2025 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
October 5-8, 2025. Vienna, Austria

2025 IEEE International Conference on Systems, Man, and Cybernetics (SMC) | 979-8-3315-3358-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/SMC58881.2025.11342590

Dual-GNN-Assisted Cooperative Hunting Optimizer for Dynamic Job Shop Scheduling

Chen Wang[1], Jing Bi[1], Ziqi Wang[2], Junqi Zhang[1], Haitao Yuan[3] and Jia Zhang[4]

[1]College of Computer Science, Beijing University of Technology, Beijing 100124, China
[2]School of Software Technology, Zhejiang University, Ningbo 315100, China
[3]School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China
[4]Dept. of Computer Science in Lyle School of Engineering, Southern Methodist University, Dallas, TX, USA

Abstract*—The Dynamic Job Shop Scheduling Problem (DJSP), a critical challenge in 3C 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 optimal schedules in large and complex environments. To solve this problem, this work proposes the Dual-Graph convolutional networks assisted Dynamic Cooperative Hunting Optimizer (DG-DCHO), which integrates graph-convolutional networks (GCN) with metaheuristic optimization to generate high-quality schedules while significantly improving computational efficiency. GCN generator processes graph representations of the job-shop environment and captures complex dependencies among jobs and machines to construct high-quality initial schedules that serve as initial solutions for the optimization process. GCN evaluator estimates makespan values directly from schedule representations and replaces costly fitness evaluation that minimizes computational overhead and improves optimization speed. Dynamic Cooperative Hunting Optimizer (DCHO) serves as the 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 state-of-the-art scheduling algorithms by producing superior solutions while requiring fewer computational resources, making itself a scalable and effective framework for real-time dynamic scheduling in large-scale 3C manufacturing systems.

***Index Terms*—Dynamic job shop scheduling problem, graph convolutional networks, intelligent optimization algorithm, and deep learning.**

I. INTRODUCTION

The multi-constrained dynamic job shop scheduling problem (DJSP) in 3C smart manufacturing is a notable optimization challenge in production and manufacturing systems [1]–[3], where the goal is to efficiently allocate resources, *i.e.*, machines, to jobs under uncertain and time-varying conditions. Traditional approaches to solving the DJSP often rely on heuristics or exact optimization algorithms. However, these methods can be computationally expensive, particularly with large and complex instances [4]–[6]. Heuristic approaches, including genetic algorithms and simulated annealing [7], frequently exhibit slow convergence rates and yield suboptimal solutions due to their exhaustive exploration of high-dimensional solution spaces. Although exact optimization techniques such as integer programming and branch-and-bound algorithms theoretically guarantee global optima [8], their exponential time complexity makes them computationally intractable for large-scale implementations, particularly in time-sensitive operational contexts requiring real-time decision-making capabilities [9]. In addition, their scalability is limited when facing dynamic situations.

To solve the above problems, machine learning and reinforcement learning are adopted. They can dynamically adjust scheduling policies more flexibly according to the problem [10]–[12]. The inherent structure of scheduling problems, where jobs and machines can be represented as graphs, makes them well suited for models based on graph convolutional networks (GCN) [13]. However, while GCN offers the potential for improved decision-making in dynamic environments, a significant challenge remains to efficiently generate high-quality solutions and provide flexible support based on real-world problems [14].

We propose a novel hybrid framework that synergistically combines Dual-Graph Convolutional Networks (DGCN) with Dynamic Cooperative Hunting Optimizer (DCHO) named DG-DCHO to address the DJSP more effectively [15]. It employs two specialized GCN models, each designed to handle distinct aspects of the scheduling process. The GCN Generator is responsible for producing an initial population of candidate schedules, represented as sequences of operations assigned to jobs. It leverages the topological structure of the job-shop environment, utilizing graph-based representations of job-machine relationships to generate feasible action sequences. The second one is the GCN Evaluator, which 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 and provides an objective fitness evaluation by assessing the temporal efficiency of each candidate solution. 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 DGCN with the optimization process is further enhanced by incorporating DCHO combined with Cauchy Mutation [16]. This hybridization allows the al-

This work was supported by the Beijing Natural Science Foundation under Grants L233005 and 4232049, the National Natural Science Foundation of China under Grants 62473014 and 62173013, and in part by Beihang World TOP University Cooperation Program. (Corresponding author: Junqi Zhang.)

gorithm to efficiently explore the vast solution space by balancing exploration and exploitation. Inspired by nature's cooperative hunting strategies, DCHO enables the algorithm to iteratively refine solutions through cooperative behaviors, while the Cauchy Mutation introduces controlled randomization to prevent the algorithm from trapping into local minima [17]. Combining these techniques with the GCN models facilitates the generation of high-quality schedules by dynamically adjusting the population's diversity and the search trajectory.

