-3) -	7.9231e-3 (3.13e-3) -	4.4718e-3 (7.98e-5) -	4.4652e-3 (8.03e-5) -	5.5127e-3 (2.63e-4) -	5.2023e-3 (2.30e-4) -	4.1469e-3 (5.81e-5)
6	500	4.1561e-3 (7.13e-5) -	8.3460e-3 (5.00e-4) -	4.2697e-3 (7.02e-5) -	5.3626e-3 (2.27e-4) -	4.5182e-3 (1.29e-4) -	9.5270e-3 (1.01e-3) -	4.7448e-3 (2.24e-4) -	4.5994e-3 (1.41e-4) -	4.5757e-3 (1.21e-4) -	6.3762e-3 (3.73e-4) -	6.3276e-3 (3.31e-4) -	**4.0889e-3 (5.80e-5)**
	1000	4.1457e-3 (6.61e-5) -	1.0660e-2 (8.52e-4) -	4.3004e-3 (7.95e-5) -	6.6713e-3 (2.71e-3) -	4.4847e-3 (9.71e-5) -	8.5547e-3 (8.64e-4) -	4.8037e-3 (2.31e-4) -	4.5449e-3 (1.03e-4) -	4.5444e-3 (1.02e-4) -	6.4089e-3 (2.96e-4) -	6.5307e-3 (2.95e-4) -	**4.1010e-3 (5.00e-5)**
	2000	4.1581e-3 (6.43e-5) -	1.1759e-2 (7.28e-4) -	4.3606e-3 (7.75e-5) -	5.9188e-3 (2.38e-4) -	4.5581e-3 (9.15e-5) -	8.0452e-3 (6.52e-4) -	4.7929e-3 (2.00e-4) -	4.5642e-3 (7.49e-5) -	4.5565e-3 (8.93e-5) -	6.5907e-3 (2.68e-4) -	6.7006e-3 (2.99e-4) -	**4.1137e-3 (5.88e-5)**
7	500	1.5010e-2 (7.84e-3) -	9.9719e-2 (9.64e-2) -	2.9335e-2 (9.96e-3) -	**4.7110e-3 (6.78e-4)** +	4.5865e-2 (1.10e-2) -	1.6598e-1 (9.45e-3) -	4.4801e-2 (1.98e-2) -	2.9884e-2 (7.40e-3) -	2.9827e-2 (6.97e-3) -	5.7590e-2 (9.14e-3) -	6.0348e-3 (9.92e-4) +	7.4730e-3 (2.47e-3)
	1000	3.2384e-2 (1.15e-2) -	1.0705e-1 (8.93e-2) -	4.7264e-2 (7.58e-3) -	6.7323e-2 (9.38e-2) -	6.0388e-2 (8.40e-3) -	1.7159e-1 (6.25e-3) -	7.4555e-2 (2.38e-2) -	4.0698e-2 (7.05e-3) -	4.0864e-2 (6.78e-3) -	7.8369e-2 (9.08e-3) -	**7.8996e-3 (1.29e-3)** +	1.6701e-2 (5.64e-3)
	2000	5.2431e-2 (1.40e-2) -	1.0270e-1 (8.20e-2) -	6.4932e-2 (6.10e-3) -	**5.8631e-3 (8.27e-4)** +	7.8038e-2 (6.79e-3) -	1.7400e-1 (5.14e-3) -	9.2411e-2 (2.32e-2) -	5.8450e-2 (5.84e-3) -	5.7199e-2 (5.37e-3) -	9.8406e-2 (6.49e-3) -	7.3562e-3 (6.76e-4) +	3.1329e-2 (7.20e-3)
8	500	**1.0816e-1 (1.14e-2)** +	2.5991e-1 (3.35e-2) -	1.4642e-1 (1.70e-2) -	1.4503e-1 (1.53e-2) -	1.6533e-1 (1.79e-2) -	3.8366e-1 (1.32e-2) -	1.3981e-1 (2.24e-2) -	1.1406e-1 (1.22e-2) +	1.2004e-1 (1.44e-2) ≈	2.0078e-1 (1.86e-2) -	1.6939e-1 (1.37e-2) -	1.2424e-1 (1.33e-2)
	1000	**1.1234e-1 (8.46e-3)** +	3.4864e-1 (2.53e-2) -	1.7315e-1 (1.33e-2) -	2.5427e-1 (1.41e-1) -	1.8463e-1 (1.37e-2) -	3.8769e-1 (1.56e-2) -	1.6634e-1 (2.04e-2) -	1.2063e-1 (9.58e-3) +	1.2005e-1 (1.05e-2) +	2.3288e-1 (1.66e-2) -	1.8930e-1 (1.18e-2) -	1.4032e-1 (1.16e-2)
	2000	**1.2119e-1 (6.69e-3)** +	4.0800e-1 (2.00e-2) -	1.9947e-1 (1.24e-2) -	1.7834e-1 (1.50e-2) -	2.1924e-1 (1.41e-2) -	3.9395e-1 (1.40e-2) -	1.8004e-1 (1.75e-2) -	1.2658e-1 (7.18e-3) +	1.2907e-1 (8.17e-3) +	2.6892e-1 (1.36e-2) -	2.0406e-1 (7.04e-3) -	1.6295e-1 (1.02e-2)
+/ − / ≈		9/14/1	0/24/0	0/23/1	6/18/0	0/22/2	3/21/0	0/24/0	4/20/0	3/20/1	0/24/0	5/17/2	

