Report

Production scheduling (Topic 6)

Content

1.Presentation of subject

a. Introduction

Production scheduling includes setting up production schedules, coordinating and assigning labor to each person, group of people, and machine, and arranging the work order at each workplace to ensure production completion on schedule.

It is an essential tool for manufacturing and engineering, which can significantly impact a process's productivity. In manufacturing, the purpose of scheduling is to keep the due dates of customers and then minimize the production time and costs by telling a production facility when to make, with which staff, and on which equipment. Production scheduling aims to maximize the efficiency of the operation, utilize the maximum resources available and reduce costs.

b. Description

- An agricultural food-providing company needs to produce N products 1,2,3,..., N
- The proposed cost of producing 1 unit of product i is c(i)
- The area of field needed to produce 1 unit of products i is a(i)
- The profit obtained from producing 1 unit of product i is f(i)
- Know that m(i) is the minimum number of units of product i needed to produce when deciding to make that product
- The total area of the field is A

Goal: the profit is the maximum and the total cost of production is less than or equal to a value of C

c. Input format

Input

- Line 1: N,A,C
- Line 2: c(1),c(2),….,c(N)
- Line 3: a(1),a(2),….,a(N)
- Line 4: f(1),f(2),….,f(N)
- Line 5: m(1),m(2),….,m(N)

d.Output format

Output

- The number of each product decided to make
- The total profit

2.Problem Modeling

a.Variables: Xi (i = 1,2,3,...N)

b.Constraints

- i=1Nai*Xi =<A
- Xi>mi(i =1,…,N) or Xi=0
- i=1Nci*xi =< C

c.Objective function

Max i=1Nfi*xi

3.Algorithms

a. Exhaustive search (ES)

This is the most basic method. It will search for every possible solution and then compare it with the variable saved previously. If the solution in this loop is bigger than the saved variable, we update it and continue doing this until the end.

As a result, the running time of this method is very long O(2^N)

b. Dynamic programming (DP)

For this algorithm, it has been improved a lot compared to ES. It is still going through all feasible solutions and comparing all to give the optimal one. The difference between it and ES is that it will keep the result of decision variables throughout its loop. So when it sees that the loop uses that result, it does not have to run again to take the result which has been run.

```
return max(f[N-1]*V[N-1] + DP(N-1, m, V), DP(N, M, n))
```

c. Constraint programming (CP)

This algorithm will apply all the conditions of this project which have been modeled to the mathematics formula. To help with this method, we use the OR-tools library for implementation. We use constraint optimization in this library for our code.

OR-tools is a powerful tool, giving us the most optimal solution in less than a second.

d. Mix-Integer programming (MIP)

This is the best method we have so far. In this algorithm, we not only use the conditions which have been modeled in the above section but also use geometry to apply a mix-integer problem. Our problem has Semi-Integer variables, so it is tough to construct in others, but it is the best way in this algorithm.

We use the Gurobi library to help us with this method, a potent tool that, in the end, gives the optimal answer in a shorter time than CP using OR-tools

e. Heuristic 1

First,we arrange the list of index of profit of products in descending order :

```
Profit_index=[i for i in range(N)]
SortedProfit_index=sorted(Profit_index,
key=lambda i:f[i],reverse=True)
MaxProfit=0
Answer=[0]*N
```

Then we loop through the SortedProfit_index list and if the index i of the product which satisfies U=min(C//c[i],A//a[i]) and U>m[i], we will produce U units of product i-th and update the MaxProfit,C,A,Answer ,respectively:

```
for i in SortedProfit_index:
  U=min(C//c[i],A//a[i])
  if U>=m[i]:
    #Update
    MaxProfit+=U*f[i]
    C=C-c[i]*U
    A=A-a[i]*U
    Answer[i]+=U
```

f. Heuristic 2.0,2.1,2.2

We have implemented 3 kinds of heuristic based on an idea:

Find the index i that satisfies both these conditions:

- U=min(C//c[i],A//a[i])
- U>=m[i]
- U*f[i] max

a.Heuristic 2.0

With found index i,we will produce U units of product i-th and update C,A,Answer,MaxProfit ,respectively(similarly in Heuristic 1).If index i does not exist,we break the loop

b.Heuristic 2.1

With found index i,we will produce m[i] units of product i-th and update C,A,Answer,MaxProfit ,respectively(similarly in Heuristic 1) and also denote m[i]=1.If index i does not exist,we break the loop

c.Heuristic 2.2

With found index i,we will produce (U+m[i])//2 units of product i-th and update C,A,Answer,MaxProfit ,respectively(similarly in Heuristic 1).If index i does not exist,we break the loop

4.Implementation

When solving our project we have some Problems/Issues/Difficulties occurring during the execution of the project and the solutions:

Using library, we try to use more than one library so that took us a lot of time doing research of how to use the library because with each tools have different ways of coding and modeling, especially our problem has Semi-Integer variables so it very hard to model it from a mathematics formula to code or something computer can understand and implement.

5.Comparison

Testing condition and methodology

• ROG Zephyrus G14:

CPU model: AMD Ryzen 9 5900HS with Radeon Graphics, instruction set [SSE2|AVX|AVX2]

Thread count: 8 physical cores, 16 logical processors, using up to 16 threads

• All algorithms are run on Jupyter Notebook 6.4.5 with Python 3.10.9

• Version of libraries:

OR-tools: ortools version 9.5.2237 (win64)

Gurobi: Gurobi Optimizer version 10.0.1 build v10.0.1rc0 (win64)

• Only start the algorithm when OS is stabilized (CPU usage is less than 5%, RAM usage 6.7 GB, Disk usage less than 2%). Airplane mode is turned on to prevent random disruption.

