Intelligent Course Scheduling System Based on Case-based Reasoning

Suqi ZHANG¹, Xinxin Wang²

¹School of Information Engineering, Tianjin University of Commerce, Tianjin 300134, China
²School of Science, Tianjin University of Commerce, Tianjin 300134, China

First author’s e-mail: suqizhang@tjcu.edu.cn

Abstract: In view of the current situation of tight teaching resources, the intelligent scheduling system alleviates this problem to a certain extent, and the advantages and disadvantages of the scheduling method determine the performance of the scheduling system. A method based on Case-based reasoning is presented in this paper to solve the course scheduling problem. Using graph theory, a case representation model is built for the course scheduling problem, a measure is established for case similarity, and a case retrieval method based on maximum common subgraph selection is brought forward. Finally, the validity of the proposed method for solving course scheduling problem based on case-based reasoning is verified through an experiment.

1. Introduction

As the scales of colleges gradually expand, the numbers of students and courses gradually increase, and teacher and classroom resources are increasingly tight. With a low efficiency and high error rate, traditional manual scheduling can no longer satisfy the demands of the teaching management in colleges today. Course scheduling is intended to arrange teachers, students, courses, classrooms and time in a reasonable way, so that the teaching plans can go on smoothly. Research on course scheduling started in the late 1950s. In 1962, Prof. Gotlieb proposed a mathematical model concerning course scheduling. In the 1970s, S. Even et al. proved that the course scheduling problem was an NP-Complete problem and then theorized this problem. Domestic studies on course scheduling started late. It was not until the 1990s, when computer technology was popularized, that the computer-based scheduling of course became the mainstream. Currently, theoretical approaches to course scheduling mainly include linear programming, integer programming graph theory method, multi-objective integer programming model, genetic algorithm, tabu search algorithm simulated annealing algorithm, ant colony algorithm, particle swarm optimization algorithm, greedy algorithm, backtracking algorithm and Particle Swarm Optimization with Selective Search. But these automatic scheduling approaches have a high complexity and poor flexibility. On the application side, S.H. Jin have considered the impact of scheduling on students' learning. At the same time, TISER from Tsinghua University has already completed course scheduling by combining traditional scheduling method with man-machine interaction, but there were also some problems, such as low efficiency and high demand for the experience of course schedulers.

In practical application, every semester, timetables must be made according to scheduling tasks. The reasons for changes in the timetable every semester mainly include the increase of students and
teachers, or the adjustment of courses, but the changes are small relative to the whole. Therefore, the purpose of this paper is to draw lessons from the previous experience and results of course scheduling, to deal with new changes relative to the previous content, so as to obtain new timetables.

Case-based reasoning (CBR)[4] is an important knowledge-based problem-solving and learning method in the field of artificial intelligence, which mainly solves new problems based on previous experience[5][6]. Its core idea is that the more similar the problems are, the more similar the solutions are. The solving process of CBR problem is generally formalized as four parts: retrieve, reuse, revise and retain. Among them, retrieve refers to retrieving from memory cases according to the new problem and finding similar cases in previous ones. Reuse refers to trying to use the solution in the retrieved case to solve the existing problem. Revise refers to evaluating and adjusting the application effect of previous solution. While retain refers to storing the new case in memory for later use, after the problem in the new case is solved and satisfactory results are obtained[7][8][9]. In the present study, the course scheduling problem in colleges is investigated by means of CBR, and solutions are suggested from case representation, retrieval, similarity calculation, and so on. It can save a lot of repetitive work and is of great significance for the improvement of scheduling efficiency.

2. Course Scheduling Problem

2.1 Description of Course Scheduling Problem

In course scheduling, scheduling tasks are expressed as a 3-tuple \( c = (l, p, s) \), where \( l \) represents the course, \( p \) represents the teacher and \( s \) represents the students the teacher is attending the class. \( C = \{c_1, c_2, ..., c_n\} \) represent all scheduling tasks. A 2-tuple \( t = (h, r) \) is used to express resources that are available for scheduling, where \( h \) represents the time slot (for example, each day is divided into five time slots) and \( r \) represents the classroom. \( T = \{t_1, t_2, ..., t_n\} \) stands for all resources that are available for scheduling. Since only one scheduling task can be arranged for the same classroom in a given time slot, the total number of resources \( n \) is the product of time slots and the number of classrooms.

