Hybrid Particle Swarm Optimization Algorithm for Process Planning

Xu Zhang 1, Pan Guo 1, Hua Zhang 2 and Jin Yao 3,*

1 Business School, Sichuan University, Chengdu 610064, China; kinich_ahua@outlook.com (X.Z.); guopan1227@gmail.com (P.G.)
2 School of Economics and Management, Zhaoqing University, Zhaoqing 526061, China; allenzhang.scu@gmail.com
3 School of Mechanical Engineering, Sichuan University, Chengdu 610064, China
* Correspondence: yaojin@scu.edu.cn
Received: 6 September 2020; Accepted: 6 October 2020; Published: 11 October 2020

Abstract: Process planning is a typical combinatorial optimization problem. When the scale of the problem increases, combinatorial explosion occurs, which makes it difficult for traditional precise algorithms to solve the problem. A hybrid particle swarm optimization (HPSO) algorithm is proposed in this paper to solve problems of process planning. A hierarchical coding method including operation layer, machine layer and logic layer is designed in this algorithm. Each layer of coding corresponds to the decision of a sub-problem of process planning. Several genetic operators of the genetic algorithm are designed to replace the update formula of particle position and velocity in the particle swarm optimization algorithm. The results of the benchmark example in case study show that the algorithm proposed in this paper has better performance.

Keywords: process planning; hybrid particle swarm optimization; hierarchical coding

1. Introduction

Process planning runs through the entire manufacturing process and has a significant impact on the series of important indicators such as product production cycle, production cost and product quality. In the earlier traditional process planning, due to the limitation of manufacturing resources, process planners would generate a relatively fixed and single process route for each product, which would often cause adverse effects such as resource conflicts, “bottleneck” processes or unbalanced load of processing equipment in actual production, thus reducing the efficiency of production activities. With the transition from traditional manufacturing mode to modern advanced manufacturing mode, the flexibility of the manufacturing system is getting higher and higher, and the flexible manufacturing system come into being. In a flexible manufacturing system, there are multiple process route options for the same product, which are embodied in: (1) Operation selection flexibility, that is, the product can be produced by a combination of multiple different operations. (2) Operation sequence flexibility, that is, the processing sequence between some operations can be exchanged. (3) Machine selection flexibility, that is, some operations can be processed by different machines or equipment. There may be more than one process route for a product, but only one can be adopted in the end [1]. Flexible process routes can improve the flexibility of production activities, provide more options for manufacturing resources, and expand the optimization space of manufacturing process. So that the production efficiency of manufacturing enterprises can be improved effectively. Flexible manufacturing technology and production environment provide a broader optimization space for process planning, but at the same time it also increases the difficulty of optimizing process planning issues. The process planning problem studied in this paper mainly...
refers to the optimization decision of the above three dimensions for some optimization objects, so as to obtain the optimal process route plan. In other words, it is a problem of searching for the optimal process plan in the feasible process plan space according to certain restrictions. Process planning is a typical combinatorial optimization problem, which has been proved to be NPC (non-deterministic polynomial complete) problem [2]. The traditional accurate solution method is difficult to solve the process planning problem with a large scale.

In view of this, this paper proposes an improved hybrid particle swarm optimization algorithm to solve the process planning problem. This algorithm uses the crossover and mutation operation of genetic algorithm to replace the particle position and velocity update formula of particle swarm optimization algorithm. It is a hybrid algorithm of particle swarm optimization algorithm (PSO) and genetic algorithm (GA). The algorithm in this paper is mainly improved in two aspects: (1) Coding method: A hierarchical coding method including operation layer, machine layer and logic layer are designed, and each layer of coding corresponds to the decision of a sub-problem of process planning. This method can reduce the decoding difficulty of existing coding methods and increase the information contained in chromosome genes. Among them, the design of logic layer coding is conducive to the layered cross operation between different individuals, and has a significant contribution to the performance of the algorithm in this paper. (2) Genetic operator: Different cross-mutation operators are proposed for each layer of coding, and different search strategies are adopted, which greatly enhances the search performance of the algorithm. Finally, the benchmark problem is used to validate the algorithm and test its performance. The results show that the algorithm presented in this paper has some advantages.

The rest of this paper is formulated as follows. Section 2 provided a brief literature reviewed of optimization method for process planning. In Section 3, the description of process planning problem was given. In Section 4, the mixed integer programming model of process planning was established. In Section 5, a hybrid particle swarm optimization (HPSO) algorithm for process programming was designed. In Section 6, two benchmark examples were used to verify the effectiveness of the algorithm and test its performance. Section 7 presents some conclusions and some proposals for future research.

2. Literature Review

Process planning is a type of typical combinatorial optimization problem, which belongs to NPC problem. For solving process planning problem, the traditional method has great limitations, so that most of researchers use approximate methods to solve the problems with large scale. A large number of modern manufacturing enterprises use computer aided process planning (CAPP) system for process planning, the optimization method of the process route is the core of the CAPP system. The existing methods in related research mainly include expert system (ES), artificial neural network (ANN), multi-agent system (MAS), petri nets, fuzzy sets and fuzzy logic, heuristic algorithms, etc. [3,4]. Heuristic algorithms have outstanding characteristics such as diverse optimization mechanisms, high optimization efficiency, relatively simple implementation, and easy embedding, and are the most widely used in process planning.

Li et al. [5] used a mixture of genetic algorithm and simulated annealing algorithm to optimize the process plan of prismatic parts, and achieved good results. The GA is carried out in the first stage to generate some initially good process plans. Based on a few selective plans with Hamming distances between each other, the SA algorithm is employed to search for alternative optimal or near-optimal process plans. The calculation results of the calculation example show that the hybrid algorithm has better performance than the single algorithm. Singh et al. [6] studied the process planning problem based on part machining characteristics, and adopted genetic algorithms to optimize the optimal operation sequence, machine tool sequence and tool sequence under different conditions. Su et al. [7] proposed an edge selection strategy based on genetic algorithm to ensure the priority constraints of processes in process planning. The strategy based on edge selection could generate feasible solutions during initialization and ensure the generation of each feasible solution with acceptable probability, so as to improve the convergence efficiency of genetic algorithm. Delolme et al. [8] adopted genetic
algorithm to generate process optimization plan, and made multi-criteria optimization decision based on them. Guo et al. [9] used particle swarm optimization algorithm to solve the problem of process sequencing in process planning, and compared the results of the improved particle swarm algorithm with the results of genetic algorithm and simulated annealing algorithm, and pointed out the different characteristics of the three algorithms. Li et al. [2] designed a multi-layer coding method based on processing characteristics, proposed a hybrid particle swarm optimization algorithm to solve process planning, and compared the results of multiple calculation examples to show the advantages of the algorithm. Wang et al. [10] used an improved particle swarm optimization algorithm to solve the process planning problem. The algorithm is improved on the basis of the traditional particle swarm algorithm, adding expansion operators for different parts of the particle code, which improves the search performance of particles. Miljković et al. [11] solve the multi-objective process planning problem which considering the indexes of minimize production cost and production time with an improved particle swarm optimization algorithm. Wu et al. [12] designed a hybrid algorithm of particle swarm optimization and genetic algorithm to solve the process planning problem, and verified the effectiveness of the algorithm with an example. Ma et al. [13] used the simulated annealing algorithm to solve the process planning. Li et al. [14] proposed a tabu search algorithm with embedded mixed constraint processing method for process planning, and compared it with the simulated annealing algorithm in a calculation example, which proved that the algorithm has more advantages. Lian et al. [15] designed a multi-dimensional tabu search algorithm to solve the process planning according to the characteristics of the process planning problem. Liu et al. [16] and Hu et al. [17] used ant colony optimization algorithm to optimize the process route and achieved relatively ideal results. Chan et al. [18] used immune algorithms (IA) to solve process planning problems. Algorithms in the above literatures include genetic algorithm (GA), particle swarm optimization (PSO) algorithm, simulated annealing (SA) algorithm, tabu search (TS) algorithm, ant colony optimization (ACO) algorithm, artificial immune algorithm (IA) and so on. There are some more novel methods except those common heuristic algorithms for solving process planning, such as the intelligent water drop algorithm used by Gao et al. [19], and DP-like heuristic algorithm used by Jin et al. [1] etc.

