engineering proceedings

MDPI

Proceeding Paper

Neuro-Evolution of Augmenting Topologies for Dynamic Scheduling of Flexible Job Shop Problem †

Jian Huang, Yarong Chen *, Jabir Mumtaz * and Liuyan Zhong

School of Mechanical and Electrical Engineering, Wenzhou University, Wenzhou 325035, China; hpersona@163.com (J.H.); liuyanzhong2020@163.com (L.Z.)

* Correspondence: yarongchen@126.com (Y.C.); jabirmumtaz@live.com (J.M.)

† Presented at the 4th International Conference on Advances in Mechanical Engineering (ICAME-24), Islamabad, Pakistan, 8 August 2024.

Abstract: In flexible production environments, challenges such as fluctuating customer demands and machine performance degradation significantly complicate production scheduling. This study introduces a neuro-evolution of augmenting topologies (NEAT) algorithm aimed at optimizing the scheduling efficiency in flexible job shops by minimizing both maximum completion and average lag times, taking into account variables like sporadic job arrivals, variable machining durations, tool wear, preventive maintenance, and equipment failures. The NEAT algorithm harnesses the features of dynamic flexible job shop scheduling problems (DFJSPs) to devise heuristic rules for job selection and machine allocation, synthesizing these rules into coherent scheduling strategies. Employing the entropy weight method, a fitness function for multiobjective optimization is formulated, facilitating the enhancement of the neural network's structural and nodal parameters through genetic algorithms. Comparative analysis with four conventional scheduling rules indicates that the NEAT approach consistently surpasses traditional methods, especially in managing complex disturbances. For example, in a scenario involving 50 jobs and 20 machines, NEAT dramatically reduced the average completion time to 142.14 s, markedly outperforming the 644.36 s achieved by the minimum operation completion rate/shortest processing time (MOCR/SPT) approach. These findings underscore the superiority of NEAT in dynamic scheduling contexts.

Keywords: flexible job shop; enhanced topological neural evolution; minimizing maximum completion time; mean time delay

Citation: Huang, J.; Chen, Y.; Mumtaz, J.; Zhong, L. Neuro-Evolution of Augmenting Topologies for Dynamic Scheduling of Flexible Job Shop Problem. *Eng. Proc.* **2024**, *75*, 19. https://doi.org/10.3390/engproc2024075019

Academic Editors: Muhammad Mahabat Khan, Muhammad Irfan, Mohammad Javed Hyder and Manzar Masud

Published: 24 September 2024

1. Introduction

With the rise and development of multispecies and small-lot production modes, modern manufacturing often involves multiple machines capable of processing a single operation. This is referred to as the flexible job-shop scheduling problem (FJSP). Initially proposed and studied by Brucker et al., the FJSP extends the traditional job-shop scheduling problem, which only requires determining the sequence of operations on a single machine. The FJSP introduces the additional complexity of assigning operations to alternative machines, while the dynamic flexible job-shop scheduling problem (DFJSP) further incorporates dynamic events [1].

Researchers typically classify methods for solving the FJSP into exact and approximate methods. Classic approximate methods include heuristic rules and metaheuristic algorithms. Currently, heuristic methods for solving the FJSP are usually based on scheduling rules designed to generate acceptable schedules within a short timeframe [2]. Common heuristics include shortest processing time first (SPT), earliest due date first (EDD), first in first out (FIFO), and least work remaining (LWR) [3]. Metaheuristic methods, which are the main approaches for solving the FJSP, include genetic algorithms, artificial bee colony algorithms, simulated annealing, and ant colony optimization. For example, Sun

et al. proposed an improved variable neighborhood search hybrid genetic algorithm for the FJSP considering machine workload balancing, which significantly outperformed other algorithms [4].

With the advent of the big data era, the acquisition and processing of shop floor data have become more accessible, and the emergence of deep reinforcement learning has provided researchers with innovative approaches to address shop floor scheduling problems involving perturbation events. Many researchers have applied deep reinforcement learning methods to solve the FJSP [5]. Luo introduced a new approach based on Deep Q-Network (DQN) for flexible job-shop scheduling with new job insertions [6]. This method involves designing six combinatorial scheduling rules as expert experience to assign jobs to feasible machines and using seven features ranging from 0 to 1 to represent the production state at each rescheduling point. Experimental results demonstrate that this method outperforms classical heuristic rules and Q-learning methods in terms of superiority and generalizability.