The remainder of this work is organized as follows. Building on the problem formulation in Section II and the algorithmic framework proposed in Section III, Section IV presents comprehensive experimental results to validate the proposed method.

II. Problem Formulation

The DJSP in 3C smart manufacturing is a resource-constrained, time-varying optimization problem that aims to allocate jobs to machines while minimizing the total time required to complete all jobs (*i.e.*, the makespan). DJSP is characterized by the dynamic and stochastic nature of the environment, including variability in production requirements, machine availability, and potential disruptions such as machine breakdowns.

A. Problem Definition

Let $M=\{M_1, M_2, \ldots, M_j, \ldots, M_m\}$ be the set of machines, where m is the number of machines, and let $J=\{J_1, J_2, \ldots, J_i, \ldots, J_n\}$ be the 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, then $R_{il}=0$, 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:

$$C_{il}=\max(C_{i,l-1}, C_{ab})+T_{il}+T_{\mathrm{w}} \tag{3}$$

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

The 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. This constraint ensures that 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. This ensures that for any given operation in the sequence of a job, exactly one machine is assigned to that operation. It prevents a job operation from being executed simultaneously on multiple machines, which would violate 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 its preceding $O_{i,l-1}$ is finished, *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 an operation has started. This assumption is necessary to maintain the integrity of the job schedule, ensuring that no operation can be interrupted once it has started, which is typical in manufacturing environments where resources are dedicated to specific tasks for the duration of their execution, *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) Machine Breakdowns: There may be machine breakdowns, which lead 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.*,

$$A_j(e)=\begin{cases} 1 & \text{if } M_j \text{ is available at time } e \\ 0 & \text{if } M_j \text{ is unavailable at time } e \end{cases} \tag{8}$$

6) 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 specified deadline, ensuring just-in-time (JIT) production and supply chain synchronization in consumer electronics manufacturing, *i.e.*,

$$C_{ik} \leq D_i, \quad \forall i \in J,\, k \in O_i \tag{9}$$

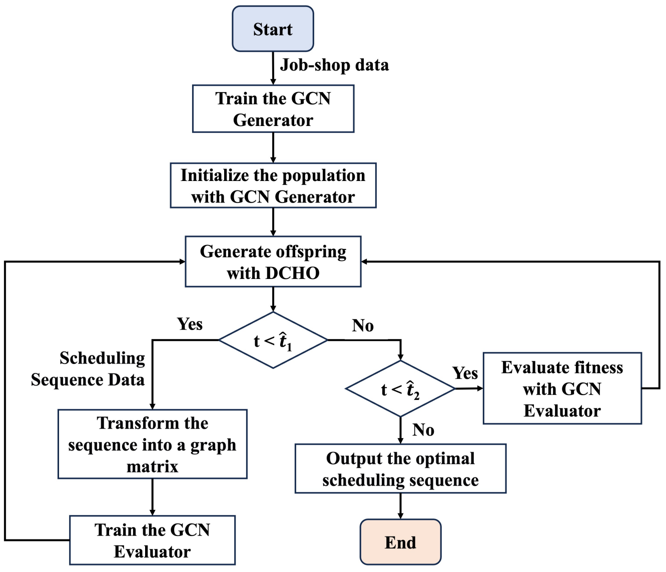
where D_i is the deadline for J_i.

III. Proposed Solution Framework for Dynamic Job Shop Scheduling

This section introduces the solution framework for solving the DJSP using a hybrid approach combining DCHO with DGCN.

A. Framework Overview

DG-DCHO operates iteratively, leveraging the strengths of DGCN for schedule generation and fitness evaluation while utilizing 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, then preprocessing 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 generated schedules is computed through conventional 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 the fitness evaluation process 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. 1 shows the framework of DG-DCHO.



