An Optimization Algorithm of Slab-Design Based on Bipartite Graph and Linear Programming

Ying-xin CHEN, Xian-hui LIU*, Wei-dong ZHAO and Yu-fei CHEN
College of Electronics and Information Engineering of Tongji University, China
*Corresponding author

Keywords: Slab-design, Combination optimization, Bipartite graph, Linear programming.

Abstract. This paper shows a novel optimization algorithm based on bipartite graph and linear programming for the self-produced slab-design problem with continuous weight and color-constrained. Firstly, we reduce the number of slabs through bipartite graph maximum weight matching. The second step uses linear programming to reduce residual material. Finally, the experimental results show that the algorithm will lead to a good slab-design scheme in a short time, which will provide decision support for the actual scheduling production.

Introduction

In the steel industry, the planner needs to design a set of steel slabs to meet all received orders with some constraints, which is called the slab-design problem. Usually, the problem is divided into two kinds: Outsourcing slab means that steel enterprises purchase a group of slabs of different types according to the orders, or the steel enterprises preproduce a group of slabs to match the follow-up orders in a make-to-stock way; Self-produced slab is that steel enterprises independently arrange a set of slabs to meet orders. They can design the size and quality of the slabs by themselves. This way is more flexible than the former, resulting in less residual material. This paper focus on the self-produced slab.

Through the abstraction of multiple-knapsack problem, Dawande proved that the inventory matching problem with color constraint is an NP-Hard problem and proposed an approximation algorithm [1, 2] according to the model. In addition, Dawande et al. first put forward a slab-design model in the literature, which has variable contract quantity, variable slab weight and color constraint. And they proposed a packing algorithm [3, 4] based on map matching, order prioritization and First-fit strategy. Jayant et al. studied the inventory matching problem of multiple types of slabs, and proposed an algorithm based on bipartite graph matching and network maximum flow [5]. Then, for the slab inventory matching problem with color constraint, some scholars have designed an algorithm based on column generation [6, 7]. Xi Yang et al. proposed a two-stage algorithm [8]; Zhang Yingjie and others applied a heuristic algorithm [9] based on contract-slab matrix calculation to the problem of variable slab weight.

The self-produced slab-design problem is flexible. Since the number and type of slabs are uncertain, it cannot be modeled by traditional linear programming or integer hybrid programming. Some of the methods mentioned above cannot be applied to this situation. Some algorithms also have room for improvement in terms of optimization and time performance. Given this situation, this paper designs an algorithm based on bipartite graph maximum weight matching [10] and linear programming [11] with the goal of minimizing the number of slabs and the residual rate. The experimental results show that both time and optimization of the algorithm have a good performance.

Problem Model

Slab-design is a combination optimization problem that minimizes the number of slabs and minimizes the residual material under the premise of satisfying certain constraints. The constraints are mainly divided into two aspects. From the point of view of the order, each order has a minimum
weight and a maximum weight that the customer will accept at delivery. From the perspective of the slab, the weight allocated on each slab should not exceed the maximum weight of slabs. Besides, the weight assigned to each slab of each order shall be an integral multiple of the legal unit weight (U) of the order. The unit weight will be acceptable if it is in the interval \([U_{\text{min}}, U_{\text{max}}]\) of the order. In addition, each order has a color attribute that corresponds to the processing path of the order. Different orders may have the same color, that is, orders share the same subsequent processing path, which means these orders can be assigned to the same slab. Before the start of the follow-up process, the steel mill can cut the slab and different post-processing paths can be applied for the divided slabs. However this part is time consuming and often becomes a bottleneck in the production process. Therefore, we stipulate that a maximum of two orders of different colors can be allocated on one slab, that is, the maximum number of cuts of a slab is one. The objectives and constraints of the problem are formalized below. The meaning of each symbol is shown in Table 1.