In the above-mentioned literatures, the algorithm encoding method can be divided into a method based on processing characteristics and a method based on processing operations. As part of the manufacturing features include multiple operations, the coding method of the process route plan for the process features reduces the computational complexity of problem solving and is more conducive to random search, but it also limits the optimization search space of the problem. Although the process-based coding method is also subject to the order constraints of the manufacturing features, it allows the processing processes that have no constraint relationship to be inserted between different processes of a certain manufacturing feature, and has more optimization space. Therefore, we design a new coding method to express the flexible process route, and proposes an improved hybrid particle swarm algorithm to solve the process planning problem in this paper.

3. Description of Flexible Process Route

For a more comprehensive description of the flexible process route of a product, multiple aspects of information need to be expressed, including the manufacturing feature information of the product (such as the optional operations of manufacturing feature, feature sequence constraints, etc.), and available processing equipment information (such as the operations that can be performed by the machine, etc.) and the information of the operations in the product manufacturing feature which is processed on a specific machine (such as the processing time of a operation on a specific machine, etc.). In many related literatures, there are many kinds of methods to describe flexible process planning, which can be roughly divided into table-based expression methods, directed graph-based expression methods, operation segments-based expression methods, set theory-based expression methods, and feature-based expression methods, and so on. Among the more common representation methods are network graphs, petri nets, AND/OR graphs, and so on. Li et al. [20] proposed an acyclic unidirectional graph that combines AND/OR graphs with network graphs to
describe the flexible process route of products, which has the advantages of simplicity and intuitiveness. Under normal circumstances, using this method in conjunction with a table-based presentation method can more fully express the flexible process route of the product. According to the above method, the flexible process route of a product can be described as follows. Table 1 is the process information table of the product, and Figure 1 is the AND/OR network figure which described the flexible process route of the product. The product has 11 manufacturing features, 15 optional processes, and is processed on 6 optional machines.

Table 1. An example of a product process information.

| Feature | Optional Operation | Optional Machine | Process Time | Sequence Constraints |
|---------|--------------------|------------------|--------------|---------------------|
| F₁      | O₁                 | M₂, M₃, M₄      | 18, 17, 15   | Before ALL          |
| F₂      | O₂                 | M₂, M₄, M₅      | 10, 12, 14   | Before F₃           |
| F₃      | O₃                 | M₁, M₂, M₃, M₄ | 12, 15, 18, 20 | Before F₁₀, F₁₁   |
| F₄      | O₄                 | M₃, M₄, M₅      | 17, 16, 19   | Before F₁₀, F₁₁   |
| F₅      | O₅                 | M₂, M₄, M₆      | 18, 20, 22   | Before F₁₀, F₁₁   |
| F₆      | O₆                 | M₁, M₂, M₃      | 20, 22, 18   | Before F₁₀, F₁₁   |
| F₇      | O₇                 | M₂, M₃, M₄      | 14, 16, 14, 18 | Before F₃, F₉   |
| F₈      | O₈→O₉              | M₂, M₃, M₅→M₄, M₅, M₆ | 11, 10, 9→10, 12, 8 |
| F₉      | O₁₀                | M₁, M₂, M₃, M₄ | 25, 28, 20   |                     |
| F₁₀     | O₁₁                | M₂, M₄, M₅      | 15, 17, 19   |                     |
| F₁₁     | O₁₂                | M₁, M₂, M₃, M₄ | 20, 22, 25, 18 |                     |
| F₁₂     | O₁₃                | M₂, M₃, M₅      | 18, 15, 20   |                     |
| F₁₃     | O₁₄                | M₃, M₄, M₆      | 18, 20, 15   |                     |
| F₁₄     | O₁₅                | M₁, M₂, M₃, M₆ | 5, 5, 6, 4   |                     |

Figure 1. An example of a product AND/OR network figure.

In the figure, the starting point S and the ending point E are virtual nodes, O₁ represents the process node, and the process route between OR and JOIN symbolic connection represents the optional process routes, and only one of them can be selected. For example, it can be chose one of the two routes between [O₇] and [O₈, O₉]. Except the optional process route between OR and JOIN, it is
necessary to traverse other operation nodes to complete the production of the product. Each operation needs to choose an available processing machine. For the example in Figure 1, a feasible process route plan for the product is:

\[ S \rightarrow (O_1, M_1) \rightarrow (O_2, M_2) \rightarrow (O_3, M_3) \rightarrow (O_4, M_4) \rightarrow (O_5, M_5) \rightarrow (O_6, M_6) \rightarrow (O_7, M_7) \rightarrow (O_8, M_8) \rightarrow (O_9, M_9) \rightarrow (O_{10}, M_{10}) \rightarrow (O_{11}, M_{11}) \rightarrow (O_{12}, M_{12}) \rightarrow (O_{13}, M_{13}) \rightarrow (O_{14}, M_{14}) \rightarrow (O_{15}, M_{15}) \rightarrow E \]

Based on the expression method of this flexible process route above, we can describe the procedure of process planning activity as shown in Figure 2.

![Figure 2. The procedure of process planning activity.](image)

4. Mathematical Model

4.1. Problem Assumptions and Related Symbol Definitions

The main task of process planning is to determine the technical requirements and processing flow of a product from the rough or raw material to the final product. In the environment of flexible production system, processing methods and technologies are not unique, which makes a product have multiple optional process routes. Products with flexible processing technology include a variety of manufacturing features (such as planes, cylindrical surfaces, holes, etc.). Each manufacturing feature can be processed by multiple technologies. Each optional processing technology includes one or more processing operations. There may be sequence constraints between operations, and each operation can be processed on multiple alternative processing machines. In other words, the product has operation selection flexibility, processing machine selection flexibility, and operation sequence selection flexibility. To generate a complete process route plan, it is necessary to make decisions in these three dimensions. The process planning problem studied in this paper mainly refers to the optimization decision of the above three dimensions for some optimization objects, so as to obtain the optimal process route plan. In order to better carry out theoretical research and mathematical modeling, it is assumed that the following assumptions should be satisfied in the flexible process planning problem of this paper.

1. Each machine can only process one operation of the product at a time;
2. The product can only be processed on one machine at a time;
(3) Each machine or equipment can be used at the initial moment;
(4) The processing time of each operation is a fixed constant value;
(5) The preparation time of each operation on the machine is included in the processing time, and
has no relations to the operation sequence;
(6) After an operation of the product is completed on the machine, it is immediately transferred to
the machine of the next operation for processing, and the transportation time of the product
between different machines is a certain constant value.

The description of the relevant parameters for process planning issues studied in this paper are
shown in Table 2.

