IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Journal Européen des Systèmes Automatisés
Vol. 53, No. 6, December, 2020, pp. 915-924
Journal homepage: http://iieta.org/journals/jesa

A Two-Stage Optimization Algorithm for Multi-objective Job-Shop Scheduling Problem Considering Job Transport

Jianfeng Ren[1,2], Chunming Ye[1*], Yan Li[1]

[1] School of Business, University of Shanghai for Science and Technology, Shanghai 200093, China
[2] School of Computer and Information Engineering, Henan University of Economics and Law, Zhengzhou 450018, China

Corresponding Author Email: 171910083@st.usst.edu.cn

https://doi.org/10.18280/jesa.530617

Received: 19 June 2020
Accepted: 1 September 2020

Keywords:
Job-shop scheduling problem (JSP), multiple objectives, job transport; two-stage optimization, improved fast elitist nondominated sorting genetic algorithm II (INSGA-II)

ABSTRACT

This paper solves the job-shop scheduling problem (JSP) considering job transport, with the aim to minimize the maximum makespan, tardiness, and energy consumption. In the first stage, the improved fast elitist nondominated sorting genetic algorithm II (INSGA-II) was combined with N5 neighborhood structure and the local search strategy of nondominant relationship to generate new neighborhood solutions by exchanging the operations on the key paths. In the second stage, the ant colony algorithm based on reinforcement learning (RL-ACA) was designed to optimize the job transport task, abstract the task into polar coordinates, and further optimizes the task. The proposed two-stage algorithm was tested on small, medium, and large-scale examples. The results show that our algorithm is superior to other algorithms in solving similar problems.

1. INTRODUCTION

Job-shop scheduling, a key link in the intelligent manufacturing system, involves multiple manufacturing resources, such as processing and transport resources. Facing the depletion of global non-renewable energies and the deterioration of the natural environment, it is of great significance to explore the job-shop scheduling problem (JSP) that considers energy consumption [1-4].

Heuristic methods have been widely used to solve multi-objective scheduling problems [5]. One of the most successful algorithms in solving such problems is the genetic algorithm (GA) [6-8]. So far, many improved versions of GA, namely, non-dominated sorting genetic algorithm (NSGA) and fast elitist NSGA (NSGA-II), have been developed, and effectively applied to multi-objective JSPs [9-13].

Lan et al. [14] introduced an improved GA to solve the complex multi-objective problem of flexible job-shop scheduling, and demonstrated the effectiveness of the algorithm through experiments. Considering the sequence of task points and its impact on trajectory distance, Baizid et al. [15] proposed a GA-based method to optimize the sequence of visiting task points, and experimentally proved the validity of the method. Baizid et al. [16] put forward a novel scheduling strategy to determine the shortest path between continuous operating points of job-shop transport devices, which indirectly saves resources. Desirable experimental results have also been achieved by solving multi-objective optimization problems with algorithms like the gray wolf optimizer [17-20]

To sum up, most studies have only obtained the Pareto solutions of the problem. Few have optimized the job transport task based on these solutions. Drawing on the features of specific scheduling problem, this paper improves the NSGA-II and combines it with Q-learning optimization to solve a multi-objective JSP. In the first stage, three objectives, namely, makespan, tardiness, and energy consumption, were optimized to obtain the corresponding Pareto optimal solutions. In the second stage, the job transport task was further optimized to minimize the number of transport robots and their transport paths. Experimental results demonstrate the feasibility and effectiveness of the proposed algorithm. The research results provide theoretical and technical support for the JSPs in similar scenarios.

2. PROBLEM DESCRIPTION

Our research focuses on a JSP with multiple transport robots. Let $J=\{J_1, J_2,\ldots, J_n\}$be the set of n jobs, and $M=\{M_1, M_2,\ldots, M_m\}$ be the set of m machines. Each job has m operations that must be completed on the m machines. In addition, the operation of job i on machine j is denoted as $O_{i,j}$; the start time, processing time, and completion time of job i on machine j are denoted as $S_{i,j}$, $p_{i,j}$, and $C_{i,j}$, respectively. After the completion of operation $O_{i,j}$, job i will be transported by the robot from machine j to machine j+1, kicking off the operation $O_{i,j+1}$. The objectives of the job-shop scheduling include makespan, tardiness, and energy consumption of the job-shop. The production process must satisfy the following hypotheses:

(a) All machines in the job-shop can start working at zero hour;

(b) Unless the release time of job is otherwise specified, the processing of each job can start at zero hour;

(c) Every job has a fixed operation sequence;

(d) Each machine can only process a job at a time, and each job can only be processed by one machine at a time;

(e) No preemption is allowed, and no machine fails during job processing;

(f) There are enough transport robots;

(g) Each robot can only transport one type of materials at a time, and does not fail throughout the process;

(h) There is a buffer zone between stations for temporary

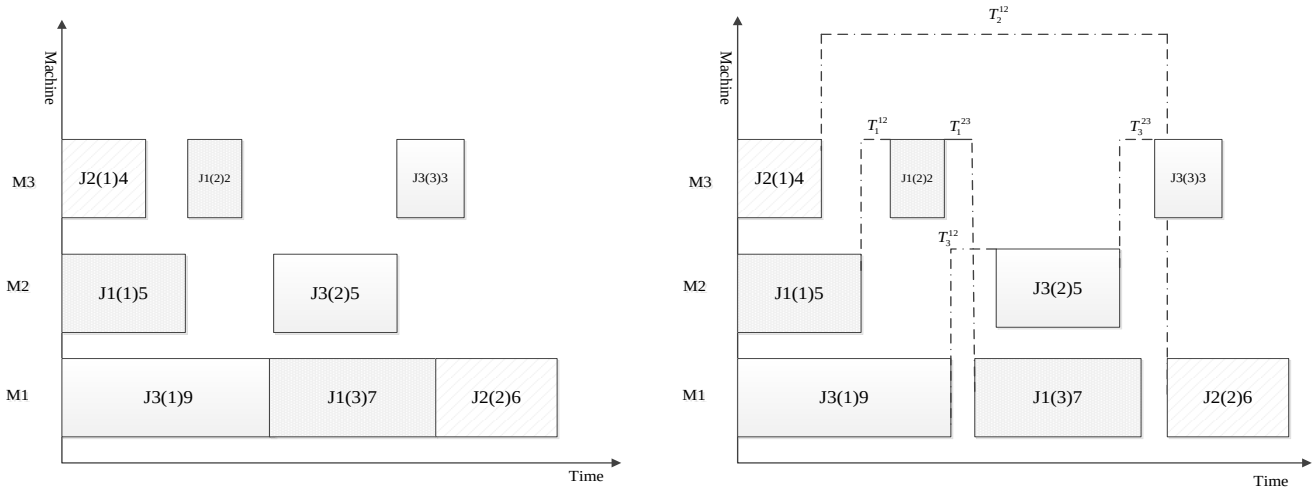
storage of raw materials and finished products;

(i) Each robot can stop at any station to provide transport service, and the loading and unloading times are so small as to be negligible;

(j) The physical distance between stations is known in advance.

