A Project report on

"STUDENT TIME TABLE BY USING GRAPH COLORING ALGORITHM"

Submitted in partial fulfillment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

By

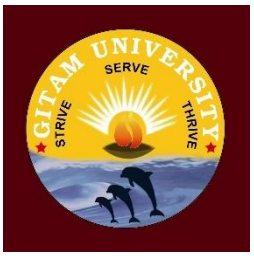
Dhanasetty Abhishek (1210312217)
Vemuri Sai Krishna (1210312263)
Rahul Chitta (1210312249)
Yuva Sai Kamal (1210312261)

Under the esteemed guidance of

T. Sankara Rao M.Tech

Associate Professor

Department of Computer Science and Engineering
GITAM Institute of Technology, GITAM University
Visakhapatnam - 530045
(2015-2016)



GITAM UNIVERSITY

(Estd. u/s 3 of the UGC Act 1956)
Accredited by NAAC with 'A' Grade
Gandhinagar Campus,
Rushikonda, VISAKHAPATNAM – 530045 (AP)

CERTIFICATE

This is to certify that the project entitled "**STUDENT TIME TABLE BY USING GRAPH COLORING ALGORITHM"** is a bonafide work carried out by **Dhanasetty Abhishek (1210312217), Vemuri Sai Krishna (1210312263), Rahul Chitta (1210312249), Yuva Sai Kamal (1210312261)** in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology, in "Algorithms" from Department of Computer Science and Engineering, GITAM Institute of Technology, GITAM University during the academic year of 2015-16. This project work is original and was not submitted earlier for the award of any degree of any institution.

Project Coordinator:

Dr. P. S. Naidu (Ph.D)
Associate Professor,
Department of C.S.E.,
GITAM Institute of Technology,
GITAM University.

Project Guide:

T. Sankara Rao, M.Tech.,(Ph.D.)
Associate Professor,
Department of C.S.E.,
GITAM Institute of Technology,
GITAM University.

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.

We intend to overcome the problems of intractability by producing a spreadsheet type system that the user can guide in an informed and useful way. This gives the user control of the search and the possibility of backtracking where no reasonable solution is found, while still letting the heuristic algorithms do the hard work. Such an approach cannot guarantee an optimal solution but it can guarantee a solution the user is happy with. It is safe to assume that any user addressing a timetabling problem in a University environment has some idea of the timetable required and is qualified to judge whether a solution is suitable or not.

DECLARATION

We hereby declare that the project entitled "**STUDENT TIME TABLE BY USING GRAPH COLORING ALGORITHM"** submitted in partial fulfilment of the requirements for the award of degree of Bachelor of Technology in computer science and engineering. This dissertation is our original work and the project has not formed the basis for the award of any degree, associate ship, fellowship or any other similar titles and no part of it has been published or sent for the publication at the time of submission.

DHANASETTY ABHISHEK (1210312217)
VEMURI SAI KRISHNA (1210312263)
RAHUL CHITTA (1210312249)
YUVA SAI KAMAL (1210312261)

ACKNOWLEDGEMENT

We wish to take this opportunity to express our deep gratitude to all those who helped, encouraged, motivated and have extended their cooperation in various ways during our project work. It is our pleasure to acknowledge the help of all those individuals who were responsible for foreseeing the successful completion of our project.

I am also thankful to **Dr. P. Sanyasi Naidu Ph.D**, Associate Professor, **K. Venkateshwarlu**, Assistant Professor, Department of Computer Science and engineering, GITAM Institute of technology, GITAM University, for Monitoring and evaluating my project work at frequent intervals.

We would like to thank **Mr. P.V. NAGESWARA RAO** Head Department of CSE, GITAM Institute of Technology, GITAM University and express our gratitude with great admiration and respect to our project guide **MR. T. SANKARA RAO M.Tech., (Ph.D)** for their valuable advice and help throughout the development of this project by providing us with required information without whose guidance, cooperation and encouragement, this project couldn't have been materialized.

Last but not the least, we would like to thank the entire respondents for extending their help in all circumstances.