Start
Job-shop data
Train the GCN Generator
Initialize the population with GCN Generator
Generate offspring with DCHO
Yes
t < $\hat{t}_1$
No
Scheduling Sequence Data
t < $\hat{t}_2$
Yes
Evaluate fitness with GCN Evaluator
No
Transform the sequence into a graph matrix
Output the optimal scheduling sequence
Train the GCN Evaluator
End

Fig. 1. Framework of DG-DCHO.

B. GCN Generator

The GCN Generator plays a crucial role in the framework by generating initial candidate schedules through a structured graph representation of the DJSP, where nodes correspond to jobs and machines, and edges encode dependencies and constraints between them. By leveraging the GCN Generator, it learns these relationships and produces an initial sequence of job-machine assignments. The input to the GCN Generator is a graph $G=(V,E)$, where V represents nodes corresponding to J_i, M_j, or specific operations O_{il}, while E represents the edges defining dependencies [18], such as precedence constraints between consecutive operations within the same J_i and potential assignments based on the machine matrix R, linking O_{il} to M_j. The initial node features can include T_{il} for operation nodes or characteristics like $A_j(e)$ for machine nodes. The input graph is processed through multiple graph convolutional layers, which update node representations based on local connectivity. The operation of each graph convolutional layer is obtained as:

$$h_v^{(q+1)} = \sigma\left(\sum_{u\in N(v)} W^{(q)} h_u^{(q)} + b^{(q)}\right) \tag{10}$$

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.*,

$$S = \mathrm{FC}(h_v^{(Q)}) \tag{11}$$

where S denotes the output sequence of job-machine assignments.

C. GCN Evaluator

The GCN Evaluator is responsible for estimating the makespan (fitness) of schedules generated by the GCN Generator. Unlike traditional fitness evaluation, which requires simulating the entire job-shop process, the GCN Evaluator extracts graph-based features and utilizes previously computed fitness values to predict makespan efficiently, significantly improving computational efficiency and optimization speed. The input to the GCN Evaluator is a graph representation of job-machine sequences derived from the sequences generated during the DCHO iteration process. This structured representation allows the GCN Evaluator to exploit spatial and relational dependencies, enhancing its ability to predict makespan effectively.

The architecture of the GCN Evaluator consists of multiple key components. The first stage applies a Graph Attention Layer (GAT) to enhance the model's ability to focus on relevant nodes and edges [19], learning the significance of different job-machine relationships in predicting makespan. This operation is obtained as:

$$h_v = \mathrm{GAT}(\{h_u, e_{uv}\}) \tag{12}$$

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 [20]. This compact representation is subsequently fed into a Multi-Layer Perceptron (MLP), 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.*,

$$\hat{C} = \text{MLP}(h_v^{(Q)}) \tag{13}$$

The GCN Evaluator is trained using the mean squared error (MSE) loss function, minimizing the difference between predicted and actual makespans [21], *i.e.*,

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\left(C_{\text{p}}(S_y) - C_{\text{u}}(S_y)\right)^2 \tag{14}$$

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

D. DCHO

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 each 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. As a metaheuristic algorithm inspired by cooperative hunting strategies in nature, DCHO iteratively improves scheduling solutions by balancing exploration and exploitation to achieve better optimization. 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:

$$S_y^{(t+1)} = S_y^{(t)} + \alpha \cdot (S_{\text{b}} - S_y^{(t)}) + \beta \cdot \mathcal{N}(0, \Gamma) \tag{15}$$

where S_{b} denotes the best schedule 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. 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:

$$\delta^{(t)} = \Delta \cdot \left(1 - \frac{t}{T}\right) + \delta \cdot \frac{t}{T} \tag{16}$$

where Δ and δ represent the initial and final step sizes, respectively, and T is the total number of iterations. 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:

$$S_y^{(t+1)} = S_y^{(t)} + \gamma \cdot \mathcal{C}(0, 1) \tag{17}$$

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. This mechanism introduces larger jumps compared to Gaussian perturbations, improving the algorithm's robustness against premature convergence.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

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

A. Ablation Study

The purpose of the ablation study is to evaluate the contribution of each component of our framework, specifically the COA, DCHO, and the DG-DCHO. We assess the performance of these three configurations under varying problem sizes across six different virtual scheduling environments. The goal is to observe how the introduction of DGCN and the improvements made to the original optimization algorithm enhance overall performance, particularly regarding solution quality (makespan) and convergence speed.

