Dual-GNN-Driven Cooperative Optimization for Makespan-Minimized and Large-Scale 3C Dynamic Job-Shop Scheduling

Jing Bi, *Senior Member, IEEE*, Chen Wang, *Graduate Student Member, IEEE*, Ziqi Wang, *Graduate Student Member, IEEE*, Junqi Zhang, Haitao Yuan, *Senior Member, IEEE*, Jia Zhang, *Senior Member, IEEE*, and Rajkumar Buyya, *Fellow, IEEE*

***Abstract*—A Dynamic Job-shop Scheduling Problem (DJSP) in 3C (*i.e.*, Computer, Communication, and Consumer Electronics) manufacturing requires efficient resource allocation under dynamically changing production conditions where jobs arrive unpredictably. Traditional optimization methods struggle to provide scalable solutions due to the high computational cost of searching for the optimal schedules in large and complex environments. To address this challenge, this work proposes a Dual Graph convolutional networks-driven Dynamic Cooperative Hunting Optimizer (DG-DCHO). It integrates Graph Convolutional Networks (GCN) with metaheuristic optimization to generate high-quality schedules and significantly improve computational efficiency. A GCN generator processes graph representations of job-shop environment, captures complex dependencies among jobs and machines, and constructs high-quality initial schedules for the optimization process. A GCN evaluator estimates makespan values directly from schedule representations and replaces costly fitness evaluation, thereby minimizing computational overhead and improving optimization speed. A Dynamic Cooperative Hunting Optimizer serves as a base optimizer and generates scheduling solutions by balancing global exploration with local exploitation through an adaptive search strategy. Experimental results across various DJSP instances demonstrate that DG-DCHO consistently outperforms advanced scheduling algorithms by producing higher-quality solutions with reduced computational resources, establishing it as a scalable and effective framework for real-time dynamic scheduling of large-scale manufacturing systems.**

Received 29 April 2025; revised 21 August 2025, 31 October 2025, and 9 January 2026; accepted 1 February 2026. Date of publication 6 February 2026; date of current version 12 February 2026. This article was recommended for publication by Associate Editor B. Yan and Editor Q.-S. Jia upon evaluation of the reviewers' comments. This work was supported in part by Beijing Natural Science Foundation under Grant L233005 and Grant 4232049, in part by the National Natural Science Foundation of China under Grant 62473014 and Grant 62173013, and in part by the Beihang World Top University Cooperation Program. *(Corresponding author: Ziqi Wang.)*

Jing Bi, Chen Wang, and Junqi Zhang are with the College of Computer Science, Beijing University of Technology, Beijing 100124, China (e-mail: bijing@bjut.edu.cn).

Ziqi Wang is with the School of Software Technology, Zhejiang University, Ningbo 315100, China (e-mail: wangziqi0312@zju.edu.cn).

Haitao Yuan is with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China (e-mail: yuan@buaa.edu.cn).

Jia Zhang is with the Department of Computer Science, Lyle School of Engineering, Southern Methodist University, Dallas, TX 75205 USA (e-mail: jiazhang@smu.edu).

Rajkumar Buyya is with the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, University of Melbourne, Melbourne, VIC 3010, Australia (e-mail: rbuyya@unimelb.edu.au).

Digital Object Identifier 10.1109/TASE.2026.3661178

***Note to Practitioners*—This paper is motivated by the practical need to rapidly generate efficient production schedules for complex job shops. We propose a novel automated approach, DG-DCHO, which uses deep learning to learn the dependencies of the production environment and rapidly generate high-quality initial schedules. DG-DCHO also estimates schedule performance without relying on lengthy simulations, accelerating the optimization process with an adaptive algorithm. To apply this approach, practitioners would provide standard manufacturing data, including the sequence of operations required for each job, the constraints between operations, the list of available machines, the potential machine assignments for each operation, and the processing times. The system uses this information to automatically build the required graph model, where DG-DCHO optimizes and outputs the best scheduling sequence. This results in the faster generation of more efficient production schedules, improving responsiveness and productivity. Although the simulation results are strong, practical implementation requires integration with factory systems and initial training in artificial intelligence models. Our future plans to extend the proposed approach to addressing other dynamic optimization challenges in logistics, intelligent manufacturing, and real-time traffic management.**

***Index Terms*—Dynamic job-shop scheduling, graph convolutional networks, intelligent optimization algorithms, deep learning.**

I. Introduction

THE multi-constrained dynamic job-shop scheduling problem (DJSP) in 3C smart manufacturing is a notable optimization challenge in production systems [1], [2], [3]. The 3C industry, which includes Computer, Communication, and Consumer Electronics, is characterized by short product lifecycles and volatile demand, making agile and efficient resource allocation essential. As a result, the problem centers on assigning machines to jobs under uncertain and dynamically changing conditions. In such dynamic environments, the unpredictable occurrence of machine breakdowns and changes in job priorities increases the complexity of generating scheduling plans.

Traditional approaches to solving DJSP often rely on meta-heuristic algorithms or exact optimization algorithms. However, these methods can be computationally expensive, particularly for large and complex problem instances [4], [5]. Meta-heuristic approaches, including genetic algorithms (GA) and simulated annealing (SA), frequently exhibit a slow convergence speed and yield suboptimal solutions due

to their exhaustive exploration of high-dimensional solution space [6]. Although exact optimization techniques such as integer programming and branch-and-bound algorithms theoretically guarantee global optima, their exponential time complexity makes them computationally intractable for sizable problems in time-sensitive operational contexts requiring real-time decision-making capabilities [7].

To solve the aforementioned challenges, researchers have increasingly turned to machine learning (ML) and reinforcement learning (RL) approaches, which enable the development of adaptive scheduling strategies that can dynamically respond to environmental changes and system uncertainties [8], [9], [10]. Unlike traditional heuristics, ML-based methods can learn from historical data and simulation feedback, capturing latent patterns in complex scheduling space and improving generalization to unseen scenarios [11]. Among these techniques, graph-based neural models, particularly Graph Convolutional Networks (GCNs), have shown considerable promise due to the intrinsic structural nature of DJSP. However, GCN-based methods face several critical challenges when applied to DJSP. Generating high-quality schedules in real-time is challenging, especially in large-scale or dynamic environments. Most existing models also lack the flexibility to adapt to real-world constraints, such as machine breakdowns and rush orders [12]. This highlights the need for a unified framework that combines GCNs for structural learning with dynamic adaptation in complex 3C manufacturing settings.

Based on the above analysis, this work proposes a novel hybrid framework that synergistically combines Dual Graph (DG) convolutional networks with Dynamic Cooperative Hunting Optimizer (DCHO) named DG-DCHO to solve DJSP efficiently. It first employs two specialized GCN models, each of which is designed to handle distinct aspects of a scheduling process. 1) GCN generator produces a relatively high-quality initial population of candidate schedules, represented as sequences of operations assigned to jobs. It leverages the topological structure of dynamic job-shop environment by modeling jobs and machines as nodes in a bipartite graph, where edges capture processing dependencies and resource constraints. Through this graph-based representation, GCN can encode both spatial and temporal interactions among operations, machines, and job precedence relationships in a compact form. This enables the model to learn rich relational patterns and generate feasible and efficient action sequences for further optimization. 2) GCN evaluator estimates the quality of these schedules by predicting the makespan, *i.e.*, the total time required to complete all jobs in the schedule. It processes these sequences, which are first represented as scheduling graphs by modeling operations as nodes and precedence/resource constraints as directed edges, and provides an objective fitness evaluation by assessing the temporal efficiency of each candidate solution. In this context, the fitness value corresponds to the predicted makespan, indicating the schedule quality. The evaluator incorporates attention mechanisms and global pooling to capture higher-order dependencies in the scheduling graph, providing a more robust prediction of the makespan.

The integration of DG with the optimization process is further enhanced by DCHO combined with Cauchy mutation [13]. This hybridization allows the algorithm to explore vast solution space by balancing exploration and exploitation. Inspired by cooperative hunting strategies, DCHO refines solutions iteratively through cooperative behaviors, while the Cauchy mutation introduces limited randomness to maximize its chance to achieve the optimal schedules. By combining these advanced optimization techniques with GCNs, the proposed algorithm can generate high-quality schedules by dynamically adjusting population diversity, search trajectory, and structural guidance throughout the scheduling process. The workflow of DG-DCHO is shown in Fig. 1.

This work aims to make new contributions to the field of DJSP in highly stochastic manufacturing environments. The key contributions are summarized as follows.

1) A dual-GCN-driven learning framework is proposed to capture structural dependencies between jobs and machines, significantly reducing the reliance on resource-intensive fitness evaluations. It overcomes the initialization bias and computational inefficiencies of traditional optimization methods.
2) A cooperative metaheuristic algorithm named DCHO is developed, integrating dynamic step size adjustment and Cauchy mutation strategies to enhance both global exploration and local exploitation in the presence of dynamic disturbances.
3) DG-DCHO is developed as a unified hybrid approach, combining the strengths of graph-based representation learning and adaptive metaheuristic optimization to enable efficient and robust scheduling in large-scale and real-time DJSP scenarios.
4) Extensive experiments on various DJSP benchmark instances and 3C manufacturing simulations demonstrate that DG-DCHO outperforms several state-of-the-art algorithms in terms of makespan, convergence speed, and robustness to dynamic disturbances.

The remainder of this work is organized as follows. Section II reviews the related work. Based on the problem formulation in Section III and the proposed algorithmic framework in Section IV, Section V gives comprehensive experimental results to validate the framework. Finally, Section VI provides the conclusions.

II. Related Work

This section reviews existing approaches to DJSP from three perspectives: traditional optimization methods, machine learning-based scheduling techniques, and hybrid frameworks that combine metaheuristics with deep learning.

