Dynamic Clustering-Based Dynamic Multiobjective Optimization for Disassembly Line Balancing

Yilin Fang
School of Information Engineering
Wuhan University of Technology
Wuhan, China
fangspirit@whut.edu.cn

Yizhe Zhang
School of Information Engineering
Wuhan University of Technology
Wuhan, China
301999@whut.edu.cn

Xin Gu
Hubei Standardization and Quality Institution
Wuhan, China
guxinlucifer@126.com

***Abstract*—Disassembly line balancing plays a key role in improving the efficiency and sustainability of remanufacturing systems. In practice, however, the conditions of end-of-life (EOL) products, task processing times and resource availability often change over time, which naturally leads to a dynamic disassembly line balancing problem (D-DLBP). This problem is inherently a dynamic multiobjective optimization problem (DMOP), where the Pareto optimal set (POS) and the Pareto front (PF) vary with the environment. Existing dynamic multiobjective evolutionary algorithms (DMOEAs) can react to environmental changes by preserving diversity, using multiple populations, memorizing historical solutions, or predicting future environments. Nevertheless, most of them focus primarily on the quality of the historical solutions and ignore the structural distribution and density characteristics of the historical POS samples. As a result, they may suffer from redundancy, negative transfer and insufficient adaptability when the environment changes significantly.**

To address this issue, this paper proposes a *Dynamic Clustering-Based Dynamic Multiobjective Evolutionary Algorithm* (DC-DMOEA) for dynamic disassembly line balancing. The algorithm collects POS samples from all historical environments and applies HDBSCAN, a hierarchical density-based clustering method, to extract representative cluster exemplars in the objective space. For each cluster, we define a shape descriptor that captures its mean, spread and density; based on these descriptors, we construct an environment-shape similarity measure between the new environment and existing clusters. The initial population in the new environment is then generated via weighted sampling of cluster exemplars according to their similarity. In this way, DC-DMOEA can produce environment-adaptive initial populations that simultaneously exploit stable historical knowledge and avoid overfitting to environment-specific outliers. The proposed framework is particularly suitable for discrete, constrained and combinatorial DMOPs such as D-DLBP, where the number of historical POS samples is large and their distributions are highly irregular.

***Index Terms*—Disassembly line balancing, dynamic multiobjective optimization, HDBSCAN, clustering, adaptive initialization, domain generalization.**

I. Introduction

With increasing awareness of environmental protection and resource scarcity, the recycling and remanufacturing of end-of-life (EOL) products have become an indispensable component of modern manufacturing. Many EOL products contain valuable components or materials that can be reused or recycled. Disassembly lines, composed of sequential workstations connected by conveyor systems, are widely adopted in industry to perform systematic and efficient disassembly operations. Batches of EOL products are transported along the line; at each workstation, one or several operators (or robots) execute a subset of disassembly tasks until all products in the batch are completely disassembled.

The *Disassembly Line Balancing Problem* (DLBP) concerns how to assign disassembly operations to workstations and operators without violating precedence constraints, while optimizing performance measures such as cycle time, number of workstations, workload balance, and cost [1]. Compared with classical assembly line balancing, DLBP is more challenging due to several distinctive characteristics: (1) product structures are often highly variable; (2) some operations may be selectively performed depending on product condition; (3) AND/OR precedence relationships arise naturally because there may exist multiple feasible disassembly sequences for the same product; and (4) uncertainties appear in task processing times, product quality, and demand for recovered parts.

In many real-world scenarios, the disassembly environment is not static. The condition of incoming products in successive batches may change due to different usage histories, leading to changes in task processing times. The availability of robots, tools or workstations may also fluctuate. These factors together result in a *dynamic* disassembly line balancing problem (D-DLBP), which is inherently a *dynamic multiobjective optimization problem* (DMOP). In D-DLBP, the objective functions and constraints depend on an environment index ℓ, and the corresponding Pareto optimal set (POS) and Pareto front (PF) are time-varying.

A variety of dynamic multiobjective evolutionary algorithms (DMOEAs) have been proposed in the literature to cope with DMOPs [2], [3]. Existing methods can be roughly classified into four main categories: diversity-based approaches, multipopulation-based approaches, memory-based approaches and prediction-based approaches. Diversity-based approaches introduce new individuals or adjust mutation operators to restore diversity after an environmental change. Multipopulation-based approaches maintain several subpopulations that explore different regions of the search space. Memory-based approaches store non-dominated solutions from previous environments and reuse them when a change occurs. Prediction-based approaches attempt to forecast the

movement of the PF or POS using regression models, reference points or machine learning techniques.

In recent years, research on dynamic multiobjective evolutionary algorithms (DMOEAs) has gradually shifted from purely reactive strategies toward more adaptive and predictive mechanisms. Representative studies have explored prediction-based models, memory-enhanced frameworks, and diversity-adaptive strategies to improve tracking performance under changing environments [4].

However, most existing DMOEAs are primarily designed for continuous benchmark problems and do not explicitly consider the structural characteristics of complex combinatorial optimization tasks. In particular, for dynamic disassembly line balancing problems (D-DLBP), the decision space is discrete, highly constrained, and tightly coupled with product structures, which limits the effectiveness of generic prediction or operator-adaptation strategies.

