STUDENT TIME TABLE BY USING GRAPH COLORING ALGORITHM

Baki Koyuncu ,Mahmut Seçir
e-mail: bkoyuncu@ankara.edu.tr *e-mail:*cihansecir@gmail.com
Ankara University Computer Engineering Department, 06500, Beşevler, Ankara, Turkey

Keywords: Graph Coloring Algorithm, Student Time Table

ABSTRACT

Graph Colouring Algorithm was used to generate the student weekly time table in a typical university department. The problem was a Node-Point problem and it could not be solved in the polynomial domain. Various constraints in weekly scheduling such as lecturer demands, course hours and laboratory allocations were confronted and weekly time tables were generated for 1st, 2nd, 3rd and 4th year students in a typical semester.

I. INTRODUCTION

Timetabling is the allocation, subject to constraints, of given resources to objects in space-time domain to satisfy a set of desirable objectives as nearly as possible. Particularly, the university timetabling problem for courses can be viewed as fixing in time and space a sequence of meetings between instructors and students, while simultaneously satisfying a number of various essential conditions or constraints.

The model takes advantage of the structural properties of conflict graph instances that arise from university timetabling problems, and is based on the effectiveness of a variety of graph coloring approaches. These are intelligently-ordered and intelligently-searched sequential coloring methods, as well as integer and constraint programming formulations of graph coloring in solving such problems.

Planning timetables is one of the most complex and error-prone applications. There are still serious problems occurring and these problems are repeating frequently. Therefore there is a great requirement for an application distributing the courses evenly and without collisions. Graph coloring algorithm is one of the most used algorithms. [1]

II. TIMETABLING AND GRAPH COLORING

In a typical semester, the courses are required to be scheduled at different times in order to avoid conflict. The problem of determining the minimum number (or a reasonable number) of time slots needed to schedule all the courses subject to restrictions is a graph coloring problem.

Figure 1 illustrates a simple timetabling problem instance in which we have five courses to be scheduled: Physics, Calculus, Electronics, Microprocessors, and Operating Systems.

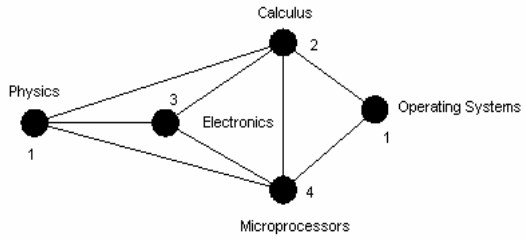

Figure 1 : Colored graph G for 5 courses

In T**able 1** below, an asterisk indicates those pairs of courses that would cause a timetabling conflict if both were scheduled at the same time.

	Physics	Calculus	Electronics	Microprocessors	Operating Systems
Physics		*	*	*	
Calculus	*		*	*	*
Electronics	*	*		*	
Microprocessors	*	*	*		*
Operating Systems		*		*	

Table 1: Course distribution

The cause for potential conflict could be any of the following example restrictions:

a) Courses Calculus and Electronics might be taught by the same professor,

b) Courses Microprocessors and Operating Systems might be taken by the same student.

Given the list of courses Physics, Calculus, Electro- nics, Microprocessors, and Operating Systems along with the set of potential conflicts, we can create a conflict-free timetable of courses by transforming the **Table 1** to the corresponding conflict graph G in **Figure 1**, and finding a minimum coloring.

A vertex in G represents a course, an edge represents a pair of courses that conflict, and a color represents the period in which that particular course is to be scheduled. We see that four periods are required to schedule all the courses without conflict: (G) = 4. According to the coloring, we can schedule courses Physics and Operating Systems for Period 1, and courses Calculus, Electronics, and Microprocessors for Periods 2, 3, and 4 respectively.

With course timetabling, it is often desirable that courses do not "student-conflict" (i.e., those two courses sharing a common student will not be scheduled at the same time).

However, in most situations, a course timetabling solution without some "student conflict" does not exist, due to the fact that university courses are almost always scheduled prior to when students choose their courses. This simply means that a student, when planning his/her semester of courses in which to enroll, may very well have to choose between two or more initially desired courses that are scheduled to take place at the same time, to resolve this "student-conflict".

Welsh and Powell [3] first showed the equivalence of timetabling problems with graph coloring problems, but were not able to solve such problems when preassigned meetings are considered. Welsh and Powell also gave a "largest degree first" coloring algorithm to accompany their graph coloring/timetabling equivalence result.

