電気学会論文誌●（●●●●●●●部門誌）
IEEJ Transactions on ●●●●●●●●●●●●●●●●
Vol.●● No.● pp.●-●● DOI: ●.●●/ieejeiss.●●.●

**Paper**

# An improved adaptive NSGA-II for multi-objective comprehensive scheduling problem of flexible assembly job shop problem with AGVs

Haofan Yang[*a)] Non-member, Shigeru Fujimura[*] Senior Member

(Manuscript received Jan. 00, 20XX, revised May 00, 20XX)

This paper addresses the comprehensive scheduling problem in a flexible assembly job shop with AGV handling (FAJSP-AGVs), where machining of processes and assembly activities between different processes are conducted. Additionally, the handling time between machines is considered, and a multi-AGV system is introduced for process handling. The optimization objectives include minimizing the makespan, machine energy consumption, and AGV working time. A mathematical model for the FAJSP-AGVs is established based on these objectives. To solve this problem, an improved adaptive NSGA-II algorithm (IA-NSGA-II) is introduced, utilizing a process constraint matrix coding to satisfy assembly constraints. The mutation operation is adapted to the process constraint matrix to ensure chromosome legality during genetic operations. Furthermore, a variable neighborhood search (VNS) is incorporated to broaden the search space, enhancing the generation of high-quality solutions. Simulation experiments validate the effectiveness of the proposed algorithm. Additionally, considering the impact of dynamic events on the original scheduling scheme, this paper extends its research to dynamic rescheduling, aiming to minimize the makespan while ensuring stability. Through simulation experiments, the feasibility of the dynamic rescheduling strategy is verified.

**Keywords** : Flexible assembly job shop, Scheduling, Multi-objective, NSGA-II, Process constraint matrix, Automated Guided Vehicle

## 1. Introduction

Flexible Job Shop Scheduling Problem (FJSP), as an extension of the Job Shop Scheduling Problem (JSP), enhances the flexibility of machine selection, thereby improving the flexibility of job processing. This makes the scheduling model more aligned with actual production needs, rendering it a more challenging NP-hard problem compared to JSP [1].

The Flexible Assembly Job Shop Scheduling Problem (FAJSP) extends FJSP by considering the assembly constraints of product components. Optimizing the coordination between production processes and assembly processes is crucial in addressing the Flexible Assembly Shop Scheduling Problem.

In recent years, many researchers have studied extensions of the FAJSP to make it more like a real production system and have made great progress. Li Zihui et al. [2] proposed a hybrid distribution estimation algorithm to solve the Flexible Assembly Flow Shop Scheduling Problem with different processes, yielding promising results. Yang Xiaojia et al. [3] established a mathematical model for flexible assembly shop problem with stochastic disturbances, adopting dynamic scheduling strategies. They designed a genetic algorithm-based periodic rolling scheduling method to solve the model, demonstrating the superiority of the algorithm through extensive experiments. Wu et al. [4] conducted research on distributed flexible assembly shop, aiming to minimize both delay time and total cost through cooperative optimization. They proposed an improved differential evolution simulated annealing algorithm to solve the model. Zhang et al. [5] focused on preventive maintenance in assembly line scheduling, using completion time and maintenance time as optimization objectives. They proposed two heuristic algorithms and combined them with iterative greedy algorithms to solve the problem, validating the effectiveness of the algorithms. Ren et al. [6] proposed a heuristic algorithm combining particle swarm optimization and genetic algorithm to jointly optimize the production efficiency and energy consumption of flexible assembly shop.

While most studies consider the processing of jobs on machines, they often overlook the transportation of jobs between machines. With modern manufacturing transitioning towards intelligence, the movement of jobs between machines is predominantly handled by Automatic Guided Vehicles (AGVs). AGVs play a crucial role in the integrated scheduling and coordinated optimization of processing equipment and transportation resources in smart manufacturing systems. Hence, the integration of AGV scheduling with the FAJSP holds significant practical value in enhancing the efficiency of intelligent manufacturing systems.

Currently, there is limited research on the FAJSP with the involvement of AGVs, and there is no established benchmark dataset available. Most studies have focused on FJSP with AGV scheduling, assembly line scheduling, and distributed flow shop scheduling. The research objectives mainly revolve around four aspects: machine allocation, operation scheduling, AGV scheduling and allocation, and AGV path planning.

Rahman et al. [7] proposed a heuristic and meta-heuristic combined optimization approach for the assembly line balancing and AGV scheduling problem in the context of Industry 4.0. Through computational experiments, this method effectively reduced the cycle time and total delay of the assembly system, providing empirical support for the efficiency optimization of

a) Correspondence to: Haofan Yang. E-mail: yanghf1995@fuji.waseda.jp
* Graduate School of Information, Production, and Systems, Waseda University, 2-7 Hibikino, Wakamatsu, Kitakyushu, Fukuoka, 808-0135, Japan

intelligent assembly systems.

In the research on AGV scheduling in unbuffered assembly lines, Wang et al. [8] proposed a forward-looking scheduling strategy based on genetic algorithms by establishing a Petri net model. They effectively addressed the issue of processing sequence conflicts. Through simulation and robotic experiments, they demonstrated its effectiveness in reducing total processing time and improving the efficiency of AGVs and machines.

Karimi et al. [9] aimed to minimize the maximum completion time and established two mixed-integer linear programming models based on sequences and positions for the FJSP considering transportation time. Experimental results indicated the superiority of the sequence-based model. Dai et al. [10] studied the FJSP considering transportation time with objectives of minimizing the maximum completion time and total energy consumption. They transformed the bi-objective problem into a single-objective one using a weighted sum method and designed an improved genetic algorithm for solving it, revealing conflicting relationships between the two objectives. Peng et al. [11] investigated the FJSP with transportation time constraints, aiming to minimize processing energy consumption. They proposed a hybrid discrete multi-objective imperialist competitive algorithm for solving the problem and validated the effectiveness of the proposed algorithm through comparisons with other algorithms. Zhou et al. [12] conducted research on the FJSP considering crane transportation and devised a hybrid algorithm based on decomposed multi-objective evolutionary algorithm and particle swarm optimization to minimize the makespan and total energy consumption. Homayouni et al. [13] studied the FJSP considering transportation time with the objective of minimizing the makespan. They designed an operation-based multi-start biased random key genetic algorithm to solve the problem, demonstrating its effectiveness in addressing similar workshop scheduling problem. Yan et al. [14] established a model for finite transportation conditions in flexible job shop environments and designed an improved genetic algorithm for solving it by discussing the constraints imposed by finite transportation conditions.

The job shop scheduling problems are all static scheduling problems, without considering various uncertainties in actual production. The reality is that most situations faced by enterprises involve more complex dynamic scheduling problems, where there may be machine breakdowns, urgent rush orders, early deliveries, and other unexpected issues. Cao et al. [15] proposed an adaptive heterogeneous earliest finish time algorithm to address the objective of minimizing the makespan in the FJSP with arrival orders. Ozturk et al. [16] combined genetic programming with gene expression programming to propose two methods for the DFJSP. Hu et al. [17] introduced a novel Petri net-based dynamic scheduling approach, utilizing deep Q-network (DQN) and graph convolutional networks to solve dynamic scheduling problems in flexible manufacturing systems. We compare the existing papers with our work in the following Table 1.

In this study, a model of FAJSP with AGVs (FAJSP-AGVs) was established, and an improved adaptive NSGA-II [18] algorithm was proposed to optimize efficiency and green indicators. The main contributions of this paper include the following aspects: (1) A mathematical model for FAJSP-AGVs was established, with three objectives: minimizing the makespan, total machine energy consumption, and AGV working time. (2) An improved adaptive NSGA-II algorithm was proposed to optimize FAJSP-AGVs. (3)

Table 1. Comparisons of existing papers

| Publications | Objectives | Algorithms |
|---|---|---|
| Li Zihui et al. [2] | Makespan | Hybrid Estimation of Distribution Algorithm (HEDA) |
| Yang Xiaojia et al. [3] | Tardiness | GA |
| Wu et al. [4] | Makespan, earliness and tardiness | Improved differential evolution simulated annealing algorithm (IDESAA) |
| Zhang et al. [5] | Makespan and maintenance time | PM-based iterated greedy algorithm |
| Ren et al. [6] | Makespan and energy consumption | PSO-GA |
| Rahman et al. [7] | Assembly line cycle time | PSO |
| Wang et al. [8] | Makespan | look-ahead scheduling algorithm (LASA) |
| Dai et al. [10] | Makespan and energy consumption | Improved GA |
| Homayouni et al. [13] | Makespan | biased random key genetic algorithm (BRKGA) |
| Yan et al. [14] | Makespan | Improved GA |
| Cao et al. [15] | Makespan | Adaptive heterogeneous earliest finish time algorithm (A-HEFT) |
| Wu et al. [23] | Makespan and energy consumption | Improved NSGA-II |
| Current study | Makespan, energy consumption and AGV working time | Improved adaptive NSGA-II (IA- NSGA-II) |

Specialized encoding methods, adaptive crossover, and adaptive mutation operators were designed. (4) A variable neighborhood search strategy was integrated to avoid local optima. (5) Practical scenarios were considered, and a dynamic scheduling strategy for machine failures was proposed and validated.