To address this issue, domain generalization has recently been introduced into dynamic multiobjective optimization for D-DLBP. Fang *et al.* [5] proposed a domain generalization-based dynamic multiobjective evolutionary algorithm (DG-DMOEA), in which a solution-generative model is trained on Pareto-optimal sets from multiple historical environments to capture transferable structural knowledge. When a new environment arrives, DG-DMOEA can directly generate a high-quality initial population without requiring evaluations in the new environment, significantly reducing response time to environmental changes.

Although DG-DMOEA effectively exploits historical POS samples, it mainly focuses on how to generate high-quality solutions from a parametric model. It does not explicitly address the following issues:

- **Redundancy and density imbalance of historical POS:** as the number of historical environments grows, the POS repository may contain a huge number of solutions, whose density and distribution can vary significantly across environments. Treating all historical POS samples equally can lead to redundancy and may obscure important structural information.
- **Lack of environment-adaptive selection of historical solutions:** many methods reuse historical POS samples without explicitly considering how similar the new environment is to previous ones. This may cause negative transfer when the new environment differs substantially from those in the past.
- **Absence of explicit modeling of POS distributional patterns:** most existing DMOEAs view historical solutions as individual points, rather than modeling their cluster structure, density, and shape in the objective space.

Motivated by these observations, this paper proposes a new dynamic multiobjective optimization framework that explicitly incorporates density-based clustering and environment-shape similarity into the reuse of historical POS.

A. Contributions

To address the above limitations, we propose a *Dynamic Clustering-Based Dynamic Multiobjective Evolutionary Algorithm* (DC-DMOEA) tailored for dynamic disassembly line balancing. The main contributions of this work are summarized as follows:

1) **HDBSCAN-based clustering of historical POS:** We construct a unified historical POS repository that aggregates non-dominated solutions from all past environments. We then apply HDBSCAN, a hierarchical density-based clustering algorithm, to partition the POS into clusters of varying densities and to discard noise. Each cluster corresponds to a region of the PF that is consistently visited across multiple environments.
2) **Environment-shape descriptors and similarity measure:** For each cluster, we compute a descriptor that captures the statistical *shape* (mean, spread and density) of the POS samples within the cluster. Similarly, we define a descriptor for the new environment based on POS samples or environment-related features. We then construct an environment-shape similarity measure, which quantifies how relevant each cluster is to the new environment.
3) **Environment-adaptive initial population generation:** Based on the similarity measure, we design a weighted sampling mechanism that selects cluster exemplars to form an environment-adaptive initial population. Clusters more similar to the current environment contribute more individuals, whereas dissimilar or low-density clusters contribute less or none. This leads to initial populations that better balance exploration and exploitation in dynamic environments.
4) **Integration with D-DLBP:** We integrate the proposed dynamic clustering and sampling mechanism into a multiobjective evolutionary algorithm for D-DLBP, resulting in a framework that can effectively track the time-varying PF under dynamic conditions. The proposed method is particularly suitable for complex discrete and constrained problems where the number of historical POS samples is large and their distribution is highly irregular.

The remainder of this paper is organized as follows. Section II reviews related work on disassembly line balancing, dynamic multiobjective optimization, and clustering techniques. Section III formulates the dynamic disassembly line balancing problem, introduces the historical POS repository and environment-shape descriptors, and details the proposed DC-DMOEA algorithm. Subsequent sections will present experimental studies and conclusions.

II. Related Work

A. Disassembly Line Balancing

The disassembly line balancing problem (DLBP) was first formally introduced by Gungor and Gupta [1]. They pointed out that although disassembly can be considered the reverse

of assembly, the problem is more complex due to uncertain product conditions, selective disassembly, and AND/OR precedence structures. In DLBP, each disassembly operation corresponds to the removal of a part or a group of parts from the product, and the operations must satisfy precedence relationships derived from the product structure and feasible disassembly sequences.

A wide variety of disassembly line configurations have been studied in the literature. Straight-line disassembly lines are conceptually simple and are often used as a starting point in modeling. More advanced layouts include U-shaped lines, where the input and output are located at the same end of the line, allowing workers or robots to process tasks from both the early and late stages of the disassembly process. Two-sided lines are suitable for large and symmetric products such as vehicles, where tasks can be executed on both sides simultaneously. Mixed-model disassembly lines handle multiple product types on the same line, improving equipment utilization and flexibility.

In addition to manual disassembly, recent studies have increasingly focused on robotic and human–robot collaborative disassembly line balancing to improve automation, efficiency, and sustainability in remanufacturing systems. Fang *et al.* [6] studied mixed-model disassembly lines with multi-robotic workstations under multiobjective settings. Guo *et al.* [7] formulated a stochastic human–robot collaborative disassembly line balancing problem and solved it using a multi-objective shuffled frog leaping algorithm, demonstrating effective trade-offs between cycle time, energy consumption, and workload balance under uncertainty. Other recent works have addressed multi-objective optimization of disassembly task sequencing and operator assignment [8], and have incorporated green and destructive disassembly considerations into human–robot cooperative frameworks [9], highlighting the growing attention to energy efficiency and environmental objectives in disassembly line optimization.

B. Dynamic Multiobjective Optimization

Dynamic multiobjective optimization problems (DMOPs) involve multiple objectives and/or constraints that change over time. A DMOP can be defined as:

$$\min\ \mathbf{F}(\mathbf{x}, t) = \big(F_1(\mathbf{x}, t), \ldots, F_M(\mathbf{x}, t)\big),$$

subject to constraints that may also depend on t. The goal is to track the time-varying Pareto-optimal set and Pareto front as t evolves.