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

The results are presented in Fig. 2, where the bar charts illustrate the makespan achieved in each case. Across all instances, DG-DCHO consistently outperforms both COA and DCHO, achieving the lowest makespan.

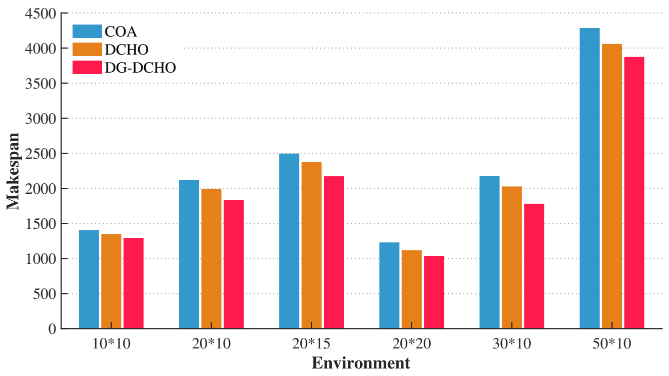


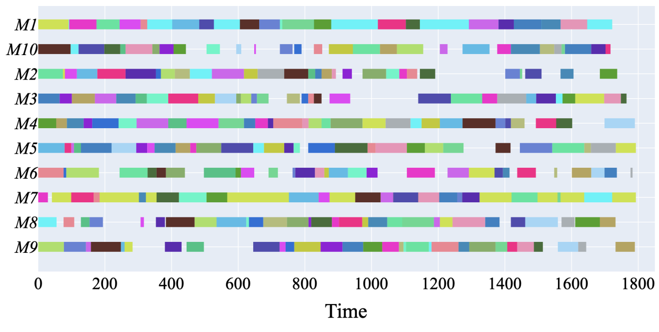
Fig. 2. Makespan comparison across different configurations.

2) Ablation Study on Scheduling Behavior: Fig. 3 shows Gantt charts illustrating the scheduling behavior of COA, DCHO, and DG-DCHO for a new 30x10 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:

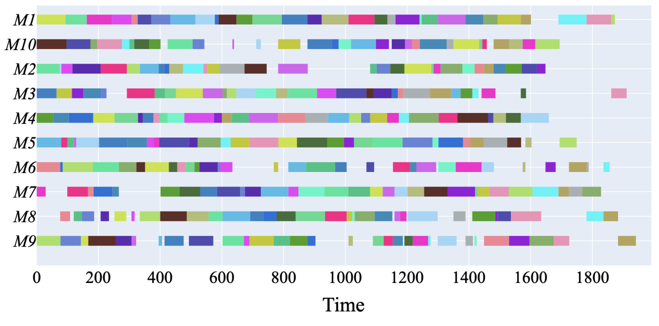
$$Te = \sum_{i \in J} \sum_{l \in O_i} T_{il} \tag{18}$$

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.

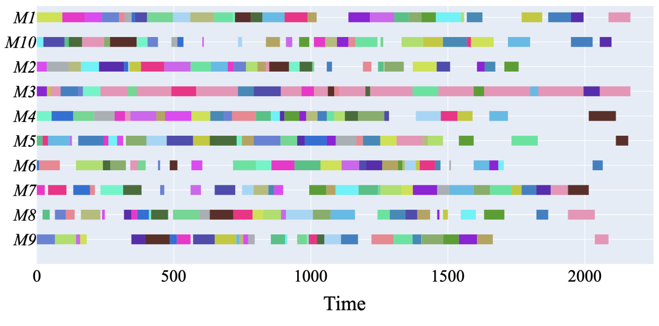
Fig. 3 shows that DG-DCHO consistently produces the most efficient schedules with the shortest makespan. In contrast, the COA and DCHO configurations show more significant idle time and longer job completion times.



(a) 30*10-DG-DCHO



(b) 30*10-DCHO



(c) 30*10-COA

Fig. 3. Gantt charts for different configurations.

B. Comparison Experiments

To further validate the performance of DG-DCHO, we compare it with several state-of-the-art scheduling algorithms, including AMGG [22], SAEO [23], SHEALED [24], COA [25], and DG-DCHO. They are conducted under the same DJSP environments to assess the solution quality and computational efficiency.

1) Comparison of Convergence Speed: Fig. 4 presents the fitness value iteration results for all five algorithms, where each algorithm's fitness is evaluated over 1000 iterations. 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 quicker than the other algorithms.