## 2. Problem description and modeling

**2.1 Problem description** Flexible assembly job shop scheduling involves both machining and assembly aspects of the production process, where the final product is assembled from multiple workpieces in strict accordance with the product bill of materials (BOM). Within this, the processing of each part consists of a set of processes with sequential constraints. By combining the processing of the workpieces with the product BOM, a product manufacturing process tree, or process tree for short, is constructed. Fig. 1 shows the process of building a process tree for a single product, with blue circles that represent the process as an assembly process, white circles that represent the machining process, and black dashed boxes that represent the different workpieces. While the number inside the circle represents the process number, the number outside the dashed box represents the workpiece number. Therefore, as shown in the figure, the product consists of 6 workpieces which are machined and assembled, and it contains 10 processes in total.

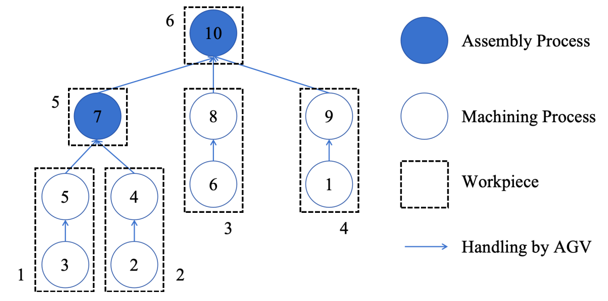


Fig. 1. Example of a process tree

Incorporating a multi-AGVs system into a flexible assembly shop forms the comprehensive scheduling problem with AGVs. AGVs are responsible for handling processes within the job shop. Initially, the sequence of process handling is determined based on the start times of processing processes. Subsequently, AGVs are selected based on their load status to handle the transportation of specific processes. The manufacturing process in the entire job shop comprises the machining and assembly stages of multiple machines, as well as the handling stages of multiple AGVs. The handling stages of each AGV affect the overall scheduling of the job shop.

Thus, FAJSP-AGVs can be described as follows: *n* workpieces are machined on *m* machines, and each workpiece is processed in several processes. These processes include machining and assembly, with constraints between them. Assembly processes can only begin after all parts involved in assembly have been processed. The machining and assembly processes can be performed on at least one machine, and the times for machining and assembly are different when different machines are used. Additionally, there are *r* AGVs responsible for handling processes between machines. The relevant symbols used in the paper are defined as shown in Table 2.

Table 2. Symbolic description of model parameters

| *Notations* | *Definition* |
|---|---|
| $n$ | Number of workpieces to be processed |
| $m$ | Number of available machines |
| $r$ | Number of AGVs |
| $s_i$ | Number of processes for workpiece *i* |
| $S_{i,j}$ | Start time of the process *j* of workpiece *i* |
| $S_{p,q,k}$ | Start time of the process *q* of workpiece *p* on machine *k* |
| $S_{i,j,k}$ | Start time of the process *j* of workpiece *i* on machine *k* |
| $F_{i,j,k}$ | Finish time of the process *j* of workpiece *i* on machine *k* |
| $C_i$ | Completion time of the workpiece *i* |
| $C_{max}$ | Total makespan |
| $L$ | An infinite number |
| $t_{i,j,k}$ | Processing time of the process *j* of workpiece *i* on machine *k* |
| $K_{i,j}$ | Start time of the handling the process *j* of workpiece *i* |
| $H_{i,j}$ | Finish time of the handling the process *j* of workpiece *i* |
| $H_{i,j,v}$ | Finish time of handling the process *j* of workpiece *i* by AGV *v* |
| $E_t$ | Total energy consumption |
| $E_{pro}$ | Energy consumption from production processes |
| $E_{st}$ | Idle time energy consumption |
| $T_v$ | Total AGV working time |
| $P^p_{i,j,k}$ | Energy consumption generated when the process *j* of workpiece *i* is produced on machine *k* |
| $P^s_k$ | Energy consumption generated by machine *k* idle time |
| $X_{i,j,k}$ | If the process *j* of workpiece *i* is processed on machine *k*, $X_{i,j,k} = 1$, otherwise $X_{i,j,k} = 0$ |
| $Y^k_{i,j,p,q}$ | If the process *j* of workpiece *i* starts before process *q* of workpiece *p* on machine *k*, $Y^k_{i,j,p,q}= 1$, otherwise $Y^k_{i,j,p,q}= 0$ |
| $V_{i,j,v}$ | If the process *j* of workpiece *i* is handled by AGV *v*, $V_{i,j,v}= 1$, otherwise $V_{i,j,v}= 0$ |
| $Z_{i,j,p,q,v}$ | If handling the process *j* of workpiece *i* by AGV *v* is before the process *q* of workpiece *p* by AGV *v*, $Z_{i,j,p,q,v}= 0$, otherwise $Z_{i,j,p,q,v}= 1$ |

**2.2 Modeling** We establish a FAJSP-AGVs model with minimizing the makespan, total machine energy consumption, and total AGV working time. The model is based on the following assumptions:

- The machining process is continuous without interruption.
- Each process can only be processed on one machine, and only one process can be processed at a time on the same machine.
- The process tree for all products is known, and the processing time for each process is fixed.
- Assembly processes can only start after all preceding production processing are completed.
- Preparation time and release time are not considered.
- At the initial time, all machines and AGVs are in the idle state.
- The entire process does not consider the loading and unloading time of AGVs.
- All AGVs have the same performance, and the speed of handling processes between all machining machines is the same.
- AGVs do not interfere with each other during the handling of processes, and collisions or other incidents do not occur.
- The issue of insufficient battery power for AGVs is not considered.

Based on the above assumptions, the FAJSP-AGVs model can be described as follows:

- Objective Functions:

a. Makespan

$$minimize\ C_{max} = max(\ C_i \mid i = 1, 2, \cdots, n\ ) \tag{1}$$

b. Total machine energy consumption

$$minimize\ E_t = E_{pro} + E_{st} \tag{2}$$

$$E_{pro} = \sum_{k=1}^{m} \sum_{i=1}^{n} \sum_{j=1}^{s_i} X_{i,j,k} \cdot t_{i,j,k} \cdot P^p_{i,j,k} \tag{3}$$

$$E_{st} = \sum_{k=1}^{m} \left(C_{max} - \sum_{i=1}^{n} \sum_{j=1}^{s_i} t_{i,j,k} \cdot X_{i,j,k}\right) \cdot P^s_k \tag{4}$$

c. Total AGV working time

$$minimize\ T_v = \sum_{i=1}^{n} \sum_{j=1}^{s_i} \left(\ H_{i,j} - K_{i,j}\ \right) \tag{5}$$

- Constraints:

$$S_{i,j} \geq 0 \quad \forall i = 1, 2, \cdots, n \quad \forall j = 1, 2, \cdots, s_i \tag{6}$$

$$\begin{gathered} S_{p,q,k} \geq S_{i,j,k} + t_{i,j,k} - L \cdot \left(1 - Y^k_{i,j,p,q}\right) \\ \forall i, p = 1, 2, \cdots, n \quad \forall j, q = 1, 2, \cdots, s_i \\ \forall k = 1, 2, \cdots, m \end{gathered} \tag{7}$$

$$\begin{gathered} F_{i,j,k} \geq S_{i,j,k} + t_{i,j,k} \cdot X_{i,j,k} \\ \forall i = 1, 2, \cdots, n \quad \forall j = 1, 2, \cdots, s_i \\ \forall k = 1, 2, \cdots, m \end{gathered} \tag{8}$$

$$\begin{gathered} \sum_{v=1}^{r} V_{i,j,v} = 1 \\ \forall i = 1,2,\cdots,n \quad \forall j = 1,2,\cdots,s_i \quad \forall v = 1,2,\cdots,r \end{gathered} \tag{9}$$

$$\begin{gathered} V_{i,j,v} \cdot V_{p,q,v} \cdot H_{i,j,v} \leq K_{p,q} + Z_{i,j,p,q,v} \cdot L \\ \forall i,p = 1,2,\cdots,n \quad \forall j,q = 1,2,\cdots,s_i \quad \forall v = 1,2,\cdots,r \end{gathered} \tag{10}$$

$$H_{i,j} \le S_{i,j} \qquad \forall i = 1,2,\cdots,n \quad \forall j = 1,2,\cdots,s_i \tag{11}$$

$$F_{i,j,k} \le S_{p,q,k} + L\cdot\left(1 - Y^{k}_{i,j,p,q}\right) \qquad \forall i,p = 1,\ 2,\ \cdots,\ n \quad \forall j,\ q = 1,\ 2,\ \cdots,\ s_i \quad \forall k = 1,\ 2,\ \cdots,m \tag{12}$$