Many dynamic multiobjective evolutionary algorithms (DMOEAs) have been proposed to solve DMOPs. The main categories include:

Diversity-based approaches. These methods maintain or restore diversity after an environmental change to avoid premature convergence. Typical strategies include re-initialization of a subset of the population, random immigrants, or adaptive mutation rates. By injecting new individuals, the algorithm gains the ability to explore new promising regions in the updated search space.

Multipopulation-based approaches. Instead of a single population, multiple subpopulations are maintained. Each subpopulation may focus on a different region of the search space or use different parameter settings. When an environmental change is detected, some subpopulations may be reinitialized or repurposed to explore newly promising regions. This strategy improves robustness and reduces the risk that the entire search process is misled by outdated information.

Memory-based approaches. Memory-based methods explicitly store elite solutions from previous environments in an archive. When the environment changes, the archive is consulted to provide candidate solutions for the new environment. This is especially effective when the environments are periodic or when consecutive environments are similar. However, naive memory usage may lead to redundancy and negative transfer when the new environment is very different from past ones.

Prediction-based approaches. These methods aim to predict the movement of the Pareto front or the POS based on historical data. Techniques such as regression models, transfer learning, and recurrent neural networks have been employed. By predicting approximate positions of future non-dominated solutions, the algorithm can initialize the population closer to the new Pareto front and thus accelerate convergence. However, accurate prediction can be difficult when the environment changes irregularly or when the distribution of solutions is highly nonlinear.

While these methods have achieved notable success, most of them treat historical solutions as individual points. Few methods explicitly model the distributional structure (e.g., clustering and density) of historical POS samples across environments. This limits their ability to distinguish between widely shared knowledge (stable patterns) and environment-specific patterns (outliers).

C. Clustering in Evolutionary Computation

Clustering methods have been used in evolutionary computation for various purposes, including niching, diversity preservation, reference vector adaptation, and decomposition of the search space [10]. For multiobjective problems, clustering can help identify regions of the Pareto front that require more attention or can be used to generate reference points to guide the search. Traditional clustering methods such as k-means and DBSCAN have been widely adopted. However, k-means assumes spherical clusters with similar densities and requires the number of clusters to be specified in advance. DBSCAN can detect arbitrarily shaped clusters and identify noise points, but it is sensitive to the choice of density parameters and may struggle with datasets where cluster densities vary significantly.

HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) extends DBSCAN by constructing a hierarchy of clusters over varying density thresholds and then extracting a flat clustering based on stability analysis [11]. HDBSCAN is particularly effective when clusters have different densities or when there is substantial noise. These

properties make HDBSCAN a promising tool for analyzing historical POS data in DMOPs.

III. Problem Definition and Proposed Method

In this section, we first recall the formulation of the dynamic disassembly line balancing problem (D-DLBP). We then introduce the historical POS repository and define environment-shape descriptors based on mean, spread and density. Finally, we present the proposed dynamic clustering-based dynamic multiobjective evolutionary algorithm (DC-DMOEA).

A. Dynamic Disassembly Line Balancing Problem

We consider a disassembly line with a set of workstations $J = \{1, \ldots, |J|\}$ and a set of robots $R = \{1, \ldots, |R|\}$. A batch of EOL products is processed in each environment ℓ. The structure of each product type is modeled by a transformed AND/OR graph, from which all feasible disassembly operations and precedence constraints can be derived [5]. Let B denote the set of all possible disassembly operations (normal nodes in the transformed AND/OR graph), and let $D^\ell \subseteq B$ denote the subset of operations that are actually selected for execution in environment ℓ.

The dynamic disassembly line balancing problem (D-DLBP) consists of three intertwined subproblems:

1) selecting a feasible disassembly sequence (or sequences) for each product in the batch;
2) assigning each selected operation to a robot;
3) allocating robots to workstations.

These decisions must satisfy precedence constraints, station capacity constraints, and possibly additional restrictions related to collision avoidance and synchronization.

Following [5], a solution for environment ℓ can be encoded by three vectors: the robot-workstation assignment vector RW^ℓ, the operation-robot assignment vector TR^ℓ, and the operation sequence vector TS^ℓ. The details of this encoding are omitted here for brevity, but we emphasize that it can represent complete solutions that satisfy the disassembly precedence and station constraints.

The objective of D-DLBP is typically to minimize the cycle time and the number of active robots. Let $CT(\ell)$ denote the cycle time of the disassembly line in environment ℓ, i.e., the longest workstation processing time. Let $NR(\ell)$ denote the number of robots used. Formally, we consider the bi-objective optimization problem:

$$\min \; F(\ell) = \langle CT(\ell), NR(\ell) \rangle. \tag{1}$$

The cycle time can be written as

$$CT(\ell) = \max_{j \in J} \left(S_j^\ell + \sum_{r \in R} \sum_{b \in B} t_{br}^\ell X_{brj}^\ell \right), \tag{2}$$

where S_j^ℓ is the starting time of operations at workstation j in environment ℓ, t_{br}^ℓ is the processing time of operation b by robot r in environment ℓ, and X_{brj}^ℓ is a binary decision variable indicating whether operation b is assigned to robot r at workstation j. The number of robots used is given by

$$NR(\ell) = \sum_{r \in R} W_r^\ell, \tag{3}$$

where W_r^ℓ is a binary variable indicating whether robot r is active in environment ℓ. The full set of constraints includes precedence relations, station capacity limits, and integrality constraints.