Table C.1
Comparison of IGD values on SMOP benchmarks at sparsity level $\theta = 0.01$. Best values are in bold. Symbols ' + ', ' - ', and '≈' indicate significantly better, worse, or similar performance compared to DMKEA.

SMOPsuite	D	MGCEA	MOEA/	PSL	MSKEA	NUCEA	PM-	MOEA	SECSO	S-NSGAII	SCEA	SGECF	SparseEA	SparseEA2	DMKEA
1	500	4.2104e-3 (6.16e-4) -	5.8329e-3 (5.47e-4) -	3.7120e-3 (8.44e-5) -	4.1140e-3 (8.89e-5) -	3.7800e-3 (1.33e-4) -	6.2376e-3 (7.91e-4) -	6.3874e-3 (1.48e-3) -	3.8334e-3 (1.12e-4) -	3.8237e-3 (1.02e-4) -	5.0827e-3 (2.86e-4) -	4.8578e-3 (1.61e-4) -	**3.6415e-3 (1.59e-5)**
	1000	4.9543e-3 (6.66e-4) -	6.1372e-3 (3.90e-4) -	3.7624e-3 (1.42e-4) -	4.2117e-3 (1.27e-4) -	3.8635e-3 (1.80e-4) -	6.7788e-3 (6.38e-4) -	7.2524e-3 (1.28e-3) -	3.8542e-3 (1.12e-4) -	3.8414e-3 (8.98e-5) -	5.3193e-3 (3.93e-4) -	4.9763e-3 (2.17e-4) -	**3.6518e-3 (7.15e-5)**
	2000	7.9580e-3 (3.88e-3) -	1.4556e-2 (8.87e-3) -	1.2522e-2 (7.22e-3) -	4.4021e-3 (2.08e-4) -	1.2528e-2 (9.29e-3) -	3.7683e-2 (2.42e-2) -	1.0373e-2 (3.97e-3) -	8.6573e-3 (4.26e-3) -	8.5994e-3 (4.02e-3) -	2.0962e-2 (1.24e-2) -	6.7799e-3 (1.63e-3) -	**3.6665e-3 (6.61e-5)**
2	500	4.9612e-3 (1.47e-3) -	6.3487e-3 (8.85e-4) -	3.8030e-3 (3.24e-4) -	4.1284e-3 (3.19e-4) -	4.0861e-3 (6.76e-4) -	1.0781e-2 (2.10e-3) -	9.6034e-3 (3.37e-3) -	3.9591e-3 (2.84e-4) -	3.9774e-3 (3.48e-4) -	6.2900e-3 (7.87e-4) -	4.9560e-3 (2.91e-4) -	**3.6434e-3 (1.67e-5)**
	1000	7.0463e-3 (1.69e-3) -	7.2577e-3 (6.54e-4) -	4.0017e-3 (4.62e-4) -	4.2074e-3 (2.08e-4) -	4.4641e-3 (7.23e-4) -	1.2624e-2 (1.59e-3) -	1.1512e-2 (2.49e-3) -	4.2201e-3 (5.20e-4) -	4.1738e-3 (4.12e-4) -	7.2982e-3 (9.88e-4) -	5.0904e-3 (2.85e-4) -	**3.6448e-3 (5.24e-5)**
	2000	9.4965e-3 (2.13e-3) -	2.1725e-2 (2.10e-2) -	2.0077e-2 (2.09e-2) -	4.4941e-3 (3.38e-4) -	2.0442e-2 (2.34e-2) -	7.9625e-2 (5.43e-2) -	2.7957e-2 (2.27e-2) -	2.0240e-2 (1.85e-2) -	1.8137e-2 (1.79e-2) -	3.5431e-2 (3.43e-2) -	8.0668e-3 (4.22e-3) -	**3.7341e-3 (2.23e-4)**
3	500	3.9533e-3 (3.78e-4) -	6.8899e-3 (1.31e-2) -	3.7011e-3 (5.09e-5) -	3.8392e-3 (5.76e-5) -	3.8085e-3 (1.48e-4) -	5.8416e-3 (8.57e-4) -	6.8963e-3 (1.61e-3) -	3.6523e-3 (5.28e-5) -	3.6535e-3 (4.91e-5) -	6.7907e-3 (4.00e-4) -	4.5056e-3 (1.11e-4) -	**3.6431e-3 (1.50e-5)**