Course scheduling is intended to assign all scheduling tasks to appropriate resources, so that there will be no conflicts among teachers, students and courses in terms of time and use of classroom. Their conflicts include two aspects. In the same time period, the same scheduling task is assigned to a teacher; in the same time period, the same scheduling task is assigned to students[10]. To wit, for a given scheduling task, \( c_i, c_j \in C \), \( t_1, t_2 \in T \) and \( t_1 \) and \( t_2 \) are corresponding timetable arrangements and \((p_i = p_j) \lor (s_i = s_j) \Rightarrow (h_1 \neq h_2)\).

The ordered pair \( r = (c, t) \) is used to denote the timetable arrangement of a scheduling task. In doing so, a course scheduling problem is converted into a problem of finding a set of \( R = \{c_i, t_j \mid c_i \in C \cup C_0, 1 \leq i \leq m, t_j \in T, 1 \leq j \leq n\} \), where \( C_0 \) is a null course.

2.2 Case Presentation of Course Scheduling Problem

Case representation is the basis of CBR. From the perspective of problem solving, a case should not only describe the overall situation of a problem, but also describe the solution to this problem. So we describe a case as \(< \text{description of problem}, \text{solution description}>\).

In course scheduling, \(<G, R>\) is used to represent a scheduling case. Where the graph \( G = (C, E) \), \( C \) is a set of vertices constituted by all scheduling tasks and \( E = \{c_i, c_j \mid c_i, c_j \in C \land (p_i = p_j \lor s_i = s_j)\} \) is a set of edges constituted by conflicts among scheduling tasks. \( R \) is the timetable. For ease of illustration, we use the matrix shown in Tab1 to describe.

| Classroom | Time  | \( h_1 \) | \( h_2 \) | \( h_3 \) | \( h_4 \) | \( h_5 \) |
|-----------|-------|----------|----------|----------|----------|----------|
| \( r_1 \) |       | \( C_1 \) | \( C_0 \) | \( C_7 \) | \( C_0 \) | \( C_9 \) |
| \( r_2 \) |       | \( C_2 \) | \( C_0 \) | \( C_8 \) | \( C_0 \) | \( C_{1\theta} \) |
Therefore, the solution of a course scheduling problem based on CBR is a process of retrieving the most similar scheduling case from memory cases, in accordance with the requirement of the new task, reusing and adjusting the timetable solution in the retrieved case, so as to establish a new R. How to retrieve the most similar case quickly is the key to the realization of course scheduling. With this in mind, we study the case retrieval method based on maximum common sub-graph selection.

3. Case Retrieval Based on Maximum Common Subgraph Selection

The first matter to be solved by case retrieval is how to define a measure of case similarity[11]. In this paper, the course scheduling problem is expressed with a graph model. Thus, the measurement of case similarity is converted into the calculation of similarity between graphs. There are mainly two methods to calculate similarity between graphs: edit in distance-based method[12] and maximum common subgraph-based method[13][14]. The edit in distance-based method carried out a series of editing (insert, delete and update) on the graphs, and calculated the similarity between graphs by calculating the editing cost. But how to obtain the editing cost of a specific problem is always a difficult point. The maximum common subgraph-based method, however, is to retrieve the maximum common part between two graphs, and derive the similarity between two graphs through the proportion accounted for by the maximum common part. The advantage of this method is that it doesn’t rely on editing cost and has fewer restrictions and no special demand for data, so it is more suitable for the measurement of case similarity. In this paper, CBR is used to solve the scheduling problem. After the most similar case is retrieved, it is necessary to reuse and adjust the solution that is obtained. The reused part is the maximum common part and the rest is adjusted. In view of the above, the maximum common subgraph-based method is used to retrieve cases.

