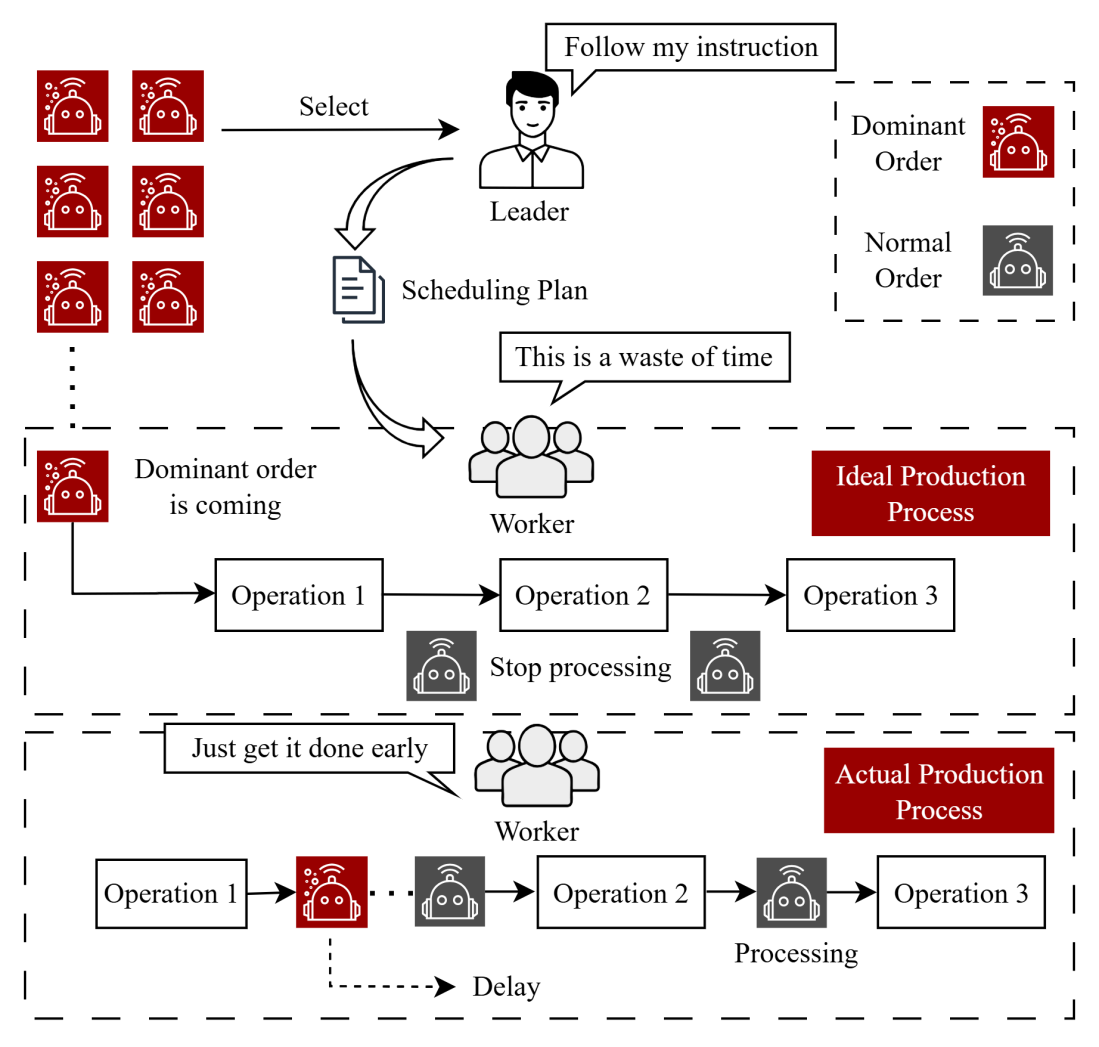
**Abstract**

1. **Introduction**

In the intelligence industrial robot manufacturing factory, there are always some dominant orders at each month, the dominant orders have its release time because the special material preparation. And these orders must be completed by its due time without any delay. It means each dominant order has its time wisdom. So, there are always conflicts between different dominant orders, some of the dominant orders may have the similar time window. How to choose these order and arrange it with the normal orders is a big problem. The normal way is give a green road to the dominant orders when it will come. However, it means some machines need to stop their work and wait for the dominant orders. For the leader of this factory, they know it’s a stupid way, but they don’t know how to arrange the sequence besides this way. So, when the factory receive the dominant orders, the work time at this month will be very long, which make the worker must spend more time on their job. In a short period, the worker thinks this is OK because they know they should work well for this factory. But in a long term, when the worker know they need to work so hard every month, they start to go on strike, choose a quicker way to arrange the production. However, a quicker way don’t means a good way for the leader, because they need to consider each order’s delay plenty and ensure the dominant