Where, formula (1) represents the objective function, which is to minimize the makespan; Formula (2) represents the total energy consumption; Formula (3) represents the machining energy consumption; Formula (4) represents the machine idle energy consumption; Formula (5) represents the AGV working time, which is composed of two parts: unloaded movement and loaded handling; Constraint (6) ensures that the start time of every process is non-negative, indicating that all tasks must begin at a valid time point on the timeline; Constraint (7) ensures that processes processed on the same machine follow the required sequence; Constraint (8) represents the temporal constraints, ensuring that the completion time of a process must always be greater than or equal to its start time plus its processing duration; Constraint (9) represents the constraint on the selection of AGVs for handling processes, where each handling process can only choose one AGV from all available AGVs for processing before the process; Constraint (10) represents the constraint on the number of processes handled by an AGV at the same time, where an AGV can only handle one process at the same time, and it can only start a new handling process after completing the previous one; Constraints (11) and (12) represent the constraints on the start time of processes, where an process can only start after the handling process of its preceding process on the same workpiece and the machining process of its preceding process on the same machine are completed.

## 3. Designed improved adaptive NSGA-II

The NSGA-II algorithm is a classic multi-objective optimization algorithm. However, it cannot be directly applied to this problem and has some drawbacks. For instance, in solving FAJSP-AGVs, the algorithm may generate infeasible solutions during the initialization of the population and genetic operations due to assembly constraints. Additionally, it is prone to getting trapped in local optima during the iterative process. Therefore, to effectively address the FAJSP-AGVs, an improved adaptive NSGA-II algorithm (IA-NSGA-II) is proposed. (1) The algorithm's encoding, crossover, and mutation methods are tailored to the problem's characteristics. (2) Using improved adaptive crossover and mutation probabilities to further assist the algorithm in improving convergence. (3) A variable neighborhood search method is incorporated into the algorithm to further expand the solution space, thereby avoiding getting stuck in local optima.

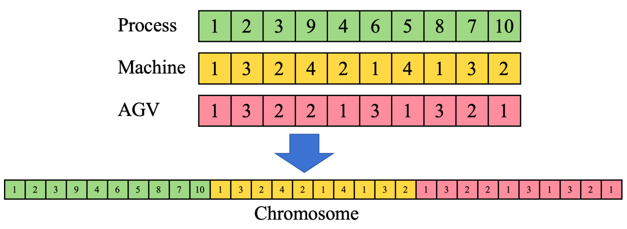


Fig. 2. Model of chromosome coding

**3.1 Encoding Method** Considering the difficulties highlighted by Blazewicz et al. [19] regarding the direct mapping of binary encoding, which is often considered unsuitable for solving combinatorial optimization problems, this paper extends the widely used two-layer integer coding method in scheduling problems. We further extend it by incorporating AGV numbering as the third part of the chromosome segment, forming a triple-layer coding. The encoded chromosome model is illustrated in Fig. 2.

The chromosome is divided into three parts using the triple-layer encoding method: the process sequence part, the machine selection part, and the AGV selection part. These parts correspond exactly to the three sub-problems that need to be solved in the FAJSP-AGVs, and when combined, they form a complete chromosome. The triple-layer encoding method is simple and efficient, providing an intuitive representation of the solution. Furthermore, this method enhances the flexibility of chromosome representation, making it easier to adapt to complex constraints. Literature [20, 21] has also validated the effectiveness of multi-layer encoding methods in solving scheduling problems in flexible manufacturing facilities.

*3.1.1 Process Sequence Encoding Method* The encoding method for the process sequence part uses the process constraint matrix. Since *0-1* programming is a unique type of integration program design, the decision variables can only take the values *0* or *1*. *0-1* variables have the advantage of quantitatively describing the relationships between discrete variables in terms of logical relationships, the order relationships, and the constraints of mutuality.

In this paper, we utilize its advantage in describing processing relationships to establish a forward process constraint matrix. Taking the tree structure shown in Fig. 1 as an example, the composition rule of the process constraint matrix is: if the value of $n_{xy}$ in matrix $H$ is 0, then process $O_j$ has no constraint on process $O_q$; if the value of $n_{xy}$ in matrix $H$ is 1, then process $O_j$ has constraint on process $O_q$. If the value of $n_i$ in the matrix $H$ is 0, it means that process $O_j$ is the schedulable process at this moment.

The specific method for generating process part chromosomes based on the process constraint matrix is as follows:

By calculating the sum of each column in the matrix, the optional processes with a sum of 0 are filtered out, and these processes represent the absence of pre-process constraints. Randomly selecting one of the optional processes as the first gene of the process chromosome and deleting the rows and columns of the matrix where the process is located. This operation ensures that the process is released from the processing constraints imposed by subsequent processes and avoids potential deadlocks. According to the above method, the processes are selected one by one to form a complete process chromosome. Using this method can effectively ensure the legality of the chromosome and prevent the generation of a chromosome that does not meet the constraints of the previous process, thereby enhancing the feasibility of the genetic operation. The process part chromosome coding for the above product is shown in Fig. 3.

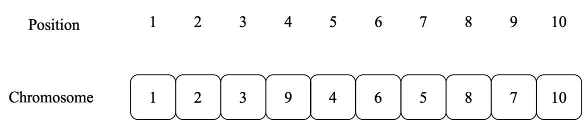


Fig. 3. Illustration of the process sequence part

*3.1.2 Machine Selection Encoding Method* For the encoding of the machine selection part, the chromosome follows a coding rule based on the number of machine positions. The gene sequence on the chromosome contains information about the selected processing machines for each process. Here, the genes on the chromosome do not directly represent the sequence number of the processing machine chosen for the process at that position. Instead, they correspond to the number of locations in the set of machines available for each process. The machine selection part chromosome coding for the above product is shown in Fig. 4.

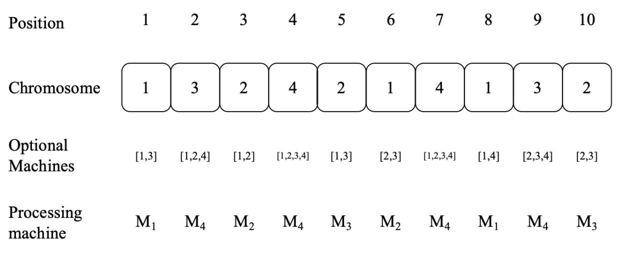


Fig. 4. Illustration of the machine selection part

*3.1.3 AGV Selection Encoding Method* The chromosome of AGV selection part is relatively straightforward. Since all AGVs are available for selection, AGV numbering is directly used for encoding. The gene sequence on the chromosome represents the AGV number responsible for handling each process. The AGV selection part chromosome coding for the above product is shown in Fig. 5.

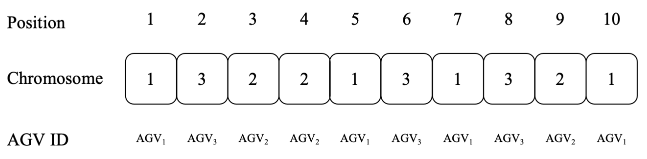


Fig. 5. Illustration of the AGV selection part

Take the triple-layer coding in Fig. 2 as an example, process *1* is the first process to be processed, the processing machine is $M_1$ and is handled by $AGV_1$; process *2* is the second process to be processed, the processing machine is $M_4$ and is handled by $AGV_3$. And so on can be completely interpreted the meaning of a chromosome.

**3.2 Genetic operations** In addition to the fact that chromosome encoding needs to satisfy the assembly constraint characteristic of FAJSP-AGVs, the algorithm still needs to be improved for this characteristic when it performs genetic operations, otherwise a large number of infeasible chromosomes will be generated during the iteration process, which will affect the quality of the population as well as the convergence of the algorithm.

