International Journal of Industrial Engineering Computations 15 (2024) 813–832

Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

A novel hybrid algorithm of genetic algorithm, variable neighborhood search and constraint programming for distributed flexible job shop scheduling problem

Leilei Meng[a*], Weiyao Cheng[a], Biao Zhang[a], Wenqiang Zou[a] and Peng Duan[a*]

[a]*School of Computer Science, Liaocheng University, Liaocheng 252000, China*

C H R O N I C L E

Article history:
Received December 14 2023
Received in Revised Format January 30 2024
Accepted March 26 2024
Available online March 26 2024

Keywords:
Distributed flexible job shop scheduling problem
Genetic algorithm
Variable neighborhood search
Constraint programming
Makespan minimization

A B S T R A C T

With the decentral and global economy, distributed scheduling problems are getting a lot of attention. This paper addresses a distributed flexible job shop scheduling problem (DFJSP) with minimizing makespan, in which three subproblems, namely operations sequencing, factory selection and machine selection must be determined. To solve the DFJSP, a novel mixed-integer linear programming (MILP) model is first developed, which can solve the small-scaled instances to optimality. Since the NP-hard characteristic of DFJSP, a hybrid algorithm (GA-VNS-CP) of genetic algorithm (GA), variable neighborhood search (VNS) and constraint programming (CP) is then designed. Specifically, the GA-VNS-CP is divided into two stages. The first stage uses the hybrid meta-heuristic algorithms of GA and VNS (GA-VNS), and the VNS is designed to improve the local search ability of GA. In GA-VNS, the encoding only considers the factory selection and the operations sequencing problems, and the machine selection problem is determined by the decoding rule. Because the solution space may be limited by the decoding rule, the second stage uses the CP to extend the solution and further improve the solution. Numerical experiments based on benchmark instances are conducted to evaluate the effectiveness of the MILP model, VNS, CP and GA-VNS-CP. The experimental results show the effectiveness of the MILP model, VNS and CP. Moreover, the GA-VNS-CP algorithm has better performance than traditional algorithms and improves 6 current best solutions for benchmark instances.

1. Introduction

Nowadays, multi-factory production exists extensively due to the decentral and global economy. With multi-factory production, production orders can be finished more quickly than in a traditional single-factory production environment. The distributed scheduling problem in multi-factory production environments is becoming more and more popular (Xu and Hu et al., 2021; J. and X. et al., 2022; Sang and Tan, 2022; He and Pan et al., 2024).There are several distributed scheduling problems, such as distributed parallel machines scheduling problem, distributed flow shop scheduling problem, distributed job shop scheduling problem, distributed flexible flow shop scheduling problem and distributed flexible job shop scheduling problem (DFJSP). Specifically, DFJSP is the multi-factory environment of flexible job shop problem (Meng and Zhang et al., 2020a). DFJSP is much harder than FJSP, and it must determine three sub-problems(Li and Xie et al., 2022): (1) select a factory for every job (factory selection problem), (2) select a machine for each operation (machine selection problem) and (3) determine the operations sequence assigned on the same machine (operations sequencing problem). Genetic algorithm (GA) is inspired by the process of natural selection and has been widely implemented to solve shop scheduling problems (Meng and Zhang et al., 2019b; Meng and Cheng et al., 2023). Moreover, GA shows its good effectiveness for solving FJSP and DFJSP (Wu and Lin et al., 2017). As a swarm intelligent algorithm, GA has a good ability of global searching. However, the local searching ability of GA is unsatisfactory. Therefore, we introduce variable neighborhood search (VNS) with good local search ability to improve the local search ability of GA (Du and Li et al., 2021; Meng and Zhang et al., 2019a). The hybrid method of GA and VNS is named GA-VNS. The solution space of meta-heuristic algorithms is determined by their encoding and decoding methods, and it may not include all the solutions of the studied problem. Therefore, to enlarge the solution space and further

* Corresponding author
E-mail mengleilei@lcu-cs.com (L. Meng) duanpeng@lcu-cs.com (P. Duan)
ISSN 1923-2934 (Online) - ISSN 1923-2926 (Print)
2024 Growing Science Ltd.
doi: 10.5267/j.ijiec.2024.3.001

improve the solution quality, constraint programming (CP) search is introduced to further improve the best solution obtained by GA-VNS. The hybrid method of GA, VNS and CP is named GA-VNS-CP. Moreover, to better describe and formulate DFJSP, a novel mixed-integer linear programming (MILP) model is developed, which can solve the small-scaled instances to optimality (Dai & Pan et al., 2023; Meng & Duan et al., 2024). In comparison with existing studies, this study has three main contributions, which are given as follows:

(1) A novel MILP is developed to solve the small-scaled instances of DFJSP to optimality.
(2) A hybrid algorithm GA-VNS-CP is designed.
(3) Four neighborhood structures in VNS are designed based on critical factory and critical operations.

The remainder of this study is organized as follows: Section 2 introduces the literature review of DFJSP. Section 3 describes the DFJSP and gives the mathematical model. Section 4 presents the GA-VNS-CP algorithm. Section 5 shows the experiments. Section 6 presents the conclusion and future work.

2. Literature review

Table 1 gives an overview of existing research about DFJSP from the published year, problem, objective and solving methods.

Table 1
Existing research about DFJSP

Reference	Year	Problem	Objective	Methods
(Jia and Fuh et al., 2002)	2002	DJSP	production cost	a GA
(Jia and Nee et al., 2003)	2003	DJSP	makespan	a modified GA (MGA)
(Chan and Chung et al., 2005)	2005	DJSP	makespan	a GA with dominant genes (GADG)
(Chan and Chung et al., 2006)	2006	DFJSP with machine maintenance	makespan	an improved GADG with a novel local search method
(Chung and Chan et al., 2009)	2009	DFJSP with machine maintenance	makespan	a modified GA
(De Giovanni and Pezzella, 2010)	2010	DFJSP	makespan	an improved GA (IGA)
(Naderi and Azab, 2014)	2014	DJSP	makespan	MILP model
(Ziaee, 2014)	2014	DFJSP	makespan	a fast heuristic algorithm
(Lu and Wu et al., 2015)	2015	DFJSP	makespan	a GA with a concise encoding (GA_JS)
(Liu and Chen et al., 2015)	2015	DFJSP	makespan	a GA with a refined encoding operator
(Chang and Liu, 2017)	2017	DFJSP	makespan	a hybrid genetic algorithm (HGA) with a novel encoding scheme
(Wu and Lin et al., 2017)	2017	DFJSP	makespan	a GA with a encoding that only considers operations sequencing (called GA_OP)
(Marzouki and Driss et al., 2018)	2018	DFJSP	makespan	a chemical reaction optimization (CRO) algorithm
(Li and Duan et al., 2018)	2018	DFJSP	makespan, workload and earliness/tardiness	a multi-objective tabu search algorithm
(Wu and Liu et al., 2018)	2018	DFJSP	earliness/tardiness and total cost	an improved differential evolution simulated annealing algorithm (IDESAA)
(Lin and Lee et al., 2019)	2019	DFJSP with machine maintenance	makespan	a GA based on SG1 or SG2
(Xie and Gao et al., 2019)	2019	DJSP	makespan and energy consumption	a multi-objective artificial bee colony algorithm (MOABC)
(Jiang and Wang et al., 2020)	2020	DJSP	makespan and energy consumption	a modified multi-objective evolutionary algorithm with decomposition (MMOEA/D)
(Meng and Ren et al., 2020)	2020	DFJSP	energy consumption	a MILP model and a hybrid multi-objective shuffled frog-leaping algorithm (HSFLA)
(Meng and Zhang et al., 2020a)	2020	DFJSP	makespan	four MILP and one CP models
(Luo and Deng et al., 2020)	2020	DFJSP with transfers	makespan, maximum workload, and total energy consumption	an efficient multi-objective memetic algorithm (EMA)
(Du and Li et al., 2021)	2021	DFJSP with crane transportations	makespan and energy consumption	a hybrid algorithm that combines estimation of distribution algorithm and VNS
(Xu and Hu et al., 2021)	2021	DFJSP	makespan, costs, quality and carbon emission	a hybrid algorithm that combines genetic algorithm and tabu search
(Ahman, 2021)	2021	DJSP	makespan	a discrete spotted hyena optimizer (DSHO)
(Li and Xie et al., 2022)	2022	DFJSP	makespan	an effective improved gray wolf optimizer (IGWO)
(Li, Gu, et al., 2022)	2022	DFJSP	makespan	a hybrid chemical reaction optimization (HCRO) algorithm with a novel encoding-decoding method
(Luo and Deng et al., 2022)	2022	DFJSP with worker arrangement	makespan, maximum workload of machines and workload of workers	an improved multi-objective memetic algorithm (IMA)
(Tang and Fang et al., 2022)	2022	DFJSP	makespan	a hybrid teaching–learning-based optimization (HTLBO) algorithm
(Sang and Tan, 2022)	2022	DFJSP	makespan, total energy consumption, running time of all equipment, delay time and processing quality	a high-dimensional many-objective memetic algorithm (HMOMA)
(Zhu and Gong et al., 2023)	2023	dynamic DFJSP with operation inspection	makespan and total energy consumption	a modified memetic algorithm (MMA)
(Bagheri Rad and Behnamian, 2023)	2023	Dynamic DJSP with availability constraints and new job arrivals	makespan and total energy consumption	an improved multi-objective memetic algorithm

As shown in Table 1, DFJSP is attracting more and more attention, and more and more papers have been published from 2002 to now. The objective develops from single objective to multi-objective, from makespan\cost to energy consumption, and from static objective to dynamic objective. Regarding the solving methods, exact method and approximation method are used. Specifically, the exact methods are mainly MILP and CP models. Approximation methods are mainly meta-heuristic algorithms, especially the GA. Because exact methods are subject to their low efficiency, approximation methods (meta-heuristic algorithms) are mostly used (Meng and Zhang et al., 2020a).

As can be seen in Table 1, the distributed job scheduling problem (DJSP), as a specific case of DFJSP, was first studied in 2002 with minimizing production cost and solved by a GA(Jia and Fuh et al., 2002). DJSP with the objective of minimizing makespan was first studied in 2003 and a modified GA was proposed (Jia and Nee et al., 2003). DFJSP was first studied in 2006 with minimizing makespan and an improved GADG with a novel local search method was designed (Chan and Chung et al., 2006). MILP model for DJSP with minimizing makespan was first designed in 2014 (Naderi and Azab, 2014). Because DFJSP is much harder than DJSP, MILP models for DFJSP were first designed in 2020 (Meng and Zhang et al., 2020a). Multi-objective DFJSP was first studied in 2018 with simultaneously minimizing makespan, workload and earliness/tardiness (Li and Duan et al., 2018). DJSP with minimizing energy consumption was first studied in 2019, and a multi-objective artificial bee colony algorithm (MOABC) was designed (Jiang and Wang et al., 2020). DFJSP with minimizing energy consumption was first studied in 2020, and a MILP model and a hybrid shuffled frog-leaping algorithm (HSFLA) were designed (Meng and Ren et al., 2020). Dynamic DJSP and DFJSP were first studied in 2023 with simultaneously minimizing makespan and total energy consumption, and multi-memetic algorithms were designed (Bagheri Rad and Behnamian, 2023; Zhu and Gong et al., 2023).

About the meta-heuristic algorithms, the encoding scheme is extremely important. As described above, three sub-problems must be determined in DFJSP. If the encoding scheme includes all the three sub-problems, then its solution space is dominant. If the encoding scheme does not include all the three sub-problems and some sub-problems must be determined in the decoding scheme by specific rules, then its solution space is non-dominant (Chang and Liu, 2017). Table 2 shows the encoding schemes in existing research for DFJSP with makespan minimization. As can be seen in Table 2, most of the studies use the non-dominant encoding scheme. Moreover, by analyzing the existing studies, the non-dominant encoding scheme is more effective than the dominant encoding scheme. This is because the solution space of the dominant encoding scheme is very large, and it is difficult to design evolution operators and find good solutions. The solution space of a non-dominant encoding scheme is much smaller than the dominant encoding scheme, and it is easy to design evolution operators and find relatively good solutions. Of course, the optimal solutions may be missed by using the dominant encoding scheme. Therefore, in this paper, the GA-VNS of our proposed GA-VNS-CP uses the same non-dominant encoding scheme to quickly obtain a good solution. To make up for the disadvantage of a non-dominant encoding scheme, GA-VNS-CP uses the CP to search the full solution space and improves the best solution obtained by GA-VNS.

