Abstract: In the process of the integrated scheduling of multi-variety and small-batch complex products, the process structure and attribute characteristics are often ignored, which affects the overall scheduling effect. Aiming at solving this problem, a heuristic integrated scheduling algorithm (HIS-IDA) based on the improved Dijkstra algorithm is proposed. The algorithm takes the processing time of the process itself as the path value of the preceding and the following adjacent processes. Firstly, the improved Dijkstra algorithm prioritized the scheduling of the process sequence with long longitudinal paths and realized the “longitudinal optimization” of the integrated scheduling. Secondly, the layer priority strategy is used to shorten the interval time of process processing and realize the “horizontal optimization” of integrated scheduling. On the basis of “vertical and horizontal optimization”, the idle time of the equipment is further reduced by using the process priority strategy of the leaf node, and the “idle optimization” of the integrated scheduling is realized, so as to optimize the overall effect of the integrated scheduling. The effectiveness and superiority of the algorithm are proved using comparison analysis.

Keywords: Dijkstra algorithm; path value; heuristic; integrated scheduling

1. Introduction

The integrated scheduling problem is to achieve optimal time efficiency and equipment utilization by scheduling limited equipment, human resources, and other resources [1, 2]. That is, the overall processing time is short, and the equipment utilization rate is high. And it has always been a pertinent topic for experts and scholars since it has a direct impact on the production efficiency and social benefit of enterprises.

With the development of computer technology and the change in social demand, product scheduling in the manufacturing industry is transformed from single machine scheduling to distributed scheduling [3, 4]. In the era of big data and artificial intelligence, product scheduling methods based on cloud computing and fog computing are also developing rapidly [5, 6]. Complex products [7], that is, the use of high-tech, the use of complex processes, the number of parts, the level of product structure, the relationship between parts and products, the value of high-value products, and their demand are becoming more and more personalized and diversified. So, the problem of product scheduling in multi-variety and small-batch complex processes needs to be solved urgently [8]. Multi-variety and small-batch complex products refer to a production method in which a greater number of product types (e.g., specifications, models, sizes, shapes, colors, etc.) are used as the object of production, and a smaller number of each type of product is produced in a specified production period. Generally speaking, this mode of production and mass production mode, compared with its low efficiency, high cost, not being easy to achieve automation, production planning, and organization, is more complex. In order to better study the
scheduling problem of complex products, Xie et al. proposed integrated scheduling, which considers the processing and assembly of the products simultaneously. And a series of scheduling optimization algorithms are given for this.

For example, the job-shop scheduling algorithm based on the ACPM (allied critical path method) and BFSM (best fit scheduling method) [9] takes into account the process compactness of key equipment. The dynamic job-shop scheduling algorithm with a dynamic set of operation having priority [10] proposes the strategy combined by layer first, short time, long path, and dynamic adjustment. The dynamic critical path multi-product manufacturing scheduling algorithm based on the operation set [11] realizes the two-way scheduling optimization, with vertical being the main consideration. The integrated scheduling algorithm with a no-wait constraint operation group [12] puts forward the strategy of “first grouping, then determining the scheduling order according to the number of working procedures in the group”, which has the constraint condition of connection. The time-selective integrated scheduling algorithm considering the compactness of serial processes [13] proposes a time-selective scheduling strategy based on process sequence sequencing.

These algorithms achieve two-way optimization by optimizing the path length of the longitudinal machining process, increasing the tightness of the serial process and increasing the parallel processing strength of the process. However, the problem of ignoring one another in horizontal and vertical optimization still exists. Vertical optimization refers to the optimization of the serial tightness of the process, while horizontal optimization refers to the optimization of the parallel processing of the device [14,15], such as the ACPM- and BFSM-based dynamic job-shop algorithm optimizing the longitudinal processing process [9] while ignoring the parallel processing of equipment. And in the integrated scheduling algorithms for working group linkages [12], when the processing time of the upper leaf node of the short path on the same equipment is greater than the starting processing time of the non-leaf node on the long path, the first applicable scheduling will fail. The reason for this is that the tight degree of cohesion between processes is increased by neglecting the scheduling of leaf nodes. In the timing algorithm considering serial process tightness [13], only when a process group is scheduled does the subsequent process group begin to schedule since the processing gap is formed in the scheduling process sequence of the process group.

In order to improve the shortcomings of ignoring the processing time of the middle node process of the complex product process tree (process tree is the flow chart of the whole product process) and the processing time of process itself in the integrated scheduling of multi-variety and small-batch complex products in recent years [16,17], which affects the collaborative processing ability of the integrated scheduling equipment resources, a heuristic integrated scheduling algorithm based on the improved Dijkstra algorithm is proposed. The integrated scheduling problem of complex products with many layers of “long path” tree is studied in this paper. And based on the idea of longitudinal optimization, this paper increases the horizontal optimization, which not only improves the tightness of the serial process but also effectively avoids the long waiting of processing equipment and optimizes the overall effect of integrated scheduling.

