

Article

Evolutionary Algorithm for Aerospace Shell Product Digital Production Line Scheduling Problem

Qing Wang ¹, Haiwei Luo ², Jian Xiong ³, Yanjie Song ^{1,*}  and Zhongshan Zhang ¹

¹ College of Systems Engineering, National University of Defense Technology, Changsha 410073, China

² Capital Aerospace Machinery Corporation Limited, Beijing 100044, China

³ School of Business Administration, Southwestern University of Finance and Economics, Chengdu 610074, China

* Correspondence: songyj_2017@163.com; Tel.: +86-13548620609

Received: 4 June 2019; Accepted: 25 June 2019; Published: 1 July 2019



Featured Application: Aerospace Shell Product Digital Production.

Abstract: The production of aerospace shell products is directly related to the safety and reliability of aerospace products. In this work, the aerospace shell product digital production line scheduling problem (ASPDPLSP) was studied, and a solution was developed. In the production process, it is necessary to decide the processing machine and time of each operation. In order to create a scientific shell product production plan, we propose an operation scheduling algorithm (OSA). Based on the constraints of the inspection process, the OSA has two heuristic task scheduling rules. Then, in order to further optimize the product production plan, an improved genetic algorithm (IGA) is proposed. Considering the repeatability caused by random search, a method for the initial population generation with similar and diverse characteristics is proposed. Two of these generation rules retain symmetry and randomness. IGA was used to optimize the order in which the products were processed, resulting in lower costs. Simulation experiments showed that the proposed algorithm solved ASPDPLSP well and provided suggestions to produce aerospace shell products.

Keywords: aerospace product; digital production line; evolutionary algorithm; genetic; scheduling

1. Introduction

With the increasing production pressure of aerospace products, the production line is also moving in the direction of digitalization and intelligence. The aerospace shell section is the core component of various types of spacecraft and is used to protect internal parts and reduce air resistance in order to increase flight speed. As a typical aerospace product, similar to other aerospace products, the aerospace shell also involves a manufacturing process of significant complexity and has high reliability requirements. Every aerospace shell product needs to undergo repeated processing and inspection before it can be used as a finished product on a spacecraft. The processing of the shell products is completed on a digital production line. The production process of each shell product involves several specialized production lines, and one shell production line is responsible for the processing of various types of shell products. Arranging a reasonable production plan for the shell products, and formulating the processing sequence and processing time for each product and process is the key in the digital production of aerospace shell products and in providing safe and reliable products for the spacecraft.

The aerospace shell product line scheduling problem is similar to the flexible job-shop scheduling problem (FJSP). Each type of housing product has multiple operations that can be performed on any of the available production lines. Each type of production line can only perform one type of

operation to mass produce and process products. Because of the extremely high safety and reliability requirements of aerospace products, professional staff are required to carry out various types of specialized processing and inspection to ensure the product's pass rate. Each type of staff does not limit the type of product, and any product that requires inspection and manual processing can be handled by the staff. ASPDPLSP involves selecting multiple products in the planning space to select the appropriate processing time and striving to make the product processing plan as reasonable as possible.

Thus far, relatively few studies on aerospace product production lines at home and abroad have been reported. Most of this research has focused on classic FJSP research, which has been very helpful for the shell production line scheduling problem. In [1], researchers proposed a multi-stage graph-based heuristic method to find the optimal solution by finding the shortest path. An improved implementation of a multi-factory production process was presented using an improved differential evolution simulated annealing algorithm (IDESAA) in [2]. In [3], researchers considered the effect of energy consumption on FJSP and proposed a modified shuffled frog-leaping algorithm (SFLA) to solve this problem. In [4], researchers used social spider optimization (SSO) to solve FJSP and focused on the distinction between the two diverse search specialists (insects): males and females. In [5], researchers analyzed the effect of the task processing time uncertainty on the problem. A problem-solving method with a high efficiency and good robustness was designed. A linear job planning method and a genetic algorithm were well combined to solve a flexible job-shop lot streaming problem in [6]. The FJSP of the fuzzy processing time was taken as the research object, and an effective teaching-learning-based optimization algorithm was designed in [7]. For the scheduling problem with the fuzzy processing time, a bionic artificial bee colony algorithm was designed in [8]. It provided better optimization results. In [9], researchers proposed a genetic algorithm in which an operation selection strategy is used to balance the load between machines. In [10], researchers studied the problem of the fuzzy processing time in the remanufacturing process and designed a discrete search method. In [11], researchers analyzed an extended FJSP that allowed precedence between the operations to be given by an arbitrary directed acyclic graph, proposed a list-based search algorithm, and extended this method to a beam search method. In [12], researchers solved a dynamic FJSP by simulation.

In the existing FJSP study, the production process of the product can be basically completed using the production line, and there are few processes in which people participate. The production process of aerospace shell products is midway between being automatic and fully manual, with automated production processes involving professional staff. Reasonable staffing of the processing and inspection work will have a significant effect on the production plan. At the same time, there is a waiting time for the product on the production line of the casing product; that is, the processing process needs to wait for a certain period of time before it can be carried out. In the present research, we considered the effect of the people involved in the automated production process and the waiting process in the product processing process, developed efficient task planning algorithms and improved genetic algorithms, obtained reasonable production plans, and optimized these production plans to reduce the production costs.