Table 2. The relevant parameter descriptions of the process planning.

| Parameter | Description |
|-----------|-------------|
| $H$       | A huge positive number; |
| $n$       | Total number of operations; |
| $m$       | Total number of processing machines; |
| $L$       | Total number of operation combinations; |
| $l$       | Alternative process combination number; |
| $n_i$     | Total operation quantity in the $l$-th operation combination; |
| $j, j_1, j_2, j'$ | Operation number; |
| $k, k_1, k_2, k'$ | Machine number; |
| $O_{lj}$  | The $j$-th operation in $l$-th optional operation combination; |
| $O_{ljk}$ | Process $O_{lj}$ on machine $k$; |
| $S_{ljk}$ | Start time of $O_{ljk}$; |
| $C_{ljk}$ | Completion time of the product; |
| $C_{ljk}$ | Completion time of $O_{ljk}$; |
| $t_{ljk}$ | Processing time of $O_{ljk}$; |
| $t_{k_l^\text{trans}}$ | Transportation time of the product from machine $k_1$ to machine $k_2$; |

$X_{l} = \begin{cases} 
1 & \text{If the } l\text{-th operation combination is selected;} \\
0 & \text{Otherwise;}
\end{cases}$

$Y_{ljk} = \begin{cases} 
1 & \text{If } O_{lj} \text{ is processed on machine } k; \\
0 & \text{Otherwise;}
\end{cases}$

$F_{k_l^j} = \begin{cases} 
1 & \text{If operation } j_1 \text{ is before operation } j_2; \\
0 & \text{Otherwise;}
\end{cases}$

$T(k,k') = \begin{cases} 
1 & \text{If the machine of the next operation of } O_{ljk} \text{ is } k', \land k \neq k' \; ; \\
0 & \text{Otherwise}
\end{cases}$

4.2. Mixed Integer Programming Model for Process Planning

Process planning issues generally take the completion time, the production cost, and the
minimum number of machine conversions throughout the product processing process of the product
as evaluation indicators. In this paper, we select the completion time of the product as the
optimization goal:

$$f = \min C = \sum_{j=1}^{n} t_{ljk} \cdot X_{l} \cdot Y_{ljk} + \sum_{j=1}^{n} X_{l} \cdot Y_{ljk} \cdot t_{k_l^\text{trans}} \cdot T(k,k')$$

(1)
The first half of the formula (1) is the sum of the processing time of the operations on the machine, and the second half is the sum of the time for the product to switch between different machines.

Generally, the following constraints needs to be satisfied in the process planning problem:

(1) Only one combination of processing operations can be selected for the product:

$$\sum_{i=1}^{L} X_i = 1$$  \(2\)

(2) Each operation can only select to be processed one machine:

$$\sum_{k=1}^{n} Y_{jk} = 1 \quad \forall l \in [1,L]; \forall j \in [1,n_j]$$  \(3\)

(3) The completion time of the operation is equal to the sum of the start time of the operation and the processing time:

$$C_{jk} \cdot X_j \cdot Y_{jk} = S_{jk} \cdot X_j \cdot Y_{jk} + t_{jk} \cdot X_j \cdot Y_{jk}$$

$$\forall l \in [1,L]; \forall j \in [1,n_j]; \forall k \in [1,m]$$  \(4\)

(4) The different operations of the product cannot be processed at the same time:

$$F_{hk} \cdot X_i \left( S_{hk} \cdot Y_{hk} - S_{hk} \cdot Y_{hk} \right) + H \left( 1 - F_{hk} \right) + H \left( 1 - Y_{hk} \right) \geq X_i \cdot Y_{hk} \cdot t_{hk}$$

$$\forall l \in [1,L]; \forall j, k \in [1, n_j] \land j_1 \neq j_2; \forall k_1, k_2 \in [1, m]$$  \(5\)

(5) Machine constraints, each machine can only process one operation at the same time:

$$F_{hk} \cdot X_i \left( C_{hk} \cdot Y_{hk} - C_{hk} \cdot Y_{hk} \right) + H \left( 1 - F_{hk} \right) + H \left( 1 - Y_{hk} \right) \geq X_i \cdot Y_{hk} \cdot t_{hk}$$

$$\forall l \in [1,L]; \forall j, k \in [1, n_j] \land j_1 \neq j_2; \forall k \in [1, m]$$  \(6\)

(6) The start and completion time of all operations are non-negative numbers:

$$S_{jk} \cdot X_j \cdot Y_{jk} \geq 0$$

$$C_{jk} \cdot X_j \cdot Y_{jk} \geq 0$$

$$\forall l \in [1,L]; \forall j \in [1,n_j]; \forall k \in [1,m]$$  \(7\)

(7) Process sequence constraint, the completion time of the preceding ones of the operation are greater than the start time of the operation:

$$\left( C_{jk} \cdot X_j \cdot Y_{jk} - S_{jk} \cdot X_j \cdot Y_{jk} \right) \cdot F_{hk} \geq 0$$

$$\forall j_1, j_2 \in [1, n_j] \land j_1 \neq j_2; \forall k_1, k_2 \in [1, m]$$  \(8\)

(8) In a process route plan, there is only one sequence between two different operations:

$$(F_{hk} + F_{jk}) \cdot X_i \leq 1$$

$$\forall l \in [1,L]; \forall j_1, j_2 \in [1,n_j] \land j_1 \neq j_2$$  \(9\)
5. Algorithm Design

For a complex product, the number of manufacturing features, the number of optional operation combinations for each manufacturing feature, and the number of optional machines for each operation is large. Different operation selection, operation sequence selection and machine selection combinations can form a large number of different processes route. In recent years, a variety of heuristic algorithms have been applied to the research of solving process planning problems and achieved some nice results. However, each algorithm has its advantages and disadvantages, and using a certain heuristic algorithm alone will inevitably suffer from performance limitations. Combining a variety of algorithms according to the characteristics of the problem to be solved, and complementing each other to form a hybrid optimization algorithm is an effective way to improve the performance of the optimization algorithm [21]. In many research fields, hybrid algorithms have been widely used and proved their superiority. PSO and GA are two very widely used metaheuristic algorithms, and they also show good performance in the optimization of process planning problems. This paper will use a hybrid particle swarm algorithm combining PSO and GA to solve the process planning problem.

5.1. Encoding and Decoding Methods

To use the HPSO algorithm to solve the process planning, the coding of the process route plan must be realized before the algorithm can be transformed into a computer program. In view of the fact that the HPSO algorithm in this paper uses genetic operations to update particles, the chromosome encoding method of GA can be considered to encode the feasible solution of the problem, so as to realize the expression of the process route plan. In the GA, the coding method determines the arrangement of the genes in the chromosome and the decoding method that transforms the genes in the chromosome into an effective solution to the problem. In addition, the coding method has played a decisive role in the operation methods of genetic operation operators such as crossover operators and mutation operators to a large extent. Generally speaking, in the operation of genetic algorithm, the main link that affects the efficiency of the algorithm is the calculation speed of the genetic operator. Therefore, designing a coding method that is conducive to genetic operations is the key to improving the efficiency of the algorithm in this paper. Through the study on the related literature of process planning, it is found that two representative coding methods are widely used.

(1) The integrated coding method, as shown in Figure 3. This method is a coding method based on the operation. The operation number and the machine number are integrated into a gene. The position of the gene indicates the processing sequence of the operation. The operation that appears in the chromosome indicates that the process is selected by the current process plan, otherwise, mean that the operation is not selected by the current routing scheme.

(2) The hierarchical coding method, as shown in Figure 4, represents a process planning encoding sample of the product in Figure 1. This method is a coding method based on product manufacturing features, using three digital sequences to map three-dimensional decision-making schemes in process planning. The first layer of coding is a feature sequence, each number in the sequence represents the product manufacturing feature number, and the position of the number represents the order of product manufacturing features. The second layer code is the process sequence, the length of the code is the same as the feature sequence, the number in the sequence...
indicates the processing process number selected by the product manufacturing feature at the corresponding position, and the second layer code can be decoded to obtain selection scheme of the operations in the process route of the product. The third layer of code is the machine sequence, the length of the code is the same as the number of all optional operations of the product, and each number indicates the sort number of the machine among the candidate machines for which the corresponding operation can select, not the machine number. Except for the first layer of feature sequence, this encoding method is an indirect encoding method, and its decoding complexity is higher than that of direct encoding. The advantage of this coding method is that the length of the chromosomes is the same, and each layer of coding is relatively independent, which is convenient for genetic operations; it can reduce the generation of illegal solutions after genetic operations to a certain extent; the three layer code correspond to the three-dimensional decision-making of process planning, which can better express the characteristics of the problem. However, this coding method is more complicated than the integrated method, and the chromosome information is not intuitive enough; the decoding method of process sequence and machine sequence is more difficult, which reduces the calculation efficiency of individual fitness function, which is also a prominent shortcoming of this coding method.

| Feature sequence | 5 3 1 4 8 7 6 9 |
|------------------|------------------|
| Process sequence | 1 2 1 3 2 1 1 2 |
| Machine sequence | 2 3 4 3 1 2 5 1 1 5 2 4 5 2 3 |

