Int j simul model 23 (2024) 1, 137-148
Original scientific paper

COLLABORATIVE PRODUCTION SCHEDULING WITH MULTI-ENTERPRISE IDLE CAPACITY SHARING

Liao, Y. G.; Zhang, H.[#]; Ren, N. & Wang, T. Y.
School of Economics and Management, Jiangsu University of Science and Technology, Zhenjiang 212100, China
E-Mail: haozh168@aliyun.com ([#] Corresponding author)

Abstract
This paper proposes a digital twin enhanced approach for optimizing collaborative production scheduling in multi-enterprise manufacturing systems. A multi-objective model is developed incorporating practical constraints such as limited time windows, different production capacities, and transportation considerations. To solve the model, an Improved Non-dominated Sorting Genetic Algorithm (INSGA-II) is designed with specialized operators and strategies. The digital twin simulation is enriched with Multi-objective Decision Making based on Interaction Structures (MDIS) to obtain higher-quality solutions. Experiments demonstrate that the MDIS digital twin approach reduces manufacturing lead times and costs while improving utilization and quality compared to standard methods. This research provides an effective optimization framework to leverage cloud manufacturing resources across organizations.
(Received in July 2023, accepted in November 2023. This paper was with the authors 2 months for 2 revisions.)

Key Words: Capacity Sharing, Idle Time Window, Differential Manufacturing Services, Collaborative Production Scheduling, INSGA-II

1. INTRODUCTION

Shared manufacturing is a new production model based on cloud manufacturing technology that shares idle manufacturing capacity. In recent years, more and more cloud manufacturing platforms have provided equipment-sharing services for manufacturing enterprises, which has become a new highlight in the development of manufacturing. For example, China's 1688 Taofactory and Aerospace Cloud platforms provide equipment leasing services for manufacturing enterprises, and these services largely reduce the costs of equipment user enterprises while compensating for the lack of production capacity in small and medium-sized enterprises due to the surge in orders.

Cloud manufacturing under capacity sharing has evolved from mass production to mass personalized production [1]. The manufacturing characteristics of product personalization, manufacturing capacity socialization, and interconnection integration are gradually emerging. These make production scheduling in a cloud manufacturing environment more complex and challenging to solve. While traditional centralized scheduling focuses on resource allocation within a single production unit, scheduling for capacity sharing requires consideration of cooperation and coordination among multiple participants. However, most of the enterprises involved in collaboration are heterogeneously distributed, the production capacity and technology of different manufacturing enterprises have great variability, the tasks and demands of customers become complex and diverse, and many enterprises share only some remaining capacity. Including the customer's reputation, the difficulty of platform scheduling, and the dynamic changeability of production resources, the cloud manufacturing under capacity sharing has many urgent problems to be solved. It is clear that traditional scheduling methods are no longer able to solve the existing problems. There is a desire to develop a modern mass customization production model that is customer demand-driven and well able to handle the important manufacturing requirements such as multiple varieties, small lot sizes, rapidity, flexibility and dynamics [2]. In order to satisfy this model, a suitable scheduling model and

more powerful optimization methods need to be designed to dynamically and flexibly adjust the production plan according to various demands.

In recent years, the scheduling problem under capacity sharing has received attention from many researchers. Xu et al. [3] studied the identical parallel machine scheduling problem with due dates to deadlines, considering both order sharing and the time value of money. Koulouris and Georgiadis [4] Calculate the scheduling problem for the minimum cycle time and all feasible cycle time ranges in the context of periodic scheduling for shared manufacturing. Chen et al. [5] put the quality of service (QOS) at the centre and proposed a model to optimize QOS performance and reduce QOS risk. Aghamohammadzadeh et al. [6] combined both logistics and production to minimize cloud entropy, as the goal of the model to simplify the complexity of the order schedule on the platform. Tong and Zhu [7] developed a customer-oriented satisfaction scheduling model targeting timeliness, taking into account available capacity time windows, adjustable processing rates, and varying processing energy consumption. Wang and Suo [8] focused on a production logistics scheduling model that considers energy consumption and carbon emissions. Chen and Zhao [9] developed a multi-objective scheduling model considering realistic constraints such as multi-cycle control and work transport time. Huo and Wang [10] proposed a dynamic scheduling method incorporating digital twins to address the impact of emergencies in workpiece production on the production schedule. The above literature has examined different aspects; however, most researchers have overlooked a key issue. The capacity available to the enterprise in a capacity sharing environment is for a few finite windows of time, not all of the time, which is one of the most distinctive features of shared manufacturing. In addition, production performance varies greatly from enterprise to enterprise, such as time, cost, energy consumption, and quality. And as scheduling for multi-enterprise collaboration, the distance factor between enterprises must also be taken into account.

