Research on optimization technology based on two-dimensional packing

Jun Ji*, Jian Cui and Feifei Xing
Beijing Polytechnic, Beijing 100176, China
*101439@bpi.edu.cn

Abstract: The problem of packing is a problem often encountered in the early stages of advanced manufacturing application field. This paper first describes the two-dimensional nesting problem, and then analyzes three optimization techniques to solve the two-dimensional nesting problem, the mathematical model and solution process of dynamic programming, knapsack algorithm and linear programming algorithm.

1. Introduction
The problem of packing is a problem often encountered in the early stages of advanced manufacturing application field. The two-dimensional layout problem is an important branch of the layout problem. According to the requirements of the actual production process, the two-dimensional layout problems are divided into two types: the two dimensional guillotine-cutting stock (TDGCS) layout problems and the two dimensional non-guillotine cutting layout issues(UDGCS). TDGCS layout problem, this problem is an important branch of the two-dimensional layout problem. The TDGCS layout problem refers to cutting a given type and quantity of blanks from a board, requiring the determination of the optimal layout plan to minimize the total value of the board consumed. In order to solve the above layout problems, three classic algorithms are generally used: dynamic programming algorithm, knapsack problem algorithm, and linear programming method of column generation.

2. Dynamic programming algorithm
Dynamic programming is a method to solve the optimization problem of multi-stage decision-making process. This method was proposed by American Mathematician Bellman (R Bellman) in the early 1950s [1]. In view of the characteristics of multi-stage decision-making problems, they put forward the optimization principle to solve such problems, and successfully solved many practical problems in production management, engineering technology, etc., thus establishing a new branch of operations research. Multi-stage decision-making process refers to a kind of activity process that can be decomposed into a number of interconnected stages in chronological order, called "period". Decisions must be made in each period. The decision of the entire process is a decision sequence, so The multi-stage decision-making problem is a sequential decision-making problem.

The goal of multi-stage decision-making process optimization is to achieve the best overall effect of the entire activity process. Since each stage of decision-making is organically linked, the execution of the decision in this period will affect the decision in the next period, so as to affect the overall effect, so the decision-maker should not only consider the best at this stage, but also consider The impact on the final goal, so as to make a decision that is optimal for the overall situation.

The mathematical model of the dynamic programming algorithm is [2]:

\[ \begin{align*}
V(n, s) &= \min_{a \in A(n, s)} \{ c(n, a) + \sum_{a' \in A(n, s)} V(n', s') \}
\end{align*} \]
In the formula, opt represents the optimization, which can be min or max according to the meaning of the question; \( s_k \) represents the state variable; \( u_k \) represents the decision variable; \( D_k \) represents the set of allowed decisions; \( V_k(s_k,u_k) \) is the index function value of the phase corresponding to when the state is \( s_k \) and the decision is \( u_k \); \( f_k(s_k) \) represents the best benefit value from the first stage to the end of the process.

The solution of the dynamic programming algorithm is divided into the following steps:

1. Analyze the meaning of the problem, identify the multi-stage characteristics of the problem, and appropriately divide it into several stages that satisfy the recurrence relationship according to the sequence of time or space, and artificially assign the "time period concept" to non-sequential static problems;
2. Appropriately select state variables and define the optimal index function;
3. When solving, starting from the boundary conditions, the direction of the sequence (or reverse) process is recursively searched for the optimization step by step. When solving each sub-problem, we must use the optimal result of the sub-problem that it has solved before. The optimal solution of the optimal sub-problem is the optimal solution of the entire problem.

3. Knapsack problem algorithm

The knapsack problem is a typical optimization problem in operations research. It has important applications in the practice of material cutting, budget control, project selection, and cargo loading, and it is often studied as a sub-problem of other problems. After the knapsack problem was first proposed by Dantzig [3] in the late 1950s, it has been extensively studied in recent years. This is not only because the knapsack problem can be directly applied in the industrial field, but also because of many theoretical reasons. Many integer programming problems depend on an effective knapsack problem algorithm, so an effective knapsack problem algorithm is very necessary.

All knapsack problems can be described qualitatively as: selecting a subset from a given set of items to maximize the benefits without exceeding the load of all knapsacks [4-6]. The knapsack problem is a subset selection problem. Among items of different values and weights, select a selection plan for some items so that the total weight of the selected items does not exceed the specified limit weight, and the sum of the values of the selected items is the largest. The mathematical model of the knapsack problem is:

\[
\begin{align*}
\max \left( x_1, x_2, \ldots, x_n \right) &= \sum_{j=1}^{n} c_j x_j \\
\text{s.t.} \sum_{j=1}^{n} a_{ij} x_j &\leq b_j \quad i = 1, \ldots, m \\
&x_j \in \{0,1\}
\end{align*}
\]

(2)

Where \( n \) is the item number; \( m \) is the resource number; \( c_j \) is the benefit amount of the first item \( j \); \( b_i \) is the budget of the first resource \( i \); \( a_{ij} \) is the amount of the first resource occupied by the first item \( j \); \( j \) is the decision variable (when the item \( j \) is \( x_j = 1 \) selected, otherwise \( x_j = 0 \)). The language description of the backpack problem: There are currently \( j ( j = 1, \ldots, n ) \) items, and each item \( m \) will consume several resources \( a_{ij} \ (i = 1, \ldots, m) \). If the items \( j \) are loaded into the backpack, it will benefit \( c_j \). At the same time, all the items in the backpack are required to consume less resources exceed \( c_j \).