Table 2
Existing encoding schemes for DFJSP with makespan minimization

Reference	Encoding	Decoding	Solution space
(Chan and Chung et al., 2006)	three sub-problems	No	dominant
(De Giovanni and Pezzella, 2010)	operations sequencing and factory selection	machine selection	non-dominant
(Lu and Wu et al., 2015)	No (jobs sequencing)	three sub-problems	non-dominant
(Liu and Chen et al., 2015) (Chang and Liu, 2017)	three sub-problems	No	dominant
(Wu and Lin et al., 2017)	operations sequencing	factory selection and machine selection	non-dominant
(Li and Xie et al., 2022)	operations sequencing and factory selection	machine selection	non-dominant
(Li, Gu, et al., 2022)	operations sequencing and factory selection	machine selection	non-dominant
(Tang and Fang et al., 2022)	three sub-problems	No	dominant

3 DFJSP descriptions

3.1 DFJSP definition

The DFJSP with minimizing makespan are defined as follows: there are a certain number of factories, and each of them is a FJSP production environment. A certain number of jobs are processed in these factories, and each of them can be machined in one factory. Moreover, every job has several operations with a determined processing route, and each of them can only be processed by only one machine. In DFJSP, three subproblems, namely operations sequencing, factory selection and machine selection must be determined. In this paper, the objective is minimizing makespan by determining three problems of DFJSP. Moreover, the assumptions of DFJSP are as follows: (1) All the jobs and the machines in all factories are ready at time 0; (2) At a time, each machine can machine only one job and each job can be processed on only one machine; (3) Once an operation is started on a machine, it must be processed without interruption; (4) All the processing times are deterministic.

3.2 Mathematical model

The notations in the MILP model are as follows:

Notations

i, i'	job indexes
n	total number of jobs
I	job set, $I = \{1, 2, \cdots, n\}$
j, j'	operation indexes
n_i	number of operations of job i
N	total number of operations, $N = \sum_{i \in I} n_i$
J_i	operation set of job i, $J_i = \{1, 2, \cdots, n_i\}$
$O_{i,j}$	j-th operation of job i
k, k'	machine indexes
f	factory index
nf	number of factories
F	factory set, $F = \{1, \ldots, nf\}$
$K_{i,j,f}$	machine set in factory f for processing $O_{i,j}$
$pt_{i,j,f,k}$	processing time of machine k in factory f for processing $O_{i,j}$
M	a very large positive number

Decision variables

$X_{i,j,f,k}$	binary decision variable, $X_{i,j,f,k} = \begin{cases} 1, & \text{if operation } O_{i,j} \text{ selects to be processed on machine } k \text{ of factory } f \\ 0, & \text{otherwise} \end{cases}$
$Y_{i,j,i',j'}$	binary decision variable, $Y_{i,j,i',j'} = \begin{cases} 1, & \text{if operation } O_{i,j} \text{ is processed before operation } O_{i',j'} \text{ on a machine} \\ 0, & \text{otherwise} \end{cases}, i < i'$
$Z_{i,f}$	binary decision variable, $Z_{i,f} = \begin{cases} 1, & \text{if job } i \text{ selects to be processed in factory } f \\ 0, & \text{otherwise} \end{cases}$
$B_{i,j}$	continuous decision variable, it represents the starting time of operation $O_{i,j}$.
$Bf_{i,j,f}$	continuous decision variable, it represents the starting time of operation $O_{i,j}$ in factory f.
$C_{\max}$	makespan
C_f	makespan of factory f

The objective is given as below,

$$\min \ C_{\max} \tag{1}$$

subject to

$$\sum_{f \in F} Z_{i,f} = 1, \forall i \in I \tag{2}$$

$$Z_{i,f} = \sum_{k \in K_{i,j,f}} X_{i,j,f,k}, \ \forall i \in I, j \in J_i, f \in F \tag{3}$$

$$B_{i,j} + \sum_{f \in F} \sum_{k \in K_{i,j,f}} (pt_{i,j,f,k} X_{i,j,f,k}) \le B_{i,j+1}, \forall i \in I, j \in \{1, 2, ..., n_i - 1\} \tag{4}$$

$$Bf_{i,j,f} + pt_{i,j,f,k} \le Bf_{i',j',f} + M(3 - Y_{i,j,i',j'} - X_{i,j,f,k} - X_{i',j',f,k}), \forall i, i' \in I, i < i', j \in J_i, j' \in J_{i'}, f \in F, k \in K_{i,j,f} \cap K_{i',j',f} \tag{5}$$

$$Bf_{i',j',f} + pt_{i',j',f,k} \le Bf_{i,j,f} + M(2 + Y_{i,j,i',j'} - X_{i,j,f,k} - X_{i',j',f,k}), \forall i, i' \in I, i < i', j \in J_i, j' \in J_{i'}, f \in F, k \in K_{i,j,f} \cap K_{i',j',f} \tag{6}$$

$$B_{i,j} = \sum_{f \in F} Bf_{i,j,f}, \forall i \in I, j \in J_i \tag{7}$$

$$C_f \ge Bf_{i,n_i,f} + \sum_{k \in K_{i,n_i,f}} (pt_{i,n_i,f,k} X_{i,n_i,f,k}), \forall i \in I, f \in F \tag{8}$$

$$C_{\max} \ge C_f, \forall f \in F \tag{9}$$

$$B_{i,j,f} \ge 0, \forall i \in I, j \in J_i \tag{10}$$

$$B_{i,j,f} \le M Z_{i,f}, \forall i \in I, j \in J_i, f \in F \tag{11}$$

$$X_{i,j,f,k} \in \{0,1\}, \forall i \in I, j \in J_i, f \in F, k \in K_{i,j,f} \tag{12}$$

$$Y_{i,j,i',j'} \in \{0,1\}, \forall i, i' \in I, i < i', j \in J_i, j' \in J_{i'} \tag{13}$$

$$Z_{i,f} \in \{0,1\}, \forall i \in I, f \in F \tag{14}$$

where, constraint set (2) enforces that each job is processed only in one factory. Constraint set (3) defines that all the operations of a job are assigned to the same factory. Constraint sets (2) and (3) together ensure that each operation is processed by only one machine. Constraint set (4) restricts the processing route of all the operations of a job. Constraint sets (5)-(6) determine the order of the operations processed on the same machine, which are described intuitively in Fig. 1. Specifically, as shown in Fig. 1, when both $X_{i,j,f,k}$ and $X_{i',j',f,k}$ are equal to 1 ($X_{i,j,f,k} = X_{i',j',f,k} = 1$), there are two cases: if $Y_{i,j,i',j'}$ is equal to 1($Y_{i,j,i',j',k} = 1$), constraint set (5) ensures that $Bf_{i',j',f}$ is no less than $Bf_{i,j,f} + pt_{i,j,f,k}$ ($Bf_{i,j,f} + pt_{i,j,f,k} \le Bf_{i',j',f}$) and constraint set (6) is relaxed; If $Y_{i,j,i',j'}$ is equal to 0 ($Y_{i,j,i',j'} = 0$), constraint set (5) is relaxed and constraint set (6) ensures that $Bf_{i,j,f}$ is no less than $Bf_{i',j',f} + pt_{i',j',f,k}$ ($Bf_{i',j',f} + pt_{i',j',f,k} \le Bf_{i,j,f}$). When at least one of $X_{i,j,f,k}$ and $X_{i',j',f,k}$ are equal to 0 ($X_{i,j,f,k} X_{i',j',f,k} = 0$), both constraint sets (5) and (6) are relaxed. Constraint set (7) shows the relationship of decision variables $B_{i,j}$ and $Bf_{i,j,f}$. Constraint set (8) defines that the makespan C_f is no less than the completion time of all the jobs assigned to factory f. Constraint set (9) defines that the makespan $C_{\max}$ is no less than the makespan C_f of all factories. Constraint sets (10)-(11) defines the range of decision variable $Bf_{i,j,f}$. Specifically, constraint sets (10)-(11) restrict that $Bf_{i,j,f}$ is equal to 0 when job i is not assigned to factory f. Constraint sets (12)-(14) present the range of binary decision variables.

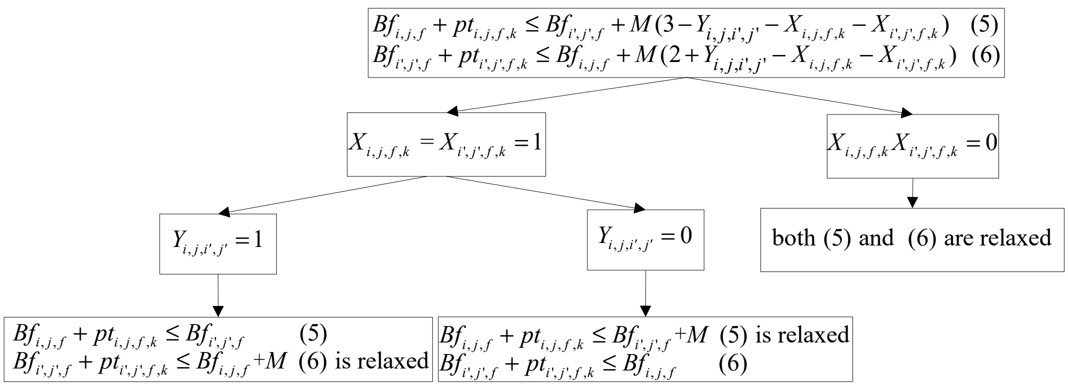


Fig. 1. Description of constraint sets (5) and (6)

3.3 An example

To better show the DFJSP and the MILP model, an example in Fig. 2 is given. This example includes two factories and three jobs. As can be seen from Fig. 2, Jobs 1 and 3 are assigned to Factory 1, and Job 2 is assigned to Factory 2. Then, the decision variable $Z_{i,f}$ of the model is as follows: $Z_{1,1} = 1$, $Z_{2,2} = 1$ and $Z_{3,1} = 1$. In Factory 1, operations $O_{1,2}$ and $O_{3,2}$ are assigned to Machine 2, and operations $O_{1,1}$, $O_{3,1}$ and $O_{1,3}$ are assigned to Machine 1. In Factory 2, operations $O_{2,1}$ and $O_{2,2}$ are assigned to Machines 1 and 2 respectively. Then, the decision variable $X_{i,j,f,k}$ of the model is as follows: $X_{1,1,1,1} = 1$, $X_{1,2,1,2} = 1$, $X_{1,3,1,1} = 1$, $X_{3,1,1,1} = 1$, $X_{3,2,1,2} = 1$, $X_{2,1,2,1} = 1$ and $X_{2,2,2,2} = 1$. On Machine 1 in Factory 1, the sequence of operations is $O_{1,1}$, $O_{3,1}$ and $O_{1,3}$, and the decision variable $Y_{i,j,i',j'}$ is as follows: $Y_{1,1,3,1} = 1$ and $Y_{1,3,3,1} = 0$. On Machine 1 in Factory 1, the sequence of operations is $O_{1,2}$ and $O_{3,2}$, and the decision variable $Y_{i,j,i',j'}$ is as follows: $Y_{1,2,3,2} = 1$. The makespan of Factories 1 and 2 are 6 and 4 respectively, and the decision variable C_f is as follows: $C_1 = 6$ and $C_2 = 4$. The makespan is 6, and the decision variable $C_{\max}$ is as follows: $C_{\max} = 6$. The starting times of operations $O_{1,1}$, $O_{1,2}$, $O_{1,3}$, $O_{2,1}$, $O_{2,2}$, $O_{3,1}$ and $O_{3,2}$ are 0, 2, 4, 0, 2, 2 and 4, and the decision variables $B_{i,j}$ is as follows: $Bf_{1,1,1} = B_{1,1} = 0$, $Bf_{1,2,1} = B_{1,2} = 2$, $Bf_{1,3,1} = B_{1,3} = 4$, $Bf_{2,1,2} = B_{2,1} = 0$, $Bf_{2,2,2} = B_{2,2} = 2$, $Bf_{3,1,1} = B_{3,1} = 2$ and $Bf_{3,2,1} = B_{3,2} = 4$. Moreover, decision variables $Bf_{1,1,2}$, $Bf_{1,2,2}$, $Bf_{1,3,2}$, $Bf_{2,1,1}$, $Bf_{2,2,1}$, $Bf_{3,1,2}$ and $Bf_{3,2,2}$ are equal to 0.