DHANASETTY ABHISHEK (1210312217)
VEMURI SAI KRISHNA (1210312263)
RAHUL CHITTA (1210312249)
YUVA SAI KAMAL (1210312261)

CONTENT

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.

Timetabling is a common example of a scheduling problem and can manifest itself in several different forms. The particular form of timetable required is specific to the environment or institutions in which it is needed.

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.

Existing and Proposed system:

Normally timetable generation is done manually. As we know all institution/organization have its own timetable, managing and maintaining these will not be difficult. Considering workload with this scheduling will make it more complex. As mentioned, when timetable generation is being done, it should consider the maximum and minimum workload that is in a college. In those case's timetable generation will become more complex. Also, it is a time consuming process.

The Timetabling Algorithm is the main component of this project which produces the HTML based timetable sheet as the output. The project takes various inputs from the user such as Teacher List, Course List, Student Set List, Room List, Day List and Timeslot as well as various rules, facts and constraints using web based forms, which are stored in XML based knowledge base. This knowledge base serves as input to the Timetable Generator Algorithm residing on the server machine.

Different problems and formulations:

A large number of variants of the timetabling problems have been proposed in the literature which differ from each other based on the type of institution involved (university or high school) and the constraints. We classify the timetabling problems in two main classes

Course timetabling: The weekly scheduling for all the classes of a high school, avoiding teachers meeting two classes in the same time, and vice versa.

Examination Timetabling: The scheduling for the exams of a set of university courses avoiding to overlap exams of courses having common students and spreading the exams for the students as much as possible.

***Solution approaches**:*

Most of the early techniques (see Schmidt and Strohlein 1979) were based on a simulation of the human way of solving the problem. All such techniques, that we call *direct heuristics*, were based on a *successive augmentation*. That is, a partial timetable is extended, lecture by lecture, until all lectures have been scheduled. The underlying idea of all approaches is "schedule the most constrained lecture first", and they differ only on the meaning they give to the expression 'most constrained'. Later on, researchers started to apply general techniques to this problem.

We therefore see algorithms based on *integer programming*, *network flow*, and others. In addition, the problem has also been tackled by reducing it to a well-studied problem: *graph coloring*.

More recently, some approaches based on search techniques used also in Artificial Intelligence appeared in the literature; among others, we have *simulated annealing*, *tab search*, *genetic algorithms*, and *constraint satisfaction*.

In this paper, we survey the solution techniques, putting the emphasis on the most recent approaches in general, and on Artificial Intelligence techniques in particular.

Notice that we include in our list of techniques also some items, e.g., *logic programming*, that are general tools for the development of the solution, rather than real solution techniques. In those cases, we specify also the technique implemented using the given tool

LITERATURE REVIEW

The literature on timetabling includes several surveys. We now briefly discuss their scope and contribution.

Schmidt and Strohlein (1979) provide an annotated bibliography including more than (200) entries, listing virtually all papers on the field appeared up to 1979.

Junginger (1986) describes the research in Germany on the school timetabling problem. In particular, he describes the various software products implemented, and their actual utilization
by the staffs of the institutions. The paper also describes the underlying approaches, most of which are based on direct heuristics.

de Werra (1985) states the various problems in a formal way, and provides different for, mulations for them. He also describes the most important approaches to the problem up to date, stressing the graph-theoretic ones. We follow mostly his paper for the terminology and the problem formulations of this paper.

Carter (1986) surveys the approaches to the examination timetable problem. He mainly focuses on the approaches based on the reduction to the graph coloring problems.

Corne, Ross, and Fang (1994) provide a survey of the application of genetic algorithms to timetabling. The paper discusses also future perspectives of such approach, and compares its results obtained so far with respect to some other approaches.

Other surveys are given in (Dempster, Lethridge, & Ulph, (1973); Hilton, 1981; Klein, 1983; Vincke, 1984; Ferland, Roy, & Loc, 1986).
The application of computers to timetabling problems has a long and varied history. Almost as soon as computers were first built there were attempts to

use them to solve these problems. The first generation of computer timetabling programs in the early 1960's were largely an attempt to reduce the associated administration work. Soon programs were presented with the aim of fitting classes and teachers to periods.