Liu et al. proposed a hierarchical distributed architecture to address the flexible job-shop scheduling problem with stochastic dynamic events [7]. They trained two types of agents—allocation and ranking—based on the Double DQN algorithm and designed an alternative reward shaping technique to enhance learning efficiency and scheduling effectiveness. Experiments proved that these agents, whether used individually or integrated, outperform existing scheduling strategies across a wide range of problem sizes. NEAT is an evolutionary algorithm that simultaneously optimizes the structure and weights of neural networks using genetic algorithms [8]. This research proposes an NEAT algorithm that integrates the features of the DFJSP and designs heuristic rules for job selection and machine allocation. These rules are combined as scheduling behaviors. Using the entropy weight method, a multiobjective optimization fitness function is designed, and the neural network's topology and node weight parameters are optimized using genetic algorithms. Experimental comparisons verify that the proposed method achieves superior objective values compared to heuristic rules.

2. Problem Description

The flexible job shop scheduling problem (FJSP) requires each job to be processed on specific machines, with each job consisting of multiple operations where each operation can select from a set of machines. The processing time for each operation varies depending on the machine. The aim of the FJSP is to assign machines for each operation and sequence these operations to devise an effective scheduling scheme that minimizes key performance indicators: the maximum completion time (cmax) and tardiness, where tardiness measures the delay of job completion past its due date. In complex and variable manufacturing environments, typical challenges such as random job arrivals, tool wear, and machine failures often disrupt the preplanned scheduling. This study proposes a dynamic scheduling method for flexible job shops to adapt scheduling in real time.

Based on the problem description, it is assumed that the flexible job shop follows the following assumptions in the production process: (1) Each machine can process only one operation at a time; (2) each job is processed on only one machine at any given moment; (3) operations within the same job follow a specified sequence: a subsequent operation cannot begin until the previous one has completed, but operations across different jobs are not sequence-dependent; (4) all jobs have the same priority in the initial state; (5) all machines are available at the start of scheduling; (6) machine setup times and inter-machine transportation times are significant and are included in the processing times of each operation to reflect their impact on the scheduling efficiency accurately.

3. Dynamic Scheduling Method for Flexible Job Shop Based on NEAT Algorithm

The NEAT deep reinforcement learning method does not require a predefined network structure, allowing the neural network structure to evolve automatically. Based on the characteristics of flexible job shops and the NEAT deep reinforcement learning method, an NEAT-based dynamic scheduling method is proposed, as shown in Figure 1.

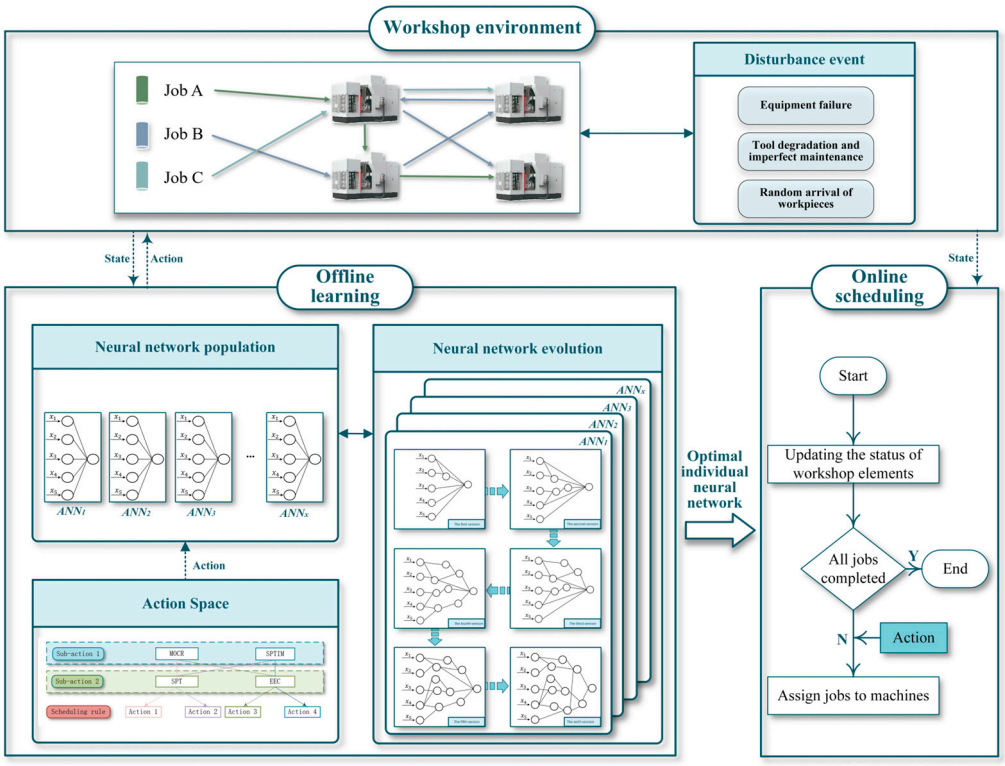

Figure 1. NEAT-based dynamic scheduling method.