A. Traditional Optimization Methods

Traditional approaches to solving DJSP primarily rely on meta-heuristic and exact optimization algorithms. Among them, meta-heuristic methods such as GA, SA, and particle swarm optimization (PSO) are widely adopted due to their flexibility in navigating high-dimensional solution spaces. Xu et al. [14] apply GA with adaptive mutation rates to

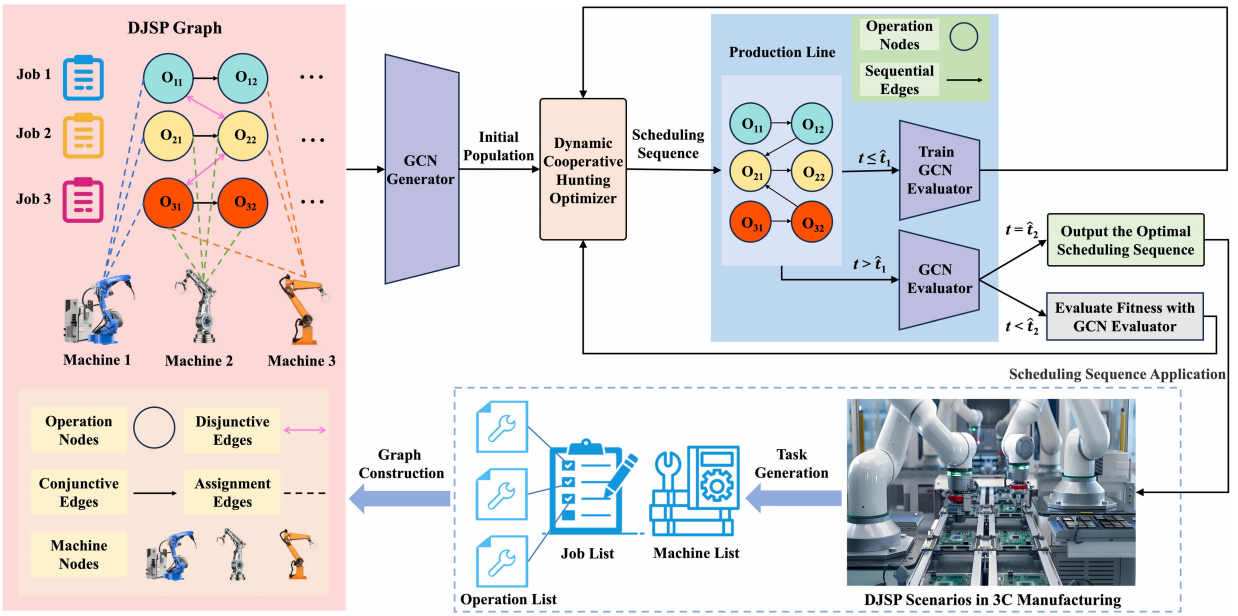


Fig. 1. The workflow of DG-DCHO. DJSP is modeled as a graph comprising operation and machine nodes, as well as different types of edges. Conjunctive edges enforce intra-job precedence, disjunctive edges represent resource conflicts, and assignment edges link operations to corresponding machines. This graph structure is fed into a GCN generator, which produces an initial population of feasible scheduling sequences, subsequently optimized by DCHO. During optimization, operation nodes and sequential edges form topological graphs to train a GCN evaluator, which rapidly predicts sequence fitness, bypassing time-consuming simulations. The final optimized schedule is executed to a 3C manufacturing production line.

minimize makespan in static job shop environments, achieving a 12% improvement over rule-based schedulers. However, the performance of GA degrades in dynamic settings due to its slow convergence and limited adaptability to stochastic events such as machine failures. To address these limitations, Feng et al. [15] propose a two-stage NSGA variant tailored for multi-objective DJSP. Their decomposition-based strategy reduces computational complexity by 30% for small-scale instances and demonstrates strong convergence performance in dynamic scenarios. Nevertheless, frequent solution repairs caused by redundant iterations significantly impact its effectiveness in large-scale problems involving over 50 machines.

Rule-based heuristics, including classical dispatching rules (*e.g.*, shortest processing time, earliest due date), are favored for their low computational cost and scalability in large-scale DJSP. Bi et al. [16] integrate an adaptive optimizer with a radial basis function network to derive near-optimal solutions, demonstrating superior performance in high-dimensional scheduling problems. In electronic manufacturing scenarios, applying the Modified Due Date (MDD) rule reduces job tardiness by 18%. However, its rigid priority structure leads to the 25% increase in machine idle time. To enhance adaptability, Zhao et al. [17] develop a fuzzy logic-based hybrid rule system capable of dynamically adjusting priorities based on real-time machine load. Despite improved flexibility, this approach remains constrained by the manual design of rules, limiting its scalability and generalization to unforeseen disruptions. Exact optimization methods, such as Mixed-Integer Linear Programming (MILP) and constraint programming, offer theoretical guarantees of optimality by exhaustively exploring the solution space. Wang et al. [18] model DJSP as an MILP problem with temporal constraints and solve it using a branch-and-cut algorithm. While their approach yields near-optimal schedules for small-scale instances, it suffers from prohibitive computational overhead when applied to large-scale job shop scenarios. To improve computational efficiency, Zhang et al. [19] introduce a hybrid branch-and-bound algorithm augmented with local search, reducing computational cost by 40% through problem decomposition. Despite these advancements, exact methods remain impractical for real-time scheduling in dynamic environments such as 3C manufacturing, where job priorities and machine availability frequently change. Moreover, exact algorithms lack inherent adaptability to uncertainties such as urgent job insertions or resource reconfigurations, limiting their applicability in highly dynamic production settings.

B. Deep Learning-Based Scheduling Methods

The integration of deep learning into DJSP has transformed adaptive scheduling by enabling data-driven decision. RL methods, such as Deep Q-Networks (DQN) and policy gradient algorithms, learn optimal scheduling strategies through trial-and-error interactions with simulated environments. Luo et al. [20] develop a DQN-based scheduler for partial-no-wait DJSP, achieving a 22% reduction in makespan compared to GA in Printed Circuit Board (PCB) manufacturing scenarios. However, their model requires extensive training iterations to reach convergence, limiting its suitability for time-sensitive deployment. To address this limitation, Pan et al. [21] propose a meta-reinforcement learning framework that enables policy transfer across similar job-shop environments, reducing training time by 60%. Despite these improvements, RL approaches still face challenges with sparse reward signals in

large-scale settings and often converge to suboptimal policies in systems with more than 100 machines.

GNNs have emerged as a powerful paradigm for capturing relational dependencies in DJSP. By modeling jobs and machines as nodes in a bipartite graph, GNNs can effectively extract features and capture spatiotemporal interactions essential for scheduling in complex environments. Liu et al. [22] propose a GNN-RL hybrid model that embeds job precedence constraints and machine capacities into graph representations. Their approach achieves a 15% reduction in makespan compared to traditional heuristics in dynamic automotive assembly lines. However, the model relies on large volumes of labeled training data, which are often expensive and difficult to obtain in real-world industrial systems. Building on this, Chen et al. [23] incorporate multi-head attention mechanisms into the GNN framework to dynamically prioritize bottleneck machines and high-priority jobs. While this improves adaptability, the graph construction process introduces significant computational overhead, especially in large-scale settings. Their results indicate that preprocessing time increases sharply when the number of machines exceeds 200. To alleviate the burden of explicit graph modeling, recent studies have explored transformer-based sequence scheduling architectures. For instance, Zhao et al. [24] introduce a heterogeneous graph transformer to model the temporal dependencies among operations, achieving state-of-the-art performance in semiconductor scheduling tasks. However, due to the quadratic complexity of the transformer on sequence length, its efficiency degrades significantly when handling jobs with more than 50 operations, limiting its scalability in practice.

C. Hybrid Optimization Frameworks

Hybrid frameworks that integrate metaheuristics with deep learning seek to combine the global search capabilities of optimization algorithms with the pattern recognition strengths of neural networks. Zhang et al. [25] explore this direction by combining genetic programming with instance-rotation surrogates, enabling the dynamic evolution of scheduling rules based on real-time performance feedback. Their method achieves a 35% reduction in makespan variability under fluctuating demand conditions but suffers from high-dimensional action spaces in multi-objective scenarios. To address this limitation, Lei et al. [26] propose a hierarchical RL architecture for flexible job-shop scheduling, decomposing the problem into task allocation and machine assignment layers. This structural decoupling reduces computational delays by 24%. However, the absence of efficient fitness evaluation mechanisms leads to premature convergence in complex scheduling environments.

Graph-enhanced metaheuristics have also emerged as a promising avenue. Inspired by the success of dual-GCN architectures in hyperspectral image classification [27], Liu et al. [28] embed graph convolutional layers into a teaching-learning-based optimizer, allowing the algorithm to exploit the topological structure of job-machine relationships. This approach achieves a 20% reduction in makespan in static job shop scenarios. Nonetheless, its performance deteriorates in dynamic environments due to the use of fixed graph representations. To enhance adaptability, Liu et al. [29] introduce a dynamic graph update mechanism that reconfigures node connections in response to real-time machine status, improving adaptability in electronics manufacturing. Despite these advantages, the iterative graph reconstruction process introduces considerable computational overhead, limiting the method's practicality in real-time applications. At the frontier of this field, meta-heuristic and deep learning co-evolution frameworks have shown promising results. Mou et al. [30] design a hybrid algorithm combing PSO and a GNN-based surrogated model, where the surrogated model predicts the makespan of candidate solutions to accelerate fitness evaluation, achieving a 70% reduction in evaluation time. However, the model exhibits reduced accuracy in highly dynamic environments, leading to suboptimal schedule selection. To address this issue, Xiao et al. [31] propose a Gaussian Mixture Model (GMM)-based surrogate that quantifies prediction uncertainty, enabling more robust filtering of candidate solutions. Although their method achieves higher prediction accuracy in stable environments, its performance declines to 65% under frequent machine breakdowns in dynamic environments.

In summary, although existing hybrid frameworks show promise in addressing the challenges of dynamic job shop scheduling, they are still limited by several critical issues: suboptimal integration of graph-based initialization with adaptive metaheuristics, high computational overhead from repeated fitness evaluations, and insufficient robustness in highly stochastic manufacturing environments. These limitations highlight the need for a unified framework that effectively bridges graph-driven schedule generation with adaptive optimization while maintaining computational efficiency and resilience to real-world uncertainties.

TABLE I
MAIN PARAMETERS

Notation	Definition
m	Number of machines
n	Number of jobs
k	Number of operations in J_i
J	Set of jobs, $J = \{J_1, J_2, \ldots, J_n\}$
M	Set of machines, $M = \{M_1, M_2, \ldots, M_m\}$
R	Assignment of operations to machines
T	Time required to process each operation
O_{il}	l-th operation of J_i
R_{il}	Machine assignment for O_{il}
T_{il}	Processing time of O_{il}
S_{il}	Start time of O_{il}
C_{il}	Completion time of O_{il}
$\hat{C}$	Makespan, *i.e.*, maximum job completion time
T_w	Waiting time due to scheduling conflicts on shared machines
x_{il}^j	Binary variable: 1 if O_{il} assigned to M_j, 0 otherwise
$\theta(M_j, O_{il})$	Indicator: 1 if O_{il} can be assigned to M_j, 0 otherwise
$C_{i,l-1}$	Completion time of previous operation $O_{i,l-1}$ in J_i
C_{ab}	Completion time of another job's operation on same machine
$A_j(e)$	Availability of M_j at time e
D_i	Deadline for completion of J_i

III. PROBLEM FORMULATION

DJSP in 3C smart manufacturing is a resource-constrained, time-varying optimization problem that allocates jobs to machines while minimizing the total time to complete all jobs (*i.e.*, the makespan). DJSP is characterized by the dynamic

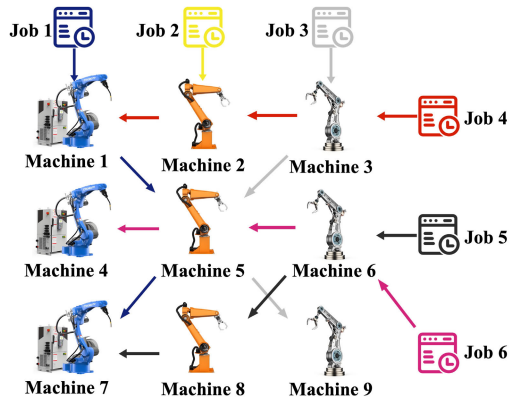


Fig. 2. DJSP scenarios in 3C manufacturing.

and stochastic nature of the environment, including variability in production requirements, machine availability, and potential disruptions such as machine breakdowns and urgent job insertions. The main notations are summarized in Table I.