2. Problem Definition

As a kind of NP problem [18], the integrated scheduling problem has been known as a pertinent topic in combinatorial optimization [19–23], and it has produced many related algorithms. The improved Dijkstra algorithm [24], integrated scheduling algorithms for working group linkages, and the time-selecting integrated scheduling algorithm considering serial process tightness will be discussed later. Graph theoretic algorithms [25], especially one of the classical algorithms: Dijkstra algorithm for finding the shortest path [26–28], have high efficiency of the optimal solution for solving the NP problem, and they are well applied in the specific application field. It provides new ideas and methods for solving the integrated scheduling problem.
The Dijkstra algorithm is a classical algorithm for finding the shortest path, which is proposed by Edsger Wybe Dijkstra [26,29], a Dutch computer scientist. The Dijkstra algorithm uses breadth-first traversal to calculate the shortest path from any node in a weighted directed graph to other nodes [26]. Therefore, it is reasonable to use the Dijkstra algorithm to obtain the approximate optimal solution of the original problem because the directed graph of a complex product process tree belongs to the category of the graph for this kind of NP problem. A heuristic algorithm is an algorithm based on intuitive or empirical construction, which gives a feasible solution to each instance of the combinatorial optimization problem to be solved at an acceptable cost (computational time and space) [30]. It is helpful to analyze the performance and application domain of the algorithm, and the unique advantages and disadvantages of each algorithm can be found by comparison, so as to improve the algorithm structure, parameters, and operators, and develop various possible efficient hybrid algorithms [31,32].

In integrated scheduling, according to the digraph of a complex product process tree with long longitudinal paths, the process self-processing time of each node is taken as the path assignment in the Dijkstra algorithm. Then, the improved Dijkstra algorithm is used to design an integrated scheduling algorithm based on layer priority and the leaf node priority scheduling strategy [33–35]. The structure characteristic and process attribute value of complex product process trees are taken as the key factors that affect the overall optimization result of integrated scheduling. Firstly, to reduce the overall time of complex products vertically, the process sequence is scheduled according to the sequence of process path values from large to small by using the improved Dijkstra algorithm. Secondly, the serial tightness of the machining process sequence is improved horizontally using the layer priority strategy. Finally, the leaf node process priority strategy [36] is used to make full use of the idle time on the processing equipment so as to improve the resource cooperative processing ability of the processing equipment and realize the optimization of the integrated scheduling of complex products.

The requirements of integrated scheduling are shown in a list.
(1) The processing process diagram of complex products is a tree-oriented structure diagram. The node represents the processing process, and the arrow direction indicates the relationship between the tight front and back constraints;
(2) Each process node has three attributes: process serial number, corresponding processing equipment serial number, and its own processing time;
(3) Processing and assembly are uniformly called processing;
(4) The same equipment can only be processed in one continuous process at a certain time;
(5) The same equipment does not exist and only one processing unit can be matched for each process.

Definition 1. Process layer priority

According to the principle of reducing the parallel process processing time, the priority of processing is put forward based on the processing process in a complex product process tree, that is, the priority of process scheduling is defined as the priority of process in the process tree of complex products. Assuming the complex product processing tree has n layers, then the priority of the root node process is defined as 1, and the priorities of all descendant node processes in the root node process are defined as 2, and so on, until the priorities of all nodes in layer n are defined as n. The priority of the root node process is the lowest, while the priority of the process on the nth layer is the highest.

Definition 2. Leaf Node Process Priority

No matter which layer of the process node is located in the complex process tree, as long as it is a node without a tight pre-constrained process, it is regarded as a leaf node process. Starting from the starting point of the processing equipment corresponding to
the leaf node process, find out whether there is an idle time period greater than or equal to the leaf node processing time, if so, then insert the leaf node and scheduled priority; otherwise, there is a seamless connection of leaf node processes on the corresponding processing equipment.

**Definition 3. Path value**

The processing time of each process in the complex product process tree is defined as the path value between the tight pre-process and the tight post-process, and the path value between the other relation processes is superimposed according to the shortest path algorithm.

**3. Algorithm Design**

3.1. Algorithm Idea

Because the attributes of the leaf node process and the processing time of the process in the directed diagram of the complex process tree are very important to the overall effect of the optimization on the integrated scheduling of complex products, the leaf node process priority strategy is proposed by adopting the improved Dijkstra algorithm on the basis of the longitudinal optimization idea. The serial tightness of the process is improved by the comprehensive application of a long-path process priority scheduling strategy. And the idle time period of the equipment is reduced, and the parallel processing ability of the processes is improved by the idea of the preferential scheduling of leaf node processes on the corresponding processing equipment. Therefore, the equipment resources co-processing capacity is improved so as to reduce the total processing time of integrated scheduling.

According to the structure of the digraph of complex product process tree, the path value of each node to the root node is calculated, and each process is scheduled according to the principle of the path value of the process obtained by the improved Dijkstra algorithm from large to small, layer priority, and leaf node process priority. The algorithm flow chart is shown in Figure 1, and the description is as follows.

