Integrated Scheduling Algorithm For Complex Products Based On The Dynamic Subtree Operation Set Inverse Coding

Qian Wang
Harbin University of Science and Technology

Zhiqiang Xie (xzq1962hrbust@126.com)
Harbin University of Science and Technology
https://orcid.org/0000-0001-9828-3034

Yilong Gao
Linyi University

Xu Yu
Qingdao University of Science and Technology

Research Article

Keywords: Tree-structured products, processing and assembly, integrated scheduling algorithm, inverse scheduling, genetic algorithm

Posted Date: January 31st, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1237010/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License
Integrated scheduling algorithm for complex products based on the dynamic subtree operation set inverse coding

Qian Wang, Zhiqiang Xie*, Yilong Gao and Xu Yu

Abstract
For the previous integrated scheduling algorithms for complex products, the migration time of operations between machines is ignored or just included in the processing time of its adjacent operations, which leads to inaccurate scheduling results and is difficult to meet the needs of the actual production scheduling environment. In this paper, based on the framework of a genetic algorithm, an integrated scheduling algorithm for complex products based on the dynamic subtree operation set inverse coding. Firstly, an inverse coding method based on the dynamic subtree operation set is proposed, which can ensure the legitimacy of the initial individuals and enhance the quality of the initial population. Secondly, based on the crossover vector, a single-point crossover method and a multi-point crossover method are proposed, both of which can ensure that the priority constraints among the same machine operation in the generated individuals will not be destroyed. Then, a mutation method based on the mutant row vector and mutant column vector is proposed to ensure the feasibility and diversity of the offspring individuals. Finally, a pre-decoding method based on device idle events driven and a conversion strategy of positive sequence schemes based on the completion time flipping are shown. The performance of the proposed algorithm is verified by several groups of comparative experiments.

Keywords: Tree-structured products; processing and assembly; integrated scheduling algorithm; inverse scheduling; genetic algorithm

1 Introduction
The traditional production scheduling problem takes processing [1, 2] and assembly [3, 4] as two independent stages. The research on production scheduling problems mainly focuses on the processing stage, such as the flow shop scheduling problem [5, 6] for the production environment of batch processing and the job shop scheduling problem for multi-variety and small-batch production environment [7, 8]. With the improvement of people’s living standards, the demand for personalized customized products gradually increases. At this time, if we still follow the previous production mode of "processing first and then assembling", the inherent parallel processing relationship of the product will inevitably be cut off, which will lead to a prolonged product completion time and affect customer satisfaction. Therefore, in recent years, more and more experts and scholars have paid attention to the integrated scheduling algorithms [9], which considers the processing and assembly at the same time. In this problem, we map a single complex product into a processing operation tree. For example, the processing operation tree of a complex product is
shown in Figure 1. Each operation is represented by a node, and each node is composed of three parts, respectively representing the operation’s name, the processing machine, and its corresponding processing time. The priority processing constraints between operations are specified by the arrows in the processing operation tree. The processing of the product starts from the leaf node, and the root node processing is finished, which means the product is completed.

At present, the research on the integrated scheduling problem mainly focuses on the simple products [10, 11], that is, the structure of the processing operation tree is relatively simple. However, there are relatively few researches on complex products. And the algorithms involved can be divided into two categories: rule-based heuristic algorithm and meta-heuristic algorithm-based integrated scheduling solutions. In the first type of algorithm, the authors [12] have conducted in-depth research on this problem and proposed a series of rule-based heuristic algorithms. For example, for the general integrated scheduling problem, the authors [13] propose a time-selective integrated scheduling algorithm with backtracking adaptation strategy, and the authors [14] propose an integrated scheduling algorithm based on event-driven by machines’ idle. Aiming at the multi-shop integrated scheduling problem, the authors [15] propose an integrated scheduling algorithm of two workshops based on the principle of the neighborhood rendering, the authors [16] propose an integrated scheduling algorithm of two workshops based on optimal time, and the authors [17] propose a reversal sequence integrated scheduling algorithm of multiple workshop with multi-procedures ended together. For the integrated scheduling problem of complex products with special constraints, the authors [18] propose an integrated scheduling algorithm with multiple-devices-operation, and the authors [19] propose an integrated scheduling algorithm with no-wait constraint operation group. Aiming at the integrated scheduling problem of special production environment, the authors [20] propose a multi-batch integrated scheduling algorithm based on time-selective, and the authors [21] propose a multi-batch processing integrated scheduling algorithm based on signal driven. However, the above algorithms are designed based on the processing operation model and for different processing operation trees with different structures, the quality of the solutions of each algorithm is different, and the diversity of the solutions is insufficient. In the second type of the integrated scheduling solutions, due to the existence of complex priority constraints among operations, it is difficult to directly apply the existing evolutionary operators to solve the integrated scheduling problem. Therefore, there are relatively few algorithms involved, and most of them are based on the genetic algorithm. For example, the authors [22, 23] put forward solutions based on the genetic algorithm, but they all need to consider the detection and repair of infeasible chromosomes generated by coding and genetic operators, which leads to the increase of calculation and the decrease of execution efficiency of the algorithm. The authors [24] propose a product comprehensive scheduling algorithm based on virtual component level division coding. The algorithm sets the level of each part of the product and codes the parts according to the level. This coding method ensures the feasibility of the initial solution, but the coding method has shortcomings. It implicitly adds additional constraints to some operations, which results in reducing the space of feasible solutions, that is, there is a risk of missing the optimal solution.
authors [25] propose a product comprehensive scheduling problems solved by genetic algorithm based on operation constraint chain coding. The designed coding method ensures the feasibility and completeness of the initial solution space, but the crossover and mutation operators designed by the algorithm will produce infeasible solutions, which requires additional detection and repair work. The authors [26] propose an integrated scheduling algorithm based on an operation relationship matrix table for tree-structured products. Firstly, an operation relationship matrix table is established for the product, and then the corresponding coding rules and evolution operators are designed based on the table. The authors [27] propose a hybrid algorithm based on improved extended shifting bottleneck procedure and GA for assembly job shop scheduling problem, but it also needs to design corresponding repair operators for infeasible chromosomes.