A. Problem Definition

Fig. 2 illustrates a DJSP instance with six jobs and nine machines, where each job follows a predefined sequence of operations across multiple machines. The arrows of the same color indicate the unique flow of a specific job, representing the predefined sequence in which operations must be executed. Each job consists of a set of operations, and each operation is assigned to a specific machine with a given processing time.

Let $M=\{M_1, M_2, \ldots, M_j, \ldots, M_m\}$ denote a set of machines, where m is the number of machines, and let $J=\{J_1, J_2, \ldots, J_i, \ldots, J_n\}$ denote a set of jobs, where n is the number of jobs. Each J_i consists of a sequence of operations $O_i=\{O_{i1}, O_{i2}, \ldots, O_{il}, \ldots, O_{ik}\}$, where k is the number of operations per job. The machine matrix $R=\{R_{il} \mid R_{il} \in \{0, M_1, M_2, \ldots, M_j, \ldots, M_m\}\}$ represents the assignment of operations to machines, where

$$R_{il}=\begin{cases} M_j & \text{if } O_{il} \text{is processed by } M_j, \\ 0 & \text{if } O_{il} \text{is not processed by any machine.} \end{cases} \tag{1}$$

The processing time matrix $T=\{T_{il} \mid T_{il} \geq 0\}$ specifies the time required to process each O_{il}. If O_{il} is not assigned to any machine, $R_{il}=0$, and its processing time T_{il} is also zero, indicating the operation does not contribute to the scheduling process and does not require execution.

The objective is to minimize the makespan $\hat{C}$ (the total time required to complete all jobs). The completion time of each O_{il} is denoted by C_{il}. $\hat{C}$ can be obtained as:

$$\hat{C}=\max\left(C_{ik}\right), \quad i \in J,\ k \in O_i, \tag{2}$$

where C_{ik} is the completion time of the last operation of J_i, and k represents the index of the last operation of J_i. The completion time of each O_{il} is determined by:

$$C_{il}=\begin{cases} 0, & l=0 \\ \max(C_{i,l-1}, C_{ab})+T_{il}+T_{\mathrm{w}}, & l \geq 1 \end{cases} \tag{3}$$

where $C_{i,l-1}$ is the completion time of the previous $O_{i,l-1}$ of J_i, C_{ab} is the completion time of the previous operation O_{ab} of another J_a on the same R_{il}, T_{w} is the waiting time for the R_{il} when it is idle but cannot process due to a scheduling conflict, *i.e.*, other jobs using the machine at the same time.

B. Constraints of Job Shop Scheduling in 3C Manufacturing

3C-based DJSP operates under several constraints that define its scheduling feasibility and optimization complexity. These constraints are formulated as follows.

1) Machine Capacity: Each machine can process only one operation at a time, reflecting the resource limitations in 3C manufacturing systems, where precision and resource allocation are critical to maintaining production efficiency. In electronics manufacturing, a surface-mount technology (SMT) machine can handle only one PCB assembly at a time. This constraint ensures that the simultaneous execution of multiple jobs on the same machine is prevented, *i.e.*,

$$\sum_{i \in J} \sum_{l \in O_i} \theta(M_j, O_{il}) \cdot x_{il}^j \leq 1, \quad \forall j \in M, \tag{4}$$

where $\theta(M_j, O_{il})$ is an indicator function that takes the value one if O_{il} of J_i is assigned to M_j, and x_{il}^j is a binary variable indicating whether O_{il} of J_i is executed on M_j.

2) Job Operation Execution: Each job operation can only be executed by one machine at a time. It prevents a job operation from being executed simultaneously on multiple machines, which violates the logical sequence of operations in a job, *i.e.*,

$$\sum_{j \in M} x_{il}^j=1, \quad \forall i \in J,\ \forall l \in O_i. \tag{5}$$

3) Job Precedence Constraints: An operation cannot start until its preceding operation is completed. The constraint ensures that the operations of a job are executed in a fixed order, representing that O_{il} cannot begin until $O_{i,l-1}$ is finished. This is essential for maintaining the workflow of multi-stage processes, particularly in 3C manufacturing environments such as electronics assembly, where each step must follow a defined sequence to ensure product integrity, *i.e.*,

$$C_{il} \geq C_{i,l-1}+T_{il}, \quad \forall l \in \{2, \ldots, k\}, \quad i \in J. \tag{6}$$

4) Operation Non-Suspension: An operation cannot be paused or terminated once it has started. This assumption is necessary to maintain the integrity of the job schedule, ensuring that no operation can be interrupted. This is typical in manufacturing environments where resources are dedicated to specific tasks for the duration of their execution. This constraint helps reduce the risk of errors and inefficiencies caused by interruptions or restarts in operations, *i.e.*,

$$C_{il}=S_{il}+T_{il}, \quad \forall i \in J,\ \forall l \in O_i, \tag{7}$$

where S_{il} is the start time of O_{il}.

5) Sequential Job Operations: All operations of the same job must follow a specific and ordered sequence. This sequential order ensures that each job undergoes all required processes and quality checks, essential for high-precision manufacturing tasks, *i.e.*,

$$C_{il} \leq C_{i,l+1}, \quad \forall l \in \{1, \ldots, k-1\}, \quad i \in J. \tag{8}$$

6) Negligible Transportation and Setup Times: The transportation and setup times among operations are assumed to be zero. This simplifies the problem by neglecting the time required for moving jobs between machines or setting up machines for new operations, thus focusing solely on the processing time for each operation. In real-world 3C manufacturing environments, such times are typically minimized through efficient layouts and automation systems. However, this assumption enables a streamlined modeling approach focusing on the most critical factors.

7) Known Machine and Processing Time Matrices: The machine matrix R and processing time matrix T are known in advance. This eliminates uncertainty in machine assignment and processing durations, formulating the scheduling problem with deterministic input. This is particularly beneficial when real-time data on machine availability and operation times can be dynamically integrated.

8) Machine Breakdowns: Machine breakdowns may occur during production, which leads to machine downtime. The availability of each machine at any given time can be stochastic, depending on whether the machine is operational or not. Let $A_j(e)$ denote the availability of M_j at time e, *i.e.*,

$$A_j(e)=\begin{cases}1 & \text{if } M_j\text{is available at time } e,\\ 0 & \text{if } M_j\text{is unavailable at time } e.\end{cases} \tag{9}$$

This ensures that the scheduler accounts for disruptions, such as machine failures or unexpected maintenance, enabling timely rescheduling and enhancing system reliability by minimizing downtime and maintaining production continuity.

9) Priority Jobs: Certain jobs have higher priority and must be scheduled before lower-priority jobs. This is important in real-world manufacturing systems where some jobs must be completed before others, *e.g.*, rush orders or time-sensitive products. This prioritizes critical jobs, helping to reduce delays and maintain overall production timelines, *i.e.*,

$$C_{ik} \le C_{dk}, \quad \text{if } J_i\text{has higher priority than } J_d. \tag{10}$$

10) Machine-Specific Operations: Certain jobs or operations can only be processed by specific machines. This constraint ensures that certain operations can only be assigned to specific machines due to technical requirements or resource limitations. For each O_{il} of J_i, *i.e.*,

$$x_{il}^j=1 \quad \text{if } O_{il}\text{can only be processed by } M_j. \tag{11}$$

11) Job Deadlines: Each job must be completed by a specified deadline D_i. This constraint ensures that the final operation of each job is completed within the deadline, ensuring just-in-time (JIT) production and supply chain synchronization in electronics manufacturing. It also ensures that critical deliveries and operations are completed on time, minimizing customer impact and reducing inventory costs, *i.e.*,

$$C_{ik} \le D_i, \quad \forall i \in J,\, k \in O_i, \tag{12}$$

where D_i is the deadline for J_i.

TABLE II
DECISION VARIABLES

Notation	Definition
T_{il}	Processing time of O_{il}
S_{il}	Start time of O_{il}
C_{il}	Completion time of O_{il}
T_w	Waiting time due to scheduling conflicts on shared machines
x_{il}^j	Binary variable: 1 if O_{il} assigned to M_j, 0 otherwise
$\theta(M_j, O_{il})$	Indicator: 1 if O_{il} can be assigned to M_j, 0 otherwise
$C_{i,l-1}$	Completion time of previous operation $O_{i,l-1}$ in J_i
C_{ab}	Completion time of another job's operation on same machine
$A_j(e)$	Availability of M_j at time e
D_i	Deadline for completion of J_i

12) Dynamic Job Arrival: Jobs may arrive at different times, introducing uncertainty into scheduling. This reflects real-time operational constraints commonly seen in 3C manufacturing, where new orders can be introduced during ongoing production. An effective scheduling system must adapt to these dynamic arrivals without disrupting existing operations.

13) Resource Utilization: Machines must be optimally utilized throughout the scheduling period to avoid underutilization or overloading. This constraint improves overall system efficiency by ensuring that each machine operates near its optimal capacity while minimizing idle time, which is especially important in large-scale 3C manufacturing focused on maximizing throughput.

C. Optimization Problem

In conclusion, Table II shows the decision variables of DJSP, and our goal is to optimize $\hat{C}$, *i.e.*,

$$\underset{\chi}{\textbf{Min}}\ \hat{C},$$

where χ represents a set of decision variables and it is subject to (2)-(12).

IV. Proposed Dynamic Job Shop Scheduling Framework

This section introduces the solution algorithm for solving DJSP using a hybrid approach combining DCHO with DG. It aims to find the optimal scheduling solution rapidly while reducing the total makespan.