Timetabling is the scheduling of a set of related events in a minimal block of time such that no resource is required simultaneously by more than one event. In university timetabling, the resources involved may be required by no more than one course at any particular time, are instructors, classrooms, and students. As mentioned earlier, timetabling (in particular, university timetabling) is a practical application of graph coloring.

The minimum coloring problem and the timetabling problem have been classified as NP-hard problems in the general case. This means that it is unlikely that it will be possible to find fast (i.e., polynomial-time) algorithms to solve these problems. In order to find optimal solutions to such NP-hard problems, it is usually necessary to consider all possible solutions to choose the best one.

III. RESULTS

In this study in which graph coloring algorithm is utilized, is designed in a user-friendly method. Ease of use and speed of the application differentiates from current approaches with these powerful features. Results are obtained in a fast manner using the algorithm mentioned. The application developed here can give efficient results in a few seconds while table forming process can take minutes or hours with similar applications.

It is a big problem in universities to create time tables which do not victimize students and instructors. Manually created programs can not deal with these problems despite the great efforts required to form a time table. The purpose of this study is to develop an efficient solution to all these problems.

Application Program runs in five phases. These are data entry, registration, constraint setting, generation of the time table and displaying the time table.

In data entry phase, users enter the student data, **Figure 2,** and course data, **Figure 3**, into the database. A crucial point in course data is the instructor constraint. An instructor may offer more than one course and these courses must not collide in the generated time table.

Student course requirements per semester are entered in the registration phase **Figure 4.** Time table will be generated according to this student data. Student constraints are introduced to provide a collision free planning in this phase.

In constraint phase, **Figure 5**, appropriate time slots in a weekly time table are set for any course or instructor. There is flexibility in defining time slots. It should be kept in mind that some instructors may require certain days off for other activities. Therefore instructor constraints must be set for those days of the week in the application program.

In the registration phase, some students are forced to take some courses which they failed in previous semesters before they take new courses. Registration is not performed if failed courses are not selected. The application program does not allow the students to take the future semester courses. It checks for the total credits taken by a student in a semester and it does not permit the student to exceed this value in registration. Higher priority is given to the courses failed by the student. Registration is not performed otherwise.

Time table for the four years are obtained in table form in display phase **Figure 6**. The application program

exports the generated time table in spreadsheet (.xls) file format. In this way, users can get the print out of the tables.

Finally a Visual menu driven application program is generated by using C++ platform [2]. This program performed the operations according to the records entered into the database. Used algorithm gives fast results in runtime but time table reliability decreases when too many constraints are introduced. The time table can not reach the final state when the constraint number exceeds the optimal value. For example, application program may fail to produce a result when two courses taken by a student are fixed to a timeslot. Collision occurs in such a condition since the application program tries to assign the same time slot for two colliding courses. These kinds of contradictions must be detected and avoided by the user.

The application program also differentiates from current approaches with the spreadsheet (.xls) file export feature. Users can print the program output and desired visual arrangement may be done before printing.

The most important phase in the study is the generation of the time table. Operations performed in this phase may be ordered as follows:

- **a)** Setting the constraints in the course matrix,
- **b)** Detection of the colliding courses,
- **c)** Color assignment to the colliding courses using graph coloring algorithm
- **d)** Placing the courses to the table according to the colors assigned.

Marking the constraints on the course matrix is done according to the student and instructor constraints and in this way colliding courses are detected.

IV. CONCLUSION

The Study has presented a successful approach of automating timetable generation by applying a new technique like graph coloring. Example weekly time tables for 1st, 2nd, 3rd and 4th year students in a typical semester in Computer Engineering department were given in **Figure 6**. The data entrance for the program by the operator was also given as an example for the reader in tabular form in **Figure 3**.

The introduction of this technique together with the known consistency preserver operators not only make convenient and efficient timetables but also decrease the computational time for this NP-hard problem. The only time is spent is entering the course information. Once all the data is entered it takes only a few seconds to display the output programs .In this study, the relevance of university timetabling problems was investigated as a natural and practical application of graph coloring. As a result, a software application was developed to solve the course collisions which were one of the biggest problems. In addition the application program was simple and functional to use.