Due to the variability of product conditions and potential changes in system configuration, the processing times t_{br}^ℓ and the feasible set D^ℓ may change from one environment to another. Consequently, the POS and PF of D-DLBP are time-varying and must be tracked by a dynamic optimization algorithm.

B. Historical POS Repository and Environment Shapes

Assume that environments arrive sequentially and are indexed by $\ell = 1, 2, \ldots$. For each environment $\tau \leq \ell$, a dynamic optimization algorithm outputs an approximate Pareto optimal set $POS^\tau = \{\mathbf{x}_1^\tau, \ldots, \mathbf{x}_{n_\tau}^\tau\}$, where each $\mathbf{x}_i^\tau$ is a feasible solution encoding (e.g., $(\mathrm{RW}^\tau, \mathrm{TR}^\tau, \mathrm{TS}^\tau)$) and has an associated objective vector $\mathbf{F}(\mathbf{x}_i^\tau, \tau) = (F_1(\mathbf{x}_i^\tau, \tau), F_2(\mathbf{x}_i^\tau, \tau))$. Over time, the algorithm accumulates a large number of POS samples.

We define the *historical POS repository* as

$$\mathcal{H} = \bigcup_{\tau=1}^{\ell} POS^\tau. \tag{4}$$

This repository is used as the data source for extracting long-term structural knowledge about the PF across environments. However, directly using all samples in $\mathcal{H}$ as the initial population in a new environment would be inefficient and potentially harmful, due to redundancy, uneven density, and environment-specific patterns.

To address this, we propose to summarize the distributional structure of $\mathcal{H}$ in the objective space via clustering, and then derive *environment-shape descriptors* that capture how the POS is shaped in each environment and in each cluster. The schematic illustration of this Dynamic Clustering Strategy (DCS) is presented in Fig.1, Step 1 shows the distribution of the old population with disconnected regions; Step 2 illustrates the identification of clusters and their geometric centers (marked by red 'X') using the clustering algorithm; Step 3 demonstrates the re-initialization of the new population $P(t)$ generated through Gaussian sampling around the identified cluster centers.

1) Clustering of Historical POS with HDBSCAN: We map each solution $\mathbf{x} \in \mathcal{H}$ to its objective vector $\mathbf{f} = \mathbf{F}(\mathbf{x}) \in \mathbb{R}^2$. Let $\mathcal{Y}$ denote the set of all such objective vectors. We then apply HDBSCAN to $\mathcal{Y}$ to obtain a set of clusters

$$\mathcal{C} = \{C_1, C_2, \ldots, C_K\}, \tag{5}$$

and a set of noise points that are not assigned to any cluster. Each cluster C_k represents a region of the PF that

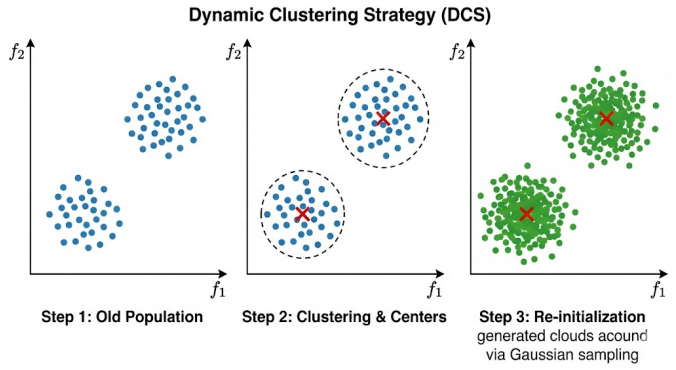

Fig. 1. Schematic diagram of the Dynamic Clustering Strategy mechanism.

is consistently occupied by historical solutions. The density-based nature of HDBSCAN allows it to find clusters of varying densities and shapes, and to discard outliers.

For each cluster C_k, we define its *representative exemplar* (or cluster center) c_k as

$$c_k = \arg\min_{\mathbf{f}\in C_k} \sum_{\mathbf{g}\in C_k} \|\mathbf{f}-\mathbf{g}\|_2. \tag{6}$$

In addition, HDBSCAN assigns a *stability* score d_k to each cluster, which reflects the persistence of the cluster over the hierarchical clustering process. Higher stability indicates that the cluster is more robust to changes in density thresholds and less likely to be an artifact of noise.

2) POS Shape Statistics: To characterize the *shape* of cluster C_k in the objective space, we compute three statistics:

- Mean vector:

$$\boldsymbol{\mu}_k = \frac{1}{|C_k|}\sum_{\mathbf{f}\in C_k} \mathbf{f}. \tag{7}$$

- Spread (as a scalar summary of the covariance):

$$s_k = \left\| \frac{1}{|C_k|}\sum_{\mathbf{f}\in C_k} (\mathbf{f}-\boldsymbol{\mu}_k)(\mathbf{f}-\boldsymbol{\mu}_k)^\top \right\|_F, \tag{8}$$

  where $\|\cdot\|_F$ denotes the Frobenius norm.
- Density (stability score from HDBSCAN):

$$d_k = \text{stability}(C_k). \tag{9}$$

We then define a cluster-shape descriptor as

$$f(C_k) = \left(\boldsymbol{\mu}_k, s_k, d_k\right). \tag{10}$$

Similarly, we can define an *environment-shape descriptor* for environment τ based on its POS POS^τ. For example, we compute the mean and spread of $\{\mathbf{F}(\mathbf{x}_i^\tau,\tau)\}_{i=1}^{n_\tau}$, and a density indicator derived from how many of these points belong to high-stability clusters. This yields

$$f(E^\tau) = \left(\boldsymbol{\mu}^\tau, s^\tau, d^\tau\right). \tag{11}$$

When a new environment $\ell{+}1$ arrives, we obtain an approximate estimate of $f(E^{\ell+1})$ from early optimization or from environment-related features (e.g., distribution of processing times). Then, we measure the similarity between the new environment and each cluster.