order should be completed by the due time. But a quicker way just consider how to deduce the makespan. So, it make the leader need to pay more delay plenty, even some of the dominant orders may be delayed too and it has caused the factory a huge losses . Now, the leader find the work don’t process the orders in his instruction. Figure1 shows the problem between leader and worker. Although the leader also punish the worker who don’t follow his instruction, this way doesn’t work because of workers often shift responsibility among themselves and it is difficult to supervise effectively, so it is hard to truly punish a specific worker. In fact, the leader just want to earn more money, if there is a way can help him to ensure he can pay little delay plenty, the dominant orders completed on time and deduce the makespan, he will choose it.

The contributions of this paper are as follow:

1. Developed a problem that conflict between the leader and worker during the production.
2. Set a threshold bad-scenario set model for this problem
3. Design a combination algorithm
4. Develop a improved way for muti-objective problem. A robust strategy is proposed to address the difficulty in selecting weight coefficients when solving multi-objective problems using weight coefficients.



**Fig. 1.** The problem between leader and worker

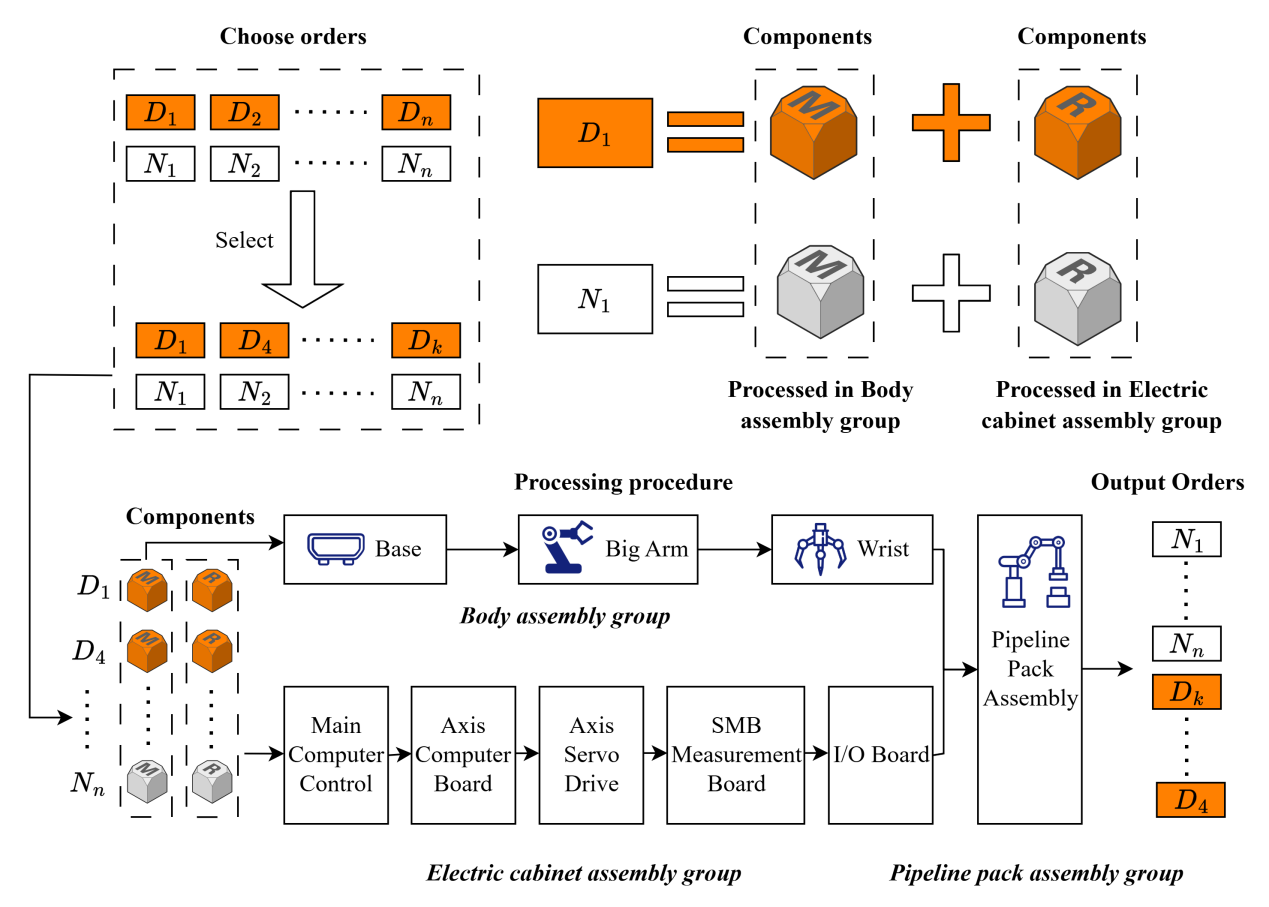
1. **Literature Review**
2. **An improved threshold bad-scenario set model**

