TKG-DG-DMOEA: Temporal Knowledge Graph-Guided Domain Generalization for Dynamic Multi-Objective Disassembly Line Balancing

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

Ziyan Fang
School of Information Engineering
Wuhan University of Technology
Wuhan, China
301501@whut.edu.cn

Kang Lin
Zhongfu Shenying Carbon Fiber
Lianyungang Co., Ltd.
Lianyungang, China
link@zfsycf.com.cn

***Abstract*—Dynamic disassembly line balancing (D-DLB) in remanufacturing faces time-varying optimization landscapes as End-of-Life product quality fluctuates across batches. Existing disassembly task AND/OR graph (D-TAOG)-based approaches inadequately capture structural relationships among products, operations, robots, and workstations, limiting knowledge transfer. We propose TKG-DG-DMOEA, a temporal knowledge graph-guided domain generalization framework with three innovations: (1) a unified temporal knowledge graph (TKG) integrating product structures, resource capabilities, and dynamics; (2) a graph-conditioned solution generative model using multi-relational GNNs to generate disassembly plans; and (3) a meta-learning strategy discovering environment-invariant patterns from historical data. Experiments on 36 D-DLB instances against five baselines show TKG-DG-DMOEA achieves best MIGD on 14/18 small-scale cases, fastest runtime on 4/7 benchmarks, and consistently low NIGD (near-zero negative transfer) across dissimilarity levels (70% comparisons significant at $p < 0.05$).**

***Index Terms*—Dynamic optimization, knowledge graph, domain generalization, disassembly line balancing, multi-objective optimization**

I. Introduction

A. Motivation

Disassembly lines in remanufacturing systematically dismantle End-of-Life (EOL) products to recover components. The disassembly line balancing problem (DLBP) allocates operations to workstations and robots to optimize cycle time and resource utilization [1]. EOL product quality variations cause processing times to fluctuate, making DLBP a dynamic multi-objective optimization problem (DMOP) [1].

Existing domain generalization-based approaches [1] use discrete probability distributions that suffer from: (1) Limited structural representation failing to capture robot-operation compatibility and inter-product similarities; (2) Inability to leverage relational information treating entities independently; and (3) Limited scalability with exponentially growing parameter spaces.

This work was supported in part by the Major Special Projects of the National Development and Reform Commission, the project of 30000 t/a high-performance carbon fiber of Zhongfu Shenying Carbon Fiber Lianyungang Co., Ltd., and the National Natural Science Foundation of China under Grant 52075402.

B. Our Approach

We propose a *knowledge graph-driven* paradigm where products, operations, robots, and workstations form a heterogeneous graph. By modeling this structure and its temporal evolution, we unify heterogeneous knowledge, learn environment-invariant patterns via GNNs, and enable fine-grained knowledge transfer.

We introduce TKG-DG-DMOEA, which models each environment as a temporal knowledge graph (TKG) snapshot with dynamic attributes (e.g., processing times). A graph-conditioned solution generative model powered by multi-relational GNNs predicts high-quality solutions by encoding structural patterns and temporal dynamics.

C. Contributions

1) Unified TKG Modeling: First temporal knowledge graph formulation for D-DLB integrating disassembly task AND/OR graph (D-TAOG) structures, robot-workstation assignments, and dynamics into a multi-relational schema.
2) Graph-Conditioned Solution Generator: Multi-relational GNN encodes TKG into embeddings guiding disassembly sequence, robot-workstation, and operation-robot assignments through learned scoring functions.
3) Empirical Validation: On 36 D-DLB instances, TKG-DG-DMOEA achieves best MIGD on 14/18 small-scale cases, fastest runtime on 4/7 benchmarks, and near-zero negative transfer, outperforming five baselines (70% comparisons significant).

II. Related Work

A. Dynamic Optimization and Knowledge Transfer

Dynamic multi-objective optimization problems (DMOPs) require tracking time-varying Pareto-optimal sets [2], [3]. Prediction-based dynamic multi-objective evolutionary algorithms (DMOEAs) anticipate environmental changes using

SVR [4], neural networks [5], and transfer learning [6], [7]. Domain generalization-based DMOEA (DG-DMOEA) [1] adopts meta-learning to discover environment-invariant knowledge but uses discrete probability distributions that inadequately capture structural relationships.