• All test files are created prior to testing to reduce randomness in I/O activities

• Step of each run:

1. Restart the computer

2. Initialize Jupyter Notebook environment, open Task Manager, wait about 5 minutes(for the OS to stabilize)

3. Execute algorithm on all 30 tests and some bigger size tests.

a. Time complexity

Regarding time complexity, we experimented with 6 sizes of test (each size has 5 random tests). The results are shown for all our algorithms: Exhaustive Search (ES), Dynamic Programming (DP), Constraint Programming (CP), Mix-Integer Programming (MIP) and 4 kinds of Heuristic (He1.0, He2.0, He2.1, He2.2).

Table 1: The comparison of the experimental results time among our methods (unit: second)

Algorithms / Size	ES	DP	CP	MIP	He1.0	He2.0	He2.1	He2.2
N=5	24.58	0.184	0.013	0.002	<0.001	<0.001	<0.001	<0.001
N=10(100<A,C<1000)	0.023	0.001	0.009	0.001	<0.001	<0.001	<0.001	<0.001
N=10(1000<A,C<10000)	>1hour	405.01	0.022	0.012	<0.001	<0.001	<0.001	<0.001
N=100	>1hour	>1hour	0.021	0.007	<0.001	<0.001	0.242	0.001
N=1000(10^6<A,C<10^7)	>1hour	>1hour	0.136	0.015	0.001	0.001	15.21	0.004
N=1000(10^11<A,C<10^12)	>1hour	>1hour	0.122	0.013	0.001	0.001	>1hour	0.016

Did the algorithms find the solution ?

For small size test, N = 5, 10 and A,C still not too large, all algorithms can give answer instantly. As we expected when come to bigger size, ES is the slowest algorithm and then DP when meet N = 100 also run very long time. CP and MIP is the 2 stable methods through out every test with less than a second, but we can all witness that MIP still slightly better than CP in all the time. For all the Heuristic algorithms, He1.0 and He2.0 show that it run fastest, even more quickly than CP and MIP. He2.2 when come to larger and lager size test, it becomes slower a bit while He2.1 come to the largest size can not solve in 1 hour anymore. But all algorithms surely can give solutions even the slowest if have enough time.

Table 2: Comparison of the experimental accuracy among our heuristic methods

(calculated by Profit of solution/Actual maximum profit*100%)

Heuristics / Size	He 1.0	He 2.0	He 2.1	He 2.2
N=5	76,55%	96,39%	100,00%	92,12%
N=20	45,56%	84,97%	96,25%	91,90%
N=100	32,19%	89,14%	98,26%	97,81%
N=1000	10,05%	88,21%	98,92%	99,00%
N=5000	3,81%	86,17%	TOO MUCH TIME TO HANDLE	98,38%
N=10000	2,47%	82,39%		94,56%

Did the algorithms find the accurate solution ?

In terms of accuracy, we experimented with 6 sizes of test(for each size have 5 random tests). The 4 first algorithms is exact method so we do not have to check it's accuracy but only for Heuristics algorithms .The results are shown for our 4 heuristic algorithms (He1.0,He 2.0,He2.1,He 2.2)

As we can see, He 1.0 shows that it has the lowest accuracy, and its accuracy tends lower as the test size becomes larger (76,55% at N=5 and only 2,47% at N=1000). He 2.0 has acceptable accuracy (always higher than 80%, even when the test size is N=10000). With small-size tests, He 2.0 is slightly better than He 2.2, while with bigger test sizes, He 2.2 has much better accuracy solutions than He 2.0 has (about 10%). The table shows that He 2.1 almost has the exact profit compared to the actual maximum profit with small-size tests. However, when the test size becomes more significant, it takes too much time to handle. However, with some test cases, we have completed the running process, which He2.1 also gives pretty good solutions with an accuracy of about 98%.

In our experiment, Three heuristic methods 1, 2.0, 2.1 sometimes give the exact solutions as the maximum solution. Vice versa, we did not find a case where He2.2 has the exact solution compared to the actual maximum solution.

6.Conclusion

In our project, we compare the optimality, we used 8 different methods for solving the problem given. Of the eight, MIP and CP are the most efficient algorithms because of their optimal solution in a very short time. For small-size tests (N = 5, N =10), all the algorithms can solve the problem in an acceptable time. For the 4 exact methods, all give the optimal solution, so we only compare with time: MIP > CP > DP > ES. For the 4 Heuristic algorithms, we compare both time and optimality. For the time running, He1.0 ~ He2.0 > He2.2 > He2.1, and for the accurate optimization result, He2.1 > He2.2 > He2.0 > He1.0, so the running time has some influence on the accuracy. Combining these two factors to give a better algorithm, we conclude He2.2 > He2.0 > He2.1 > He1.0. Although He 2.1 has better accuracy than all the others 3 but when it comes to large-size tests, it can not be done in an acceptable time(> 1 hour)

7.Task Assignment

Task	Executor
Doing ES, DP, CP, MIP algorithms	Minh
Doing 4 Heuristics algorithms	Cuong
Generating test cases	Cuong
Analyzing Time complexity	Minh
Analyzing Accuracy	Cuong
Writing Report	Minh, Cuong
Making Slides	Long

Contributors (Group 17)

1. Nguyen Minh Cuong 20210140, email:cuong.nm210140@sis.hust.edu.vn

2. Nguyen Viet Minh 20214917, email: minh.nv214917@sis.hust.edu.vn

3. Nguyen Tuan Long 20214963, email: Long.nt214963@sis.hust.edu.vn