However, the above-mentioned integrated scheduling algorithms for complex products don’t consider the transportation time of operations between machines or simply include the transportation time in the relevant processing time of its predecessors or successors. However, in the actual production scheduling process, the transportation time of operations between different machines is difficult to ignore. Therefore, in this paper, aiming at the integrated scheduling problem of complex products with tree structure considering the migration time, an integrated scheduling algorithm based on the dynamic subtree operation set inverse coding is proposed. The algorithm adopts the idea of inverse scheduling, which simplifies the problem that the completion time of the predecessors affects the start time of the current operation when scheduling in positive order. An encoding method based on inverse coding of dynamic subtree operation set is presented. Then, based on the designed individual representation method, different evolutionary operators based on crossover vector and mutation vector are given respectively. Finally, a simple pre-decoding method based on device idle event-driven and a conversion strategy from reverse-order scheduling scheme to forward-order scheduling scheme are given respectively. The superiority of the proposed algorithm is verified by a number of comparative experiments.

The rest of the paper is arranged as follows. In the next section, a detailed description of the integrated scheduling problem considering the migration time will be given. Then, Section 3 describes the design process of the proposed algorithm in detail, including the encoding and decoding method, the genetic operators. The superiority of the proposed algorithm is verified by comparative experiments in Section 4. Finally, in Section 5 we draw some conclusions and prospects for future research work.

2 Mathematical model

The integrated scheduling problem of complex products considering the operation migration time studied in this paper can be described as: a single complex tree-structured product consisting of \( n \) operations needs to be collaboratively processed by multiple machines, and each machine has a certain geographical distance. Therefore, the transportation time of operations between different machines cannot be ignored. Assuming that the operation set is \( \{O_i\}_{1 \leq i \leq n} \), the processing machine of operation \( O_i \) is \( M_i \in M = \{1, 2, ..., m\} \), and \( p_i \) is the processing time of \( O_i \) on
machine $M_i$. $T_{M_k,M_i}$ is the transportation time of an operation between the machine $M_k$ and $M_i$. The processing priority among operations is specified by the constraint relationship set $C = \{(i, j) | i, j \in \{1, 2, ..., n\}\}$. If $(i, j) \in C$, it means that operation $O_i$ is the immediate predecessor of $O_j$, that is, operation $O_j$ can not be processed until $O_i$ is finished. $s_i$ and $c_i$ denote the start processing time and the completion time of $O_i$ respectively. $x_{ik} = 1$ indicates that operation $O_i$ is processed on machine $k$; $y_{ij} = 1$ indicates that operation $O_i$ and $O_j$ are assigned to the same machine, and $O_i$ is the immediately preceding operation of $O_j$, that is, $O_j$ can only be processed after $O_i$ is completed. The scheduling goal of the problem is to make the product’s completion time as short as possible. So the mathematical description of this problem can be describe as follows:

$$\min \left\{ \max \{c_i\}_{i=1,...,n} \right\}$$  \hspace{1cm} (1)

s.t.:

$$\sum_{k=1}^{m} x_{ik} = 1, \ i = 1, \ldots, n$$  \hspace{1cm} (2)

$$s_i \geq \max \{c_j + T_{M_i,M_j}\}, \ (j, i) \in C, \ i, j = 1, \ldots, n$$  \hspace{1cm} (3)

$$c_i = s_i + p_i, \ i = 1, \ldots, n$$  \hspace{1cm} (4)

$$y_{ij} + y_{ji} \geq x_{ik} + x_{jk} - 1, \ k \in M_i \cap M_j$$  \hspace{1cm} (5)

$$s_j \geq s_i + p_i - (1 - y_{ij})L, \ M_i \cap M_j \neq \emptyset$$  \hspace{1cm} (6)

$$s_i \geq 0, \ i = 1, \ldots, n$$  \hspace{1cm} (7)