Wu et al. [22] also uses the NSGA-II and improved it. The main areas of improvement are the use of the precedence-preserving order-based crossover operator [23] and gene insertion mutation operator [24] to achieve that the algorithm will not generate infeasible solutions when performing genetic operations. However, he did not consider the typical problem that NSGA-II is easily trapped in local optimum. In this paper, the genetic operator is improved with the same purpose and further improved for the problem of local optimum. The crossover and mutation probabilities are optimized to adaptive probability, and the solution space is further expanded by combining the variable neighborhood search method to avoid the algorithm falling into local optimum.

*3.2.1 Crossover* For chromosome in process sequence part, using the improved two-point crossover method: Fig. 6 illustrates the crossover process. Parent 1 and Parent 2 are the two parent chromosomes paired for crossover. Two crossover points are randomly generated, dividing both chromosomes into three parts: the head, the middle, and the tail. Then, taking out the middle part of Parent 1 and searching for the order of the genes in the middle part of Parent 1 in Parent 2. The genes in the middle part of Parent 1 are then replaced according to this order, resulting in a new chromosome. The same process applies to Parent 2, resulting in two new chromosomes.

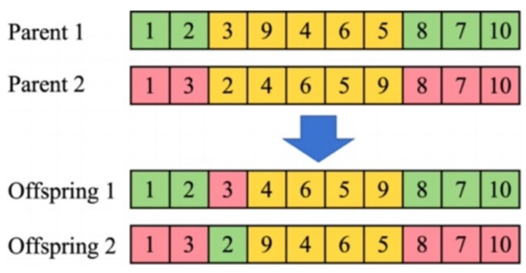
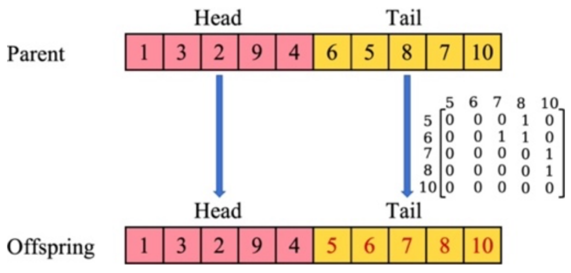


Fig. 6. Schematic diagram of crossover operation

For the chromosomes in the machine selection part and AGV selection part, since they don't have special order constraints that would create infeasible chromosomes due to crossover, I chose to use a normal two-point crossover. That is, two gene locations are randomly selected on the parent chromosomes, and the gene segments between the two points are exchanged to get two new offspring chromosomes.

*3.2.2 Mutation* For chromosome in process sequence part, a mutation point is randomly generated to determine the location of the mutation. Then, the head of the selected parent is placed at the head of the offspring after mutation, and the mutation operator reconstructs the tail of the offspring using the tail of the parent.

As shown in Fig. 7, this process is implemented by randomly generating a mutation point. In this case, the mutation point is 5. The digits in the gene sequence [1:5] are all copied to the same positions in the offspring individual. Then, the digits in the gene sequence [6:10] are regenerated according to the process constraint matrix. At this point, all rows and columns corresponding to the digits in the gene sequence [1:5] in the process constraint matrix are deleted, resulting in a new partial process constraint matrix. Based on this partial process constraint matrix, a random encoding can be obtained using the initial population generation method, which satisfies the constraints of priority relationships.



$$\begin{array}{c|ccccc} & 5 & 6 & 7 & 8 & 10 \\ \hline 5 & 0 & 0 & 0 & 1 & 0 \\ 6 & 0 & 0 & 1 & 1 & 0 \\ 7 & 0 & 0 & 0 & 0 & 1 \\ 8 & 0 & 0 & 0 & 0 & 1 \\ 10 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Fig. 7. Schematic diagram of mutation operation

A single-point mutation method was employed for the machine selection part and AGV selection part chromosomes. For the machine selection part, the mutation involved randomly selecting a gene position on the chromosome and replacing the assigned machine with one of the alternative machines available for the process. For the AGV selection part, the mutation involved randomly selecting a gene position on the chromosome and replacing the original AGV ID with an alternative AGV ID.

Through the above improvements in genetic operations, the validity of chromosomes is always ensured, preventing the generation of infeasible solutions and thus maintaining the efficiency of the algorithm.

*3.2.3 Adaptive Crossover and Mutation Probability* The crossover and mutation are the primary operations of global and local search in NSGA-II, which significantly determine the algorithm's performance. Typically, in the early stages of evolution, when the quality of the initial population is poor, higher crossover and mutation probabilities are required to explore the solution space as much as possible and find more advantageous solutions. As the evolution progresses and the quality of the population improves to some extent, smaller crossover and mutation probabilities are needed to reduce the disruption of the parental information of advantageous individuals and accelerate the convergence speed. However, the commonly used adaptive genetic operations described above provide only a rough description of the genetic operations probability requirements at different evolutionary stages of the population. To further improve the adaptive efficiency, this paper not only considers the requirements of crossover and mutation probabilities at different iterations of the algorithm but also integrates the non-dominant rank situation of crossover and mutation objects at different evolutionary stages to more specifically reflect the evolution of population individuals, and thus can make the algorithm's searching ability to be effectively improved. The formulas for calculating the adaptive genetic operations probability $P_{ac}$ and $P_{am}$ are as follows:

$$P_{ac} = 0.8 + \alpha_{ci} e^{-\alpha gen_i} + \alpha\, ln(1 + \delta_r rak_i) \tag{13}$$

$$P_{am} = 0.03 + \alpha_{mi} e^{-\alpha gen_i} + \alpha ln(1 + \delta_r rak_i) \tag{14}$$

In the formula, the parameters $\alpha_{ci}$ and $\alpha_{mi}$ represent the crossover probability control coefficient and the mutation probability control coefficient, respectively. $\alpha$ and $\delta_r$ are positive parameters, and $i$ indicates the current iteration number of the population. $rak_i$ represents the non-dominant rank of the crossover or mutation objects under the current iteration number. Specifically, during crossover, $rak_i$ represents the smaller non-dominant rank among the two selected individuals. If they are equal, one is randomly chosen. During mutation, $rak_i$ represents the non-dominant rank of the selected individual itself.

**3.3 Variable Neighborhood Search** The variable neighborhood search method (VNS) systematically switches between different neighborhood structures, leading to a wider search scope and higher solution quality compared to single-neighborhood search algorithms. It prevents the search from being trapped in local optima and enhances local search capability [25]. The basic idea of the VNS is to systematically change multiple neighborhood structures within a local range of a solution. It mainly has two loops: an outer loop for changing the neighborhood structure and conducting local searches within several different neighborhood structures, and an inner loop for searching within a specific neighborhood structure. By adopting the "greedy acceptance" strategy, the algorithm obtains new solutions that are better than the original one and replaces inferior solutions iteratively until termination, thereby obtaining a local optimal solution.

To incorporate the variable neighborhood search algorithm into improved NSGA-II and enhance the algorithm's local search capability, it is necessary to design multiple neighborhood structures. In this paper, three types of neighborhood structures are designed from three perspectives:

$N_1$: This is a neighborhood structure involving the movement of two processes. Compared to other common neighborhood structures, this structure is smaller and more efficient. The basic idea is to select chromosome from the population with the highest non-dominated ranks and randomly swap the order of two processes. The specific structure is shown in Fig. 8.

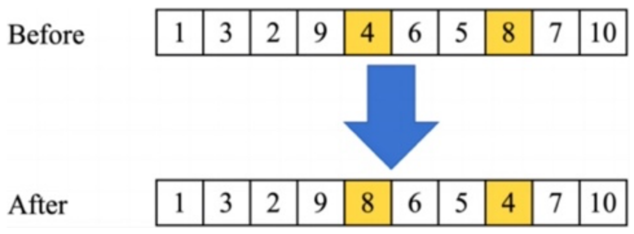


Fig. 8. Neighborhood structure of two-opt swap

$N_2$: This is a neighborhood structure involving the random transformation of processing machines. The basic idea is to select chromosome from the population with the highest non-dominated ranks and randomly change the processing machine for one process. The specific structure is shown in Fig. 9.

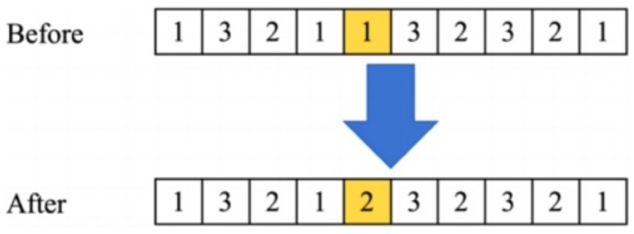


Fig. 9. Neighborhood structure of machine change

$N_3$: This is a neighborhood structure involving the random transformation of AGVs. The basic idea is to select chromosome from the population with the highest non-dominated ranks and randomly change one AGV responsible for handling process. The specific structure is shown in Fig. 10.