**Step 1**, the complex product process tree is simplified into a digraph according to the relationship between processes;

**Step 2**, the processing time of the process is referred to as the path value of the adjacent process; then calculate the path value of each node process to the root node process;

**Step 3**, determine whether the path values of each process are the same. If they are the same, the processes with the same path values are transferred to Step 4. If not, according to the improved Dijkstra algorithm, these path values are sorted in order from large to small, and each node process on the complex product process tree is scheduled in this order;

**Step 4**, according to the principle of layer priority, each node process is scheduled in descending order according to the layer priority. The later the layer where the node process is located, the earlier it is scheduled. If the layer priorities of the processes are the same, then go to Step 5;

**Step 5**, the priority principle of leaf node: scheduling the leaf node process in accordance with the principle.

Because the complex product process tree structure is a directed graph, the process time of the process is denoted as the directed path value of each vertex in the Dijkstra algorithm. The improved Dijkstra algorithm model is constructed according to the constraint relation of each process. Then, the path values from the complex process tree root node to each process are calculated. Finally, the process sequence is outputted in reverse sequence according to the path value.

Assuming that the shortest path from the root node process \( p_1 \) to each process \( p_k \) in a complex process tree is \( L_{1k} = p_1 p_2 \ldots p_k \), the length is denoted as \( d(p_1 p_k) = d_{1k} \). If \( L_{1k} \) is the shortest path from the root node process \( p_1 \) to each process \( p_k \), then the subpaths \( p_1 p_2 \ldots p_i \) and \( p_i p_{i+1} \ldots p_k \) are the shortest path from process \( p_1 \) to process \( p_i \) and the shortest path from \( p_i \) to process \( p_k \), respectively. The algorithm steps are as follows:
Figure 1. Flow chart of the algorithm in this paper.

Step1: Algorithm initialization. The root node process is stored in the process sequence $A$ in which the path value has been determined. At this time, only the root node process is stored in it. Then, the processes with undetermined path values are stored in the sequence $\bar{A}$. The initial state sequence $\bar{A}$ includes all processes except the root node process.

Let $i = 1$, then

$$L_1(p_1) = 0, d(p_1, p_1) = 0;$$  

$$d_i \in \sum_{i=1}^{n} p_u;$$

$$L_1(p_j) = +\infty (j = 2, 3, \cdots, n);$$

Set $L_1(p) = \sum_{p_j \in P} L_1(p_j);$  

$$A = \{ p_1 \};$$

$$\bar{A} = V - A.$$  

Formula (1) represents the starting point which is the root node process. Formula (2) represents the sequence of adjacent paths values which can be obtained by the processing time of each process. Formula (3) represents each process node path that traverses the complex process tree from the root node process. Formula (4) represents the shortest path from the root node to the $j$th process. Formula (5) represents a sequence of processes already traversed. Formula (6) represents a sequence of processes not yet traversed.
Step2: In the process sequence $A$ with undetermined path values, calculate the path value from the root node process to the tight pre-process and put the process with the shortest path into sequence $A$, then delete it from sequence $\bar{A}$.

For $p_j \in A$, then

$$i = i + 1;$$

$$L_i(p_j) = \min\{L_{i-1}(p_j), L_{i-1}(p) + W(p, p_j)\};$$

Set $L_i(p) = \min_{p_j \in A} L_2(p_j);$  \hspace{1cm} (9)

$$A = A \cup \{p\};$$

$$\bar{A} = \bar{A} - \{p\};$$

$$d(p_1p) = L_i(p);$$  \hspace{1cm} (12)

Formula (8) represents the shortest path from the $i$th process to the $j$th process; Formula (9) represents the shortest path from the root node to the $i$th process; Formula (10) represents the node process on the shortest path added to the process sequence with defined path processes; Formula (11) represents the corresponding process deleted from process sequence $\bar{A}$ with undetermined path value processes; and Formula (12) represents the path values from the root node to the $i$th process.

Step3: In the digraph of a complex product process tree, if $d(p_1, p) = +\infty$, then the node path from $p_1$ to sequence $\bar{A}$ does not exist;

Step4: If $i = n$, then all the process nodes of the complex process tree are traversed and the algorithm is stopped; otherwise, go to Step2;

Step5: Reverse the output of each process $p$ of the above steps that are added to the sequence $A$, and then the algorithm is stopped.

The improved Dijkstra algorithm flow chart is shown in Figure 2.

Figure 2. Flow chart of the improved Dijkstra algorithm.
3.2. Algorithm Code and Complexity

3.2.1. Algorithm Pseudo-Code

In Algorithm 1, the algorithmic pseudo-code shows the main steps of the heuristic integrated scheduling algorithm based on the improved Dijkstra algorithm and expresses the algorithmic functions, algorithmic ideas, and logic of this algorithm.

**Algorithm 1 Algorithm Pseudo-code**

Integrated scheduling (A)
Input: Set Directed tree graph; /* create directed tree maps */
Input: Input S(A); /* input process sets A */
Input: SetAdjacent pathvalue (T)←Self processing time from Ai; /* take the processing time of Ai as the path value of adjacent processes */
Input: Set(P)←Compute Layerpriority to Ai; /* statistics the layer priority Lpi of processes Ai */
For each node Ai∈A do
Compute Pathvalue(Ai) /* statistics the path value Pvi from process Ai to root node process */
End for
If same Pathvalue(Ai) then /* whether the path values of process Ai are the same */
| If same Layerpriority (Ai) then /* whether the process priority of process Ai are the same */
| | If Leaf node (Ai) then /* whether process Ai is leaf node process */
| | | Scheduling (Ai) According to the process priority strategy of leaf node, /* schedule process Ai by leaf node process priority strategy */
| | Elseif Scheduling (Ai) According to the layer priority strategy /* schedule process Ai by layer priority strategy */
| End if
| End if
End if
Output: Improved Dijkstra (Ai) /* input process sequence by improved Dijkstra algorithm */