For solving, intelligent optimization algorithms are widely used in production scheduling problems because they have significant advantages in solving complex nonlinear optimization problems by combining their own search mechanism and efficient neighbourhood search. Among them, the non-dominated sorting genetic algorithm (NSGA-II) proposed by Deb et al. [11] has the advantages of fast operation speed and good convergence due to the use of a fast non-dominated sorting method, an elite retention strategy, and a congestion comparison operator. After this, many researchers have improved the local search capability of NSGA-II by designing specific neighbourhood search strategies [12, 13]. Some researchers have also made some improvements to the crossover and mutation operations of NSGA-II to address specific scheduling models [14, 15]. Of course, there are some other improvements, such as Li et al. [16] included two heuristic selection strategies in NSGA-II for solving the multi-row shop layout problem. Liu et al. [17] inserted a local search strategy for obtaining intersection and sparse points based on non-dominated sorting results into the NSGA-II to solve a specific scheduling problem. It can be found that in the field of production scheduling, researchers are keen to improve the convergence and diversity of NSGA-II by improving genetic operators and elite retention strategies, or by adopting local search strategies and global improvement mechanisms. However, few studies have focused on population initialization rules and adaptive improvement, whereas the quality of the population has a significant impact on the performance of the algorithm, and the constant cross-variance probability limits the exploration potential of the algorithm. In addition, truly effective local search strategies often need to be designed according to the characteristics of the problem.

In summary, in order to bridge the gap in existing research, this paper proposes a Multi-objective Decision Making based on Interaction Structures approach to optimize the scheduling of spare capacity sharing across multiple enterprises. The model focuses on the reality of the manufacturing time window available to enterprises while considering the diversity of customer demand, manufacturing capacity, processing time, processing cost and processing quality

variability, and the distance between collaborating enterprises. For the solution method, we have made some improvements to NSGA-II in order to reasonably balance its global exploration and local search capabilities and to exploit the potential of the algorithm. Firstly, the initialization strategy is designed according to the heuristic rule to improve the quality of the population. Then an adaptive cross-variation probability is designed to make the algorithm more focused on exploring the regions that are conducive to obtaining the optimization results during the search process, so as to improve the convergence speed and stability of the algorithm. Finally, a variable neighbourhood search strategy incorporating five operations is designed according to the objectives of the model to further strengthen the local search capability of the algorithm. The experimental results show that our algorithm has better performance, and the proposed model can significantly improve the capacity utilization of the enterprise. The research content of this paper is shown in Fig. 1.

This paper is structured as follows. In Section 1, a literature review of recent scheduling studies and solution algorithms for idle capacity sharing is presented. Then, in Section 2, we propose a cooperative scheduling model for idle capacity sharing with time, cost, and quality as the model objectives. Section 3 introduces the INSGA-II algorithm, focusing on describing the improvement operations of the algorithm. Section 4 verifies the superiority of INSGA-II through comparative experiments and uses a case study to illustrate that the proposed scheduling model improves the capacity utilization of the enterprise with better performance than the traditional approach. Finally, Section 4 summarizes the full text and outlines directions for future research.

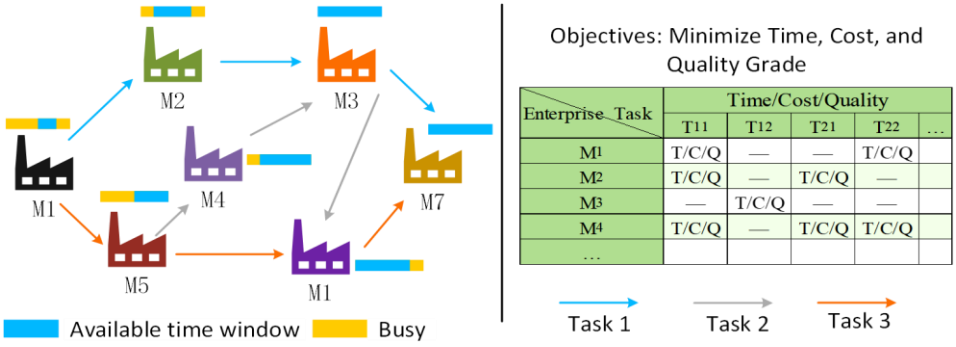


Figure 1: Production scheduling with shared idle capacity.

2. A SCHEDULING MODEL FOR MULTI-ENTERPRISE IDLE CAPACITY SHARING

2.1 Problem description

The cooperative scheduling problem for sharing idle manufacturing capacity of multiple enterprises can be described as follows: there are n tasks, and each task has a different number of subtasks, denoted as T_{ij} ($i = 1, 2, ..., n, j = 1, 2, \ldots, q_i$). We need to select one of the m enterprises (denoted as M_k $\{M_1, M_2, \ldots, M_m\}$) on the shared manufacturing platform that has the processing capability to process the subtask T_{ij} and schedule each of the remaining subtasks. Consideration also needs to be given to the fact that (1) the manufacturing capacity provided by each enterprise is within a few defined time windows; (2) for different sub-tasks, the processing time, cost, and quality of the different manufacturing enterprises are different; and (3) the transport distances and transport costs between enterprises are different.

The objective of this paper is to determine the processing enterprise for each sub-task and the processing time for different tasks on each enterprise to achieve the optimal combination of total completion time, total processing cost, and total processing quality.

2.2 Definition of symbols

The definitions of the symbols used in this paper are shown in Table I.

Table I: Definition of symbols.