	1000	4.4731e-3 (5.56e-4) -	5.3101e-3 (5.08e-3) -	3.7320e-3 (1.02e-4) -	3.8909e-3 (9.79e-5) -	3.8655e-3 (1.72e-4) -	1.5232e-2 (8.50e-2) -	8.4104e-3 (1.43e-3) -	3.6508e-3 (3.19e-5) -	3.6529e-3 (3.67e-5) -	7.1375e-3 (6.35e-4) -	4.5821e-3 (1.52e-4) -	**3.6397e-3 (1.43e-5)**
	2000	5.2748e-3 (1.11e-3) -	1.0023e-2 (8.22e-3) -	7.0283e-3 (5.13e-3) -	4.0442e-3 (1.13e-4) -	1.0015e-2 (9.15e-3) -	3.6584e-2 (2.42e-2) -	1.2562e-2 (5.78e-3) -	3.8884e-3 (4.92e-4) -	3.8521e-3 (3.55e-4) -	1.4754e-2 (1.14e-2) -	6.1843e-3 (2.36e-3) -	**3.6415e-3 (1.66e-5)**
4	500	4.1203e-3 (6.82e-5) -	5.1288e-3 (3.96e-4) -	4.1215e-3 (6.47e-5) -	4.1442e-3 (6.85e-5) -	**4.0544e-3 (5.90e-5)** ≈	4.0747e-3 (8.43e-5) ≈	5.6071e-3 (3.94e-4) -	4.1314e-3 (7.03e-5) -	4.1212e-3 (7.00e-5) -	4.7336e-3 (1.94e-4) -	4.7885e-3 (2.39e-4) -	4.0546e-3 (5.49e-5)
	1000	4.1168e-3 (6.94e-5) -	4.4386e-4) -	4.1272e-3 (6.32e-5) -	4.1249e-3 (7.03e-5) -	4.0720e-3 (6.49e-5) -	4.0821e-3 (7.68e-5) -	4.0821e-3 (3.43e-4) -	4.1358e-3 (7.10e-5) -	4.1305e-3 (6.55e-5) -	4.7350e-3 (2.21e-4) -	4.8051e-3 (2.49e-4) -	**4.0505e-3 (4.18e-5)**
	2000	4.1150e-3 (6.45e-5) -	5.1916e-3 (4.33e-4) -	4.1351e-3 (6.64e-5) -	4.1357e-3 (6.32e-5) -	4.0812e-3 (6.51e-5) -	**4.0264e-3 (8.05e-5)** +	5.4018e-3 (4.39e-4) -	4.1153e-3 (6.43e-5) -	4.1300e-3 (7.47e-5) -	4.7667e-3 (2.32e-4) -	4.8139e-3 (2.11e-4) -	4.0487e-3 (5.09e-5)
5	500	4.0843e-3 (6.52e-5) -	5.6061e-3 (3.43e-4) -	4.1224e-3 (6.04e-5) -	4.3661e-3 (9.08e-5) -	4.1028e-3 (8.76e-5) -	4.6154e-3 (1.25e-4) -	4.9093e-3 (3.24e-4) -	4.1601e-3 (7.63e-5) -	4.1711e-3 (8.39e-5) -	4.8326e-3 (2.14e-4) -	4.9361e-3 (2.34e-4) -	**4.0546e-3 (4.60e-5)**
	1000	4.0971e-3 (7.79e-5) -	5.6024e-3 (3.74e-4) -	4.1306e-3 (7.08e-5) -	4.3284e-3 (9.15e-5) -	4.0890e-3 (7.17e-5) -	4.7097e-3 (1.12e-4) -	4.8475e-3 (2.81e-4) -	4.1550e-3 (8.63e-5) -	4.1662e-3 (7.86e-5) -	4.8248e-3 (2.12e-4) -	4.9372e-3 (2.19e-4) -	**4.0482e-3 (5.87e-5)**
	2000	4.1669e-3 (1.98e-4) -	6.5333e-3 (1.67e-3) -	4.1452e-3 (6.85e-5) -	4.2760e-3 (8.60e-5) -	4.1673e-3 (1.50e-4) -	6.1845e-3 (2.27e-3) -	6.1845e-3 (1.84e-3) -	4.2442e-3 (1.60e-4) -	4.2521e-3 (1.80e-4) -	5.0716e-3 (3.85e-4) -	4.9728e-3 (2.54e-4) -	**4.0410e-3 (4.69e-5)**
6	500	4.1248e-3 (6.97e-5) -	5.8329e-3 (4.00e-4) -	4.1359e-3 (7.14e-5) -	4.3638e-3 (7.50e-5) -	4.0679e-3 (5.92e-5) -	4.7118e-3 (2.80e-4) -	4.9819e-3 (2.92e-4) -	4.1522e-3 (6.92e-5) -	4.1588e-3 (6.96e-5) -	4.8541e-3 (2.10e-4) -	4.9509e-3 (2.07e-4) -	**4.0465e-3 (4.90e-5)**