Among them, Equation (1) is the objective function of the problem, that is, to minimize the maximum completion time of the operations. Equation (2) means that each operation should be assigned to only one machine for processing. Constraint (3) indicates that $O_i$ can only be processed after all its technology predecessors having finished and having been transported to the current processing machine. Equation (4) denotes the completion time of operation $O_i$. Constraints (5) and (6) indicate that for two consecutive operations assigned to the same machine, the start time of the latter operation cannot be earlier than the completion time of the former operation. $L$ represents a sufficiently large positive constant. Constraint (7) indicates that the start time should have general significance.

3 Algorithm design

As an intelligent search algorithm based on population evolution, genetic algorithm has been applied to solve many combinatorial optimization problems [28, 29], and
has achieved good results. Therefore, in this paper, based on the framework of a genetic algorithm, we propose an integrated scheduling solution for complex products considering the operation migration time.

3.1 The inverse encoding method based on the dynamic subtree operation set

From the processing operation tree, we can see that when the positive sequence scheduling is started from the leaf node operations, there may be multiple preceding operations for a certain operation. Therefore, if the scheduling sequence of the predecessors is unreasonable, the completion time of the current operation will be delayed, which will affect the completion time of the whole product. However, when we consider it in reverse order, that is, the root node operation is scheduled first in the scheduling process, and the leaf node operation is finally scheduled. In this reverse order scheduling method, each operation has at most one immediately preceding operation. The start processing time of the current pending process is only related to the completion time of its predecessor, the migration time, and the processing conditions of the corresponding machine. From this, we can see that in the reverse order scheduling method, the number of operations that affect the start time of the pending operation will be reduced, which makes it easier to schedule among operations. Therefore, in this section we adopt the idea of inverse scheduling to encode chromosomes, that is, each time the root node of the processing operation tree is selected for scheduling. In this paper, an individual coding structure based on multi-chain is adopted, that is, each chromosome is represented by $m$ operation chains, where $m$ is the number of machine.

Based on the idea of inverse scheduling and the individual representation method adopted, this paper presents an inverse coding method based on the dynamic subtree operation set. The main idea is: firstly obtain the current subtree operation set, the only element in the subtree operation set at the initial time is the current product’s processing operation tree. Secondly, randomly select a subtree in the subtree operation set, obtain the root node operation of the subtree, take the root node operation as the currently selected operation, obtain the processing machine of the selected operation, and insert the selected operation into the first idle position of the operation chain corresponding to the current machine. Then, update the subtree operation set, that is, if the currently selected subtree is a single-node operation tree, it will be directly removed from the subtree operation set; if the currently selected subtree is a multi-node process tree, delete the root node operation from the currently selected subtree to generate new subtrees, and add the newly generated subtree to the subtree operation set. Repeat the above steps until the subtree operation set is empty, then the encoding process ends.

Figure 2 is a schematic diagram of generating chromosomes using the inverse encoding method based on the dynamic subtree operation set described herein. The figure shows a simple product. Initially, the processing operation tree contains 6 operation nodes, and operations of the same color are processed on the same machine. During the first iteration, the subtree operation set contains only one subtree, so this subtree is selected. The root node of the currently selected subtree is operation 1, so operation 1 is scheduled first and inserted into the first free position of the machine operation chain corresponding to the chromosome. Since
the currently selected subtree is a multi-node processing operation tree, deleting the node corresponding to operation 1 can generate two new subtrees, and the newly generated subtrees are shown in the second step. Similarly, in the second iteration process, the subtree is randomly selected, its root node is obtained, and it is inserted into the corresponding position of the chromosome, and then the subtree operation set is updated. Repeat the above steps until the subtree operation set is empty. It can be seen from Figure 2 that the coding of this simple product example has been iterated 6 times in total, and the resulting chromosome structure is as shown in the sixth step. Figure 3 is a schematic diagram of chromosomes obtained by applying the designed encoding method to encode the product shown in Figure 1.

3.2 Selection method
In this paper, we use a tournament selection strategy to determine the parent chromosomes participating in evolution, that is, four individuals are randomly selected from the parent population, and two individuals with shorter completion time are selected as the paired parent chromosomes to perform the subsequent evolutionary operators. Also, in this paper, we use the elite retention strategy, that is, the best parent individual directly replaces the worst offspring individual.

