Greedy Algorithm-Based Solution to Ship Loading in Knapsack Problem

Jia Hu

School of mathematics and Information Science, Nanchang Normal University, Nanchang, Jiangxi, China

Keywords: knapsack problem; Greedy algorithm; ship loading

Abstract: The research purpose of this text is to solve and optimize the freight transport and dispatch in the course of transporting, the method is to solve the partial most optimized loading and placing through the computer. The result is to realize the partial most optimized through the greedy algorithm in the knapsack problem. Innovative point is solving in the domain of definition by limiting method in the loading procession. The work of this paper also includes improving solution that is the single-source shortest path in the network. This paper mainly discusses the knapsack problem in the computer solution of the loading problem, using the greedy algorithm, from the weight of the loaded items, value and urgency of three aspects to consider, and gives the algorithm. Finally, the shortest path solution of complex path is added in the prospect.

1. Introduction

As a kind of optimal loading scheme with the greatest value, the knapsack problem means to maximize the value of items in knapsack in the case of the knapsack bearing certain weight, which can be also applied to the fields of commerce, agriculture and light industry, with its solutions conducted by brute force method, backtracking method and greedy algorithm. Greedy algorithm is usually used in daily life, for example, we look forward to the maximization and optimization of cargo value on the premise of the ship meeting the requirement of bearing capacity, that is to say, it is better not to have too much containers.

As an optimal choice in a sense, greedy algorithm needs to sort the items after setting the standards for greedy choices, and then conducts the selection and cumulative calculation according to the principle of selecting one object at a time, thus judging whether it meets the requirements of bearing weight [1]. Assume that with W as capacity, the weight and value of n items respectively as w_i and $val_i(1\leq i\leq n)$, xi valued as 0 or 1 is to control whether to put the items into knapsack or not, and the data model created in advance by knapsack problem is $Max\sum_{i=1}^{n} val_i x_i$;S.T. $\sum_{i=1}^{n} w_i x_i \leq W$ (S.T. indicates the restrictions); $x_i \in \{0,1\}, w_i > 0, x_i > 0$.

2. Solution of knapsack problem

This is a problem of finding the optimal solution, which can be expressed by a n-element 0-1 vector, with its physical meaning as follows: Assume that the i-th component is $x_i=0$ $(1\leq i\leq n)$, thus indicating that the i-th item is not loaded into the knapsack, otherwise, loaded into the knapsack [2,3]. The solution space composed of all n-element 0-1 vectors is recorded as $V=\{(x_1,x_2,...,x_n)|xi \in \{0,1\},1\leq i\leq n\}$, thus there are 2^n n-dimensional 0-1 solution space vectors. Then, at this time, in the tree organized from solution space (as shown in Figure 1), let the layer of root be the first layer to get 2^n branches leading to the leaf nodes, that is to say, the knapsack problem can have 2^n knapsack solutions, while what we need is to choose a feasible and optimal scheme among them.

DOI: 10.25236/mmmce.2021.046

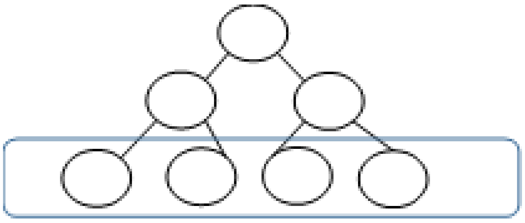

Figure 1 solution space legend

The pseudo code for mathematical recursion is as follows:

```
If(S≥W && n==0) return BREAK;
Else {i=n;S=Tsum( n-1)+$VAL_i$;}
```

It can be seen from the pseudo code that a kind of feasible knapsack scheme here is randomly obtained, while the casual arrangement of goods makes us possibly obtain other solutions. An algorithm is described as follows:

```
Knapsack(double&w[],double &v[],int W)
{printf(" It should be loaded and placed within the legal limits.");
double weigSum=0,valSum=0;
count=0;                    /*The count starts at 0*/
for(idis=1;idis<=n;idis++)
{weigSum+=*(w[]+i);     /*Accumulate the weight of the i-th item*/
valSum+=*(v[]+i);       /*Accumulate the value of the i-th item*/
if(weigSum>=W)        /*Reduce the  current item immediately once the loading and placing are out of range*/
   {weigSum-=*(w[]+i); valSum-=*(v[]+i); break;}
count++;
   }
 printf("Weight and value of the output loading weigSum, valSum");
}
```