Symbol	Description
k, g	Number of the enterprise, k, $g \in \{1, 2, 3, ..., m\}$
i	Number of the task, $i \in \{1, 2, 3, ..., n\}$
j, r	Number of the task, j, $r \in \{1, 2, 3, ..., q_i\}$
t	The t^{th} idle capacity time window of enterprise k, $t \in \{1, 2, 3, ..., t_k\}$
T_{ij}	Subtask j of task i
ST_{ij}	Theoretical start time for task T_{ij}
S_{ij}^k	Actual start time of task T_{ij} in enterprise k
F_{ij}	Completion time for task T_{ij}
F_{iq_i}	Completion time of the last subtask of task i
P_{ij}^k	Processing time of task T_{ij} in enterprise k
PC_{ij}^k	Processing costs of task T_{ij} on enterprise k
TT_{gk}	Transport time between enterprises g and k
TC_{gk}	Transport costs between enterprises g and k
Q_{ij}^k	Production quality of task T_{ij} on enterprise k
JS_t^k	The start of the t^{th} idle time window for enterprise k
JF_t^k	The end of the t^{th} idle time window for enterprise k
X_{ij}^k	Value 1 if task T_{ij} is processed on enterprise k, 0 otherwise
Y_{ij}^k	Value 1 if enterprise k can process task T_{ij}, 0 otherwise
δ	Value 1 if task $T_{i_2j_2}$ is processed after task $T_{i_1j_1}$, 0 otherwise
R	Value 1 if subtask j is after subtask r, 0 otherwise

2.3 Mathematical model

Based on the characteristics of the problem, we develop a mathematical model to minimize the total completion time, total cost, and total quality grade.

$$minT = \max_i F_{iq_i} \tag{1}$$

$$minC = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{m}\left(X_{ij}^k PC_{ij}^k\right) + \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{m}\sum_{g=1}^{m}\left(X_{ij}^k X_{ij+1}^k TC_{ij}^k\right) \tag{2}$$

$$minQ = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{m}\left(X_{ij}^k Q_{ij}^k\right) \tag{3}$$

$$\sum_{k=1}^{m}\left(X_{ij}^k Y_{ij}^k\right) = 1 \tag{4}$$

$$(S_{i_2j_2}^k - F_{i_1j_1}^k)\delta + (S_{i_1j_1}^k - F_{i_2j_2}^k)(1-\delta) \geq 0 \tag{5}$$

$$\left[S_{ij}^k - \left(F_{ir}^g + TT_{kg}\right)\right]R + \left[S_{ir}^g - \left(F_{ij}^k + TT_{kg}\right)\right](1-R) \geq 0 \tag{6}$$

$$S_{ij}^k X_{ij}^k = max\left\{JS_t^k, ST_{ij}, F_{i_0j_0} X_{i_0j_0}^k\right\} \tag{7}$$

$$\left(P_{ij}^k + S_{ij}^k\right)X_{ij}^k \leq \min_t JF_t^k \tag{8}$$

$$JF_{t_k}^k \geq \max_i \max_j \left(S_{ij}^k + P_{ij}^k\right)X_{ij}^k \tag{9}$$

$$F_{ij}^{k} = S_{ij}^{k} + P_{ij}^{k} \tag{10}$$

$$ST_{ij} = \begin{cases} \max\left(F_{ir}^{g} + TT_{kg}\right) X_{ij}^{k}, r \in pre\ j \\ 0, \quad j = 1 \end{cases} \tag{11}$$

Eq. (1) is to minimize the maximum completion time. Eq. (2) is to minimize the total cost, where the total cost is the sum of the total manufacturing cost and the total transport cost. Eq. (3) is to minimize the total quality grade. Where a lower quality grade represents a higher product pass rate and the production pass rate of all processable enterprises meets the minimum requirements of the task. Eq. (4) indicates that each subtask can be processed by only one enterprise that can process it. Eq. (5) indicates that an enterprise cannot process more than one task at the same time. Eq. (6) is a priority constraint between different subtasks, where a later subtask can only start when the previous subtask is completed and transported to the next enterprise. Eq. (7) is the actual start time constraint of the task, and in enterprise *k*, T_{i0j0} denotes the immediately preceding task of T_{ij}. Eq. (8) indicates that each task is to be completed within the most forward available time window in the enterprise. Eq. (9) indicates that the completion time of all tasks on enterprise *k* is less than the end time of its last time window. Eq. (10) is the completion time of the task. Eq. (11) is the theoretical start time constraint for the task, which is moment 0 when $j = 1$.

3. IMPROVED OPERATION OF THE INSGA-II ALGORITHM

The INSGA-II algorithm in this paper mainly improves the initialization, adaptive operation, and variable neighbourhood search strategy and designs a reasonable decoding method. The elite selection strategy uses the non-dominated sorting method in NSGA-II to determine the fitness value of an individual based on the objective function to evaluate the individual's merit. The pseudocode of the algorithm is shown in Fig. 2.

Improved Non-dominated Sorting Genetic Algorithm (INSGA-II)

Input: Original dataset, population size N, number of iterations Gen, crossover probability Pc=(minPc, maxPc), mutation probability Pm=(minPm, maxPm)
Output: Pareto Front PF
1: Three initialization rules were used to generate the initial population
2: Decode and remove invalid individuals, then place the non-dominated solution set in an external population (EP)
3: **For** t=1 to Gen
4: Set the adaptive Pc and Pm according to section 3.6
5: **For** n=1 to N
6: **If** rand<Pc
7: According to the crossover strategy in Section 3.4, the IPOX approach is used for task codes and the MPX approach is used for enterprise codes
8: **End**
9: **If** rand<Pm
10: According to the mutation strategy in section 3.5, task codes use the insertion mutation and enterprise codes use the hybrid mutation
11: **End**
12: Local search using the variable neighborhood search strategy of Section 3.7
13: **End**
14: Decode and remove invalid individuals, then use the elite strategy to select the next generation of the population
15: Update EP
16: **End**

Figure 2: Flowchart of INSGA-II algorithm.

