Design and Application of Intelligent Course Selection Algorithm for School

Guixiang Qiu¹, Zhixu Liu² and Yuye Sun³*

¹Information Center, Northeast YuCai School, Shenyang, Liaoning, 110179, China
²The Gifted Education Experiment Division, Northeast YuCai School, Shenyang, Liaoning, 110179, China
³Marine Electrical Engineering College, Dalian Maritime University, Dalian, Liaoning, 116026, China

*Corresponding author's e-mail: qiu_yc@sina.com

Abstract. Course supermarkets are emerging in many schools in China. Therefore, choosing the right courses has become a practical problem for many students. One of the primary concerns of students is to earn enough credits in a particular time limit. In this paper, we study the problem of credits maximization and evaluate the accuracy and efficiency of three different algorithms: greedy method, search method, and dynamic programming. Finally, we propose a practical implementation of the course selection algorithm that would provide new ideas for students to choose courses and suggestions for schools to set up online curriculums and course selection models.

1. Introduction
It has become more and more common in school education to optimize the curriculum structure, open selective courses, and give students the opportunity to choose their learning path. At present, "curriculum supermarket" model emerges in many schools. Choosing the right courses has become a practical problem for teachers and students. Among many considerations, the pursuit of maximum credits is one of the most important one.

2. Analysis of Supermarket Course Selection Needs

2.1. Supermarket Course Requirements Description
Each school in the design of curriculum supermarkets, has their own ideas. Taking Northeast Yucai School as an example, there are hundreds of courses given by the school. Students choose one course at the beginning of each semester, and super-excellent students are allowed to re-select courses in the middle of the semester. As a result, students will have multiple opportunities to select their courses. However, students need to consider the prior conditions of the course. Table I lists part of the school's course list for reference analysis.

Table 1. Supermarket Credit and Pre-study Requirements for Northeast Yucai School Curriculum (Part)

| NO. | Course title                  | Credit | Pre-study Requirements |
|-----|-------------------------------|--------|------------------------|
| 1   | English intensive reading     | 3      |                        |
2.2. Needs analysis and induction
Simply looking at hundreds of course lists, it is easy for students to choose courses according to their interests, but the corresponding cost is randomness, lack of overall planning, especially the analysis of credit.
Table 1 shows the different credits of each course and the dependency between the courses. Therefore, in order to obtain the maximum credits, we should consider not only the credit of each course, but also the condition of taking the course.

3. Design of Intelligent Course Selection Algorithm for Course Supermarket

3.1. Model of Intelligent Course Selection in Course Supermarket

3.1.1. Symbol Description
\( N \) – total number of courses;
\( M \) – the total number of courses taken by each student;
\( A_1 \) – the selected course number;
\( S \) – current total credits;
\( MaxS \) – maximum credits;
\( W[i] \) – credits for the i course.

3.1.2. Issue Assumptions
Hypothesis 1. Assume that each student must choose only one course at a time;
Hypothesis 2. For any students, \( M \) course selection is faced with the same list of courses;
Hypothesis 3. Each course can have 0 or several pre-study courses, but only one direct pre-study course;
Hypothesis 4. Does not take alternative requirements into account other than the pre-study requirements;
Hypothesis 5. Assume that the course can be selected to obtain credits, regardless of the non-qualified study.

3.1.3. Building Models
Under these hypotheses, the problem can be explained as follows:
Among \( N \) courses, each course has different credits. The condition of successful selection is that the first course must have been completed. Now choosing \( M \) from the \( N \) courses can get the maximum total credits.
Thus, the solution of the problem becomes the calculation of the following expressions.
\[ MaxS = \max\{S\} = \max\{W[A[1]] + W[A[2]] + \cdots + W[A[M]]\} \]
3.2. Analysis and Design of Intelligent Course Selection Algorithm for Course Supermarket

3.2.1. Greedy algorithm

To find the maximum value of the sum of a set of data, a naïve strategy is choosing the biggest item at every decision. But this greedy algorithm does not work when considering about the pre-study courses. A better solution is using topological sorting. Taking the course number as the node serial number, the course credit is marked next to the node according to the pre-study relationship, then the course map is established.

![Course Map](image1)

Figure 1. Course Map

Only a node with 0 in-degree currently can be freely selected, and after the node is selected, the in-degree of its subsequent node can be reduced by 1.

Based on the above analysis, the greedy algorithm is improved as follows.