	1000	4.1297e-3 (7.84e-5) -	5.8594e-3 (4.74e-4) -	4.1280e-3 (6.85e-5) -	4.3438e-3 (7.14e-5) -	4.0605e-3 (6.36e-5) ≈	4.8505e-3 (2.25e-4) -	4.8460e-3 (2.84e-4) -	4.1475e-3 (6.98e-5) -	4.1438e-3 (6.61e-5) -	4.8341e-3 (2.26e-4) -	4.8582e-3 (2.24e-4) -	**4.0476e-3 (4.29e-5)**
	2000	4.1479e-3 (6.90e-5) -	7.5784e-3 (2.85e-3) -	4.2035e-3 (1.21e-4) -	4.3222e-3 (8.05e-5) -	4.2166e-3 (2.47e-4) -	5.9224e-3 (1.49e-3) -	4.8330e-3 (2.80e-4) -	4.2708e-3 (2.06e-4) -	4.2634e-3 (2.12e-4) -	5.3719e-3 (8.57e-4) -	5.3996e-3 (8.84e-4) -	**4.0590e-3 (4.53e-5)**
7	500	5.4757e-3 (1.67e-3) -	4.1343e-2 (7.70e-2) -	4.2777e-3 (6.00e-4) -	4.2241e-3 (5.88e-5) -	4.7181e-3 (1.04e-3) -	1.4535e-2 (2.49e-3) -	1.3848e-2 (5.23e-3) -	4.3097e-3 (4.96e-4) -	4.2735e-3 (4.53e-4) -	5.7783e-3 (7.68e-4) -	5.0483e-3 (1.61e-4) -	**4.0250e-3 (2.23e-5)**
	1000	8.9000e-3 (2.39e-3) -	4.7033e-2 (7.96e-2) -	4.4549e-3 (6.26e-4) -	4.4794e-3 (6.37e-5) -	5.2200e-3 (1.14e-3) -	1.7095e-2 (2.00e-3) -	1.8380e-2 (3.55e-3) -	4.5840e-3 (6.35e-4) -	4.5119e-3 (5.03e-4) -	6.5899e-3 (9.84e-4) -	5.1228e-3 (1.65e-4) -	**4.0234e-3 (2.63e-5)**
	2000	2.5701e-2 (1.94e-2) -	8.1367e-2 (9.25e-2) -	2.2990e-2 (2.79e-2) -	4.3041e-3 (6.14e-5) -	2.8500e-2 (3.38e-2) -	6.5051e-2 (7.21e-2) -	4.3337e-2 (3.91e-2) -	2.1094e-2 (2.48e-2) -	1.9930e-2 (2.32e-2) -	3.5225e-2 (4.14e-2) -	5.8539e-3 (1.15e-3) -	**4.0702e-3 (1.24e-4)**
8	500	1.3979e-2 (5.28e-3) -	2.5013e-2 (8.70e-3) -	1.0091e-2 (3.26e-3) -	1.1518e-2 (3.57e-3) -	1.2314e-2 (3.96e-3) -	2.7900e-2 (3.57e-3) -	3.5543e-2 (1.07e-2) -	9.5291e-3 (2.75e-3) -	9.5784e-3 (2.90e-3) -	1.4900e-2 (3.62e-3) -	1.0958e-2 (2.97e-3) -	**8.0516e-3 (1.48e-3)**
	1000	1.9188e-2 (5.06e-3) -	3.0417e-2 (6.47e-3) -	1.1708e-2 (2.47e-3) -	1.3885e-2 (2.51e-3) -	1.4176e-2 (3.26e-3) -	3.3904e-2 (3.40e-3) -	3.9803e-2 (8.52e-3) -	**9.4009e-3 (2.27e-3)** ≈	9.9657e-3 (2.14e-3) ≈	1.6325e-2 (3.16e-3) -	1.3513e-2 (2.31e-3) -	9.6018e-3 (1.67e-3)
	2000	5.6037e-2 (4.34e-2) -	2.4750e-1 (1.87e-1) -	7.0489e-2 (8.69e-2) -	1.5565e-2 (1.93e-3) -	1.3010e-1 (1.02e-1) -	1.4341e-1 (1.63e-1) -	8.6459e-2 (6.33e-2) -	7.4884e-2 (5.84e-2) -	7.5327e-2 (5.83e-2) -	9.5100e-2 (1.15e-1) -	7.1771e-2 (8.71e-2) -	**1.0958e-2 (1.38e-3)**
+/− / ≈		0/24/0	0/24/0	0/24/0	0/24/0	0/24/0	0/22/2	1/22/1	0/24/0	0/23/1	0/23/1	0/24/0	0/24/0		