3.1 Coding rules

In this paper, two layers of chromosome coding are used, the first layer is task code, in which each subtask is represented by a corresponding task serial number, and the k^{th} occurrence of the task serial number from left to right indicates the k^{th} subtask of the task; the second layer is enterprise code, which is used to determine the manufacturing enterprise corresponding to each subtask. As shown in Fig. 3, the order of task processing and the corresponding manufacturing companies are as follows: (T_{21}, M_3), (T_{11}, M_1), (T_{22}, M_4), (T_{12}, M_3), (T_{31}, M_4), (T_{13}, M_2), (T_{23}, M_5), and (T_{32}, M_2).

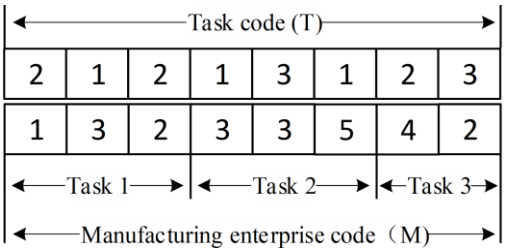


Figure 3: Two-layer chromosome coding.

3.2 Decoding rules

Since this paper is insertion scheduling at idle capacity, the chromosomes generated by the algorithm may be invalid scheduling schemes, which requires decoding the newly generated population and removing the invalid individuals in the following steps:

(1) Determine the available time window. First, we need to determine the manufacturing period available for processing in each enterprise.

(2) Sequential decoding. Determining the processing order of each task based on the task code.

(3) Time scheduling. Determine the processing enterprise for each subtask based on the enterprise code, find its processing time and transport time to the next enterprise, insert the task forward to the corresponding enterprise's available period for processing, and ensure that it will not conflict with other processing tasks. By repeating the above operation until all sub-tasks have corresponding enterprises completed, the actual scheduling plan can be obtained.

3.3 Initialisation

To improve the quality and diversity of the initial population, we designed an initialization strategy that incorporates three rules.

Random initialization rules. The task codes are randomly sorted, and then the enterprise codes for the corresponding tasks are randomly selected.

Hybrid initialization rules. Task codes are randomly generated, and then the enterprise with the smallest processing time, the smallest processing quality level, or the smallest processing cost is selected for each subtask. The ratio of using each of the above three methods is set to 1/3.

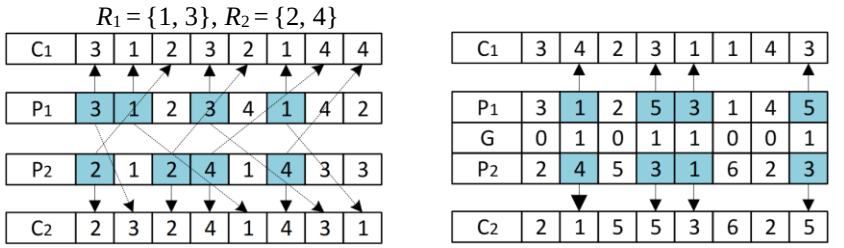
Non-dominated solution priority rule. Execute the above two rules separately to produce two initial populations, both of size N. Then use non-dominated sorting to select the top N individuals as the initial population.

3.4 Crossover strategy

The task code section uses improved precedence operation crossover (IPOX) as shown in Fig. 4 a. Firstly, assign all tasks randomly into two datasets R_1 and R_2 such that $R_1 \cup R_2 = R$,

$R_1 \cap R_2 = \emptyset$. Next, copy the task gene of R_1 in parent P_1 into child C_1, copy the task gene of R_2 in parent P_2 into child C_2, and keep the position of the task gene unchanged before and after copying. Finally, empty gene positions in child C_1 are filled in sequentially with the task genes of R_2 in parent P_2, and empty gene positions in child C_2 are filled in sequentially with the task genes of R_1 in parent P_1.

The enterprise code section uses multipoint preservative crossover (MPX) operation, as shown in Fig. 4 b. Firstly, randomly generate an array G of 0 and 1 with the length of the total number of tasks. Then randomly select two parent chromosomes P_1 and P_2 based on the manufacturing enterprises' codes and exchange the codes of P_1 and P_2 at the position where the number 1 appears in the array G to generate the manufacturing enterprises' codes of the child.