3) Environment-Shape Similarity: We define the similarity between cluster C_k and environment $E^{\ell+1}$ as

$$\text{sim}(C_k, E^{\ell+1}) = \exp\left(-\alpha \left\| f(C_k) - f(E^{\ell+1}) \right\|_2\right), \tag{12}$$

where $\alpha > 0$ is a user-defined parameter that controls the sharpness of the similarity function. A smaller distance between the descriptors implies a higher similarity. The similarity values will be used as weights in the initial population generation process.

C. Environment-Adaptive Initial Population Generation

Let N denote the desired population size in the new environment $E^{\ell+1}$. We first normalize the similarity values across all clusters:

$$p_k = \frac{\text{sim}(C_k, E^{\ell+1})}{\sum_{j=1}^{K} \text{sim}(C_j, E^{\ell+1})}, \quad k = 1,\ldots,K. \tag{13}$$

We then determine the number of individuals to sample from each cluster as

$$N_k = \lfloor p_k \cdot N \rfloor, \tag{14}$$

with the remaining individuals assigned to the clusters with the highest residual fractional parts until the total population size reaches N.

From each cluster C_k, we sample N_k solutions. The simplest option is to sample uniformly at random from C_k. Alternatively, we can bias the sampling toward the cluster exemplar c_k or toward more central points within the cluster. Each sampled objective vector $\mathbf{f}$ is associated with one or more candidate decision vectors $\mathbf{x}$ in the historical repository; we can retrieve these $\mathbf{x}$ as initial individuals. If multiple decision vectors map to similar objective vectors, we may select one at random or apply small perturbations (e.g., local neighborhood moves on the disassembly sequence or robot assignments) to increase diversity.

In this way, the initial population for environment $E^{\ell+1}$ is composed of individuals drawn from clusters that are most similar to the new environment, according to the environment-shape descriptors. Clusters that are dissimilar or low-density contribute few or no individuals, thereby reducing negative transfer from environment-specific outliers.

This section presents the experimental configuration adopted to evaluate the proposed DC-DMOEA on dynamic disassembly line balancing problems. The experimental protocol is designed based on the benchmark framework used in DG-DMOEA, while incorporating a simplified yet representative set of dynamic test instances.

D. Overall DC-DMOEA Framework

To clearly illustrate the overall workflow of the proposed DC-DMOEA, Fig. 2 presents its algorithmic framework. DC-DMOEA operates in a dynamic environment stream and consists of three main components: historical knowledge construction, clustering and environment-shape modeling, and environment-adaptive population initialization, which are seamlessly integrated with a base MOEA optimizer.

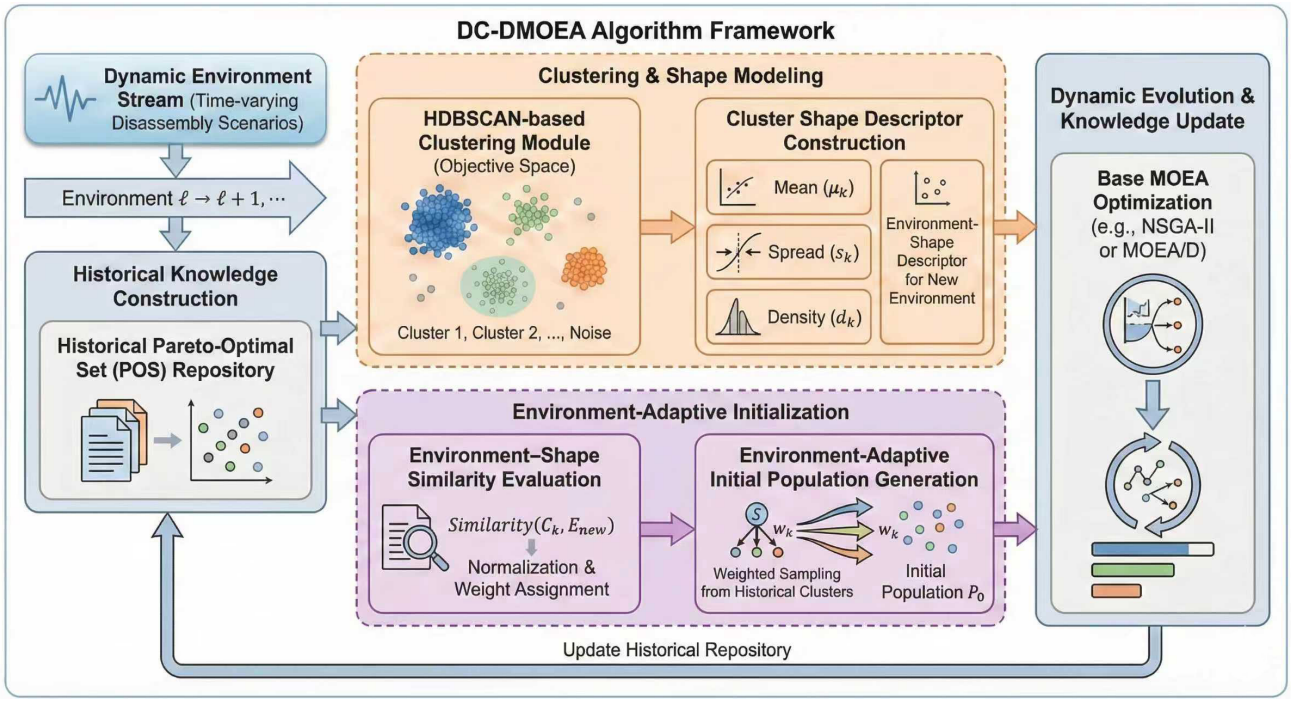

Fig. 2. Overall framework of the proposed DC-DMOEA.