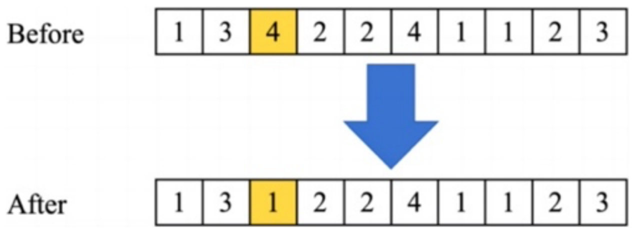


Fig. 10. Neighborhood structure of AGV change

These neighborhood structures aim to diversify the search space and enhance the local search capability of the improved adaptive NSGA-II algorithm when solving the FAJSP-AGVs.

**3.4 Rescheduling strategy** Production rescheduling [26] in the job shop refers to the process of rearranging scheduling plans when various disturbances occur during the processing, rendering the original schedule infeasible. In response to disruptive events, it is crucial to determine when and how to carry out rescheduling. This paper adopts an event-driven rescheduling strategy and employs a complete rescheduling approach to address the re-scheduling caused by machine failures during the machining process. This ensures the generation of a relatively appropriate rescheduling plan following a machine breakdown. Given the characteristic that interruptions other than machine failures are not allowed during the machining process, the specific steps of the complete rescheduling strategy are designed as follows.

*3.4.1 Complete rescheduling* Complete rescheduling refers to the process where, upon the occurrence of dynamic events, the system immediately reorders, reassigns machines, and reallocates AGVs for all remaining processes, generating a new scheduling plan. Complete rescheduling considers the entire system globally, ensuring the optimization of the scheduling plan.

*3.4.2 Steps of complete rescheduling* The basic steps of complete rescheduling are as follows.

**Step 1:** Determine the time node for rescheduling and the set of processes that need to be rescheduled. The set of processes requiring rescheduling includes those that cannot be completed before the rescheduling time node and those interrupted due to machine breakdown.

**Step 2:** Determine the start time for each machine, with the start time for faulty machines set as the estimated repair time.

**Step 3:** Utilize the IA-NSGA-II algorithm to reschedule the processes that need to be rescheduled.

*3.4.3 Performance metrics based on scheduling stability*

Performance metrics based on scheduling stability are commonly evaluated by the degree of delay (*DE*), which refers to two efficiency metrics of the rescheduling scenario and the initial scheduling scenario. That is, the difference between the makespan and the total AGV working time. The calculation formula is shown in equation (16).

$$DE=1+\mu_1\cdot\left(C^*_{max}-C_{max}\right)/C_{max}+\mu_2\cdot\left(L^*_{max}-L_{max}\right)/L_{max} \qquad (15)$$

where $C_{max}$ denotes the makespan of the initial scheduling scheme, $C^*_{max}$ denotes the makespan of the rescheduling scheme, $L_{max}$ denotes the total AGV working time of the initial scheduling scheme, $L^*_{max}$ denotes the total AGV working time of the rescheduling scheme, $\mu_1$ and $\mu_2$ denote the weight coefficients, and $\mu_1+\mu_2=1$.

## 4. Experimental Results

To evaluate the feasibility and effectiveness of the proposed algorithm, several experiments were conducted, including self-validation and comparative analysis with existing algorithms. The experiments aimed to evaluate the algorithm's ability to generate high-quality schedules for flexible assembly job shop.

**4.1 Parameter settings** Currently, there are many standard benchmark instances available for traditional JSP and FJSP. However, no established standard benchmark instance library has been created or fully standardized for FAJSP. To verify the feasibility of the proposed method, existing benchmark instances from the reference Wu et al. [22] are selected for testing. For ease of presentation and clarity, the improved adaptive NSGA-II scheduling method introduced in this paper is referred to as "IA-NSGA-II", while the original scheduling method proposed in the literature is referred to as "INSGA-II." The parameters related to the algorithm are set as follows: the initial population size is 100, the number of genetic generations is 100, and the implementation is carried out using the MATLAB programming language. For each instance, the solution is obtained independently through 10 runs to ensure reliability and consistency.

Then, in the context of FAJSP-AGVs, instances cited from Wu et al. [27] are chosen for testing. The scheduling method proposed in the literature is referred to as "IGA." The obtained results will be compared with those obtained using both IGA and NSGA-II. Algorithm-related parameters are set as follows: the initial population size is 100, the number of genetic generations is 100, and the implementation is done using the MATLAB programming language. For each instance, the solution is independently obtained through 20 runs.

**4.2 Experiment 1** There are three objectives to be optimized in this paper, and in Experiment 1, the better results were obtained for both the makespan and total energy consumption metrics. Fig. 11 shows the structure of the two identical products used for testing. Table 3 demonstrates the specific process processing times and energy consumption.

The NSGA-II algorithm has been improved in the literature, and six test results have been calculated using the designed test cases. The metrics used for evaluation include tardiness and total machine energy consumption. In this specific context, the tardiness is defined as the excess of the makespan in completing the product over the specified product completion time. The two products considered in the study have different specified completion times, which are set as 120 and 110, respectively. Therefore, this study conducts comparative experiments using the same performance metrics for consistency. Specifically, this paper investigates and compares the differences in total machine energy consumption observed under identical tardiness conditions.

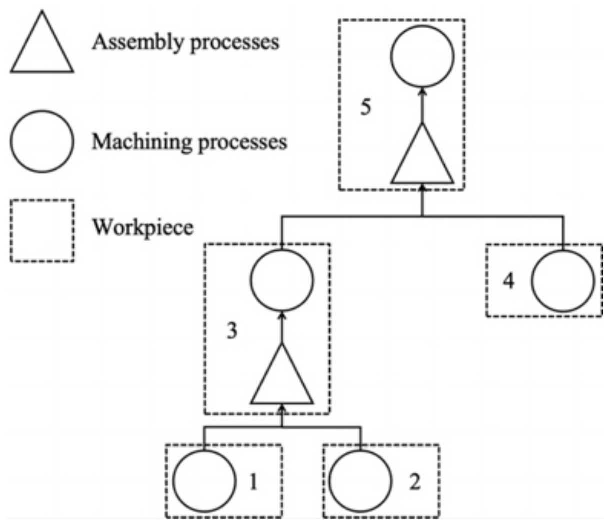


Fig. 11. Product structure for experiment 1

Table 3. Detail of process processing times and energy consumption

| Index of workpiece | Processes | Processing time, processing energy consumption per unit | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_5$ | $M_6$ | $M_7$ |
| 1 | *1* | — | — | — | — | 10,43 | 7,57 | — |
| | *2* | 9,48 | 8,58 | — | — | — | — | — |
| | *3* | — | — | — | — | 7,53 | 7,57 | — |
| | *4* | — | — | 8,56 | 5,56 | — | — | — |
| | *5* | — | — | — | — | 8,56 | 6,60 | — |
| 2 | *6* | — | — | 6,53 | 10,42 | — | — | — |
| | *7* | — | — | 7,42 | 7,52 | — | — | — |
| | *8* | — | — | — | — | 8,57 | 10,51 | — |
| | *9* | 5,53 | 9,43 | — | — | — | — | — |
| 3 | *10* | — | — | — | — | — | — | 8,52 |
| | *11* | — | — | — | — | 7,54 | 7,60 | — |
| | *12* | 6,57 | 10,53 | — | — | — | — | — |
| | *13* | — | — | 10,46 | 9,55 | — | — | — |
| | *14* | — | — | 8,49 | 6,49 | — | — | — |
| | *15* | 5,52 | 7,47 | — | — | — | — | — |
| 4 | *16* | — | — | 6,60 | 10,52 | — | — | — |
| | *17* | — | — | 7,58 | 7,50 | — | — | — |
| | *18* | — | — | 7,49 | 5,60 | — | — | — |
| | *19* | — | — | — | — | 5,59 | 9,47 | — |
| | *20* | — | — | 6,42 | 6,57 | — | — | — |
| | *21* | 6,53 | 8,57 | — | — | — | — | — |
| 5 | *22* | — | — | — | — | — | — | 5,45 |
| | *23* | 6,45 | 10,46 | — | — | — | — | — |
| | *24* | 5,52 | 6,59 | — | — | — | — | — |
| | *25* | — | — | 10,53 | 5,42 | — | — | — |
| | *26* | — | — | — | — | 10,48 | 9,57 | — |

Fig. 12 illustrates the comparison results of total machine energy consumption using IA-NSGA-II and INSGA-II for the same tardiness. From Fig. 12, at the same tardiness, all the results obtained using IA-NSGA-II proposed in this paper dominate the results obtained using INSGA-II.