It is also feasible to build an object-oriented structure for the convenient description by putting the weight and value of an item into the same structure, which is expressed as follows:

```
Struct Object
{long int id;          /* Serial number of items/
 double weight;
 double val;
}
```

However, the optimal solution may not be obtained by the above method, so another algorithm added with the idea of sorting is proposed here to find the optimal solution, which sorts the value of items from large to small, and then the items are loaded into knapsack according to the principle of loading the valuable item first, while the process of sorting only uses a temporary space. The sorting algorithm Sort() is as follows:

```
   Sort(Struct Object &ob[],n)
   { Struct Object temp;           /*Temporary swap space to be used later*/
    int i,loc;
    For(i=0;i<=n-1;i++)
     {temp=*(ob[]+i);
     loc=i-1;
     While(temp.val<*(ob[]+i).val && loc>=1)
       *(ob[]+loc+1)=*(ob[]+loc);
         *(ob[]+loc)=temp;
          }
     }
```

From the perspective of space complexity, it only uses a double-precision variable unit, so the space complexity is S(1). The time complexity is O(n) under the best situation, while it is $O(n^2)$ under the worst situation.

3. Knapsack problem under greedy algorithm

on the premise of items in good sorting, suppose a standard for greedy choices, which is to maximize the profit or weight of items after loading into knapsack, that is to say, load the items into knapsack according to their value from high to low, and then further carry on the screening of sequenced items once the situation of exceeding the bearing capacity happens. The algorithm based on the above is as follows:

```
Sort(Struct Object &ob[],n);
Knapsack(Struct Object &ob[],n,int W)
{double weigSum=0,valSum=0;
int tag;          /*Optimization tag*/
count=0;                 /The court starts at 0*/
for(i=1;i<=n;i++)
{weigSum+=*(&ob[]+i).weight;
valSum+=*(&ob[]+i).val;
if(weigSum>=W)
    {weigSum-=*(&ob[]+i).weight;
    valSum-=*(&ob[]+i).val;
    Sub=W-weigSum;
    j=i+1;tag=0;
    While(j<=n)
    { If(Sub>=*(&ob[]+j).weight )
    {tag=1;                         /* Start the optimization tag when meeting the condition*/
    weigSum+=*(&ob[]+i).weight;          /*loading*/
    count++;
    Sub=W-weigSum;
       }
     else { j++;
             tag=0;
             }
    j++;
    }

    break;
}
count++;
}
}
```

Assume that the standard of greed is to load and place the items according to their urgency. With the same degree of urgency, the items are loaded and placed according to their value from large to small, and the members that can add urgency to the structure are as follows:

```
     Struct Object
    {long int id;
     int urge;          /*Urgency*/
     double weight;
     double val;
    }
```

At this time, the algorithm needs to carry on sorting for two times respectively according to the urgency and value of items, and then a judgment on value is added to the process of loading and placing.

4. Combination of international logistics and knapsack problem

Knapsack problem has been widely applied to various fields, such as asset budget, cargo loading, warehouse storage, etc [4].

As a kind of transnational logistics activities, the international logistics includes the delivery and customs declaration at the place of origin [2], and the customs declaration and delivery at the place of destination. With international transportation served as the core of international logistics, customs clearance procedures (customs declaration procedures) playing an extremely role in logistics refers to that the exporter or importer applies to the relevant customs for export or import, accepts the customs supervision and inspection, and performs the customs prescribed procedures. Then, after the completion of customs clearance procedure, the goods can pass, with the description of this operating system shown in Figure 2. The agent receiving the goods from seller will first carry on customs declaration, and then deliver the goods to carrier for international transportation after customs examination; while the carrier will carry on customs declaration again, and then deliver the goods to the international freight agent for shipment and delivery after the second-time customs examination. In short, customs clearance procedures include declaration, examination, taxation and release.