Figure 1 (a) describes the simple JSP without considering transport robots, and Figure 1 (b) presents the Gantt charge of centralized scheduling for machines and robots. In Figure 1 (a), operation 1 of J_1 is processed on M_2, and operation 2 of that job is processed on M_1; there is obviously no time interval between the two operations. Figure 1 (b) expresses the sequence of jobs, as well as the transport time $T_i^{k,k+1}$ of each job between every two stations. For example, T_1^{12} means the transport time of job 1 from station 1 to station 2; T_1^{23} means the transport time of job 1 from station 2 to station 3. Although the operation sequence is the same, J_1 needs to be transported by robot between M_2 and M_1, which respectively process jobs 1 and 2. This leads to a transport time between the two operations, which depends on the physical distance between the stations and the efficiency of the robot. In total, five transport tasks T_1^{12}, T_1^{23}, T_2^{12}, T_3^{12}, and T_3^{23} need to be completed. The transport time of each task hinges on the processing time of each operation, the processing capacity of each machine, the physical distance between stations, and the operating speed of the robot.

In addition, the number of robots can be optimized based on the urgency of the processing task, and the cost and energy consumption of robots. Each robot belongs to either load state or no-load state. The load state means the robot carries materials from the current station to the next station; the no-load state means the robot moves from current station to the next station, or waits at the current station, without carrying any materials (Figure 1).



(a) (b)

Figure 1. The Gantt charts of simple scheduling (a) and centralized scheduling (b)

Under the joint constraint of machines and robots, the transport time of jobs is uncertain. But there must exist an optimal transport time. Therefore, the scheduling of our problem aims to select the machines and robots befit the jobs, such that the job processing can start as early as possible and meet the requirement on processing time; Meanwhile, the no-load state of robots needs to be minimized to reduce the number of robots in operation. Further, the processing sequence of jobs and transport paths of robots should be rationalized to satisfy the job transport demand of the job-shop at the lowest cost.

3. PROBLEM MODELING

3.1 Modeling of makespan

For the JSP considering job transport time, the first objective is to minimize the maximum makespan of the jobs, the second is to minimize the tardiness of the jobs, and the third is to minimize the energy consumption of the job-shop. The scheduling plan outputted by the algorithm must satisfy the relevant constraints. The first objective can be modeled as:

$$C_{max} = min\ (maxC_{i,j}), 1 \le i \le n, 1 \le j \le m \tag{1}$$

$$s.t.\ \ C_{i,j} = S_{i,j} + p_{i,j} \tag{2}$$

$$C_{i,j} \ge 0 \tag{3}$$

$$S_{i,j} \ge \max(C_{i,g} + t_{g,j}) \tag{4}$$

$$C_{i,j} + W \cdot (1 - \alpha_{igj}) \ge C_{i,g} + t_{g,j} + p_{i,j}, 1 \le i \le n, 1 \le j, g \le m \tag{5}$$

$$C_{i',j} + W \cdot (1 - \beta_{ii'j}) \ge C_{i,j} + p_{i',j}, 1 \le i, i' \le n, 1 \le j \le m \tag{6}$$

$$\alpha_{igj} = \begin{cases} 1, & \text{if job } i \text{ is processed on machine } g \text{ before job } j \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

If job i is processed on machine g before job j, otherwise

$$\beta_{ii'j} = \begin{cases} 1, & \text{if job } i \text{ is processed on machine } j \text{ before job } i' \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

Formula (1) is the first optimization objective, i.e., the minimization of the maximum makespan of the production task; Formulas (2) and (3) are the completion time of job i and its constraint, respectively; Formula (4) is the start time for the processing of job i, where $t_{g,j}$ is the transport time of the job from machine g to machine j; Formula (5) is the technological constraint, i.e., the operations of the same job are processed in different sequences; Formula (6) is the machine constraint, i.e., each machine can only process one job at a time, where W is a sufficiently large positive number; Formulas (7) and (8) are indicator variables.

3.2 Modeling of tardiness

Based on the model of the first objective, the tardiness of the production task can be modelled as:

$$D_{\text{total}} = \sum_{i=1}^{n} \max\left(0, C_i - C_i^{'}\right) \tag{9}$$

where, C_i and C'_i are the completion time and delivery date of job i, respectively.

3.3 Modeling of energy consumption

The energy consumption of machines was divided into a load-independent component E_1 and a load-dependent component E_2. Let $P_j^{constant}(t)$ be the load-independent comprehensive power of machines, e.g., the energy consumed by the activation of relevant parts during the startup and shutdown of machines. Then, the load-independent component E_1 can be expressed as:

$$E_1 = \sum_{j=1}^{m} \int_0^{t_{start}} C^{constant} \cdot P_j^{constant}(t) dt \tag{10}$$

where, t_{start} is the startup time of machines; $C^{constant}$ is a constant of energy consumption.

The load-dependent component E_2 of machines mainly manifests as the resistance to elastic deformation, the resistance to plastic deformation, and the friction against the tool surface of the material during the cutting process. In actual applications, the resultant cutting force is generally divided into three mutually perpendicular components: cutting force, feed force, and thrust force. Let P_{ij}^{cut} be the comprehensive power of machines in the cutting process. Then, the load-dependent component E_2 can be expressed as:

$$E_2 = \sum_{j=1}^{m} \sum_{i}^{n} C^{cut} \cdot P_{ij}^{cut} \cdot p_{i,j} \tag{11}$$

where, C^{cut} is a constant of energy consumption in cutting; $p_{i,j}$ is the processing time of job i on machine j.