## Problem description

The problem is how to yield a scheduling plan and then help the leader of factory make more profit and help the worker spend less time. If we want to solve this problem, we need to know the processing procedure firstly. As shown in Figure2, the factory will receive dominant orders and normal orders each month. Here, we see each order as a job. However, there are time conflict among different dominant jobs, the factory can not process all these dominant jobs in it’s time window. So, the factory need to select these dominant jobs. For normal jobs, these jobs are allowed delay, all these jobs will be selected. Finally, after the selecting of jobs, there are dominant jobs and normal jobs. When the leader give instruction to the worker what they need to process, then the worker process it.

In Figure2, there is a production line about intelligence industrial robot, which consist of 3 groups: Body assembly group, Electric cabinet assembly group, Pipeline pack assembly group. One job is consists of two components which is be processed in body assembly group and electric cabinet assembly group, respectively. While body assembly group has three operations (“Base”, “Big arm” and “Wrist”) and electric cabinet assembly group has five operations (“Main computer control”, “Axis computer board”, “Axis servo drive”, “SMB measurement board” and “I/O board”), so the completion time of component in electric cabinet assembly group is greater than in body assembly group in most case. The body assembly group and the electric cabinet assembly group are mutually independent and do not affect each other. When the components have been assembled in these two assembly groups, they will go to the pipeline pack assembly group and assemble for a complete job. As the last stage in this production line, the pipeline pack assembly just has one operation. In addition, every job has its own components, so components from different jobs can not be assembled to a complete job. The original objectives are to maximize the total profit and minimize the makespan. The problem is how to choose the dominant orders and make a produce sequence that can achieve these objectives.

There are some thing we should know, the normal orders’ delay cost is not so high, because in the actual business, for the orders that not too important, the manufacturer can choose to communicate with the buyer then change the due date and it cost nothing. Some of the normal buyers don’t care the due date, what they care is that they can get the products. So, in our model, we shall give the delay cost of the normal orders a reasonable value. However, for the dominant order buyer, they can stand the delay, because they think they are the VIP buyer if the manufacturer receive their jobs. So, for dominant orders, the manufacturer just have two options, agree it and do it, or refuse this job.



**Fig. 2.** Order selection and production process

Base on the description of our problem, we want to set a weight for these two objectives. We use the and to balance total profit and makespan. Setting the as a new objective, while is a constant that magnify the makespan, because the value of total profit maybe million, but the value of makespan maybe just hundred, if we don’t magnify the makespan, it will be difficult to set the weights in a reasonable range, whatever we set the weight coefficients in range of [0.1, 0.9], the makespan can not yield big influence to the total profit. However, although we set weight coefficients to solve these two objectives, the leaders don’t know how to set the weights in reality. If the weight coefficients they set place too much emphasis on total profit, workers will not follow the plan, while place too much emphasis on makespan, they will also lose a lot of profit. So, it’s very hard to handle this problem, previous literature has also rarely explored how to address this issue. We considered to solve it with robust model and proposed an improved threshold bad-scenario set model for this problem, set the possible subset as the scenarios, set the weight sum of previous history total profit and makespan as the value of . Then we shall get a solution that can fit any scenarios and get a not bad result.