3.1 Basic Definitions

Definition 1 Subgraph: for a given graph \( G = \langle V, E \rangle \), its subgraph \( G_1 = \langle V_1, E_1 \rangle \) is defined as flows: \( V_1 \subseteq V \), \( E_1 = E \cap (V_1 \times V_1) \). And the symbol \( G_1 \subseteq G \) is used to represent that \( G_1 \) is a subgraph of \( G \).

Definition 2 Isomorphic subgraph \( G = \langle V, E \rangle \), \( G' = \langle V', E' \rangle \). \( G_1 = \langle V_1, E_1 \rangle \). If \( G_1 \subseteq G \) and \( G' \cong G_1 \), then \( G' \) is called an isomorphic subgraph of \( G \).

Definition 3 Common subgraph \( G_1 \) and \( G_2 \) are three graphs. If both \( G \) and \( G_1, G \) and \( G_2 \) are in isomorphic subgraph, then \( G \) is a common subgraph of \( G_1 \) and \( G_2 \).

Definition 4 Maximum common subgraph: \( G \) is the maximum common subgraph of \( G_1 \) and \( G_2 \) if and only if there is no \( G' \), which is also the common subgraph of \( G_1 \) and \( G_2 \) has more nodes than \( G \). The maximum common subgraph is denoted as \( \text{mcs}(G_1, G_2) \).

Definition 5 Similarity between graphs: the similarity between \( G_1 \) and \( G_2 \) is denoted as:

\[
S(G_1, G_2) = \frac{2 \times |\text{mcs}(G_1, G_2)|}{|G_1| + |G_2|}
\]

Where \( |G| \) is the sum of the numbers of vertices and edges in graph \( G \), and \( |\text{mcs}(G_1, G_2)| \) is the sum of the numbers of vertices and edges in the maximum common sub-graph. It is easy to see that the greater value \( |\text{mcs}(G_1, G_2)| \) has the greater similarity between two graphs.

3.2 The solving algorithm for the maximum common

Subgraph \( G_1 = \langle V_1, E_1 \rangle, G_2 = \langle V_2, E_2 \rangle \), the initial value of \( m \) is the smaller of the numbers of vertices in two graphs. \( M = \min\{|V_1|, |V_2|\} \). Define the threshold as \( k = m/2 \). The solving algorithm for the maximum common sub-graph of two groups is:

Step 1: If \( m \geq k \), then find the sub-graph sets of \( m \) nodes of \( G_1 \) and \( G_2 \). Go to Step 2; If \( m < k \), it is believed that \( G_1 \) and \( G_2 \) have very little isomorphic and the isomorphic can be ignored. The two graphs are considered to have no isomorphic sub-graph, and the output is null.

| r_3 | C_3 | C_5 | C_0 | C_0 | C_{11} |
|-----|-----|-----|-----|-----|-------|
| r_4 | C_4 | C_6 | C_0 | C_0 | C_{12} |

| r_5 | C_5 | C_0 | C_0 | C_{11} |
|-----|-----|-----|-----|-------|
| r_6 | C_6 | C_0 | C_0 | C_{12} |

IOP Publishing
Journal of Physics: Conference Series 1920 (2021) 012086 doi:10.1088/1742-6596/1920/1/012086
Step 2: Whether there are isomorphic elements between two sub-graph sets is judged using the method to judge isomorphic graph shown in Fig 1[15]. If there are isomorphic elements, then output $m_{cs}(G_1, G_2)$ and the correspondence among nodes. If there are no isomorphic elements, then $m = m - 1$, return to Step 1.

Fig 1. the Judging Process of Graph Isomorphic

4. The Realization of Course Scheduling Based on CBR

To explain how CBR solved the course scheduling problem more clearly, the scheduling problems for junior students of computer major in a certain college from 2016 to 2018 was taken as an example. The scheduling problems in 2016, 2017 and 2018 were denoted as Case 1 $< G_1, R_1 >$, Case 2 $< G_2, R_2 >$ and Case 3 $< G_3, R_3 >$ respectively, where Case 1 and Case 2 were known cases in memory cases, and Case 3 was a problem to be solved. There were 100 classrooms in the college. Suppose there were 4 time slots per day, and 5 working days in a week, so there were a total of 20 time slots. The goal of calculation was to assign all scheduling tasks in Case 3, in order to obtain corresponding teaching resources.