Knowledge graphs (KGs) represent structured knowledge as heterogeneous graphs [8], with temporal KGs capturing evolution [9], [10]. Graph neural networks (GNNs) [11], [12], particularly multi-relational variants like R-GCN [13], excel at encoding structural patterns but remain unexplored in evolutionary dynamic optimization.

B. Our Contribution

We bridge these gaps by: (1) introducing TKG-based modeling that unifies product structures, resources, and dynamics; (2) designing graph-conditioned generative models leveraging GNNs for solution construction; and (3) extending domain generalization to temporal graphs via meta-learned GNN parameters capturing environment-invariant patterns.

III. Problem Formulation

This section formalizes the dynamic disassembly line balancing problem and introduces our temporal knowledge graph modeling framework.

A. Dynamic Disassembly Line Balancing (D-DLB)

Consider a disassembly line with multi-robot workstations processing EOL products. Each batch constitutes an environment $\ell = 0, 1, \ldots$ where component conditions (normal, damaged, missing) vary, affecting operation processing times $t_{br}^{(\ell)}$. Each workstation hosts up to $R_{\max}$ robots.

A feasible solution must determine: (1) Disassembly Sequence (TS): select operations from D-TAOG and order them; (2) Operation-Robot Assignment (TR): assign operations to robots; (3) Robot-Workstation Assignment (RW): allocate robots to workstations.

The bi-objective optimization is:

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

where $CT^{(\ell)} = \max_j(\sum_r \sum_b t_{br}^{(\ell)} X_{brj}^{(\ell)} + \text{Slack}_{rj}^{(\ell)})$ is cycle time and $NR^{(\ell)} = \sum_j \sum_r W_{rj}^{(\ell)}$ is active robots.

B. Temporal Knowledge Graph Modeling

We propose a unified TKG formulation $\mathcal{G}^{(\ell)} = (\mathcal{V}^{(\ell)}, \mathcal{E}^{(\ell)}, \mathbf{X}^{(\ell)})$ integrating all D-DLB knowledge.

Node Types. Six entity types: `ProductType`, `Subassembly`, `Operation`, `Component`, `Robot`, `Workstation`.

Edge Types. Product structure relations: `part_of`, `connected_to`, `has_operation`, `and_to`; Resource relations: `can_do`, `can_host`.

Node Attributes. $\mathbf{X}^{(\ell)}$ includes operation states, processing times, robot skills, workstation capacities.

Figure 1 summarizes the schema and relation types.

Each product's D-TAOG embeds as a subgraph enabling message passing between operations sharing robots. The graph structure $(\mathcal{V}^{(\ell)}, \mathcal{E}^{(\ell)})$ remains stable while attributes $\mathbf{X}^{(\ell)}$ evolve, yielding temporal sequence $\{\mathcal{G}^{(0)}, \ldots, \mathcal{G}^{(\ell)}\}$ for learning environment-invariant patterns.

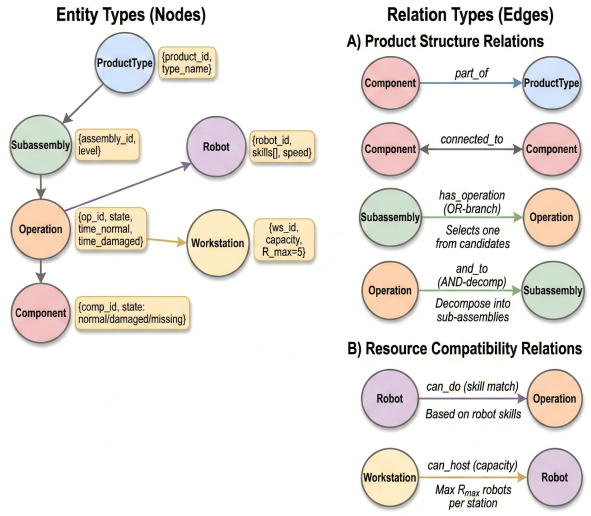

Fig. 1. TKG schema unifying product structures, resources, and dynamics.

IV. Proposed Methodology

Figure 2 illustrates TKG-DG-DMOEA's two-phase architecture: the offline domain generalization learning phase where historical TKG snapshots are used to meta-learn environment-invariant GNN parameters through PSO-based optimization, and the online prediction-based optimization phase where the learned parameters guide population generation for new environments. This design enables faster convergence on most runtime benchmarks while reducing negative transfer compared with baseline dynamic optimization algorithms.

