IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 18, Issue 6, Ver. I (Nov. - Dec. 2016), PP 39-43
www.iosrjournals.org

Scheduling Algorithm for University Timetabling Problem

Francis K.[1], Manga I.[2] Sarjiyus O. [3]
[1]*(Department of Computer Science, Adamawa State University Mubi, Nigeria)*
[2]*(Department of Computer Science, Adamawa State University Mubi, Nigeria)*
[3]*(Department of Computer Science, Adamawa State University Mubi, Nigeria)*

Abstract: *Scheduling for timetabling is one of the challenges faced by most Universities in developing countries. In this research work, consideration is made in developing of a scheduling algorithm capable of providing solution to a timetabling problem in Universities. Hence, a practical approach is created by incorporating Local Search Procedures into Constraints Programming for generating lecture timetable which was tested on some universities timetabling problems and was found to provide a better solution than most existing methods.*
Keywords: *Scheduling algorithm, Timetabling, Search, Lecture, Constraints.*

I. Introduction

University timetabling problem is a constraint satisfaction problem mostly created manually in many institutions due to its inherent difficulties of finding a solution that does not violate a set of given constraints. A diverse variety of university timetabling problems exist, but three main categories have been identified: school, examination and course timetabling [9, 8, 1] respectively.

Timetabling problem belongs to the NP-hard class combinational optimization problem whereby its computational time grows exponentially with an increase in the number of variables involved [5]. Different methods have been proposed ranging from Local Search Procedures [9] to Constraint Programming [1]. Although the Local Search Procedures are good for optimizing initial feasible solution, its major setback is it does not take into consideration hard constraints and especially finding an initial feasible solution. Constraint Programming which has the advantage of identifying initial feasible solution does not consider weak constraints hence creates a possible problem of improving the initial feasible solution. A blend of Local Search Procedure and Constraint Programming have the advantage of being more efficient in terms of taking into consideration both strong and weak constraints, finding and modifying an initial feasible solution to overcome the initial setbacks of Local Search Procedure or Constraint Programming when used alone.

II. Review of Literatures

Timetabling problem comes up every year in educational institutions, which has been solved by leveraging human resource for a long time. The problem is a special version of the optimization problems; it is computationally NP-hard [5]. As a result, only the major inevitable conditions can be considered during the manual arrangement process. However the manual process takes into account soft constraints whereas automated system might not consider them. This is a major shortcoming of automated systems, wherein they don't give due importance to human feelings. If we want to have a system which works like humans, it would be necessary to make it aware of the soft constraints of humans. Hence we propose a method to add this aspect in the timetable generation to achieve an artificial intelligent computer system more close to human. We propose to use a mechanism to mine rules which can later be incorporated in the automated system to draw its attention to the soft constraints.

A number of efforts have explored how to reap better performing timetables such as University Timetabling [1, 8, 9, 7], Examination Timetabling [2], and UniTime [3] respectively. Furthermore, there exist many problem solving methods, which usually use the concepts of standard optimization algorithms such as Backtracking, Evolutionary Algorithms or Constraint Logic Programming [4, 5, 6].

III. Methodology

The algorithm is the main component of our research which generates the HTML format as an output. Various inputs from the user are required, such as lectures details, course details, the semester, lecture venues and their capacities, working days and timeslots as well as various rules (Constraints), which are stored in an XML format and serves as an input to our Timetable Generator Algorithm as shown in fig.1.

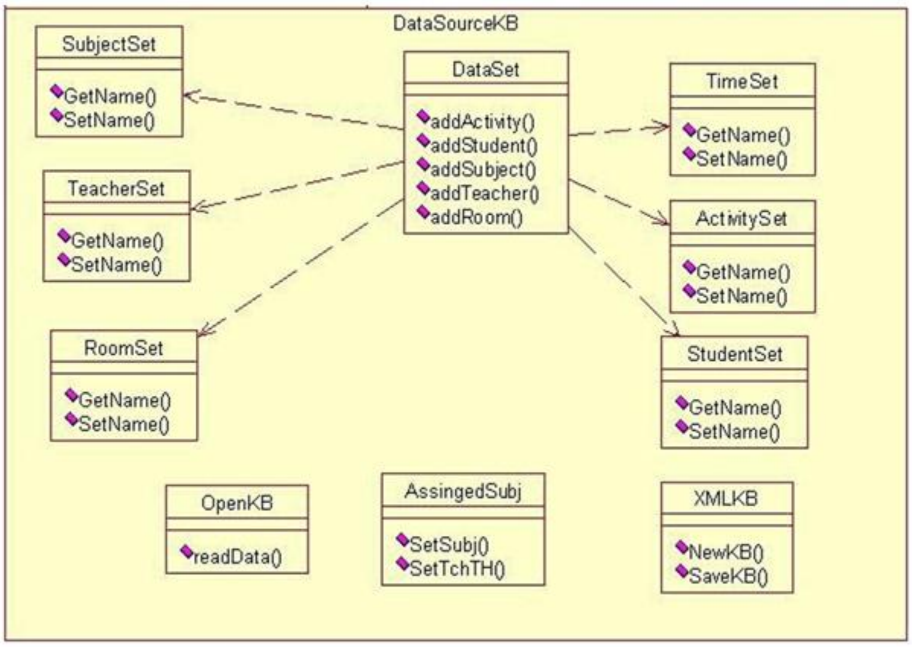


Figure 1: E-R Diagram for Timetabling Data Source

Packages: Packaging allowed us to break up the many large number of objects into related groupings to provide scope and division to classes and interfaces.