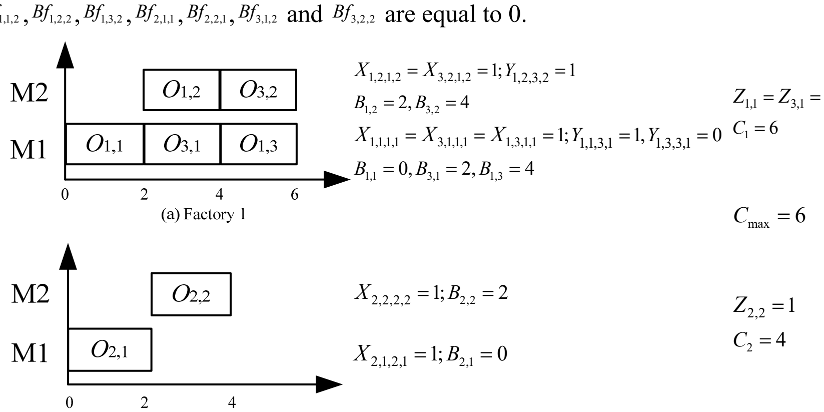


Fig. 2. An example for DFJSP

4. The GA-VNS-CP algorithm for DFJSP

In the following sections, the proposed GA-VNS-CP algorithm is described from GA, VNS and CP in detail. Specifically, the GA-VNS-CP are divided into two stages. The first stage uses the hybrid meta-heuristic algorithms of GA and VNS (GA-VNS), and the VNS is used to improve the local search ability of GA. In GA-VNS, the encoding only considers the factory selection and the operations sequencing subproblems, and the machine selection subproblem is determined by decoding rule with specific rules. Because the solution space maybe limited by the decoding rule, the second stage use the CP search to extend the solution space and further improve the best solution obtained by GA-VNS.

4.1 Workflow of the proposed GA-VNS-CP

Fig. 3 shows the flow chart of the proposed GA-VNS-CP, and the detailed steps of the GA-VNS-CP are given as follows:

Step 1: Initialization: Initialize the parameters and the initial population and set $t = 1$. Go to Step 2.
Step 2: Genetic evolutions: Execute the genetic operations, namely selection operators in Section 4.2.4, crossover operators in Section 4.2.5 and mutation operators in Section 4.2.6. Go to Step 3.
Step 3: Elitist solution set (ESS) criteria: If the ESS criteria is met, go to Step 4; otherwise, go to Step 7. Specifically, the ESS criteria is that the iteration t is the multiples of *Nt*. In other words, population diversity check, ESS updating and VNS on ESS are conducted in each *Nt* iteration.
Step 4: Population diversity check: Execute the population diversity check according to the methods in Section 4.2.7. Go to Step 5.
Step 5: ESS updating: Update the ESS according to the methods in Section 4.2.8. Go to Step 6.
Step 6: VNS on ESS: Firstly, conduct the VNS on ESS according to the methods in Section 4.3. Then, replace the top worst 5%*Np* solutions with the solutions in ESS. Go to Step 7.
Step 7: $t = t + 1$. Go to Step 8.
Step 8: Termination: Is the stopping criteria of GA-VNS reached? If the stopping criteria is satisfied, go to Step 9; otherwise, go to Step 2.
Step 9: CP search: Conduct the CP search on the best solution obtained by GA-VNS until the CP stopping criteria is met. Specifically, the best solution obtained by GA-VNS is set as the initial solution of CP. Go to Step 10.
Step 10: Output the final best solution.

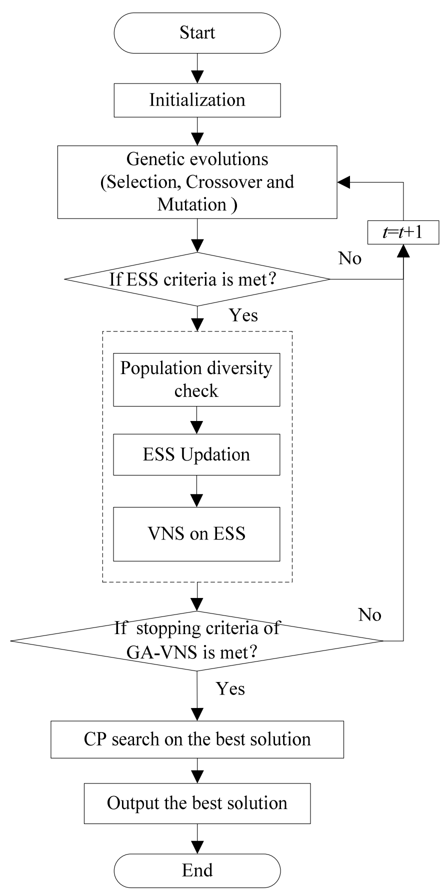


Fig. 3. The flow chart of GA-VNS-CP

4.2 GA

The GA is described from the following eight aspects, namely initialization, encoding scheme, decoding scheme, selection operators, crossover operators, mutation operators, population diversity check and elitist solution set.

4.2.1 Initialization

About GA, the initial population and parameters must be determined first. Specifically, as to the initial population, each individual is randomly produced on the basis of the following encoding and encoding schemes. For the parameters of GA, there are five parameters that should be determined, namely the iteration number *Nt*, the population size *Np*, the crossover probability *Pc*, the mutation probability *Pm* and the stopping criteria.

4.2.2 Encoding scheme

Regarding GA, the encoding is used to represent an individual, and all the operators are conducted on individuals. As described in Section 2, the encoding is extremely important for meta-heuristic algorithms, and it determines the solution space. In this paper, the non-dominant encoding SFS that only considers operations sequence (OS) string and factory selection (FS) string is used (Li and Xie et al., 2022). Specifically, OS and FS strings determine the operation sequencing and factory selection subproblems respectively. As to the machine selection subproblem, it is determined in the decoding scheme. For OS string, it defines all the operations, and its length equals the total number of operations. Specifically, the operations of the same job are presented as the same job number. For FS string, its genes represent the selected factories for all jobs, and its length equals the total number of jobs. To intuitively show the encoding, an example that includes three jobs and two factories are given is Fig. 4. As can be seen in Fig. 4, the operations sequence is $O_{1,1}$, $O_{3,1}$, $O_{1,2}$, $O_{1,3}$, $O_{3,2}$, $O_{2,1}$ and $O_{2,1}$, and Jobs 1-3 are processed in factories 1, 2 and 1 respectively.

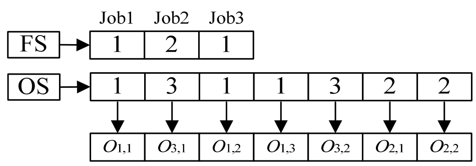


Fig. 4. Encoding scheme SFS in this paper

4.2.3 Decoding scheme

The function of decoding is to transform an encoding chromosome to a real schedule, in which all the three subproblems must be determined. Moreover, specific starting and ending times of all operations are determined in decoding. The heuristics for determining the machine selection problem are minimum current makespan (MCM) and shortest process time (SPT) (Li and Xie et al., 2022). Specifically, with regard to each operation, according to the operations sequence in OS string, it selects the machine that can machine itself at the earliest (In other words, for each operation, the machine with the minimum current makespan is selected). When multiple machines are with the same completion times, the machine with SPT is selected. Specifically, the relationship between the encoding and decoding schemes are shown in Fig. 5. With decoding, a real scheduling scheme can be obtained, in which all the starting times, ending times, machine selections and factories can be seen intuitively.

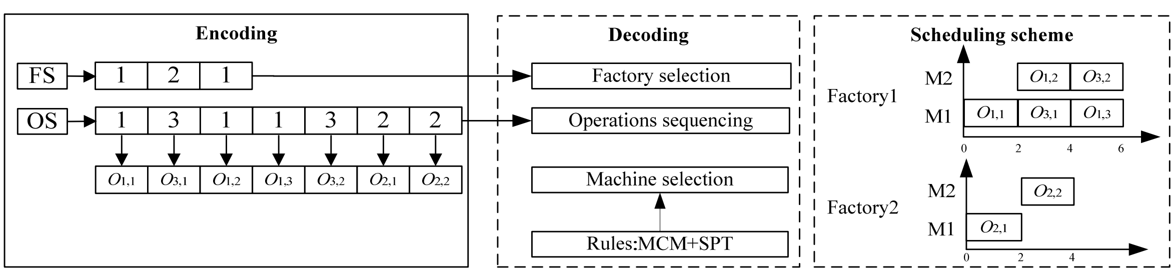


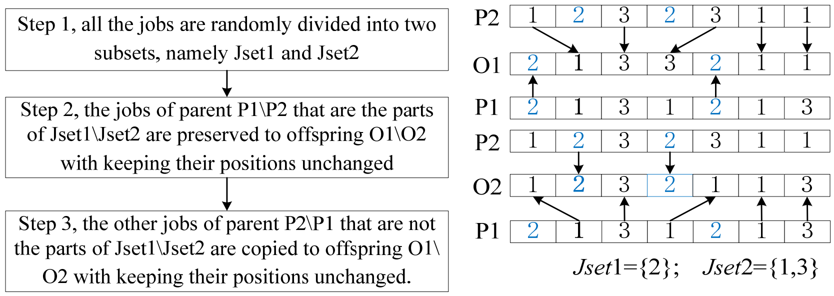
Fig. 5. Relationship between the encoding and decoding schemes

4.2.4 Selection operators

In GA, the selection operator is to transmit individuals from parent population to offspring population according to fitness. In this paper, the fitness is the makespan. We use two selection operators, namely binary tournament selection and elitist selection. Specifically, the binary tournament selection randomly selects two individuals from the parent population and preserves the best one to the offspring population. The elitist selection preserves the best individual of the parent population directly to the offspring population.

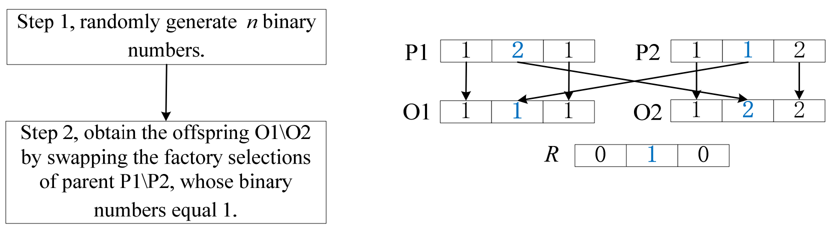
4.2.5. Crossover operators

For OS and FS, precedence operation crossover (POX) and uniform crossover (UC) are used respectively(Meng and Cheng et al., 2023; Meng and Zhang et al., 2023). Specifically, POX includes three steps, and it is shown in Fig. 6(a). Moreover, a small example is given in Fig. 6(b) to intuitively show the POX. The steps of UC are shown in Fig. 7(a), and a small example of UC is shown in Fig. 7(b).



(a) Steps of POX crossover (b) Example of POX crossover for OS

Fig. 6. Steps and example of POX crossover for OS



(a) Steps of UC crossover (b) Example of UC crossover for FS

Fig. 7. Steps and example of UC crossover for FS

4.2.6. Mutation operators

In this paper, Swap operator is used for OS string and reassign operator is adopted for FS string. Specifically, Swap operator exchanges two randomly selected operations. The reassign operator randomly selects one job and changes its factory selection. In each iteration of GA, Swap and Reassign operators are randomly selected with 50% probability (Meng & Cheng et al., 2023).

4.2.7. Population diversity check

For classical GA, with the iterating of population evolutions, some individuals may become very similar or even identical. In other words, the population diversity decreases, the population converges to local optimum easily(Meng and Cheng et al., 2023). In order to improve this kind of condition, the population is regularly checked, and similar individuals are reproduced. In other words, if the makespan in each factory of two individuals is identical, one individual is regenerated. If in each iteration, the population diversity is checked, the advantage of selection operators cannot be fully utilized. Therefore, the population diversity check is executed in each *Nt* generation.

4.2.8 Elitist solution set (ESS)

ESS preserves the relatively good solutions obtained in the evolution of GA. The size of ESS is set as 5%*Np*. If in each generation, ESS is updated, all the solutions in the population and ESS must be ordered, it will be very time-consuming. Therefore, in each *Nt* generations, the ESS is updated by top 5%*Np* solutions in the population.

4.3 VNS

VNS is a well-known local search method and has been proved effective in many scheduling problems (Karimi and Rahmati et al., 2012; Meng and Zhang et al., 2019a; Meng and Ren et al., 2020; Meng and Zhang et al., 2023). VNS works by systematically exploring several different neighborhood structures, and thus local optimal solutions in these neighborhoods are obtained. By comparing these local optima, a better solution even the global optimal solution can be archived (Meng and Zhang et al., 2023). In general, VNS is based on three perceptions, which are given as follows:

(1) A local optimum of one neighborhood structure is not necessarily a local one for another neighborhood structure.
(2) A global optimum is a local optimum with respect to all possible neighborhood structures.
(3) For many problems, local optima with respect to one or several neighborhoods are relatively close to each other.