The innovation of this study was to design a reasonable scheduling algorithm for the aerospace shell production line and reduce the production cost while realizing a digital production. A method that takes both symmetry and randomness into consideration is proposed for the initial population generation. The influence of the machine setting and the professional staff on the optimization target was also analyzed to find a reasonable number of machines and staff to achieve reasonable management results.

The rest of this paper is organized as follows. In Section 2, we introduce the mathematical model and constraints of the aerospace shell production line scheduling problem. In Section 3, we present the proposed algorithm, including the overall flow of the algorithm and the specific implementation of the problem in the algorithm. In Section 4, we verify the proposed algorithm through experiments and provide reasonable suggestions to produce aerospace shell products. In Section 5, we summarize the conclusions of this study and provide a future research direction.

2. Digital Production Line Scheduling Problem

2.1. Problem Description

In the production workshop of aerospace shell products, there are several automated production lines. Each of these lines can be processed for an operation of the products. Each shell product requires multiple processes to become the final product from the raw material; that is, a product is obtained after the raw and intermediate materials pass through multiple production lines from processing to completion. During the processing, not all processes can be completed by the production line, and professional staff are required to carry out a series of tasks such as painting and inspection. Thus, ASPDPLSP can be described as follows:

In the aerospace shell product production lines, there are n independent jobs. These jobs constitute a task set $J = \{J_1, J_2, \dots, J_n\}$. In each job J_i , represented by a sequence of operations O_{ij} , job i has a total of n_i procedures. Each job has a release time r_j and due time d_j . All the operations in J_i need to be completed within the scope of the release time and the due time. The order of these operations is from the first to the last. Before the current process is completed, the next process cannot be started. For some of these processes, it is necessary to wait for a certain waiting time $w_{i,j,k}$ after the completion of the previous process. Each process must be done on a specific type of machine. Professional staff are also involved in the production and processing of the product, completing one or more processes of a job. The goal of scheduling is to choose a satisfactory production plan for these jobs.

2.2. Mathematical Model

In this section, a mathematical model of ASPDPLSP will be presented. Before building the scheduling model, we transformed a complex practical engineering problem into a scientific problem by making several assumptions; these assumptions were as follows:

1. The entire process of each job is known and determined.
2. Different jobs are independent of one another.
3. After a process starts, it cannot be interrupted.
4. The situation of a machine failure is not considered.
5. There is no difference between machines that can perform the same process.
6. A machine can start the production of another process immediately after completing one process.
7. A professional staff member can be considered to be a machine that processes the production process.

Therefore, all the variables involved in the scheduling model are given and can be expressed as follows:

$J = \{J_1, J_2, \dots, J_n\}$: Set of jobs

O_{ij} : Operation j of job i

$M = \{M_1, M_2, \dots, M_m\}$: Set of machines

r_i : Release time of job i

d_i : Due time of job i

$w_{i,j,k}$: Waiting time on machine k for operation j of job i

$p_{i,j,k}$: Processing time on machine k for operation j of job i

Decision variables:

$x_{i,j,k}$: If machine k is selected to process job i of operation j , $x_{i,j,k}$ is 1; otherwise, it is 0.

$s_{i,j,k}$: Start time of job i operation j on machine k

$c_{i,j,k}$: Completion time of job i operation j on machine k

In ASPDPLSP, the goal of optimization is to minimize the makespan and the sum of the waiting times between operations. The consideration of the waiting time in the objective function is to shorten the wait of different operations and improve the productivity of the production lines.

Objective function:

$$\min \left\{ \max(c_i) + \sum_{j=1}^{n_i-1} \sum_{k=1}^m \sum_{i=1}^n [(s_{i,j+1,k} - c_{i,j,k}) \cdot x_{i,j,k}] \right\} \quad (1)$$

Constraints:

- Operation j must be executed and completed within the allowable time range of job i :

$$r_i \leq O_{i,j} \leq d_i \quad (2)$$

- If machine k is unable to process operation j , the waiting time and the machining time will be positive infinity:

$$w_{i,j,k} = +\infty, \text{ if machine } k \text{ cannot do operation } j \quad (3)$$

$$p_{i,j,k} = +\infty, \text{ if machine } k \text{ cannot do operation } j \quad (4)$$

- A process needs to be started only after the previous process is completed:

$$s_{i,j} + p_{i,j} + w_{i,j} \leq s_{i,j+1} \quad (5)$$

ASPDPLSP is a complex scheduling problem. The main difficulty of the problem lies in the need to simultaneously determine the processing time of the processing machines and processes in each process. In addition, the existence of the waiting time increases the complexity of the problem. Professional staff needs to participate in the production and the processing of most of the jobs, which implies that the staff will limit the efficiency of the system, and the coordination of the task sequence of the staff will considerably affect the final scheduling results.