Based on the above statements, we set an improved threshold bad-scenario set model , set the penalties on this model as the objective. And the specific parameters and math symbol information are as follows:

|  |  |
| --- | --- |
| Indices | Description |
|  | The index of the operations on body assembly group, |
|  | The index of the operations on electric cabinet assembly group, } |
|  | The collection of dominant job |
|  | The index of jobs, which include the normal job and the dominant job, |
|  | The set of all possible scenarios |
|  | The number of all possible scenarios of Λ |
|  | The index of scenario, () which contains and , |
|  | The set of feasible solution, |
| Parameters | Description |
|  | Processing time of the job in the operation of the body assembly group, |
|  | Processing time of the order in the operation of the electric cabinet assembly group, |
|  | Processing time of the job in the pipeline pack assembly group, |
|  | A large number |
|  | A constant used to amplify the makespan |
|  | Income of the job, |
|  | Release time of the job, , if the job is the normal job, then |
|  | Deadline of the job, |
|  | The delay penalty of the job, |
|  | The average cost of the job, |
|  | The threshold value |
| Variables | Description |
|  | Starting time of the job assembled in the operation of the body assembly group, |
|  | Starting time of the job assembled in the operation of the electric cabinet assembly group, |
|  | Starting time of the job assembled in the pipeline pack assembly group, |
|  | Completion time of the job assembled in the operation of the body assembly group, |
|  | Completion time of the job assembled in the operation of the electric cabinet assembly group, |
|  | Completion time of the job assembled in the pipeline pack assembly group, |
|  | The delay time of the job |
|  | Makespan |
|  | Total profit |
|  | The function of decision variable and scenario, |
|  | The set of the bad scenario based on the threshold value *T* under the feasible solution , |
|  | The penalties on TBS |
| Decision variables | Description |
|  | If the job is chosen, then =1;otherwise =0, . If the job is the normal order, then =1 |
|  | If the job is processed before the job on the body assembly group, then =1; otherwise =0, |
|  | If the job is processed before the job on the electric cabinet assembly group, then =1; otherwise =0, |
|  | If the job is processed before the job on the pipeline pack assembly group, then =1; otherwise =0, |

The mixed-integer linear programming model（MILP）is as below：

Objective:

|  |  |
| --- | --- |
|  | (1) |

s.t:

|  |  |  |
| --- | --- | --- |
|  |  | (2) |
|  |  | (3) |
|  |  | (4) |
|  |  | (5) |
|  |  | (6) |
|  |  | (7) |
|  |  | (8) |
|  |  | (9) |
|  |  | (10) |
|  |  | (11) |
|  |  | (12) |
|  |  | (13) |
|  |  | (14) |
|  |  | (15) |
|  |  | (16) |
|  |  | (17) |
|  |  | (18) |
|  |  | (19) |
|  |  | (20) |
|  | | (21) |
|  | | (22) |
|  | | (23) |

Formula (1) is the objective function of the model, aiming to minimize the penalties on TBS. Constraints (2) and (3) specify that the completion time of each operation for an order's components on the main body and electrical cabinet assembly groups must be greater than or equal to the sum of its start time and processing time. Constraints (4) and (5) stipulate that the start time of the subsequent operation for an order's components on the main body and electrical cabinet assembly groups must be greater than or equal to the completion time of the preceding operation. Constraints (6) and (7) state that the start time of the next component on an operation within the body and electrical cabinet assembly groups must be greater than or equal to the completion time of the preceding component. This ensures processing of the next component only commences after the previous one is fully processed. Constraints (8) and (9) specify that the start time of an order on the pipeline assembly must be greater than or equal to the completion time of the final operation for that order's components on the body and electrical cabinet assembly groups. Constraint (10) specifies that the completion time of an order on the pipeline packaging assembly group must be greater than or equal to the sum of its start time and processing time, ensuring the order undergoes complete assembly operations. Constraint (11) requires that the completion time of the subsequent order on the pipeline packaging assembly group must be greater than or equal to the start time of the preceding order. Constraint (12) indicates that special orders yield significant profit, meaning manufacturers still profit even if overtime is required to complete such orders. Constraints (13)-(15) impose order constraints, stipulating that an order's processing start time must occur after its release time and that testing must be completed within the delivery deadline. Constraint (16) defines the relationship between special order selection and order position sequencing. Constraints (17)-(22) constrain decision variables, ensuring each order occupies only one position and each position accommodates only one order. Constraint (23) requires the manufacturing span to exceed the completion time of each workpiece on the pipeline packaging assembly. Constraints (24)–(27) introduce new variables U and V, where U represents overtime hours and V represents the actual time if the general production cycle is not exceeded. The primary purpose of setting U and V is to linearize the objective function. Constraint (28) represents the objective function expression. Except for constraints (13) to (16), which apply specifically to special jobs, all other constraints apply to both special and regular jobs. Constraint (29) expresses the objective function.