Algorithm 1 DC-DMOEA for Dynamic Disassembly Line Balancing

Input: A DMOP $F(\ell)$, $\ell = 0, 1, \ldots$; population size N; historical POS sets $\{POS(0), \ldots, POS(\ell)\}$.
Output: POS for $E^{\ell+1}$.

1: Construct historical POS repository $\mathcal{H} = \bigcup_{\tau=1}^{\ell} POS^{\tau}$.
2: Map $\mathcal{H}$ to objective space and apply HDBSCAN to obtain clusters $\mathcal{C} = \{C_1, \ldots, C_K\}$.
3: **for** each cluster C_k **do**
4: Compute cluster descriptor $f(C_k) = (\mu_k, s_k, d_k)$
5: **end for**
6: Estimate environment-shape descriptor $f(E^{\ell+1})$ for the new environment.
7: **for** each cluster C_k **do**
8: $sim_k \leftarrow \exp\big(-\alpha \| f(C_k) - f(E(\ell)) \|\big)$
9: **end for**
10: $p_k \leftarrow \dfrac{sim_k}{\sum_{j=1}^{K} sim_j}, \quad N_k \leftarrow \lfloor p_k N \rfloor$
11: $P^{\ell+1} \leftarrow \text{Sample}\big(\{C_k, N_k\}_{k=1}^{K}\big)$
12: $POS(\ell+1) \leftarrow \text{MOEA}\big(P(\ell+1), F(\ell+1)\big)$
13: $\mathcal{H} \leftarrow \mathcal{H} \cup POS(\ell+1)$
14: Detect environment change; if changed, set $\ell \leftarrow \ell + 1$
15: Optionally update clusters $\mathcal{C}$ and descriptors $f(C_k)$

We now summarize the overall DC-DMOEA algorithm in Algorithm 1. The framework assumes that environmental changes are detected by an external mechanism (e.g., monitoring of processing times or product quality) and that a base multiobjective evolutionary algorithm (MOEA) is available for solving static instances of DLBP.

The proposed DC-DMOEA can be combined with different MOEAs (e.g., NSGA-II [12], MOEA/D [13]) as the base optimizer. The key novelty lies in Steps 2–12, where historical POS samples are clustered via HDBSCAN, environment-shape descriptors are computed, and an environment-adaptive initial population is generated by weighted sampling according to the similarity between clusters and the new environment. This design allows the algorithm to exploit long-term knowledge stored in the historical POS repository, while maintaining robustness against environment-specific outliers and redundancy.

IV. Experimental Setup and Performance Metrics

A. Dynamic Test Instances

The experiments are conducted on a set of dynamic disassembly line balancing (D-DLB) instances constructed from dynamic task-and-operation graphs (D-TAOGs). Eight benchmark TAOGs are first transformed into their dynamic counterparts by introducing environment-dependent variations in task processing times.

Based on these D-TAOGs, a total of eight mixed D-DLB instances are generated, including four small-scale and four large-scale instances. Small-scale instances are constructed by combining two or three D-TAOGs (M1–M4), whereas large-scale instances are formed by integrating four or five D-TAOGs (M5–M8).

To simulate dynamic environments with varying degrees of similarity, each mixed instance is further associated with two environment similarity levels, namely High and Low. The High similarity environment represents mild dynamic changes

TABLE I
MIGD RESULTS ON SMALL-SCALE AND LARGE-SCALE INSTANCES

Test Instance	SGEA	D-SVR	DVC	**DC-DMOEA**
M1H	**1.5819E-1**†	2.4895E-1‡	1.7187E-1†	1.6914E-1
M2H	1.4715E-1‡	2.5164E-1‡	1.5393E-1‡	**1.4127E-1**
M3H	1.0905E-1‡	1.5483E-1‡	1.1793E-1‡	**9.7846E-2**
M4H	1.1536E-1‡	1.9701E-1‡	1.6218E-1‡	**9.1510E-2**
M5H	9.8416E-2‡	9.6894E-2‡	9.9544E-2‡	**8.4123E-2**
M6H	8.9338E-2†	1.0989E-1‡	9.4246E-2‡	**8.8955E-2**
M7H	8.4323E-2‡	1.1735E-1‡	8.7175E-2‡	**7.4611E-2**
M8H	7.5932E-2‡	8.4691E-2‡	7.0118E-2‡	**6.9522E-2**
M1L	1.8151E-1‡	2.3342E-1‡	1.8527E-1‡	**1.7390E-1**
M2L	1.8432E-1‡	3.2626E-1‡	**1.7911E-1**†	2.2112E-1
M3L	1.2677E-1‡	2.0160E-1‡	1.2253E-1‡	**1.1613E-1**
M4L	1.5619E-1‡	2.2199E-1‡	1.5428E-1‡	**1.3573E-1**
M5L	9.6366E-2‡	1.3464E-1‡	9.7383E-2‡	**8.9772E-2**
M6L	9.9519E-2‡	2.0581E-1‡	**8.1308E-2**†	9.4663E-2
M7L	9.3134E-2‡	1.1976E-1‡	8.5585E-2‡	**8.3519E-2**
M8L	8.9776E-2‡	1.0771E-1‡	8.7628E-2‡	**8.6366E-2**