The design of neighborhoods is very important (Meng and Zhang et al., 2023). In this study, four neighborhood structures are applied to produce new solutions. The first three neighborhood structures namely Swap, Insertion and Reversion are for OS string. The fourth neighborhood structure is Reassign, and it is for FS string. Because the makespan of DFJSP is determined by the makespan of the critical factory (the factory is with the maximum makespan among all factories), and the makespan of critical factory is determined by the critical operations. Therefore, Swap, Insertion and Reversion must change the sequencing of the critical operations in critical factory. Reassigning must change the factory of critical jobs (jobs are with critical operations). Fig. 8 gives an example of the four neighborhood structures.

N_1 (Swap): Randomly select one critical operation and another operation (critical or non-critical) and exchange their order.
N_2 (Insertion): Firstly, randomly select one critical operation and another operation (critical or non-critical) and move the operation in the back is moved just before the operation in the front.
N_3 (Reversion): Randomly select one critical operation and another operation (critical or non-critical), and reverse the operations between them.
N_4 (Reassign): Randomly select one critical job and change its factory selection.

The detailed steps of VNS are given as follows:

Step 1: Randomly generate the initial solution x and the set of neighborhood structure $N_k(x), k=1...k_{max}$.
Step 2: Repeat the following Steps 3-6 until the stop criteria is satisfied $k > k_{max}$.
Step 3: Set $k=1$.
Step 4(Shaking): Generate a solution x' randomly from the kth neighborhood of x, $x' \in N_k(x)$.
Step 5(Local search): Apply some local search method with x' as initial solution. The local search is as follow:
Step 5.1: Set $t=1$;
Step 5.2: Generate a solution x'' from the kth neighborhood of x' ($x'' \in N_k(x')$);
Step 5.3: If x'' is better than solution x', replace x' with x'' and $t=t+1$;otherwise, $t=t+1$;
Step 5.4: Repeat Step 5.2-5.3 until t reaches N.
Step 6: If solution x' is better than solution x, replace x with x' and set $k=1$;otherwise, $k=k+1$.

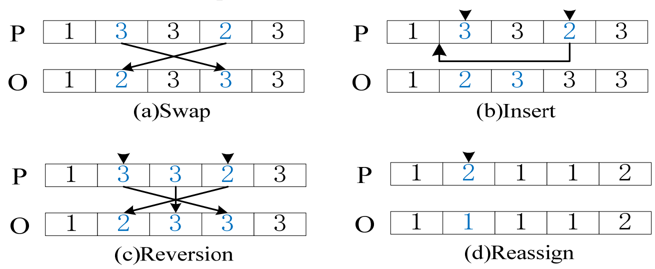


Fig. 8. Four neighborhood structures of VNS

4.4 CP

With regard to CP, it can obtain optimal solutions and has been proved to be effective for solving shop scheduling problems(Ham and Cakici, 2016; Bukchin and Raviv, 2018; Gedik and Kalathia et al., 2018; Ham and Park et al., 2021; Meng and Lu et al., 2021; de Abreu and Araújo et al., 2022). In CP, constraint propagation (filtering) method is used to transmit information between constraints and decision variables (Meng and Zhang et al., 2020b; Zhang and Yu et al., 2021; Meng and Gao et al., 2022). Different from MILP models, CP defines two new types of interval variables, namely interval decision variable and sequence decision variable. Due to no standardization in defining constraints, variables and functions in CP, the models formulated in different CP solvers, such as Cplex, OR Tools and Gecode are different. In this paper, the CP model is solved by Cplex (Meng and Zhang et al., 2020b), and the related parameters, decision variables and functions are described as follows:

Parameters:

$Op_{i,j}$	It represents $O_{i,j}$.
$Mod_{i,j,k,pt}$	It represents the machine and processing time for processing $O_{i,j}$.
Decision variables:	
$op_{i,j}$	It represents interval variable for $Op_{i,j}$.
$mod_{i,j,k}$	It represents optional interval variable for $Mod_{i,j,k,pt}$.
$mchs_k$	It represents sequence decision variable and consists of all the optional interval variables $mod_{i,j,k}$ of machine k
Functions:	
$endOf(a)$	It returns the end time of interval variable a.
$endBeforeStart(a,b)$	It constrains that interval variable $b$can start only when interval variable a is finished.
$alternative(op_{i,j}, mod_{i,j,k})$	It means that only one of optional interval variables $mod_{i,j,k}$ for interval variable $op_{i,j}$ can be present.

$noOverlap(mchs_k)$ It restricts the non-overlapping of the optional interval variables $mod_{i,j,k}$ present in sequence variable $mchs_k$.

Objective function (15) states that the makespan is the maximum completion of all jobs.

$$\min\ C_{\max} = \max_{i\in I}(endOf(op_{i,n_i})) \tag{15}$$

$$alternative(op_{i,j}, mod_{i,j,k}), \forall i\in I, j\in J_i \tag{16}$$

$$noOverlap(mchs_k), \forall k\in K \tag{17}$$

$$endBeforeStart(op_{i,j}, op_{i,j+1}), \forall i\in I, j\in \{1,...,n_i-1\} \tag{18}$$

where, constraint set (16) guarantees that each operation can only be assigned to one machine. In other words, for each operation, $alternative(op_{i,j}, mod_{i,j,k})$ forces only one of $mod_{i,j,k}$ can be selected by $op_{i,j}$.Constraint set (17) assures that all the operations assigned to the same machine cannot overlap. In detail, $noOverlap(mchs_k)$ assures that all the present variables $mod_{i,j,k}$ of $mchs_k$ cannot overlap. Constraint set (18) ensures the sequence of the operations for each job.

As to DFJSP, each factory is an FJSP environment. Therefore, for the best solution obtained by GA-VNS, it is further improved by the CP search with warm start. Specifically, for each factory, in which a CP search is formulated to optimize the operations sequencing and machine selections of the jobs assigned. Fig. 9 shows an example of CP search with the initial solution being the best solution obtained by GA-VNS. As can be seen from Fig. 9, Jobs 1 and 3 are assigned to Factory 1, and Job 2 is assigned to Factory. Therefore, CP1 is formulated for Factory1 to optimize the operations sequencing and machine selections of Jobs 1 and 3, and CP2 is formulated for Factory 2 to optimize the operations sequencing and machine selections of Job 2. Moreover, the CP1 and CP2 start with the initial solution of the best solution obtained by GA-VNS.

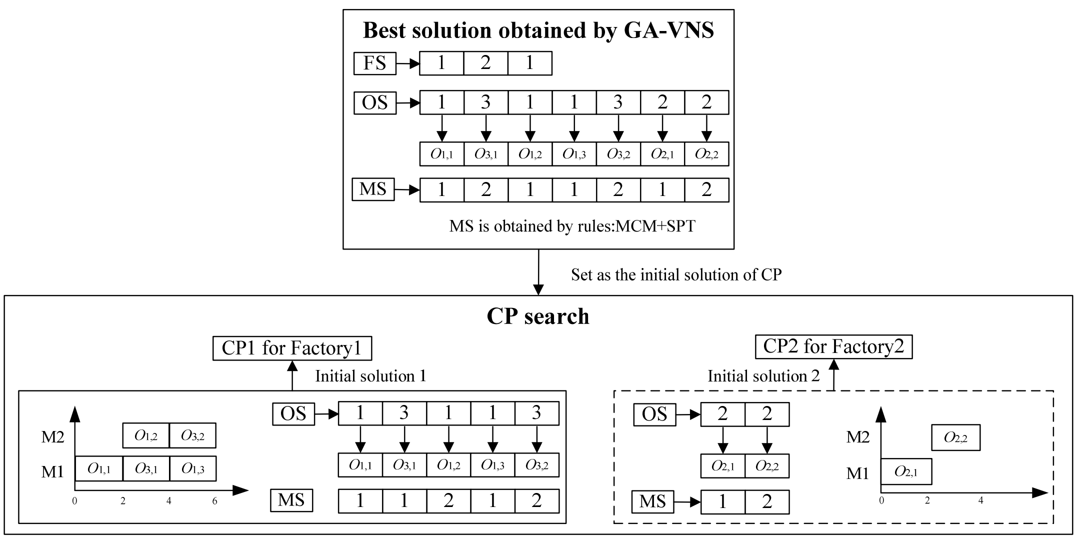


Fig. 9. Example of CP search

If the CP starts with specific initial solution sol, the following constraints should be added.

$$sol = new\ IloOplCPSolution() \tag{19}$$

$$cp.setStartingPoint(sol) \tag{20}$$

$$sol.setStart(op_{i,j}, Start_{i,j}), \forall i\in I, j\in J_i \tag{21}$$

$$sol.setPresent(mod_{i,j,MS_{i,j}}), \forall i\in I, j\in J_i \tag{22}$$

where, function (19) defines the initial solution sol. Function (20) constraints that the CP starts with the initial solution sol. Constraint (21) transmits the starting times of all operations in scheduling scheme obtained by decoding scheme to the variables of initial solution sol. Constraint (22) transmits the machine selections of all operations in scheduling scheme obtained by decoding scheme to the variables of initial solution sol. Specifically, function $IloOplCPSolution()$ is used to generate a solution of CP, function $setStartingPoint(sol)$ constraints the CP to start with a specific solution sol. Function $setStart(op_{i,j}, Start_{i,j})$ denotes that the starting time of $O_{i,j}$ must be $Start_{i,j}$, and $Start_{i,j}$ is the starting time of $O_{i,j}$ in scheduling scheme obtained by decoding scheme. Function $setPresent(mod_{i,j,MS_{i,j}})$ means that the machine $MS_{i,j}$ must be selected for $O_{i,j}$, and $MS_{i,j}$ is the selected machine of $O_{i,j}$ in scheduling scheme obtained by decoding scheme.

5. Experimental results

To prove the effectiveness of the MILP model and GA-VNS-CP, 23 instances with 2-4 factories are conducted (De Giovanni and Pezzella, 2010). All the proposed algorithms are run on a computer with a CPU of i7-10700 and RAM of 24 GB. All the algorithms are coded in C++ with Visual Studio 2019, and IBM CPLEX Studio IDE 12.7.1 is used to provide the CP and CPLEX solvers. The timelimit of all the algorithms are set to 2*N* seconds. For GA-VNS-CP, the runtime of GA-VNS and CP is all set to *N* seconds. For the comparison of meta-heuristic algorithms, each algorithm is executed 20 runs.

5.1 Effectiveness of MILP model

Tables 3-5 show the results of 2-4 factories for MILP model respectively. In Tables 3-5, "NB", "NC" and "NCT" represent the number of binary decision variables, the number of continuous decision variables and the number of constraints respectively. "Cmax" represents the obtained solution within the timelimit, and "Gap" represents the optimality gap of the obtained solution. If the Gap value is equal to 0, then the obtained solution is optimal (Meng and Zhang et al., 2019c). As can be seen from Tables 3-5, the MILP model can obtain 13, 18 and 22 optimal solutions out of 23 instances within the timelimit. Specifically, when the size of the instance increases, the solution space enlarges, and the NB, NC and NCT increases.