A. Framework Overview

DG-DCHO is an iterative optimization algorithm that combines the strengths of DG for schedule generation and fitness evaluation with the capabilities of DCHO for refinement and optimization. The process begins with initialization, where the workshop environment is modeled by abstracting job-shop scheduling problems and their constraints into graph matrices and then pre-processing the data for training the GCN generator. Once the activation condition is met, the GCN generator processes the environment's graph structure to generate an initial population of feasible scheduling solutions. These schedules are then optimized using DCHO, aiming to minimize the makespan. During the early optimization phase, when the iteration count is less than $\hat{t}_1$, the fitness of all

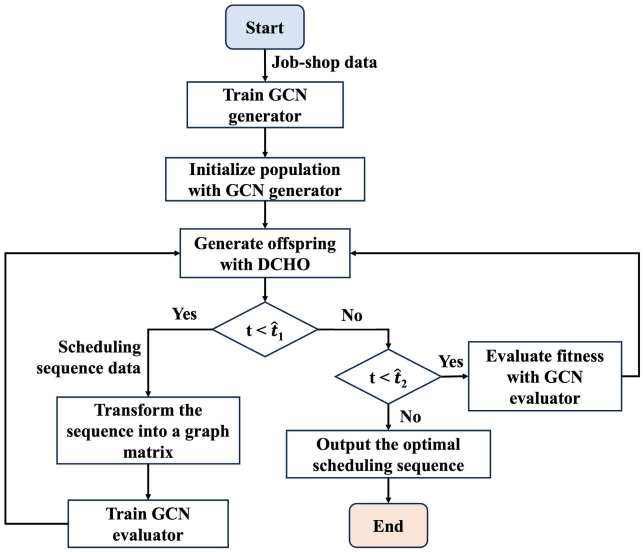


Fig. 3. Framework of DG-DCHO.

generated schedules is computed through conventional fitness evaluation, and the scheduling sequences are transformed into graph matrices. These sequence graphs and their corresponding fitness values are used to train the GCN evaluator. As the iteration count surpasses $\hat{t}_1$ but remains below the termination threshold $\hat{t}_2$, the GCN evaluator assists fitness evaluation by predicting makespan values, significantly accelerating optimization while reducing computational cost. The iterative refinement continues until it reaches the maximum iteration count $\hat{t}_2$. Finally, the best scheduling solution that minimizes the makespan is selected as the final optimized schedule. Fig. 3 shows the overall framework of DG-DCHO, which integrates DG for both schedule generation and accelerated fitness evaluation while utilizing DCHO for adaptive and efficient optimization. By leveraging the GCN generator to construct high-quality initial schedules and the GCN evaluator to predict makespan values with minimal computational overhead, it significantly reduces the cost of fitness evaluation.

B. Graph Construction

To effectively leverage GCNs for scheduling optimization, both the dynamic job shop environment and candidate solutions are transformed into structured graphs. This section details the construction process for the distinct input graphs required by the GCN generator and the GCN evaluator.

The graph constructed for the GCN generator captures the structural and dynamic features of the DJSP instance. Formally, the input graph is denoted as $G = (V, E)$, where the node set V consists of operation nodes O_{il} and machine nodes M_j [32]. Each operation node represents a specific operation and carries attributes such as the processing time T_{il}, arrival time, and other dynamic status indicators. Each machine node includes information such as the current availability state $A_j(e)$ or breakdown-related features, capturing real-time constraints in the production environment. The edge set E encodes dependencies and constraints. Directed conjunctive edges connect consecutive operations within the same job, enforcing job precedence and the correct execution order. Assignment edges link operation nodes to machine nodes that are capable of processing them, reflecting machine-specific capabilities and ensuring each operation is assigned to exactly one machine. In addition, disjunctive edges are introduced between operations that may contend for the same machine, modeling the resource conflict constraint that no two operations can occupy the same machine simultaneously. Through this unified representation, the constructed graph integrates job sequencing, machine assignment, and resource contention, while naturally supporting dynamic conditions such as breakdowns, job arrivals, and job priorities. By processing this graph, the GCN generator learns to produce initial scheduling sequences that satisfy both static and dynamic feasibility requirements.

The graph constructed for the GCN evaluator focuses on representing individual candidate schedules generated during optimization. Formally, each candidate schedule is transformed into a graph $G' = (V', E')$, where the node set V' consists solely of operation nodes O_{il}, carrying features such as processing time T_{il} and operation identifiers. The edge set E' comprises directed sequential edges that represent the actual execution flow of operations in the candidate schedule. The resulting graph G' forms a directed acyclic graph (DAG) that captures the topological order of operations and inherently respects the operation non-suspension constraint.

Through these graph representations, the proposed method comprehensively models scheduling constraints, including machine capacity, job precedence, operation non-suspension, machine-specific assignments, breakdown tolerance, job prioritization, and dynamic job arrivals. This graph-based formulation provides a unified and scalable foundation for intelligent scheduling under complex 3C manufacturing conditions.

C. GCN Generator and GCN Evaluator

The constructed graph is then processed by the GCN generator through multiple graph convolutional layers, which iteratively update node representations based on local connectivity. The operation of each graph convolutional layer is formulated as:

$$h_v^{(q+1)} = \sigma\left(\sum_{u \in N(v)} W^{(q)} h_u^{(q)} + b^{(q)}\right), \tag{13}$$

where $h_v^{(q)}$ denotes the feature vector of node v at layer q, $N(v)$ represents the neighboring nodes of v, $W^{(q)}$ and $b^{(q)}$ are the learnable weight matrix and bias for layer q, and σ is the ReLU activation function. After passing through the convolutional layers, a Fully Connected (FC) layer generates a sequence of machine assignments, representing the initial schedule, *i.e.*,

$$S = \mathrm{FC}(h_v^{(Q)}), \tag{14}$$

where S denotes the output sequence of job-machine assignments. The GCN generator guarantees the feasibility of initial schedules through its end-to-end learning process, ensuring that all solutions originate within the valid search space. By processing a graph that encodes precedence and resource

constraints, it learns to produce inherently feasible schedules. The training objective, aimed at minimizing makespan, implicitly penalizes infeasible sequences, driving the model to explore only solutions that respect both job precedence and machine assignment rules. This feasibility is preserved during optimization, where any candidate generated in the iterative search that would cause a machine conflict is immediately discarded.

The GCN evaluator predicts the makespan of candidate schedules based on their graph representations, enabling efficient fitness evaluation without full process simulation. By leveraging spatial and relational information from job-machine sequences, it approximates schedule quality with significantly reduced computational cost.

The model begins with a Graph Attention Layer (GAT), which enhances the network's capacity to focus on relevant operational relationships and selectively aggregate features for more accurate makespan prediction [33]. This operation is formulated as:

$$h_v = \mathbf{GAT}(\{h_u, e_{uv}\}), \tag{15}$$

where h_u represents the features of neighboring nodes and e_{uv} represents the edge between nodes u and v. The extracted features are then aggregated through global pooling, where the entire graph is transformed into a compact representation using global mean pooling [34]. This compact representation is subsequently fed into a Multi-Layer Perceptron (MLP) layer, which is a neural network composed of a sequence of fully connected layers. The MLP processes the aggregated features and ultimately predicts the makespan value, *i.e.*,

$$\hat{C} = \mathbf{MLP}(h_v^{(Q)}). \tag{16}$$

The GCN evaluator is trained using the mean squared error (MSE) loss function, minimizing the difference between predicted and actual makespans [35], *i.e.*,

$$\mathcal{L} = \frac{1}{N}\sum_{y=1}^{N}\left(C_{\mathrm{p}}(S_y^{(t)}) - C_{\mathrm{u}}(S_y^{(t)})\right)^2, \tag{17}$$

where N is the number of samples, $S_y^{(t)}$ represents the y-th scheduling solution at iteration t, $C_{\mathrm{p}}(S_y^{(t)})$ represents the predicted makespan of schedule $S_y^{(t)}$, and $C_{\mathrm{u}}(S_y^{(t)})$ denotes its true makespan.

The GCN evaluator employs a dual-safeguard mechanism to ensure both predictive accuracy and computational efficiency in dynamic environments, thereby avoiding the need for continuous online retraining while still enabling essential updates. During the iterative process, it is not exclusively relied on to identify the best individual. Instead, it serves as a rapid prescreening tool for the entire population, selecting a small subset of the most promising candidates (*e.g.*, $k = 10$) for further evaluation. Subsequently, we calculate the true fitness of only these k individuals through conventional, resource-intensive simulation. The final selection of the best individual in each generation is made using these precise, true fitness values. This hybrid strategy greatly reduces computational overhead, as most of the population is evaluated by the fast GCN model, while the accuracy of elite solution identification is ensured by validating only the top candidates. The second layer of this safeguard is a monitoring and triggered-retraining mechanism. As we compute the true fitness for the top-k individuals which are denoted as the set $P_{\text{top-k}}$, we simultaneously compare these values against the predictions made by the GCN evaluator. The discrepancy is quantified by the monitoring loss, $\mathcal{L}'$, defined as:

$$\mathcal{L}' = \frac{1}{k}\sum_{S_y \in P_{\text{top-k}}}\left(f_{\mathrm{p}}(S_y) - f_{\mathrm{u}}(S_y)\right)^2, \tag{18}$$

where $f_{\mathrm{p}}(S_y)$ is the predicted fitness and $f_{\mathrm{u}}(S_y)$ is the true fitness for a solution S_y. If this monitoring loss exceeds a predefined threshold, it signals a significant concept drift, indicating that the evaluator's model no longer aligns well with the current state of the dynamic environment. Upon such a trigger, the optimization process is briefly paused to fine-tune the GCN evaluator for a few epochs using these k newly validated data points. To empirically demonstrate the stability of our evaluator, we recorded the frequency of this fine-tuning process. As shown in Table III, it is triggered in only a small fraction of iterations. This low trigger rate confirms that the GCN evaluator is reliable in most cases, while the safeguard ensures that its accuracy is maintained by correcting errors as they arise, affirming the overall robustness of our approach.

TABLE III
FREQUENCY OF TRIGGERED RETRAINING UNDER DISTURBANCES

Instance	Iterations	Retraining Triggers	Trigger Rate (%)
15×15 DJSP	450	26	5.78
20×20 DJSP	450	31	6.89
50×10 DJSP	450	35	7.78
Average	**450**	**30.7**	**6.82**

D. Dynamic Cooperative Hunting Optimizer

The DCHO serves as the core search engine by leveraging the inherent strengths of metaheuristics to address complex problems such as DJSP. Its population-based design provides a flexible framework for navigating vast solution spaces while mitigating premature convergence. DCHO's cooperative hunting strategy offers an intuitive mechanism for dynamically balancing global exploration and local exploitation, inherently enhancing robustness to environmental changes. This resilience is further strengthened by the integration of Cauchy mutation, whose heavy-tailed properties enable large exploratory jumps to escape local optima caused by sudden disturbances such as machine breakdowns. By maintaining this adaptive balance, DCHO efficiently discovers high-quality schedules within acceptable polynomial time complexity, making it suitable for large-scale industrial applications where exact methods are computationally prohibitive. Moreover, its update rules are grounded in clearly defined behavioral components, providing strong explainability that aids parameter tuning and fosters operational trust, which distinguishes it from more opaque, non-transparent approaches.

As a metaheuristic algorithm inspired by cooperative hunting strategies in nature, DCHO iteratively improves scheduling

solutions by balancing exploration and exploitation. The optimization process is divided into two phases: an exploration phase based on hunting coordination and an exploitation phase simulating predator escape. During exploration, the position of each individual is updated by referencing the current best solution. The exploration mechanism in DCHO draws inspiration from coordinated hunting behaviors observed in animal groups. A core principle involves individuals adjusting their search trajectory based on the location of the current best individual. The representation of such movement towards the current best target is denoted as:

$$p'_{f,g}=p_{f,g}+r\cdot(p_{b,g}-z\cdot p_{f,g}), \tag{19}$$

where $p'_{f,g}$ is the potential new g-th dimension of individual f, calculated based on its current position $p_{f,g}$. It moves towards the best solution's position $p_{b,g}$, modulated by a random factor $r\in[0,1]$ and a stochastic element z. This promotes convergence towards promising regions while preserving sufficient stochasticity for exploration.