TABLE II
MHV RESULTS ON SMALL-SCALE AND LARGE-SCALE INSTANCES

Test Instance	SGEA	D-SVR	DVC	**DC-DMOEA**
M1H	**6.8943E-1**†	4.5848E-1‡	6.6134E-1‡	6.2579E-1
M2H	7.2547E-1‡	5.5198E-1‡	6.9721E-1‡	**7.4483E-1**
M3H	7.1648E-1‡	6.0467E-1‡	6.8742E-1‡	**7.4049E-1**
M4H	7.2642E-1‡	6.8474E-1‡	7.1671E-1‡	**7.6819E-1**
M5H	8.1762E-1‡	7.6482E-1‡	8.0957E-1‡	**8.2511E-1**
M6H	8.3406E-1‡	8.0049E-1‡	8.5144E-1‡	**8.6378E-1**
M7H	8.5374E-1‡	8.1347E-1‡	8.5947E-1‡	**8.9491E-1**
M8H	**8.9624E-1**†	8.3916E-1‡	8.5701E-1†	8.5333E-1
M1L	6.1781E-1†	2.8872E-1‡	**6.2488E-1**†	6.1679E-1
M2L	5.9595E-1‡	3.6951E-1‡	5.3699E-1‡	**6.2764E-1**
M3L	6.4971E-1‡	5.8137E-1‡	**7.4112E-1**†	6.7929E-1
M4L	7.1886E-1‡	5.0987E-1‡	6.9977E-1‡	**7.3552E-1**
M5L	7.4327E-1‡	6.4799E-1‡	7.7322E-1‡	**8.2637E-1**
M6L	8.3124E-1‡	7.1798E-1‡	7.9113E-1‡	**8.3479E-1**
M7L	**8.6691E-1**†	7.8764E-1‡	8.3355E-1†	8.1598E-1
M8L	8.5746E-1‡	8.0173E-1‡	8.3876E-1‡	**8.6361E-1**

between consecutive environments, while the Low similarity environment corresponds to more drastic variations. For each instance, a sequence of ten dynamic environments is generated.

B. Compared Algorithms and Parameter Settings

The proposed DC-DMOEA is compared with representative dynamic multiobjective evolutionary algorithms, including SGEA [14], D-SVR [15], and DVC [16]. All algorithms are implemented under the same experimental framework and evaluated using identical problem instances.

For fairness, the population size, crossover probability, mutation probability, and termination criteria are set identically for all algorithms. Each algorithm is independently executed ten times on every test instance under each environment similarity level.

C. Performance Metrics

To evaluate the performance of the algorithms in dynamic environments, two widely used metrics are employed: Inverted Generational Distance (IGD) and Hypervolume (HV). Lower IGD values and higher HV values indicate better convergence and diversity of the obtained Pareto fronts.

Following the evaluation protocol of DG-DMOEA, the mean performance over dynamic environments and independent runs is reported. Specifically, the Mean IGD (MIGD) and Mean HV (MHV) are calculated as:

$$\mathrm{MIGD} = \frac{1}{R}\sum_{r=1}^{R}\left(\frac{1}{L}\sum_{\ell=1}^{L}\mathrm{IGD}_{r,\ell}\right), \tag{15}$$

$$\mathrm{MHV} = \frac{1}{R}\sum_{r=1}^{R}\left(\frac{1}{L}\sum_{\ell=1}^{L}\mathrm{HV}_{r,\ell}\right), \tag{16}$$

where $R = 10$ denotes the number of independent runs and $L = 10$ represents the number of dynamic environments. This evaluation strategy reflects the long-term performance and stability of algorithms in dynamic optimization scenarios.

V. RESULTS AND DISCUSSION

This section reports and analyzes the experimental results of DC-DMOEA in comparison with state-of-the-art dynamic multiobjective optimization algorithms. The performance is evaluated using MIGD and MHV metrics on both small-scale and large-scale dynamic disassembly line balancing instances under different environment similarity levels.

A. Overall Performance Comparison

Tables I and II summarize the MIGD and MHV results obtained by all compared algorithms on the proposed benchmark instances. For MIGD, smaller values indicate better convergence to the true Pareto front, while for MHV, larger values reflect better diversity and coverage. In the tables, ‡ and † indicate that DC-DMOEA performs significantly better than and equivalently to the corresponding competitor, respectively.

Overall, DC-DMOEA achieves the best performance on the majority of test instances. In terms of MIGD, DC-DMOEA consistently outperforms SGEA, D-SVR, and DVC on both small-scale (M1–M4) and large-scale (M5–M8) instances. The improvement is particularly pronounced in low-similarity environments, where the dynamic changes between consecutive environments are more drastic.

For MHV, DC-DMOEA also exhibits superior performance in most cases. The advantage becomes more evident as the problem scale increases. On large-scale instances, DC-DMOEA achieves noticeably higher hypervolume values than the competing algorithms, indicating its stronger ability to maintain population diversity under complex and dynamic conditions.