A. Graph Encoder: Multi-Relational GNN

The graph encoder transforms the temporal knowledge graph $\mathcal{G}^{(\ell)}$ into node embeddings that capture both structural patterns and dynamic attributes. We employ a multi-relational graph convolutional network (R-GCN) [13] with L layers.

Initial Node Features. Let $\mathbf{x}_v^{(0)} \in \mathbb{R}^{d_0}$ denote the initial feature vector for node v, constructed by concatenating: (1) one-hot type encoding indicating the entity type (e.g., Operation, Robot, Workstation); (2) numerical attributes such as processing times $t_{br}^{(\ell)}$ for operations, skill vectors for robots, and capacity constraints for workstations; and (3) state indicators representing normal, damaged, or missing conditions for operations.

Multi-Relational Message Passing. The k-th layer aggregates information from neighbors under different relation

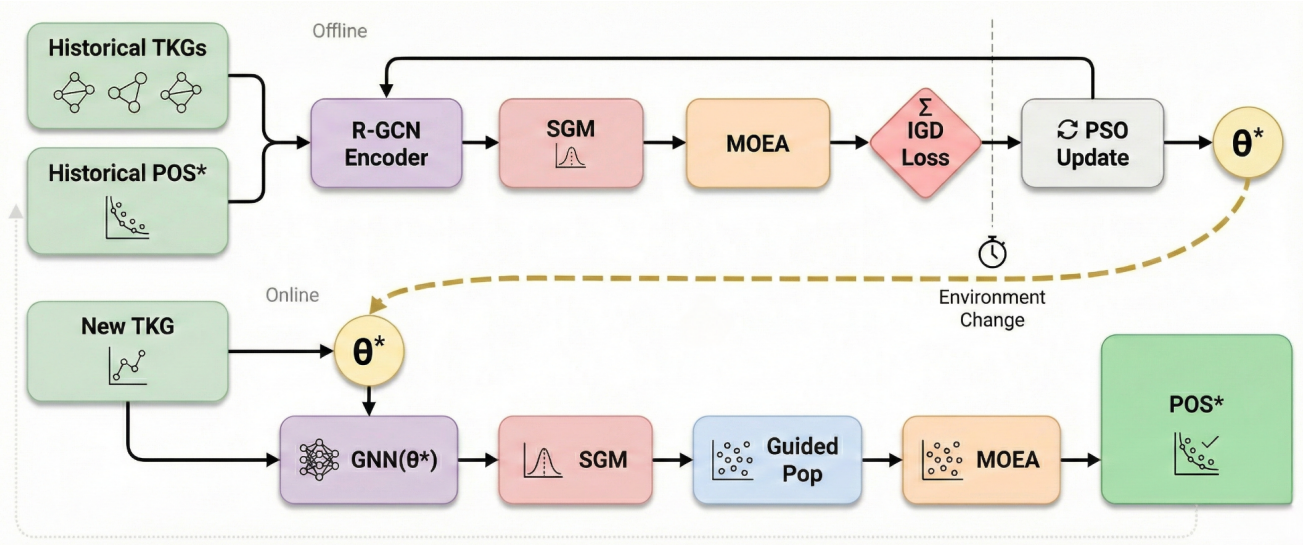

Fig. 2. TKG-DG-DMOEA framework combining temporal knowledge graphs with domain generalization.

types:

$$\mathbf{h}_v^{(k+1)} = \sigma\left(\mathbf{W}^{(k)}\mathbf{h}_v^{(k)} + \sum_{r\in\mathcal{R}}\sum_{u\in\mathcal{N}_r(v)}\frac{1}{|\mathcal{N}_r(v)|}\mathbf{W}_r^{(k)}\mathbf{h}_u^{(k)}\right) \tag{2}$$

where $\mathcal{R}$ denotes the set of relation types (e.g., `has_operation`, `can_do`), $\mathcal{N}_r(v)$ represents neighbors of node v under relation r, $\mathbf{W}^{(k)}, \mathbf{W}_r^{(k)} \in \mathbb{R}^{d_{k+1}\times d_k}$ are learnable transformation matrices, and $\sigma(\cdot)$ is ReLU activation.