All the scheduling tasks in Case 3 are shown in Tab 2. $c^i_j$ represents the j$^{th}$ scheduling task of the i$^{th}$ case, $l_i$ represents the course to be scheduled, $p_j$ represents the teacher and $s_i$ represents the students. According to the task set, the scheduling problem was structured Fig 2 shows the graph description $G_3$ of case 3.

Solving Steps
Step 1: The maximum common subgraphs between $G_7$ and $G_1$ and between $G_7$ and $G_2$, and the correspondence among nodes were calculated respectively, using the way of finding the maximum common subgraph proposed in this paper. It was concluded that $mcs(G_7, G_1) = \{c_1^1, c_2^1, c_3^1, \ldots, c_{10}^1, \ldots, c_{18}^1\} = \{c_3^3, c_4^3, c_2^3, c_3^3, c_2^3, c_3^3, c_1^3, \ldots, c_{18}^3\}$, $mcs(G_7, G_2) = \{c_1^2, c_2^2, c_3^2, \ldots, c_{18}^2\}$.

Tab 2. List of Scheduling Tasks in Case 3

| Course No. | Course Title        | Teacher | Students | <l, p, s>-Representation |
|------------|---------------------|---------|----------|--------------------------|
| c_1^3      | Mao Zedong Thought | Wang Lin p1 | s1s2s3   | l1, p1, s1s2s3           |
| c_2^3      | Mao Zedong Thought | Wang Lin p1 | s4s5s6   | l1, p1, s4s5s6           |
| c_3^3      | Principles of Microcomputer | Zhang Kun p2 | s1s2s3 | l2, p2, s1s2s3         |
| c_4^3      | Principles of Microcomputer | Fu Jun p3 | s4s5s6 | l2, p3, s4s5s6         |
| c_5^3      | Database            | Peng Qing p4 | s1s2s3 | l1, p4, s1s2s3         |
| c_6^3      | Database            | Peng Qing p4 | s4s5s6 | l1, p4, s4s5s6         |
| c_7^3      | Fundamentals of Compiling | Xu Hong p5 | s1s2s3 | l4, p5, s1s2s3         |
| c_8^3      | Fundamentals of Compiling | Wu Shuo p6 | s4s5s6 | l4, p6, s4s5s6         |
| c_9^3      | Software Engineering | Ma Hong p7 | s1s2s3 | l5, p7, s1s2s3         |
| c_10^3     | Software Engineering | Ma Hong p7 | s4s5s6 | l5, p7, s4s5s6         |
| c_11^3     | .Net l1             | Shi Qing p8 | s1s2s3 | l6, p8, s1s2s3         |
| c_12^3     | .Net l3             | Li Na p9  | s4s5s6 | l6, p9, s4s5s6         |
| c_13^3     | Operating System l1 | Li Wei p10 | s1s2s3 | l7, p10, s1s2s3        |
| c_14^3     | Operating System l2 | Li Wei p10 | s4s5s6 | l7, p10, s4s5s6        |
| c_15^3     | System Structure l1 | Wang Xiaofang p11 | s1s2s3 | l6, p11, s1s2s3       |
| c_16^3     | System Structure l2 | Wang Xiao p11 | s4s5s6 | l6, p11, s4s5s6        |
| c_17^3     | SCM l1              | Tian Li p12 | s1s2s3 | l9, p12, s1s2s3       |
| c_18^3     | SCM l2              | Tian Li p12 | s4s5s6 | l9, p12, s4s5s6        |
| c_19^3     | Network System l10  | Yan Qiang p13 | s1s2s5s6 | l10, p13, s1s2s5s6    |

Fig 2. A Problem Description $G_3$ of case 3.