REFERENCES

1. Andrea Schaerf. A survey of automated timetabling.Technical Report CS-R9567, CWI - Centrum voor Wiskunde en Informatica, 1995.
2. Borland Co, “C++ Builder 6.0 Enterprise Suite Version 6”, Borland Corporation,1983-2000,we page source:http:/www.borland.com
3. WELSH D.J.A. and POWELL M.B. (1967) An Upper Bound for the Chromatic Number of a Graph and Its Application to Timetabling Problems Comp.Jrnl."10, 85-86.

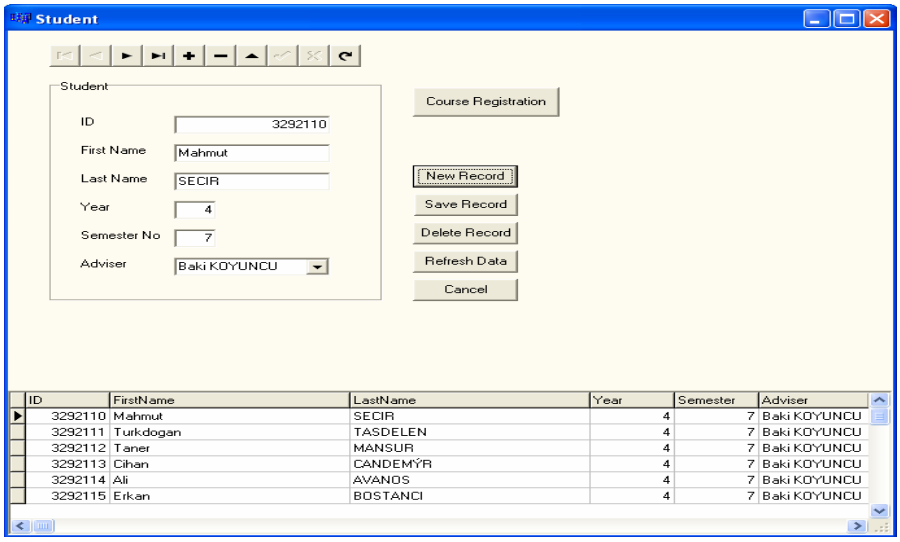

Figure 2 : Student Data Entrance

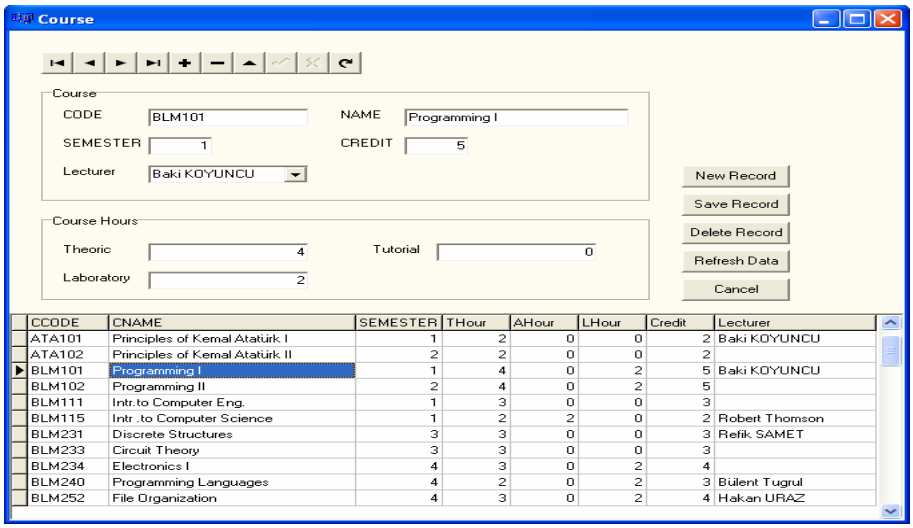

Figure 3 : Course Data Entrance

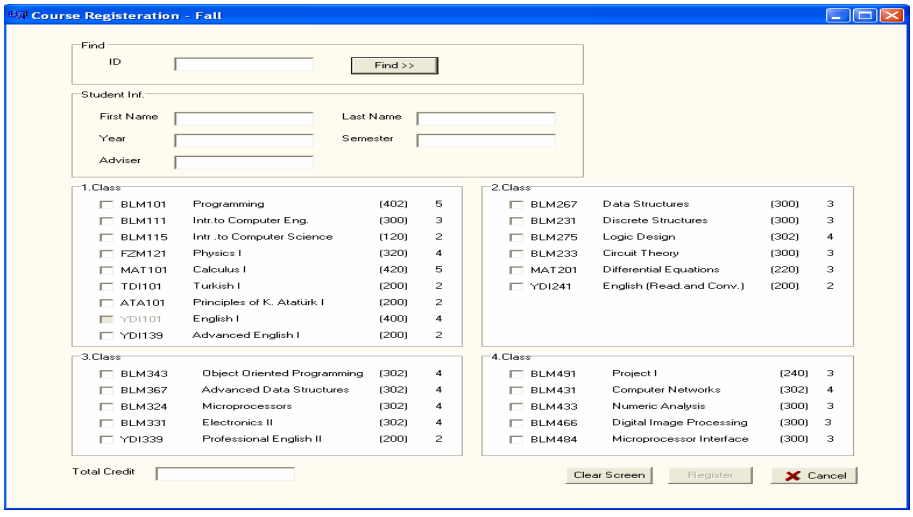

Figure 4: Course Registration Entrance

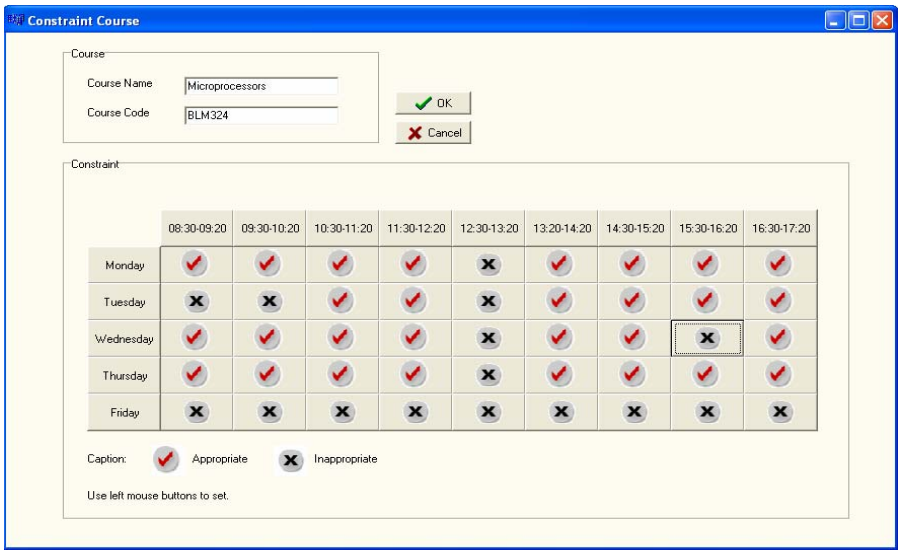

Figure 5 : Constraint Set

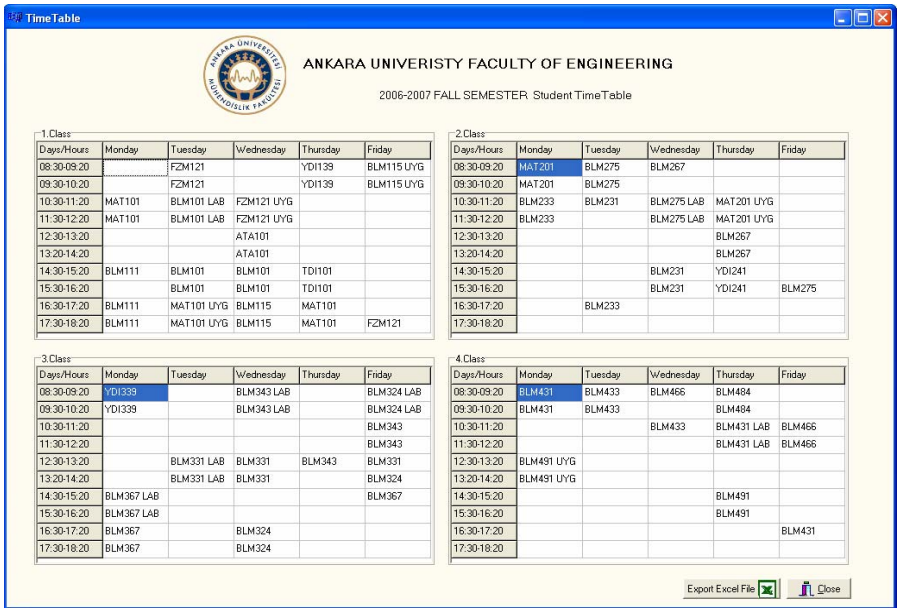

Figure 6 : Resultant Timetable