In the exploitation phase, individuals refine their search by adaptively adjusting their positions based on perturbations around a promising target. The exploitation mechanism is inspired by behaviors in which individuals make fine-tuned adjustments in a localized area. This often involves searching within progressively smaller and more focused regions around the current position. The scope of the local search can be dynamically reduced over iterations t. Representing the lower L_g and upper U_g bounds for parameter g, the local search bounds (L'_g, U'_g) is denoted as:

$$L'_g=L_g/t,\quad U'_g=U_g/t. \tag{20}$$

Within these progressively shrinking bounds, a new position can be generated by adding a random perturbation to the current position $p_{f,g}$:

$$p''_{f,g}=p_{f,g}+(1-2r)\cdot(L'_g+r\cdot(U'_g-L'_g)), \tag{21}$$

where $p''_{f,g}$ is the updated position. $(1-2r)$ provides a random direction $[-1,1]$, and the subsequent term generates a random point within the shrinking local bounds $[L'_g, U'_g]$. This facilitates precise local search.

Once the GCN generator generates the initial scheduling solutions, DCHO is applied to optimize them. DCHO iterates over a multitude of scheduling solutions S, where S implicitly defines C_{il} for all O_{il} based on T_{il} and the sequence order. During the iterative process, any S with $C_{ik}>D_i$, $\theta(M_j, O_{il})=0$, or causing machine conflicts will be discarded. Unlike conventional optimization approaches, DCHO dynamically adjusts its search behavior using an adaptive step size mechanism and Cauchy mutation, allowing it to navigate complex scheduling landscapes more effectively. The update rule of the schedule adjustment process is obtained as:

$$S_y^{(t+1)}=S_y^{(t)}+\alpha\cdot(S_b-S_y^{(t)})+\beta\cdot\mathcal{N}(0,\Gamma), \tag{22}$$

where S_b denotes the best schedule solution found so far, α and β are coefficients that regulate the balance between exploration and exploitation, and $\mathcal{N}(0,\Gamma)$ is a Gaussian noise term introduced to maintain population diversity and prevent premature convergence. The coefficients α and β are crucial for balancing exploitation and exploration. Instead of using fixed values, we employ a dynamic adaptive strategy to enhance the algorithm's performance across different optimization stages. The exploitation factor α is designed to increase non-linearly, strengthening the pull towards the best-known solution as the search progresses. This is governed by:

$$\alpha(t)=\alpha_{\min}+(\alpha_{\max}-\alpha_{\min})\cdot\left(\frac{t}{T}\right)^2, \tag{23}$$

where t is the current iteration, T is the maximum number of iterations, and $\alpha_{\min}$ and $\alpha_{\max}$ are set within the ranges of [0.1, 0.3] and [0.6, 0.9], respectively. Concurrently, the exploration factor β, which controls the magnitude of random perturbations, is used to decrease non-linearly as follows:

$$\beta(t)=\beta_{\max}\cdot\left(1-\frac{t}{T}\right)^2, \tag{24}$$

where its initial value $\beta_{\max}$ set in the range of [0.1, 0.2]. This co-adaptive mechanism enables DCHO to intelligently transition its focus from broad exploration in the early phases to fine-grained exploitation in the later phases, thereby improving overall convergence quality. DCHO employs a dynamic step size adaptation mechanism to enhance search efficiency further, ensuring that the search process remains aggressive in early iterations but gradually stabilizes as the solution converges. The step size at iteration t is obtained as:

$$\delta^{(t)}=\Delta\cdot\left(1-\frac{t}{T}\right)+\delta\cdot\frac{t}{T}, \tag{25}$$

where Δ and δ represent the initial and final step sizes. This dynamic adjustment allows the algorithm to explore broadly in the early phase while focusing on fine-tuning solutions in later stages. Additionally, Cauchy mutation is introduced to further enhance diversity in candidate schedules by perturbing solutions in a heavy-tailed manner, increasing the probability of escaping local optima. It is applied as:

$$S_y^{(t+1)}=S_y^{(t)}+\gamma\cdot\mathcal{C}(0,1), \tag{26}$$

where γ is a mutation scaling factor, and $\mathcal{C}(0,1)$ represents a random variable drawn from a Cauchy distribution with zero mean and unit scale. The Cauchy mutation scale factor γ determines the average magnitude of the perturbation, which is critical for escaping local optima due to the distribution's heavy tails. The selection of γ requires balancing the risk of a large value disrupting solution structures against a small value failing to provide enough force to exit a local minimum. Through extensive experimentation, we have found that setting γ to a fixed value within the range of [0.05, 0.2] provides a robust balance between exploration and convergence stability. For our implementation, a default value of 0.1 is used, proving effective in providing the necessary momentum for escaping local traps without compromising the algorithm's ability to converge efficiently.

To empirically justify parameter configurations, we conducted a comprehensive sensitivity analysis on a representative 20×20 DJSP instance. The results, presented in Table IV, compare the performance of DG-DCHO configuration against a range of variations in its key hyperparameters, including the

TABLE IV
PARAMETER SENSITIVITY ANALYSIS ON A 20×20 DJSP INSTANCE

Parameter Group	Value	Avg. Makespan	Std. Dev.
No Adaptation	$\alpha = 0.5, \beta = 0.1$	1291.8	20.3
	$\alpha = 0.2, \beta = 0.2$	1305.3	22.1
	$\alpha = 0.8, \beta = 0.6$	1312.6	26.5
Adaptive Strategy		**1255.4**	**15.2**
Cauchy Scale (γ)	0.01	1280.7	18.5
	0.1	**1255.4**	**15.2**
	0.5	1315.2	25.8
	1.0	1342.1	31.4
Step Size (Δ, δ)	$\Delta = 1, \delta = 0.01$	1277.3	16.8
	Δ=5, δ=0.05	**1255.4**	**15.2**
	$\Delta = 10, \delta = 0.5$	1291.5	24.1
	$\Delta = 10, \delta = 1.0$	1308.9	28.6
Evaluator LR	0.0005	1289.6	19.1
	0.001	1275.9	17.5
	0.01	**1255.4**	**15.2**
	0.1	1283.4	22.3

Cauchy mutation scale γ and the dynamic step size boundaries (Δ, δ).

It is shown in Table IV that the configuration for DG-DCHO consistently yields the best makespan with the lowest standard deviation, indicating superior performance and stability. Disabling the adaptive strategy for α and β leads to a notable performance degradation, confirming the value of dynamic control. Furthermore, the Cauchy scale γ exhibits a clear optimal range, as insufficient exploratory power from a low value and destabilization from a high value both worsen solution quality. The dynamic step size boundaries (Δ, δ) are similarly crucial for balancing search intensity, with conservative strategies leading to premature convergence and aggressive ones hindering effective fine-tuning. These findings confirm that the parameters and adaptive mechanisms are integral to the superior and robust performance of the DG-DCHO framework. By integrating dynamic step size adaptation and Cauchy mutation, DCHO dynamically refines the schedules generated by the GCN generator, incorporating feedback from the GCN evaluator to iteratively minimize the makespan.

V. PERFORMANCE EVALUATION

This section presents the experimental results of DG-DCHO. We conduct both comparison experiments and ablation studies to demonstrate the superiority and effectiveness of DG-DCHO in terms of makespan and convergence speed.

A. Comparison Experiments

To validate the performance of DG-DCHO, we compare it with several advanced scheduling algorithms, including Asymptotically MSE-Optimal Gaussian-GMM Estimator (AMGG) [36], Surrogate-Assisted Autoencoder-Embedded Evolutionary Optimization Algorithm (SAEO) [37], Surrogate-Assisted Hybrid Evolutionary Algorithm with Local Estimation of Distribution (SHEALED) [38], Coati Optimization Algorithm (COA) [39], Multi-factorial Evolutionary Algorithm (MFEA) [40], Starfish Optimization Algorithm (SFOA) [41], and Alpha Evolution (AE) [42].

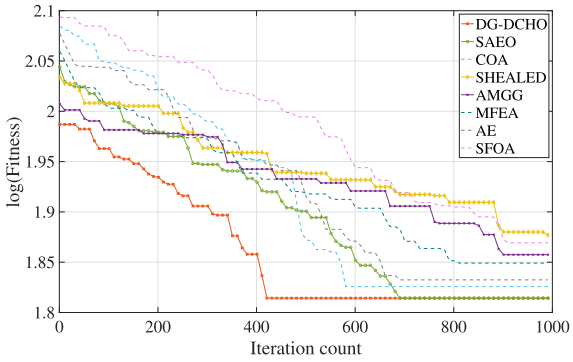


Fig. 4. Comparison of convergence speed for eight algorithms.

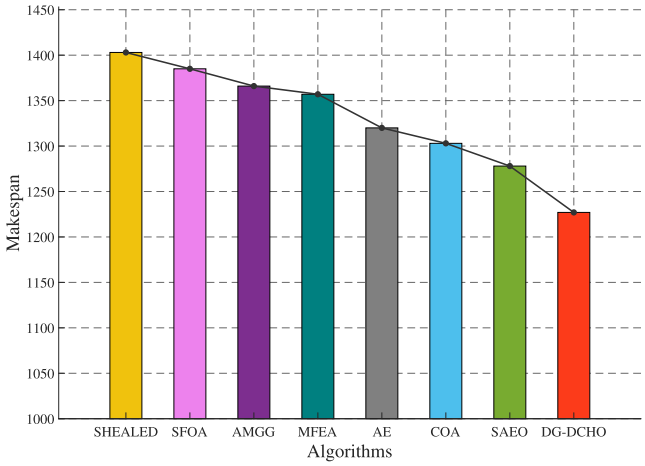


Fig. 5. Comparison of makespan for eight algorithms.

These methods are selected due to their representative strengths in model-based optimization, surrogate-assisted scheduling, and bio-inspired search strategies, which align with the challenges posed by large-scale dynamic job shop problems. All algorithms are evaluated under the same DJSP environments to assess solution quality and computational efficiency.

Fig. 4 presents the iteration curves of all eight algorithms, where each algorithm's fitness value is evaluated over 1,000 iterations. The solid lines indicate surrogate-assisted algorithms, whereas the dashed lines represent algorithms without surrogated model. The fitness value represents the quality of the schedule, with a lower value indicating better performance. It is shown in Fig. 4 that DG-DCHO achieves the lowest fitness value at both the start and end of the iterations. Furthermore, DG-DCHO shows the fastest convergence rate, reaching the optimal solution significantly faster than the other algorithms.

Fig. 5 presents a comparative analysis of the makespan of eight different algorithms in solving the 20×20 DJSP. The results are visualized in a bar chart, illustrating the effectiveness of each method in minimizing the total completion time of all jobs. It is shown that DG-DCHO achieves the smallest $\hat{C}$, highlighting its capability to generate superior scheduling solutions compared to alternative methods.