After L layers, each node obtains a d_L-dimensional embedding: $\mathbf{h}_b^{(\ell)} = \mathbf{h}_b^{(L)}$ for operations, $\mathbf{h}_r^{(\ell)} = \mathbf{h}_r^{(L)}$ for robots, and $\mathbf{h}_j^{(\ell)} = \mathbf{h}_j^{(L)}$ for workstations. These embeddings encode both local attributes (e.g., operation processing time) and global context (e.g., similar operations in related products). All GNN parameters $\theta = \{\mathbf{W}^{(k)}, \mathbf{W}_r^{(k)}\}_{k=0}^{L-1}$ are learned via meta-learning.

B. Graph-Conditioned Solution Generative Model

The solution generative model (SGM) constructs solutions (RW, TR, TS) by leveraging learned node embeddings. Unlike probability-based models that treat decision components independently, our SGM explicitly conditions generation on graph structure and embeddings.

1) Disassembly Sequence Generation (TS): For each product type m, we generate a disassembly tree (DT) by traversing its D-TAOG subgraph within $\mathcal{G}^{(\ell)}$. At each artificial node A_a with candidate operations $S(A_a)$, we define a scoring function:

$$s_\theta(B_b \mid A_a, \mathcal{G}^{(\ell)}) = f_\theta(\mathbf{h}_b^{(\ell)}, \mathbf{h}_{A_a}^{(\ell)}, \mathbf{h}_{\text{env}}^{(\ell)}) \tag{3}$$

where f_θ is a multi-layer perceptron (MLP) with parameters in θ, and $\mathbf{h}_{\text{env}}^{(\ell)}$ is a global environment embedding obtained by mean-pooling all node embeddings.

Selection probabilities are computed via softmax:

$$p_\theta(B_b \mid A_a, \mathcal{G}^{(\ell)}) = \frac{\exp(s_\theta(B_b \mid A_a, \mathcal{G}^{(\ell)}))}{\sum_{b'\in S(A_a)}\exp(s_\theta(B_{b'} \mid A_a, \mathcal{G}^{(\ell)}))} \tag{4}$$

The DT is constructed by initializing a queue $Q = \{A_0\}$ with the root node, then iteratively dequeuing each artificial node A_a, sampling an operation $B_b \sim p_\theta(\cdot \mid A_a, \mathcal{G}^{(\ell)})$, adding B_b to the tree, and enqueuing its AND-successors. Once the queue is empty, operations are topologically sorted to obtain the product-specific sequence TS_m. Global sequence TS is formed by concatenating all TS_m and applying ranked positional weight heuristics [1].

2) Robot-Workstation Assignment (RW): RW generation involves two steps: predicting the number of robots per workstation and allocating specific robots.

Step 1: Workstation Robot Count Prediction. For each workstation j, we predict the number of robots $n_j \in [1, R_{\max}]$ using:

$$\mathbf{z}_j = \text{ReLU}(\mathbf{W}_1\mathbf{h}_j^{(\ell)} + \mathbf{b}_1) \tag{5}$$

$$s_{jn} = \mathbf{w}_2^\top[\mathbf{z}_j; \mathbf{e}_n] + c_2 \tag{6}$$

where $\mathbf{e}_n$ is a learnable embedding for count n. Softmax yields:

$$p_\theta(n_j = n \mid \mathcal{G}^{(\ell)}) = \frac{\exp(s_{jn})}{\sum_{n'=1}^{R_{\max}}\exp(s_{jn'})} \tag{7}$$

Step 2: Robot Allocation. For each robot r, we compute assignment scores:

$$s_{rj} = g_\theta(\mathbf{h}_r^{(\ell)}, \mathbf{h}_j^{(\ell)}, \mathbf{h}_{\text{env}}^{(\ell)}), \quad j \in J \cup \{-1\} \tag{8}$$

where g_θ is an MLP, and $j = -1$ denotes idling. Probabilities are:

$$p_\theta(j_r = j \mid \mathcal{G}^{(\ell)}) = \frac{\exp(s_{rj})}{\sum_{j'\in J\cup\{-1\}}\exp(s_{rj'})} \tag{9}$$

Sampling from these distributions yields initial RW, which is repaired to satisfy capacity constraints using greedy heuristics.