- **Step1.** \( S \leftarrow 0, \) course counter \( C \leftarrow 0; \)
- **Step2.** select the highest credit \( A[i] \) from the zero in-degree node;
- **Step3.** \( S \leftarrow S + W[A[i]]; \)
- **Step4.** \( C \leftarrow C + 1; \)
- **Step5.** reduce the entry of \( A[i] \) subsequent nodes by 1;
- **Step6.** delete \( A[i]; \)
- **Step7.** if \( C < M, \) goto Step2;
- **Step8.** MaxS \( \leftarrow S. \)

3.2.2. Search Algorithm

The greedy algorithm, although chooses the highest credit course at each time, does not consider the restriction relationship of the course, which makes it only pay attention to the present and ignore the subsequent influence of this step. As a result, there is a phenomenon of dissociation. In order to get a better solution, the search method must be considered naturally.

Based on topological sorting, figure 1 can be transformed into the following directed acyclic graph. In order to express the starting point of course selection, a "super node" 0 can be added and its credit is 0. The course selection must start from 0, so the length of the final course selection sequence becomes 4.

![Directed Acyclic Graph](image2)

Figure 2. A topologically ordered directed acyclic graph

Each course credit is regarded as the weight from its precursor to itself, and the problem is transformed to finding the maximum weighted path from node 0 in the directed acyclic graph.
The key part of the search is function \( g(o(s,i,j)) \), which records the course situation at the arrival node and the selected course. The function is described as follows.

At the beginning, make \( \text{MaxS} \leftarrow 0 \).

\[
\begin{align*}
A[j] & \leftarrow i; \\
S & \leftarrow SW[i]; \\
\text{If } j = M + 1, \text{then} \\
\quad & \text{// compare of current S and MaxS: } \\
\quad & \text{if } S > \text{MaxS, then update the MaxS;} \\
\quad & \text{otherwise} \\
\quad & \text{enumerate all subsequent nodes } k \text{ of } i \\
\quad & \text{// mark node } k \text{ occupied;} \\
\quad & \text{go } (s + w[k], k, j + 1); \\
\quad & \text{// cancel the node } k \text{ occupation mark;}
\end{align*}
\]

By using this algorithm, the maximum credit value is calculated as follows.

\[\text{MaxS} = \max\{8,8,8,8,6,9,7,8\} = 9.\]

3.2.3. Dynamic Programming

Analysis figure 1, since each course has only one direct first node, they form a forest. Let “super node” 0 be the father of every 0 in-degree node, and the forest turns into a tree.

![Figure 3. Course Tree](image)

Now the problem is to choose one from \( M + 1 \) courses (because 0 must be chosen, so the total number of the chosen courses becomes \( M + 1 \)), and the chosen course must satisfy all its ancestors. Concrete implementation is to start from the root, and the \( M + 1 \) of an opportunity is to be allocated.

For this optimization problem, which can be based on the data structure of tree, tree dynamic programming is naturally thought of.

The state transfer equation can be constructed by using \( F[i,k] \) to represent the maximum credits with \( k \) selected courses in a subtree rooted in \( i \).

\[
F[i,j] = \max\{F[t,k]F[i,j-k]\}
\]

Where \( 0 \leq k \leq j - 1 \), \( t \) is the son node of \( i \), and \( F[i, 1] = W[i] \) as initialized.

Next, updating from the deepest point by depth first search, and as the result, \( F[0, M + 1] \) is the highest credit. The steps of the deep priority search node i algorithm DFS (i) are as follows:

Enumerate all the sons of i:

\[
\begin{align*}
& \text{// record the current son node with } t; \\
& \text{DFS (t);} \\
& j \leftarrow \text{from } M \text{ to } 1 \\
& k \leftarrow \text{from } 0 \text{ to } j - 1 \\
& F[i,j] \leftarrow \max\{F[i,j], F[t,k] + F[i,j-k]\}
\end{align*}
\]

For further analysis, the above equation of state needs to enumerate several son nodes of each node, and the selection of each son needs to be recursively processed. It is more troublesome when one’s son node is numerous. Is there a simpler way?

If the problem is just modelled as a binary tree, the problem is transformed into a binary tree solution question. Using the left son-right brother method, the brothers are connected by lines as the
right child of the root, while the original first son is retained as the left child. At the same time, nodes of the second son and all its subordinate sons are deleted. After further finishing, the tree is transformed into a binary tree.