When a machine waits for the next job, an idling energy consumption E_3 will occur. Let P_j^{idle} be the power of idling energy consumption. Then, the idling energy consumption E_3 can be expressed as:

$$E_3 = \sum_{j=1}^{m} C^{idle} \cdot t_j^{idle} \cdot P_j^{idle} \tag{12}$$

where, C^{idle} is a constant of energy consumption in idling; t_j^{idle} is the idling time of machine j.

The transport energy consumption E_4 of a robot refers to the energy consumed by the robot to transport a job between machines. Let $P_{jj'}^{transport}$ be the power of transport energy consumption. Then, the transport energy consumption E_4 can be expressed as:

$$E_4 = C^{transport} \cdot P_{jj'}^{transport} \cdot \sum t_{jj'}, j, j' = 1, 2, \cdots, m \tag{13}$$

where, $C^{transport}$ is a constant of energy consumption in transport; $t_{jj'}$ is the transport time of the robot from machine j to machine j'.

Through the above analysis, the optimization objective of total energy consumption of the job-shop can be established as:

$$E = min\{E_1 + E_2 + E_3 + E_4\} \tag{14}$$

The three aspects of multi-objective optimization were combined into one objective function $f=\{f_1,f_2,f_3\}$ that contains three sub-objective functions. This function can be solved by handling each sub-objective function in turn:

The first objective: minimizing the maximum makespan

$$f_1 = \min(C_{\max}) \tag{15}$$

The second objective: minimizing the tardiness

$$f_2 = \min(D_{total}) \tag{16}$$

The third objective: minimizing the energy consumption

$$f_3 = \min(E) \tag{17}$$

4. IMPROVED NSGA-II (INSGA-II)

The improved fast elitist nondominated sorting genetic algorithm II (INSGA-II) was employed to solve the multi-objective JSP.

4.1 Coding and decoding

By operation sequence coding, all the operations in the production task were coded with natural numbers starting from 1. For each job, the serial number of each operation and that of the corresponding machine were included in the code. Take a JSP with 3 machines and 3 jobs as an example to explain the coding method. Table 1 presents the operations in the problem and their processing sequence.

Table 1. The information of the processing task

J	Processing time			Operation sequence			Serial number of operations		
	M_1	M_2	M_3	M_1	M_2	M_3	M_1	M_2	M_3
J_1	3	4	2	3	2	1	(3)	(2)	(1)
J_2	1	2	1	2	1	3	(5)	(4)	(6)
J_3	1	2	1	1	3	2	(7)	(9)	(8)

Randomly generate chromosome 1 according to the process number. For the three processes of the workpiece, according

to the chromosomal order, the processing order is Process 2 Process 1 Process 3, the processing order of the workpiece is Process 1 Process 3 Process 2, and the processing order of the workpiece is Process 1 Process 3 Process 2. Partially sort each workpiece to obtain legal chromosome 2, as shown in Figure 2.

Chromosome 1 is randomly generated based on the serial number of the operations. For job J_1, the three operations are sorted by the chromosome as: operation 2 → operation 1 → operation 3; For job J_2, the operations are sorted as: operation 1 → operation 3 → operation 2; For job J_3, the operations are sorted as: operation 1 → operation 3 → operation 2.

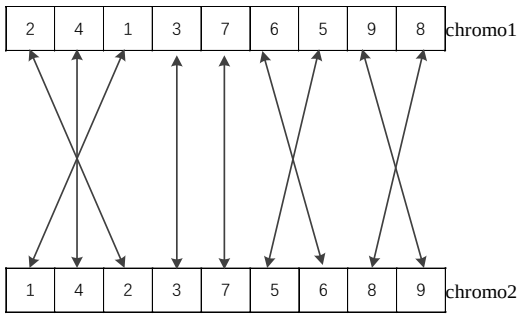


Figure 2. The local sorting of chromosomes

According to the start time and completion time of each operation, the value of the objective function was solved. The decoding process can be specified as follows:

Step 1. Identify the operation at the first gene of the chromosome, find the corresponding job in the operation list, process the job on the corresponding machine, and record the start time and completion time of the operation.

Step 2. Identify the operation at the next gene, find the corresponding job in the operation list, arrange a suitable machine to process the job according to the operation sequence of the job and the idle state of machines, and record the start time and completion time of the operation.

During the machine arrangement, scan all the idle periods of the corresponding machine, and judge if:

(a) The length of idle period [$Idle_{start}$, $Idle_{end}$] is greater than the processing time of the operation $p_{i,j}$;

(b) The start time point of the idle period $Idle_{start}$ is earlier than the completion time point of the previous operation $O_{i\text{-}1,j}$.

If both conditions hold, insert operation $O_{i,j}$ into idle period [$Idle_{start}$, $Idle_{end}$]; otherwise, insert the operation behind the current operation on the machine.

Step 3. Repeat (b) until the operations at all genes of the chromosomes have been inserted.

Step 4. Calculate the makespan C_{max} of the scheduling plan.

4.2 Selection, crossover, and mutation

Our coding method has the natural advantage of retaining excellent chromosome fragments. The excellent genes can be passed down to offspring chromosomes. Then, the crossover operator can be executed to realize global search. However, the offspring chromosomes are not always feasible scheduling plans. Sometimes, the adjacent chromosomes cannot fully demonstrate the operation features on each machine.

(1) Selection

Individuals are chosen by tournament selection. The genes with low non-dominated levels are selected first, while the elite individuals are retained; if two individuals are of the same level, the one with the larger crowding distance is selected. In the parent population, two individuals are randomly selected, and the one with the higher fitness will be retained for the next generation.

(2) Crossover

The precedence preserving order-based crossover (POX) operator was selected for crossover, which can effectively inherit the excellent genes of the parent chromosomes, and achieve better scheduling results under the same conditions. For an $m\times n$ production task, the parent chromosomes of operation codes are denoted as P_1 and P_2, and the offspring chromosomes obtained through POX are denoted as C_1 and C_2. Then, the POX procedure can be summarized as follows (Figure 3):

Step 1. Randomly divide each chromosome $\{1, 2, \ldots, m\times n\}$ into two non-empty chromosome fragments S_1 and S_2.