References

[1] Y. Tian, X. Zhang, C. Wang, Y. Jin, An evolutionary algorithm for large-scale sparse multiobjective optimization problems, IEEE Trans. Evol. Comput. 24 (2) (2019) 380–393.

[2] J. Xiao, T. Zhang, J. Du, X. Zhang, An evolutionary multiobjective route grouping-based heuristic algorithm for large-scale capacitated vehicle routing problems, IEEE Trans. Cybern. 51 (8) (2019) 4173–4186.

[3] M. Kaucic, F. Piccotto, G. Sbaiz, G. Valentinuz, A hybrid level-based learning swarm algorithm with mutation operator for solving large-scale cardinality-constrained portfolio optimization problems, Inf. Sci. 634 (2023) 321–339.

[4] Y. Xue, T. Tang, A.X. Liu, Large-scale feedforward neural network optimization by a self-adaptive strategy and parameter based particle swarm optimization, IEEE Access 7 (2019) 52473–52483.

[5] C. He, R. Cheng, C. Zhang, Y. Tian, Q. Chen, X. Yao, Evolutionary large-scale multi-objective optimization for ratio error estimation of voltage transformers, IEEE Trans. Evol. Comput. 24 (5) (2020) 868–881.

[6] H. Zille, S. Mostaghim, Comparison study of large-scale optimisation techniques on the LSMOP benchmark functions, in: 2017 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2017, pp. 1–8.

[7] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.

[8] Q. Zhang, H. Li, MOEA/D: a multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (6) (2007) 712–731.

[9] M.N. Omidvar, X. Li, X. Yao, A review of population-based metaheuristics for large-scale black-box global optimization–part i, IEEE Trans. Evol. Comput. 26 (5) (2021) 802–822.

[10] J. Liu, R. Sarker, S. Elsayed, D. Essam, N. Siswanto, Large-scale evolutionary optimization: a review and comparative study, Swarm Evol. Comput. 85 (2024) 101466.