3.2.2. Complexity Analysis

Suppose that the total order of the product is \( n \), and the number of equipment is \( m \). The time complexity of the algorithm depends on the number of repeated executions of statements containing basic operations in the deepest loop. Therefore, according to the algorithm code in 4.2.1, the number of repeated executions of statements in the for loop is \( n \), and the time complexity of the algorithm that determines the set of processes and allocates the process time to the path value of the adjacent processes is \( \mathcal{O}(n) \). The process of calculating the layer priority of each process and judging whether the process is a leaf node process needs to be executed \( n \) times, so the time complexity of this process is \( \mathcal{O}(n) \). Similarly, if the improved Dijkstra algorithm is implemented with linked lists or arrays, then its time complexity is \( \mathcal{O}(n^2) \). Thus, the time complexity of the entire algorithm is \( \max\{\mathcal{O}(n), \mathcal{O}(n^2)\} = \mathcal{O}(n^2) \).

4. Example Analysis

In this section, examples are given to further explain the idea and advantages of this algorithm. Because the algorithm is not based on concrete examples and it has universal-
ity with multi-variety and small-batch processes, the algorithm is also effective for other examples. Assume that the complex product process tree is shown in Figure 3, and it has 11 layers and 27 processes. The process tree is the flow chart of the whole product process. In the process tree, each node represents an operation. And it contains three kinds of information: the operation number, the machine number, and the processing time. The arrow connects each operation and its immediate successor operation. It represents the precedence constraint between the operations, and each arrow points to the operation with lower precedence. For example, the node representative of A16/1/2 must process two working hours on machine 1 to complete the processing of A16. Its immediate preceding nodes are A17 and A19. The tight node is A15. Therefore, A16 can only be processed after A17 and A19 are completed. Similarly, A15 can only be processed after A16 processing.

Figure 3. The process tree of complex product.

Convert the complex product process tree shown in Figure 3 into a directed diagram with the process time as the path value. The directed diagram is shown in Figure 4.

Figure 4. Directed path graph of complex product process tree.
4.1. Process Path and Path Values

The processing time of the process is taken as the path value of the adjacent processes with precedence constrain. And the paths and path values from the root node process to each process in the complex product process tree in Figure 3 are calculated using the improved Dijkstra algorithm and are listed in Table 1.

<table>
<thead>
<tr>
<th>Process Node</th>
<th>Path from Root Node Process to Each Process</th>
<th>Path Value</th>
<th>Layer</th>
<th>Leaf Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>{A1, A2}</td>
<td>0</td>
<td>1</td>
<td>N</td>
</tr>
<tr>
<td>A2</td>
<td>{A1, A2, A3}</td>
<td>3</td>
<td>2</td>
<td>N</td>
</tr>
<tr>
<td>A3</td>
<td>{A1, A2, A3}</td>
<td>4</td>
<td>3</td>
<td>N</td>
</tr>
<tr>
<td>A4</td>
<td>{A1, A2, A3, A4}</td>
<td>7</td>
<td>4</td>
<td>N</td>
</tr>
<tr>
<td>A5</td>
<td>{A1, A2, A3, A4, A5}</td>
<td>11</td>
<td>5</td>
<td>N</td>
</tr>
<tr>
<td>A6</td>
<td>{A1, A2, A3, A4, A5, A6}</td>
<td>14</td>
<td>6</td>
<td>N</td>
</tr>
<tr>
<td>A7</td>
<td>{A1, A2, A3, A4, A5, A6, A7}</td>
<td>16</td>
<td>7</td>
<td>N</td>
</tr>
<tr>
<td>A8</td>
<td>{A1, A2, A3, A4, A5, A6, A7, A8}</td>
<td>17</td>
<td>8</td>
<td>Y</td>
</tr>
<tr>
<td>A9</td>
<td>{A1, A2, A9}</td>
<td>7</td>
<td>4</td>
<td>Y</td>
</tr>
<tr>
<td>A10</td>
<td>{A1, A2, A3, A4, A5, A6, A10}</td>
<td>15</td>
<td>7</td>
<td>Y</td>
</tr>
<tr>
<td>A11</td>
<td>{A1, A2, A11}</td>
<td>6</td>
<td>3</td>
<td>N</td>
</tr>
<tr>
<td>A12</td>
<td>{A1, A2, A11, A12}</td>
<td>9</td>
<td>4</td>
<td>N</td>
</tr>
<tr>
<td>A13</td>
<td>{A1, A2, A11, A12, A13}</td>
<td>10</td>
<td>5</td>
<td>N</td>
</tr>
<tr>
<td>A14</td>
<td>{A1, A2, A11, A12, A13, A14}</td>
<td>11</td>
<td>6</td>
<td>N</td>
</tr>
<tr>
<td>A15</td>
<td>{A1, A2, A11, A12, A13, A14, A15}</td>
<td>12</td>
<td>7</td>
<td>N</td>
</tr>
<tr>
<td>A16</td>
<td>{A1, A2, A11, A12, A13, A14, A15, A16}</td>
<td>14</td>
<td>8</td>
<td>N</td>
</tr>
<tr>
<td>A17</td>
<td>{A1, A2, A11, A12, A13, A14, A15, A16, A17}</td>
<td>15</td>
<td>9</td>
<td>Y</td>
</tr>
<tr>
<td>A18</td>
<td>{A1, A2, A11, A12, A13, A14, A15, A18}</td>
<td>15</td>
<td>8</td>
<td>Y</td>
</tr>
<tr>
<td>A19</td>
<td>{A1, A2, A11, A12, A13, A14, A15, A16, A19}</td>
<td>16</td>
<td>9</td>
<td>N</td>
</tr>
<tr>
<td>A20</td>
<td>{A1, A2, A11, A12, A13, A14, A15, A16, A17, A20}</td>
<td>18</td>
<td>10</td>
<td>N</td>
</tr>
<tr>
<td>A21</td>
<td>{A1, A2, A11, A12, A13, A14, A15, A16, A17, A20, A21}</td>
<td>22</td>
<td>11</td>
<td>Y</td>
</tr>
<tr>
<td>A22</td>
<td>{A1, A2, A11, A12, A13, A14, A15, A16, A17, A20, A22}</td>
<td>20</td>
<td>11</td>
<td>Y</td>
</tr>
<tr>
<td>A23</td>
<td>{A1, A2, A11, A12, A13, A14, A23}</td>
<td>12</td>
<td>6</td>
<td>N</td>
</tr>
<tr>
<td>A24</td>
<td>{A1, A2, A11, A12, A13, A23, A24}</td>
<td>14</td>
<td>7</td>
<td>N</td>
</tr>
<tr>
<td>A25</td>
<td>{A1, A2, A11, A12, A13, A23, A24, A25}</td>
<td>16</td>
<td>8</td>
<td>N</td>
</tr>
<tr>
<td>A26</td>
<td>{A1, A2, A11, A12, A13, A23, A24, A26}</td>
<td>15</td>
<td>8</td>
<td>Y</td>
</tr>
<tr>
<td>A27</td>
<td>{A1, A2, A11, A12, A13, A23, A24, A25, A27}</td>
<td>19</td>
<td>9</td>
<td>Y</td>
</tr>
</tbody>
</table>