To further validate the quality stability of DG-DCHO in dynamic job-shop scheduling tasks, we select a representative

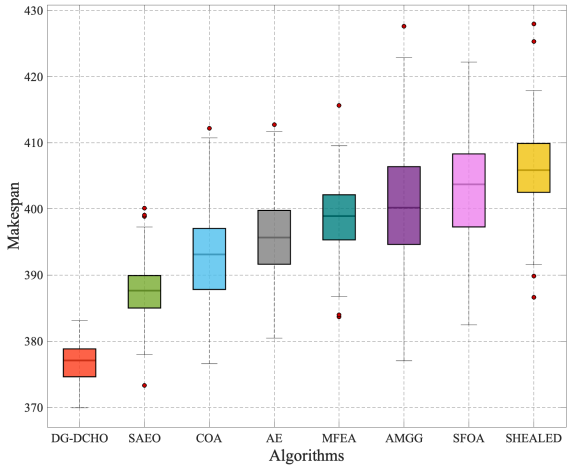


Fig. 6. Comparison of statistical robustness for eight algorithms.

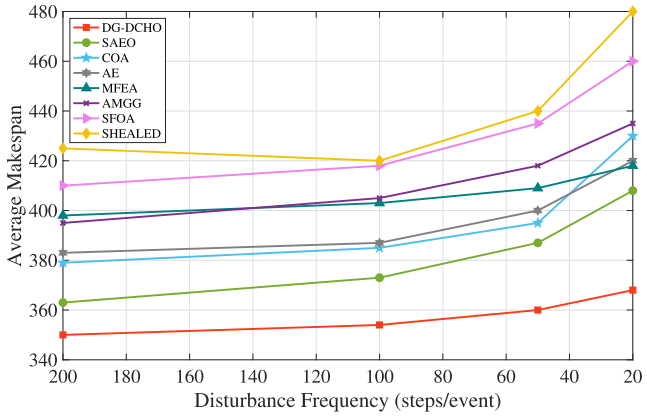


Fig. 7. Robustness analysis under dynamic disturbances.

12 × 12 scale DJSP instance. Under identical experimental settings, DG-DCHO is compared with the above peers, and each is executed 150 times independently. Boxplots of the makespan distributions are generated to visualize the variability in solution quality. As shown in Fig. 6, DG-DCHO consistently outperforms the other algorithms in terms of median makespan and exhibits a significantly narrower interquartile range (*i.e.*, smaller box height), indicating greater stability and robustness across repeated runs. Additionally, the absence of outliers in DG-DCHO further confirms its strong convergence capability when handling random initial populations and dynamic disturbances.

Furthermore, to evaluate the robustness of DG-DCHO under highly dynamic manufacturing conditions, we conduct a controlled experiment within a 15×15 dynamic job-shop environment. This environment simulates two representative types of real-world disturbances: random machine breakdowns and urgent job insertions. By varying the disturbance frequency from low to high, we emulate different levels of environmental volatility. Each algorithm is executed 30 times under each disturbance level, and the average total processing time is recorded. As shown in Fig. 7, DG-DCHO consistently achieves the lowest total processing time across all disturbance levels,

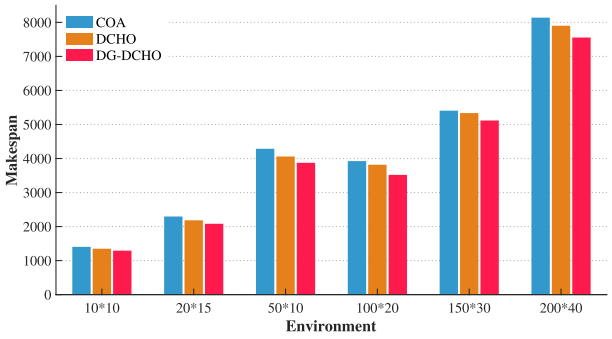


Fig. 8. Makespan comparison across different configurations.

demonstrating strong robustness and exceptional adaptability to frequent disruptions.

To validate the generalization ability of DG-DCHO, Table V comprehensively compares the performance across six different problem scales. The large-scale 200 × 40 instance, along with its associated operational constraints, is sourced from a real-world factory to evaluate the performance of DG-DCHO in a practical industrial scenario [26]. In addition to solution quality, we further analyze the computational efficiency. The minimum, average, and maximum runtimes required by each algorithm to solve DJSP instances of varying scales are recorded. To ensure a rigorous and consistent comparison, a single experimental trial is defined as a complete execution of the algorithm terminating at a fixed limit of 1,000 iterations. All experiments are conducted on the same platform, and each run is repeated 30 times to obtain a stable average. Regarding solution quality, DG-DCHO consistently yields the lowest makespan values. The performance gap widens as the problem size increases, demonstrating the algorithm's strong scalability and generalization ability in complex 3C manufacturing environments. In terms of computational cost, the reported runtime indicators show that DG-DCHO is significantly faster and more stable than the baselines. Notably, even the worst-case runtime of DG-DCHO is consistently lower than the best-case runtime of the runner-up algorithms, confirming its absolute efficiency advantage.

B. Ablation Study

The purpose of the ablation study is to evaluate the contribution of each component of our framework, including COA, DCHO, and DG-DCHO. We assess the performance of these three configurations under varying problem sizes across six different virtual scheduling environments.

1) Ablation Study on Makespan: To assess the impact of DG and the improvements made to the optimization algorithm, we compare the makespan obtained by COA, DCHO, and DG-DCHO across six DJSP environments of different sizes: 10 × 10, 20 × 15, 50 × 10, 100 × 20, 150 × 30, and 200 × 40.

The results of ablation study are presented in Fig. 8, where each bar illustrates the makespan achieved by each configuration. The performance improvements reveal a clear progression. DCHO optimizer, with its advanced search capabilities, already provides a substantial improvement over the baseline COA. The most significant performance gain is

TABLE V
COMPARISON OF MAKESPAN AND RUNTIME FOR DIFFERENT ALGORITHMS ON DIFFERENT DJSP SCALES

Algorithm	Metric	Problem Scale					
		10×10	20×15	50×10	100×20	150×30	200×40
DG-DCHO	Makespan	**1292**	**2082**	**3874**	**3519**	**5117**	**7555**
	Runtime (Avg)	**15.8**	**38.5**	**85.2**	**151.6**	**230.4**	**315.7**
	Runtime (Min / Max)	**14.2 / 17.5**	**35.1 / 42.8**	**78.5 / 93.4**	**140.2 / 165.5**	**212.5 / 251.8**	**295.6 / 340.2**
SAEO	Makespan	1327	2171	4094	3770	5290	7842
	Runtime (Avg)	23.8	63.3	155.4	315.0	549.7	833.1
	Runtime (Min / Max)	20.5 / 28.1	55.8 / 72.5	135.2 / 178.6	280.5 / 355.2	490.8 / 615.4	750.2 / 925.5
AMGG	Makespan	1432	2352	4184	4025	5375	8014
	Runtime (Avg)	28.5	74.3	180.1	380.4	653.0	986.7
	Runtime (Min / Max)	24.2 / 33.5	65.5 / 85.1	160.4 / 205.2	340.2 / 425.8	595.5 / 718.2	905.4 / 1078.5
COA	Makespan	1404	2296	4286	3924	5406	8139
	Runtime (Avg)	18.3	49.7	110.5	243.1	412.8	610.4
	Runtime (Min / Max)	16.1 / 21.4	43.5 / 57.2	98.5 / 125.8	215.2 / 275.5	370.4 / 460.2	550.8 / 681.2
MFEA	Makespan	1418	2331	4256	3997	5558	7936
	Runtime (Avg)	24.2	65.1	158.9	321.5	560.9	845.3
	Runtime (Min / Max)	21.5 / 27.8	58.2 / 73.5	142.5 / 178.4	290.8 / 355.2	510.5 / 615.8	770.4 / 930.5
AE	Makespan	1425	2342	4271	3983	5618	8050
	Runtime (Avg)	22.1	55.4	130.8	275.3	450.1	688.2
	Runtime (Min / Max)	19.5 / 25.4	48.8 / 63.5	115.4 / 148.5	245.2 / 310.5	405.8 / 502.4	620.5 / 765.8
SFOA	Makespan	1451	2365	4390	4076	5694	8215
	Runtime (Avg)	26.0	70.2	171.6	355.8	615.2	921.9
	Runtime (Min / Max)	22.8 / 30.1	62.5 / 79.4	152.4 / 195.8	318.5 / 400.2	560.4 / 675.5	840.5 / 1016.6
SHEALED	Makespan	1473	2385	4448	4108	5763	8280
	Runtime (Avg)	25.6	68.9	165.3	340.7	598.6	890.5
	Runtime (Min / Max)	22.4 / 29.5	60.5 / 78.8	145.2 / 188.5	305.4 / 380.2	540.5 / 665.8	805.2 / 990.0

achieved by DG-DCHO, which consistently outperforms both its standalone optimizer and the baseline to achieve the lowest makespan across all instances. This tiered improvement confirms that while the DCHO optimizer itself is highly effective, the integration of the GCN framework is critical for reaching top-tier performance. The design of DG-DCHO enables the model to learn complex job-machine relationships, generate higher-quality initial schedules, and accelerate convergence, leading to these superior scheduling solutions.

2) Ablation Study on Scheduling Behaviors: Fig. 9 shows Gantt charts illustrating the scheduling behavior of COA, DCHO, and DG-DCHO for a new 30 × 10 environment. Different colors represent different jobs, each consisting of several operations, and their scheduling order is displayed in the Gantt chart. We also compare the scheduling results for the three configurations, where the time taken for job completion is shown. Specifically, the time efficiency in each of the three configurations can be obtained as:

$$Te = \sum_{i \in J} \sum_{l \in O_i} T_{il}, \tag{27}$$

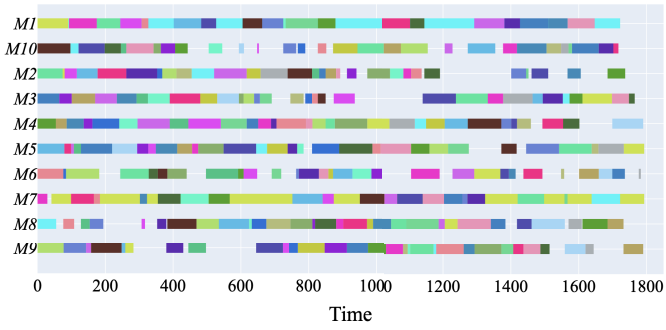
where Te is the total time taken for the completion of all operations in a schedule. The Gantt charts visually represent the scheduling behaviors and the improvement in time efficiency achieved by DG-DCHO.

Fig. 9 shows that DG-DCHO consistently produces the most efficient schedules with the shortest makespan. In contrast, COA and DCHO configurations show more significant idle time and longer job completion times. This indicates that incorporating DG and the improvements made to the original optimization algorithm leads to more efficient scheduling by improving the initial generation of schedules, thereby reducing the overall makespan.