[11] L.M. Antonio, C.A.C. Coello, Use of cooperative coevolution for solving large scale multiobjective optimization problems, in: 2013 IEEE Congress on Evolutionary Computation, IEEE, 2013, pp. 2758–2765.

[12] A. Song, Q. Yang, W.N. Chen, J. Zhang, A random-based dynamic grouping strategy for large scale multi-objective optimization, in: 2016 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2016, pp. 468–475.

[13] X. Ma, F. Liu, Y. Qi, X. Wang, L. Li, L. Jiao, M. Yin, M. Gong, A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables, IEEE Trans. Evol. Comput. 20 (2) (2015) 275–298.

[14] X. Zhang, Y. Tian, R. Cheng, Y. Jin, A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization, IEEE Trans. Evol. Comput. 22 (1) (2016) 97–112.

[15] H. Zille, H. Ishibuchi, S. Mostaghim, Y. Nojima, A framework for large-scale multiobjective optimization based on problem transformation, IEEE Trans. Evol. Comput. 22 (2) (2017) 260–275.

[16] C. He, L. Li, Y. Tian, X. Zhang, R. Cheng, Y. Jin, X. Yao, Accelerating large-scale multiobjective optimization via problem reformulation, IEEE Trans. Evol. Comput. 23 (6) (2019) 949–961.

[17] C. He, R. Cheng, D. Yazdani, Adaptive offspring generation for evolutionary large-scale multiobjective optimization, IEEE Trans. Syst. Man Cybern. 52 (2) (2020) 786–798.

[18] Y. Tian, X. Zheng, X. Zhang, Y. Jin, Efficient large-scale multiobjective optimization based on a competitive swarm optimizer, IEEE Trans. Cybern. 50 (8) (2019) 3696–3708.

[19] Y. Xue, H. Zhu, J. Liang, A. Słowik, Adaptive crossover operator based multi-objective binary genetic algorithm for feature selection in classification, Knowl. Based Syst. 227 (2021) 107218. https://www.sciencedirect.com/science/article/pii/S0950705121004809. https://doi.org/10.1016/j.knosys.2021.107218

[20] B. Xue, M. Zhang, W.N. Browne, Particle swarm optimization for feature selection in classification: a multi-objective approach, IEEE Trans. Cybern. 43 (6) (2012) 1656–1671.

[21] L. Tang, L. Zhang, P. Luo, M. Wang, Incorporating occupancy into frequent pattern mining for high quality pattern recommendation, in: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, 2012, pp. 75–84.

[22] J.E. Fieldsend, S. Singh, Pareto evolutionary neural networks, IEEE Trans. Neural Netw. 16 (2) (2005) 338–354.

[23] A. Ponsich, A.L. Jaimes, C.A.C. Coello, A survey on multiobjective evolutionary algorithms for the solution of the portfolio optimization problem and other finance and economics applications, IEEE Trans. Evol. Comput. 17 (3) (2012) 321–344.

[24] Y. Zou, Y. Liu, J. Zou, S. Yang, J. Zheng, An evolutionary algorithm based on dynamic sparse grouping for sparse large scale multiobjective optimization, Inf. Sci. 631 (2023) 449–467. https://doi.org/10.1016/j.ins.2023.02.062

[25] Z. Ding, L. Chen, D. Sun, X. Zhang, A multi-stage knowledge-guided evolutionary algorithm for large-scale sparse multi-objective optimization problems, Swarm Evol. Comput. 73 (2022) 101119. https://doi.org/10.1016/j.swevo.2022.101119

[26] P. Wah Tang, P. San Chua, S. Kee Chong, M. Saberi Mohamad, Y. Wen Choon, S. Deris, S. Omatu, J. Manuel Corchado, W. Howe Chan, R. Abdul Rahim, A review of gene knockout strategies for microbial cells, Recent Pat. Biotechnol. 9 (3) (2015) 176–197.