Objectives:

Minimize the number of slabs

\[
\text{Minimize } \sum_{i=1}^{NUM} X_i
\]  

(1)

Minimize residual material

\[
\text{Minimize } \sum_{i=1}^{NUM} \left(S_i \times X_i - \sum_{j=1}^{N} W^i_j\right)
\]  

(2)

\[
NUM = \left[\left(\sum_{j=1}^{N} O^j_{\text{min}} \times \text{PS}\right) \times (1 + \text{PS}) \div S_{\text{min}}\right]
\]  

(3)

The value of the slab preset number (NUM) can be estimated by the Eq. 3. PS is the empirical residual rate, generally about 10%.

| symbol | meaning |
|-------|---------|
| N     | Total number of orders |
| NUM   | Preset number of slabs |
| O^j_{\text{min}} O^j_{\text{max}} | Minimum weight and maximum weight of order j |
| S_i   | Weight of Slab i |
| S_{\text{min}} S_{\text{max}} | Minimum weight and maximum weight of slabs |
| U^j_{\text{min}} U^j_{\text{max}} | Unit number and unit weight of order j assigned to slab i |
| U^i_{\text{min}} U^i_{\text{max}} | Minimum unit weight and maximum unit weight of order j |
| c(j)  | Color of order j |
| C_i   | Color set of orders assigned to slab i |
| | Number of C_i |
| dq_j  | designed quantity for order j |
| W^i_j | Weight of order j assigned to order i |
| Y^j_i | 1 if slab j is assigned to order i; 0 otherwise |
| X_i   | 1 if slab i is assigned to some order(s); 0 otherwise |

Constraints:

\[
0 \leq \sum_{j=1}^{N} W^j \leq S^i \times X^i \leq S_{\text{max}}
\]  

(4)
The constraint Eq. 4 indicates that the total weight distributed on each slab should not exceed the maximum weight of slabs, and the weight can be calculated by Eq. 8. The constraint Eq. 5 indicates that the assigned weight of each order on each slab should be an integral multiple of its legal unit weight, and the legal range of the unit weight is as shown in Eq. 14. Constraint Eq. 6 shows the designed weight $d_{q_j}$ of the order $j$ shall be within the acceptance interval of the order, where $d_{q_j}$ can be calculated by Eq. 9. The constraint Eq. 7 means that the color number of each slab should not exceed 2, and the color set of slabs can be calculated by Eq. 10.

The model has multiple objectives and there are cases where the decision variables are multiplied in the constraint conditions. Thus, the problem model cannot be solved by the linear programming method.

Algorithm Design

For the simplification of the algorithm, the input order data will be pre-processed before the algorithm starts. Here, the orders are grouped so that any two orders in a group can be allocated on the same slab. Taking the groups of order data as the input in turn will solve the whole problem.

Since the slab design problem has two objectives, it is difficult to optimize both at the same time. So the algorithm will optimize the two targets separately in two stages. The first stage cycle uses the maximum weight matching of the bipartite graph to fill the slab as much as possible until the initial design weight of all orders is satisfied, which intends to reduce the number of slabs used. The second stage use linear programming for the unfilled slabs to reduce slab residual. The overall flow chart of the algorithm is shown in Fig. 1.
Bipartite Graph Maximum Weight Matching

Firstly, the algorithm constructs a bipartite graph based on the order and slab information. The left part of the bipartite graph includes all orders, one node for an order, and the total number is N. The initial designed quantity $dq$ of each order is an integral multiple of its minimum unit weight and is within the accepted weight range of the order, that is

$$\text{max} : dq_j = n \times U_{\min}$$

$$O_{\min} \leq dq_j \leq O_{\max}$$

$$n \in \{1,2,...\}$$

The right part of the bipartite graph represents all slabs, and the total number is NUM. The initial weight of each slab is set to the maximum slab weight $S_{\max}$. Since the orders are grouped before the algorithm starts, all the orders can be assigned to the same slab. Thus, the initial bipartite graph is a complete graph. The weight of the edge $e(O_j, S_i)$ between any order $O_j$ and any slab $S_i$ is the largest integer multiple of minimum unit of the order that the slab can accommodate, that is

$$\text{max} : w^i_j = n \times U_{\min}^j$$

$$0 \leq w^i_j \leq \min\{S_{\max}, dq_j\}$$

$$n \in \{1,2,...\}$$

The constructed bipartite graph is as shown in Fig. 2. There are three orders of different colors and four slabs. The initial capacity of each slab is the maximum slab weight $S_{\max}$.

Figure 1. Algorithm flow chart.