![Figure 4. Converts the Course Tree to a Binary Tree](image)

In figure 4, the left child is the child of the original root node, while the right child is the brother of the original root node. Therefore, when the left child allocates the course selection resources, the root node must be selected, while the root need not to be chosen when selecting the right child.

\[ F(i, j) = \max\{F[i, \text{right}, j], F[i, \text{left}, k]F[\text{the right child of } i, j - k - 1] + A[i]\}\]

Where \(0 \leq k < j < N\), \(i, \text{right}\) represents the right child of the node \(i\) and \(i, \text{left}\) represents the left child of node \(i\).

### 3.3. Comparative Analysis of Three Course Selection Algorithms

#### 3.3.1. Examination of Maximum Credit Acquisition

For the convenience of programming and testing, 10 groups of test data are designed to test whether the three algorithms can get the correct maximum credits.

The input format of the unified data includes several rows. The first row has two data representing the total number of courses \(N\) and the number of selected courses \(M\). The next row shows the serial number of each behaviour course and the serial number of the first course. The output is only one number, indicating the highest credit, while assuming \(N \leq 300, W[i] \leq 10\).

Each algorithm is programmed to realize three algorithms, and the test platform based on SYZOJ is used for evaluation. The time limit of each data is set to 1 minute. The correctness of the algorithm, the running time, and the memory occupied by the runtime are as follows (A represents greedy method in the table, B represents search method, and C represents dynamic programming).

| NO. | Algorithm | Score (%) | Runtime(ms) | Memory(KiB) |
|-----|-----------|-----------|-------------|-------------|
| 1   | A         | 100       | 1           | 240         |
|     | B         | 100       | 2           | 2172        |
|     | C         | 100       | 2           | 756         |
| 2   | A         | 0         | 2           | 228         |
|     | B         | 100       | 2           | 2300        |
|     | C         | 100       | 2           | 2312        |
| 3   | A         | 0         | 3           | 2172        |
|     | B         | 100       | 2           | 332         |
|     | C         | 100       | 2           | 2172        |
| 4   | A         | 0         | 2           | 2172        |
|     | B         | 100       | 3           | 328         |
|     | C         | 100       | 2           | 2300        |
### 3.3.2. Performance Comparison of the Three Algorithms

In Table 2, when the maximum credits are found for specific data, the three algorithms show great differences: greedy method and dynamic programming have obvious advantages on speed, while search method cannot run the results within ten minutes for the last five groups of data. Although, the search method must be correct in theory, the program runs too slowly to be put into practice, only 5 groups of data get the correct result in the specified time. The correct rate is 50%, and the other data does not get the result in the specified time. Therefore, right or wrong cannot be easily judged. Only dynamic programming get a 100% correct rate in the specified time, and the total spent time of 10 groups of data is only 118 ms.

To further analysis, for data 1, the three algorithms can find exactly the same course sequence; for data 2, 3, 4, search method and dynamic programming find out the inconsistent course sequence that can get the maximum credits; for data 5, the three algorithms find the inconsistent course selection sequence that can get the maximum credits; for data 6, greedy method and dynamic programming find different course selection sequence that can get the maximum credits.

### 4. Conclusion

According to the above analysis, different algorithms may produce different course selection sequences when the maximum feasible credits is considered. In the comparison of space-time complexity, dynamic programming wins with absolute advantage and becomes the first choice for obtaining the maximum credits efficiently. At present, the application of intelligent course selection program with dynamic planning and design is stable in Northeast Yucai School, which has become an important reference basis for related staff to design courses.

### Acknowledgments

This thesis is supported by Revitalizing Liaoning Talents Program.

### References

[1] Qihua Zheng. (2005) Data Structure. Tsinghua University Press, Beijing.

[2] Guoye Liang . (2004) Mathematical Modeling. Metallurgical Industry Press, Beijing.

[3] Qizong Wu . (2002) Operations Research. Machinery Industry Press, Beijing.

[4] Thomas H.Cormen, Charles E.Leiserson,Ronald L. Rivest,Clifford Stein. (2012) Introduction to Algorithms. Mechanical Industry Press, Beijing.

[5] Guixiang Qiu. (2009) Algorithm Art. Liaoning Education Press, Shenyang.
[6] Bingying Li, Limin Xia, Yuanzhong Shu. (2004) Algorithm of Online Course Selection System under Credit System. Jiangxi Science, 05: 358-360.

[7] Zhanglian Xie. (2010) The Realization of High School Students' Right to Freedom of Educational Choice under the Background of Credit Management. Education Academic Monthly, 02: 15-19.