3. Methods

As a complex scheduling problem, ASPDPLSP needs to determine the processing tasks on each production line and the start time of these tasks. To achieve an optimal solution to this problem, we need to design an efficient solution algorithm. Therefore, we propose a task planning algorithm and an improved genetic algorithm. The mission planning algorithm is used to select the appropriate machine and the appropriate production processing time for each of the individual tasks. The improved genetic algorithm improves the evolution of the biological gene and the ASPDPLSP based on the simple genetic algorithm, in order to improve the speed while ensuring the optimization effect. The task planning algorithm is nested in the IGA and is used to calculate the fitness of the individuals in the population. IGA performs population improvement through genetic manipulation according to the fitness function to achieve optimal results.

3.1. Operation Scheduling Algorithm

In ASPDPLSP, it is necessary to select the appropriate machine and processing time for each operation to reduce the value of the objective function proposed in the previous section. To this end, we propose a task planning algorithm. The task planning algorithm uses a single operation arrangement; that is, the next operation in the task sequence is arranged after the previous one is scheduled with an appropriate execution time. In the process of arranging an operation, all the constraints need to be checked. When the operation can satisfy the constraints in the previous part, the appropriate execution time is determined for the operation.

The pseudo code table of the task planning algorithm is shown in the Algorithm 1.

Algorithm 1: Operation Scheduling Algorithm (OSA)

Input: set of jobs J , set of machines M , release time of job i r_j , due time of job j d_j , $w_{i,j,k}$ waiting time on machine k for operation j of job i , $p_{i,j,k}$ process time on machine k for operation j of job i

Output: product processing plan $Plan$

Get scheduling order from IGA

For each operation in each job by scheduling order

 Check operation $O_{i,j}$ in scheduling horizon (r_j, d_j)

If satisfied constraint check available machines

$AM \leftarrow$ available machines // AM is a set of available machines

Else

 Turn to the next operation

End If

$Machine\ No. \leftarrow$ Select suitable machine use AssignmentRule1($O_{i,j}$, AM)

$Start\ time \leftarrow$ Use AssignmentRule2($O_{i,j}$, $Machine\ No.$)

 Add to processing plan ($Machine\ No.$, $Start\ time$)

End For

Return $Plan$

The operation scheduling algorithm first checks one by one according to the constraints involved in the ASPDPLSP, and finds all the machines and times that can perform each of the jobs. Then, we used the two heuristic rules that we proposed to determine the time of the machine and the production process for the production task for each process.

The constraint judgment process of the task planning algorithm is as follows: First, it is judged whether the process is within the planned space, and if it is within the planned space, the next constraint judgment is performed. Thereafter, it is judged whether the available machine can complete the process, as the task can be performed only on the machine that can complete the process. Finally, the task is arranged in a suitable position after the completion of the previous process, that is, when the scheduled task execution position has completed the previous product processing work.

- AssignmentRule1: The machine with the highest idle time is preferred.
- AssignmentRule2: The operation is scheduled to be executed at the previous time position.

In order to express AssignmentRule2 more intuitively, we will discuss the task scheduling of two jobs on two machines as an example. Assume that each task has two operations, and the scheduling process is as shown in Figure 1. After AssignmentRule1 sorted the machines, M_1 and M_2 were selected as the two machines that completed J_2 . Figure 1a shows a position where the job can be executed before J_2 has not used the scheduling rule. Figure 1b shows the execution position of J_2 after using AssignmentRule2. After the assigning, the two jobs become compact in the task execution plan. Thus, the completion time will be shortened.

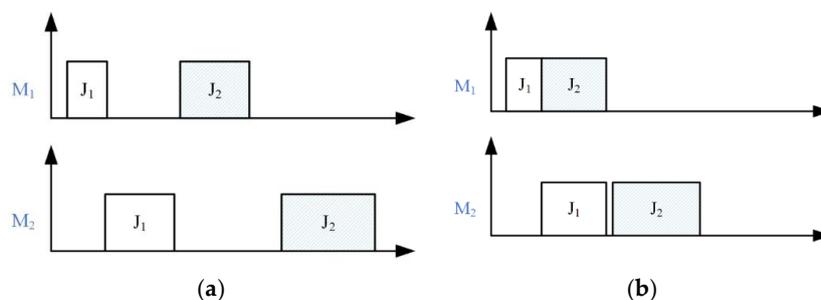


Figure 1. Scheduling result after using AssignmentRule2. (a) shows two jobs which were not assigned by AssignmentRule2. (b) shows two jobs were assigned by AssignmentRule2.

The two heuristic algorithms in the mission planning algorithm are very helpful in selecting the appropriate production and processing machines and production processing time for the task. Rule 1 can take advantage of earlier idle machines. Rule 2 is a good way to make the task more compact and effectively reduce the waiting time between two processes.

3.2. Improved Genetic Algorithm

The genetic algorithm is a meta-heuristic algorithm that mimics the evolution of biological genes. In ASPDPLSP, because of the large number of jobs and the number of machines, it is difficult to obtain good optimization results with simple genetic algorithms. To this end, we propose an improved genetic algorithm. This improved algorithm is used to adjust the order of execution of the task sequences and to find better solutions through population improvement. The pseudo code for improving the genetic algorithm is shown in Algorithm 2 of the code. Next, we will introduce the termination conditions of the representation in the genetic algorithm, initial population generation, fitness calculation, selection, population elimination mechanism, generation of new generation population, and algorithm.