**Figure 4.** An example of hierarchical coding method for process planning problems.

Combining the advantages of the above two coding methods, a new coding method for process planning is designed in this paper. This method draws on the hierarchical representation principle of the hierarchical coding method, and directly encodes based on the information of each candidate processing operation of the product, which can be regarded as a combination of the integrated coding method and the hierarchical coding method. The coding method in this paper divides the coding of process plan into three layers the same as the hierarchical coding method, but the digital sequence of each layer adopts the form of direct coding, and the process route information expressed is also different to a certain extent. The three layers of coding are operation layer, machine layer, and logic layer, which correspond to the three dimensions of operation sequence selection, operation machine selection, and operation selection in process planning. Figure 5 is an example of the new coding method, from which the process route plan of the product can be obtained more intuitively. In Figure 6, the first layer of the code is the operation layer, the codes represent all the optional operation numbers of the product, and the positions of the number indicate the processing sequence of the operations; the second layer of the code is the machine layer, and the codes indicate the processing machine number selected by the operation corresponding to the position of the first layer; the third layer of the code is the logic layer, the code 1 indicates that the operation corresponding to the position of the first layer is selected by the process route plan, the code 0 indicates that the corresponding operation is not selected, to express the selection of the operations for the process route.

| Operation layer | 1 5 4 6 2 3 7 8 9 12 11 10 14 15 13 |
|-----------------|-------------------------------------|
| Machine layer   | 3 6 4 1 4 2 6 5 5 1 2 3 3 2 3      |
| Logic layer     | 1 1 1 1 1 0 1 1 0 1 1 1 1 0          |

**Figure 5.** An examples of new coding methods designed in this article.
The decoding method of this encoding method is very simple. It only needs to read the information of each operation according to the operation layer of the chromosome and delete the ones with a code ‘0’ in the logic layer to form the process route plan of the product. The code shown in Figure 6 can be decoded into the following process route:

\[(O_1, M_3) \rightarrow (O_3, M_6) \rightarrow (O_4, M_4) \rightarrow (O_6, M_9) \rightarrow (O_2, M_4) \rightarrow (O_3, M_2) \rightarrow (O_4, M_8) \rightarrow (O_9, M_4) \rightarrow (O_{10}, M_3) \rightarrow (O_{14}, M_3) \rightarrow (O_{15}, M_2)\]

5.2. Hybrid Particle Swarm Optimization Algorithm for Process Planning

Through the basic principles of the standard PSO, it can be found that the updates for the velocity and position of the particles are according to some specific formulas. Each particle in the algorithm is set as an \(R\)-dimensional vector \(X_i = (X_{i0}, X_{i1}, ..., X_{in})^T\). The components \(X_{ib}\) of the vector are independent of each other, and there is no sequence constraint between them. According to the velocity and position update formulas of the PSO, it is not difficult to find that the standard PSO is designed for the continuous optimization problem, and each component \(X_{ib}\) of the particle changes continuously. Although some kinds of discrete particle swarm optimization (DPSO) algorithms were proposed for solving discrete optimization problems, and some new velocity and position of particles update formulas were designed, most DPSO algorithms do not fully consider the specific characteristics of discrete combination optimization problems (such as the order constraints between the components vector of particle, etc.). As a result, the algorithm search efficiency is not high and there is a certain degree of redundant expression [22].

In a typical flexible process planning problem, the sequence constraints between operations need to be satisfied, and the update method for velocity and position of the particles in the PSO algorithm needs to be redesigned. In some classical discrete optimization problems, hybrid algorithms combining PSO and GA have been applied. This method abandons the velocity and position update model, and uses crossover and mutation operations of GA to update the particles. Li et al. [2] applied this method to the optimization of process planning problems, constructed a memory population in the iteration process to record the optimal position of particles which are in the population, and used a series of genetic operation operators and random search operator replaces the position and velocity update formula of the PSO. Adopting other methods to update the particles instead of the velocity and position update model in the standard PSO algorithm is a common method of HPSO. A large number of studies have shown that designing a particle update method that is more in line with the characteristics of the optimization problem can effectively improve the search performance of the PSO.

From what has been discussed above, we improve the coding method and genetic operator of the hybrid algorithm of PSO and GA, and applies it to the optimal solution of process planning problems in this article. According to the characteristics of the process planning problem, a new coding method is designed to express different process route schemes, which makes the decoding process easier and improves the efficiency of the algorithm; in the crossover operation, two paired crossover methods of individual random crossover and group optimal crossover are used to update the particle position, in order to further enhance the search performance of the algorithm. The main process framework of the improved HPSO algorithm proposed in this paper is shown in Figure 6.

The steps of the hybrid particle swarm optimization algorithm are as follows:

Step 1: According to the problem, generate the initial population \(\text{Initial}_\text{Pop}\) according to a certain method, so that each particle \(\text{Ind}_i(i)\) is a feasible solution, the number of initialization iterations \(\text{gen} = 0\), and the population size is \(N\), where \(i = 1, 2, ..., N\);
Step 2: Calculate the fitness \(\text{fit}_i(i)\) of each particle in the population according to the objective function of the problem;
Step 3: Update the individual extreme value \(\text{Pbest}_i(i)\) and the group extreme value \(\text{Gbest}(i)\) according to the fitness value of the particles;
Step 4: Determine whether the algorithm termination condition is satisfied, if so, output the calculation result; otherwise, execute the next step and increase the number of iterations, \( gen = gen + 1 \);

Step 5: Random crossover. Cross the particle \( Indi(i) \) with other random particles \( Indi(r) \), and calculate the fitness value \( RC\_fit(i) \) of the offspring \( RC\_Indi(i) \) which is generated by the crossover operation. If a particle with better fitness is obtained, that is, \( RC\_fit(i) \) is better than \( fit(i) \), set \( Indi(i) = RC\_Indi(i) \), and execute the next step after updating the current particle; otherwise, execute the next step directly;

Step 6: Optimal crossover. Cross the particle \( Indi(i) \) with the group extreme particle \( Indi(G\_best) \), and calculate the fitness value \( GC\_fit(i) \) of the offspring \( GC\_Indi(i) \) which is generated by crossover operation. If a particle with better fitness is obtained, that is, \( GC\_fit(i) \) is better than \( fit(i) \), set \( Indi(i) = GC\_Indi(i) \), and execute the next step after updating the current particle; otherwise, execute the next step directly;

Step 7: Mutation operation, the offspring generated by the mutation operation of particle \( Indi(i) \) is \( M\_Indi(i) \), if a particle with better fitness is obtained, that is, \( M\_fit(i) \) is better than \( fit(i) \), so that \( Indi(i) = M\_Indi(i) \), jump to Step 3 after updating the current particle; otherwise, jump directly to Step 3.

![Figure 6](image)

**Figure 6.** The process framework of the improved hybrid particle swarm optimization (HPSO) algorithm for process planning.

5.3. **Algorithm Details**

The key modules of the algorithm will be explained in this part.

5.3.1. **Particle Swarm Initialization**

Solutions that do not conform to the sequence constraints of the product’s manufacturing features are called infeasible solutions or illegal solutions. In the random initialization process of particle swarms and subsequent mutation operation, it is inevitable to generate infeasible solutions. There are many ways to deal with infeasible solutions, such as directly deleting infeasible solutions, using penalty functions, and transforming feasible solutions and so on. In this paper, a feasible solution transformation method proposed in the literature [5]—the constraint adjustment method is used to adjust the infeasible solution to make it a feasible solution.
In order to obtain the initial population quickly, the method of randomly initializing the particle swarm is adopted in this article. The specific steps are as follows:

Step 1: Generate a digital sequence from 1 to \( n \) as the original code of the operation layer of the particle (chromosome), where \( n \) is the total number of optional operations for the product;

Step 2: According to the process information table, select the available processing machines randomly for each operation to form the machine sequence as the original code of the machine layer;

Step 3: According to the optional process of the product’s manufacturing features, randomly select one operation combination that satisfies the product completion condition (or adopt the tree diagram traversal method to randomly generate the operation combination that satisfies completion condition of the product according to the AND/OR diagram), and generate 0–1 digital sequence as the original code of the logic layer;

Step 4: Combine the above three layers of codes to form a \( 3 \times n \) coding matrix;

Step 5: Generate a random sequence, and sort the above coding matrix according to the random sequence in units of columns;

Step 6: Use the constraint adjustment method in the literature [5] to adjust the operation sequence that does not satisfy the constraint conditions of the product’s manufacturing feature sequence to make it a feasible solution;

Step 7: If the size of the particle swarm is \( N \), repeat the above step \( N \) times to obtain the initial particle swarm.