In 1964 Broder and Cole both presented heuristic approaches to timetabling.
In 1967, Welsh and Powell pointed out the similarity between this problem and the one of colouring the vertices of a graph. Here, the vertices are taken to be equivalent to courses and the arcs between them represent conflicts. Colouring the graph amounts to placing courses in appropriate periods. The algorithm they present is similar to Broder's . They order the vertices according to degree and attempt to colour the graph without using an upper limit on the number of colours. Since 1967 Welsh and Powell's observation has led to many timetabling algorithms based on graph colouring. Indeed, a graph colouring algorithm is an integral part of our system.

Matula, Marble and Isaacson in 1972 presented a *smallest degree last recursive* sequential algorithm. They also presented an interchange which involves looking for a colour swap in vertices adjacent to the one which is currently trying to be coloured when the normal method would introduce a new colour, adding a limited search ability to the algorithm.
The problem of colouring a graph was proved to be NP-complete in Karp's 1972 paper along with the Knapsack problem or the problem of fitting classes in rooms. Two years later, Johnson showed that for any heuristic graph colouring algorithm there are graphs on which it will perform arbitrarily badly.
Grimmett and McDiarmid in 1975 produced a paper giving probablistic values for the chromatic number and largest clique in a random graph against which the results of the heuristic algorithms could be compared.
They also prove that the most basic random order sequential algorithm should on average take twice as many colours as is optimal, a rather more encouraging result.

In 1978, Desroches, Laporte and Rousseau presented HOREX, a computer program that follows a series of steps to give a complete exam timetable. The program firstly finds a colouring of the conflict graph, then

trying to equalize the number of exams in each period. Since there generally need not be any sequence to the exams, the periods are ordered and weekends inserted so as to minimise the number of consecutive
exams. The exams were also allocated rooms to give a full system.
In the next few years a couple of heuristic algorithms emerged which are still largely unrivalled.

Brelaz's *Largest Saturation degree* is a variation on the sequential method, choosing the vertex adjacent to the most already assigned vertices to be the next to colour. Dutton and Brigham however took a different
approach. Taking each colour in turn, the two vertices with the most common adjacents are merged continually until a clique is formed, then all the vertices merged into the same one are coloured identically.

Carter's survey in 1986 brought together the various attempts made at timetabling systems from Broder onwards showing that the graph theoretic approach was by far the most popular.

Meanwhile Sabin and Winter's Case Study suggests that because of the ever-changing nature of educational establishments, a single solution to the problem of timetabling is never likely to be found. More recently, probablistic search methods seemed to have found favour. Hertz and de Werra in 1987 used a Tabu search technique, similar to annealing only with a set of forbidden moves. The final algorithm they use is a combination, starting by consecutively finding colour sets as in an Almost Maximal Independent Set (AMIS) algorithm, then when there are only a certain number of unassigned vertices left, by colouring the vertices and successively swapping ones that are adjacent to other vertices of the same colour. There is also a Web page and a mailing list for the timetabling community, which includes bibliographies and papers

ALGORITHM:

Algorithm to color a planar graph using four colors

Input: A plane graph G in adjacency list form on n vertices.
Output: A coloring c of G on four colors.

```
1: procedure RingAnalysis (Ring R, Graph G)
2:      Let E, I be the subgraphs of G with E on the exterior of R and I on the interior
3:      Let H1 = G - E
4:      Triangulate H1
5:      c2 ← Color(H1)                                          |V (H1)| < |V (G)|
6:      Let H2 = G - I
7:      Triangulate H2
8:      c2 ← Color(H2)                                          |V (H1)| < |V (G)|
9:       Kempe c1 and c2 to match on R and let c be the resulting coloring
10:     |V (H1)| + |V (H1)| ≤ |V (G)| + 6
11:     return c
12: end procedure
13: procedure Color(Graph G)
14:     if n ≤ 4 then
15:             Assign each vertex a di_erent color, and return c.
16:   end if
17:     if G has a face F of degree at least 4 then
18:             Let x; y be two non-adjacent vertices on F.          They exist by planarity
19:             Let H be a graph formed by merging x and y into a vertex z in G
20:             c = Color(H)
21:             c(x); c(y)←c(z)
22:     return c
23:     else if G has a vertex x with deg(∞) ≤4 then
24:             Then, the cycle C surrounding x is a k-ring.
25:     return RingAnalysis(C,G)
26:     else            G has minimum degree 5, it contains a configuration in U
27:             Let K be a configuration of U weakly contained in G.
28:          if K is not strongly contained in G then
29:             Let C be a k-ring for k ≤4 or a non-trivial 5-ring on G.
30:             return RingAnalysis(C , G)
31:     else                                                    (K is strongly contained in G)
32:             Let r be the ring size of K                     r ≤14 since this is true of any
con_guration of U
33:             Let H = G - V (G)
34:             Let T ⊆ E(H) be a set of edges according to the de_nition of reducibility
```