Algorithm 2: Improved Genetic Algorithm (IGA)

Input: set of jobs J , population size P , crossover probability p_c , mutation probability p_m , maximum generation Gen .

Output: scheduling order $Order$

Start construct initial generation //use three rules for initial generation

 Use GenerationRule1 (P , 80%)

 Use GenerationRule2 (P , 10%)

 Use GenerationRule3 (P , 10%)

Finish construct initial generation, get population p_0

Initialize best solution $best^*$

While generation $gen < Gen$ **do**

 Calculate fitness and assign jobs using OSA(p)

If minimize cost calculated by OSA(p)

 Order* ← Record task execution order based on minimum cost

End If

 Update worst individual (p)

 Roulette selection (p)

If random number less than or equal to crossover probability p_c

 Do partially mapped crossover operation

End If

If random number less than or equal to crossover probability p_m

 Do mutation operation

End If

Get new population p

$gen = gen + 1$

End While

Order ← Select minimum cost in order*

Return Order

3.2.1. Representation

In order to use the genetic algorithm, we will represent the optimized task sequence in an encoded way. Our improved genetic algorithm is represented by the processes in each job. The advantage of this representation is that the genetic algorithm can be directly mapped to a specific ASPDPLSP without the need to again generate a scheduling sequence on the basis of the results obtained by the genetic algorithm. Each gene in an individual of a genetic algorithm represents a job. The total length of the genes in an individual depends on the total number of processes.

Chromosome [1 2 4 3 2 1 3 1 2 4]
 Index 1 2 3 4 5 6 7 8 9 10

For example, suppose a chromosome is denoted as [1 2 4 3 2 1 3 1 2 4]. There are four jobs in this chromosome. There are three processes in the first and second jobs, and two processes in the third and fourth jobs. Among them, 1 means that the work belongs to the first job. Here, the first occurrence of 1 indicates the first operation of the first job, the second occurrence is the second operation of the first job, and so on.

3.2.2. Initial Population

The generation of the initial population affects the evolution speed of the genetic algorithm. All randomizations may result in re-searching, where it is difficult to improve the optimization. We use three strategies to generate the initial population. These three strategies produce an initial population at a rate of 80%, 10%, and 10%, respectively. The first strategy is called GenerationRule1, which is the method in which the individual is randomly generated by the single genetic algorithm. The second strategy is called GenerationRule2, and operations of all the tasks are sorted in a positive order. The first process of all the tasks is ranked first, followed by the analogy. The top 25% of these individuals are equal, and the remaining 75% of the chromosomal fragments are generated randomly. The third strategy is called GenerationRule3, which is sorted in the reverse order according to the operation of all the jobs. The last process of all the jobs is ranked last and then forwarded in turn. After this sorting is obtained, the latter 25% adopt the same sorting sequence, and the remaining 75% are generated randomly.

- GenerationRule1: Randomly generate individuals in the simplest way.
- GenerationRule2: The first 25% of the gene fragments in the chromosome are generated according to the positive order of the process, and the remaining 75% of the gene fragments are generated randomly.
- GenerationRule3: The last 25% of the gene fragments in the chromosome are generated in the reverse order of the process, and the remaining 75% of the gene fragments are generated randomly.

Using these three initial population generation strategies can increase the diversity of individual populations while satisfying the initial solution quality. Rules 2 and 3 generate segments of individuals in a symmetrical manner. After obtaining the initial population, the fitness function is calculated and the population improvement process of the genetic algorithm is started.

3.2.3. Fitness Calculation

The criterion for evaluating whether the population of the genetic algorithm is improved is the determination of whether the fitness function is improved by calculating the fitness function. The fitness function is obtained in ASPDPLSP by calculating the objective function given in Section 2.2. Because individuals with a good performance in genetic algorithms correspond to a greater fitness, we need to take the inverse function of the objective function as the individual's fitness in the IGA.

3.2.4. Selection

The selection operation is the premise of selecting the genetic operation, and the crossover and mutation operations are performed on the selected individuals. The IGA chooses to use roulette, which is determined by the individual's fitness. When the number of individuals in the population is P , the gain of the k -th individual is fit_k . As the optimization goal is to find the minimum value of the objective function, in order to make the selection of the better individual easier, it is necessary to select each individual by the inverse number. Therefore, the selection formula can be expressed as follows:

$$p(t) = fit_t / \sum_{i=1}^P fit_i \quad (6)$$

The advantage of using roulette to make a choice is that individuals who perform well in the population have a greater probability of being selected, while individuals who do not perform well are difficult to select and may even be removed from the population through the elimination mechanism introduced in our next section. Figure 2 shows the roulette selection process for five individuals, with the highest fitness individual (Individual 3) having the highest probability of being selected.