Table 3
Results of 2 factories for MILP model

Inst.	NB	NC	NCT	Cmax	Gap
la01	885	152	3673	413	0
la02	884	152	3585	394	0
la03	942	152	3933	349	0
la04	951	152	4045	369	0
la05	964	152	4213	380	0
la06	1856	227	7987	434	4.8
la07	1953	227	8607	413	9.0
la08	1918	227	8439	418	11.7
la09	1896	227	8307	451	15.3
la10	1960	227	8703	443	0
la11	3526	302	16429	605	31.7
la12	3401	302	15841	515	20.8
la13	3405	302	15657	583	34.5
la14	3359	302	15485	595	25.5
la15	3382	302	15641	601	37.1
la16	2060	302	8041	717	0
la17	1939	302	7497	646	0
la18	2042	302	7953	663	0
la19	1951	302	7617	617	0
la20	2019	302	7885	756	0
mt06	467	110	1857	47	0
mt10	1975	302	7685	655	0
mt20	3314	302	15137	596	35.1

Table 4
Results of 3 factories for MILP model

Inst.	NB	NC	NCT	Cmax	Gap
la01	991	203	5459	413	0
la02	988	203	5327	394	0
la03	1051	203	5879	349	0
la04	1062	203	6017	369	0
la05	1077	203	6269	380	0
la06	2012	303	11905	413	0
la07	2115	303	12835	376	0
la08	2078	303	12583	369	0
la09	2055	303	12385	382	0
la10	2122	303	12979	443	0
la11	3749	403	24543	479	13.8
la12	3620	403	23661	408	0
la13	3623	403	23385	414	7.7
la14	3576	403	23127	479	7.5
la15	3600	403	23361	434	12.9
la16	2271	403	11961	717	0
la17	2142	403	11145	646	0
la18	2251	403	11829	663	0
la19	2157	403	11325	617	0
la20	2228	403	11727	756	0
mt06	547	147	2749	47	0
mt10	2181	403	11427	655	0
mt20	3528	403	22605	440	12.0

Table 5
Results of 4 factories for MILP model

Inst.	NB	NC	NCT	Cmax	Gap
la01	1097	254	7245	413	0
la02	1092	254	7069	394	0
la03	1160	254	7765	349	0
la04	1173	254	7989	369	0
la05	1190	254	8325	380	0
la06	2168	379	15823	413	0
la07	2277	379	17063	376	0
la08	2238	379	16727	369	0
la09	2214	379	16463	382	0
la10	2284	379	17255	443	0
la11	3972	504	32657	436	0
la12	3839	504	31481	408	0
la13	3841	504	31113	397	3.8
la14	3793	504	30769	443	0
la15	3818	504	31081	378	0
la16	2482	504	15881	717	0
la17	2345	504	14793	646	0
la18	2460	504	15705	663	0
la19	2363	504	15033	617	0
la20	2437	504	15569	756	0
mt06	627	184	3641	47	0
mt10	2387	504	15169	655	0
mt20	3742	504	30073	387	0

5.2 Parameter calibration of GA-VNS-CP

As described above, there are four parameters, namely *Nt*, *Np*, *Pc* and *Pm* should be determined in GA-VNS-CP. Therefore, Taguchi method of design of experiment (DOE) is used, and the DOE is conducted for instance mt20 with 2 factories. For each parameter, three levels are tested. Specifically, three levels of [100, 300, 500] for *Nt*, three levels of [100, 300, 500] for *Np*, three levels of [0.7, 0.8, 0.9] for *Pc* and three levels of [0.1,0.15,0.2] of *Pm* are selected. For each combination, the test is repeated 20 times, and the mean value (Mean) is calculated and set as the response value. Table 6 shows the results of the DOE test. Fig. 10 shows the changing trend of mean value according to each parameter. Because our objective is minimizing makespan of DFJSP, the smaller value of the Mean is, the better the algorithm performs. Fig. 10 shows that the best combined parameter configuration is : *Nt*=500, *Np*=300, *Pc* =0.7 and *Pm*=0.2.

Table 6
Results of DOE test

Test	*Nt*	*Np*	*Pc*	*Pm*	*Mean*
1	100	100	0.7	0.1	528.0
2	100	300	0.8	0.15	527.3
3	100	500	0.9	0.2	527.5
4	300	100	0.8	0.2	529.4
5	300	300	0.9	0.1	529.9
6	300	500	0.7	0.15	527.8
7	500	100	0.9	0.15	524.6
8	500	300	0.7	0.2	522.0
9	500	500	0.8	0.1	526.0

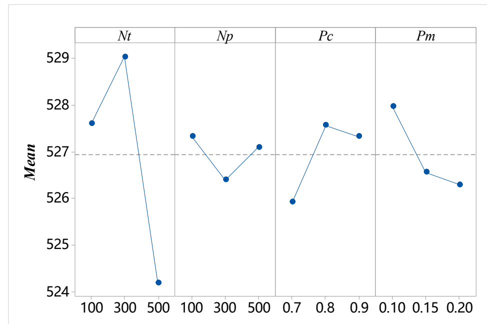


Fig. 10. Main effect plots of four parameters

5.3 Effectiveness of VNS and CP

To prove the effectiveness of VNS and CP, GA, GA-VSN and GA-VNS-CP are compared. Specifically, GA is without considering the VNS and CP, and GA-VSN only considers VNS. Table 7-8 show the comparison results of GA, GA-VSN and GA-VNS-CP for 2 and 3 factories respectively. In Tables 7-8, “Best” means the best solution obtained in 20 repeated times, “AV” shows the mean value of best solutions obtained in 20 repeated times, and the values in bold are the best among all algorithms.

Table 7
Comparison results of 2 factories for GA, GA-VSN and GA-VNS-CP

Inst.	GA		GA-VNS		GA-VNS-CP	
	Best	AV	Best	AV	Best	AV
la01	**413**	**413**	**413**	**413**	**413**	**413**
la02	**413**	**413**	**394**	**394**	**394**	**394**
la03	**349**	**349**	**349**	**349**	**349**	**349**
la04	**369**	**369**	**369**	**369**	**369**	**369**
la05	**380**	**380**	**380**	**380**	**380**	**380**
la06	**413**	430.9	**413**	428.6	**413**	**421.4**
la07	395	400.7	394	399.8	**386**	**392.7**
la08	403	415	400	409.7	**391**	**400.2**
la09	452	456.8	448	455.1	**436**	**447.4**
la10	**443**	**443**	**443**	**443**	**443**	**443**
la11	545	548.4	542	546.7	**538**	**544.8**
la12	474	479	474	477.9	**469**	**473.4**
la13	528	535.4	526	532.4	**521**	**531**
la14	542	548.6	541	546.2	**537**	**543**
la15	555	561.5	550	559.7	**549**	**555.8**
la16	**717**	**717**	**717**	**717**	**717**	**717**
la17	**646**	**646**	**646**	**646**	**646**	**646**
la18	**663**	**663**	**663**	**663**	**663**	**663**
la19	**617**	**618.8**	**617**	**617**	**617**	**617**
la20	**756**	**756**	**756**	**756**	**756**	**756**
mt06	**47**	**47**	**47**	**47**	**47**	**47**
mt10	**655**	**655**	**655**	**655**	**655**	**655**
mt20	529	537.2	524	530.7	**515**	**522.0**

As can be seen from Table 7, about Best, GA-VNS performs equal to and better than GA for 15 and 8 instances, and GA-VNS-CP performs equal to and better than GA-VNS for 14 and 9 instances. In terms of AV, GA-VNS performs equal to and better than GA for 13 and 10 instances, and GA-VNS-CP performs equal to and better than GA-VNS for 13 and 10 instances. Specifically, for 13 easy instances la01-05, 10, 16-20, mt06 and 10, all the algorithms can easily obtain the optimal solutions in each test. In other words, Best and AV for each of these 13 easy instances are the same. For 10 relatively hard instances la06-09, 11-15 and Mt20, different algorithms perform differently in terms of Best and AV. As can be seen from Table 8, with regard to Best, GA-VNS performs equal to and better than GA for 20 and 3 instances, and GA-VNS-CP performs equal to and better than GA-VNS for 22 and 1 instances. In terms of AV, GA-VNS performs equal to and better than GA for 18 and 5 instances, and GA-VNS-CP performs equal to and better than GA-VNS for 19 and 4 instances. Specifically, for 18 easy instances, namely la01-08, 10, 12, 14, 16-20, mt06 and 10, all the algorithms can easily obtain the optimal solutions in each test. In other words, Best and AV for each of these 18 easy instances are the same. For 5 relatively hard instances, namely la09, 11, 13, 15 and mt20, the values of Best and AV obtained by different algorithms are different. Moreover, Table 9 shows the paired-t test at 95% confidence level of AV values for 2-factory experiments. Obviously, the p-values are all less than 0.05. Specifically, the p-values of GA-VNS vs. GA, GA-VNS-CP vs. GA-VNS and GA-VNS-CP vs. GA are 0.024, 0.002 and 0.001 respectively. Therefore, GA-VNS-CP is statistically better than GA-VNS and GA, and GA-VNS is statistically better than GA. In conclusion, the VNS and CP are very effective in GA-VNS-CP.

Table 8
Comparison results of 3 factories for GA, GA-VSN and GA-VNS-CP

Inst.	GA		GA-VNS		GA-VNS-CP	
	Best	AV	Best	AV	Best	AV
la01	**413**	**413**	**413**	**413**	**413**	**413**
la02	**394**	**394**	**394**	**394**	**394**	**394**
la03	**349**	**349**	**349**	**349**	**349**	**349**
la04	**369**	**369**	**369**	**369**	**369**	**369**
la05	**380**	**380**	**380**	**380**	**380**	**380**
la06	**413**	**413**	**413**	**413**	**413**	**413**
la07	**376**	**376**	**376**	**376**	**376**	**376**
la08	**369**	**369**	**369**	**369**	**369**	**369**
la09	**382**	382.3	**382**	**382**	**382**	**382**
la10	**443**	**443**	**443**	**443**	**443**	**443**
la11	**413**	414.6	**413**	413.7	**413**	**413.1**
la12	**408**	**408**	**408**	**408**	**408**	**408**
la13	385	398.6	**382**	395.7	**382**	**391.9**
la14	**443**	**443**	**443**	**443**	**443**	**443**
la15	405	421.9	398	414.9	**387**	**405.4**
la16	**717**	**717**	**717**	**717**	**717**	**717**
la17	**646**	**646**	**646**	**646**	**646**	**646**
la18	**663**	**663**	**663**	**663**	**663**	**663**
la19	**617**	**617**	**617**	**617**	**617**	**617**
la20	**756**	**756**	**756**	**756**	**756**	**756**
mt06	**47**	**47**	**47**	**47**	**47**	**47**
mt10	**655**	**655**	**655**	**655**	**655**	**655**
mt20	397	406.1	**387**	403.6	**387**	**392.2**

Table 9
Paired t-test for the AV values of 2 factories

Comparison	p-value	Remark
GA-VNS vs. GA	0.024	<0.05
GA-VNS-CP vs. GA-VNS	0.002	<0.05
GA-VNS-CP vs. GA	0.001	<0.05

5.4 Effectiveness of GA-VNS-CP

To prove the superiority and effectiveness of GA-VNS-CP, it is compared with state-of-the-art algorithms, namely IGA(De Giovanni and Pezzella, 2010), GA_JS(Lu and Wu et al., 2015), GA_OP(Wu and Lin et al., 2017), CP(Meng and Zhang et al., 2020a) and IGWO(Li and Xie et al., 2022), by using the 23 benchmark instances of 2-4 factories. The values in bold are the best among all algorithms. The solutions with "*" are new best current solutions obtained by GA-VNS-CP. Best and AV represent the best and average makespan of several repetitions respectively. LB means the lower bound, and RPE represents the relative percent error of Best to LB. In the Appendix, we give detailed information of the improved best current solutions and some solutions for difficult benchmark instances.

Table 10 shows the comparison results of 2 factories. As can be seen from Table 10, GA-VNS-CP outperforms all the other algorithms in terms of Best and AV. Specifically, in terms of Best, IGA, GA_JS, GA_OP, CP, IGWO and GA-VNS-CP can obtain 12,13,13,18, 15 and 23 best solutions. In terms of mean RPE, the values of IGA, GA_JS, GA_OP, CP, IGWO and GA-VNS-CP are 12.4, 10.0,9.6, 9.2, 9.1 and 8.5. Most importantly, GA-VNS-CP obtains new best current solutions of 5 benchmark instances, namely la11,13-15 and mt20. More specifically, for la11, GA-VNS-CP improves the best current solution 539 obtained by CP to 538. For la13-15 and mt 20, the best current solutions 523, 538, 550 and 519 obtained by IGWO are improved by GA-VNS-CP to 521, 537, 549 and 515 respectively.

In terms of AV, GA-VNS-CP outperforms IGA, GA_JS, GA_OP and IGWO for la6-09, la11-15 and mt20. For the other instances, GA-VNS-CP performs no worse than IGA, GA_JS, GA_OP and IGWO. Moreover, Table 11 shows the paired-t test at 95% confidence level of AV values for 2-factory experiments. Obviously, the p-values are all less than 0.05. Specifically, the p-values of GA-VNS-CP vs. IGA, GA_JS, GA_OP and IGWO are 0.000, 0.001, 0.001 and 0.001 respectively. Therefore, GA-VNS-CP is statistically better than IGA, GA_JS, GA_OP and IGWO.