Figure 2. Initial bipartite graph.
After the graph creation, the bipartite graph is matched to the maximum weight. Each match will assign the weight of each order to a certain slab as much as possible. After each match, the current bipartite graph needs to be adjusted, which is mainly divided into three parts. We will describe the adjustment in conjunction with Fig. 3.

For each order, if the designed quantity has been satisfied, the node corresponding to the order and the edges connected to it should be removed from the graph. For example, after this match, order 2 in Figure 3 and the edges connected to it will be deleted.

There are two cases that need adjustment for each slab. If a slab is fully dispensed, the slab is added to the full slab queue $L_{full}$. If a slab has less capacity to accommodate the smallest unit weight of any orders that the slab allows, the slab is added to the adjustment queue $L_{adjust}$. For the two cases, a new slab should be replaced at the position of the original and the weight of the new slab is set to the maximum slab weight. As shown in Fig. 3, the remaining space of slab 4 is insufficient to accommodate the minimum unit weight of any order. Then slab 4 should be added to the adjustment queue and a new slab will replace it at the same position.

For the edges in the bipartite graph, we could consider from the angle of the slab. For each slab, if the color number of the slab is less than 2, all the remaining order nodes could be connected to the slab. If the color number is 2, the order nodes of the corresponding two colors could be connected to the slab. The weight assignment strategy of both cases is the same as when creating the bipartite graph. As shown in Figure 3, the slab 1 has two colors, thus, only order 1 and order 2 can be connected to it. Slab 2 and slab 3 could connect to all the remaining order nodes (considering the existence of other orders).

![Figure 3. Adjust bipartite graph.](image)

After the iteration of bipartite graph maximum weight matching is completed, the initial designed quantity $dq$ of all orders has been satisfied, which is an integer multiple of the minimum unit weight of orders. Since the initial capacity of the slab is the maximum slab weight and slabs are filled as much as possible during the filling process. Thus, the number of slabs used is as small as possible. There is no residual material in the full slab queue. Then we will adjust slabs in the adjustment queue $L_{adjust}$ to reduce the residual rate.

**Linear Programming**

After the first step, the weight of the distribution on each slab is an integral multiple of the minimum unit weight of the corresponding order when each slab take the maximum slab weight, which provides adjustment space for reducing the residual material. For a slab that is not full, there are 2 ways to minimize the residual. On the one hand, for the orders allocated on each slab, the weight of each order can be increased to make the slab as full as possible. On the other hand, the slab capacity could shrink to reduce the residual material.

For an order $O_j$ assigned to a slab $S_i$ which is not full, the upper bound of the weight that $O_j$ could append is

$$\min\{U^j_{num}(U^j_{max} - U^j_{min}), O^j_{max} - dq_j\}$$ (21)
Maintaining the units number of order \( j \) assigned to slab \( i \), the space for each unit to increase is the difference between the maximum unit weight and the minimum unit weight of the order. If the appended weight exceeds the maximum weight of the order, the upper bound is the difference between the maximum weight of the order and the current assigned weight. In this way, the assigned order weight will not exceed the upper limit of the order weight.

For a slab \( i \), the upper bound to shrink is

\[
\min \{ S_{\max} - S_{\text{ass}}^i, S_{\max} - S_{\min} \}
\]  

(22)

If the weight assigned to a slab exceeds the minimum slab weight, the upper bound to shrink is the residual of the slab, otherwise the upper limit of shrinkage is the difference between the maximum slab weight and the minimum slab weight. In addition, the sum of the increment of the orders on the slab and the amount of slab shrinkage should be no larger than the residual of the slab. Now we could adjust the slabs in \( L_{\text{adjust}} \) based on linear programming. The meanings of the new symbols are shown in Table 2.

\[
\text{max} : \sum_{i=1}^{M} \sum_{j=1}^{NUM} RO_j^i + \sum_{i=1}^{M} SS_i
\]

(23)

\[
\sum_{j=1}^{NUM} RO_j^i + SS_i \leq RS_i
\]

(24)

\[
\sum_{i=1}^{M} RO_j^i \leq RO_j
\]

(25)

\[
0 \leq SS_i \leq \min \{ S_{\max} - S_{\min}^i, RS_i \}
\]