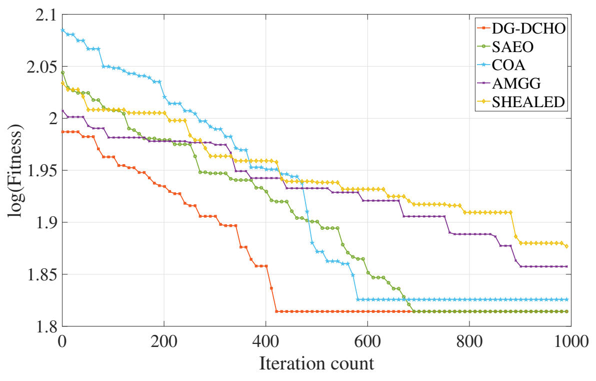


Fig. 4. Comparison of convergence speed for five algorithms.

2) Comparison of Makespan: Fig. 5 presents a comparative analysis of the makespan of five 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 in Fig. 5 that DG-DCHO achieves the smallest $\hat{C}$, highlighting its capability to generate superior scheduling solutions compared to alternative methods.

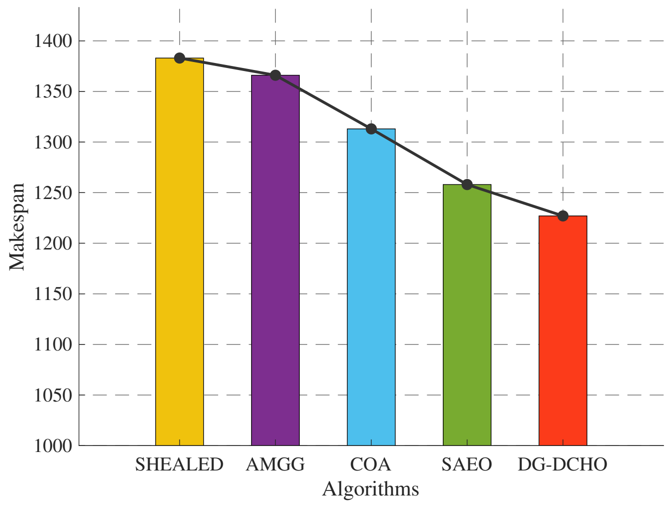


Fig. 5. Comparison of makespan for five algorithms.

V. CONCLUSIONS

In modern 3C manufacturing systems, efficient 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 difficult to balance solution quality and computational efficiency in large-scale dynamic environments. To address these limitations, this work introduces a novel hybrid framework that synergistically combines Dual-Graph Convolutional Networks (DGCN) with Dynamic Cooperative Hunting Optimizer (DCHO) named DG-DCHO. 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. Moreover, 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 an effective approach for large-scale, real-time scheduling.

Future work will 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 environments. Additionally, the framework will be adapted to real-world industrial settings, integrating real-time data streams to further improve scheduling flexibility and decision-making accuracy.

REFERENCES

[1] 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 Transactions on Evolutionary Computation*, vol. 27, no. 5, pp. 1192–1206, Oct. 2023.

[2] 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 Transactions on Industrial Informatics*, vol. 20, no. 1, pp. 1007–1018, Jan. 2024.

[3] 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 Transactions on Industrial Informatics*, vol. 18, no. 1, pp. 600–610, Jan. 2022.

[4] Y. Feng, Y. Lin, Z. Yang, Y. Xu, D. Li, X. Li, and D. Yang, "A Two-Stage Individual Feedback NSGA-III for Dynamic Many-Objective Flexible Job Shop Scheduling Problem," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 1673–1683, Apr. 2024.

[5] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Collaborative Multifidelity-Based Surrogate Models for Genetic Programming in Dynamic Flexible Job Shop Scheduling," *IEEE Transactions on Cybernetics*, vol. 52, no. 8, pp. 8142–8156, Aug. 2022.

[6] 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 Transactions on Evolutionary Computation*, vol. 27, no. 6, pp. 1705–1719, Dec. 2023.

[7] M. Xu, Y. Mei, S. Zhu, B. Zhang, T. Xiang, F. Zhang, and M. Zhang, "Genetic Programming for Dynamic Workflow Scheduling in Fog Computing," *IEEE Transactions on Services Computing*, vol. 16, no. 4, pp. 2657–2671, Aug. 2023.