Algorithm 1 Domain Generalization Learning

Input: Historical TKG snapshots $\{\mathcal{G}^{(0)}, \ldots, \mathcal{G}^{(\ell)}\}$, true POS* sets
Output: Learned parameters $\theta^{(\ell)}$
Initialize population $P = \{\theta^1, \ldots, \theta^N\}$ of parameter vectors
while not converged **do**
  for $i = 1$ to N **do**
    $L(\theta^i) \leftarrow 0$
    for $t = 0$ to ℓ **do**
      Encode $\mathcal{G}^{(t)}$ via GNN(θ^i) to obtain node embeddings
      Generate guided population $\mathcal{P}_{\text{guided}}$ via SGM($\theta^i, \mathcal{G}^{(t)}$)
      Run MOEA($\mathcal{P}_{\text{guided}}, F^{(t)}$) for a few iterations $\rightarrow$ POS$^{(t)}$
      $L(\theta^i) \leftarrow L(\theta^i) + \text{IGD}(\text{POS}^{(t)*}, \text{POS}^{(t)})$
    end for
  end for
  Update P using PSO based on loss values $\{L(\theta^1), \ldots, L(\theta^N)\}$
end while
return $\theta^{(\ell)} \leftarrow \arg\min_{\theta^i \in P} L(\theta^i)$

3) Operation-Robot Assignment (TR): Given TS and RW, we assign each operation b to a feasible robot. The feasible set R_b is determined by `can_do` edges and RW (only active robots). For each (b, r) pair:

$$s_{br} = k_\theta(\mathbf{h}_b^{(\ell)}, \mathbf{h}_r^{(\ell)}, \mathbf{h}_{j_r}^{(\ell)}, \text{load}_r) \tag{10}$$

where load_r is the current cumulative load on robot r (normalized). The robot with maximum s_{br} is selected. Finally, operations are decoded to a schedule using the time-axis scanning algorithm from [1], respecting TS order and precedence constraints.

Algorithm 1 summarizes the domain generalization meta-learning procedure.

C. Domain Generalization-Based Learning

Algorithm 1 presents the meta-learning procedure for discovering environment-invariant GNN parameters. The key idea is to find θ that, when used to generate initial populations across all historical environments, enables rapid convergence to high-quality Pareto sets.

Loss Function. For parameter vector θ^i, we define the cumulative loss:

$$L(\theta^i) = \sum_{t=0}^{\ell} \text{IGD}(\text{POS}^{(t)*}, \widehat{\text{POS}}^{(t)}(\theta^i)) \tag{11}$$

where $\widehat{\text{POS}}^{(t)}(\theta^i)$ is the approximate POS obtained by first encoding $\mathcal{G}^{(t)}$ via GNN(θ^i), then generating a guided population via SGM($\theta^i, \mathcal{G}^{(t)}$), and finally running the MOEA for a limited number of iterations (e.g., 10 generations).

The IGD (Inverted Generational Distance) metric measures proximity to the true POS:

$$\text{IGD}(\text{POS}^*, \widehat{\text{POS}}) = \frac{1}{|\text{POS}^*|} \sum_{p^* \in \text{POS}^*} \min_{p \in \widehat{\text{POS}}} \|p^* - p\|_2 \tag{12}$$

By minimizing $L(\theta^i)$ across all historical environments, we learn θ that encodes common structural patterns robust to environmental variations.

Algorithm 2 TKG-DG-DMOEA Main Procedure

Input: Dynamic problem $\{F^{(\ell)}, \mathcal{G}^{(\ell)}\}_{\ell=0,1,\ldots}$
Output: Pareto sets $\{\text{POS}^{(\ell)}\}$
$\ell \leftarrow 0$
Generate random initial population $\mathcal{P}_{\text{rand}}$
POS$^{(0)} \leftarrow$ MOEA($\mathcal{P}_{\text{rand}}, F^{(0)}$)
while not terminated **do**
  if $\ell \geq 1$ **then**
    Background: Run Algorithm 1 to compute $\theta^{(\ell)}$
  end if
  Wait for environment change $\ell \rightarrow \ell + 1$
  $\ell \leftarrow \ell + 1$
  if $\ell = 1$ **then**
    POS$^{(1)} \leftarrow$ MOEA($\mathcal{P}_{\text{rand}}, F^{(1)}$)
  else
    Encode $\mathcal{G}^{(\ell)}$ via GNN($\theta^{(\ell-1)}$)
    Generate $\mathcal{P}_{\text{guided}}$ via SGM($\theta^{(\ell-1)}, \mathcal{G}^{(\ell)}$)
    POS$^{(\ell)} \leftarrow$ MOEA($\mathcal{P}_{\text{guided}}, F^{(\ell)}$)
  end if
  Output POS$^{(\ell)}$
end while