4.2. Complex Product Scheduling Process

The shortest path principle means that among multiple paths with different path values, the path with the shortest path value is selected first, and then the longer path is selected. That is, the order of selection is the sorting of path values from shortest to longest. In the previous steps, the improved Dijkstra algorithm has been used to find out the path value of each node. Sort them in the reverse order of the shortest path principle: {A21, A22, A27, A20, A8, A7, A19, A25, A10, A17, A18, A26, A6, A16, A24, A15, A23, A5, A14, A13, A12, A4, A9, A11, A3, A2, A1}, where the path values of processes in {A19, A25, A7} are equal to 16. However, A19 is on the 9th layer, A25 is on the 8th layer, and A7 is on the 7th layer. Then, A19, A25, A7 are scheduled in order according to the layer priority strategy; the path values of processes in {A10, A17, A18, A26} are equal to 15, and they are all leaf node processes. A17 is on the 9th layer while A18 and A26 are on the 8th layer. A10 is on the 7th layer. Then, A17, A18, A26, A10 are scheduled in order according to the layer priority strategy. The path values of processes in {A6, A16, A24} are equal to 14. And A16 is on the 8th layer. A24 is on the 7th layer. A6 is on the 6th layer. Then, A16, A24, A6 are scheduled in order according to the layer priority strategy. The path values of processes in {A15, A23} are equal to 12. And A15 is on the 5th layer. A23 is on the 6th layer. Then, A23, A15 are scheduled in order according to the layer priority strategy. The path values of processes in {A5, A14} are equal to 11. And A5 is on the 5th layer. A14 is on the 6th layer.
Then, A14, A5 are scheduled in order according to the layer priority strategy. The path values of processes in \{A4, A9\} are equal to 7. And A4 and A9 are both on the 5th layer. A9 is the leaf node process. Then, A9, A4 are scheduled in order according to the layer priority strategy.

In summary, according to the algorithm in this paper, the processing sequence is obtained as \{A21, A22, A27, A20, A8, A19, A25, A7, A17, A18, A26, A10, A16, A24, A6, A23, A15, A14, A5, A13, A12, A9, A4, A11, A3, A2, A1\}. The total processing time is 25 man-hours. The scheduling state of each equipment, time, and process of complex product A is shown in Table 2, and the Gantt chart is shown in Figure 5.