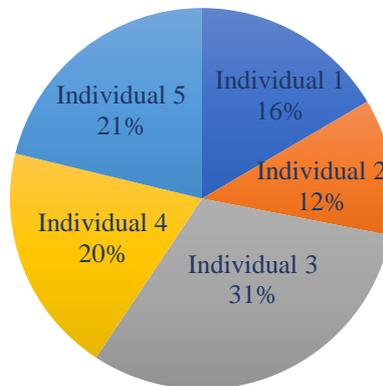


Figure 2. Roulette selection.

3.2.5. Elimination Mechanism

The elimination mechanism is designed to allow genetic algorithms to be optimized at a faster rate. In general, individuals with a poor performance are replaced. In the proposed IGA, the design of the elimination mechanism is based on the order of the individual returns in each generation of the populations, the selection of the worst performing individuals for replacement, and the replacement of randomly generating a chromosome to obtain new individuals. The phase-out mechanism is a promising method to quickly improve individual populations and takes up very little time. We only eliminate the worst performing individuals in each generation. This implies that IGA will accept many unsatisfactory solutions. This method of accepting inferior solutions is very beneficial for producing individuals with a particularly good performance.

3.2.6. Offspring

Genetic algorithms use genetic manipulation to obtain new structural chromosomes through the process of population multi-generation improvement. The genetic operations used are crossovers and mutations. These operations are the core part of the IGA. It is difficult to improve the population into a local search by using the process of mimicking chromosomal changes to change an individual's performance and increase the diversity of the population. Crossover is a change to a gene fragment in a chromosome. The mutation operation changes individual alleles in the chromosome. The frequency of manipulation of the two genes is also different, which is similar to the genetic evolution of the organism. In the generation of new populations, the generation of new chromosome structures by crossover operations is frequent, and the probability of gene mutations is considerably lower than that of gene crossovers.

Crossover: Crossover selects a segment of a chromosome in a population to make changes. The crossover used in this study is the partially mapped crossover (PMX). The PMX approach randomly selects two segments in an individual for a positional exchange. The reason for using this crossover operation is that after the crossover the new individual is still legal and can be directly used for subsequent mission planning and population improvement processes. The PMX process is divided into two steps: First, select an individual in the population. This process is a sequential selection of each individual in the population. To control the intersection of the population, the crossover probability is used to control the degree of change in the population. Then, a random number between 0 and 1 is generated and compared with the crossover probability. If it is less than or equal to the crossover

probability, the PMX operation is performed, and if it is greater than the crossover probability, this crossover operation is not performed.

Mutation: Mutation is a genetic operation in which a small probability occurs to alter the existing chromosome structure in a population to achieve an optimized effect. Before performing the mutation operation, we need to use a random number to determine whether to perform the mutation operation. If the mutation is performed, the two gene positions are randomly selected, and the positions of the two genes are exchanged to produce a completely new chromosome structure. Although the number of mutations that occur is not large, it can play a good role in improving the overall performance of the population.

3.2.7. Stop Criterion

The setting of the termination condition determines the overall running time of the algorithm. The condition for the end of the proposed IGA algorithm is to end the operation of the algorithm after reaching the pre-set genetic algorithm population improvement algebra. The termination condition has a considerable influence on the final optimization effect. The small optimization algebra ensures that the IGA does not fully exert the optimization effect, and the evolutionary algebra is too large to be prone to a waste of computing resources. The optimization algebra of a specific IGA needs to be determined according to the size of the search space. In ASPDPLSP, the total number of processes and the number of machines in all the jobs are determined together.

4. Computational Result

The OSA and IGA proposed in Section 3 were implemented using Matlab2017a on a desktop with Core i7-7700 3.6-GHz Central Processing Unit (CPU). We selected a large number of experimental examples for the experimental validation. In order to test the effect of the more-convincing test algorithm, we chose two heuristic algorithms and a meta-heuristic algorithm as the comparison algorithms. The two heuristic algorithms were Heuristic1 and Heuristic2, respectively. These two algorithms were adapted and matched according to this problem on the basis of the heuristic algorithm of other scheduling problems. Heuristic1 was selected according to the length of all the operations waiting in the process, and the task with a long waiting time was scheduled first, because the waiting time could be the processing time of the other jobs. Heuristic2 sorted according to the processing time of the operations, and first scheduled jobs with a long processing time, because these jobs had a significant effect on the final task completion time. The meta-heuristic algorithm for the comparison was the neighborhood local search (NLS).

We selected different numbers of jobs in the experimental part, and a number of different machines was used as the experimental dataset. In terms of jobs, we designed a dataset containing 10 jobs and 35 operations, 13 jobs and 80 operations, 17 jobs and 120 operations, 20 jobs and 140 operations, and 25 jobs and 180 operations. In the manufacturing workshop, we converted the professional staff into a type of machine, with six types (5, 10, 15, 20, 25, and 30) of production shop datasets. It is represented in the form of MWX/Y in each table, where X represents the number of tasks and Y represents the number of machines.

First, we analyzed the scheduling results of different jobs in 10 machines. MW1/1 means 10 tasks, 10 machines, MW2/1 means 20 tasks, 10 machines, and so on. Every scenario in this section runs 30 times. The results are shown in Table 1. In the production workshop of five machines, from 35 operations to 180 operations, neither the IGA algorithm nor the NLS algorithm could stably achieve the optimal solution. This reflected the lack of capacity to complete these jobs with 10 machines. As the size of the task increased, the degree of volatility of the algorithm also showed an increasing trend. When the number of tasks was 25, the total cost increased rapidly as compared to when the number of tasks was 20, indicating that the production process had exceeded the running capacity of the machine.