[8] G. -G. Wang, D. Gao, and W. Pedrycz, "Solving Multiobjective Fuzzy Job-Shop Scheduling Problem by a Hybrid Adaptive Differential Evolution Algorithm," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 12, pp. 8519–8528, Dec. 2022.

[9] 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," *Information Sciences*, vol. 630, no. 56, pp. 463–481, Jun. 2023.

[10] H. Yuan, Q. Hu, M. Wang, S. Wang, and J. Bi, "Data-Filtered Prediction With Decomposition and Amplitude-Aware Permutation Entropy for Workload and Resource Utilization in Cloud Data Centers," *IEEE Internet of Things Journal*, vol. 12, no. 12, pp. 19189–19201, Jun. 2025.

[11] H. Zhao, Z. Wang, G. Cheng, W. Qian, P. Chen, J. Yin, S. Dustdar, and S. Deng, "Online Workload Scheduling for Social Welfare Maximization in the Computing Continuum," *IEEE Transactions on Services Computing,* doi: 10.1109/TSC.2025.3570845.

[12] J. Bi, Z. Wang, H. Yuan, X. Wu, J. Zhang, and M. Zhou, "Long-term Water Quality Prediction with Transformer-based Spatial-Temporal Graph Fusion," *IEEE Transactions on Automation Science and Engineering*, doi: 10.1109/TASE.2025.3535415.

[13] C. -L. Liu and T. -H. Huang, "Dynamic Job-Shop Scheduling Problems Using Graph Neural Network and Deep Reinforcement Learning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 11, pp. 6836–6848, Nov. 2023.

[14] J. Chen, L. Wang, Y. Liang, Y. Yu, J. Feng, J. Zhao, and X. Ding, "Order Dispatching Via GNN-Based Optimization Algorithm for On-Demand Food Delivery," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 10, pp. 13147–13162, Oct. 2024.

[15] X. He, Y. Chen, and P. Ghamisi, "Dual Graph Convolutional Network for Hyperspectral Image Classification With Limited Training Samples," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–18, Mar. 2022.

[16] 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," *2023 IEEE International Conference on Robotics and Automation (ICRA)*, London, United Kingdom, pp. 7966–7972, 2023.

[17] H. Yuan, J. Bi, Z. Wang, J. Yang, and J. Zhang, "Partial and Cost-minimized Computation Offloading in Hybrid Edge and Cloud Systems," *Expert Systems with Applications*, vol. 250, no. 15, pp. 1–13, Sept. 2024.

[18] 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 Transactions on Network Science and Engineering*, vol. 12, no. 2, pp. 1297–1310, Apr. 2025.

[19] 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 Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 1322–1331, Feb. 2023.

[20] R. Lin, Z. Zhou, S. You, R. Rao, and C. . -C. J. Kuo, "Geometrical Interpretation and Design of Multilayer Perceptrons," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 2, pp. 2545–2559, Feb. 2024.

[21] M. Koller, B. Fesl, N. Turan, and W. Utschick, "An Asymptotically MSE-Optimal Estimator Based on Gaussian Mixture Models," *IEEE Transactions on Signal Processing*, vol. 70, pp. 4109–4123, Jul. 2022.

[22] J. Bi, J. Zhai, H. Yuan, Z. Wang, J. Qiao, J. Zhang, and M. Zhou, "Multi-swarm Genetic Gray Wolf Optimizer with Embedded Autoencoders for High-dimensional Expensive Problems," *2023 IEEE International Conference on Robotics and Automation (ICRA)*, London, United Kingdom, pp. 7265–7271, 2023.

[23] M. Cui, L. Li, M. Zhou, and A. Abusorrah, "Surrogate-Assisted Autoencoder-Embedded Evolutionary Optimization Algorithm to Solve High-Dimensional Expensive Problems," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 4, pp. 676–689, Aug. 2022.

[24] 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 of Things Journal*, vol. 11, no. 9, pp. 16672–16683, May 2024.

[25] M. Dehghani, Z. Montazeri, E. Trojovská, and P. Trojovský, "Coati Optimization Algorithm: A new bio-inspired metaheuristic algorithm for solving optimization problems," *Knowledge-Based Systems*, vol. 259, pp. 1–43, Jan. 2023.