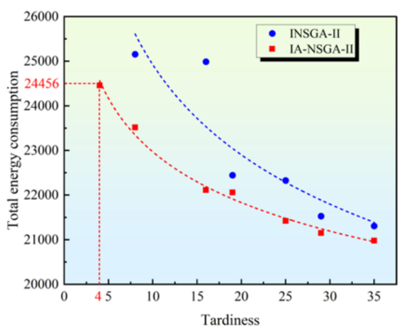


Fig. 12. Comparison results by using IA-NSGA-II and INSGA-II

Furthermore, as shown in Fig. 12, by using the proposed IA-NSGA-II, better result was obtained in terms of tardiness. The total tardiness is 4 and the total machine energy consumption is 24456. Although this paper has achieved better results by improving NSGA-II, I believe there is still room for further improvement. While Wu et al. [22] ensures the legality of the chromosomes, it tends to get stuck in local optima. The IA-NSGA-II proposed in this paper effectively alleviates this issue. However, the algorithm's current search process still exhibits some randomness, which will be the focus of my future improvements.

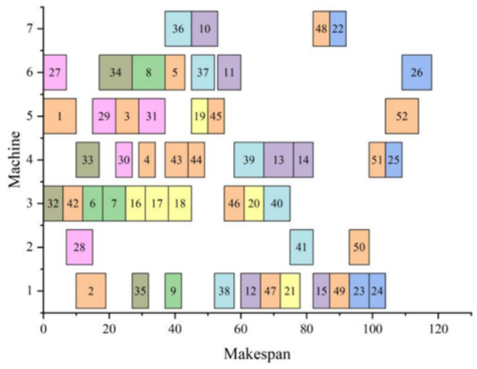


Fig. 13. Gantt chart generated using IA-NSGA-II

Fig. 13 illustrates the Gantt chart obtained using IA-NSGA-II with the total tardiness of 4. Each box in the figure represents a process sequence. In this experiment, two identical products are subject to processing, each comprising 26 processes. Consequently, *Product 1* concludes its processing upon completion of process 26, while *Product 2* is finished after process 52. Notably, the makespan for completing *Product 1*'s processing stands at 118, whereas for *Product 2*, it is 114. Remarkably, *Product 1*'s processing is completed prior to the designated deadline, whereas *Product 2* exceeds the deadline by 4 units of time. Meanwhile, employing INSGA-II from Wu et al. [22] yields makespan of 122 and 116 for *Product 1* and *2*, respectively, along with total tardiness of 8. These results demonstrate that using IA-NSGA-II leads to better scheduling outcomes.

These results show that the proposed method can obtain a better solution when solving the FAJSP, thus validating the feasibility and effectiveness of the method.

**4.3 Experiment 2** In the experiment 1, the feasibility and performance improvement of IA-NSGA-II in solving the FAJSP have been validated. The experiment 2 compared the proposed method with IGA proposed in Wu et al. [27] and NSGA-II to verify its advantages in addressing the FAJSP-AGVs. Better results were obtained in Experiment 2 for both the makespan and AGV working time metrics.

Fig. 14 shows the process tree of the flexible assembly job shop. In this experiment, three AGVs are set up. Table 4 shows the handling times of processes between various processing machines by the AGVs, and Table 5 lists the processing times for each process on each machine.

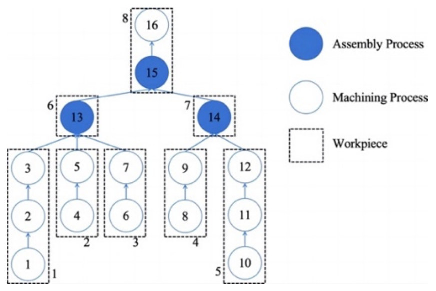


Fig. 14. Product structure for experiment 2

Although the Wu et al. [27] introduced a multi-AGV system for workpiece handling, it improved a genetic algorithm commonly used for solving single-objective problems. To address the dual objectives of minimizing makespan and AGV working time, it introduced weight coefficients. However, the experimental results show that there is still some room for optimizing the effectiveness of this approach when dealing with multi-objective problem.

Table 4. AGV handling time between machines

| | S\E | $M_1$ | $M_2$ | $M_3$ | $M_4$ |
|---|---|---|---|---|---|
| S\E | 0 | 6 | 5 | 4 | 8 |
| $M_1$ | 6 | 0 | 7 | 3 | 6 |
| $M_2$ | 5 | 7 | 0 | 6 | 5 |
| $M_3$ | 4 | 3 | 6 | 0 | 7 |
| $M_4$ | 8 | 6 | 5 | 7 | 0 |

Fig. 15 shows the comparison results and distribution of data for makespan after 20 searches among IA-NSGA-II, NSGA-II, and IGA. Fig. 16 is the comparison results and distribution of data for AGV working time. In the figure, the box represents the range where 50% of the data falls, the horizontal line in the middle of the box represents the median, and the hollow square represents the mean. Since there are no outliers in this experiment, the upper and lower whiskers of the box represent the maximum and minimum values, respectively. Table 6 and Table 7 shows the specific data for each part of the box plot.

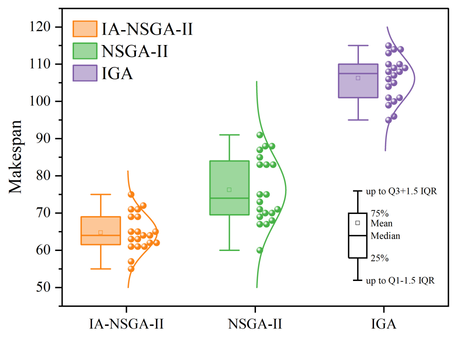


Fig. 15. Box plot of makespan and distribution of data

Table 5. Detail of process processing times

| Index of workpiece | Processes | Processing time | | | |
|---|---|---|---|---|---|
| | | $M_1$ | $M_2$ | $M_3$ | $M_4$ |
| 1 | *1* | 8 | – | 14 | 11 |
| | *2* | 7 | 10 | – | 7 |
| | *3* | 12 | – | 8 | 5 |
| 2 | *4* | – | 7 | 7 | 8 |
| | *5* | – | 9 | 4 | 6 |
| 3 | *6* | 9 | 8 | – | 10 |
| | *7* | 4 | – | 15 | – |
| 4 | *8* | 6 | – | 2 | 15 |
| | *9* | 8 | 13 | 8 | – |
| 5 | *10* | – | 10 | 2 | 7 |
| | *11* | 4 | 5 | 9 | – |
| | *12* | – | 9 | 9 | 6 |
| 6 | *13* | 7 | 3 | – | 10 |
| 7 | *14* | 4 | 5 | – | 9 |
| 8 | *15* | – | 6 | 3 | 4 |
| | *16* | – | – | 6 | 4 |

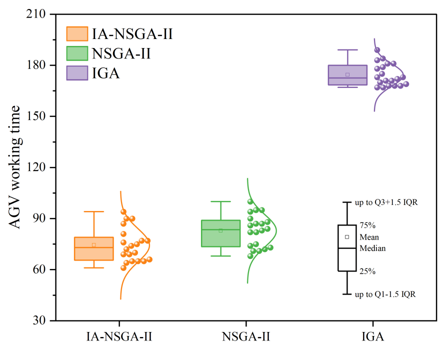


Fig. 16. Box plot of AGV working time and distribution of data

Table 6. Specific data for makespan box plot

| Box plot structure | *IA-NSGA-II* | *NSGA-II* | *IGA* |
|---|---|---|---|
| Lower whisker | 55 | 60 | 95 |
| Lower end of the box | 61.5 | 69.5 | 101 |
| Median | 64 | 74 | 107.5 |
| Mean | 65.65 | 76.2 | 106.2 |
| Upper end of the box | 70 | 84 | 110 |
| Upper whisker | 83 | 91 | 115 |

Table 7. Specific data for AGV working time box plot

| Box plot structure | *IA-NSGA-II* | *NSGA-II* | *IGA* |
|---|---|---|---|
| Lower whisker | 61 | 68 | 167 |
| Lower end of the box | 65.5 | 73.5 | 168.5 |
| Median | 73 | 83.5 | 172.5 |
| Mean | 74.35 | 82.85 | 174.35 |
| Upper end of the box | 79 | 89 | 180 |
| Upper whisker | 94 | 100 | 189 |

From Fig. 15 and Table 6, the box plot for IA-NSGA-II is the lowest, with the median less than the mean, and it achieved the minimum value of 55. This indicates that the algorithm not only produces better results but also maintains high overall data quality. In contrast, the median of the box plot for IGA is higher than the mean, indicating that the results obtained by IGA are relatively larger.