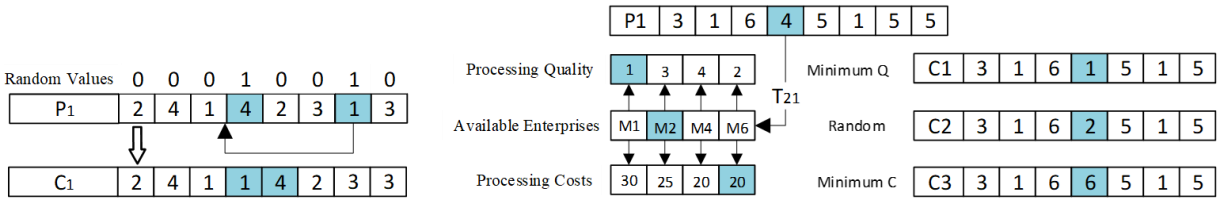
a) IPOX operations for task code b)MPX operations for enterprise code

Figure 4: Crossover operation.

3.5 Mutation strategy

Task code uses an insertion mutation operation, as shown in Fig. 5 a. The operation is as follows, two position points, S_1 and S_2 are randomly selected, and if $S_1 < S_2$, the task gene located in S_2 is inserted before S_1.

The enterprise code uses a combined mutation operation, as shown in Fig. 5 b. A gene location is randomly found in the enterprise code, and in the set of optional enterprises at that location, (1) its gene is randomly replaced with another enterprise, (2) its gene is replaced with the enterprise with the smallest processing quality grade, and (3) its gene is replaced with the enterprise with the smallest processing cost. The ratio used in all three of these ways is set to 1/3.



a) Task code mutation operations b)Enterprise code mutation operations

Figure 5: Mutation operation.

3.6 Adaptive cross-mutation strategy

The crossover and mutation operations and their probabilities affect the search performance of the algorithm. In the pre-evolutionary stage of the population, the excellent solutions are far away from the Pareto front, when a larger crossover probability (P_c) can be used to improve the local search ability; in the late stage of the population evolution, the number of excellent solutions is larger, when a larger mutation probability (P_m) can be used to improve the global search ability. These adaptive crossover and mutation probabilities follow Eqs. (12) and (13).

$$P_c(i) = minP_c + \frac{1}{2}(maxP_c - minP_c)\left(1 + cos(\pi \frac{i}{Gen})\right) \quad (12)$$

$$P_m(i) = minP_m + \frac{1}{2}(maxP_m - minP_m)\left(1 + sin(\pi\frac{i}{Gen} - \frac{\pi}{2})\right) \quad (13)$$

Where: $P_c(i)$ is the crossover probability of generation i; $P_m(i)$ is the variation probability of generation i; Gen is the total number of iterations; $maxP_c$ is the maximum crossover probability; $minP_c$ is the minimum crossover probability; $maxP_m$ is the maximum variation probability; $minP_m$ is the minimum variation probability.

3.7 Variable Neighbourhood Search Strategy

To increase the convergence ability of the algorithm, we designed five variable neighbourhood search (VNS) strategies. Five neighbourhood actions NS_i $\{i \in (1, 2, 3, 4, 5)\}$ are used separately for all individuals X in the population to search in the neighbourhood of solution X to obtain five new neighbourhood solutions. Decode the new solution and retain the non-inferior solutions among the old and new solutions using non-dominated ordering. The five neighbourhood actions are as follows:

(1) NS_1: Find the enterprise code of the last completed subtask and replace it with the one with the smallest processing time among the available enterprises.

(2) NS_2: Find the enterprise code with the largest processing cost for the corresponding subtask and replace it with the one with the smallest processing cost among the available enterprises.

(3) NS_3: Find the enterprise code with the largest processing quality grade for the corresponding subtask and replace it with the one with the smallest processing quality grade among the available enterprises.

(4) NS_4: Reverse Sequencing, in which two gene positions are randomly generated and the gene segments between the two points are reversed on a task code gene string.

(5) NS_5: Disrupting the order, in which some gene locations are randomly generated, disrupts the order of the genes on these task codes.

4. NUMERICAL EXPERIMENT

The INSGA-II algorithm in this paper is programmed using MATLAB 2020, running on CPU Intel Core i5-8th Gen and Windows 10 operating system. The population size is $N = 300$, the maximum number of iterations is $Gen = 100$, the crossover probability range is set to $(0.4, 0.8)$, and the mutation probability range is set to $(0.01, 0.2)$.

4.1 Data generation

We modified the actual data from a manufacturing platform to generate experimental data that meet the problems of this paper (*https://pan.baidu.com/s/1099lKY9zxJ7ixxER4UhIGw?pwd =n3xu*). Where the processing time is simplified to an integer within $(2, 8)$ and the processing cost is randomly generated within $(10, 30)$ based on the size of the processing time. The quality level is randomly generated in the range $(1, 5)$, and the lower the level, the better the quality. The transport time is randomly generated in the range $(1, 3)$ and tried not to be greater than the processing time of the task, and the unit transport cost is set to 3. For example, Table II is a small set of case data that will be used in the example simulation section. The time unit is an hour and the cost unit is yuan.

Table II: Case data.