Step 2. Copy the operations of P_1 in fragment S_1 to C_1, and those of P_2 in fragment S_1 to C_2, while keeping the position of each operation unchanged.

Step 3. Copy the operations of P_2 in fragment S_2 to C_1, and those of P_1 in fragment S_2 to C_2, while keeping the position of each operation unchanged.

Step 4. Obtain the offspring chromosomes through crossover.

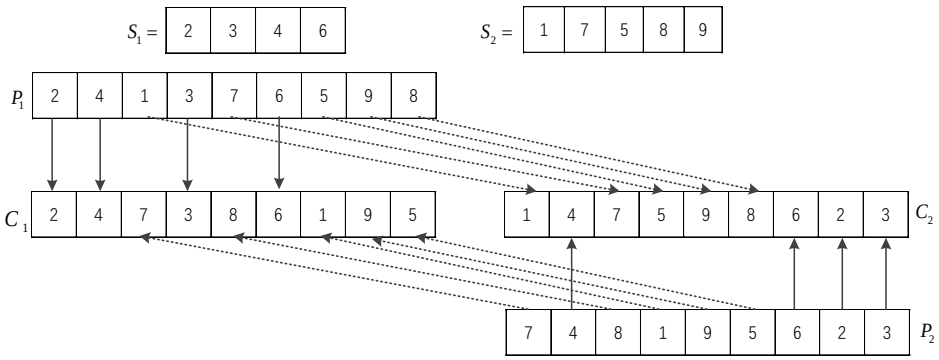


Figure 3. The POX crossover

(3) Mutation

To diversify the population, mutation is implemented by locally disturbing the chromosomes. The common mutation operators include insertion, inversion, and exchange. Different mutation probabilities bring varied benefits. If the mutation probability is low, the excellent gene fragments in

chromosomes can be preserved well, but the optimal solution is very likely to be missed due to the weak search ability; if the probability is high, the algorithm will degenerate into random search, which affects the stability and convergence. This paper implements mutation with a strategy based on neighborhood search, and improves offspring chromosomes through local search. As shown in Figure 4, the mutation process can be detailed as follows:

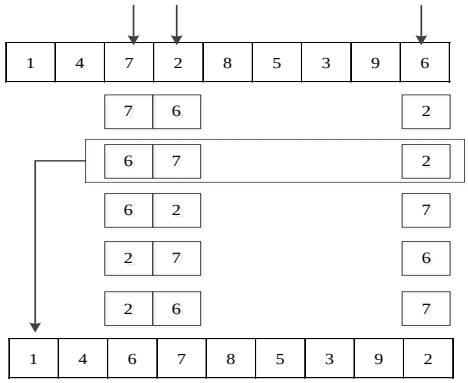


Figure 4. The mutation operation

Step 1. Set the loop variable x=0, and the maximum number of mutations X.

Step 2. Randomly generate the judgment factor μ, and judge the relationship between μ and the mutation probability P_m. If $\mu \leq P_m$, go to the next step; otherwise, terminate the mutation process.

Step 3. Select σ operations from a chromosome, and generate all the neighborhoods of these operations.

Step 4. Calculate the fitness values of all neighborhoods, and identify the one with the highest fitness.

Step 5. Increase x by 1.

Step 6. Terminate the mutation process if $x>X$.

(4) Improved N5 neighborhood search strategy

Based on the classic N5 neighborhood search strategy [21], the N5 neighborhood structure was improved, and coupled with the non-dominated relationship into a new local search strategy:

Step 1. Calculate the total number N_p of individuals in the population, randomly select an individual P_i from the population, and initialize the value of P_i as 1.

Step 2. Move the N5 neighborhood of individual i to generate the corresponding neighborhood individual C_i, and perform decoding on the new individual.

Step 3. Calculate the objective values of the scheduling plan for individual i; if the objective values of C_i are better than those of P_i, then C_i dominates P_i, i.e., $P_i<C_i$, and replace P_i with C_i; otherwise, preserve P_i.

Step 4. Increase P_i by 1; if $P_i \leq N_p$, execute Step 2; otherwise, terminate the iterative process.

5. FURTHER OPTIMIZATION OF JOB TRANSPORT TASK

Despite the consideration of energy consumption in transport, the scheduling plan obtained by INSGA-II only takes account of the theoretical transport energy consumption of jobs in the solving process. This section further optimizes the completion of the transport tasks in the scheduling plan.

5.1 Position scan

During the position scan, the demand points were grouped before path selection. Each demand point was expressed as polar coordinates. Taking a random demand point as the starting point, the zoning was performed in the clockwise or counterclockwise direction under the constraint of robot capacity. Then, the exchange method was implemented to sort the demand points, thereby minimizing the number and energy consumption of robots.

5.2 Path selection

The ant colony algorithm (ACA) was adopted for path selection. During the selection, each ant selects the next node at a certain probability. That is, the transition node with the highest probability is chosen by the ant:

$$p_{ij}^{k}=\begin{cases}\dfrac{\tau_{ij}^{\alpha}(\mathrm{t})*\eta_{ij}^{\beta}(\mathrm{t})}{\sum_{s\in allow_k}\tau_{ij}^{\alpha}(\mathrm{t})\eta_{ij}^{\beta}(\mathrm{t})} & s\in allow_k \\ 0 & s\notin allow_k\end{cases} \tag{18}$$

where, p_{ij}^{k} is the transition probability of ant k from node i to node j; α is the heuristic information (the greater the α value, the more likely for the ant to select the already chosen paths, and the less likely for the ant to randomly explore other paths; the smaller the α value, the smaller the search scope, and the algorithm is prone to falling into the local optimum); β is the expected heuristic factor (the greater the β value, the more likely for the ant to select the local optimal path, the faster the convergence, and the greater the proneness to falling into the local optimum); $\tau_{ij}^{\alpha}(t)$ is the pheromone from node i to node j; $\eta_{ij}^{\beta}(t)$ is the heuristic factor from node i to node j.

To prevent ants from repeatedly selecting nodes that have been visited, the visited nodes were recorded in a tabu list. After time t, all ants completed a traversal search. Then, the path length covered by each ant was calculated, and the minimum was saved. To balance convergence speed with solution quality, threshold $\theta_0 \in (0,1)$ was introduced to the node selection strategy. Before an ant makes a selection, a threshold θ' was randomly generated. If $\theta' \leq \theta_0$, the ant will choose the node with the largest $\tau_{ij}^{\alpha}(t) * \eta_{ij}^{\beta}(t)$; otherwise, the ant will choose another node.