[27] Y. Tian, S. Shao, G. Xie, X. Zhang, A multi-granularity clustering based evolutionary algorithm for large-scale sparse multi-objective optimization, Swarm Evol. Comput. 84 (2024) 101453. https://doi.org/10.1016/j.swevo.2023.101453

[28] Y. Tian, C. Lu, X. Zhang, F. Cheng, Y. Jin, A pattern mining-based evolutionary algorithm for large-scale sparse multiobjective optimization problems, IEEE Trans. Cybern. 52 (7) (2022) 6784–6797. https://doi.org/10.1109/TCYB.2020.3041325

[29] C. Wu, Y. Tian, Y. Zhang, H. Jiang, X. Zhang, A sparsity-guided elitism co-evolutionary framework for sparse large-scale multi-objective optimization, in: 2023 IEEE Congress on Evolutionary Computation (CEC), 2023, pp. 1–8. https://doi.org/10.1109/CEC53210.2023.10253988

[30] Y. Zhang, C. Wu, Y. Tian, X. Zhang, A co-evolutionary algorithm based on sparsity clustering for sparse large-scale multi-objective optimization, Eng. Appl. Artif. Intell. 133 (2024) 108194. https://doi.org/10.1016/j.engappai.2024.108194

[31] J. Jiang, F. Han, J. Wang, Q. Ling, H. Han, Y. Wang, A two-stage evolutionary algorithm for large-scale sparse multiobjective optimization problems, Swarm Evol. Comput. 72 (2022) 101093. https://doi.org/10.1016/j.swevo.2022.101093

[32] J. Jiang, H. Wang, J. Hong, Z. Liu, F. Han, Improving two-layer encoding of evolutionary algorithms for sparse large-scale multiobjective optimization problems, Complex Intell. Syst. (2024). https://doi.org/10.1007/s40747-024-01489-x

[33] Y. Zhang, Y. Tian, X. Zhang, Improved SparseEA for sparse large-scale multi-objective optimization problems, Complex Intell. Syst. 9 (2) (2023) 1127–1142. https://doi.org/10.1007/s40747-021-00553-0

[34] S. Shao, Y. Tian, X. Zhang, A non-uniform clustering based evolutionary algorithm for solving large-scale sparse multi-objective optimization problems, in: International Conference on Bio-Inspired Computing: Theories and Applications, Springer, 2023, pp. 103–116.

[35] Y. Tian, C. Lu, X. Zhang, K.C. Tan, Y. Jin, Solving large-scale multiobjective optimization problems with sparse optimal solutions via unsupervised neural networks, IEEE Trans. Cybern. 51 (6) (2021) 3115–3128. https://doi.org/10.1109/TCYB.2020.2979930

[36] H. Geng, J. Shen, Z. Zhou, K. Xu, An improved large-scale sparse multi-objective evolutionary algorithm using unsupervised neural network, Appl. Intell. 53 (9) (2023) 10290–10309. https://doi.org/10.1007/s10489-022-04037-7

[37] M. Gao, X. Feng, H. Yu, X. Li, An efficient evolutionary algorithm based on deep reinforcement learning for large-scale sparse multiobjective optimization, Appl. Intell. 53 (18) (2023) 21116–21139. https://doi.org/10.1007/s10489-023-04574-9

[38] I. Kropp, A.P. Nejadhashemi, K. Deb, Benefits of sparse population sampling in multi-objective evolutionary computing for large-scale sparse optimization problems, Swarm Evol. Comput. 69 (2022) 101025. https://doi.org/10.1016/j.swevo.2021.101025

[39] X. Wang, K. Zhang, J. Wang, Y. Jin, An enhanced competitive swarm optimizer with strongly convex sparse operator for large-Scale multiobjective optimization, IEEE Trans. Evol. Comput. 26 (5) (2022) 859–871. https://doi.org/10.1109/TEVC.2021.3111209

[40] X. Wang, B. Zhang, J. Wang, K. Zhang, Y. Jin, A cluster-based competitive particle swarm optimizer with a sparse truncation operator for multi-objective optimization, Swarm Evol. Comput. 71 (2022) 101083. https://doi.org/10.1016/j.swevo.2022.101083

[41] Y. Tian, R. Cheng, X. Zhang, Y. Jin, PlatEMO: a MATLAB platform for evolutionary multi-objective optimization [educational forum], IEEE Comput. Intell. Mag. 12 (4) (2017) 73–87.

[42] I. Kropp, A.P. Nejadhashemi, K. Deb, Improved evolutionary operators for sparse large-scale multiobjective optimization problems, IEEE Trans. Evol. Comput. 28 (2) (2023) 460–473.

[43] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, V.G. Da Fonseca, Performance assessment of multiobjective optimizers: an analysis and review, IEEE Trans. Evol. Comput. 7 (2) (2003) 117–132.