3.3 Crossover method
Based on the designed encoding method, we give two crossover methods in this section. Firstly, randomly generate a crossover indicator vector, the vector is a column vector, the size is equal to the number of machines, and its element value is 0 or 1. 1 indicates that the corresponding operation chain will undergo crossover process, and 0 indicates that the corresponding operation chain will be inherited intact to the offspring individual. After defining the crossover indicator vector, we give two crossover methods as follows, which are executed randomly.

1) The designed crossover method based on single-point crossover row vector
In this crossover method, we randomly generate a row vector whose size is equal to the number of operations in the longest operation chain. The element value is 0 or 1, and only one element can have a value of 1. In order to ensure that two paired parent chromosomes can generate new offspring individuals, we stipulate that the element value 1 cannot be located in the first or last position of the crossover row vector. If the current operation chain implements the single-point row vector-based crossover method described in this paper, we will take out all operation genes before or at the position where the element value is 1, and then backfill them to the idle positions of the current chromosome in the order in which these genes appear in the other paired chromosome. The operation genes of the remaining positions remain unchanged. Repeat the above steps until all the relevant operation chains are executed. Figure 4 is a schematic diagram generated by applying the single-point row vector-based crossover method described herein to generate new offspring individuals.

2) The designed crossover method based on multi-point crossover row vector
In this crossover method, we randomly generate a row vector whose size is equal to the number of operations in the longest operation chain, and the element value is 0 or 1. If the current operation chain implements the crossover method based on the multi-point crossover row vector described in this paper, we will take out the
operation genes corresponding to the continuous element value of 1 in turn, and then refill them to the idle position according to the order in which they appear in the other paired parent chromosome. Perform the above steps on all sub-operation strings with a continuous element value of 1, and the elements in the position with an element value is 0 remain unchanged. Repeat the above steps until all the relevant operation chains are executed. Figure 5 is a schematic diagram of generating new offspring individuals by applying the crossover method based on the multi-point crossover row vector.

3) Analysis of the crossover method

In the crossover method based on the single-point crossover row vector described above, the sequence of the operation genes before the element value 1 is specified by another paired chromosome. Since the sequence of the recombinant genes in the paired chromosomes are legal arrangements, the resulting offspring individuals must also be legal chromosomes. In the crossover method based on multi-point crossover row vector described above, the sequence of operations in the recombination string corresponding to the continuous element value of 1 is determined by the order of appearance of the corresponding operation genes in the paired chromosome. Since the sequence of the operations in the corresponding recombination string in each paired parent chromosome is legal, the offspring individuals generated must also be legal chromosomes. In summary, performing the crossover methods described herein will not produce infeasible chromosomes.

3.4 Mutation method

In this paper, we present two different mutation methods. During the execution of the algorithm, the designed mutation method is randomly selected for execution.

1) The designed mutation method based on the mutant row vector

In this mutation method, we randomly generate a mutant row vector whose size is the number of operations in the longest operation chain. The element value is 0 or 1, and only one element value is 1. To ensure the effect of mutation, we stipulate that the first element of the mutant row vector cannot be 1. The main idea is: obtain all the operations before the position of the element value 1, according to the priority constraints specified by the original processing operation tree, map the involved operations into a new processing operation tree, generate a new schedule arrangement following the encoding method described in this paper, and fill it back to the corresponding idle position. The operations corresponding to the position where the element value is 0 remain unchanged. Figure 6 is a schematic diagram of applying the mutation method based on the mutant row vector described herein to generate a new individual.

2) The designed mutation method based on the mutant column vector

In this mutation method, we randomly generate a mutant column vector whose size is equal to the number of machines, and the element value is 0 or 1. An element value of 1 indicates that the operation chain corresponding to the element needs to participate in mutation, and an element value of 0 means that the operation chain corresponding to the element will not change. For the mutated operation string, in order to ensure that the mutated operations on the same machine still satisfies the priority constraint relationships, we use pipelines for mutation. The operation in
the mutation substring are taken out one by one, and added to different sub-pipes according to the constraint relationship, so that the operations in the same pipeline have constraint relationships, and there is no constraint relationship between the operations in different pipelines. Then randomly take operations out from different pipes and place them in the first idle positions of the mutation string in turn. Finally, the newly generated operation string is put back to the original position of the chromosome, and the mutation process ends. Figure 7 is a schematic diagram of applying the mutation method based on the mutant column vector described herein to generate a new individual.

3) Analysis of the mutation method
In the mutation method based on the mutant row vector described above, since the principle of the encoding method described in this paper is adopted, the arrangement of operations corresponding to new subtree must meet the constraints, and the new mutation individuals generated must also be a legal chromosome. In the second mutation method described above, since the pipe is used to ensure that the operation constraint relationship on the same machine is not destroyed, the newly generated mutation substring is a legal substring, thereby the newly generated mutation individual is a legal individual. In summary, implementing the mutation method described herein will not produce infeasible chromosomes.