Table 1. Scheduling metric of different jobs in 10 machines.

Name	IGA			NLS			Heur.1	Heur.2
	Min.	Ave.	SD.	Min.	Ave.	SD.		
MW1/1	1476	1507.6	24.53	1500	1587.0	67.20	1888	1944
MW2/1	6107	6424.7	147.89	6244	6972.9	428.96	7866	6591
MW3/1	15,647	16,322.1	318.33	15,824	16,966.6	623.03	16,142	17,799
MW4/1	20,972	21,493.6	297.91	22,119	22,796.3	571.71	20,863	24,810
MW5/1	37,511	38,131.2	315.84	39,124	40,657.9	1071.24	32,869	44,726

*IGA: Improved Genetic Algorithm; NLS: Neighborhood Search.

Thereafter, we compared the results of different mission planning when the number of machines was 20, as shown in Table 2. MW1/2 means 10 tasks, 20 machines, MW2/2 means 20 tasks, 20 machines, and so on. This table shows that when the number of tasks was 10, the scheduling results of the IGA and NLS algorithms reached an optimal solution in each search optimization process. In most cases, the two-element heuristics performed better than the two heuristics, but Heuristic1 performed better than the NLS algorithm when the number of tasks was 25. This implied that heuristic algorithms could be used to solve production scheduling problems in certain specific scenarios.

Table 2. Scheduling metric of different jobs in 20 machines.

Name	IGA			NLS			Heur.1	Heur.2
	Min.	Ave.	SD.	Min.	Ave.	SD.		
MW1/2	1224	1224.0	0.00	1224	1224.0	0.00	1346	1230
MW2/2	4579	4602.0	21.42	4580	4676.7	46.35	5106	4617
MW3/2	10,015	10,192.3	96.47	10,097	10,463.7	170.69	10,834	11,456
MW4/2	12,306	12,481.6	92.67	12,426	12,810.9	207.54	12,796	13,661
MW5/2	19,318	20,723.2	146.74	21,147	21,502.0	322.58	19,463	22,432

Next, we also analyzed the task scheduling results when the number of machines was 30, as shown in Table 3. MW1/3 means 10 tasks, 30 machines, MW2/2 means 20 tasks, 30 machines, and so on. When the number of tasks was 10 or 13, 30 machines could achieve an optimal scheduling in both cases. When the number of tasks was 10, the four algorithms obtained the optimal solution, which reflected the existence of excess production resources, and there was no need for scheduling. It was only possible to put each operation on the machine that could be executed.

Table 3. Scheduling metric of different jobs in 30 machines.

Name	IGA			NLS			Heur.1	Heur.2
	Min.	Ave.	SD.	Min.	Ave.	SD.		
MW1/3	1188	1188.0	0.00	1188	1188.0	0.00	1188	1188
MW2/3	4434	4434.0	0.00	4434	4434.0	0.00	4530	4434
MW3/3	9312	9338.7	13.18	9364	9394.4	25.71	9498	9514
MW4/3	10,681	10,751.2	35.26	10,783	10,871.2	66.29	11,040	10,966
MW5/3	16,260	16,446.6	118.37	16,428	16,792.5	258.69	16,864	16,738

After comparing the same number of machines for production scheduling for different numbers of jobs, we analyzed the appropriate number of machines for each task number to advise on aerospace shell production. We selected 10, 13, 17, and 20 tasks to comprise the experimental datasets. All of the production machines were also included in the experimental datasets. The results are shown in Figure 3.