3.1. Algorithm Pseudocode

In the first module, the system actively simulates dynamic events occurring within the workshop environment, such as the arrival of new jobs and the completion of existing tasks. It continuously updates the state of the workshop in real time, adapting to changes and disruptions. A well-trained neural network plays a crucial role in this module, where it analyzes the current state of the shop floor and determines the most effective scheduling actions based on predefined criteria and past learning. This process is ongoing and does not cease until either all jobs have been successfully processed or the system reaches a predetermined limit of decision-making cycles. At the conclusion of each cycle, the system evaluates the performance of the neural network using a fitness value, which reflects the efficacy and efficiency of the decisions made by the system. This feedback loop helps in fine-tuning the neural network over time, improving its accuracy and reliability in decision-making.

The second module implements the NEAT algorithm (Algorithm 1), responsible for the evolution of neural networks. Within each generation, the system first calculates the distances between individual neural network entities within the population. It classifies the population into species based on a species compatibility threshold and performs selection, mutation, and crossover operations to generate new neural networks. Moreover, these newly formed networks are tested through simulated interactions with the workshop environment, and their fitness is assessed accordingly. Subsequently, networks that perform poorly are culled, and the population is updated to enhance overall performance. This iterative process is carried out until a termination condition is met, culminating in the output of the optimally performing neural network.

3.2. State Space

Based on the characteristics of flexible job shops, the workshop environment is described by the state features of jobs and machines, denoted as $S = \left\{ D_i^{DC}, R_i^O, R_i^J, t_i^S, t_k^A, t_k^{RUL}, t_k^W, t_k^R, ur_i, \overline{t_{ij}^A} \right\}$. Here, D_i^{DC} is the difference between the due date of the job in the buffer

area and the decision time (current time), and R_i^O is the completion rate of the process at the decision moment. It is expressed as the ratio of the number of processes that have been processed on the job to the total number of processes on the job. R_i^J is the overall job completion rate at the decision time, t_i^S is the slack time of the job, t_k^A is the cumulative processing time of the machine, t_k^{RUL} is the remaining useful life of the tool corresponding to the machine, t_k^W is the processing completion time of the job scheduled on M_k, t_k^R is the remaining processing time for the job on machine M_k, ur_i is the urgency of job i, and $\overline{t_{ij}^A}$ is the expected processing time for the next operation of the job waiting in the buffer on an idle machine.

Algorithm 1 NEAT-based dynamic scheduling algorithm for flexible job shops

```
Initialize task sequence and state S
while termination condition is not met:
    # Module 1: Agent and Workshop Environment Interaction Learning
    while not all jobs are processed and t < maximum decision count:
            Update state on new job arrivals or completions
            if job completed:
                Update production sequence and job completion time
            t += 1
            Update the instantaneous state St and input to neural network P
            Neural network P outputs Q values four scheduling actions:
            Choose action a = argmax(Q)
            Execute action a_t
        if t < maximum decision count:
            Calculate the fitness value G of neural network P
        else:
            G = float('-inf')
    # Module 2: NEAT Neural Network Population Evolution
        for each generation g:
            for each network P:
                Calculate distances, classify into species
                Select top individuals for new generation
                Mutate and cross to produce new networks
        Evaluate fitness of new and offspring networks using "Interaction Learning"
        Eliminate low fitness individuals, update species
        Output optimal network P_best
```

3.3. Action Space

The action space comprises the set of actions an agent can choose at decision points, triggered by process completion or new job arrival. Actions are selected based on the shop floor state and can be broken down into two subactions: (1) selecting the next job to process and (2) selecting a suitable machine. Two heuristic rules are designed for each sub-action, combining into four scheduling actions:

Action 1: Minimum operation completion rate (MOCR): Selects the job with the lowest operation completion rate. If multiple jobs are selected, one is chosen randomly.

Action 2: Shortest processing time on idle machine (SPTIM): Selects the job with the shortest next operation time on an idle machine. If multiple jobs are selected, one is chosen randomly.

Action 3: Shortest processing time (SPT): This action prioritizes the selection of the machine, offering the shortest processing time for the forthcoming operation. In instances where several machines present identical shortest processing times, a machine is selected at random to ensure equitable machine utilization and operational efficiency.