The pheromone can be updated by:

$$\begin{cases}\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\tau_{ij} \\ \tau_{ij}=Q'/fitness(i)\end{cases} \tag{19}$$

where, ρ is the pheromone volatilization factor; 1-ρ is the residual pheromone factor (if ρ value is small, there will be too many residual pheromones on the selected path; in this case, the illegal paths will be searched continuously, which impedes algorithm convergence; If ρ value is large, the invalid paths can be excluded from the search, but the legal paths are not necessarily covered in the search space, which will affect the quality of the optimal solution.); Q' is a constant. After all ants completed a traversal search, the tabu list was emptied, and all ants would return to the initial node, and perform the next round of search.

5.3 Obstacle avoidance

The path optimization and dynamic obstacle avoidance were realized through reinforcement learning (RL), which autonomously interacts with the environment, and obtains the mapping function from state to action by maximizing the reward function. The Q-learning algorithm is a model-free time-series difference algorithm for estimating the Q value. The core idea is to estimate the action of the Q value based on the increment of the reward value, and select the maximum action value of the next state iteratively:

$$Q_{k+1}(s_t, a_t) \leftarrow Q_k(s_t, a_t) + \alpha(r_t + \gamma \max_a Q_k(s_{t+1}, a) - Q_k(s_t, a_t)) \quad (20)$$

As the state transfers from s_t to s_{t+1}, the corresponding reward value r_t is received. The algorithm selects the maximum action value through iterations, and eventually converge to the optimal strategy. Taking the opposite of the path length between nodes as the reward, the path selection by greedy strategy can be expressed as:

$$a_k = \arg \max_{a \in A(s_k)} \{r_{k+1}^a + V(S_{k+1}^a)\} \quad (21)$$

6. EXPERIMENTS AND RESULTS ANALYSIS

6.1 Experimental setup

Table 2. The main parameters of the INSGA-II

Name	Meaning	Value
N_{popu}	Initial population	300
N_{pare}	Size of Pareto solution set	40
N_{iter}	Number of iterations	400
p_{sele}	Selection probability	0.02
p_{cros}	Crossover probability	0.90
p_{muta}	Mutation probability	0.10

The main parameters of the INSGA-II are presented in Table 2. The main cost parameters of the transport task are given in Table 3. In addition, the other parameters were configured as: the pheromone importance factor=1; constant Q=1; the importance of heuristic function=5; pheromone volatilization factor=0.1; crossover probability=0.85; genetic gap=0.8; mutation probability=0.1, learning rate=0.1; discount factor=0.9.

Table 3. The main cost parameters of the transport task

Type	Name	Value	Weight
Basic cost	Basic cost of robot assignment	300	0.3
Transport cost	Unit distance cost of robot	5	0.5
Time window penalty costs	Penalty coefficient for early arrival	0.002	0.2
	Penalty coefficient for late arrival	0.04	
	Operating speed of robot	30	
	Maximum capacity of robot	4	

6.2 Comparative analysis

(a) Multi-objective scheduling experiments

Tables 4, 5, and 6 record the processing sequence, processing time, and energy consumption power of each job on each machine, respectively.

First, the INSGA-II and NSGA-II were separately adopted to find the reasonable scheduling plan for an 8×8 JSP, with the aim to minimize the maximum makespan and energy consumption. Each algorithm was run 40 times independently. The optimal results of the two algorithms were compared. As shown in Table 7, most results of NSGA-II were dominated by those of INSGA-II. The results of INSGA-II had higher non-dominance level. Thus, INSGA-II clearly outperformed NSGA-II.

Table 4. The processing sequence

	Processing sequence							
	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
J_1	2	5	4	8	3	7	6	1
J_2	8	5	7	2	1	4	6	3
J_3	7	3	5	8	2	4	1	6
J_4	4	8	5	3	2	1	7	6
J_5	2	4	7	6	1	5	8	3
J_6	2	4	3	5	1	8	7	6
J_7	1	5	4	3	8	2	6	7
J_8	6	5	7	2	4	3	1	8

Table 5. The processing time

	Processing time								Release time	Delivery date
	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	0	150
J_1	17	18	11	14	18	18	10	15	0	120
J_2	11	8	21	18	12	8	19	25	30	190
J_3	17	23	21	9	18	20	9	9	0	120
J_4	13	20	11	19	10	23	15	19	0	180
J_5	15	18	11	8	12	10	13	15	0	180
J_6	17	8	25	18	10	8	9	7	0	-
J_7	7	18	9	10	14	18	12	13	10	190
J_8	11	8	25	18	12	9	22	13	0	150

Table 6. The power of energy consumption

	Machine							
Energy consumption	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
Cutting energy consumption	22	15	9	10	6	7.5	8	12
Idling energy consumption	3.2	2.3	1.8	2.1	1.9	4.4	3.4	4.0
Other energy consumption	1.2	1.3	0.9	0.7	1.2	1.4	1.6	1.4
Transport energy consumption	3	3	3	3	3	3	3	3

Table 7. Some solutions of the two algorithms

INAGA-II			NAGA-II		
Makespan (C_{max})	**Energy consumption (E)**	**Tardiness (D_{total})**	**Makespan (C_{max})**	**Energy consumption (E)**	**Tardiness (D_{total})**
159	405	84	163	409	89
160	400	86	164	405	91
161	394.9	87	166	404	92
162	390	87.5	166	403	93
163	385	88	167	400	93
164	383	88.5	168	399	94
165	380	89	168	397	94
166	372	90	169	390	95
167	370	90	170	388	96
168	365	91	171	386	97

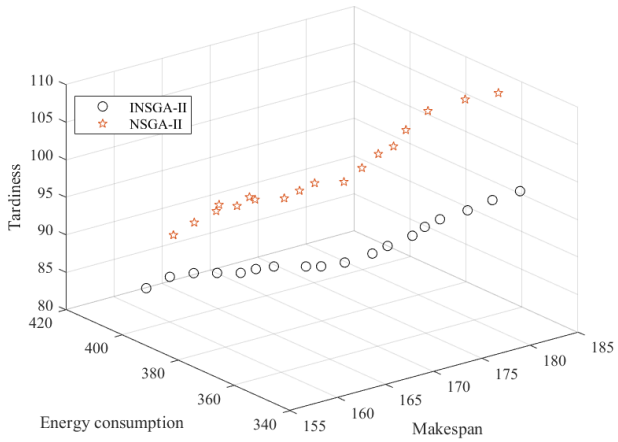


Figure 5. The distribution of Pareto solutions

The distribution of Pareto solutions of the two algorithms are compared in Figure 5, where the three dimensions stand for energy consumption, makespan, and tardiness, respectively. Obviously, the Pareto solutions of INSGA-II were nearly all on the upper right of those of NSGA-II.