## Case study

We proposed an improved threshold bad-scenario set model for the manufacturing factory, for easy understanding to this problem, we set a small case. In this case, there are 4 normal jobs and 4 dominant jobs. We note the normal job as and note the dominant job as . And the problem is to minimize the threshold penalty. Table 1 shows the information of different jobs. In this table, the unit of time is hour and the unit of money is Yuan. For , the processing time on the body assembly group is {5, 3, 5}, which means the time of different operation. There are three operation on the body assembly group, so there are three values in this Curly brackets. The symbol “-” means nothing, because the dominant jobs can not be delayed and the profit of normal jobs can not be calculated without job sequence.

**Table 1** Information of dominant jobs and normal jobs

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Job |  |  |  |  |  |  |  |  | Profit |
|  | {5, 3, 5} | {1, 3, 2, 4, 5} | 3 | 5 | 25 | 39000 | - | 500 | 23500 |
|  | {3, 5, 3} | {2, 3, 2, 3, 2} | 4 | 20 | 40 | 30000 | - | 442 | 18066 |
|  | {2, 3, 2} | {1, 3, 2, 1, 3} | 5 | 25 | 43 | 27000 | - | 403 | 18134 |
|  | {5, 3, 4} | {4, 4, 1, 2, 3} | 6 | 20 | 41 | 35000 | - | 415 | 26700 |
|  | {5, 2, 4} | {2, 4, 2, 4, 1} | 3 | 0 | 25 | 13500 | 20 | 318 | - |
|  | {4, 3, 4} | {5, 1, 2, 3, 1} | 2 | 0 | 20 | 12400 | 20 | 275 | - |
|  | {2, 5, 3} | {2, 4, 2, 1, 4} | 3 | 0 | 15 | 13300 | 25 | 294 | - |
|  | {5, 4, 2} | {4, 3, 1, 1, 3} | 2 | 0 | 25 | 15900 | 20 | 272 | - |

When the dominant jobs are coming, the leader need to choose these jobs. For the leader of the manufacturing factory, it’s impossible to abandon any jobs for the early rest time of workers. They want their workers to spend more time on their job and then they can receive more jobs. So, the only reason that the leader will abandon some jobs is they can’t produce these jobs at the same time. They need to choose the jobs that will yield him greater profits.

In this case, there are 4 normal jobs and 4 dominant jobs, the normal jobs are allowed to delay and don’t have release time, so the leader need not to choose the normal jobs, all of them can be processed. For the dominant jobs, there are two jobs have the similar time window(One job which has release time and due time), which are and . So the leader should consider how to choose the dominant job and then he can get greater profit. The profit of is greater than . Actually, the leader will choose in his production plan, because there are lots of dominant jobs each month and they don’t have lots of time to consider which one will take more profit, the easy and effective way is to choose the dominant job with the greater profit if there are more than two jobs at the similar time window. However, this is not the way that can achieve the biggest profit. We can calculate the profit if the leader choose . In such situation, will not be selected if the leader choose , the total profits of dominant job will be 50200. In another situation, the leader choose rather than , the can be processed, the total profit of dominant jobs will be 59700. We can find the profit that the leader choose is greater than choose . So, the best choose plan is . In the reality, there are not only 4 dominant jobs each month, so we proposed a choose model to solve the problem of choosing dominant job. And the content of choose model shall be introduced in the chapter 4.