These results demonstrate that DC-DMOEA provides a favorable balance between convergence and diversity across different problem scales and environment similarity levels.

B. Impact of Environment Similarity

A clear performance gap can be observed between high-similarity and low-similarity environments. In high-similarity

scenarios, all algorithms are able to exploit temporal continuity to some extent, leading to relatively small performance differences. However, DC-DMOEA still maintains the lowest MIGD and highest MHV in most high-similarity instances, suggesting effective utilization of historical knowledge.

In low-similarity environments, where environmental changes are abrupt, the performance degradation of baseline algorithms becomes more severe. By contrast, DC-DMOEA shows significantly improved robustness. The clustering-based historical Pareto set reuse mechanism allows DC-DMOEA to selectively transfer relevant knowledge from similar past environments while suppressing negative transfer from dissimilar ones.

This adaptive initialization strategy enables DC-DMOEA to rapidly recover after environmental changes, resulting in smaller performance oscillations and improved long-term optimization quality.

C. Small-Scale versus Large-Scale Instances

On small-scale instances (M1–M4), DC-DMOEA consistently achieves competitive MIGD and MHV values. Although the performance gap between algorithms is relatively modest due to the limited problem size, DC-DMOEA still demonstrates stable superiority.

On large-scale instances (M5–M6), the advantage of DC-DMOEA becomes more pronounced. As the number of disassembly tasks and precedence constraints increases, the search space expands rapidly, posing greater challenges to dynamic optimization. The proposed clustering-based population initialization effectively guides the search toward promising regions, enabling DC-DMOEA to maintain both convergence speed and solution diversity.

These observations indicate that DC-DMOEA scales well with problem size and is particularly suitable for large-scale dynamic disassembly line balancing problems.

VI. Conclusion

The superior performance of DC-DMOEA can be attributed to three key factors. First, the HDBSCAN-based clustering strategy captures the intrinsic structure of historical Pareto-optimal solutions, allowing meaningful grouping without requiring a predefined number of clusters. Second, the environment-shape descriptors provide a compact yet informative representation of environmental characteristics, facilitating accurate similarity estimation. Third, the similarity-driven weighted sampling mechanism ensures that the initial population is biased toward historically relevant solutions while preserving diversity.

Together, these components enable DC-DMOEA to effectively exploit long-term historical knowledge and adapt to dynamic environmental changes. The experimental results confirm that the proposed framework offers a robust and scalable solution for dynamic multiobjective optimization in disassembly line balancing scenarios.

References

[1] A. Güngör and S. M. Gupta, "Disassembly line in product recovery," *International Journal of Production Research*, vol. 40, no. 11, pp. 2569–2589, 2002.

[2] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 6, pp. 1–24, 2012.

[3] S. Jiang, J Zou, and S Yang, "Evolutionary dynamic multi-objective optimisation: A survey," *ACM Computing Surveys*, vol. 55, no. 4, pp. 1–47, 2022.

[4] K. Yu, D. Zhang, J. Liang, K. Chen, C. Yue, K. Qiao, and L. Wang, "A correlation-guided layered prediction approach for evolutionary dynamic multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 5, pp. 1398–1412, 2022.

[5] Y. Fang, F. Liu, and M. Li, "Domain generalization-based dynamic multiobjective optimization: A case study on disassembly line balancing," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 6, pp. 1851–1865, 2023.

[6] Y. Laili, Y. Wang, and Y.Fang, "Solutions for mixed-model disassembly line balancing with multi-robot workstations," *Optimisation of robotic disassembly for remanufacturing*, pp. 153–180, 2021.

[7] X. Guo, C. Fan, M. Zhou, S. Liu, J. Wang, S. Qin, and Y. Tang, "Human–robot collaborative disassembly line balancing problem with stochastic operation time and a solution via multi-objective shuffled frog leaping algorithm," *IEEE Transactions on Automation Science and Engineering*, vol. 21, no. 3, pp. 4448–4459, 2024.

[8] O. Laouini, I. Slama, F. Hnaien, and Z. Jemai, "Multi-objective disassembly line optimization with collaborative robots," *IFAC-PapersOnLine*, vol. 59, no. 10, pp. 49–54, 2025.

[9] W. Zhang, Y. Li, K. Wang, W. Xu, and L. Gao, "A green and efficient disassembly line balancing with human-robot collaboration and destructive disassembly," *Robotics and Computer-Integrated Manufacturing* , vol. 97, pp. 103081, 2025.

[10] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, and C. A. Coello Coello, "Survey of multiobjective evolutionary algorithms for data mining," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 1, pp. 4–38, 2014.

[11] R. J. G. B. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," *Pacific-Asia conference on knowledge discovery and data mining*, pp. 160–172, 2013.

[12] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[13] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[14] S. Jiang, Z. Yang, and S. Yao, "A steady-state and generational evolutionary algorithm for dynamic multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 1, pp. 65–82, 2016.

[15] L. Cao, L. Xu, E. D. Goodman, C. Bao, and S. Zhu, "Evolutionary dynamic multiobjective optimization assisted by a support vector regression predictor," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 305–319, 2020.

[16] Z. Liang, T. Wu, and X. Ma, "A Dynamic Multiobjective Evolutionary Algorithm Based on Decision Variable Classification," *IEEE Transactions on Cybernetics*, vol. 52, no. 3, pp. 1602–1615, 2022.