Suppose the theoretical optimal solution is point (155, 75, 320). The spatial distance between the solutions of each algorithm and the theoretical value can be solved by:

$$Distance_{pateto} = \frac{1}{N}\sum_{i=1}^{N}\sqrt[2]{(C_{max}^{i}-155)^{2}+(D_{total}^{i}-75)^{2}+(E^{i}-320)^{2}} \quad (22)$$

For comparability, the three terms under the radical in (22) were normalized by:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (23)$$

It was calculated that the $Distance_{pateto}$ of INSGA-II stood at 3.333, and that of NSGA-II at 3.357. Obliviously, the result of INSGA-II is closer to the theoretical optimal solution.

Next, the mean maximum makespan and mean number of iterations of each algorithm in 40 independent runs were calculated to compare their convergence speeds. As shown in Figure 6, INSGA-II converged in an average of 200 iterations, while NSGA-II converged in an average of 216 iterations. Thus, INSGA-II converges faster than NSGA-II.

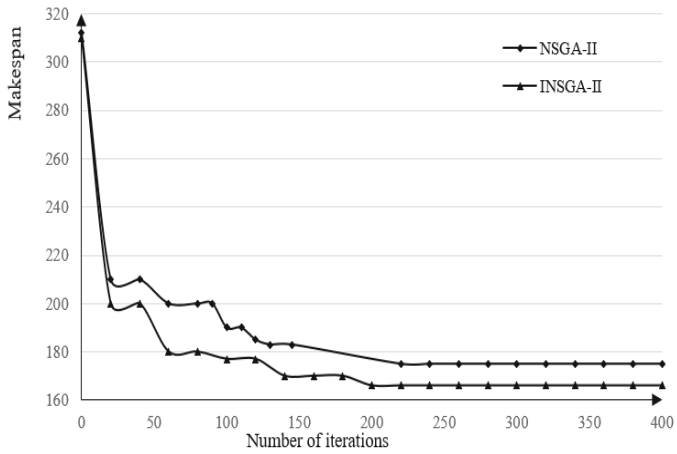


Figure 6. The comparison of convergene curves

Through the experiments on the small-scale problem of 8×8, the proposed INSGA-II outperforms NSGA-II in solution quality and convergence speed. To further validate its performance, our algorithm was compared with NSGA-II and SPEA2 (Strength Pareto Evolutionary Algorithm 2) [22] on medium and large-scale problems. Four numbers of jobs were selected: 20, 50, 80, and 100; Each number of jobs were processed on 4, 6, 8, and 10 machines, respectively. Each algorithm was run independently for 40 times on each problem. The makespan of the processing task was generated from a uniform distribution $U(30,100)$. The total makespan of jobs p can be expressed as:

$$p = \sum_{i=1}^{n} p_i \quad (24)$$

The delivery date of jobs can be calculated by:

$$D_i = p\cdot(1-\xi) + (p/1.2 + p\cdot(1-\xi))/(n-1) \quad (25)$$

where, ξ is a set of values from 0.2 to 1.0; the step size is 0.2.

Table 8 compares the gradient descent (GD) of the three algorithms on the 16 medium and large-scale problems. It can be seen that INSGA-II outshined the other algorithms on most examples: SPEA2 achieved the best results on 1 medium-scale example; NSGA-II achieved the best results on 4 examples; INSGA-II achieved the best results on 11 examples. This further verifies the excellent convergence ability of INSGA-II. Moreover, the best results on all four large-scale examples belonged to INSGA-II, an evidence to the superiority of our algorithm in solving large-scale JSPs.

Table 8. The comparison of GD indices

Problem	SPEA2		NSGA-II		INSGA-II	
	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation
20×4	3.96E-03	7.16E-03	8.06E-03	7.47E-03	5.48E-03	3.75E-03
20×6	9.33E-03	1.39E-03	3.67E-03	8.10E-03	1.20E-03	2.03E-03
20×8	8.77E-03	3.22E-03	2.31E-03	8.12E-03	1.05E-02	1.84E-03
20×10	2.23E-03	8.36E-03	3.49E-03	3.09E-03	1.17E-03	2.52E-03
50×4	5.61E-02	1.07E-02	3.59E-02	3.67E-02	2.62E-02	1.30E-02
50×6	6.70E-03	9.59E-03	2.47E-03	2.66E-03	9.10E-03	3.91E-03
50×8	6.71E-03	1.69E-03	9.47E-03	7.70E-03	4.09E-03	4.26E-03
50×10	1.92E-03	7.89E-03	1.36E-03	9.45E-03	4.74E-03	6.92E-03
80×4	5.82E-03	1.12E-03	1.26E-03	9.69E-03	4.30E-03	6.93E-03
80×6	5.23E-03	2.23E-03	8.39E-03	8.29E-03	2.06E-03	1.28E-03
80×8	3.83E-02	8.14E-02	3.66E-02	2.57E-02	1.06E-02	4.26E-02
80×10	9.60E-03	7.59E-03	3.03E-03	1.12E-03	3.43E-03	4.08E-03
100×4	7.49E-02	2.81E-02	4.77E-02	1.90E-02	3.42E-02	1.08E-02
100×6	1.33E-03	4.34E-03	1.77E-03	3.72E-03	1.01E-03	5.56E-03
100×8	9.40E-03	7.36E-03	6.42E-03	6.39E-03	4.24E-03	9.36E-03
100×10	3.22E-03	2.62E-03	1.82E-03	4.98E-03	1.04E-02	3.35E-02
Accuracy		1/16		4/16		11/16