3.5 The pre-decoding method based on the machine idle event-driven
Aiming at the encoding method designed in this paper, a pre-decoding method based on the machine idle event-driven is proposed. We define that if the scheduling state is at the initial processing time (0 moment) or the operation completion time, a machine idle event will be generated. If the machine idle event occurs, we will schedule the relevant schedulable operations. The main idea is: if a machine idle event is generated, the idle machine set at the current moment is obtained. Get an idle machine from the idle machine set in turn, find the operation chain corresponding to the current machine in the chromosome, and determine whether the current operation chain is empty, if it is empty, remove the machine element from the idle machine set; otherwise, obtain the first schedulable operation of the corresponding operation chain and judge whether all its predecessors have been processed. If the processing is completed, put the operation at the current moment or the moment when the migration of the relevant operations is completed, and delete the corresponding operation from the chromosome. otherwise, remove the machine from the idle machine set. If the current idle machine set is empty, the above process is repeated when the next machine idle event occurs. If all the operation genes in the chromosome are read, the pre-decoding process ends. Figure 8 is a reverse order scheduling scheme obtained by decoding the individuals shown in Figure 3 using the pre-decoding method based on the machine idle event-driven described herein.

3.6 The conversion strategy based on completion time flipping
Since the encoding method adopted in this paper is reverse order coding, it is also in reverse order when decoding a chromosome into a specific scheduling scheme. In order to convert the reverse order scheduling scheme to the positive order scheduling scheme, that is, the scheduling of the product starts from the leaf node, this paper
presents a scheduling scheme conversion strategy based on the completion time flipping. Suppose we know the final completion time of the product, the inverse start time, and the inverse completion time of each operation. Then the specific steps of the conversion strategy described in this paper are as follows: Firstly, subtract the product completion time of the current scheme from the inverse start time and inverse completion time of each process in reverse scheduling; then, reverse the results corresponding to each operation, that is, each start time and completion time value become a non-negative number; finally, the start time and completion time corresponding to each operation in the positive sequence scheduling can be obtained by exchanging the corresponding start time and completion time of each operation. Note that after the transition to the positive sequence scheduling scheme, if all the immediate predecessors of an operation have been processed, and there is an idle time period before the starting time of the current operation, the current operation can move forward. However, no matter in the positive sequence scheduling scheme or the reverse sequence scheduling scheme, the final completion time of the product is unchanged. Therefore, in this paper, we don’t consider the movement of the relevant operation. Figure 9 shows the Gantt chart of positive sequence scheduling obtained by applying the conversion strategy described herein for the reverse sequence scheduling scheme shown in Figure 8. In the process of algorithm iteration, we only need to know the product completion time corresponding to the current individual, and we don’t need to know the positive sequence scheduling time of each operation. Therefore, we only need to implement the conversion strategy for the optimal reverse scheduling scheme at the end of the algorithm.

3.7 The overall structure of the designed algorithm

The integrated scheduling algorithm of complex products considering operation migration time based on the dynamic subtree operation set inverse coding is shown in Figure 10.

4 Experimental results

At present, there are a large number of benchmark instances for traditional job shop scheduling problems, but no available benchmarks have been found for the integrated scheduling problem of tree-structured complex products. In the previous literature on the integrated scheduling problems, the comparative examples are all single product examples. For the complex product integrated scheduling problem with operation migration time considered in this paper, we first compare the existing algorithms with a single example in the previous literature without considering the operation migration time. In addition, in order to fully test the solving ability of the proposed algorithm, we randomly generated 100 instance. Each instance is randomly generated by the following parameters: the number of operations is [50,100], the number of machines is [4,8], the transportation time between machines is [1,10], the processing time is [1,20], and the layer number of processing operation tree is [2, [n/2]], where n is the number of operations in the current processing operation tree. In this section, the comparison algorithms are: the dynamic critical paths multi-product manufacturing scheduling algorithm based on operation set (DCPMMSA_OS) [30], the integrated scheduling algorithm based on event-driven
by machines’ idle (ISA_EDMI) [14], the time-selective integrated scheduling algorithm considering the compactness of serial processes (TISA_CCSP) [31], the product comprehensive scheduling algorithm based on virtual component level division coding (CSA_VCLDC) [24], the integrated scheduling algorithm based on an operation relationship matrix table for tree-structured products (ISA_ORMT) [26], and the algorithm ISA_STOS proposed in this paper. All algorithms are programmed by MATLAB R2016a and run on a computer with Win10, 64-bit operating system, Intel(R) Core(TM) i7-4810MQ CPU @2.80GHz, and 16G memory. The algorithms CSA_VCLDC, ISA_ORMT, and ISA_STOS independently run and solve each instance 10 times. The parameters are set as follows: population size is $n \times 4$, evolutionary generation is 100, crossover probability is 0.8, and mutation probability is 0.1.