### Table 2. Statistics of path and path value from root node operation to each node operation.

<table>
<thead>
<tr>
<th>Scheduling Time</th>
<th>Device</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
</tr>
</thead>
<tbody>
<tr>
<td>t = 0 to t = 1</td>
<td>Processing A21</td>
<td>Processing A22</td>
<td>Processing A27</td>
<td>Processing A17</td>
<td></td>
</tr>
<tr>
<td>t = 1 to t = 2</td>
<td>Processing A21</td>
<td>Processing A22</td>
<td>Processing A27</td>
<td>Processing A18</td>
<td></td>
</tr>
<tr>
<td>t = 2 to t = 3</td>
<td>Processing A21</td>
<td>Processing A26</td>
<td>Processing A27</td>
<td>Processing A18</td>
<td></td>
</tr>
<tr>
<td>t = 3 to t = 4</td>
<td>Processing A21</td>
<td>-----</td>
<td>Processing A8</td>
<td>Processing A18</td>
<td></td>
</tr>
<tr>
<td>t = 4 to t = 5</td>
<td>Processing A7</td>
<td>Processing A20</td>
<td>-----</td>
<td>Processing A10</td>
<td></td>
</tr>
<tr>
<td>t = 5 to t = 6</td>
<td>Processing A7</td>
<td>Processing A20</td>
<td>-----</td>
<td>-----</td>
<td></td>
</tr>
<tr>
<td>t = 6 to t = 8</td>
<td>-----</td>
<td>Processing A25</td>
<td>Processing A19</td>
<td>-----</td>
<td></td>
</tr>
<tr>
<td>t = 8 to t = 10</td>
<td>Processing A16</td>
<td>Processing A6</td>
<td>-----</td>
<td>-----</td>
<td></td>
</tr>
<tr>
<td>t = 10 to t = 11</td>
<td>Processing A24</td>
<td>Processing A6</td>
<td>-----</td>
<td>-----</td>
<td></td>
</tr>
<tr>
<td>t = 11 to t = 12</td>
<td>Processing A24</td>
<td>Processing A15</td>
<td>-----</td>
<td>-----</td>
<td></td>
</tr>
<tr>
<td>t = 12 to t = 14</td>
<td>Processing A5</td>
<td>Processing A9</td>
<td>-----</td>
<td>Processing A23</td>
<td></td>
</tr>
<tr>
<td>t = 14 to t = 15</td>
<td>Processing A5</td>
<td>Processing A9</td>
<td>-----</td>
<td>Processing A14</td>
<td></td>
</tr>
<tr>
<td>t = 15 to t = 16</td>
<td>Processing A5</td>
<td>-----</td>
<td>Processing A13</td>
<td>-----</td>
<td></td>
</tr>
<tr>
<td>t = 16 to t = 19</td>
<td>-----</td>
<td>-----</td>
<td>Processing A4</td>
<td>Processing A12,</td>
<td></td>
</tr>
<tr>
<td>t = 19 to t = 20</td>
<td>-----</td>
<td>Processing A3,</td>
<td>Processing A11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t = 20 to t = 22</td>
<td>-----</td>
<td>-----</td>
<td>Processing A11</td>
<td>Processing A2,</td>
<td></td>
</tr>
<tr>
<td>t = 22 to t = 23</td>
<td>-----</td>
<td>-----</td>
<td>Processing A2,</td>
<td>Processing finished</td>
<td></td>
</tr>
<tr>
<td>t = 23 to t = 25</td>
<td>Processing A1,</td>
<td>Processing finished</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 5. Gantt chart of the algorithm in this paper with 25 man-hours.
5. Algorithms Comparison

In this paper, the algorithm is compared with the integrated scheduling algorithm, which is closely connected with the working procedure group and the timing integrated scheduling algorithm which considers the serial process tightness. The results show that the equipment resources collaborative processing ability and the overall optimization effect of the proposed algorithm are much better than the other two algorithms.

5.1. Comparison of Integrated Scheduling Algorithms for Working Group Linkages

In the integrated scheduling algorithms for working group linkages, all the processes in the process tree are divided into two types: tight connection and non-tight connection. First, the first adaptive scheduling strategy is used to prioritize the processes in the tightly connected working procedure group and their tight pre-processes, and then the ACPM is used to schedule the tightly connected working procedure group; the final plan is \{A27, A26, A21, A22, A20, A19, A17, A18, A16, A15, A14, A25, A24, A23, A13, A8, A10, A7, A6, A5, A4, A9, A3, A12, A11, A2, A1\}, as shown in Figure 6, and the total time is 28 man-hours.

![Gantt chart](image)

Figure 6. Gantt chart of integrated scheduling algorithm with working group linkages with 28 man-hours.

The Gantt charts of the proposed algorithm in this paper and the integrated scheduling algorithm with working group linkages are shown in Figures 5 and 6, respectively. It is obvious that the start time of A3 in Figure 4 is \( t = 19 \), which is earlier than \( t = 23 \) in Figure 5. That means that the start time of A3 is four man-hours ahead using the proposed algorithm. This is because A4, the tight pre-process of A3, starts at \( t = 16 \) on M3. And the start time of A3 in Figure 4 is \( t = 19 \), which is earlier than \( t = 23 \) in Figure 5. That means that the start time of A3 is four man-hours ahead by using the proposed algorithm. This is because A25, the tight pre-process of A3, starts at \( t = 6 \) on M2, which is 2 man-hours earlier than \( t = 8 \) in Figure 6. And because A7 starts processing at \( t = 4 \) on M1, which is 8 man-hours earlier than \( t = 12 \) in Figure 6, the start time of its post-process A5 start 5 man-hours earlier, A3 starts 4 man-hours earlier, and A2 starts 3 man-hours earlier. And then the overall processing time of complex products is shortened.