The distribution curves on the right side provide a more intuitive view: the results of IA-NSGA-II are mostly clustered near the minimum value, with only a few points above, and the curve is skewed downward. This demonstrates the superior optimization performance of the proposed method. The curve for NSGA-II is relatively smooth, resembling a normal distribution, whereas the curve for IGA is more peaked, showing that only a small number of IGA results are close to the minimum value.

Fig. 16 shows the comparison of AGV working times after 20 searches among IA-NSGA-II, NSGA-II, and IGA. Table 7 presents the specific data for each box plot. From the figure, it can be seen that IA-NSGA-II also achieved better results compared to the other two algorithms, not only obtaining the minimum value of 61 but also showing that the points on the curve are distributed closer to the minimum value, indicating higher quality results obtained by the proposed method.

Fig. 17 shows a comparison of Gantt charts for the optimal makespan solutions obtained using IA-NSGA-II, NSGA-II, and IGA. Unlike Fig. 13 from Experiment 1, this experiment includes time blocks describing the working states of AGVs in the Gantt chart, as AGVs are responsible for handling process between machines. In the Fig. 17, $R_1$ to $R_3$ represent the AGV numbers, and $M_1$ to $M_4$ represent the machine numbers. The hollow time blocks for AGVs indicate unloaded movement, while the colored time blocks show AGVs handling processes. Each color corresponds to a specific process, with the starting position and target machine marked at the lower left and upper right corners of the time block, respectively. The number in the middle represents the process being handled by the AGV. For example, in Fig. 17(a), the first blue time block for $R_1$ represents $AGV_1$ moving process 10 from start point to *Machine 2*.

From the figure, the result obtained using IA-NSGA-II is better than those obtained using NSGA-II and IGA. This is because NSGA-II tends to get trapped in local optima, preventing it from finding the optimal solution, while IGA is not effectively designed to handle dual-objective problems, leading to multiple changes in processing machines for processes. This requires AGVs to perform multiple movements to handling processes. Since the start time of a process is determined by the AGV handling time, both the makespan and AGV working time are significantly increased. This validates that IGA is not suitable for solving the multi-objective FAJSP-AGVs. The IA-NSGA-II proposed in this paper not only achieves the smallest makespan in the Gantt chart but also shows that each time block is relatively more compact and the number of machine changes for processes is reduced, thereby alleviating AGV working time.

From the above results, the IA-NSGA-II proposed in this paper achieves better results in solving the FAJSP-AGVs. However, since there is currently no standard dataset for such problems, it is challenging to verify the effectiveness of IA-NSGA-II across different scenario scales. In the future, I will also attempt to establish larger datasets to test the scalability of the proposed method.

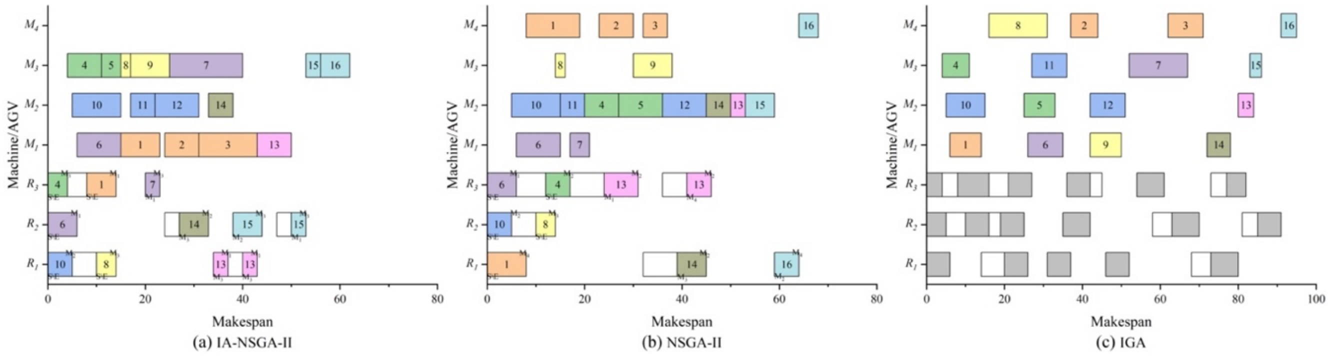


Fig. 17. Comparison of Gantt charts using IA-NSGA-II, NSGA-II, and IGA

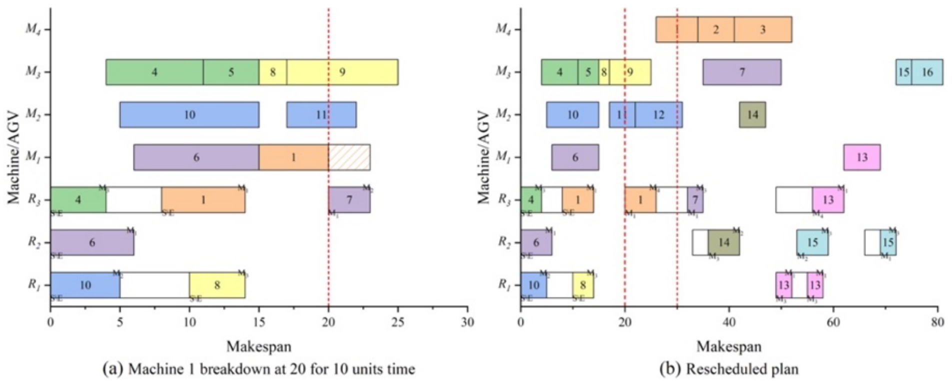


Fig. 18. Gantt charts before and after rescheduling

**4.4 Experiment 3** When a machine fails during production, the machine cannot process any processes until the repair is completed, and scheduling tasks for this machine must wait until the repair is finished. In this experiment, Machine 1 is set to fail during processing, with failure times set to 5 unit times, 10 unit times, 15 unit times, and 20 unit times. Upon failure, complete rescheduling of the remaining processes is performed based on the duration of the failure. Based on the 20 scheduling results obtained from Experiment 2, three sets of results were randomly selected for rescheduling experiments to test the stability of IA-NSGA-II in dynamic scheduling. The objective metric used is the delay degree DE as indicated in Equation (15), where both $\mu_1$ and $\mu_2$ are set to 0.5.

Table 8. Rescheduling results and degree of delay

| *Index of result* | $C_{max}$ | $L_{max}$ | $T_B$ | $C^*_{max}$ | $L^*_{max}$ | *DE* |
|---|---|---|---|---|---|---|
| 1 | 62 | 70 | 5 | 74 | 79 | 1.16 |
| | | | 10 | 81 | 90 | 1.29 |
| | | | 15 | 87 | 94 | 1.37 |
| | | | 20 | 93 | 106 | 1.51 |
| 2 | 65 | 81 | 5 | 75 | 88 | 1.12 |
| | | | 10 | 81 | 97 | 1.22 |
| | | | 15 | 88 | 101 | 1.3 |
| | | | 20 | 99 | 111 | 1.45 |
| 3 | 72 | 90 | 5 | 79 | 99 | 1.1 |
| | | | 10 | 91 | 113 | 1.26 |
| | | | 15 | 97 | 118 | 1.33 |
| | | | 20 | 104 | 116 | 1.36 |

Table 8 presents the test results after rescheduling using IA-NSGA-II. Here, $T_B$ represents the machine failure duration, $C_{max}$ and $C^*_{max}$ represent the makespan for the initial schedule and the rescheduled plan, respectively. $L_{max}$ and $L^*_{max}$ represent the AGV working time for the initial schedule and the rescheduled plan, respectively, and *DE* denotes the delay degree.

As seen from the table, the makespan increases with the duration of the failure. After a failure occurs, the job shop loses one processing machine, naturally leading to an increase in the makespan. The values of *DE* are generally under 0.5, indicating that IA-NSGA-II exhibits good stability in handling dynamic scheduling.

Fig. 18 shows the original scheduling plan at the time of failure and the rescheduling plan using IA-NSGA-II when *Machine 1* experiences a failure lasting 10 unit times. In Fig. 18(a), when the failure occurs, *Process 1* is being processed on *Machine 1*. Due to the failure, *Process 1* needs to be rescheduled. From Fig. 18(b), it can be observed that *Process 1* starts processing after the failure is resolved, and no changes in processing machines occur for subsequent processes compared to the initial scheduling plan. This demonstrates that IA-NSGA-II exhibits good stability in handling rescheduling situations caused by machine breakdowns.