Table 9. The comparison of IGD indices

Problem	SPEA2		NSGA-II		INSGA-II	
	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation
20×4	4.05E-03	3.75E-03	3.91E-03	9.77E-03	1.06E-04	7.43E-04
20×6	4.32E-03	5.37E-03	7.42E-03	6.69E-03	1.27E-03	1.61E-03
20×8	8.46E-03	3.19E-03	5.73E-03	3.20E-03	3.43E-04	7.68E-04
20×10	8.34E-03	4.23E-03	1.08E-03	2.39E-03	8.83E-04	7.77E-04
50×4	1.08E-02	1.06E-02	4.04E-03	2.82E-03	1.10E-04	2.23E-04
50×6	8.37E-03	3.82E-03	8.80E-03	2.23E-03	2.16E-03	4.75E-03
50×8	3.82E-03	6.67E-03	8.76E-03	7.07E-03	8.26E-04	8.26E-04
50×10	1.01E-02	4.20E-03	8.33E-03	1.28E-03	6.81E-04	9.38E-04
80×4	4.08E-03	5.51E-03	8.72E-03	7.66E-03	6.04E-04	4.07E-04
80×6	2.83E-03	8.64E-03	3.14E-03	1.21E-03	9.93E-04	9.36E-04
80×8	1.01E-03	8.69E-03	6.79E-03	5.01E-03	9.32E-04	1.01E-04
80×10	8.65E-03	6.96E-03	9.55E-03	8.46E-03	7.46E-03	9.12E-03
100×4	3.55E-03	4.05E-03	1.05E-03	6.07E-03	7.31E-03	3.11E-03
100×6	6.31E-03	1.02E-02	4.90E-03	8.03E-03	6.09E-04	1.03E-04
100×8	2.17E-03	3.29E-03	7.91E-03	2.49E-03	9.69E-04	4.97E-04
100×10	8.31E-03	6.75E-03	4.54E-03	8.56E-03	2.35E-04	1.10E-04
Accuracy		0/15		1/16		15/16

Table 10. The comparison of spread indices

Problem	SPEA2		NSGA-II		INSGA-II	
	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation
20×4	3.14E-01	2.11E-01	1.36E-01	6.47E-01	8.41E-01	5.29E-01
20×6	6.67E-01	5.66E-01	7.54E-01	1.84E-01	2.16E-01	4.43E-01
20×8	8.13E-01	9.89E-01	1.05E-02	5.14E-01	4.36E-01	8.40E-01
20×10	8.48E-00	6.01E-00	8.43E-00	6.47E-00	3.74E-00	1.02E-00
50×4	9.73E-02	9.58E-02	1.83E-02	1.01E-02	5.68E-02	2.18E-02
50×6	1.02E-02	3.76E-02	9.63E-02	2.29E-02	2.89E-02	6.58E-02
50×8	1.02E-02	5.13E-02	1.01E-02	6.63E-02	9.20E-02	7.31E-02
50×10	4.10E-02	4.46E-02	4.40E-02	8.12E-02	1.95E-02	4.12E-02
80×4	7.31E-02	4.42E-02	9.25E-02	4.31E-02	3.29E-02	1.54E-02
80×6	4.47E-01	3.42E-01	1.01E-02	2.95E-01	1.08E-02	2.67E-01
80×8	5.09E-01	9.49E-01	3.30E-01	9.42E-01	1.01E-01	3.91E-01
80×10	1.66E-02	6.26E-02	1.08E-02	2.40E-02	1.05E-02	8.26E-02
100×4	8.65E-02	6.47E-02	7.87E-02	4.36E-02	7.50E-02	5.57E-02
100×6	8.99E-02	7.16E-02	9.80E-02	1.07E-02	5.19E-02	1.02E-02
100×8	9.99E-00	1.14E-00	4.85E-00	1.15E-00	3.09E-00	3.62E-00
100×10	3.60E-02	5.62E-02	8.50E-02	7.77E-02	1.67E-02	1.01E-02
Accuracy		1/16		5/16		10/16

The incremental gradient descent (IGD) of the three algorithms on the 16 medium and large-scale problems are compared in Table 9. It can be seen that INSGA-II achieved the best results on 15 examples, and NSGA-II achieved the best results on 1 example. Therefore, INSGA-II has much better performance than the other algorithms in terms of IGD.

Table 10 compares the spread of the three algorithms on the 16 medium and large-scale problems. It can be seen that INSGA-II achieved the best results on 10 examples, NSGA-II achieved the best results on 5 examples, and SPEA2 achieved the best results on 1 example. This means the solution distribution of INSGA-II is better than that of the other algorithms.

(b) Further optimization of job transport task

A total of 20 examples were selected, including five 10×4 examples, five 10×6 examples, five 10×8 examples, five 10×10 examples, five 20×4 examples, five 20×6 examples, five 20×8 examples, and five 20×10 examples. The proposed algorithm was running 30 times on each example. The number of assigned robots, transport energy consumption, and robot utilization rate were calculated. Among them, the robot utilization rate refers to the ratio of the load state time t_w of a robot to the total time t_v from entering into transport task to the completion of the transport of the last job:

$$Utilization\ ratio = \frac{t_w}{t_v} \times 100\% \tag{26}$$

The relevant results are recorded in Table 11. It can be seen that the further optimization of job transport task effectively suppressed the number of assigned robots and the transport energy consumption, and greatly enhanced the robot utilization rate. This fully demonstrates the effectiveness of our algorithm.

Table 11. The comparison of transport results

Problem	Results of first stage			Post-optimization results		
	Number	Transport energy consumption	Utilization ratio	Number	Transport energy consumption	Utilization ratio
10×4	10	50.4	36.5%	6.2	30.5	90.0%
10×6	10	80.7	30.7%	6.1	48.8	88.3%
10×8	10	92.6	27.4%	7.3	65.3	80.9%
10×10	10	107.5	22.9%	8	80.2	78.2%
20×4	20	78.9	33.4%	12	50.7	87.3%
20×6	20	90.3	28.3%	11	60.3	83.2%
20×8	20	123.5	22.3%	11	71.2	78.1%
20×10	20	189.3	15.7%	9	110.6	74.5%

7. CONCLUSIONS

This paper designs a two-stage optimization algorithm for multiple objectives of JSP with job transport, including the makespan, tardiness, and energy consumption of the production task, and proves the feasibility and effectiveness of the algorithm through experiments. The research also proves that the combination between RL and GA can greatly improve optimization performance. The future research will further probe into the synergy between the two techniques.