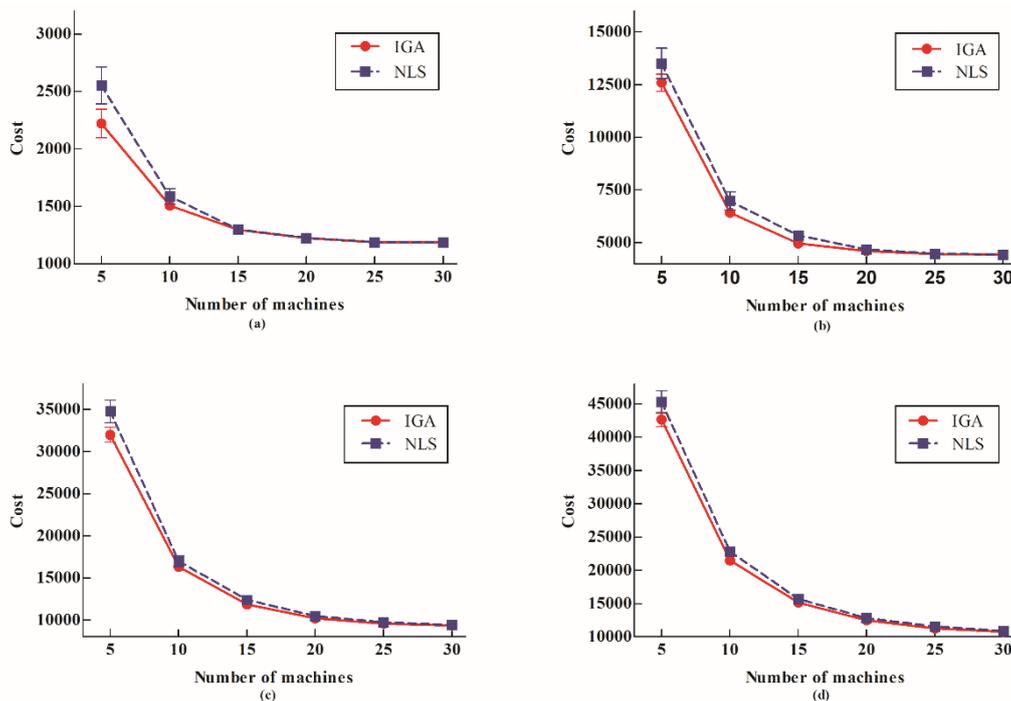


Figure 3. Scheduling results for different machines. There are 10 tasks in (a). The number of tasks in (b) is 13. Experiment under 17 tasks is shown in (c). 20 tasks are tested in (d).

Figure 3 shows that when the number of operations was 10, the scheduling result of the IGA in 10 machines was very volatile, and the optimal scheduling result was achieved when the number of machines was 15 and the NLS algorithm was used. In the scheduling results of 15 machines, it was not possible to achieve an optimal production plan for each scheduling. In terms of the production cost, the scheduling results of 20 machines were not significantly better than those of 15 machines. When the cost of the machine was considered, 10 tasks could achieve the optimal production plan by using 15 machines. Figure 3b shows that the use of 30 machines allowed the production scheduling results of 13 tasks to be optimal each time. At the same time, when the number of machines was increased from 15 to 30, the value of the optimization function was reduced to a small extent. From the perspective of rationality, the production process of 13 tasks could be completed well by using 10 machines. Next, Figure 3c shows that when the number of machines was increased from 5 to 15, the target function value was reduced by more than half. When the number of machines reached 15, the production cost decreased with an increase in the number of machines. In Figure 3d, in the case of 20 machines, the task execution plan was more suitable for production in terms of both the optimization effect and the resulting volatility. The NLS algorithm was used in the case of 20 tasks. When the number of machines was less than 30, the volatility of the results was large. The volatility of the proposed IGA algorithm was less than that of the NLS algorithm.

In order to calculate the improvement of the IGA with respect to the optimization results, we used the Dev. index to evaluate the optimization effect; the calculation formula can be expressed as follows:

$$Dev. = (average_{IGA} - average_{NLS}) / average_{NLS} \quad (7)$$

In ASPDPLSP, the involvement of personnel is unique to the problem. In the case of 20 tasks and 20 machines, we first removed all the staff and added staff one by one. The results are shown in Table 4. This table shows that when the number of workers increased from 1 to 3, the value of the optimization function dropped rapidly, and when the staff increased further, the rate of the cost reduction slowed down. This reflected that the number of staff members was not as good as possible. In the case of 20 jobs, it was reasonable to have three staff members involved in the production process. A comparison

with the NLS algorithm revealed that the average of the IGA algorithm planning results was better than that of the NLS algorithm (2.5% to 7.2%).

Table 4. Scheduling results under different staff.

Name	IGA			NLS			Dev. (%)
	Min.	Ave.	SD.	Min.	Ave.	SD.	
MW2/2-1	20,386	20,728.8	256.38	21,306	22,343.7	744.39	−7.23
MW2/2-2	13,357	13,540.1	135.03	13,737	14,076.7	246.41	−3.81
MW2/2-3	12,415	12,552.2	102.80	12,524	12,999.6	211.51	−3.44
MW2/2-4	12,306	12,481.6	92.67	12,426	12,810.9	207.54	−2.57
MW2/2-5	12,154	12,274.3	70.87	12,566	12,799.8	193.76	−4.11

In our OSA, the task scheduling rules were set to improve the task production plan. To verify the validity of the heuristic rules, we compared the results of using AssignmentRule and not using AssignmentRule, as shown in Table 5. This table shows that the use of scheduling rules was very effective in achieving both optimal results and the average optimization effect of the algorithm. The average of the scheduling results implied that the usage rules could reduce the cost by 1.7%–4.2%. Therefore, the use of scheduling rules improved the optimization effect considerably, which was meaningful for reducing the production costs.

Table 5. Comparison with AssignmentRule.