Processing Time/Processing Cost/Quality Grade								Transportation Time/Costs						
Task	Subtask	M1	M2	M3	M4	M5	M6		M1	M2	M3	M4	M5	M6
T1	1	-	-	4/11/3	7/28/3	-	-	M1	0/0	2/6	1/3	3/9	2/6	3/9
	2	6/16/1	-	6/24/5	-	6/24/5	5/20/2	M2		0/0	2/6	1/3	1/3	3/9
	3	-	-	-	-	-	8/29/5	M3			0/0	3/9	1/3	1/3
	4	-	-	6/16/5	0/0/0	7/14/3	6/19/2	M4				0/0	3/9	3/9
T2	1	-	-	-	8/28/1	-	4/18/1	M5					0/0	2/6
	2	-	-	7/22/4	7/20/4	-	-	M6						0/0
	3	8/17/4	-	6/18/3	6/20/4	-	6/10/4	-	-	-	-	-	-	-
	4	-	7/15/2	8/23/3	7/12/2	-	-	-	-	-	-	-	-	-
T3	1	-	-	6/12/1	5/24/5	-	-	-	-	-	-	-	-	-
	2	-	-	-	-	-	4/19/3	-	-	-	-	-	-	-
	3	-	-	7/16/4	8/17/4	6/15/4	5/27/2	-	-	-	-	-	-	-
T4	1	-	6/24/1	6/23/1	7/26/5	-	-	-	-	-	-	-	-	-
	2	4/27/1	7/26/1	6/20/1	4/17/4	-	-	-	-	-	-	-	-	-
	3	-	6/19/4	-	4/14/5	4/28/4	-	-	-	-	-	-	-	-
	4	-	-	-	6/14/1	4/22/4	6/19/1	-	-	-	-	-	-	-
T5	1	7/20/1	-	4/23/4	0/0/0	4/27/2	7/24/3	-	-	-	-	-	-	-
	2	-	-	-	4/13/2	-	-	-	-	-	-	-	-	-

4.2 Algorithmic test

In order to assess the correctness of the model and the superiority of the algorithms, we tested five sets of data of different sizes using the two algorithms NSGA-II and INSGA-II, each of which was run independently 10 times, and took all the optimal nondominated solutions of the 10 experiments as the results. Where *n* is the number of tasks and *m* is the number of enterprises. In the comparative analysis, we list the minimum values of the three objectives obtained by the different algorithms for each set of data as well as the Inverse Generation Distance (*IGD*) metric, as shown in Table III. Where *IGD* is a comprehensive evaluation metric commonly used in algorithm performance evaluation [18]. Finally, the Pareto comparison plots of three of the algorithms are listed, as shown in Fig. 6.

Table III: Test results for 5 sets of data.

Example	Size ($m\times n$)	NSGA-II				INSGA-II			
		T	*C*	*Q*	*IGD*	*T*	*C*	*Q*	*IGD*
SJ01	10×6	22	598	33	0.0747	**21**	**586**	**33**	0.0106
SJ02	10×8	**28**	**784**	**66**	0.2416	**28**	**784**	**66**	0.0021
SJ03	15×8	27	944	71	0.1585	**25**	**880**	**59**	0.0137
SJ04	15×10	31	1473	127	0.1834	**33**	**1359**	**99**	0.0260
SJ05	15×15	**42**	2259	219	0.1779	43	**2112**	**169**	0.0029

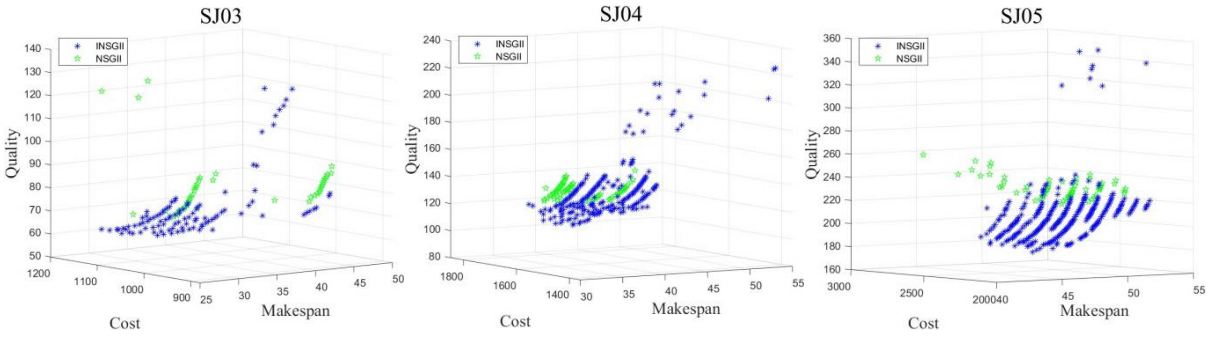


Figure 6: Pareto chart for 3 sets of data.

In Table III, the bolded indicator values denote the values that are optimal for each set of data. It can be seen that INSGA-II takes the optimal value under all five sets of test data except for the completion time of SJO5, which indicates that the algorithm not only takes a better solution in terms of the objective value but also has better diversity and stronger convergence in terms of IGD performance than NSGA-II. Further observation of Fig. 6 clearly shows that the solutions of the INSGA-II algorithm basically dominate the solutions of the NSGA-II algorithm under the three sets of test data, and the solutions of INSGA-II are more numerous and more evenly distributed. This indicates that our algorithm can fully explore the solution space and find more and better optimal non-dominated solutions. The above test results prove the validity of the model and the superiority of the algorithm in this paper, and this effect is achieved because the initialization strategy proposed in this paper accelerates the convergence speed of the algorithm, the adaptive strategy enables the algorithm to search the solution space more extensively, and the variable neighbourhood search strategy strengthens the algorithm's ability to search locally.