Table 10
Comparison results of 2 factories

Inst.	LB	IGA			GA_JS			GA_OP			CP		IGWO			GA-VNS-CP		
		Best	AV	RPE	Best	AV	RPE	Best	AV	RPE	Best	RPE	Best	AV	RPE	Best	AV	RPE
la01	413	**413**	**413**	**0**	**413**	**413**	**0**	**413**	**413**	**0**	**413**	**0**	**413**	**413**	**0**	**413**	**413**	**0**
la02	394	**394**	**394**	**0**	**394**	**394**	**0**	**394**	**394**	**0**	**394**	**0**	**394**	**394**	**0**	**394**	**394**	**0**
la03	349	**349**	**349**	**0**	**349**	**349**	**0**	**349**	**349**	**0**	**349**	**0**	**349**	**349**	**0**	**349**	**349**	**0**
la04	369	**369**	**369**	**0**	**369**	**369**	**0**	**369**	**369**	**0**	**369**	**0**	**369**	**369**	**0**	**369**	**369**	**0**
la05	380	**380**	**380**	**0**	**380**	**380**	**0**	**380**	**380**	**0**	**380**	**0**	**380**	**380**	**0**	**380**	**380**	**0**
la06	413	445	449.6	7.7	421	435.8	1.9	424	432.7	2.7	**413**	**0**	**413**	430.3	**0**	**413**	**421.4**	**0**
la07	376	412	419.2	9.6	396	408.5	5.3	390	403.6	3.7	**386**	**2.7**	389	401.6	3.5	**386**	**392.7**	**2.7**
la08	369	420	427.8	13.8	406	417.4	10	397	411.7	7.6	**391**	**6.0**	393	412.0	6.5	**391**	**400.2**	**6.0**
la09	382	469	474.6	22.8	447	459	17	444	455.7	16.2	**436**	**14.1**	439	457.3	14.9	**436**	**447.4**	**14.1**
la10	443	445	448.6	0.5	**443**	444.1	**0**	**443**	443.2	**0**	**443**	**0**	**443**	**443**	**0**	**443**	**443**	**0**
la11	413	570	571.6	38	548	557.1	32.7	541	549.9	31	545	32.0	539	548.8	30.5	**538***	**544.8**	**21.4**
la12	408	504	508	23.5	483	492.5	18.4	474	482.3	16.2	**469**	**15.0**	471	478.8	15.4	**469**	**473.4**	**15.0**
la13	382	542	552.2	41.9	530	538.4	38.7	529	538.1	38.5	525	37.4	523	533.3	36.9	**521***	**531.0**	**36.4**
la14	443	570	576	28.7	545	557.3	23	544	553.7	22.8	542	22.3	538	548.3	21.4	**537***	**543.0**	**21.4**
la15	378	584	588.8	54.5	554	568.7	46.6	554	566.6	46.6	555	46.8	550	561.9	45.5	**549***	**555.8**	**45.2**
la16	717	**717**	**717**	**0**	**717**	**717**	**0**	**717**	**717**	**0**	**717**	**0**	**717**	**717**	**0**	**717**	**717**	**0**
la17	646	**646**	**646**	**0**	**646**	**646**	**0**	**646**	**646**	**0**	**646**	**0**	**646**	**646**	**0**	**646**	**646**	**0**
la18	663	**663**	**663**	**0**	**663**	**663**	**0**	**663**	**663**	**0**	**663**	**0**	**663**	**663**	**0**	**663**	**663**	**0**
la19	617	**617**	617.2	**0**	**617**	622.1	**0**	**617**	617.5	**0**	**617**	**0**	**617**	**617**	**0**	**617**	**617**	**0**
la20	756	**756**	**756**	**0**	**756**	**756**	**0**	**756**	**756**	**0**	**756**	**0**	**756**	**756**	**0**	**756**	**756**	**0**
mt06	47	**47**	**47**	**0**	**47**	**47**	**0**	**47**	**47**	**0**	**47**	**0**	**47**	**47**	**0**	**47**	**47**	**0**
mt10	655	**655**	**655**	**0**	**655**	**655**	**0**	**655**	**655**	**0**	**655**	**0**	**655**	**655**	**0**	**655**	**655**	**0**
mt20	387	560	566	44.7	530	547.7	37	525	534.4	35.7	523	35.1	519	532.8	34.1	**515***	**522.0**	**33.1**
Ave.		12		12.4	13		10.0	13		9.6	18	9.2	15		9.1	**23**		**8.5**

Table 11
Paired t-test for the AV values of 2 factories

Comparison	p-value	Remark
GA-VNS-CP vs. IGA	0.000	<0.05
GA-VNS-CP vs. GA_JS	0.001	<0.05
GA-VNS-CP vs. GA_OP	0.001	<0.05
GA-VNS-CP vs. IGWO	0.001	<0.05

Table 12 shows the comparison results of 3 factories. As can be seen from Table 8, in terms of both Best and AV, GA-VNS-CP performs better than all the other algorithms. Specifically, in terms of Best, IGA, GA_JS, GA_OP, CP, IGWO and GA-VNS-CP can obtain 19, 20, 20, 22, 22 and 23 best solutions. In terms of mean RPE, the values of IGA, GA_JS, GA_OP, CP, IGWO and GA-VNS-CP are 2.0, 0.8, 0.7, 0.11, 0.21 and 0.10. Most importantly, GA-VNS-CP obtains new best current solution 387 of la15. In terms of AV, GA-VNS-CP outperforms IGA, GA_JS and GA_OP for la11,13,15 and mt20. For the other instances, GA-VNS-CP performs no worse than IGA, GA_JS and GA_OP.

Table 12
Comparison results of 3-factory DFJSP

Inst.	LB	IGA			GA_JS			GA_OP			CP		IGWO		GA-VNS-CP		
		MK	AV	RPE	MK	AV	RPE	MK	AV	RPE	MK	RPE	MK	RPE	MK	AV	RPE
la01	413	**413**	413	**0**	**413**	413	**0**	**413**	413	**0**	**413**	**0**	**413**	**0**	**413**	413	**0**
la02	394	**394**	394	**0**	**394**	394	**0**	**394**	394	**0**	**394**	**0**	**394**	**0**	**394**	394	**0**
la03	349	**349**	349	**0**	**349**	349	**0**	**349**	349	**0**	**349**	**0**	**349**	**0**	**349**	349	**0**
la04	369	**369**	369	**0**	**369**	369	**0**	**369**	369	**0**	**369**	**0**	**369**	**0**	**369**	369	**0**
la05	380	**380**	380	**0**	**380**	380	**0**	**380**	380	**0**	**380**	**0**	**380**	**0**	**380**	380	**0**
la06	413	**413**	413	**0**	**413**	413	**0**	**413**	413	**0**	**413**	**0**	**413**	**0**	**413**	413	**0**
la07	376	**376**	376	**0**	**376**	376	**0**	**376**	376	**0**	**376**	**0**	**376**	**0**	**376**	376	**0**
la08	369	**369**	369	**0**	**369**	369	**0**	**369**	369	**0**	**369**	**0**	**369**	**0**	**369**	369	**0**
la09	382	**382**	387.4	**0**	**382**	382	**0**	**382**	382	**0**	**382**	**0**	**382**	**0**	**382**	382	**0**
la10	443	**443**	443	**0**	**443**	443	**0**	**443**	443	**0**	**443**	**0**	**443**	**0**	**443**	443	**0**
la11	413	425	436.8	2.9	**413**	419.3	**0**	**413**	418	**0**	**413**	**0**	**413**	**0**	**413**	413.1	**0**
la12	408	**408**	408	**0**	**408**	408	**0**	**408**	408	**0**	**408**	**0**	**408**	**0**	**408**	408	**0**
la13	382	419	430.2	9.7	396	407.6	3.7	395	408.4	3.4	**382**	**0**	**382**	**0**	**382**	391.9	**0**
la14	443	**443**	448.8	**0**	**443**	443	**0**	**443**	443	**0**	**443**	**0**	**443**	**0**	**443**	443	**0**
la15	378	451	456	19.3	413	423.7	9.3	417	430	10.3	388	2.6	396	4.8	**387***	405.4	**2.4**
la16	717	**717**	717	**0**	**717**	717	**0**	**717**	717	**0**	**717**	**0**	**717**	**0**	**717**	717	**0**
la17	646	**646**	646	**0**	**646**	646	**0**	**646**	646	**0**	**646**	**0**	**646**	**0**	**646**	646	**0**
la18	663	**663**	663	**0**	**663**	663	**0**	**663**	663	**0**	**663**	**0**	**663**	**0**	**663**	663	**0**
la19	617	**617**	617	**0**	**617**	617	**0**	**617**	617	**0**	**617**	**0**	**617**	**0**	**617**	617	**0**
la20	756	**756**	756	**0**	**756**	756	**0**	**756**	756	**0**	**756**	**0**	**756**	**0**	**756**	756	**0**
mt06	47	**47**	47	**0**	**47**	47	**0**	**47**	47	**0**	**47**	**0**	**47**	**0**	**47**	47	**0**
mt10	655	**655**	655	**0**	**655**	655	**0**	**655**	655	**0**	**655**	**0**	**655**	**0**	**655**	655	**0**
mt20	387	439	442.6	13.4	407	415.8	5.2	397	412.7	2.6	**387**	**0**	**387**	**0**	**387**	392.2	**0**
Ave.		19		2.0	20		0.8	20		0.7	22	0.11	22	0.21	23		**0.10**

Table 13 shows the comparison results of 4 factories. As can be seen from Table 13, GA-VNS-CP performs no worse all the other algorithms in terms of Best and AV. Specifically, in terms of Best, IGA, GA_JS, GA_OP, CP, IGWO and GA-VNS-CP can obtain 22, 23, 23, 23, 23 and 23 best solutions. In terms of mean RPE, the values of IGA, GA_JS, GA_OP, CP, IGWO and GA-VNS-CP are 0.2, 0,0, 0, 0.21 and 0. In terms of AV, GA-VNS-CP can obtain the optimal solutions for all the instances in each repeat. However, for la15, IGA, GA_JS and GA_OP cannot guarantee to obtain the optimal solution in each repeat. Except for la15, IGA cannot guarantee to obtain the optimal solution in each repeat for la13 and mt20.

Table 13
Comparison results of 4-factory DFJSP

Inst.	LB	IGA			GA_JS			GA_OP			CP		IGWO		GA-VNS-CP		
		MK	AV	RPE	MK	AV	RPE	MK	AV	RPE	MK	RPE	MK	RPE	MK	AV	RPE
la01	413	**413**	**413**	**0**	**413**	**413**	**0**	**413**	**413**	**0**	**413**	0	**413**	**0**	**413**	**413**	**0**
la02	394	**394**	**394**	**0**	**394**	**394**	**0**	**394**	**394**	**0**	**394**	0	**394**	**0**	**394**	**394**	**0**
la03	349	**349**	**349**	**0**	**349**	**349**	**0**	**349**	**349**	**0**	**349**	0	**349**	**0**	**349**	**349**	**0**
la04	369	**369**	**369**	**0**	**369**	**369**	**0**	**369**	**369**	**0**	**369**	0	**369**	**0**	**369**	**369**	**0**
la05	380	**380**	**380**	**0**	**380**	**380**	**0**	**380**	**380**	**0**	**380**	0	**380**	**0**	**380**	**380**	**0**
la06	413	**413**	**413**	**0**	**413**	**413**	**0**	**413**	**413**	**0**	**413**	0	**413**	**0**	**413**	**413**	**0**
la07	376	**376**	**376**	**0**	**376**	**376**	**0**	**376**	**376**	**0**	**376**	0	**376**	**0**	**376**	**376**	**0**
la08	369	**369**	**369**	**0**	**369**	**369**	**0**	**369**	**369**	**0**	**369**	0	**369**	**0**	**369**	**369**	**0**
la09	382	**382**	**382**	**0**	**382**	**382**	**0**	**382**	**382**	**0**	**382**	0	**382**	**0**	**382**	**382**	**0**
la10	443	**443**	**443**	**0**	**443**	**443**	**0**	**443**	**443**	**0**	**443**	0	**443**	**0**	**443**	**443**	**0**
la11	413	**413**	**413**	**0**	**413**	**413**	**0**	**413**	**413**	**0**	**413**	0	**413**	**0**	**413**	**413**	**0**
la12	408	**408**	**408**	**0**	**408**	**408**	**0**	**408**	**408**	**0**	**408**	0	**408**	**0**	**408**	**408**	**0**
la13	382	**382**	386	**0**	**382**	**382**	**0**	**382**	**382**	**0**	**382**	0	**382**	**0**	**382**	**382**	**0**
la14	443	**443**	**443**	**0**	**443**	**443**	**0**	**443**	**443**	**0**	**443**	0	**443**	**0**	**443**	**443**	**0**
la15	378	397	402	5.0	**378**	381.9	**0**	**378**	385.8	**0**	**378**	0	**378**	**0**	**378**	**378**	**0**
la16	717	**717**	**717**	**0**	**717**	**717**	**0**	**717**	**717**	**0**	**717**	0	**717**	**0**	**717**	**717**	**0**
la17	646	**646**	**646**	**0**	**646**	**646**	**0**	**646**	**646**	**0**	**646**	0	**646**	**0**	**646**	**646**	**0**
la18	663	**663**	**663**	**0**	**663**	**663**	**0**	**663**	**663**	**0**	**663**	0	**663**	**0**	**663**	**663**	**0**
la19	617	**617**	**617**	**0**	**617**	**617**	**0**	**617**	**617**	**0**	**617**	0	**617**	**0**	**617**	**617**	**0**
la20	756	**756**	**756**	**0**	**756**	**756**	**0**	**756**	**756**	**0**	**756**	0	**756**	**0**	**756**	**756**	**0**
mt06	47	**47**	**47**	**0**	**47**	**47**	**0**	**47**	**47**	**0**	**47**	0	**47**	**0**	**47**	**47**	**0**
mt10	655	**655**	**655**	**0**	**655**	**655**	**0**	**655**	**655**	**0**	**655**	0	**655**	**0**	**655**	**655**	**0**
mt20	387	**387**	388.4	**0**	**387**	**387**	**0**	**387**	**387**	**0**	**387**	0	**387**	**0**	**387**	**387**	**0**
Ave.		22		0.2	23		**0**	23		**0**	23	0		**0**	23		**0**