The following is solved by the dynamic programming sequential solution method:

1. Stage \( k \): Sort the optional items by \( 1, \ldots, n \), and each section contains one item \( n \), which is divided into stages, namely \( k = 1, \ldots, n \).
(2) State variable $s_{k+1}$: At the beginning of the paragraph $k$, the total weight of the previous items allowed in the backpack $k$.

(3) Decision variable $x_k$: select the number of items $k$ in the first category.

(4) State transition equation: $s_k = s_{k+1} - a_k x_k$

(5) Allow decision set: $D_k (s_{k+1}) = \{x_k | 0 \leq x_k \leq [s_{k+1}/a_k], x_k \in N\}$

(6) The optimal index function represents the maximum use value when the total weight of items allowed in the backpack does not exceed, and the optimal strategy is used to load only the first items. Then the sequence recurrence equation of the dynamic programming algorithm can be obtained as:

$$f_k (s_{k+1}) = \max_{x_k=0,1,...,\lfloor s_{k+1}/a_k \rfloor} \left\{ c_k x_k + f_{k+1} \left( s_{k+1} - a_k x_k \right) \right\}, k = 1,...,n$$

$$f_0 (s) = 0$$

(3)

The forward dynamic programming method is used to gradually calculate $f_1 (s_2), f_2 (s_3),..., f_n (s_{n+1})$ and the corresponding decision function $x_1 (s_2), x_2 (s_3),..., x_n (s_{n+1})$, and the final result $f_n (a)$ is the maximum value sought, and the corresponding optimal strategy is calculated by inverse calculation.

4. column generated linear programming method
Gilmore and Gomory proposed a linear programming method [7-8] to solve the large-scale TDGCS layout problem, and this method has been widely used at present. First, an accurate algorithm for the new layout method is generated to solve the UTDGC layout problem, and then the above algorithm is combined with the linear programming method of column generation to solve the large-scale TDGCS layout problem. Because the layout scheme given by linear programming consists of multiple layout methods, and to generate these layout methods, a large number of UTDGC layout problems need to be solved, that is, the effectiveness of linear programming mainly depends on the effectiveness of the algorithm used to solve UTDGC layout problems. Therefore, the linear programming method layout can verify the effectiveness of the new layout method generation algorithm proposed in this paper.

In the production and management work, it is necessary to plan or plan frequently. Although the content of plans or plans varies greatly from industry to industry, they can all be summed up as follows: under the constraints of various resources, how to determine the plan to achieve the best expected goal. The mathematical model of a linear programming problem is a set of linear constraints (including equations and inequalities), focusing on optimizing (maximizing or minimizing) a linear objective function, usually solved by the simplex method.

The mathematical model of linear programming consists of three elements:

(1) Variable (decision variable) refers to the unknown quantity to be determined in the problem, it can indicate the plan and measure expressed in quantity in the plan, which can be determined and controlled by the decision maker;

(2) The objective function refers to the function of the decision variable, and max or min is added in front of the function according to the optimization objective;

(3) Constraints refer to the constraints of various resource conditions when the values of decision variables are selected, usually expressed as equations or inequalities containing decision variables.

5. Conclusions
This article first describes the UTDGC layout problem and the TDGCS layout problem. When solving the UTDGC layout problem and the TDGCS layout problem, three classic methods are generally used, the dynamic programming algorithm, the knapsack problem algorithm and the linear programming method of column generation. This paper studies the mathematical models and solving process of the above three classic algorithms.
ACKNOWLEDGEMENTS
This research is supported by the Beijing Municipal Education Research Normal project (No. KM201910858004) and Natural Science Foundation of Beijing Polytechnic (No. 2021Z014-KXZ, 2019Z002-003-KXB).

References
[1] PK Agrawal. Minimizing trim loss in cutting rectangular blanks of a single size form a rectangular sheet using orthogonal guillotine cuts[J]. European Journal of Operational Research, 1993, 64(3): 410-422.
[2] Hu Yunquan. Operational Research Course (Third Edition) [M]. Beijing: Tsinghua University Press, 2007: 198-217.
[3] GB Dantzig. Discrete Variable Extremum Problems[J]. Operations Research, 1957, 5(2): 266-277
[4] Kuo Shouheng, zhou Tao. New algorithm of solving 0-1 knapsack problem based on dynamic telescopic strategy[J]. Computer Engineering and Applications, 2012, 48(4): 47-49
[5] HP Huang, M Lawley, T Morin. Tight bounds for periodicity on the unbounded knapsack problem[J]. European Journal of Operational Research, 2011, 215(2): 319-324
[6] A Rong, JR Figueira, MV Pato. A two state reduction based dynamic programming algorithm for the bi-objective 0-1 knapsack problem[J]. Computers & Mathematics with Applications, 2011, 62(8): 2913-2930
[7] PC Gilmore, RE Gomory. A linear programming approach to the cutting stock problem[J]. Operations Research, 1961, 9: 849-859
[8] PC Gilmore, RE Gomory. A linear programming approach to the cutting stock problem-part II[J]. Operations Research, 1963, 11: 863-888