Name	IGA			Unassign_IGA			Dev. (%)
	Min.	Ave.	SD.	Min.	Ave.	SD.	
MW1/2	1224	1224.0	0.00	1224	1224.0	0.00	0.00
MW2/2	4579	4602.0	21.42	4598	4682.5	52.81	−1.72
MW3/2	10,015	10,192.3	96.47	10,207	10,495.3	193.95	−2.89
MW4/2	12,306	12,481.6	92.67	12,461	12,934.9	294.4303162	−3.50
MW5/2	19,318	20,723.2	146.74	20,985	21,785.1	537.03	−4.87

The above-mentioned experimental results demonstrated that the proposed OSA for planning the task execution and the IGA for optimizing the task sequence were very effective in solving ASPDPLSP. Furthermore, IGA was better than the three other comparison algorithms in terms of the optimization effect, and the stability of the algorithm was better than that of another meta-heuristic algorithm. We also carried out experiments on the selection of the number of machines under different scale tasks. Through these experiments, we found a suitable number of production machines for each task size and could thus determine the appropriate production environment. We also analyzed the effect of personnel on the production process. The results showed that having a large staff number did not lead to the best production plan. We also validated the effect of the assignment rule in OSA on the optimization results, which was very effective in improving production planning and reducing costs. In summary, the two proposed algorithms could solve the ASPDPLSP problem well.

5. Conclusions

In this study, we investigated the aerospace shell product digital production line scheduling problem in an actual production process. This problem contains two new features. One is that some tasks need to wait for a certain period of time before they can be started. Second, the production and processing process requires the professional staff to cooperate. These two characteristics make the problem more complicated than the original FJSP problem, and an effective solution algorithm needs to be set. To this end, we have proposed two algorithms: one is the operation scheduling algorithm for task planning, and the other is an improved genetic algorithm for optimizing task sequences. In order to verify the effect of the algorithm, we conducted a number of experiments. The experimental results

showed that the proposed OSA and IGA could solve ASPDPLSP and resulted in a good production plan improvement. These two algorithms could support the actual production of the shell product well, while reducing the production cost and ensuring the production efficiency.

Author Contributions: Writing—original draft preparation, Q.W., H.L.; writing—review and editing, Q.W., J.X., Y.S.; investigation, Q.W., Z.Z.; methodology Q.W.

Acknowledgments: This work was supported by the National Natural Science Foundation of China under Grants 71501181, 71501179, and 71501180.

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analysis, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Corominas, A.; Alberto, G.; Néstor, A.; Pastor, R. A multistage graph-based procedure for solving a just-in-time flexible job-shop scheduling problem with machine and time-dependent processing costs. *J. Oper. Res. Soc.* **2018**, *70*, 620–633. [[CrossRef](#)]
2. Wu, X.; Liu, X.; Zhao, N. An improved differential evolution algorithm for solving a distributed assembly flexible job shop scheduling problem. *Memet. Comput.* **2018**, *6*, 1–21. [[CrossRef](#)]
3. Zhang, X.; Ji, Z.; Wang, Y. An improved sfla for flexible job shop scheduling problem considering energy consumption. *Mod. Phys. Lett. B* **2018**, 1840112. [[CrossRef](#)]
4. Kavitha, S.; Venkumar, P.; Rajini, N.; Pitchipoo, P. An efficient social spider optimization for flexible job shop scheduling problem. *J. Adv. Manu. Sys.* **2018**, *17*, 181–196. [[CrossRef](#)]
5. Naiming, X.; Nanlei, C. Flexible job shop scheduling problem with interval grey processing time. *Appl. Soft. Comput.* **2018**, *70*, 513–524. [[CrossRef](#)]
6. Defersha, F.M.; Bayat-Movahed, S. Linear programming assisted (not embedded) genetic algorithm for flexible jobshop scheduling with lot streaming. *Comput. Ind. Eng.* **2018**, *117*, 319–335. [[CrossRef](#)]
7. Xu, Y.; Wang, L.; Wang, S.Y.; Liu, M. An effective teaching–learning-based optimization algorithm for the flexible job-shop scheduling problem with fuzzy processing time. *Neurocomputing* **2015**, *148*, 260–268. [[CrossRef](#)]
8. Gao, K.Z.; Suganthan, P.N.; Pan, Q.K.; Chua, T.J.; Chong, C.S.; Cai, T.X. An improved artificial bee colony algorithm for flexible job-shop scheduling problem with fuzzy processing time. *Expert Syst. Appl. Int. J.* **2016**, *65*, 52–67. [[CrossRef](#)]
9. Moghadam, A.M.; Wong, K.Y.; Piroozfard, H. An Efficient Genetic Algorithm for Flexible Job-Shop Scheduling Problem. In Proceedings of the 2014 IEEE International Conference on Industrial Engineering & Engineering Management, Bandar Sunway, Malaysia, 9–12 December 2016; 2014; pp. 1409–1413. [[CrossRef](#)]
10. Quan, K.P. An effective discrete harmony search algorithm for flexible job shop scheduling problem with fuzzy processing time. *Int. J. Prod. Res.* **2015**, *53*, 1–16. [[CrossRef](#)]
11. Birgin, E.G.; Ferreira, J.E.; Ronconi, D.P. List scheduling and beam search methods for the flexible job shop scheduling problem with sequencing flexibility. *Eur. J. Oper. Res.* **2015**, *247*, 421–440. [[CrossRef](#)]
12. Geyik, F.; Dosdoğru, A.T. Process plan and part routing optimization in a dynamic flexible job shop scheduling environment: An optimization via simulation approach. *Neural Comput. Appl.* **2013**, *23*, 1631–1641. [[CrossRef](#)]



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).