6. Conclusions and future study

This paper designs a novel MILP model and a hybrid algorithm GA-VNS-CP of GA, VNS and CP search with minimizing the makespan of DFJSP. The effectiveness of the MILP model is verified by the CPLEX solver. The GA-VNS-CP evolves with two stages, namely GA-VNS and CP search. Experimental results show that the VNS and CP are effective in improving the optimization ability of GA. More importantly, the proposed GA-VNS-CP outperforms the existing algorithms and finds the 6 best new solutions for benchmark instances. Specifically, the new best solutions 538, 521, 537, 549 and 515 for la11, 13-15 and mt20 with 2 factories are obtained, and the new best solution 387 for mt20 with 3 factories is obtained.

In future research, we will try to solve DFJSP with novel objectives, such as energy consumption, total tardiness, and multi-objectives. Moreover, preventive maintenance and machine life will be considered.

Acknowledgements

This research is supported by the Funds for National Natural Science Foundation of China [grant numbers 52205529 and 62303204], the Natural Science Foundation of Shandong Province [grant numbers ZR2021QE195 and ZR2021QF036], the Youth Innovation Team Program of Shandong Higher Education Institution [2023KJ206], the Guangyue Youth Scholar Innovation Talent Program support received from Liaocheng University [LCUGYTD2022-03] and the National Undergraduate Innovation and Entrepreneurship Training Program [202310447185].

References

Bagheri Rad, N., & Behnamian, J. (2023). Multi-objective collaborative job shop scheduling in a dynamic environment: Non-dominated sorting memetic algorithm. *Journal of Ambient Intelligence and Humanized Computing, 14*(3), 2657-2671.

Bukchin, Y., & Raviv, T. (2018). Constraint programming for solving various assembly line balancing problems. *Omega, 78*, 57-68.

Chan, F. T., Chung, S. H., Chan, L. Y., Finke, G., & Tiwari, M. K. (2006). Solving distributed FMS scheduling problems subject to maintenance: Genetic algorithms approach. *Robotics and Computer-Integrated Manufacturing, 22*(5-6), 493-504.

Chan, F. T., Chung, S. H., & Chan, P. L. Y. (2005). An adaptive genetic algorithm with dominated genes for distributed scheduling problems. *Expert Systems with Applications, 29*(2), 364-371.

Chang, H. C., & Liu, T. K. (2017). Optimisation of distributed manufacturing flexible job shop scheduling by using hybrid genetic algorithms. *Journal of Intelligent Manufacturing, 28*, 1973-1986.

Chung, S. H., Chan, F. T., & Chan, H. K. (2009). A modified genetic algorithm approach for scheduling of perfect maintenance in distributed production scheduling. *Engineering Applications of Artificial Intelligence, 22*(7), 1005-1014.

Dai, L. L., Pan, Q. K., Miao, Z. H., Suganthan, P. N., & Gao, K. Z. (2023). Multi-Objective Multi-Picking-Robot Task Allocation: Mathematical Model and Discrete Artificial Bee Colony Algorithm. IEEE Transactions on Intelligent Transportation Systems.

De Abreu, L. R., Araújo, K. A. G., de Athayde Prata, B., Nagano, M. S., & Moccellin, J. V. (2022). A new variable neighbourhood search with a constraint programming search strategy for the open shop scheduling problem with operation repetitions. *Engineering Optimization, 54*(9), 1563-1582.

De Giovanni, L., & Pezzella, F. (2010). An improved genetic algorithm for the distributed and flexible job-shop scheduling problem. *European journal of operational research, 200*(2), 395-408.

Du, Y., Li, J. Q., Luo, C., & Meng, L. L. (2021). A hybrid estimation of distribution algorithm for distributed flexible job shop scheduling with crane transportations. *Swarm and Evolutionary Computation, 62*, 100861.

Gedik, R., Kalathia, D., Egilmez, G., & Kirac, E. (2018). A constraint programming approach for solving unrelated parallel machine scheduling problem. *Computers & Industrial Engineering, 121*, 139-149.

Ham, A. M., & Cakici, E. (2016). Flexible job shop scheduling problem with parallel batch processing machines: MIP and CP approaches. *Computers & Industrial Engineering, 102,* 160-165.

Ham, A., Park, M. J., & Kim, K. M. (2021). Energy-aware flexible job shop scheduling using mixed integer programming and constraint programming. *Mathematical Problems in Engineering, 2021*, 1-12.

He, X., Pan, Q. K., Gao, L., Neufeld, J. S., & Gupta, J. N. (2024). Historical information based iterated greedy algorithm for distributed flowshop group scheduling problem with sequence-dependent setup times. *Omega, 123*, 102997.

Li, J., Gu, X., Zhang, Y., & Zhou, X. (2022). Distributed flexible job-shop scheduling problem based on hybrid chemical reaction optimization algorithm. *Complex System Modeling and Simulation, 2*(2), 156-173.

Jia, H. Z., Nee, A. Y., Fuh, J. Y., & Zhang, Y. F. (2003). A modified genetic algorithm for distributed scheduling problems. *Journal of Intelligent Manufacturing, 14*, 351-362.

Jia, H. Z., Fuh, J. Y., Nee, A. Y., & Zhang, Y. F. (2002). Web-based multi-functional scheduling system for a distributed manufacturing environment. *Concurrent Engineering, 10*(1), 27-39.

Karimi, H., Rahmati, S. H. A., & Zandieh, M. (2012). An efficient knowledge-based algorithm for the flexible job shop scheduling problem. *Knowledge-Based Systems, 36*, 236-244.

Li, J. Q., Duan, P., Cao, J., Lin, X. P., & Han, Y. Y. (2018). A hybrid Pareto-based tabu search for the distributed flexible job shop scheduling problem with E/T criteria. *IEEE Access, 6*, 58883-58897.

Li, X., Xie, J., Ma, Q., Gao, L., & Li, P. (2022). Improved gray wolf optimizer for distributed flexible job shop scheduling problem. *Science China Technological Sciences, 65*(9), 2105-2115.

Lin, C. S., Lee, I. L., & Wu, M. C. (2019). Merits of using chromosome representations and shadow chromosomes in genetic algorithms for solving scheduling problems. *Robotics and computer-integrated manufacturing, 58*, 196-207.

Liu, T. K., Chen, Y. P., & Chou, J. H. (2014). Solving distributed and flexible job-shop scheduling problems for a real-world fastener manufacturer. *IEEE Access, 2,* 1598-1606.

Lu, P. H., Wu, M. C., Tan, H., Peng, Y. H., & Chen, C. F. (2018). A genetic algorithm embedded with a concise chromosome representation for distributed and flexible job-shop scheduling problems. *Journal of Intelligent Manufacturing, 29,* 19-34.

Luo, Q., Deng, Q., Gong, G., Zhang, L., Han, W., & Li, K. (2020). An efficient memetic algorithm for distributed flexible job shop scheduling problem with transfers. *Expert Systems with Applications, 160,* 113721.

Luo, Q., Deng, Q., Gong, G., Guo, X., & Liu, X. (2022). A distributed flexible job shop scheduling problem considering worker arrangement using an improved memetic algorithm. *Expert Systems with Applications, 207*, 117984.

Marzouki, B., Driss, O. B., & Ghédira, K. (2018). Solving distributed and flexible job shop scheduling problem using a chemical reaction optimization metaheuristic. *Procedia Computer Science, 126,* 1424-1433.

Meng, L., Lu, C., Zhang, B., Ren, Y., Lv, C., Sang, H., ... & Zhang, C. (2021). Constraint programing for solving four complex flexible shop scheduling problems. *IET Collaborative Intelligent Manufacturing, 3*(2), 147-160.

Meng, L., Zhang, C., Zhang, B., & Ren, Y. (2019a). Mathematical modeling and optimization of energy-conscious flexible job shop scheduling problem with worker flexibility. *IEEE Access, 7*, 68043-68059.

Meng, L., Zhang, C., Shao, X., Ren, Y., & Ren, C. (2019b). Mathematical modelling and optimisation of energy-conscious hybrid flow shop scheduling problem with unrelated parallel machines. *International Journal of Production Research, 57*(4), 1119-1145.

Meng, L., Zhang, C., Shao, X., & Ren, Y. (2019c). MILP models for energy-aware flexible job shop scheduling problem. *Journal of cleaner production, 210*, 710-723.

Meng, L., Zhang, C., Ren, Y., Zhang, B., & Lv, C. (2020a). Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem. *Computers & industrial engineering, 142*, 106347.

Meng, L., Zhang, C., Shao, X., Zhang, B., Ren, Y., & Lin, W. (2020b). More MILP models for hybrid flow shop scheduling problem and its extended problems. *International journal of production research, 58*(13), 3905-3930.

Meng, L., Zhang, C., Zhang, B., Gao, K., Ren, Y., & Sang, H. (2023). MILP modeling and optimization of multi-objective flexible job shop scheduling problem with controllable processing times. *Swarm and Evolutionary Computation, 82*, 101374.

Meng, L., Gao, K., Ren, Y., Zhang, B., Sang, H., & Chaoyong, Z. (2022). Novel MILP and CP models for distributed hybrid flowshop scheduling problem with sequence-dependent setup times. *Swarm and Evolutionary Computation, 71*, 101058.

Meng, L., Duan, P., Gao, K., Zhang, B., Zou, W., Han, Y., & Zhang, C. (2024). MIP modeling of energy-conscious FJSP and its extended problems: From simplicity to complexity. *Expert Systems with Applications, 241*, 122594.

Meng, L., Cheng, W., Zhang, B., Zou, W., Fang, W., & Duan, P. (2023). An Improved Genetic Algorithm for Solving the Multi-AGV Flexible Job Shop Scheduling Problem. *Sensors, 23*(8), 3815.

Meng, L., Ren, Y., Zhang, B., Li, J. Q., Sang, H., & Zhang, C. (2020). MILP modeling and optimization of energy-efficient distributed flexible job shop scheduling problem. *IEEE Access, 8*, 191191-191203.

Naderi, B., & Azab, A. (2014). Modeling and heuristics for scheduling of distributed job shops. *Expert Systems with Applications, 41*(17), 7754-7763.

Şahman, M. A. (2021). A discrete spotted hyena optimizer for solving distributed job shop scheduling problems. *Applied Soft Computing, 106*, 107349.

Sang, Y., & Tan, J. (2022). Intelligent factory many-objective distributed flexible job shop collaborative scheduling method. *Computers & Industrial Engineering, 164*, 107884.

Tang, H., Fang, B., Liu, R., Li, Y., & Guo, S. (2022). A hybrid teaching and learning-based optimization algorithm for distributed sand casting job-shop scheduling problem. *Applied Soft Computing, 120,* 108694.

Wang, L., & Peng, Z. P. (2020). Solving energy-efficient distributed job shop scheduling via multi-objective evolutionary algorithm with decomposition. *Swarm and Evolutionary Computation, 58*, 100745.