Step 2: The similarities between $G_3$ and $G_1$, and between $G_3$ and $G_2$ were calculated using Definition 5.

$$S(G_3, G_1) = \frac{2 \times mcs(G_3, G_1)}{|G_3| + |G_1|} \approx 0.81$$

$$S(G_3, G_2) = \frac{2 \times mcs(G_3, G_2)}{|G_3| + |G_1|} \approx 0.91$$

Considering that $S(G_2, G_3) > S(G_1, G_3)$, Case 3 was more similar to Case 2. So we utilized the so-
olution in Case 2, R₂, to carry out reuse and adjustment. The solution in Case 2 is shown in Tab 3, where * represents scheduled tasks for other schools, and C₀ represents null courses that were available for scheduling.

Step 3: R₂ was reused and adjusted in order to obtain the solution in Case 3.

(A). Reuse: the scheduling tasks in mcs(G₃, G₂) were arranged first. The timetable R₂ was traversed. When cᵢ² ∈ mcs(G₃, G₂), then < cᵢ², tᵢ > → < cᵢ³, tᵢ >; when cᵢ² ∉ mcs(G₃, G₂), then < cᵢ², tᵢ > → < c₀, tᵢ >. In doing so, the reuse result R was obtained. According to the correspondence among nodes in the maximum common subgraph between Case 2 and Case 3 stated in Step 1, the scheduling tasks in Case 2 can be replaced with corresponding scheduling tasks in Case 3, as shown in Tab 3 (in the brackets are the reuse and adjustment results of the cases).

(B). Adjust: courses not included in mcs(G₃, G₂) were arranged then. If cᵢ³ ∈ C₃ \ cᵢ³ ∉ mcs(G₃, G₂), the courses were assigned to R in a conflict-free way and the solution R₃ was obtained. Using the traditional method, that is, to traverse the time slots without scheduled courses in R, it can be guaranteed that when a course was assigned to this time slot, it wouldn’t conflict with other scheduled courses in other classrooms in the same time slot. If all positions are in conflict, rearrange them by backtracking and adjusting the relevant courses. The course not scheduled in this case was cᵢ₅. It was assigned to an unoccupied time slot in a conflict-free way, as shown in red in Tab 3.

| r₀ | r₁ | r₂ | r₃ | r₄ | r₅ | r₆ | r₇ | r₈ | r₉ | r₁₀ | r₁₁ | r₁₂ | r₁₃ | r₁₄ | r₁₅ | r₁₆ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | C₁₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ |
| 2 | C₁₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ |
| 3 | C₁₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ |
| 4 | C₁₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ |
| 5 | C₁₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ |
| 6 | C₁₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ |
| 7 | C₁₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ |
| 8 | C₁₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ |
| 9 | C₁₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ |
| 10 | C₁₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ |
| 11 | C₁₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ |
| 12 | C₁₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ |
| 13 | C₁₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ |
| 14 | C₁₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ |
| 15 | C₁₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ |
| 16 | C₁₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ |
| 17 | C₁₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ |
| 18 | C₁₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ |
| 19 | C₁₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ |
| 20 | C₁₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ | C₀ |

5. Experimental Results and Analysis

To compare the efficiency of different algorithms, we selected cases with 20, 40 and 60 scheduling tasks respectively for testing and compared the results obtained with those of backtracking algorithm[10] commonly used in scheduling software. The hardware platform of the experiment was Intel(R) Core (TM) 3.0GHz CPU, 4.00GB memory and 500G hard disks. And the software platform was Windows 7 and Matlab8.0 programming environment. The experimental data and results are shown in Fig 3. It can be seen from the results that with the increase of the number of courses, the time spent in course scheduling was also greatly increased, but the overall efficiency of CBR was high.
6. Conclusion
In this paper, a new method based on CBR is presented to solve course scheduling problem. For a new course scheduling problem, we can solve it by finding the most similar scheduling in memory cases, reusing and adjusting corresponding solutions. Compared with the existing scheduling method, the proposed method can save a lot of repetitive work and improve scheduling efficiency. How to improve the retrieval efficiency is a difficult point in CBR to solve complex scheduling problems. For complex scheduling problems which require considering the chronological order of courses, the requirement for classroom by the number of students, the requirement for classroom facilities by the course, and the special requirement for time by the course, etc., a breakthrough point is to describe the problem using complex label charts, rather than the graphs used in this paper. By doing so, the focus of retrieval is converted to the measurement of similarity between label charts.