35: Let J be the graph obtained by contracting T in H.
36: c0 Color(J) (c0 is a coloring of J)
37: Kempe the vertices of V (H) until we find a coloring c that extends into K.
38: end if
39: end if
40: end procedure
41: return Color(G)

Example

Suppose we have twelve exams which need to be timetabled and that they have the following conflict graph:

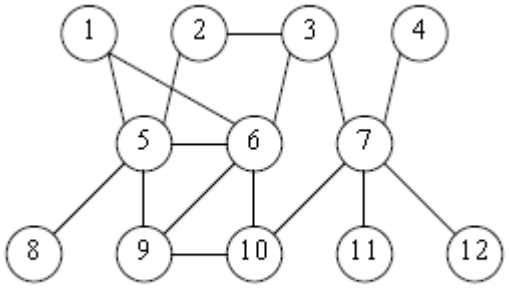


Figure 1

This graph may be coloured using the algorithm. Both vertices 5 and 6 have degree 5 so start with vertex 5. Only vertices 3 and 10 have any common adjacent vertices with 5, both having 2, so chose 3 to merge with 5 becoming the vertex **V**1.

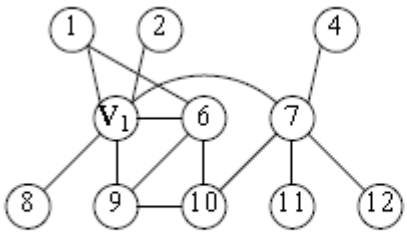


Figure 2

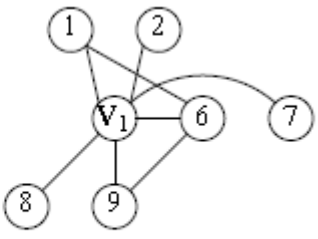


Figure 3

Continuing like this, vertex 10 is selected next, then vertices 4, 11 and 12 at which point, all vertices are adjacent to **V**1 (See Figure 3). Now **V**1 is deleted

and the process repeated, eventually giving three colour sets (denoted by three different shapes). The Final colouring of the graph is:

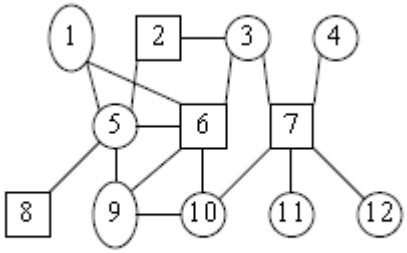


Figure 4

Thus, given no other constraints, the timetable may be constructed in three periods, one period per colour.

In this unlikely and idealistic scenario, we would have finished all we need to do. However, this is not the case in most situations. In a simple case, as in this example, the algorithm is likely to pick out an optimal solution. For larger timetables or graphs this is much less likely to be the case, since the graph colouring problem is NP-complete. However, the algorithm will at least give a reasonable colouring if not an optimal one. Either way, we must have a way of ensuring that all the exams have a room in which they may be held.

SYSTEM REQUIREMENTS

Hardware Requirements

- Processor : Intel or AMD processor computer
- RAM : 256 MB or more
- Hard Disk space : 1 GB or more

Software Requirements

- Operating System : Windows XP SP3 or above
- Environment : Visual Studio 2015
- Language : Java
- Backend : Oracle 10g

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.