Action 4: Estimated earliest complete (EEC): Selects the machine expected to complete the process earliest. If multiple machines meet this condition, the machine that will be available latest is chosen.

Action 5: Unselect job (UJ): This action is employed when no job is chosen for processing due to various constraints, such as a lack of jobs in the staging area or the absence of idle machines capable of processing the jobs. Action 5 is not viable if all machines are idle while jobs are present in the staging area.

The selection of action 5 obeys the following constraints: (1) If a new decision point is triggered and the staging area is empty, action 5 must be selected. (2) If a new decision point is triggered with jobs present in the staging area, but no machines are available or capable of processing, action 5 must be selected. (3) If a new decision point is triggered, jobs are available in the staging area, and all machines are idle, action 5 cannot be selected.

3.4. Fitness Function

To achieve the objectives of minimizing both the maximum completion time and total tardiness in a flexible job shop, the entropy weighting method is utilized to convert these scheduling goals into quantifiable scores [9]. The method is detailed as follows:

(1) Defining Evaluation Objects and Criteria

Each action within the job shop is considered as an evaluation object. The set of x evaluation objects forms the scheduling plan set $S = \{S_1, S_2, S_3, \cdots, S_x\}$, There are y evaluation criteria, forming the criteria set $I = \{I_1, I_2, I_3, \cdots, I_y\}$, with the value of criterion I_y for plan S_x denoted as V_{xy} Thus, the matrix for the scheduling criteria is

$$V = \begin{bmatrix} V_{11} & \cdots & V_{1y} \\ \vdots & \ddots & \vdots \\ V_{x1} & \cdots & V_{xy} \end{bmatrix} \tag{1}$$

(2) Normalization

To address the issue of unequal ranges among criteria, normalization is applied where smaller values indicate better performance. The normalization formula is given by

$$V'_{ab} = \frac{V_{max} - V_{ab}}{V_{max} - V_{min}} \tag{2}$$

(3) Entropy Calculation of Criteria

Firstly, we compute the proportion of each criterion value for each scheduling plan relative to all plans for that criterion:

$$V_{ab} = \frac{V'_{ab}}{\sum_{a=1}^{x} V'_{ab}} \tag{3}$$

Then, we calculate the entropy for each criterion:

$$E_b = -\frac{\sum_{a=1}^{x} V_{ab} \ln V_{ab}}{\ln x} \tag{4}$$

(4) Weight Calculation of Criteria

After determining the entropy values, we calculate the weights for each criterion:

$$\omega_b = \frac{1 - E_b}{\mathrm{y} - \sum_{b=1}^{y} E_b} \tag{5}$$

(5) Scoring of Scheduling Plans

Using the normalized values and weights, the score for each scheduling plan is computed as follows:

$$f_a = \sum_{b=1}^{y} \omega_b \times V'_{ab} \tag{6}$$

(6) Scoring of Scheduling Plans

The objective value for each plan is derived using the following formula:

$$f_a = \omega_1 \times V_{a1} + \omega_2 \times V_{a2} \tag{7}$$

(7) Scoring of Scheduling Plans

To enhance distinction between plans and improve the effectiveness of the training for the agents, the fitness function is defined as

$$F = \frac{1}{f_a} \tag{8}$$

4. Numerical Example and Analysis

4.1. Parameter Settings

In this paper, the number of machines m and the number of jobs n classify the problem into small and medium scales [10]. The number of processes s_i, machining time t^O_{ijk}, arrival time T^A_i, and delivery time T^D_i of the jobs are generated based on the uniform distribution, and the experimental parameters are shown in Table 1.

Table 1. Experimental parameters for the dynamic scheduling problem in flexible job shop.

Problem	m	n	s_i	t^O_{ijk}	T^A_i	T^D_i
Small-scale	2, 5, 10	10	U [1, 5]	U [1, 50]	U [1, 10]	U [150, 250]
Medium-scale	5, 10, 20	50	U [1, 10]	U [1, 50]	U [1, 50]	U [400, 700]

4.2. Results Analysis

According to the design of the algorithm parameters in Table 1, different scale scheduling problems under perturbation events are solved and compared with the designed scheduling rules, and the mean and standard deviation data of the objective values of different dynamic scheduling methods under perturbation events are shown in Table 2. The NEAT algorithm consistently shows lower mean values across almost all configurations compared to other methods (MOCR/SPT, SPTIM/SPT, MOCR/EEC, SPTIM/EEC). This indicates a more efficient handling of dynamic scheduling tasks, especially under complex conditions marked by simultaneous disturbances. The standard deviation values associated with NEAT are significantly lower compared to other methods in most cases. This suggests that NEAT not only performs better on average but also offers more stability and reliability in its performance under perturbing conditions.