Wu, M. C., Lin, C. S., Lin, C. H., & Chen, C. F. (2017). Effects of different chromosome representations in developing genetic algorithms to solve DFJS scheduling problems. *Computers & Operations Research, 80*, 101-112.

Wu, X., Liu, X., & Zhao, N. (2019). An improved differential evolution algorithm for solving a distributed assembly flexible job shop scheduling problem. *Memetic Computing, 11*, 335-355.

Xie, J., Gao, L., Pan, Q. K., & Tasgetiren, M. F. (2019). An effective multi-objective artificial bee colony algorithm for energy efficient distributed job shop scheduling. *Procedia Manufacturing, 39*, 1194-1203.

Xu, W., Hu, Y., Luo, W., Wang, L., & Wu, R. (2021). A multi-objective scheduling method for distributed and flexible job shop based on hybrid genetic algorithm and tabu search considering operation outsourcing and carbon emission. *Computers & Industrial Engineering, 157*, 107318.

Zhang, L., Yu, C., & Wong, T. N. (2021). A graph-based constraint programming approach for the integrated process planning and scheduling problem. *Computers & Operations Research, 131*, 105282.

Zhu, K., Gong, G., Peng, N., Zhang, L., Huang, D., Luo, Q., & Li, X. (2023). Dynamic distributed flexible job-shop scheduling problem considering operation inspection. *Expert Systems with Applications, 224,* 119840.

Ziaee, M. (2014). A heuristic algorithm for the distributed and flexible job-shop scheduling problem. *The Journal of Supercomputing, 67*, 69-83.

Appendix

la07：

factory_number: 2
makespan: 386
operation sequence:
12 2 1 8 0 5 12 2 9 0 8 2 1 12 5 1 9 8 0 5 1 9 2 0 12 8 9 1 5 9 5 8 2 12 0 13 11 14 3 7 10 4 13 3 14 7 11 13 11 4 6 3 10 14 3 11 13 7 6 3 10 14 6 4 14 10 7 6 13 4 6 11 10 7 4
machine selection:
0 4 1 0 2 2 1 3 3 1 3 1 2 1 4 3 1 4 3 3 3 1 1 2 2 1 4 4 4 1 2 4 4 4 4 2 3 0 0 0 4 0 2 3 3 0 0 0 4 2 4 0 1 2 3 0 1 0 4 2 1 3 3 2 0 4 0 3 2 1 1 2 1 2 3
factory selection:
0 0 0 1 1 0 1 1 0 0 1 1 0 1 1

la08:

factory_number: 2
makespan: 391
operation sequence:
14 11 9 13 6 6 5 3 13 11 9 14 6 11 5 9 13 3 14 11 9 5 6 5 13 14 11 14 3 5 3 6 13 9 3 10 0 7 2 8 4 2 1 2 7 12 4 8 10 0 7 12 1 8 1 10 7 8 4 12 2 4 0 1 2 0 7 1 10 8 12 4 10 0 12
machine selection:
3 2 0 0 1 2 1 1 4 2 2 0 0 4 2 3 4 0 3 3 2 4 3 1 1 4 3 2 4 0 4 1 0 2 2 1 2 0 0 4 4 3 4 2 1 1 2 2 0 1 0 1 4 3 2 0 0 1 3 1 1 3 3 3 3 3 1 0 3 1 2 4 1 1 4
factory selection:
1 1 1 0 1 0 0 1 1 0 1 0 1 0 0

la09:

factory_number: 2
makespan: 436
operation sequence:
5 1 11 9 12 7 14 12 6 11 5 9 7 14 11 6 1 5 9 12 6 14 7 14 1 11 5 12 6 9 1 7 14 5 6 7 12 9 11 1 3 0 2 10 8 4 13 3 10 2 8 13 0 3 10 4 13 13 2 10 8 4 3 0 4 8 10 13 0 2 8 4 3 2 0
machine selection:

1 3 2 0 4 3 2 2 3 0 2 0 1 2 0 0 1 2 4 2 0 4 2 3 1 0 3 4 1 4 3 2 1 1 0 1 1 3 4 3 4 4 3 0 3 2 4 3 1 1
3 2 1 0 1 1 0 3 0 4 4 2 0 2 2 3 3 0 0 4 4 0 2 4 3
factory selection:
1 0 1 1 1 0 0 0 1 0 1 0 0 1 0

la11:
factory_number: 2
makespan: 538
operation sequence:
6 4 11 9 1 14 8 1 19 11 17 14 8 6 15 19 9 8 14 4 6 15 1 19 17 4 9 4 6 17 14 15 19 1 11 17 9 14 15 6
19 8 17 11 9 15 4 11 1 8 2 7 18 3 13 10 2 0 3 16 12 10 18 5 2 0 16 7 18 5 13 0 12 3 13 5 16 10 12 16
7 10 18 2 7 0 3 5 13 10 18 12 3 0 5 13 7 2 16 12
machine selection:
2 1 0 0 4 2 2 4 0 2 0 3 3 4 3 2 3 4 3 1 1 4 1 1 4 4 4 0 2 2 0 4 2 1 0 4 3 1 2 0 4 3 0 2 1 3 1 0 4 1
0 2 2 4 1 4 2 0 1 3 1 3 2 0 1 3 1 1 0 3 4 0 3 2 1 3 0 4 3 0 4 0 3 3 2 1 0 2 2 4 1 0 2 1 3 4 2 3 3 3
factory selection:
1 0 1 1 0 1 0 1 0 0 1 0 1 1 0 0 1 0 1 0

la12:
factory_number: 2
makespan: 469
operation sequence:
7 8 2 12 3 8 13 7 5 2 3 7 0 14 8 0 3 12 5 13 8 14 2 12 3 7 19 8 0 5 19 13 17 5 2 7 19 17 12 14 2 17
19 12 0 5 13 17 19 14 3 0 13 14 17 18 6 15 4 16 15 11 6 16 1 18 16 4 11 10 9 18 1 15 16 4 10 9 11 4
15 18 11 9 10 9 1 15 4 1 6 18 9 1 11 10 6 16 10 6
machine selection:
1 0 4 0 3 3 2 4 1 4 4 0 1 3 3 3 3 2 0 3 2 4 0 0 4 4 1 2 4 1 0 2 4 2 0 4 2 1 1 2 2 1 2 3 0 4 3 4 1
4 2 3 2 2 2 3 1 2 0 1 2 0 4 2 0 1 0 3 1 0 3 4 1 2 4 4 0 4 4 2 1 1 3 4 1 3 0 1 3 0 4 1 1 3 4 1 4 4 1
factory selection:
0 1 0 0 1 0 1 0 0 1 1 1 0 0 0 1 1 0 1 0

la13:
factory_number: 2
makespan: 521
operation sequence:
19 2 17 16 14 5 16 13 14 17 16 6 5 19 13 14 7 17 2 16 5 7 13 6 15 2 13 7 19 5 16 17 14 7 19 15 15 2
14 15 17 13 6 6 2 6 15 7 19 5 18 0 10 11 12 12 0 8 3 10 1 4 18 3 0 9 4 8 1 3 18 12 11 3 1 10 9 4 11
0 8 12 1 9 3 10 9 11 18 8 1 12 0 4 18 9 11 10 4 8
machine selection:
3 0 2 0 2 2 0 4 3 4 3 1 3 3 4 2 0 3 0 3 3 3 3 0 0 1 3 2 4 4 1 1 4 4 0 4 3 1 2 2 3 1 2 3 1 4 0 1 4 1
2 4 1 2 1 0 1 4 1 3 1 1 3 4 4 2 4 1 0 3 4 4 3 1 1 4 3 3 0 1 1 3 3 4 2 2 0 0 0 2 4 1 2 0 4 0 2 2 4 3
factory selection:
1 1 0 1 1 0 0 0 1 1 1 1 1 0 0 0 0 0 1 0

la14:
factory_number: 2
makespan: 537
operation sequence:
16 9 13 18 14 2 17 14 13 11 18 14 8 13 16 17 11 10 17 13 2 9 14 17 16 8 10 10 2 13 16 18 11 8 16 17
2 8 10 11 9 9 2 9 8 10 14 18 11 18 3 7 5 4 15 1 12 4 7 15 5 6 1 7 0 19 0 6 4 19 15 1 5 0 5 12 6 3 19
15 5 1 0 3 19 12 6 15 19 3 0 6 4 12 12 1 7 4 3 7
machine selection:
3 4 2 0 1 1 2 0 3 3 0 0 3 4 0 0 3 4 3 4 2 3 3 0 1 1 4 4 1 1 1 0 4 1 3 3 2 1 0 0 3 2 4 0 4 0 2 4 4 1
4 0 0 1 2 2 1 0 3 3 0 3 0 3 2 4 4 1 3 1 2 2 1 3 3 4 0 2 0 2 1 0 4 4 2 3 2 2 1 3 3 4 2 3 1 3 1 2 2 4
factory selection:
1 1 0 1 1 1 1 1 0 0 0 0 1 0 0 1 0 0 0 1

la15:
factory_number: 2
makespan: 549
operation sequence:
15 3 9 6 14 13 16 0 10 16 15 16 0 13 3 14 4 1 6 0 3 16 14 13 1 9 4 10 14 15 0 6 3 15 10 13 3 0 16 10
1 9 15 1 4 9 6 13 9 4 10 14 4 1 6 7 12 2 11 8 12 11 19 2 7 11 17 18 12 5 7 17 11 19 8 5 18 2 7 12 19
17 5 2 8 12 18 19 8 7 5 18 5 17 2 19 18 17 8 11
machine selection:
0 2 1 0 4 0 3 4 4 1 1 4 2 0 0 2 4 3 4 4 2 1 1 2 4 0 3 1 2 2 3 4 1 2 3 0 0 3 4 3 4 0 3 1 4 0 2 1 2 3
0 4 2 0 1 2 2 2 2 3 3 3 1 3 4 2 3 2 2 0 4 0 4 3 2 1 1 4 0 3 0 3 3 0 3 3 4 2 1 1 4 1 2 4 4 1 2 4 0 3

factory selection:
0 0 1 0 0 1 0 1 1 0 0 1 1 0 0 0 0 1 1 1

mt20:
factory_number: 2
makespan: 515
operation sequence:
12 18 3 7 10 16 16 2 10 16 1 4 17 4 3 1 18 7 12 17 16 1 3 2 18 4 3 12 2 17 18 10 7 12 7 4 2 1 17 7 16 3 4 10 1 17 18 2 12 10 15 13 14 9 5 6 19 0 19 8 0 0 15 6 13 9 14 0 11 9 14 19 8 15 0 13 6 9 19 14 5 11 13 15 9 8 11 11 5 6 15 13 19 8 5 14 11 6 8 5
machine selection:
0 1 1 3 4 2 1 1 3 0 4 4 1 4 2 1 3 4 2 3 0 1 0 1 4 4 1 0 2 2 1 3 2 3 0 2 4 0 2 4 0 3 2 0 1 3 1 0 3 4 4 3 0 2 0 1 0 1 1 4 0 2 4 0 4 2 2 1 3 4 0 0 4 2 3 1 4 0 0 2 2 2 1 3 1 0 0 4 2 3 3 2 2 3 1 0 1 2 4 1
factory selection:
1 0 0 0 0 1 1 0 1 1 0 1 0 1 1 1 0 0 0 1

la15:
factory_number: 3
makespan: 387
operation sequence:
7 5 17 16 18 11 18 16 4 16 7 5 17 16 11 7 18 5 4 18 17 16 7 4 11 5 17 7 5 4 18 17 4 11 11 19 2 15 12 8 12 6 2 8 19 15 12 6 19 2 19 12 6 8 2 15 8 2 15 12 19 15 6 8 6 0 14 3 9 13 10 0 1 9 13 10 14 0 3 10 1 13 3 14 9 13 14 10 0 1 9 13 1 3 10 9 0 14 3 1
machine selection:
0 2 1 0 4 0 3 1 0 1 1 4 2 2 1 2 4 0 4 1 2 0 1 1 4 4 3 4 2 2 1 2 1 2 0 0 0 3 0 3 4 0 4 2 1 3 2 1 2 3 0 4 2 4 1 2 0 0 1 3 3 3 1 3 4 4 3 2 2 2 1 0 4 3 4 2 3 4 0 3 1 3 3 1 3 3 4 2 4 1 2 1 2 1 4 0 2 4 0 3
factory selection:
2 2 1 2 0 0 1 0 1 2 2 0 1 2 2 1 0 0 0 1