## 5. Conclusions

This paper focuses on addressing the complex scheduling challenges in flexible assembly job shops with AGVs to improve the efficiency of workshop. First, a mathematical model for the FAJSP-AGVs was established, with the objectives set to minimize makespan, total machine energy consumption, and AGV working time. Then, an improved adaptive NSGA-II algorithm was proposed. The encoding was designed according to the problem characteristics. For the process sequence part, a process constraint matrix-based encoding method was proposed, effectively avoiding infeasible solutions due to assembly constraints when generating the initial population. Similarly, improved crossover and mutation operators were proposed to ensure the legality of solutions obtained during genetic operations. A variable neighborhood search was also incorporated to further expand the solution space and effectively avoid the algorithm getting trapped in local optima. Finally, feasibility and effectiveness comparison experiments were conducted using test cases proposed in the literatures. The stability of the proposed method in dealing with dynamic scheduling problems was also tested. Experimental results show that the proposed method can effectively solve the FAJSP-AGVs.

In the future, more scenarios closer to actual production can be considered, such as distributed assembly shop scheduling, conflict-free path planning for AGVs, and dynamic scheduling triggered by other special conditions.

## References

(1) Li, X., He, X., Wang, L., and Tang, H. : "Research on the green flexible job shop scheduling considering the transportation time of workpieces", Digital Manufacturing Science, Vol.02, pp.102-106 (2020)
(2) Li, Z., Qian, B., Fang, D., Hu, R., and Zhang, G. : "A hybrid estimation of

distribution algorithm for a certain kind of flexible assembly flow shop scheduling problem", Journal of Industrial Engineering and Engineering Management, Vol.04, pp.200-208 (2017)

(3) Yang, X., Liu, J., Chen, Q., and Mao, N. : "Performance analysis of flexible assembly job shop scheduling under variable disturbance intensity", Computer Integrated Manufacturing Systems, Vol.03, pp.800-814 (2021)

(4) Wu, X., Liu, X., and Zhao, N. : "An improved differential evolution algorithm for solving a distributed assembly flexible job shop scheduling problem", Memetic Computing, Vol.11, pp.335-355 (2019)

(5) Zhang, Z., and Tang, Q. : "Integrating flexible preventive maintenance activities into two-stage assembly flow shop scheduling with multiple assembly machines", Computers & Industrial Engineering, Vol.159, pp.107493 (2021)

(6) Ren, W., Wen, J., Yan, Y., et al. : "Multi-objective optimisation for energy-aware flexible job-shop scheduling problem with assembly operations", International Journal of Production Research, Vol.59, No.23, pp.7216-7231 (2021)

(7) Rahman, H. F., Janardhanan, M. N., and Nielsen, P. : "An integrated approach for line balancing and AGV scheduling towards smart assembly systems", Assembly Automation, Vol.40, No.2, pp.219-234 (2020)

(8) Wang, X., Wu, W., Xing, Z., Zhang, T., and Niu, H. : "A look-ahead AGV scheduling algorithm with processing sequence conflict-free for a no-buffer assembly line", Journal of Advanced Mechanical Design, Systems, and Manufacturing, Vol.17, No.5 (2023)

(9) Karimi, S., Ardalan, Z., Naderi, B., et al. : "Scheduling flexible job-shops with transportation times: Mathematical models and a hybrid imperialist competitive algorithm", Applied Mathematical Modelling, Vol.41, pp.667-682 (2017)

(10) Dai, M., Tang, D., Giret, A., and Salido, M. A. : "Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints", Robotics and Computer-Integrated Manufacturing, Vol.59, pp.143-157 (2019)

(11) Peng, Z., Zhang, H., Tang, H., et al. : "Research on flexible job-shop scheduling problem in green sustainable manufacturing based on learning effect", J Intell Manuf, Vol.33, pp.1725-1746 (2022)

(12) Zhou, B., and Liao, X. : "Particle filter and Levy flight-based decomposed multi-objective evolution hybridized particle swarm for flexible job shop greening scheduling with crane transportation", Applied Soft Computing, Vol.91, Article 106217 (2020)

(13) Homayouni, S. M., Fontes, D. B. M. M., and Gonçalves, J. F. : "A multistart biased random key genetic algorithm for the flexible job shop scheduling problem with transportation", International Transactions in Operational Research 30, 2, 688–716. (2023)

(14) Yan, J., Liu, Z., Zhang, C., Zhang, T., Zhang, Y., and Yang, C. : "Research on flexible job shop scheduling under finite transportation conditions for digital twin workshop", Robotics and Computer-Integrated Manufacturing, Vol.72, Article 102198 (2021)

(15) Cao, Z., Zhou, L., Hu, B., and Lin, C. : "An adaptive scheduling algorithm for dynamic jobs for dealing with the flexible job shop scheduling problem", Business & Information Systems Engineering (2019)

(16) Ozturk, G., Bahadir, O., and Teymourifar, A. : "Extracting priority rules for dynamic multi-objective flexible job shop scheduling problems using gene expression programming", International Journal of Production Research, Vol.01, pp.1-17 (2018)

(17) Hu, L., Liu, Z., Hu, W., et al. : "Petri-net-based dynamic scheduling of flexible manufacturing system via deep reinforcement learning with graph convolutional network", Journal of Manufacturing Systems, Vol.55, No.000, pp.1-14 (2020)

(18) Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. : "A fast and elitist multiobjective genetic algorithm: NSGA-II", IEEE Transactions on Evolutionary Computation, Vol.6, No.2, pp.182-197 (2002)

(19) Blazewicz, J., Ecker, K. H., Pesch, E., Schmidt, G., and Weglarz, J. : "Scheduling computer and manufacturing processes", Journal of the Operational Research Society, Vol.48, No.6, pp.659-659 (1997)

(20) Wei, Y. B., and Wang, R. C. : "Research on job shop scheduling problems based on an improved genetic algorithm", Science Technology and Engineering, Vol.22, No.13, pp.283-285 (2009)

(21) Baykasoglu, A., Owen, S., and Gindy, N. : "A tabu search-based approach to find the Pareto optimal set in multiple objective optimization", Engineering Optimization, Vol.31, No.6, pp.731-748 (1999)

(22) Wu, X., Zhang, Y., and Zhao, K. : "An improved NSGA-II for solving reentrant flexible assembly job shop scheduling problem", Advances in Swarm Intelligence: ICSI 2023. Lecture Notes in Computer Science, Vol.13968 (2023)

(23) Guo, W., Lei, Q., Song, Y., and Lyu, X. : "A learning interactive genetic algorithm based on edge selection encoding for assembly job shop scheduling problem", Computers & Industrial Engineering, Vol.159, Article 107455 (2021)

(24) Weifei, G., Qi, L., Yuchuan, S., Xiangfei, L., and Lei, L. I. : "Integrated scheduling algorithm of complex product with no-wait constraint based on virtual component", Journal of Mechanical Engineering, Vol.56, pp.246 (2020)

(25) Cui, Z., and Gu, X. : "A discrete group search optimizer for hybrid flowshop scheduling problem with random breakdown", Mathematical Problems in Engineering, Vol.2014, No.3, Article 621393 (2014)

(26) Chan, F. T. S., Wong, T. C., and Chan, L. Y. : "Lot streaming for product assembly in job shop environment", Robotics and Computer-Integrated Manufacturing, Vol.24, No.3, pp.321-331 (2008)

(27) Wu, H., and Dun, W. : "Solving scheduling problems of flexible assembly workshop with multiple AGVs", Hoisting and Conveying Machinery, Vol.18, pp.32-38 (2020)

**Haofan Yang**

(Non-member) was born in Beijing, China, on March 23, 1995. He received the Master's degree in engineering from Waseda University, Japan, in 2020. Currently, he is pursuing the Ph.D. degree at the Graduate School of Information, Production, and Systems, Waseda University. His research interests include scheduling problem, especially the multi-objective scheduling problem for different kind of job shops.

**Shigeru Fujimura**

(Senior Member) received the B.E., M.E., and Dr. Eng. degrees from Waseda University in 1983, 1985, and1995, respectively. From 1985 to 2003, he was with the Yokogawa Electric Corporation. Currently, is a Professor with the Graduate School of Information, Production, and Systems, WU. His research interests include production management, production scheduling, intelligent interface agent, objectoriented modeling, and software engineering.