After the choose of dominant jobs, the final production plan is .The original objective is to maximize the total profit and minimize the makespan, a commonly used method is set weight for multi-objective, such as:

However, it’s very hard to set the weights for multi-objective, if you set a weight that is benefit to leader, the workers will not obey the instruction of the leader too, and then it will yield a terrible result. If you set a weight that is benefit to the worker, then the workers will obey your instruction, but the leader will lose the profit. The leader don’t know how to set a good weight too, and they don’t have too much time and source to test which weight is better. Based on this situation, we propose an improved threshold bad-scenario set model to fit the problem how to set the weight, we just need a weight range and then can yield a solution that can fit any weight and get not bad result. The objective of this model is minimize the . In this case, we set the weights in range {0.3, 0.4, 0.5, 0.6, 0.7} and set . In this case, we solve this problem by Gurobi and get the best , then set it as the value of T. By calculating of Gurobi solver, T = 21865. We compared two methods, the first method is arrange the produce sequence by the scheduling rules of factory(**SRF**), which are three points: Fist come and then first processed, minimize lead time sorting rule and dominant jobs take absolute priority. The second method is arrange the produce sequence by Gurobi, because there are just 7 jobs, it’s easy to get the best result by Gurobi. For the judgement of bad scenarios, set the If , then scenario is the bad scenario. Based on these two methods, we calculated the makespan and get the final sequences of two methods, shows in Figure 3. In these gantt figure, (a) shows the produce sequence of different job under the SRF, the final output sequence is . It’s easy to find the dominant job is very special, which don’t contain any waiting time and be processed immediately after release. And when it coming, all the processing jobs have been stop. This SRF is easy to understand but with bad results. In this scenario, the makespan is 43, which is greater than Gurobi’s result. Although the difference is just 2, but in the scenario with large number of jobs, the difference will be very big. And this SRF will make more job’s delay. (b) shows the produce sequence of different job under Gurobi, the final output sequence is . In this produce sequence, the dominant job is allowed to keep waiting time, but it’s not delayed too. The makespan is 41, smaller than the SRF’s result, and its’ delay penalties are smaller.

|  |  |
| --- | --- |
|  |  |
| 1. Gantt under SRF | 1. Gantt under Gurobi |

**Fig. 3.** Gantt of case

**Table 2** Running results of case

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ***SRF*** | | | ***Gurobi*** | | |
|  |  |  |  |  |  |
|  | -4801 | 26666 | 41799 | **-3281** | **25146** | **37719** |
|  | 7932 | 13933 | **9292** | **12573** |
|  | 20665 | 1200 | **21865** | **0** |
|  | 33398 | 0 | **34438** | 0 |
|  | 46131 | 0 | **47011** | 0 |

Tip:

Meanwhile, we have calculated the threshold bad-scenario penalty of two methods, the results of two methods are show in Table 2. is the weighted sum of total profit and makespan. is the threshold bad-scenario penalty. According to the value of , we can find the number of bad-scenario of SRF is 3, while the number of Gurobi is 2. And no matter of what scenario, the value of of Gurobi is greater than SRF. These show the perform of Gurobi is better than SRF and SRF is not the best scheduling rule. By comparing the , the value of SRF is 41799, while Gurobi is 37719, the difference is 4080. Although the number of jobs is just 7, the difference of under 5 scenarios has already reached 4080, which means SRF lacks the risk-bearing capacity, with the increasing of the job quantity and scenarios, SRF will perform worse. So, it’s imperative to propose a strategy to alter this situation, and the model we present can adapt to various scenarios, while the proposed algorithm demonstrates outstanding performance.

1. **Gurobi and Choose model**

In this part, we shall introduce a XX algorithm, and this algorithm is XX

We proposed a algorithm to solve the firstly, and then we can get the job information after the choose operation. In the choose operation, we only choose the dominant job, for the normal job, we set the to be 1, which means all of the normal job will be chose. Then we can study how to arrange the sequence of jobs.

1. Choose the dominant job

In our model, we have set four decision variables, which are , , , . If we solve this model directly, it’s difficult to get the solution. It may spend a lot of time and not get the good results. We find the decision variables , , is effected by the decision variable . So, if we get the firstly, this problem will be easier. The function of is to choose the dominant job. Mapping this point to real life, because the dominant jobs have their own time windows and must be submitted within their due time, so there are time conflicts between dominant jobs, the factory should choose the dominant job that can be processed in its’ cycle time and make a dominant job collection to ensure they can get enough profit. This is why we need to set the decision variable . In order to choose the dominant job quickly, we separated the dominant job. Although just consider the dominant job, it’s very hard to get the . So, it’s very important to simplified the original model. Set the choose progress as a single machine scheduling problem, the processing time is set as the sum of the processing time of pipeline packing assembly and the maximize value between the sum of processing time of electric cabinet assembly and the sum of processing time of body assembly. However, this setting is not enough, there are at least 9 operations in the production, it don’t consider the waiting time between different operation that set the sum of processing time at each operation as the new processing time in single machine.