The above steps can be described by the following pseudo code, \( \text{POP} \) represents population, \( \text{Indi} \) represents individual, \( N \) represents population size, \( \text{ON} \) represents number of operations, \( \{\text{OS}\} \) represents combination of optional operation, \( \{\text{JM}\} \) represents set of optional machines, \( \{\text{COO}\} \) represents constraints matrix set of the operations, \( \text{LO} \) represents the operation layer code, \( \text{LM} \) represents the machine layer code, \( \text{LL} \) represents the logic layer code, and \( M(j) \) represents the machine selected by operation \( j \).

---

**Generate Initial Population**

1. Input: \( N, \text{ON}, \{\text{OS}\}, \{\text{JM}\}, \{\text{COO}\} \)
2. Output: \( \text{Initial POP} \)
3. Set \( \text{Initial POP}=\{\phi\} \)
4. For \( n=1 \) to \( N \) do
5. \( \text{LO}=[\phi], \text{LM}=[\phi], \text{LL}=[\phi], M(j)=[\phi] \)
6. \( \text{LO}=\text{ones}(1 \text{ to } \text{ON}) \)
7. For \( j=1 \) to \( \text{ON} \) do
8. \( M(j)=\text{rand}(\text{JM}(j)) \)
9. End For
10. \( \text{LM}=[\text{LM}, M(j)] \)
11. \( \text{Index}_L=\text{rand}(\text{OS}(i)) \)
12. \( \text{LL}=[\text{zeros}(1, \text{ON}) \]
13. \( \text{LL} (\text{Index}_L)=1 \)
14. \( \text{Temp}_\text{Indi}(n)=\{\text{LO}; \text{LM}; \text{LO}\} \)
15. \( \text{Index}_\text{Total}(n)=\text{randperm}(\text{TN}) \)
16. \( \text{Indi}(n)=\text{Temp}_\text{Indi}(n)(\text{Index}_\text{Total}(n)) \)
17. if \( \text{Indi}(n)\rightarrow\{\text{COO}\}\rightarrow\text{illegal solution} \)
18. \( \text{run Constraint Adjustment function} \)
19. \( \text{Indi}(n)\rightarrow\{\text{COO}\}\rightarrow\text{feasible solution} \)
20. End if
21. \( \text{Initial POP}=\{\text{Initial POP}; \text{Indi}(n)\} \)
22. End For
5.3.2. Crossover Operators

In the crossover operation, each column of the chromosome serves as the basic unit of position transformation. When the position of any gene in a column changes, all genes in the column undergo the same position change. In this paper, three separate crossover operators are designed respectively for the operation layer, machine layer and logic layer, and iteratively searched from three different dimensions. The specific operations are as follows.

(1) Operation layer crossover. The crossover for the operation layer adopts a method similar to OX (order crossover) that can avoid generating infeasible offspring individuals. Figure 7 shows an example of crossover for the operation layer.

The steps of crossover for operation layer:
Step 1: Select two parent individuals, extract the operation layer, and record it as \( P_1 \) and \( P_2 \);
Step 2: Mark the gene whose logical layer code is 1 in \( P_1 \) or \( P_2 \) (gray in the figure);
Step 3: Select two intersection points randomly, record them as \( pos_1 \) and \( pos_2 \) from left to right;
Step 4: Copy the genes on the left side of \( pos_1 \) and the right side of \( pos_2 \) in \( P_1 \) to the offspring individual \( O_1 \);
Step 5: Mark the same genes in \( P_2 \) which are copied from \( P_1 \) to \( O_1 \);
Step 6: Insert the genes in \( P_2 \) that are different from those in \( O_1 \) into the vacancy of \( O_1 \) in order;
Step 7: According to the mark in Step 2, adjust the logic layer of \( O_1 \), make the selection of operations for \( O_1 \) the same as \( P_1 \) or \( P_2 \), to get the offspring \( O_1 \) after crossover.

(2) Machine layer crossover. The crossover for the machine layer adopt an operator similar to SEX (Subtour Exchange Crossover). Figure 8 shows an example of crossover for machine layer.

The steps of crossover for machine layer:
Step 1: Select two parent individuals, extract the operation layer and the machine layer, and record them as \( P_1 \) and \( P_2 \);
Step 2: Copy \( P_1 \) as \( O_1 \);
Step 3: Select several operations in \( P_1 \) randomly to conduct the crossover for machine layer (gray in the figure), and mark the same genes in \( P_2 \) which are selected previously in \( P_1 \);
Step 4: Replace the machine layer code of the selected operations in \( O_1 \) with the machine layer code of the same operation in \( P_2 \), to get the offspring \( O_1 \).
(3) Logic layer crossover. The logical layer crossover operator is similar to the machine layer crossover, but the operations of the product need to be segmented before crossover. According to the product process information table, find the manufacturing features with flexible processes, and segment the operations belonging to these manufacturing features according to different ones. An example of crossover for logical layer is shown in Figure 9, the operations of the same color belong to the same manufacturing feature in the figure.

The steps of crossover for logic layer:

Step 1: Select two parent individuals, extract the operation layer and logic layer, and record them as P1 and P2;
Step 2: Copy P1 as O1;
Step 3: Select one or more flexible operation segments of manufacturing features to be crossed. In the example, the first operation segment (7, 8, 9) belonging to a feature and the third operation segment (13, 14) belonging to another feature are selected;
Step 4: Extract operation genes which are selected for crossover in Step3 from P2, and reorder them to make the sequence of these operations consistent with the sequence in P1;
Step 5: Use the logic layer code after reorder which are extracted from P2 in Step 4 to replace the corresponding logic layer code in O1, to obtain the offspring O1 after crossover.

5.3.3. Mutation Operators

Generally, the mutation operation determines the local search performance of the genetic algorithm, and it cooperates with the crossover operation with global search capability to jointly complete the optimization process of the optimization problem. The same is true in the hybrid PSO algorithm of this article. In the mutation operation, like the crossover operation, when the position of
any gene in a column changes, all genes in the column undergo the same position change. In the following mutation operations for the three-layer codes, only the operation layer mutation will change the positions of genes. In addition, the operation layer variation may produce infeasible solutions, which need to be converted into feasible solutions by using the constraint adjustment method. The three mutation operators designed for different dimensions are as follows.

(1) Operation layer mutation. An exchange mutation (EM) method is adopted to exchange the positions of the operations. An example of mutation for operation layer is shown in Figure 10.

The steps of mutation for operation layer:
Step 1: Select one parent individual, extract the operation layer, and record it as \( P \);
Step 2: Copy \( P \) as \( O \);
Step 3: Within the length range of \( O \), select two mutation positions \( pos1 \) and \( pos2 \) randomly;
Step 4: Exchange the positions of genes in \( pos1 \) and \( pos2 \) of \( O \);
Step 5: Check that whether the operation sequence after mutation satisfies the constraint requirements of manufacturing feature. If the constraint conditions are met, the offspring \( O \) after the mutation is obtained; otherwise, convert it into a feasible solution by using the constraint adjustment method to obtain the offspring \( O \) after the mutation.

(2) Machine layer mutation. A single-point mutation (SPM) method is adopted to randomly select a position in the machine layer code for mutation. An example of mutation for machine layer is shown in Figure 11.