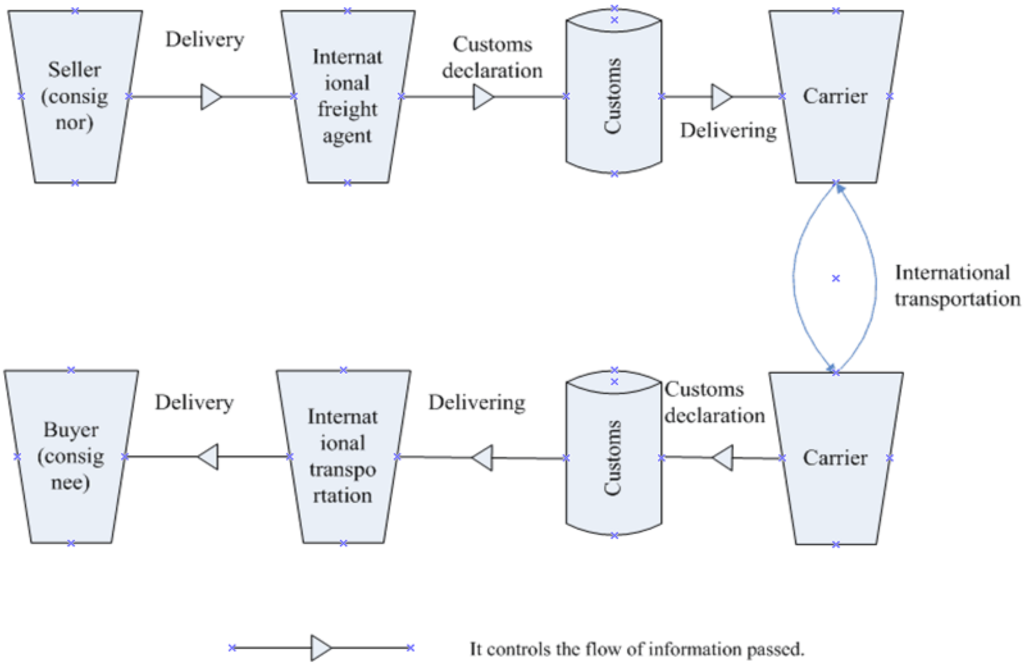

Figure 2 Information flow

Belonging to the arterial subsystem of international logistics, international transportation has significant difference from domestic transportation in features and functions. Generally, domestic transportation mainly involves transportation management in China, technical equipment and other relatively simple problems, while international transportation involves the quantity, structure, variety, model and characteristics of goods for international import and export, which requires different transportation modes, transportation channels and transportation management. There are various freight transport modes in international logistics, such as sea transportation, railway transportation, air transportation, highway transportation, etc., among which, international multimodal transport as the most important mode is used in electronic products, computers, electrical appliances, fashionable dress and toys with strong seasonal characteristics.

The important position for the speed of international logistics originates from the following aspects: On the one hand, the long transportation time and the growing funds occupation caused by the long transportation distance push the acceleration of resource allocation and liberation of occupied funds to be particularly important; on the other hand, the slow speed will make the market price miss a good opportunity and cause the decline of economic benefit. Container shipping as a mode of transportation on liner routes uses a kind of special container with certain specification and strength, for example, 20-foot and 40-foot containers with specific loading capacity are usually used in sea and land transportation, with the loading mode of full container load (FCL) and less than container load (LCL). In transportation, the FCL mode means that the container filled with items is served as the unit for shipment, while during the process, under the supervision of customs, the empty container transported to factory or warehouse is conducted with sealing and lead sealing after being filled with goods by owner, and then delivered to the carrier, thus the owners can obtain the receipt from the station, so as to obtain the bill of lading. The LCL mode means to classify, sort, centralize and pack the goods at the freight station or transfer station according to the nature and destination of goods when the quantity of items consigned by the owner is less than the full container.