4.3 Example simulation

We used the small case data in Table II for simulation and obtained the scheduling scheme with the minimum completion time under both algorithms, as shown in Fig. 7. In the Gantt chart, Tasks 1–10 represent the original production plan, Tasks 11–15 represent the scheduling plan after inserting the new tasks into the free capacity, and the blue progress bar represents the transport time for each task to be transferred to the next enterprise for production. Where the total completion time of the original production plan was 71 hours. As can be seen in the figure, in INSGA-II's scheduling scheme, all new tasks are inserted and completed within the cycle of the original production plan without affecting the production of the original order or taking up capacity in the next cycle. This fully utilizes the free capacity of the enterprise and improves its revenue of the enterprise. However, in the NSGA-II scheduling program, many of the new tasks were completed beyond the cycle time of the original production plan and took up capacity in the next production cycle. This may affect the production schedule of the next cycle and lead to the failure of this mission.

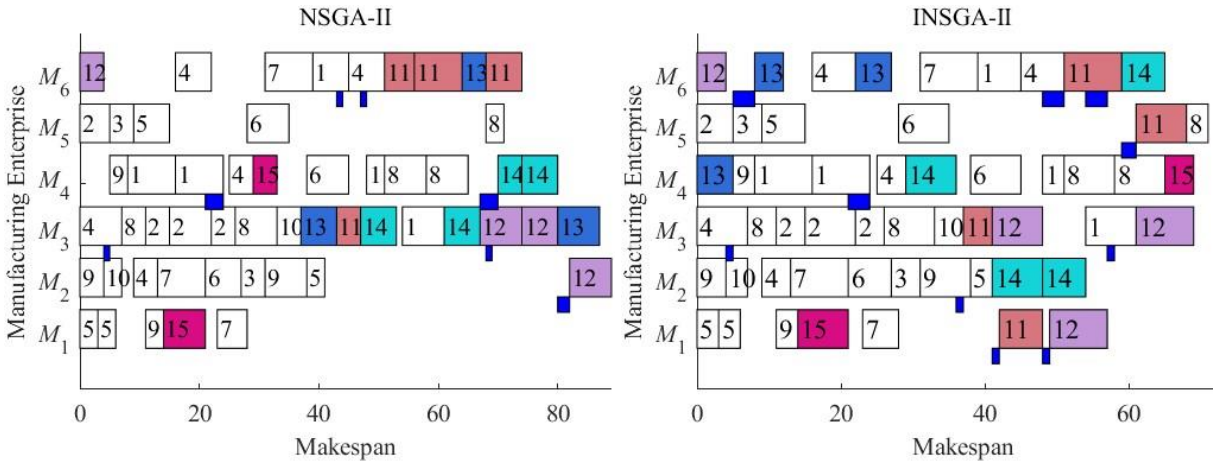


Figure 7: Scheduling schemes for NSGA-II and INSGA-II.

Further analysis of the capacity utilization under the above two scheduling options is shown in Table IV. We take the total completion time of the original production plan as a manufacturing cycle. The table analyses the capacity utilization of each enterprise under the two algorithms in a cycle when producing according to the original tasks and after adding new tasks. Bolded data indicates better values. It can be seen that the use of spare capacity for the production of new tasks greatly increases the capacity utilization of each enterprise, which fully

reflects the value of capacity sharing. Under the scheduling scheme of INSGA-II, all the enterprises except M3 have higher utilization increments than NSGA-II, which indicates that our scheduling method can better improve productivity and resource utilization.

Table IV: Analysis of capacity utilization.

		M1	M2	M3	M4	M5	M6
Original tasks	Utilization time (h)	14	39	44	47	25	26
	Utilization rate (%)	20 %	55 %	62 %	66 %	35 %	37 %
Addition of new tasks (NSGA-II)	Utilization time (h)	21	39	66	51	25	46
	Utilization rate (%)	30 %	55 %	93 %	72 %	35 %	65 %
	Increase (%)	10 %	0 %	**31 %**	6 %	0 %	28 %
Addition of new tasks (INSGA-II)	Utilization time (h)	35	52	63	63	33	53
	Utilization rate (%)	49 %	73 %	89 %	89 %	46 %	75 %
	Increase (%)	**30 %**	**18 %**	27 %	**23 %**	**11 %**	**38 %**

5. CONCLUSIONS

This paper presents an innovative digital twin model to optimize distributed scheduling for multi-enterprise manufacturing systems. The proposed techniques integrate sensor data with multi-objective decision making to enhance collaborative scheduling. Customized INSGA-II operators and variable neighbourhood search heuristics help discover high-quality solutions. Extensive experiments demonstrate the MDIS digital twin model's capabilities in reducing lead times, costs and improving asset utilization over conventional approaches.

While promising, certain limitations exist. Additional real-world manufacturing data would further validate the scalability and test the robustness against uncertainties. Exploring ensemble optimization methods with MDIS is another worthwhile direction. This research contributes both novel modelling and algorithmic insights for next-generation smart manufacturing. The digital twin paradigm provides a powerful and interpretable platform to aid multi-party decision making. As global manufacturing ecosystems become increasingly interconnected, such optimization capabilities will only grow in importance.