Table 2. Objective values (mean/std dev) for three simultaneous perturbing events.

n	m	NEAT	MOCR/SPT	SPTIM/SPT	MOCR/EEC	SPTIM/EEC
10	2	446.50/70.7	3030.4/337.98	2097.71/589.3	2768.76/613.09	1832.85/533.22
	5	90.63/5.26	162.91/20.66	129.54/14.23	185.25/18.76	117.48/10.14
	10	63.68/3.19	76.46/8.21	59.29/9.74	78.31/8.64	73.92/8.54
50	5	2852.62/31.71	4966.13/522.89	3703.33/675.08	4613.15/732.89	4114.33/747.37
	10	431.52/62.15	2051.28/357.46	1858.69/464.74	1993.52/530.72	1846.08/425.59
	20	142.14/5.88	644.36/241.5	170.48/9.79	945.21/300.1	169.55/9.07

In the dynamic scheduling problem that considers three types of disturbances simultaneously, the NEAT deep reinforcement learning approach continues to achieve superior average objective values. Moreover, in scheduling problems with limited machine resources, the objective values obtained by the NEAT deep reinforcement learning method

significantly surpass those of heuristic rules. The standard deviation of the NEAT method is also more favorable compared to heuristic rules.

5. Conclusions

For the dynamic scheduling problem in flexible job shops with random job arrivals, tool degradation, and machine breakdowns, an NEAT-based method was studied. This method combines genetic algorithms and neural networks to optimize the network structure and parameters. The trained NEAT agent efficiently solves large-scale, randomly generated problems. Compared to heuristic rules, the NEAT-based method achieves better objective values and adaptively selects the best scheduling rules, offering an effective approach for practical workshop scheduling.

Author Contributions: Conceptualization, J.H. and Y.C.; methodology, J.H. and L.Z.; software, J.H. and J.M.; validation, J.H. and J.M.; formal analysis, J.H. and Y.C.; writing—review and editing, Y.C. and L.Z.; visualization, L.Z.; supervision, J.H.; funding acquisition, Y.C. All authors have read and agreed to the published version of the manuscript.

Funding: Basic scientific research project of Wenzhou City (G202306 & G20240020).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data are available from the corresponding author upon request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Brucker, P.; Schlie, R. Job-shop scheduling with multi-purpose machines. *Computing* **1991**, *45*, 369–375. [CrossRef]
2. Panwalkar, S.; Iskander, W. A survey of scheduling rules. *Oper. Res.* **1977**, *25*, 45–61. [CrossRef]
3. Jun, S.; Lee, S.; Chun, H. Learning dispatching rules using random forest in flexible job shop scheduling problems. *Int. J. Prod. Res.* **2019**, *57*, 3290–3310. [CrossRef]
4. Sun, K.; Zheng, D.; Song, H. Hybrid genetic algorithm with variable neighborhood search for flexible job shop scheduling problem in a machining system. *Expert Syst. Appl.* **2022**, *215*, 119359. [CrossRef]
5. Song, W.; Chen, X.; Li, Q. Flexible job-shop scheduling via graph neural network and deep reinforcement learning. *IEEE Trans. Ind. Inform.* **2023**, *19*, 1600–1610. [CrossRef]
6. Luo, S. Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. *Appl. Soft Comput.* **2020**, *91*, 106208. [CrossRef]
7. Liu, R.; Piplani, R.; Toro, C. Deep reinforcement learning for dynamic scheduling of a flexible job shop. *Int. J. Prod. Res.* **2022**, *60*, 4049–4069. [CrossRef]
8. Lang, S.; Reggelin, T.; Schmidt, J.; Muller, M.; Nahhas, A. NeuroEvolution of augmenting topologies for solving a two-stage hybrid flow shop scheduling problem: A comparison of different solution strategies. *Expert Syst. Appl.* **2021**, *172*, 114666. [CrossRef]
9. Chen, P. Effects of the entropy weight on TOPSIS. *Expert Syst. Appl.* **2021**, *168*, 114186. [CrossRef]
10. Ding, L.; Guan, Z.; Rauf, M. Multi-policy deep reinforcement learning for multi-objective multiplicity flexible job shop scheduling. *Swarm Evol. Comput.* **2024**, *87*, 101550. [CrossRef]