The steps of mutation for machine layer:
Step 1: Select one parent individual, extract the machine layer, and record it as \( P \);
Step 2: Copy \( P \) as \( O \);
Step 3: Within the length range of \( O \), randomly select a mutation position \( pos \) whose logic layer code is not 0, and record the machine number as \( k \) before mutation;
Step 4: Read the optional machine set \( M(j) \) of operation \( j \) which is on the position selected in Step 3;
Step 5: If there are multiple candidate machines in \( M(j) \), select other machine randomly in \( M(j) \) whose serial number are not equal to \( k \) to replace the one in position \( pos \) of offspring \( O \), and obtain the offspring \( O \) after mutation; Otherwise, go to Step 3 to reselect the position for mutation.

(3) Logic layer mutation. This mutation operator is similar as the crossover of logic layer, the operations of the product need to be segmented according to different manufacturing features before operation. Using a method similar to machine layer mutation, a feature with flexible process is randomly selected for mutation. An example of mutation for logic layer is shown in Figure 12.

The steps of mutation for logic layer:
Step 1: Select one parent individual, extract the logic layer, and record it as \( P \);
Step 2: Copy \( P \) as \( O \);
Step 3: Within the length range of \( O \), select a mutation position \( pos \) whose logic layer code is not 0, and record the machine number as \( k \) before mutation;
Step 4: Read the optional machine set \( M(j) \) of operation \( j \) which is on the position selected in Step 3;
Step 5: If there are multiple candidate machines in \( M(j) \), select other machine randomly in \( M(j) \) whose serial number are not equal to \( k \) to replace the one in position \( pos \) of offspring \( O \), and obtain the offspring \( O \) after mutation; Otherwise, go to Step 3 to reselect the position for mutation.
The steps of mutation for logic layer:

Step 1: Select a parent individual, extract the operation layer and logic layer, and record it as $P$;
Step 2: Copy $P$ as $O$;
Step 3: Randomly select an operation segment corresponding to one manufacturing feature, which has the flexible process, for logic layer mutation. In the example, the operation segment $(11,12)$ belonging a feature is selected, and read the corresponding logic code $(0,1)$;
Step 4: Randomly select other optional process of the manufacturing feature to generate a new logic code, in the example $(1,0)$;
Step 5: Use the new logic code generated by Step 4 to replace the original logic code in $O$ to obtain the offspring $O$ after mutation.

6. Case Study

In order to prove the feasibility of the algorithm and test the performance of the algorithm, we conduct two calculation experiments in this section. The HPSO algorithm for process planning is programmed with an engineering computing software and runs on a notebook computer with an intel i-7 (Core 7700 HQ 2.8 GHz) CPU and 16 G RAM. After many tests, the parameters of the algorithm are set as follows: the maximum value of iterations is 200; the population size is 200; the three-layer probability of crossover are 0.8, 0.8, and 0.6; the three-layer probability of mutation are 0.1, 0.8, and 0.1, respectively. For each calculation example, the algorithm is run 20 times, and the optimal value and average value of the results are recorded.

6.1. Case 1

The data of case 1 comes from the literature [2]. The product contains 11 manufacturing features, 17 optional operations and 15 optional processing machines. The AND/OR network diagram of the product is shown in Figure 13. The process information of the product is shown in Table 3. It is set in this paper that it will take a certain of time to change the processing machine of a product in the manufacture process. In order to facilitate the comparison of results, the data of switch time between different machines of a product which is in literature [2] are also adopted here. Table 4 shows the transportation time required for the product between different machines.

Figure 12. An example of mutation for logic layer.

Figure 13. The AND/OR network of the product in case 1.
Table 3. The process information of the product in case 1.

| Feature | Optional Operation | Optional Machine | Process Time | Sequence Constraints |
|---------|--------------------|------------------|--------------|---------------------|
| F1      | O1                 | M5, M6           | 8, 13        | Before F2, F3       |
|         | O2→O3              | M5, M6, M7→M8    | 16, 12, 13→21| Before F3           |
| F2      | O1→O5              | M1, M5, M10→M9   | 13, 16, 18→17|                     |
| F3      | O6                 | M5, M8           | 46, 47       |                     |
| F4      | O7                 | M2, M7, M13      | 44, 48, 49   | Before F5, F6, F7   |
| F5      | O8                 | M5, M6, M13      | 17, 14, 10   | Before F9, F7       |
| F6      | O9                 | M5, M15          | 16, 13       |                     |
| F7      | O10                | M3, M11, M15     | 28, 27, 30   | Before F7           |
| F8      | O11                | M10, M13         | 48, 50       |                     |
| F9      | O12                | M5, M13, M15     | 31, 32, 36   | Before F9, F10, F11|
|         | O13                | M3, M6, M9       | 30, 28, 26   | Before F10, F11     |
|         | O14→O15            | M7→M1, M14       | 11→16, 18    |                     |
| F10     | O16                | M4, M16          | 18, 19       | Before F11          |
| F11     | O17                | M3, M10, M14     | 36, 32       |                     |

Table 4. The transportation time of a product between different machines.

| Machine | M1 | M2 | M3 | M4 | M5 | M6 | M7 | M8 | M9 | M10 | M11 | M12 | M13 | M14 | M15 |
|---------|----|----|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|
| M1      | 0  | 5  | 7  | 9  | 10 | 11 | 7  | 6  | 14 | 13  | 12  | 10  | 5   | 6   | 9   |
| M2      | 5  | 0  | 3  | 4  | 5  | 7  | 2  | 7  | 6  | 4   | 7   | 12  | 10  | 7   | 8   |
| M3      | 7  | 3  | 0  | 6  | 5  | 4  | 3  | 7  | 2  | 4   | 3   | 5   | 6   | 8   | 9   |
| M4      | 9  | 4  | 6  | 0  | 4  | 4  | 6  | 7  | 4  | 10  | 12  | 13  | 14  | 15  | 16  |
| M5      | 10 | 5  | 5  | 4  | 0  | 10 | 12 | 7  | 8  | 9   | 10  | 12  | 11  | 10  | 8   |
| M6      | 11 | 7  | 4  | 4  | 10 | 0  | 4  | 4  | 5  | 5   | 6   | 7   | 6   | 7   | 8   |
| M7      | 7  | 2  | 3  | 6  | 12 | 4  | 0  | 5  | 6  | 6   | 6   | 7   | 7   | 7   | 8   |
| M8      | 6  | 7  | 7  | 7  | 7  | 4  | 5  | 0  | 4  | 2   | 3   | 4   | 2   | 4   | 3   |
| M9      | 14 | 6  | 2  | 4  | 8  | 5  | 6  | 4  | 0  | 5   | 7   | 4   | 7   | 6   | 8   |
| M10     | 13 | 4  | 4  | 10 | 9  | 5  | 6  | 2  | 5  | 0   | 8   | 10  | 12  | 14  | 7   |
| M11     | 12 | 7  | 3  | 12 | 10 | 6  | 6  | 3  | 7  | 8   | 0   | 7   | 10  | 14  | 10  |
| M12     | 10 | 12 | 5  | 13 | 12 | 7  | 7  | 4  | 4  | 10  | 7   | 0   | 10  | 12  | 10  |
| M13     | 5  | 10 | 6  | 14 | 11 | 6  | 7  | 2  | 7  | 12  | 10  | 10  | 0   | 8   | 8   |
| M14     | 6  | 7  | 8  | 15 | 10 | 7  | 7  | 4  | 6  | 14  | 14  | 12  | 8   | 0   | 9   |
| M15     | 9  | 8  | 9  | 16 | 8  | 8  | 8  | 3  | 8  | 7   | 10  | 10  | 8   | 9   | 0   |

Table 5 shows the comparison between the calculation results of the algorithm in this paper and the calculation results of several algorithms in literature [2].