4.1 Experiment without considering migration time

For the product processing operation tree shown in Figure 1, in this section we use the above 6 algorithms to solve the product scheduling. In the solution process, we do not consider the migration time of the operation between machines, that is, assume that the migration time is 0. The solution results of each algorithm are shown in Table 1. It can be seen from Table 1 that the solution results of the algorithms CSA_VCLDC, ISA_ORMT and ISA_STOS are better than those of other algorithms. This is because the first three algorithms are rule-based heuristic algorithms, which have different ability to solve instances with different structures. Among the last three solutions based on the meta-heuristic algorithm, the minimum completion time and average completion time obtained by the proposed algorithm ISA_STOS are better than the other two algorithms. Figure 11 shows the optimal scheduling scheme obtained by the proposed algorithm for the product shown in Figure 1.

| Algorithm       | DCPMMSA | ISA_EDMI | TISA_CCSP | CSA_VCLDC | ISA_ORMT | ISA_STOS |
|-----------------|---------|----------|-----------|-----------|----------|----------|
| Min Makespan    | 23      | 24       | 27        | 23        | 20       | 20       |
| Average Makespan| -       | -        | -         | 23.03     | 21.07    | 20       |

4.2 Experiment with considering migration time

In this section, for 100 randomly generated instances, we verify the solving capabilities of the algorithms CSA_VCLDC, ISA_ORMT, and the algorithm ISA_STOS proposed in this paper. Figure 12 shows the minimum makespan obtained by the above three algorithms for each instance. Figure 13 shows the average makespan obtained by the above three algorithms for each case. It can be seen from Figure 12 and Figure 13 that the solution curve of the proposed algorithm in this paper is generally located below the graph, which shows that the solution ability of the proposed algorithm in this paper is generally better than the other two comparison algorithms.

We define $M_{\text{min}} = \min(Makespan_{\text{isa}}), isa \in \{\text{CSA_VCLDC, ISA_ORMT, ISA_STOS}\}$, where $Makespan_{\text{isa}}$ represents the makespan obtained by applying algorithm $isa$ for a complex product instance. For a complex product instance, if
Makespan_{isa} is equal to M_{min}, then the value of ATC of Algorithm isa for solving the current instance is set to 1, otherwise, the value of ATC is set to 0. We use the value of ATC to assess the ability of the algorithm for solving the current complex product instance, if it is 1, it means that the algorithm is effective for solving the current instance. Therefore, we count the sum of ATC value of each algorithm for solving the instances mentioned in Figure 12 and Figure 13. The statistical results are shown in Figures 14. The first group of data is the statistical results of the minimum completion time, and the second group of data is about the average completion time. For 100 randomly generated instances, the algorithm CSA_VCLDC obtained the minimum makespan for 8 instances and the minimum average makespan for 8 instances. For 100 randomly generated instances, the algorithm ISA_ORMT obtained the minimum makespan for 32 instances and the minimum average makespan for 13 instances. For 100 randomly generated instances, the proposed algorithm ISA_STOS obtained the minimum makespan for 83 instances and the minimum average makespan for 91 instances. It can be seen from Figure 14 that the algorithm ISA/MITDD’s solving ability is significantly better than the other two algorithms.

For a randomly generated tree structure complex product, we count the optimal makespan and average makespan in each generation during the solution process by applying the above three algorithms. The specific change curve is shown in Figure 15 and Figure 16. From Figure 15 and Figure 16, it can be seen that the convergence speed of the proposed algorithm is significantly better than the other two comparison algorithms, and can converge to a better solution.

5 Conclusion and future research
Aiming at the integrated scheduling problem of complex products considering the operation migration time, we present an integrated scheduling algorithm based on the dynamic subtree operation set inverse coding. Based on the idea of reverse scheduling, the algorithm first starts scheduling from the root node, which reduces the factors affecting the starting time of the current scheduling operation during the scheduling process. The proposed inverse encoding method based on the dynamic subtree operation set ensures the feasibility of the initial population. The proposed crossover methods based on crossover vectors ensure the legitimacy of the generated offspring individuals. The proposed mutation methods based on mutant vectors enhance the diversity of the population. The proposed pre-decoding scheme based on machine idle event-driven can convert different individuals into reasonable scheduling schemes. The different comparative experiments verify the superiority of the proposed algorithm.

At present, the meta-heuristic algorithm-based solutions involved are mostly genetic algorithm-based solutions, and the method described in this paper is still a genetic algorithm-based integrated scheduling solution. Therefore, in order to explore the ability of other intelligent optimization algorithms to solve the integrated scheduling problems, in this paper, we only consider the single-objective scheduling problem, but in the actual production scheduling problem, there are many objectives that need to be considered, such as cost and other objectives. Therefore, the multi-objective integrated scheduling problem can be considered in the future. The
production environment studied in this paper is only the single-workshop production environment, and with the development of various technologies, the integrated scheduling problem of tree-structured complex products in the cloud manufacturing production environment is also one of the research directions in the future.