Undoubtedly, some problems in this study warrant further consideration: 1. some teachers are too old and have a heavy teaching load. If we can try our best to avoid consecutive classes during scheduling, we can ensure there is enough time for adjustment and improve classroom efficiency. 2. The proposed algorithm fails to distinguish the type of courses. In fact, it would be better to separate theory courses from laboratory courses, so as to improve students’ enthusiasm for learning. 3. The existing test is only targeted at a certain grade of computer major. Whether it is suitable for other majors remains to be tested and optimized.

Acknowledgments
This work was supported by the National Natural Science Foundation of China (No.61802282) and the Tianjin Natural Science Foundation (No.18JCQNJC70000).

References:
[1] Palma, C.D., Bornhardt, P. (2020)Considering section balance in an integer optimization model for the curriculum-based course timetabling problem[J]. Mathematics
[2] Hossain, S.I., Akhand, M.A.H., Shuvo, M.I.R., Siddique, N., Adeli, H. (2019)Optimization of university course scheduling problem using particle swarm optimization with selective search[J]. Expert Systems with Applications, 127(AUG.):9-24.
[3] Jin,S.H., Yoo, M., Cui, X.N.(2021)Impact of Class Scheduling on Learning Achievement of Computer Engineering Students[J]. International Journal Of Engineering Education, 37(1):270-277.
[4] Mantras, D., Lopez, R. (2005) Retrieval, reuse, revision and retention in case-based reasoning [J]. Knowledge Engineering Review, 20(3):215-240.

[5] Xu, L., Huang, C., Niu, J., Li, C., Wang, X. (2021) an improved case-based reasoning method and its application to predict machining performance [J]. Soft Computing, 1-15.

[6] Duan, J., Jiao, F. (2021) Novel case-based reasoning system for public health emergencies [J]. Risk Management and Healthcare Policy, 14:541-553.

[7] Colander, J. (2004) Case-based reasoning [M]. San Francisco: Morgan Kaufmann[C]:101-150.

[8] Plaza, E., Steve, F., Garcia, P. (1998) a logical approach to case-based reasoning using fuzzy similarity relations [J]. Information Sciences-informatics & Computer Science: An International Journal, 106(1-2): 105-122.

[9] Huang, L., Zhang, S.J. (2016) Research on the Case-based Reasoning Learning Design under the Domain of Learning Science [J]. Modern Educational Technology, 6(1): 67-73.

[10] Yang, Y.P., Song, J.Q. (2011) Application of a Two-level Teaching Management Mode in Course Scheduling of Colleges [J]. Forum on Contemporary Education (Comprehensive Research), (12): 83-84.

[11] Tadrat, J., Boonjing, V., Pattaraintakorn, P. (2012) a new similarity measure in formal concept analysis for case-based reasoning. [J]. Expert Systems with Applications, 39(1): 967-972.

[12] Gao, X.B., Xiao, B., Tao, D.C., Li, X.L. (2010) A survey of graph edit distance [J]. Pattern Analysis and Applications, 13(1):113-129.

[13] Bunker, H., Shearer, K. (1998) A graph distance metric based on the maximal common subgraph[J]. Pattern Recognition Letters, 19(3-4):255-259.

[14] Bunker, H., (1997) on a relation between graph edit distance and maximum common [J]. Pattern Recognition Letters, 18 (8):689-694.

[15] Hans-Christian, E., Matthias, R. (2011) Maximum common subgraph isomorphic algorithms and their applications in molecular science[J]. Computational Molecular Science, 1(1):68-79.

[16] Chen, M.H., Chang, H.Y., Di, Y. (2012) Based on the Curriculum and the Intelligent Analysis Backtracking Algorithm Research [J]. Mathematics in Practice and Theory, 42(16):173-17.