(26)

\[
0 \leq RO_j^i \leq \min \{ RO_j, U_{\text{num}}^j \times (U_{\max}^j - U_{\min}^j) \}
\]

(27)

\[
i = \{1,2...M\}
\]

(28)

\[
j = \{1,2...NUM\}
\]

(29)

Table 2. Linear programming symbol table

| symbol | meaning |
|--------|---------|
| \( M \) | Number of slabs to adjust |
| \( RO_j^i \) | Weight of order \( j \) appending to slab \( i \) |
| \( RS_i \) | Residual weight of slab \( i \) |
| \( RO_j \) | Residual weight of order \( j \)(difference between maximum weight of order \( j \) and initial designed quantity) |
| \( SS_i \) | Amount of shrinkage of slab \( i \) |
| \( S_{\text{ass}}^i \) | Assigned weight of slab \( i \) |

The objective is to maximize the sum of appending weight of all orders and the total amount of slab shrinkage. Constraint Eq. 24 indicates that the sum of the order increment on each slab and the slab shrinkage is less than the slab residual. Constraint Eq. 25 indicates that the sum of the increments of each order on all slabs is no larger than the residual of the order. Constraint Eq. 26 and constraint Eq. 27 have been described above.

After the second step, the slabs and orders are adjusted based on the results of the linear programming. The capacity of each slab is reduced by slab shrinkage. The assigned weight of each slab is increased by the additional weight of the orders assigned to the slab. And the designed quantity
of each order is increased by the sum of the additional weight of the order appended to all slabs. The corresponding formulas are as follows.

\[ S_i := S_i - SS_i \]  

(30)

\[ S_{ass}^i := S_{ass}^i + \sum_{j=1}^{NUM} RO_j^i \]  

(31)

\[ dq_j := dq_j + \sum_{i=1}^{M} RO_i^j \]  

(32)

So far, the algorithm has completed two objectives of minimizing the number of slabs and minimizing the residual material.

**Experiment**

The algorithm is implemented in C++. Bipartite graph maximum weight matching calls LEMON and the linear programming calls GLPK, both of them are open-source library. The experimental environment is Intel Core2 2. 93GHz, 4GB RAM Windows7 PC. Three sets of different experimental data were from steel enterprises. Slab weight ranges from 20 tons to 30 tons. The specific data is shown in Table 3, the unit is tons(t). It can be seen from the table that the order weight and the unit weight of these three sets vary widely, which proves that the order data covered by these three sets is sufficiently complete.

| No | Number of orders | Number of colors | Interval of Total weight of all orders | Maximum order weight | Minimum order weight | Maximum unit weight | Minimum unit weight |
|----|------------------|------------------|---------------------------------------|----------------------|----------------------|---------------------|---------------------|
| 1  | 24               | 15               | [3405, 3765]                          | 1050                 | 2                    | 2                   | 1                   |
| 2  | 70               | 25               | [10892, 12044]                        | 1050                 | 3                    | 3                   | 1                   |
| 3  | 101              | 28               | [14248, 15747]                        | 1050                 | 3                    | 3                   | 1                   |

We compare our algorithm with manual experience on the data set. The strategy of the manual experience is to successively design slabs for each order. Firstly, fill a slab to the maximum weight and copy this slab until the order is satisfied. Once an order is completed, start to match the next order. After completing the matching of all orders, enter the adjustment. Reduce the remaining material of each slab within the scope allowed by the constraint. The comparison experiment results are shown in Table 4.

| No | method                  | total weight of slabs | Number of slabs | Weight of residual | Residual rate | running time | average Slab weight |
|----|-------------------------|-----------------------|-----------------|-------------------|---------------|--------------|--------------------|
| 1  | manual experience       | 3588                  | 126             | 125               | 3.5%          | 5min         | 28.48              |
|    | Our algorithm           | 3611                  | 123             | 7                 | 0.19%         | 2s           | 29.36              |
| 2  | manual experience       | 11900                 | 437             | 488               | 4.1%          | 40min        | 27.23              |
|    | Our algorithm           | 11580                 | 400             | 10                | 0.08%         | 19s          | 28.95              |
| 3  | manual experience       | 15658                 | 603             | 831               | 5.3%          | 60min        | 25.97              |
|    | Our algorithm           | 15224                 | 529             | 17                | 0.11%         | 39s          | 28.79              |
By analyzing the experimental results, the following conclusions can be drawn:

1. In the three sets of comparative experiments, the algorithm is superior to manual experience in the number of slabs, residual rate and running time.