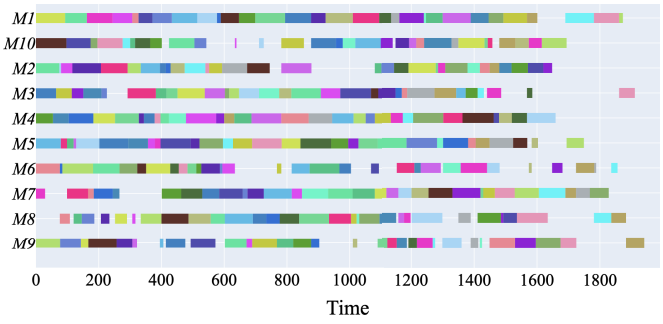
TABLE VI
COMPARISON OF PER-ITERATION TIME AND CONVERGENCE SPEED

Instance	Metric	Algorithm	
		DCHO	DG-DCHO
20×15	Time (ms)	58.4	**38.5**
	Min / Max	55.2 / 62.5	**37.1 / 40.2**
	Iterations	629	**410**
	Min / Max	545 / 687	**372 / 439**
50×10	Time (ms)	162.6	**85.2**
	Min / Max	145.5 / 178.2	**82.5 / 88.9**
	Iterations	726	**454**
	Min / Max	653 / 790	**401 / 514**
200×40	Time (ms)	810.2	**315.7**
	Min / Max	770.5 / 865.8	**305.2 / 328.5**
	Iterations	917	**655**
	Min / Max	852 / 970	**580 / 726**

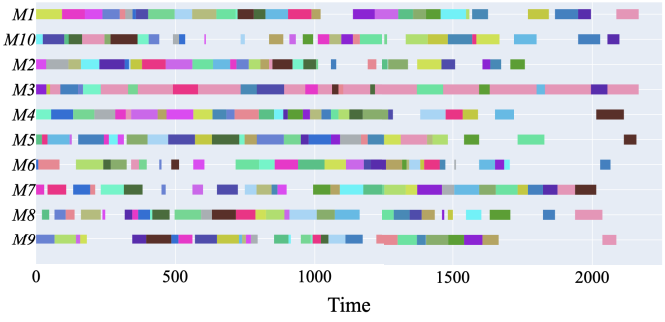
3) Empirical Analysis of Dual-GNN Effectiveness: We record the per-iteration computational time and the number of iterations required for convergence. Table VI presents a



(a) 30*10-DG-DCHO



(b) 30*10-DCHO



(c) 30*10-COA

Fig. 9. Gantt charts for different configurations.

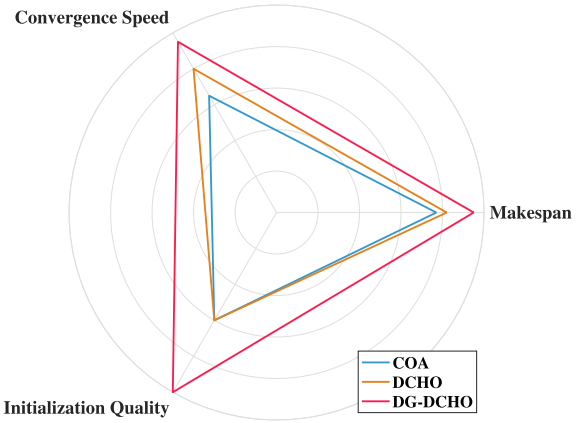


Fig. 10. Tri-metric radar analysis of scheduling performance.

detailed comparison between the baseline DCHO and the proposed DG-DCHO across representative instances. The data reports the average, minimum, and maximum values derived from 30 independent runs. It is shown that DG-DCHO exhibits a substantial reduction in both time consumption and search steps. This efficiency stems directly from the proposed dual-GNN framework. The GCN evaluator replaces the vast majority of time-consuming and simulation-based fitness evaluations with rapid, low-cost predictions. Furthermore, the GCN generator provides a high-quality initial population, which allows the DCHO optimizer to reach a high-quality solution in fewer iterations. This dual advantage confirms that DG-DCHO strikes an exceptional balance between solution quality and computational cost, making it highly practical for real-time scheduling in dynamic 3C manufacturing environments.

4) Tri-Metric Radar Analysis of Scheduling Performance: To intuitively demonstrate the overall performance of different components in terms of scheduling quality, search efficiency, and initialization capability, a radar chart is constructed using three core metrics: final makespan, convergence speed to the optimal solution, and the quality of the initial scheduling sequence. All metrics are normalized and directionally adjusted, and higher values indicate better performance. As illustrated in Fig. 10, DG-DCHO outperforms all methods across the three dimensions, forming the largest and most balanced region on the radar chart. This reflects its comprehensive strength in generating high-quality schedules, accelerating convergence, and producing structurally informed initial solutions. DCHO incorporates dynamic step adaptation and Cauchy mutation, and exhibits strong convergence behavior. However, its use of randomly initialized populations results in greater variability and lower quality in early-stage solutions. In contrast, the baseline COA lacks structural guidance during initialization and is more susceptible to getting trapped in local optima, leading to significantly inferior makespan outcomes compared to DG-DCHO. Overall, the radar chart highlights the advantages of DG-DCHO's dual-GCN architecture and dynamic cooperative optimization mechanism, yielding a well-balanced, efficient, and robust scheduling strategy.

VI. Conclusion and Future Work

In modern 3C manufacturing systems, efficiently solving Dynamic Job-shop Scheduling Problem (DJSP) remains a critical challenge due to unpredictable job arrivals, machine breakdowns, and varying processing times. Traditional optimization methods often struggle with scalability and adaptability, making it challenging to balance solution quality and computational efficiency when solving large-scale scheduling problems in dynamic environment. To address such limitations, this work introduces a novel hybrid framework that synergistically combines Dual-Graph convolutional networks with Dynamic Cooperative Hunting Optimizer named DG-DCHO. Its GCN generator effectively models job-machine dependencies, generating high-quality initial schedules, while DCHO's adaptive search strategies refine these solutions, balancing global exploration and local exploitation. Its GCN evaluator accelerates fitness estimation by learning from prior scheduling results, significantly reducing computational overhead and improving optimization speed. Experimental results demonstrate that DG-DCHO outperforms state-of-the-art

algorithms in both solution quality and computational efficiency, making it a practical approach to large-scale and real-time scheduling problems.

Our future work should focus on extending DG-DCHO to multi-objective optimization scenarios, incorporating additional constraints such as energy consumption and production cost, while further enhancing robustness in highly dynamic environment. We have to adapt the framework to real-world industrial settings by using real-time data streams to improve scheduling flexibility and decision-making accuracy.

REFERENCES

[1] L. He, R. Chiong, W. Li, S. Dhakal, Y. Cao, and Y. Zhang, "Multiobjective optimization of energy-efficient JOB-shop scheduling with dynamic reference point-based fuzzy relative entropy," *IEEE Trans. Ind. Informat.*, vol. 18, no. 1, pp. 600–610, Jan. 2022.

[2] J.-P. Huang, L. Gao, and X.-Y. Li, "A hierarchical multi-action deep reinforcement learning method for dynamic distributed job-shop scheduling problem with job arrivals," *IEEE Trans. Autom. Sci. Eng.*, vol. 22, pp. 2501–2513, 2025.

[3] X. Chen, J. Li, Z. Wang, Q. Chen, K. Gao, and Q. Pan, "Optimizing dynamic flexible job shop scheduling using an evolutionary multitask optimization framework and genetic programming," *IEEE Trans. Evol. Comput.*, vol. 29, no. 5, pp. 1502–1516, Oct. 2025, doi: 10.1109/TEVC.2025.3543770.

[4] F. Zhang, Y. Mei, S. Nguyen, K. C. Tan, and M. Zhang, "Task relatedness-based multitask genetic programming for dynamic flexible job shop scheduling," *IEEE Trans. Evol. Comput.*, vol. 27, no. 6, pp. 1705–1719, Dec. 2023.

[5] P. Zhang, S. Song, S. Niu, and R. Zhang, "A hybrid artificial immune-simulated annealing algorithm for multiroute job shop scheduling problem with continuous limited output buffers," *IEEE Trans. Cybern.*, vol. 52, no. 11, pp. 12112–12125, Nov. 2022.

[6] S.-C. Liu, Z.-G. Chen, Z.-H. Zhan, S.-W. Jeon, S. Kwong, and J. Zhang, "Many-objective job-shop scheduling: A multiple populations for multiple objectives-based genetic algorithm approach," *IEEE Trans. Cybern.*, vol. 53, no. 3, pp. 1460–1474, Mar. 2023.

[7] X. Wang et al., "Hybrid flow shop scheduling with learning effects and release dates to minimize the makespan," *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 54, no. 1, pp. 365–378, Jan. 2024.

[8] H. Yuan, J. Bi, Z. Wang, J. Zhang, M. Zhou, and R. Buyya, "Multiperspective and energy-efficient deep learning in edge computing," *IEEE Internet Things J.*, vol. 13, no. 3, pp. 3988–4003, Feb. 2026, doi: 10.1109/JIOT.2025.3640292.

[9] C. Lin, Z. Cao, and M. Zhou, "Learning-based grey wolf optimizer for stochastic flexible job shop scheduling," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 4, pp. 3659–3671, Oct. 2022.

[10] Z. Zhang, H. Liu, M. Zhou, and J. Wang, "Solving dynamic traveling salesman problems with deep reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 4, pp. 2119–2132, Apr. 2023.

[11] J. Bi et al., "Long-term water quality prediction with transformer-based spatial–temporal graph fusion," *IEEE Trans. Autom. Sci. Eng.*, vol. 22, pp. 11392–11404, 2025, doi: 10.1109/TASE.2025.3535415.

[12] C.-L. Liu and T.-H. Huang, "Dynamic job-shop scheduling problems using graph neural network and deep reinforcement learning," *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 53, no. 11, pp. 6836–6848, Nov. 2023.

[13] J. Bi, Z. Wang, H. Yuan, J. Qiao, J. Zhang, and M. Zhou, "Self-adaptive teaching-learning-based optimizer with improved RBF and sparse autoencoder for complex optimization problems," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, London, U.K., May 2023, pp. 7966–7972.

[14] M. Xu et al., "Genetic programming for dynamic workflow scheduling in fog computing," *IEEE Trans. Services Comput.*, vol. 16, no. 4, pp. 2657–2671, Jul. 2023.

[15] Y. Feng et al., "A two-stage individual feedback NSGA-III for dynamic many-objective flexible job shop scheduling problem," *IEEE Trans. Autom. Sci. Eng.*, vol. 22, pp. 1673–1683, 2025.

[16] J. Bi, Z. Wang, H. Yuan, J. Zhang, and M. Zhou, "Self-adaptive teaching-learning-based optimizer with improved RBF and sparse autoencoder for high-dimensional problems," *Inf. Sci.*, vol. 630, pp. 463–481, Jun. 2023.