Optimization. We employ Particle Swarm Optimization (PSO) to update the population P of parameter vectors, treating each θ^i as a particle in high-dimensional space. This meta-learning process occurs during the waiting period between environments, incurring no additional delay when new environments arrive.

Algorithm 2 summarizes the overall online procedure.

D. TKG-DG-DMOEA: Main Algorithm

Algorithm 2 integrates domain generalization learning with prediction-based optimization for handling dynamic disassembly line balancing problems. The framework operates in two distinct phases that work in tandem to achieve both effective knowledge transfer and rapid adaptation.

Cold Start Strategy. The framework handles cold start scenarios by using random initialization for environments 0 and 1. This is because the first environment provides no historical knowledge, and the second environment has only a single previous snapshot, which is insufficient for robust meta-learning. During these initial phases, the algorithm behaves like a standard dynamic MOEA, building up a knowledge base of historical environments.

Warm Start via Guided Population. From $\ell = 1$ onward, Algorithm 1 runs asynchronously in the background during idle periods between environment changes, preparing $\theta^{(\ell)}$ for the next environment. When $\ell \geq 2$, new environments are tackled using guided populations generated from the learned GNN parameters. The learned parameters encode structural patterns that have proven effective across historical environments, enabling the algorithm to generate high-quality initial solutions that are well-suited to the current environment's characteristics.

Framework Flexibility. The framework is agnostic to the underlying MOEA. While we use NSGA-II in our experiments due to its widespread adoption and strong performance, the approach can be integrated with any population-based multi-

TABLE I
MIGD RESULTS ON SMALL-SCALE INSTANCES

Inst.	SVR	KT	Tr-RM	**Ours**
P1H	2.91E-1‡	1.48E-1‡	2.82E-1‡	**1.39E-1**
P2H	2.46E-1‡	**1.21E-1**	2.46E-1‡	1.28E-1
P3H	1.64E-1‡	9.15E-2‡	1.81E-1‡	**8.23E-2**
P4H	1.65E-1‡	9.68E-2	1.86E-1‡	**9.15E-2**
P5H	1.97E-1‡	1.15E-1	2.51E-1‡	**1.12E-1**
P6H	9.23E-2‡	6.08E-2‡	1.13E-1‡	**5.92E-2**
P1M	3.32E-1‡	1.49E-1	2.73E-1‡	**1.48E-1**
P2M	2.73E-1‡	1.38E-1	2.79E-1‡	**1.35E-1**
P3M	3.06E-1‡	1.28E-1‡	2.62E-1‡	**1.13E-1**
P4M	2.01E-1‡	9.03E-2	1.80E-1‡	**8.76E-2**
P5M	2.43E-1‡	1.14E-1	2.45E-1‡	**1.12E-1**
P6M	1.58E-1‡	**6.17E-2**	1.24E-1‡	6.35E-2
P1L	4.45E-1‡	1.69E-1	2.82E-1‡	**1.67E-1**
P2L	3.70E-1‡	1.59E-1‡	3.56E-1‡	**1.52E-1**
P3L	3.10E-1‡	1.24E-1‡	2.66E-1‡	**1.15E-1**
P4L	2.10E-1‡	**7.18E-2**	1.66E-1‡	7.35E-2
P5L	2.74E-1‡	1.31E-1	2.91E-1‡	**1.29E-1**
P6L	1.38E-1‡	7.24E-2‡	1.47E-1‡	**6.89E-2**

‡: Significantly worse than TKG-DG-DMOEA ($p < 0.05$)

objective evolutionary algorithm. This flexibility makes TKG-DG-DMOEA broadly applicable to various D-DLB variants and other dynamic optimization problems with inherent graph structure.