Acknowledgements
The authors acknowledged the anonymous reviewers and editors for their efforts in valuable comments and suggestions.

Funding
This work was supported by the National Natural Science Foundation of China [grant numbers 61772160, 61370086].

Abbreviations
GA: the genetic algorithm; DCPMMSA, OS: the dynamic critical paths multi-product manufacturing scheduling algorithm based on operation set; ISA, EDM: the integrated scheduling algorithm based on event-driven by machines’ idle; TISA, CCSP: the time-selective integrated scheduling algorithm considering the compactness of serial processes; CSA, VCLDC: the product comprehensive scheduling algorithm based on virtual component level division coding; ISA, ORMT: the integrated scheduling algorithm based on an operation relationship matrix table for tree-structured products; ISA, STOS: the algorithm proposed in this paper.

Availability of data and materials
Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

Authors’ contributions
QW proposes the innovation ideas and theoretical analysis, and ZX carries out experiments and data analysis. YG conceived of the study, and participated in its design and coordination and helped to draft the manuscript. All authors read and approved the final manuscript.

Declarations
Competing interests
The authors declare that they have no competing interests.

Author details
1 School of Computer Science and Technology, Harbin University of Science and Technology, 150080 Harbin, China. 2 School of Information Science and Engineering, Linyi University, Linyi, China. 3 School of Information Science and Technology, Qingdao University of Science and Technology, Qingdao, China.

References
1. Jelvez, E., Morales, N., Nancel-Penard, P., Cornillier, F.: A new hybrid heuristic algorithm for the precedence constrained production scheduling problem: A mining application. Omega 94, 102046 (2020)
2. Qin, H., Fan, P., Tang, H., Huang, P., Fang, B., Pan, S.: An effective hybrid discrete grey wolf optimizer for the casting production scheduling problem with multi-objective and multi-constraint. Computers & Industrial Engineering 128, 458–476 (2019)
3. Framinan, J.M., Perez-Gonzalez, P., Fernandez-Viagas, V.: Deterministic assembly scheduling problems: A review and classification of concurrent-type scheduling models and solution procedures. European Journal of Operational Research 273(2), 401–417 (2019)
4. Wu, C.-C., Azzouz, A., Chung, I.-H., Lin, W.-C., Ben Said, L.: A two-stage three-machine assembly scheduling problem with deterioration effect. International Journal of Production Research 57(21), 6634–6647 (2019)
5. Li, J.-Q., Song, M.-X., Wang, L., Duan, P.-Y., Han, Y.-Y., Sang, H.-Y., Pan, Q.-K.: Hybrid artificial bee colony algorithm for a parallel batching distributed flow-shop problem with deteriorating jobs. IEEE Transactions on Cybernetics 50(6), 2425–2439 (2019)
6. Ribas, I., Companyx, R., Tort-Martorell, X.: An iterated greedy algorithm for solving the total tardiness parallel blocking flow shop scheduling problem. Expert Systems with Applications 121, 347–361 (2019)
7. Dai, M., Tang, D., Giret, A., Salido, M.A.: Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints. Robotics and Computer-Integrated Manufacturing 59, 143–157 (2019)
8. Ying, K.-C., Lin, S.-W.: Solving no-wait job-shop scheduling problems using a multi-start simulated annealing with bi-directional shift timetabling algorithm. Computers & Industrial Engineering 146, 106615 (2020)
9. Xie, Z., Yang, J., Yang, G., Tan, G.: Dynamic job-shop scheduling algorithm with dynamic set of operation having priority. Chinese Journal of Computers 31(3), 502–508 (2008)
10. Komaki, G., Sheikh, S., Malakooti, B.: Flow shop scheduling problems with assembly operations: a review and new trends. International Journal of Production Research 57(10), 2926–2955 (2019)
11. Khodke, P., Bhoangade, A.: Real-time scheduling in manufacturing system with machining and assembly operations: a state of art. International Journal of Production Research 51(16), 4966–4978 (2013)
12. Xie, Z., Hao, S., Ye, G., Tan, G.: A new algorithm for complex product flexible scheduling with constraint between jobs. Computers & Industrial Engineering 57(3), 766–772 (2009)
13. Zhang, X., Xie, Z., Xin, Y., Yang, J.: Time-selective integrated scheduling algorithm with backtracking adaptation strategy. Expert Systems 36(3), 12305 (2019)
14. Xie, Z., Xin, Y., Yang, J.: Integrated scheduling algorithm based on event-driven by machines’ idle. Journal of Mechanical Engineering 47(11), 139–147 (2011)
15. Xie, Z., Yu, J., Chen, D., Yang, J.: Integrated scheduling algorithm of two workshops based on the principle of
the neighborhood rendering. Journal of Mechanical Engineering 52(1), 149–159 (2016)
16. Zhang, X., Xie, Z., Xin, Y., Yang, J.: Integrated scheduling algorithm of two workshops based on optimal time.
Computer Integrated Manufacturing Systems (9), 1938–1949 (2017)
17. Xie, Z., Guo, H., Su, W., Xin, Y., Yang, J.: Reversal sequence integrated scheduling algorithm of multiple
workshop with multi-procedures ended together. Journal of Jilin University(Engineering and Technology Edition
048(2), 578–587 (2018)
18. Xie, Z., Qi, Y., Yang, J.: Integrated scheduling algorithm with multiple-devices-operation. Journal of
Mechanical Engineering 50(24), 191–200 (2014)
19. Xie, Z., Teng, Y., Yang, J.: Integrated scheduling algorithm with no-wait constraint operation group. Acta
Automatica Sinica 37(3), 371–379 (2011)
20. Zhang, X., Ma, C., Wu, J.: Multi-batch integrated scheduling algorithm based on time-selective. Multimedia
Tools and Applications 78(21), 29999–30010 (2019)
21. Xie, Z., Xin, Y., Zhang, X.: Multi-batch processing integrated scheduling algorithm based on signal driven.
Chinese Journal of Computers 36(4), 818–828 (2013)
22. Wang, F., Zhao, G., Jia, Z., Lu, X., Wang, L.: Assembly job shop scheduling based on feasible solution space
genetic algorithm. Computer Integrated Manufacturing Systems 16(1), 115–120 (2010)
23. Wang, L., Jia, Z., Wang, F., Zhao, G.: Solving complete job shop scheduling problem using genetic algorithm in
feasible domain. Journal of Dalian University of Technology 51(2), 205–209 (2011)
24. Zhao, S., Han, Q., Wang, G.: Product comprehensive scheduling algorithm based on virtual component level
division coding. Computer Integrated Manufacturing Systems 21(9), 2435–2445 (2015)
25. Shi, F., Zhao, S.: Product comprehensive scheduling problems solved by genetic algorithm based on operation
constraint chain coding. China Mechanical Engineering 28(20), 2483–2492 (2017)
26. Lei, Q., Guo, W., Song, Y.: Integrated scheduling algorithm based on an operation relationship matrix table for
three-structured products. International Journal of Production Research 56(16), 5437–5456 (2018)
27. Shi, F., Zhao, S., Meng, Y.: Hybrid algorithm based on improved extended shifting bottleneck procedure and ga
for assembly job shop scheduling problem. International Journal of Production Research 58(9), 2604–2625
(2020)
28. Metawa, N., Hassan, M.K., Elhoseny, M.: Genetic algorithm based model for optimizing bank lending decisions.
Expert Systems with Applications 80, 75–82 (2017)
29. Hosseinabadi, A.A.R., Vahidi, J., Saemi, B., Sangaiah, A.K., Elhoseny, M.: Extended genetic algorithm for
solving open-shop scheduling problem. Soft computing 23(13), 5099–5116 (2019)
30. Xie, Z., Yang, J., Zhou, Y., Zhang, D., Tan, G.: Dynamic critical paths multi-product manufacturing scheduling
algorithm based on operation set. Chinese Journal of Computers 34(2), 406–412 (2011)
31. Xie, Z., Zhang, X., Gao, Y., Xin, Y.: Time-selective integrated scheduling algorithm considering the
compactness of serial processes. Journal of Mechanical Engineering 054(006), 191–202 (2018)