5.2. Comparison of Time-Selecting Integrated Scheduling Algorithm Considering Serial Process Tightness

The time-selecting integrated scheduling algorithm considers that serial process tightness is based on the principle of the first integral and then division. According to the structure position of the process in the process tree, the sorting strategy is used to prioritize the process sequence with a close serial relationship, and then the time-selected strategy is adopted on the scheduled processing sequence. The minimum and earliest schemes of the total processing time are selected from the set of scheduling schemes.
For the process tree shown in Figure 3, the initial scheduling scheme is formed by the time selection algorithm considering that the tightness of the serial process is \{A1, A2, A11, A12, A13, A14, A15, A16, A19, A20, A21\}. Then, this is adjusted in the order of \{A3, A4, A6, A7, A8, A23, A24, A25, A27, A9, A18, A22, A26, A10, A17, A21\}. The results of the Gantt chart are shown in Figure 7, and the total processing time is 31 man-hours.

![Gantt chart](image)

**Figure 7.** Gantt chart of the time-selecting integrated scheduling algorithm considering serial process tightness with 31 man-hours.

The Gantt charts of the proposed algorithm in this paper and the time-selecting integrated scheduling algorithm considering serial process tightness are shown in Figures 5 and 7, respectively. In the proposed algorithm, all devices are processed from the starting point of \( t = 0 \), which is generally higher than the connection degree of each process in the time-selected algorithm.

The last process on M4 finished at \( t = 19 \) in Figure 4. And in Figure 7, the total number of idle time periods from \( t = 0 \) to \( t = 6 \), \( t = 9 \) to \( t = 10 \), \( t = 11 \) to \( t = 14 \), and \( t = 20 \) to \( t = 21 \) before the last process on M4 finished is 12. In Figure 5, the total number of idle time periods from \( t = 5 \) to \( t = 12 \) and \( t = 15 \) to \( t = 16 \) is eight. It is obvious that the number of idle time in Figure 7 is more than it is in Figure 5. And it increased 50% in equipment utilization.

The last process on M3 finished at \( t = 23 \) in Figure 5. And in Figure 7, the total number of idle time periods from \( t = 0 \) to \( t = 2 \), \( t = 10 \) to \( t = 14 \), \( t = 16 \) to \( t = 26 \), and \( t = 27 \) to \( t = 28 \) before the last process on M4 finished is 17. In Figure 5, the total number of idle time periods from \( t = 4 \) to \( t = 6 \) and \( t = 8 \) to \( t = 15 \) is seven. It is obvious that the number of the idle time in Figure 7 is more than that in Figure 5. And it increased 12.5% in equipment utilization.

The last process on M2 finished at \( t = 20 \) in Figure 5. And in Figure 7, the total number of idle time periods from \( t = 0 \) to \( t = 2 \), \( t = 14 \) to \( t = 18 \) and \( t = 19 \) to \( t = 24 \) before the last process on M4 finished is 13. In Figure 5, the total number of idle time periods from \( t = 3 \) to \( t = 4 \) and \( t = 15 \) to \( t = 19 \) is five. It is obvious that the number of idle time in Figure 7 is more than that in Figure 5. And it increased 21.4% in equipment utilization.

The last process on M1 finished at \( t = 25 \) in Figure 5. And in Figure 7, the total number of idle time periods from \( t = 0 \) to \( t = 4 \) and \( t = 18 \) to \( t = 29 \) before the last process on M4 finished is 15. In Figure 5, the total number of idle time periods from \( t = 6 \) to \( t = 8 \) and \( t = 16 \) to \( t = 23 \) is nine. It is obvious that the number of the idle time in Figure 7 is more than that in Figure 5. And it increased 20.4% in equipment utilization.

In summary, from the point of view of the equipment utilization ratio on each equipment, the proposed algorithm reduces the scheduling interval between processes (groups) vertically. And the priority scheduling of the leaf node process further reduces the idle time period on the processing equipment, so the utilization ratio of the equipment is improved from the two angles of working procedure and equipment.

The equipment utilization of the three algorithms is shown in Table 3.
Table 3. Comparative analysis of equipment utilization of three algorithms.

<table>
<thead>
<tr>
<th>Equipment Utilization</th>
<th>M1, M2, M3, M4 Total Idle Time</th>
<th>Total Processing Time of Product A</th>
<th>Equipment Utilization Ratio</th>
<th>Relative Improvement Ratio of Equipment Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integrated Scheduling Algorithms for Working Group Linkages</td>
<td>39</td>
<td>28</td>
<td>59.4%</td>
<td>7.3%</td>
</tr>
<tr>
<td>Time-Selecting Integrated Scheduling Algorithm Considering Serial Process Tightness</td>
<td>56</td>
<td>31</td>
<td>50.4%</td>
<td>16.3%</td>
</tr>
<tr>
<td>The proposed algorithm</td>
<td>29</td>
<td>25</td>
<td>66.7%</td>
<td>-----</td>
</tr>
</tbody>
</table>

5.3. Comparison and Analysis of Three Algorithm Scheduling Results

According to the structure diagram of complex product A shown in Figure 3, the total processing time of the proposed algorithm is 25 man-hours, the total processing time of the time-selected algorithm is 31 man-hours, the total processing time of integrated scheduling algorithms for working group linkages is 28 man-hours, and the comparison and analysis of the three algorithm scheduling results are shown in Table 4.