If the sorting of containers according to weight is available, the sorting based on critical weight and value can be used in the LCL mode. The algorithm is as follows:

```
Sort1(Struct Object &ob[],n);      /*Carry on sorting according to the weight in items*/
Sort2(Struct Object &ob[],n);    /*Carry on sorting according to the critical weight in items*/
Sort3(Struct Object &ob[],n);      /*Carry on sorting according to the value in items*/
Knap(Struct Object &ob[],n,int W)
{while(Loading capacity limit)
   {i=1;
   while(i<=n)
   {If(*(&ob[]+i).urge==*(&ob[]+i+1).urge   /* With the same urgency degree, the item with large value is firstly loaded into knapsack*/
        If(*(&ob[]+i).val>=*(&ob[]+i+1).val)
              weigSum=*(&ob[]+i).val
         else weigSum=*(&ob[]+i+1).val}
         }
      }
```

5. Prospect and algorithm implementation of knapsack problem

In the loading problem, from the perspective of urgency, clothing and food generally have higher priority relative to electronic products [5,6]. Then, for the setting of weight based on tax rate, the weight of heavy tax rate will be set lower, while the weight of low tax rate will be set higher. In addition, the consideration of path distance makes the selection of a shorter path available. The following algorithm for the shortest path of the improved loading path is the single-source shortest path and can be solved by adjacency matrix.

The steps to find the single-source shortest path are as follows:

First define a node (one port) as single source;

Record the paths from single source to other nodes, so as to select the shortest path and put it into a found set F;

If there are N nodes, then carry on N-1 cycles with the following operations:

Go down the tail node on the shortest path selected in the previous step to check whether there is a shorter path to other nodes, and then carry on the filtering again to select the shortest path and put it into F.

```
CreateGraph();
Shortdist()
{
scanf("%d",vertex);   /*vertex is the sequence number of single-source point*/
j=vertex;
q=0;
Min=INF;p=0;q=0;   /* P is the head pointer of an array queue, q is the tail pointer of an array queue*/
For(i=1;i<=N; i++)
{
 If(i==vertex) {SET[i]=1; continue;}
```

```
Dist[i]=edges[j][i];       /*Distance from single source to other points when the path length is 1*/
If(Dist[i]<Min) {Min=Dist[i]; queue[++q]=i;p=q;}
If(Dist[i]==Min)
        queue[++q]=i;
}
While(p>q)
    For(k2=1;k2<=N;k2++)
    {   d=queue[p];
    If(SET[k2]==0&&edges[d][i]+dist[d]<dist[i])
    {dis[i]=edges[d][i]+dist[d];
    Min=Dist[1];x=0;
       For(i=1; i<=N&&SET[i]==0;i++)
       {If(Dist[i]<Min) {Min=Dist[i];queu[++x]=i;y=x;}
        If(Dist[i]==Min){ queu[++x]=i;}
    For(i=y;i<=x;i++)     queue[++q]=queu[i];
    q++;
       }
    }
```

6. Summary

In this paper, the analysis of loading problem based on the introduction of knapsack problem and the solution for greedy algorithm shows that on the premise of meeting bearing capacity, the loading in transportation shall firstly consider urgency degree and then the value. The final part describes the algorithm of shortest path selection in complex transportation.

References

[1] Wu Ping, Fang Huan. Application of greedy algorithm with preference. Computer Knowledge and Technology, 2020, 16 (03): 96-97.

[2] Yan Li. Algorithmic decision analysis of 0-1 knapsack problem. Computer Knowledge and Technology, 2020, 16 (04): 259-260+264.

[3] Zhang Dandan, Wang Xiaofeng, Feng Wanjing, Zuo Fengyuan. A confidence propagation algorithm for solving 0-1 backpack problem. Journal of Zhengzhou University (Science Edition), 2021, 53 (01): 29-34.

[4] International freight and logistics agent. China Machine Press, 2003.

[5] Wang Xiaodong. Computer algorithm design and analysis (4th edition). Electronic Industry Press, 2012.

[6] Chen Yang, Li Xinyang. Summary of greed strategies in algorithms works. China Journal of Multimedia and NetworkTeaching (III). 2019, (11): 27-35.