Table 5. The result comparison of different algorithms in case 1.

| Algorithm     | Simple SA | Simple GA | Modified PSO | Hybrid PSO | Improve Rate |
|---------------|-----------|-----------|--------------|------------|--------------|
| Optimal result| 377       | 377       | 377          | 356        | 5.6%         |
| Average result| 378.1     | 380.2     | 377          | 358.5      | 4.9%         |
The comparison results show that the optimal and average results of HPSO algorithm are better than the other three algorithms, which fully proves that the algorithm proposed in this paper has better search performance. A large part of the improvement of algorithm performance is due to the new coding method adopted in the algorithm of this article. The optimal process route plan of case 1 is as follows:

\[(O_7, M_3) \rightarrow (O_4, M_3) \rightarrow (O_4, M_4) \rightarrow (O_{12}, M_{13}) \rightarrow (O_8, M_3) \rightarrow (O_{13}, M_9) \rightarrow (O_5, M_9) \rightarrow (O_6, M_2) \rightarrow (O_6, M_8) \rightarrow (O_{17}, M_{10}) \rightarrow (O_{11}, M_{10})\]

6.2. Case 2

The data in the case 2 comes from the literature [23], including 18 independent products with different degrees of flexibility, each of which contains different features and operations, and there are 15 optional machines. The details of the optional machines, processing time and technological process (sequence constraints) of each operation of the products can be found in the appendix of the literature [24]. In order to facilitate comparison, the data of the transportation time of a product between the machines in case 1 (Table 4) is adopted in this case. In the literature [2], simple SA, simple GA and modified PSO were used by Li et al. to carry out the process planning of these 18 products, and the calculation result comparison is provided. The optimal results and average results comparison of the above three algorithms with the H-PSO proposed in this paper are shown in the Tables 6 and 7.

| Product Number | Optimal Result | Improved Rate | Computing Time (s) |
|----------------|----------------|---------------|-------------------|
| 1              | S-SA | S-GA | M-PSO | H-PSO | 3.63% | 11.57 |
| 2              | 359  | 359  | 359  | 351   | 2.23% | 17.66 |
| 3              | 502  | 502  | 498  | 489   | 1.81% | 19.76 |
| 4              | 314  | 314  | 314  | 349   | -11.15% | 17.74 |
| 5              | 314  | 409  | 408  | 391   | 4.17% | 19.14 |
| 6              | 304  | 304  | 304  | 304   | 0.00% | 17.72 |
| 7              | 358  | 358  | 358  | 353   | 1.40% | 18.69 |
| 8              | 393  | 392  | 391  | 390   | 0.26% | 19.23 |
| 9              | 264  | 264  | 264  | 264   | 0.00% | 14.64 |
| 10             | 271  | 271  | 271  | 266   | 1.85% | 12.75 |
| 11             | 442  | 442  | 442  | 432   | 2.26% | 19.15 |
| 12             | 216  | 216  | 216  | 215   | 0.46% | 18.19 |
| 13             | 269  | 269  | 269  | 244   | 9.29% | 16.12 |
| 14             | 358  | 357  | 357  | 354   | 0.84% | 16.83 |
| 15             | 248  | 248  | 248  | 244   | 1.61% | 17.98 |
| 16             | 314  | 314  | 314  | 300   | 4.46% | 18.62 |
| 17             | 361  | 361  | 360  | 356   | 1.11% | 18.44 |

Table 7. The average result comparison of different algorithms in case 2.

| Product Number | Average Result | Improved Rate | Computing Time (s) |
|----------------|----------------|---------------|-------------------|
| 1              | S-SA | S-GA | M-PSO | H-PSO | 3.63% | 10.87 |
| 2              | 361.2 | 360.7 | 359 | 352.4 | 1.84% | 18.06 |
| 3              | 504.9 | 503.6 | 500.3 | 492.2 | 1.62% | 17.58 |
| 4              | 315.7 | 314.8 | 314 | 349.4 | -11.27% | 18.69 |
| 5              | 315.8 | 315.1 | 314 | 282 | 10.19% | 20.28 |
| 6              | 411.3 | 410.3 | 408.8 | 392.3 | 4.04% | 18.32 |
The comparison results show that in the No. 1–18 problems of the process planning for products, the results of optimal value and average value calculated by the H-PSO algorithm in this paper are better than the other three algorithms in 15 problems, and equal in 2 problems. Table 8 provides the optimal process route scheme for each product calculated by H-PSO.

| Product Number | Optimal Process Route Scheme |
|----------------|-----------------------------|
| 1              | \((O_1, M_{10}) \rightarrow (O_1, M_{13}) \rightarrow (O_3, M_{15}) \rightarrow (O_5, M_{13}) \rightarrow (O_6, M_{12}) \rightarrow (O_7, M_{11})\) |
|                | \((O_8, M_9)\) |
| 2              | \((O_1, M_{10}) \rightarrow (O_4, M_{12}) \rightarrow (O_2, M_{14}) \rightarrow (O_6, M_{12}) \rightarrow (O_{12}, M_{14}) \rightarrow (O_{13}, M_{12}) \) |
|                | \((O_{15}, M_{12})\) |
| 3              | \((O_1, M_{10}) \rightarrow (O_4, M_{12}) \rightarrow (O_2, M_{14}) \rightarrow (O_6, M_{12}) \rightarrow (O_{12}, M_{14}) \rightarrow (O_{13}, M_{12}) \) |
|                | \((O_{15}, M_{12})\) |
| 4              | \((O_1, M_{10}) \rightarrow (O_4, M_{12}) \rightarrow (O_2, M_{14}) \rightarrow (O_6, M_{12}) \rightarrow (O_{12}, M_{14}) \rightarrow (O_{13}, M_{12}) \) |
|                | \((O_{15}, M_{12})\) |
| 5              | \((O_1, M_{10}) \rightarrow (O_{14}, M_{12}) \rightarrow (O_{15}, M_{13}) \rightarrow (O_{16}, M_{15}) \rightarrow (O_{17}, M_{16}) \rightarrow (O_{18}, M_{18}) \) |
|                | \((O_{19}, M_{19})\) |
| 6              | \((O_1, M_{10}) \rightarrow (O_{14}, M_{12}) \rightarrow (O_{15}, M_{13}) \rightarrow (O_{16}, M_{15}) \rightarrow (O_{17}, M_{16}) \rightarrow (O_{18}, M_{18}) \) |
|                | \((O_{19}, M_{19})\) |
| 7              | \((O_1, M_{10}) \rightarrow (O_{14}, M_{12}) \rightarrow (O_{15}, M_{13}) \rightarrow (O_{16}, M_{15}) \rightarrow (O_{17}, M_{16}) \rightarrow (O_{18}, M_{18}) \) |
|                | \((O_{19}, M_{19})\) |
| 8              | \((O_1, M_{10}) \rightarrow (O_{14}, M_{12}) \rightarrow (O_{15}, M_{13}) \rightarrow (O_{16}, M_{15}) \rightarrow (O_{17}, M_{16}) \rightarrow (O_{18}, M_{18}) \) |
|                | \((O_{19}, M_{19})\) |
| 9              | \((O_1, M_{10}) \rightarrow (O_{14}, M_{12}) \rightarrow (O_{15}, M_{13}) \rightarrow (O_{16}, M_{15}) \rightarrow (O_{17}, M_{16}) \rightarrow (O_{18}, M_{18}) \) |
|                | \((O_{19}, M_{19})\) |
| 10             | \((O_1, M_{10}) \rightarrow (O_{14}, M_{12}) \rightarrow (O_{15}, M_{13}) \rightarrow (O_{16}, M_{15}) \rightarrow (O_{17}, M_{16}) \rightarrow (O_{18}, M_{18}) \) |
|                | \((O_{19}, M_{19})\) |
(O_1, M_{12}) \rightarrow (O_{12}, M_{12}) \rightarrow (O_{13}, M_{12}) \rightarrow (O_{14}, M_{12}) \rightarrow (O_{15}, M_{12}) \\
(0, M_{10}) \rightarrow (O_{16}, M_{10}) \rightarrow (O_{17}, M_{10}) \rightarrow (O_{18}, M_{10}) \rightarrow (O_{19}, M_{10}) \rightarrow (O_{20}, M_{10}) \\
(0, M_{9}) \rightarrow (O_{21}, M_{9}) \rightarrow (O_{22}, M_{9}) \rightarrow (O_{23}, M_{9}) \rightarrow (O_{24}, M_{9}) \rightarrow (O_{25}, M_{9}) \\

7. Conclusions

Based on the calculation results of the above two calculation result of cases, the HPSO algorithm designed in this paper is an effective method to solve the process planning problem. It has strong search performance and can obtain a better process route plan for the minimum completion time objective function. The reasons are as follows: First, a coding method which adopted the direct coding based on product operations is more in line with the characteristics of the problem for process planning was designed in this algorithm. Compared with the traditional integrated coding method and hierarchical coding method, it contains more comprehensive process information, reduces the difficulty of decoding in, thereby improving the efficiency and search performance of the algorithm. Second, the multiple genetic operation operators designed in this paper are more compatible with the decision-making dimensions of process planning. The three crossover operators and mutation operators cooperate with each other. Global and local searches are respectively performed in each dimension from the selection of the processing technology, the operation sequence, and the processing machine for the product, which further strengthens the optimization performance of the algorithm. Therefore, the HPSO algorithm designed to solve the process planning problem in this paper is a more effective method.