ACKNOWLEDGMENT

The study was supported by “Key Soft Science Project of "Science and Technology Innovation Action Plan of Shanghai Science and Technology Commission, China (Grant No.: 20692104300); National Natural Science Foundation, China (Grant No.: 71840003); Technology Development Project of University of Shanghai for Science and Technology, China (Grant No.: 2018KJFZ043).”.

REFERENCES

[1] Zhang, R., Chiong, R. (2016). Solving the energy-efficient job shop scheduling problem: A multi-objective genetic algorithm with enhanced local search for minimizing the total weighted tardiness and total energy consumption. Journal of Cleaner Production, 112: 3361-3375. https://doi.org/10.1016/j.jclepro.2015.09.097

[2] Ding, J.Y., Song, S., Wu, C. (2016). Carbon-efficient scheduling of flow shops by multi-objective optimization. European Journal of Operational Research, 248(3): 758-771. https://doi.org/10.1016/j.ejor.2015.05.019

[3] Mansouri, S.A., Aktas, E., Besikci, U. (2016). Green scheduling of a two-machine flowshop: Trade-off between makespan and energy consumption. European Journal of Operational Research, 248(3): 772-788. https://doi.org/10.1016/j.ejor.2015.08.064

[4] Dai, M., Tang, D., Giret, A., Salido, M.A. (2019). Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints. Robotics and Computer-Integrated Manufacturing, 59: 143-157. https://doi.org/10.1016/j.rcim.2019.04.006

[5] Türkyılmaz, A., Şenvar, Ö., Ünal, İ., Bulkan, S. (2020). A research survey: Heuristic approaches for solving multi objective flexible job shop problems. Journal of Intelligent Manufacturing, 31: 1949-1983. https://doi.org/10.1007/s10845-020-01547-4

[6] Nouri, H.E., Driss, O.B., Ghédira, K. (2016). Simultaneous scheduling of machines and transport robots in flexible job shop environment using hybrid metaheuristics based on clustered holonic multiagent model. Computers & Industrial Engineering, 102: 488-501. https://doi.org/10.1016/j.cie.2016.02.024

[7] May, G., Stahl, B., Taisch, M., Prabhu, V. (2015). Multi-objective genetic algorithm for energy-efficient job shop scheduling. International Journal of Production Research, 53(23): 7071-7089. https://doi.org/10.1080/00207543.2015.1005248

[8] Huang, X., Yang, L. (2019). A hybrid genetic algorithm

for multi-objective flexible job shop scheduling problem considering transportation time. International Journal of Intelligent Computing and Cybernetics, 12(2): 154-174. https://doi.org/10.1108/IJICC-10-2018-0136

[9] Fu, H.C., Liu, P. (2019). A multi-objective optimization model based on non-dominated sorting genetic algorithm. International Journal of Simulation Modelling, 18(3): 510-520. https://doi.org/10.2507/IJSIMM18(3)CO12

[10] González-Rodríguez, I., Puente, J., Palacios, J.J., Vela, C.R. (2020). Multi-objective evolutionary algorithm for solving energy-aware fuzzy job shop problems. Soft Computing, 24: 16291-16302 https://doi.org/10.1007/s00500-020-04940-6

[11] Wu, X., Che, A. (2020). Energy-efficient no-wait permutation flow shop scheduling by adaptive multi-objective variable neighborhood search. Omega, 94: 102117. https://doi.org/10.1016/j.omega.2019.102117

[12] Wen, X., Wang, K., Li, H., Sun, H., Wang, H., Jin, L. (2020). A Two-stage solution method based on NSGA-II for green multi-objective integrated process planning and scheduling in a battery packaging machinery workshop. Swarm and Evolutionary Computation, 61: 100820. https://doi.org/10.1016/j.swevo.2020.100820

[13] Yazdani, M., Zandieh, M., Tavakkoli-Moghaddam, R. (2019). Evolutionary algorithms for multi-objective dual-resource constrained flexible job-shop scheduling problem. OPSEARCH, 56(3): 983-1006. https://doi.org/10.1007/s12597-019-00395-y

[14] Lan, M., Xu, T.R., Peng, L. (2010). Solving flexible multi-objective JSP problem using an improved genetic algorithm. JSW, 5(10): 1107-1113. https://doi.org/10.4304/jsw.5.10.1107-1113

[15] Baizid, K., Chellali, R., Yousnadj, A., Meddahi, A., Bentaleb, T. (2010). Genetic algorithms based method for time optimization in robotized site. In 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010: 1359-1364. https://doi.org/10.1109/IROS.2010.5651948

[16] Baizid, K., Yousnadj, A., Meddahi, A., Chellali, R., Iqbal, J. (2015). Time scheduling and optimization of industrial robotized tasks based on genetic algorithms. Robotics and Computer-Integrated Manufacturing, 34: 140-150. https://doi.org/10.1016/j.rcim.2014.12.003

[17] Komaki, G.M., Kayvanfar, V. (2015). Grey wolf optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time. Journal of Computational Science, 8: 109-120. https://doi.org/10.1016/j.jocs.2015.03.011

[18] Mirjalili, S. (2016). Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Computing and Applications, 27(4): 1053-1073. https://doi.org/10.1007/s00521-015-1920-1

[19] Gao, L., Li, X., Wen, X., Lu, C., Wen, F. (2015). A hybrid algorithm based on a new neighborhood structure evaluation method for job shop scheduling problem. Computers & Industrial Engineering, 88: 417-429. https://doi.org/10.1016/j.cie.2015.08.002

[20] Zhu, Z., Zhou, X. (2020). An efficient evolutionary grey wolf optimizer for multi-objective flexible job shop scheduling problem with hierarchical job precedence constraints. Computers & Industrial Engineering, 140: 106280. https://doi.org/10.1016/j.cie.2020.106280

[21] Nowicki, E., Smutnicki, C. (1996). A fast taboo search algorithm for the job shop problem. Management Science, 42(6): 797-813. https://doi.org/10.1287/mnsc.42.6.797

[22] Zitzler, E., Laumanns, M., Thiele, L. (2001). SPEA 2: Improving the strength Pareto evolutionary algorithm, TIK report 103. Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Zurich, Switzerland, 236.