Table 4. Comparison and analysis of scheduling results of three algorithms.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule plan</td>
<td>The first adaptive scheduling strategy is used in the priority scheduling process (group), and the non-tightly connected processes are scheduled according to the ACPM.</td>
<td>The time-selecting strategy is used to analyze the scheduled process, the subsequent process strategy is considered to analyze the unscheduled process, and the starting time point of the process is determined synthetically.</td>
<td>By using the inverse sequence idea of the shortest path of Dijkstra algorithm, the scheduling strategies of larger path value priority, layer priority, and leaf node process priority are adopted.</td>
</tr>
<tr>
<td>Total processing time</td>
<td>28</td>
<td>31</td>
<td>25</td>
</tr>
</tbody>
</table>
followed by TS, while WGL takes the most time, except for a few instances. To compare these three algorithms more intuitively, the optimal solution ratio and variance curves are plotted in Figures 9 and 10, respectively.

In Figure 9, it can be seen from the comparison of the optimal solution ratio that HIS-IDA always maintains a higher ratio, while the ratios of TS and WGL gradually decrease. The results of HIS-PCVM are better than the other two algorithms. From Figure 10, it can be seen that the stability of HIS-PCVM is the best. The variance curve of HIS-PCVM is always at a lower level and more stable than the other two algorithms. This means that the solution obtained by HIS-PCVM can be closer to the best one.

Figure 8. Results of HIS-IDA, TS, and WGL for integrated scheduling 40 product instances: (a) 20 operations, (b) 50 operations, (c) 100 operations, and (d) 200 operations.

Figure 9. Optimal solution ratio of HIS-IDA, TS, and WGL for integrated scheduling 40 product instances.
An improved Dijkstra algorithm is used to schedule each process according to their path values in a descending order. The main reason why this algorithm is better is as follows.

According to the structure characteristics of the complex product process tree, the algorithm according to the improved Dijkstra algorithm to schedule the processing process by the path value from large to small, its essence is to improve the serial tightness of the processing process sequence vertically. The layer priority strategy also improves the parallel processing ability of the process. At the same time, according to the leaf node process priority strategy, the process as the leaf node is scheduled as a priority. Then, the connection degree of all tight processes (groups) in the leaf node process is also improved.

Compared with the integrated scheduling algorithms for working group linkages, this algorithm fully examines the relative position of each process in the complex product process tree, which not only pays attention to the process in the tight connection process group but also takes into account the process in the non-tight connection group of the process tree. However, the integrated scheduling algorithms for working group linkages adopt the scheduling scheme of grouping first and then making decisions. First, all the processes are divided into a tight connection process group and non-tight connection process group. When there are leaf node processes in the tight connection process group, the ACPM adopted for the non-tight connection process group reduces the constraint of process in the process tree, thus affecting the overall effect of integrated scheduling. Such as that M2 has a long idle time period from \( t = 17 \) to \( t = 23 \), M3 has a long idle time period from \( t = 8 \) to \( t = 14 \), and M4 has a long idle time period at 5 to 11.

Compared with the time-selecting integrated scheduling algorithm considering serial process tightness, the priority strategy of the leaf node process adopted in the proposed algorithm not only takes into account the same layer process but also makes full use of the idle time period of the equipment, so that the process has a good advantage in the early processing. However, in the time-selecting integrated scheduling algorithm considering serial process tightness, the minimum and earliest scheme of total processing time is inserted into the serial process sequence, which increases the connection between the non-leaf node before and after the tight process and does not fully take into account the utilization of the equipment. Such that M1 idled from \( t = 18 \) to 29, so did M3. This led to having an overall impact on the processing process of complex products.

6. Conclusions

An improved Dijkstra algorithm is used to schedule each process according to their path values in a descending order. Its essence is “longitudinal optimization”, that is, priority scheduling the process with a longer longitudinal path to shorten the overall processing time. On this basis, we used the
layer priority strategy for “horizontal optimization”, that is, reducing the processing time intervals between processes. And the leaf node priority strategy is “idle optimization”, that is, making full use of the idle time on the device. Although most of the current scheduling algorithms have better optimized the effect of comprehensive scheduling, there is a problem of taking one into account and losing the other in horizontal and vertical optimization. The proposed algorithm optimizes the overall processing time of complex product scheduling from three perspectives. Compared with integrated scheduling algorithms for working group linkages and the time-selecting algorithm, which considers the tightness of serial process, the equipment utilization ratio is increased by 7.3% and 16.3%, respectively, thus achieving a better scheduling effect. Therefore, this algorithm is better. It provides a new method to solve the integrated scheduling of complex products and expands new ideas. It has a certain theoretical and practical significance.

Author Contributions: Conceptualization, W.Z. and Z.X.; writing original draft preparation, Z.T. and W.Z.; validation, Z.T. and P.Z.; data curation, P.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported in part by the National Natural Science Foundation of China (61772160); The “New Generation Information Technology Innovation Project” of the China University Industry, University and Research Innovation Fund (ITA05024, ITA05034); Jilin Normal University Doctoral Program (2022014).

Data Availability Statement: Not applicable.

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

References

25. Mohammad, P.; Aritra, M.; Shreyas, S. Graph-theoretic approaches for analyzing the resilience of distributed control systems: A tutorial and survey. Automatica 2023, 157, 111264. [CrossRef]