Figures

![Figure 1](image_url) The processing operation tree of a complex product
Figure 2 The process of coding a simple product using the proposed encoding method

Figure 3 The schematic diagram of a chromosome for the product shown in Figure 1

Figure 4 The schematic diagram generated by applying the single-point row vector-based crossover method to generate new individuals
Figure 5. The schematic diagram generated by applying the crossover method based on the multi-point crossover row vector to generate new individuals.

Figure 6. The schematic diagram generated by applying the mutation method based on mutant row vector to generate new individual.

Figure 7. The schematic diagram generated by applying the scramble mutation method based on the mutant column vector to generate a new individual.
Figure 8  The inverse gantt chart generated by applying the designed pre-decoding strategy on the chromosome shown in Figure 3.

Figure 9  The Gantt chart of positive sequence scheduling by applying the conversion strategy to the scheme shown in Figure 8.
Figure 10 The overall flow chart of the designed algorithm

Figure 11 The optimal scheduling scheme obtained by the proposed algorithm for the product shown in Figure 1
Figure 12 The minimum makespan obtained for 100 randomly generated instances

Figure 13 The mean makespan obtained for the instances in Figure 12
Figure 14 The statistical results about ATC of three algorithms for instances in Figures 12 and 13

Figure 15 When $G = 200$, the optimal makespan obtained in each generation by applying the above three algorithms
Figure 16 When $G = 200$, the average makespan obtained in each generation by applying the above three algorithms.