[17] F. Zhao, Y. Du, C. Zhuang, L. Wang, and Y. Yu, "An iterative greedy algorithm for solving a multiobjective distributed assembly flexible job shop scheduling problem with fuzzy processing time," *IEEE Trans. Cybern.*, vol. 55, no. 5, pp. 2302–2315, May 2025, doi: 10.1109/TCYB.2025.3538007.

[18] G.-G. Wang, D. Gao, and W. Pedrycz, "Solving multiobjective fuzzy job-shop scheduling problem by a hybrid adaptive differential evolution algorithm," *IEEE Trans. Ind. Informat.*, vol. 18, no. 12, pp. 8519–8528, Dec. 2022.

[19] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Collaborative multifidelity-based surrogate models for genetic programming in dynamic flexible job shop scheduling," *IEEE Trans. Cybern.*, vol. 52, no. 8, pp. 8142–8156, Aug. 2022.

[20] S. Luo, L. Zhang, and Y. Fan, "Real-time scheduling for dynamic partial-no-wait multiobjective flexible job shop by deep reinforcement learning," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 4, pp. 3020–3038, Oct. 2022.

[21] Z. Pan, L. Wang, J. Wang, and J. Lu, "Deep reinforcement learning based optimization algorithm for permutation flow-shop scheduling," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 7, no. 4, pp. 983–994, Aug. 2023.

[22] C.-L. Liu, C.-J. Tseng, and P.-H. Weng, "Dynamic job-shop scheduling via graph attention networks and deep reinforcement learning," *IEEE Trans. Ind. Informat.*, vol. 20, no. 6, pp. 8662–8672, Jun. 2024.

[23] R. Chen, W. Li, and H. Yang, "A deep reinforcement learning framework based on an attention mechanism and disjunctive graph embedding for the job-shop scheduling problem," *IEEE Trans. Ind. Informat.*, vol. 19, no. 2, pp. 1322–1331, Feb. 2023.

[24] Y. Zhao et al., "Learning bi-typed multi-relational heterogeneous graph via dual hierarchical attention networks," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 9, pp. 9054–9066, Sep. 2023.

[25] F. Zhang, Y. Mei, S. Nguyen, K. C. Tan, and M. Zhang, "Instance-rotation-based surrogate in genetic programming with brood recombination for dynamic job-shop scheduling," *IEEE Trans. Evol. Comput.*, vol. 27, no. 5, pp. 1192–1206, Oct. 2023.

[26] K. Lei, P. Guo, Y. Wang, J. Zhang, X. Meng, and L. Qian, "Large-scale dynamic scheduling for flexible job-shop with random arrivals of new jobs by hierarchical reinforcement learning," *IEEE Trans. Ind. Informat.*, vol. 20, no. 1, pp. 1007–1018, Jan. 2024.

[27] X. He, Y. Chen, and P. Ghamisi, "Dual graph convolutional network for hyperspectral image classification with limited training samples," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, Mar. 2022, Art. no. 5502418.

[28] Y. Liu, F. Zhang, Y. Sun, and M. Zhang, "Evolutionary trainer-based deep Q-network for dynamic flexible job-shop scheduling," *IEEE Trans. Evol. Comput.*, vol. 29, no. 3, pp. 749–763, Jun. 2025, doi: 10.1109/TEVC.2024.3367181.

[29] A. Liu, P. B. Luh, K. Sun, M. A. Bragin, and B. Yan, "Integrating machine learning and mathematical optimization for job shop scheduling," *IEEE Trans. Autom. Sci. Eng.*, vol. 21, no. 3, pp. 4829–4850, Jul. 2024.

[30] J. Mou, K. Gao, P. Duan, J. Li, A. Garg, and R. Sharma, "A machine learning approach for energy-efficient intelligent transportation scheduling problem in a real-world dynamic circumstances," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 12, pp. 15527–15539, Dec. 2023.

[31] Q.-z. Xiao, J. Zhong, L. Feng, L. Luo, and J. Lv, "A cooperative coevolution hyper-heuristic framework for workflow scheduling problem," *IEEE Trans. Services Comput.*, vol. 15, no. 1, pp. 150–163, Jan. 2022.

[32] Y. Huang, W. Xu, Y. Dai, S. Maharjan, and Y. Zhang, "Graph deep reinforcement learning for multi-cycle queuing and scheduling in deterministic networking," *IEEE Trans. Netw. Sci. Eng.*, vol. 12, no. 2, pp. 1297–1310, Mar. 2025.

[33] W. Sun, Y. Zou, N. Guan, X. Zhang, G. Du, and Y. Wen, "Graph attention network–based deep reinforcement learning scheduling framework for in-vehicle time-sensitive networking," *IEEE Trans. Ind. Informat.*, vol. 20, no. 7, pp. 9825–9836, Jul. 2024.

[34] R. Lin, Z. Zhou, S. You, R. Rao, and C.-C.-J. Kuo, "Geometrical interpretation and design of multilayer perceptrons," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 2, pp. 2545–2559, Feb. 2024.

[35] M. Koller, B. Fesl, N. Turan, and W. Utschick, "An asymptotically MSE-optimal estimator based on Gaussian mixture models," *IEEE Trans. Signal Process.*, vol. 70, pp. 4109–4123, 2022.

[36] J. Bi et al., "Multi-swarm genetic gray wolf optimizer with embedded autoencoders for high-dimensional expensive problems," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, London, U.K., May 2023, pp. 7265–7271.

[37] M. Cui, L. Li, M. Zhou, and A. Abusorrah, "Surrogate-assisted autoencoder-embedded evolutionary optimization algorithm to solve high-dimensional expensive problems," *IEEE Trans. Evol. Comput.*, vol. 26, no. 4, pp. 676–689, Aug. 2022.
[38] J. Bi, Z. Wang, H. Yuan, J. Zhang, and M. Zhou, "Cost-minimized computation offloading and user association in hybrid cloud and edge computing," *IEEE Internet Things J.*, vol. 11, no. 9, pp. 16672–16683, May 2024.
[39] M. Dehghani, Z. Montazeri, E. Trojovská, and P. Trojovská, "Coati optimization algorithm: A new bio-inspired metaheuristic algorithm for solving optimization problems," *Knowl.-Based Syst.*, vol. 259, pp. 1–43, Jan. 2023.
[40] Z. Liu, G. Li, H. Zhang, Z. Liang, and Z. Zhu, "Multifactorial evolutionary algorithm based on diffusion gradient descent," *IEEE Trans. Cybern.*, vol. 54, no. 7, pp. 4267–4279, Jul. 2024.
[41] C. Zhong, G. Li, Z. Meng, H. Li, A. R. Yildiz, and S. Mirjalili, "Starfish optimization algorithm (SFOA): A bio-inspired metaheuristic algorithm for global optimization compared with 100 optimizers," *Neural Comput. Appl.*, vol. 37, no. 5, pp. 3641–3683, Dec. 2024.
[42] H. Gao and Q. Zhang, "Alpha evolution: An efficient evolutionary algorithm with evolution path adaptation and matrix generation," *Eng. Appl. Artif. Intell.*, vol. 137, pp. 1–31, Nov. 2024.

Jing Bi (Senior Member, IEEE) received the B.S. and Ph.D. degrees in computer science from Northeastern University, Shenyang, China, in 2003 and 2011, respectively. From 2013 to 2015, she was a Post-Doctoral Researcher with the Department of Automation, Tsinghua University, Beijing, China. From 2018 to 2019, she was a Visiting Research Scholar with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, USA. She is currently a Professor with the College of Computer Science, Beijing University of Technology, Beijing. She has over 200 publications in international journals and conference proceedings. Her research interests include distributed computing, cloud and edge computing, large-scale data analytics, machine learning, industrial internet, and performance optimization. She is an Associate Editor of IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS; and IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS.

Chen Wang (Graduate Student Member, IEEE) received the B.E. degree in software engineering from Qingdao University in 2023. He is currently pursuing the master's degree with the College of Computer Science, Beijing University of Technology, Beijing, China. His research interests include task scheduling, intelligent optimization algorithms, deep learning, and machine learning.

Ziqi Wang (Graduate Student Member, IEEE) received the B.S. degree in the Internet of Things and the M.S. degree in software engineering from Beijing University of Technology, China, in 2022 and 2025, respectively. He is currently pursuing the Ph.D. degree with the School of Software Technology, Zhejiang University, China. His research interests include mobile edge computing, task scheduling, intelligent optimization algorithms, and deep learning. He received the Best Paper Award in 2024 ICAIS and ISAS and the Best Application Paper Award in 21st IEEE ICNSC.

Junqi Zhang received the B.S. and Ph.D. degrees from the Department of Automation, Tsinghua University, in 2015 and 2021, respectively. From 2021 to 2022, he was an Algorithm Engineer with JD.com, focusing on ads retrieval. From 2022 to 2023, he transitioned to Qiyuan Laboratory, where he assumed a research role with a focus on vision-language models. His research interests include information retrieval, recommender systems, multi-modal models, self-supervised learning, and explainable AI.

Haitao Yuan (Senior Member, IEEE) received the Ph.D. degree in computer engineering from New Jersey Institute of Technology (NJIT), Newark, NJ, USA, in 2020. He is currently the Deputy Director with the Department of Science and Technology Innovation, Wenchang International Aerospace City, Hainan, China. He is also an Associate Professor with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. He is named in the world's top 2% of scientists list. His research interests include the Internet of Things, edge computing, deep learning, data-driven optimization, and computational intelligence algorithms. He received the Chinese Government Award for Outstanding Self-Financed Students Abroad, the 2021 Hashimoto Prize from NJIT, the Best Paper Award in 17th ICNSC, and the Best Student Paper Award Nominees in 2024 IEEE SMC. He is an Associate Editor of IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS; IEEE INTERNET OF THINGS JOURNAL; and *Expert Systems With Applications*.

Jia Zhang (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Illinois at Chicago. She is currently the Cruse C. and Marjorie F. Calahan Centennial Chair in Engineering and a Professor with the Department of Computer Science, Lyle School of Engineering, Southern Methodist University. Her research interests emphasize the application of machine learning and information retrieval methods to tackle data science infrastructure problems, with a focus on scientific workflows, provenance mining, software discovery, knowledge graph, and interdisciplinary applications of all of these interests in earth science.

Rajkumar Buyya (Fellow, IEEE) received the B.E. degree in computer science and engineering from the University of Mysore in 1992, the M.E. degree from Bangalore University in 1995, and the Ph.D. degree in computer science and software engineering from Monash University, Melbourne, Australia, in 2002. He is a Redmond Barry Distinguished Professor and the Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne, Australia. He has authored over 800 publications and seven textbooks. He was a Future Fellow of Australian Research Council from 2012 to 2016. He is one of the highly cited authors in computer science and software engineering worldwide, with over 156 600 citations and an H-index of 170. He was recognized as a "Web of Science Highly Cited Researcher" from 2016 to 2021 by Thomson Reuters.