This article mainly studied the mathematical model and solution method of process planning. First, the related literatures of the solve method for process planning problem were reviewed. Second, the representation method of flexible process route was elaborated. Third, the problem assumptions and related symbols of the mathematical model were explained, and the mathematical model of mixed integer programming for process planning with an object function of minimum completion time was established. Fourth, the HPSO algorithm is improved according to the main characteristics of process planning, an improved hierarchical coding method including operation layer, machine layer and logic layer was proposed, and several genetic operators were designed for improving the optimization performance of the algorithm. Finally, the effectiveness of the algorithm is proved by two experiments in the case calculation, and at the same time, it is proved that the algorithm in this paper has better search performance by comparing with other algorithms. The algorithm for process
planning in this paper can be embedded into the software of CAPP system, so that it can be applied in manufacturing enterprises.

With regard to directions for future research, we will try to propose more novel methods to solve process planning problems, and consider more dynamic factors of the actual production environment when performing the process planning. In addition, the integration of process planning with other systems is also a direction of the future research. Among them, the integrated optimization of process planning and scheduling is a kind of research with nice practical value.

Author Contributions: Conceptualization, X.Z. and J.Y.; methodology, X.Z. and J.Y.; software, X.Z.; validation, P.G. and H.Z.; formal analysis, X.Z.; resources, J.Y.; data curation, X.Z., P.G. and H.Z.; writing—original draft preparation, X.Z.; writing—review and editing, X.Z. and J.Y.; visualization, X.Z.; supervision, J.Y.; funding acquisition, J.Y.; project administration, J.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by China University Research Fund: 2010scu2101.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jin, L.; Zhang, C. Process Planning Optimization with Energy Consumption Reduction from a Novel Perspective: Mathematical Modeling and a Dynamic Programming-like Heuristic Algorithm. IEEE Access. 2019, 7, 7381–7396.
2. Li, X.; Gao, L.; Wen, X. Application of an Efficient Modified Particle Swarm Optimization Algorithm for Process Planning. Int. J. Adv. Manuf. Technol. 2013, 67, 1355–1369.
3. Yusof, Y.; Latif, K. Survey on Computer-aided Process Planning. Int. J. Adv. Manuf. Technol. 2014, 75, 77–89.
4. Al-Wswasi, M.; Ivanov, A.; Makatsoris, H. A survey on Smart Automated Computer-aided Process Planning (ACAPP) Techniques. Int. J. Adv. Manuf. Technol. 2018, 97, 809–832.
5. Li, W.D.; Ong, S.K.; Nee, A.Y.C. Hybrid Genetic Algorithm and Simulated Annealing Approach for the Optimization of Process Plans for Prismatic Parts. Int. J. Prod. Res. 2002, 40, 1899–1922.
6. Singh, D.K.J.; Jebraj, C. Feature-based Design for Process Planning of Machining Processes with Optimization Using Genetic Algorithms. Int. J. Prod. Res. 2005, 43, 3855–3887.
7. Su, Y.; Chu, X.; Chen, D.; Sun, X. A genetic algorithm for operation sequencing in CAPP using edge selection based encoding strategy. J. Intell. Manuf. 2015, 29, 313–332.
8. Delolme, L.; Antomarchi, A.-L.; Durieux, S.; Duc, E. Decision-making for multi-criteria optimization of process planning. Mech. Ind. 2019, 20, 806.
9. Guo, Y.W.; Mileham, A.R.; Owen, G.W.; Li, W.D. Operation sequencing optimization using a particle swarm optimization approach. Proc. Inst. Mech. Eng. Part B J. Eng. Manuf. 2006, 220, 1945–1958.
10. Wang, J.; Kang, W.; Zhao, J.; Chu, K. A simulation approach to the process planning problem using a modified particle swarm optimization. Adv. Prod. Eng. Manag. 2016, 11, 77–92.
11. Miljković, Z.; Petrović, M. Application of Modified Multi-Objective Particle Swarm Optimisation Algorithm for Flexible Process Planning Problem. Int. J. Comput. Integr. Manuf. 2016, 30, 271–291.
12. Wu, W.; Zeng, J.; Huang, Z. A Hybrid Optimization Approach for Setup Planning with Tolerance Constraints. MATEC Web Conf. 2018, 249, 1–5.
13. Ma, G.H.; Zhang, Y.F.; Nee, A.Y.C. A Simulated Annealing-Based Optimization Algorithm for Process Planning. Int. J. Prod. Res. 2000, 38, 2671–2687.
14. Li, W.D.; Ong, S.K.; Nee, A.Y.C. Optimization of Process Plans Using a Constraint-Based Tabu Search Approach. Int. J. Prod. Res. 2004, 42, 1955–1985.
15. Lian, K.; Zhang, C.; Shao, X.; Zeng, Y. A multi-dimensional tabu search algorithm for the optimization of process planning. Sci. China Ser. E Technol. Sci. 2011, 54, 3211–3219.
16. Liu, X.J.; Yi, H.; Ni Z.H. Application of Ant Colony Optimization Algorithm in Process Planning Optimization. J. Intell. Manuf. 2013, 24, 1–13.
17. Hu, Q.; Qiao, L.; Peng, G. An ant colony approach to operation sequencing optimization in process planning. Proc. Inst. Mech. Eng. Part B J. Eng. Manuf. 2016, 231, 470–489.
18. Chan, F.T.S.; Swarnkar, R.; Tiwari, M.K. Fuzzy Goal-Programming Model with an Artificial Immune System (AIS) Approach for a Machine Tool Selection and Operation Allocation Problem in a Flexible Manufacturing System. Int. J. Prod. Res. 2005, 43, 4147–4163.
19. Gao, B.; Hu, X.; Peng, Z.; Song, Y. Application of intelligent water drop algorithm in process planning optimization. *Int. J. Adv. Manuf. Technol.* **2020**, *106*, 5199–5211.

20. Li, X.Y.; Shao, X.Y.; Gao, L. Optimization of Flexible Process Planning by Genetic Programming. *Int. J. Adv. Manuf. Technol.* **2008**, *38*, 143–153.

21. Marinakis, Y.; Migdalas, A.; Sifaleras, A. A hybrid particle swarm optimization--variable neighborhood search algorithm for constrained shortest path problems. *Eur. J. Oper. Res.* **2017**, *261*, 819–834.

22. Cagnina, L.C.; Esquivel, S.C.; Gallard, R. Particle swarm optimization for sequencing problems: A Case Study. In Proceedings of the Congress on Evolutionary Computation, Portland, OR, USA, 19–23 June 2004; pp. 536–541.

23. Kim, Y.K.; Park, K.; Ko, J. A Symbiotic Evolutionary Algorithm for the Integration of Process Planning and Job Shop Scheduling. *Comput. Oper. Res.* **2003**, *30*, 1151–1171.

24. Li, X. Research on the Solution Methods of Integrated Process Planning and Scheduling. Ph.D. Thesis, Huazhong University of Science and Technology, Wuhan, China, 2009.

© 2020 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/).