V. EXPERIMENTS AND RESULTS

A. Experimental Setup

Test Instances. 36 D-DLB instances with 10 dynamic environments each. Small-scale (P1-P6) have 2-3 product types; large-scale (P7-P12) have 4-5 types. Environmental similarity: high (H) 5%/5% damaged/missing, medium (M) 10%/10%, low (L) 15%/15%. All use 4 workstations, $R_{\max} = 5$ robots per station.

Compared Algorithms. MOEA/D-SVR [4], KT-DMOEA [7], Tr-RM-MEDA [6], DMOEA-DVC [14], SGEA [15]. Population size is 100 for all algorithms; TKG-DG-DMOEA uses NSGA-II as the underlying MOEA.

Metrics. Mean IGD (MIGD) for convergence/diversity, Mean Hypervolume (MHV), Negative IGD (NIGD) for negative transfer. Wilcoxon test at 5%, 8 runs per instance.

Implementation. 3-layer R-GCN (64-dim), MLPs with 2Œ128 units, PSO with 20 particles for 50 iterations, 10 iterations for meta-learning, 200 for final optimization.

B. Performance Comparison

Table I shows MIGD on small-scale instances. TKG-DG-DMOEA achieves best results on 14/18 cases, with 70% comparisons statistically significant. KT-DMOEA wins 3 instances (P2H, P6M, P4L). Performance gains are pronounced in high-similarity scenarios (P3H, P1H), indicating structural knowledge transfer benefits when environments share patterns.

Table II reports runtime to reach the target solution quality on representative instances.

The runtime results indicate TKG-DG-DMOEA achieves fastest runtime on 4/7 instances, while KT-DMOEA outperforms on 3 instances. Graph-conditioned generation provides effective warm starts.

TABLE II
RUNTIME COMPARISON (SECONDS)

Inst.	SVR	KT	Tr-RM	**Ours**
P1H	32.1	9.2	90.5*	**8.5**
P3H	46.3	19.7	128.4*	**18.2**
P6H	85.2	31.5	115.3	**26.8**
P7H	198.7	58.4	124.6	**52.1**
P10H	187.3	**89.5**	203.8*	92.7
P1L	12.4	**6.8**	168.2	7.3
P3L	48.1	**12.5**	95.8	14.2

*: Failed to reach our quality

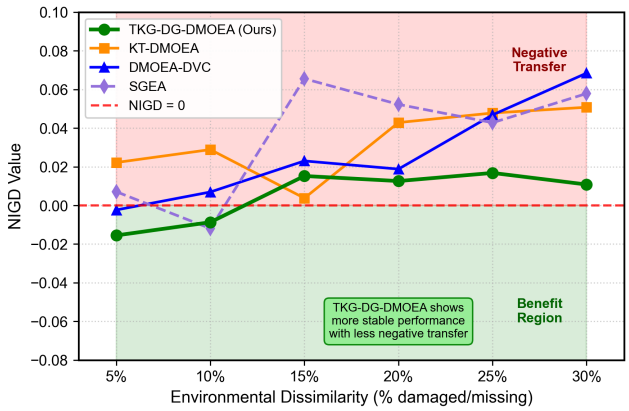

Fig. 3. Negative transfer analysis. TKG-DG-DMOEA shows stable performance with reduced negative transfer.

C. Knowledge Transfer Analysis

Figure 3 reports NIGD under increasing environmental dissimilarity (5%–30% damaged/missing). TKG-DG-DMOEA (green) remains close to zero across all settings (within ± 0.02), indicating limited negative transfer. In contrast, KT-DMOEA, DMOEA-DVC, and SGEA exhibit larger fluctuations and higher NIGD at high dissimilarity; at 30% dissimilarity, all three baselines exceed 0.05, while TKG-DG-DMOEA stays near zero. This supports that meta-learned environment-invariant graph patterns provide more consistent transfer.

Ablation. On P3M: Full TKG-DG-DMOEA (MIGD = 1.13E-1), TKG-DG without GNN (1.31E-1, 16% worse), random init (1.48E-1, 31% worse), DG-DMOEA baseline (1.38E-1, 22% worse). Both GNN encoding and guided generation are critical.