The release time and due time of dominant job is consistent with the original model. Setting the and to be the start time and completion time. Setting the as the decision variable, when the dominant job is processed before the dominant job, then , otherwise .

Then, this problem can be described as . We need to proof this problem is a NP-hard problem, the proof is as below:

**Theorem 1.***The single machine scheduling problem*

*where each job has release time , processing time , due time and profit , and only accepted jobs must finish within their time windows is NP-hard, even when all processing times and profits are integers.*

**Proof.** We prove NP-hardness by a reduction from the 0/1 Knapsack problem, which is NP-complete.

**Knapsack problem:** Given a set of items, each with a weight and value , a knapsack capacity , and a target value the decision version asks whether there exists a subset such that , and .

We construct an instance of the scheduling problem from an arbitrary instance of Knapsack as follows:

* We create jobs, each corresponding to an item.
* For each job :
* That is all jobs are available at time 0, all share a common due date , and job requires processing time equal to its weight and yields profit equal to its value.
* The objective is to select a subset of jobs such that all selected jobs can be processed before and total profit is maximized.
* We set a threshold value .
* The decision version asks whether there exists a feasible subset and a schedule such that all satisfy and *.*

The construction is clearly polynomial-time.

We now prove equivalence between the two problems.

**Claim 1 ( Knapsack ⇒ Scheduling )**

If the Knapsack instance has a feasible subset such that , and , then the constructed scheduling instance is a YES-instance.

Proof of Claim 1.

Schedule all jobs in consecutively starting at time 0 with no idle time. Then . Hence every job in finishes by its due date . The total profit is *==Y.* Therefore, the scheduling instance admits a feasible schedule satisfying the decision bound.

**Claim 2 ( Scheduling ⇒ Knapsack )**

If the constructed scheduling instance is a YES-instance, then the corresponding Knapsack instance is also a YES-instance.

Proof of Claim 2.

Let be the accepted jobs in a feasible schedule such that and all . Because and the machine is single, we may left-justify the schedule so that no idle time appears before . Then . Feasibility implies . Moreover, Thus is a feasible subset for the original Knapsack instance, satisfying both the capacity and target-value constraints.

Both directions hold, and the reduction is polynomial-time. Therefore, the decision version of is NP-hard.

Based on theorem 1, we know there is no exact algorithm for this problem within polynomial time. We used Gurobi to solve it, so we need to set a tight M, then the Gurobi can get the results easier. In this model, we set all jobs’ processing time in the product line as the big M. The equation of M is as below:

The choose model is as below:

Objective:

|  |  |
| --- | --- |
|  | (1) |

s.t:

|  |  |  |
| --- | --- | --- |
|  |  | (2) |
|  |  | (3) |
|  |  | (4) |
|  |  | (5) |
|  |  | (6) |
|  |  | (7) |
|  |  | (8) |
|  |  | (9) |

Formula (1) is the objective which is to make the maximize total profits for the dominant jobs. Constraint (2) stands the start time of the dominant job is more than its release time. Constraint (3) stands the complete time of the dominant job is more than its start time. Constraint (4) stands the start time of the dominant job is more than its previous dominant job’s complete time. Constraint (5) stands the dominant job must be completed within its due time. Constraints (6)-(8) is subject to the decision variables and . Constraint (9) is the equation of the total profit.

For solving this choose model, we not only used the Gurobi solver but also compared some meta-heuristic algorithms. The compared algorithms are GA, GWO, SA, WHO. For Gurobi, it’s wildly knew that Gurobi can not solve the problem contain big amount example, so we set a time limit of 20 minutes. By the experience, we find the results show in the Table 3

**Table 3** The results of different algorithms



In Table 3, The number values in this table stand for the total profit. If the value is big, which means the algorithm is good. No matter how much the dominant jobs is, we can find the results of Gurobi is more than other meta-heuristic algorithms, which means Gurobi can get the bigger profit among all algorithms. Then, we can use Gurobi to choose the dominant jobs.

1. **A algorithm with DQN net**

Encoding and decoding

Flow chart

结果比较

参数来源

实验参数调整

比较XX

管理启示：

结论：