REFERENCES

[1] Chiappa, S.; Videla, E.; Viana-Céspedes, V.; Piñeyro, P.; Rossit, D. A. (2023). Cloud manufacturing architectures: state-of-art, research challenges and platforms description, *Journal of Industrial Information Integration*, Vol. 32, Paper 100472, 16 pages, doi:10.1016/j.jii.2023.100472

[2] Priego, R.; Iriondo, N.; Gangoiti, U.; Marcos, M. (2017). Agent-based middleware architecture for reconfigurable manufacturing systems, *The International Journal of Advanced Manufacturing Technology*, Vol. 92, No. 5-8, 1579-1590, doi:10.1007/s00170-017-0154-z

[3] Xu, Y.; Zhi, R.; Zheng, F.; Liu, M. (2022). Parallel machine scheduling with due date-to-deadline window, order sharing and time value of money, *Asia-Pacific Journal of Operational Research*, Vol. 39, No. 2, Paper 2150024, 15 pages, doi:10.1142/S021759592150024X

[4] Koulouris, A.; Georgiadis, G. P. (2023). An exact algorithm for calculating the minimum and feasible ranges of cycle time in periodic scheduling with shared resources, *Computers & Chemical Engineering*, Vol. 175, Paper 108286, 12 pages, doi:10.1016/j.compchemeng.2023.108286

[5] Chen, F.; Dou, R.; Li, M.; Wu, H. (2016). A flexible QoS-aware web service composition method by multi-objective optimization in cloud manufacturing, *Computers & Industrial Engineering*, Vol. 99, 423-431, doi:10.1016/j.cie.2015.12.018

[6] Aghamohammadzadeh, E.; Malek, M.; Valilai, O. F. (2020). A novel model for optimisation of logistics and manufacturing operation service composition in Cloud manufacturing system focusing on cloud-entropy, *International Journal of Production Research*, Vol. 58, No. 7, 1987-2015, doi:10.1080/00207543.2019.1640406

[7] Tong, H.; Zhu, J. (2022). A novel method for customer-oriented scheduling with available manufacturing time windows in cloud manufacturing, *Robotics and Computer-Integrated Manufacturing*, Vol. 75, Paper 102303, 19 pages, doi:10.1016/j.rcim.2021.102303
[8] Wang, Z. J.; Suo, J. (2022). Optimization of flexible production logistics under low carbon constraint, *International Journal of Simulation Modelling*, Vol. 21, No. 1, 184-195, doi:10.2507/IJSIMM21-1-CO5
[9] Chen, D.; Zhao, X. R. (2021). Production management of hybrid flow shop based on genetic algorithm, *International Journal of Simulation Modelling*, Vol. 20, No. 3, 571-582, doi:10.2507/IJSIMM20-3-CO12
[10] Huo, L.; Wang, J. Y. (2022). Flexible job shop scheduling based on digital twin and improved bacterial foraging, *International Journal of Simulation Modelling*, Vol. 21, No. 3, 525-536, doi:10.2507/IJSIMM21-3-CO14
[11] Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, 182-197, doi:10.1109/4235.996017
[12] Han, Y.; Chen, X.; Xu, M.; Gu, F. (2020). A study on multi-objective flexible job shop scheduling problem using a non-dominated sorting genetic algorithm, *Proceedings of IncoME-V & CEPE Net-2020*, 745-755, doi:10.1007/978-3-030-75793-9_72
[13] Han, Y.; Chen, X.; Xu, M.; An, Y.; Gu, F.; Ball, A. D. (2022). A multi-objective flexible job-shop cell scheduling problem with sequence-dependent family setup times and intercellular transportation by improved NSGA-II, *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, Vol. 236, No. 5, 540-556, doi:10.1177/09544054211044660
[14] Zhou, Z.; Xu, L.; Ling, X.; Zhang, B. (2024). Digital-twin-based job shop multi-objective scheduling model and strategy, *International Journal of Computer Integrated Manufacturing*, Vol. 37, No. 1-2, 87-107, doi:10.1080/0951192X.2023.2204475
[15] Liang, X.; Liu, Y.; Huang, M. (2020). Improved NSGA2 algorithm to solve multi-objective flexible job shop scheduling problem, *2020 IEEE 8th International Conference on Computer Science and Network Technology (ICCSNT)*, 22-25, doi:10.1109/ICCSNT50940.2020.9304984
[16] Li, H.; Duan, J.; Zhang, Q. (2021). Multi-objective integrated scheduling optimization of semi-combined marine crankshaft structure production workshop for green manufacturing, *Transactions of the Institute of Measurement and Control*, Vol. 43, No. 3, 579-596, doi:10.1177/0142331220945917
[17] Liu, Y.; Wang, X.; Zhang, Y.; Liu, L. (2023). An integrated flow shop scheduling problem of preventive maintenance and degradation with an improved NSGA-II algorithm, *IEEE Access*, Vol. 11, 3525-3544, doi:10.1109/ACCESS.2023.3234428
[18] Caldeira, R. H.; Gnanavelbabu, A. (2021). A Pareto based discrete Jaya algorithm for multi-objective flexible job shop scheduling problem, *Expert Systems with Applications*, Vol. 170, Paper 114567, 20 pages, doi:10.1016/j.eswa.2021.114567