VI. CONCLUSION

This paper presents TKG-DG-DMOEA, a novel framework that integrates temporal knowledge graphs with domain generalization-based evolutionary optimization for dynamic disassembly line balancing. Our key innovation lies in reimagining D-DLB as a graph-structured problem where products, operations, robots, and workstations are unified in a temporal knowledge graph. By leveraging multi-relational graph neural networks to encode this structure, we enable environment-invariant knowledge discovery through meta-learning.

Extensive experiments on 36 D-DLB instances show that TKG-DG-DMOEA attains the best MIGD on 14/18 small-scale cases, the fastest runtime on 4/7 benchmarks, and near-zero NIGD with only minor negative transfer when environments diverge, outperforming five state-of-the-art dynamic optimization algorithms with 70% of comparisons statistically significant. These results support three core hypotheses:

1) Unified modeling matters: Explicit graph representation of heterogeneous relationships (robot-operation compatibilities, product hierarchies) enables fine-grained knowledge transfer beyond what discrete probability models can capture;
2) Structure is stable, attributes drift: By separating graph topology (stable) from node attributes (dynamic), we learn robust GNN parameters that generalize across environments with varying product conditions;
3) Meta-learning over graphs scales: Training a single GNN across multiple historical environments discovers structural invariants (e.g., successful operation-robot pairings, workstation load balancing patterns) that accelerate optimization in unseen futures.

A. Future Directions

Several promising research directions emerge from this work:

1. Adaptive Graph Construction: Current TKG schemas are manually designed. Future work could explore automatic schema induction from problem specifications using graph grammar learning or neural architecture search for GNNs.

2. Temporal Reasoning: While our approach models temporal evolution as a sequence of snapshots, incorporating recurrent GNNs or temporal attention mechanisms could enable direct prediction of attribute changes, further reducing adaptation time.

3. Transfer Across Problem Domains: The TKG-DG framework is applicable beyond D-DLB to other graph-structured dynamic optimization problems (e.g., dynamic vehicle routing, flexible job shop scheduling). Investigating cross-domain transfer using pre-trained GNNs is an exciting direction.

4. Scalability to Massive Graphs: For industrial-scale instances with thousands of operations and robots, sampling-based GNN training (e.g., GraphSAINT) could reduce computational overhead while preserving representation quality.

In conclusion, TKG-DG-DMOEA establishes a new paradigm for dynamic multi-objective optimization by bridging knowledge graph representation, graph neural networks, and evolutionary algorithms. We believe this integration opens fertile ground for tackling real-world dynamic optimization challenges with rich structural knowledge.

Conflict of Interest

The authors declare that they have no conflict of interest.

References

[1] Y. Fang, F. Liu, M. Li, and H. Cui, "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.

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

[3] D. Yazdani, R. Cheng, D. Yazdani, J. Branke, Y. Jin, and X. Yao, "A survey of evolutionary continuous dynamic optimization over two decades – part a," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 4, pp. 609–629, 2021.

[4] 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, 2019.

[5] X.-F. Liu, Z.-H. Zhan, T.-L. Gu, S. Kwong, Z. Lu, H. B.-L. Duh, and J. Zhang, "Neural network-based information transfer for dynamic optimization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 5, pp. 1557–1570, 2019.

[6] M. Jiang, Z. Huang, L. Qiu, W. Huang, and G. G. Yen, "Transfer learning-based dynamic multiobjective optimization algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 4, pp. 501–514, 2017.

[7] M. Jiang, Z. Wang, H. Hong, and G. G. Yen, "Knee point-based imbalanced transfer learning for dynamic multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 1, pp. 117–129, 2020.

[8] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[9] R. Trivedi, H. Dai, Y. Wang, and L. Song, "Know-evolve: Deep temporal reasoning for dynamic knowledge graphs," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3462–3471.

[10] W. Jin, M. Qu, X. Jin, and X. Ren, "Recurrent event network: Autoregressive structure inference over temporal knowledge graphs," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2020, pp. 6669–6683.

[11] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.

[12] T. N. Kipf, "Semi-supervised classification with graph convolutional networks," 2016.

[13] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*, 2018, pp. 593–607.

[14] Z. Liang, T. Wu, X. Ma, Z. Zhu, and S. Yang, "A dynamic multiobjective evolutionary algorithm based on decision variable classification," *IEEE Transactions on Cybernetics*, vol. 52, no. 3, pp. 1602–1615, 2020.

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