2. With the increase of problem scale, the performance of the manual experience on the residual rate is getting worse while the performance of our algorithm is relatively stable and maintains at a low level. This is because the second step of our algorithm utilize linear programming to minimize residual material. And the result of linear programming is the global optimal based on the first step.

3. The average slab weight in the table is the quotient of the total weight of slabs divided by the number of slabs. Theoretically, in the case of orders unchanged, maximum slab weight of the average slab weight and zero residual ratio is the best result that can be achieved. But limited by the data and the various constraints of the problem, this result is almost impossible to achieve. However, this can provide a certain reference for the experimental results. If the average slab weight is closer to the maximum slab weight (30 tons) and the residual ratio is closer to 0, it proves that the slab scheme is closer to the optimal solution. It can be seen from the table that the average slab weight of the three groups of our algorithm is close to 30, which is superior to the manual experience, and the residual rate is also lower. As the scale of the problem increases, the average slab weight of both methods decreases, but the slowdown of our algorithm is slower. The performance of the manual experience optimization method is more sensitive to the scale of the problem.

Summary

Slab-design is an important part in the production management of steel enterprises. This paper proposes a slab-design algorithm based on bipartite graph maximum matching and linear programming for the slab-design problem with uncertain weight and color constraint. Our algorithm optimizes the number of slabs and residual material in two stages. In the first stage, the bipartite graph maximum weight matching is used to fill the slab as much as possible while satisfying all the orders. The initial capacity of the slab is the maximum slab weight to keep the number of slabs used at a low level. In the second stage, we apply linear programming to the slabs that are not full, adjust the assigned weight on slabs and the capacity of slabs and minimize the residual material under constraints. The experiment result proves that our algorithm has good performance in time and optimization, which will provide decision support for the actual scheduling production.

Acknowledgement

This research was financially supported by:
- National Key Research and Development Program 2017YFB030410
- Construction Specific Project of Shanghai Engineering Research Center 17DZ2251600
- Central University Fund

References

[1] Dawande, M., et al, Approximation Algorithms for the Multiple Knapsack Problem with Assignment Restrictions, J. Journal of Combinatorial Optimization. 4. 2(2000):171-186.

[2] Martello, Silvano, and P. Toth, Knapsack problems: algorithms and computer implementations, J. Journal of the Operational Research Society. 42. 6(1991):513-513.

[3] Dawande M, Kalagnanam J, Lee H S, et al, The Slab Design Problem in the Steel Industry, in: Handbook of Production Scheduling, Springer, US, 2006, pp. 215-225.

[4] Friesen, D K, and M. A. Langston, Variable Sized Bin Packing, J. Siam Journal on Computing. 15. 1(2006):222-230.
[5] Kalagnanam, Jayant R., et al, The residual Inventory Matching Problem in the Process Industry, J. Operations Research. 48. 4(2000):505-516.

[6] Ladanyi, Laszlo, J. J. Forrest, and J. R. Kalagnanam, Column Generation Approach to the Multiple Problem with Color Contraints, J. Informs Journal on Computing. 18. 1(2007):129-134.

[7] Song, Sang Hwa, A nested column generation algorithm to the meta slab allocation problem in the steel making industry, J. International Journal of Production Research. 47. 13(2009):3625-3638.

[8] Xi, Li, An optimal algorithm for slab designing of fixed weight slab, J. Journal of University of Science and Technology Beijing. 30. 10(2008):1179-1183.

[9] Zhang, Xi, Li, Model and Algorithm for Slab Design With Interval Value Attributes, J. Foundry Technology. 29. 7(2008):945-948.

[10] Ahuja R K, Magnanti T L, Orlin J B., Chapter IV Network flows, in: Handbooks in Operations Research and Management Science., Elsevier B. V., 1989, pp. 211-369.

[11] Khachiyan, L. G., A polynomial algorithm in linear programming, J. Doklady Adademiia Nauk Cccp. 244. 80(1979):1-3.