IV. Use Case Diagram

A use case is made up of a set of scenarios (such as: Login, Data Management, Rules settings etc.). Each scenario is a sequence of steps that encompass an interaction between a user and a system.

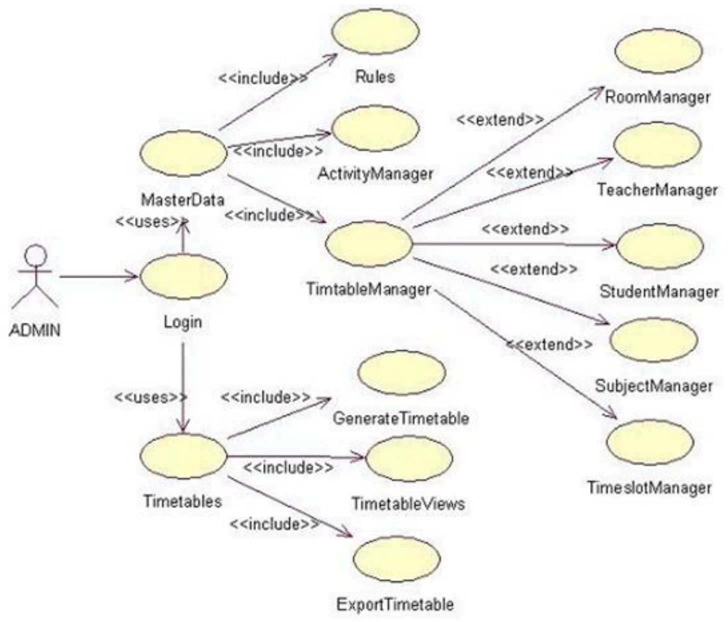


Figure 2: Use Case Diagram for Timetabling Scheduling

V. Timetable Generating Algorithm

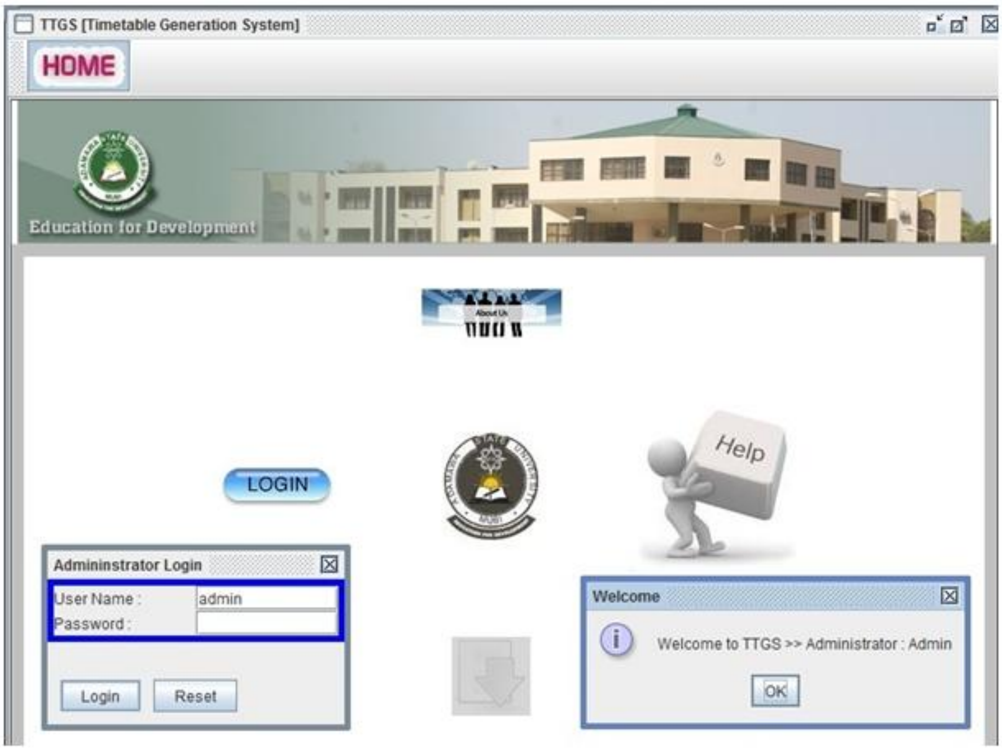
The algorithm considers the entries in the requirement matrix one after the other allocating to each a suitable lecture hour. During activity generation, if an allocation has been made at a certain time t1 for venue v1 and lecture j1, then one is subtracted from the integer in row *k* and column v1 of the current requirement matrix, the current lecturer availability matrix is marked true at row j1 and column t1 and the value of j1 is inserted at a point on the *k*th row and the *g*th column of the activity matrix. This process is illustrated in *fig.3* below.
The algorithm performs four basic operations. These include,

1. Initiate storage matrices
2. Perform allocation by checking resource availability
3. Checks If any resource exist
4. Else it places the activity at current timeslot and day.

```
Activity generator:  For c: = 1 step until total_lecture_venues do
                         For n: = 1 step 1 until number_of_hours do
                             Begin allocate_one_lecturer;
                                 If allocation_fails then
                                     Begin alter_conditions;
                                         Copy initial_matrices;
                                         Goto activities
                             End
                     End;
```

Figure 3: Timetabling Main Pseudo code

VI. Results

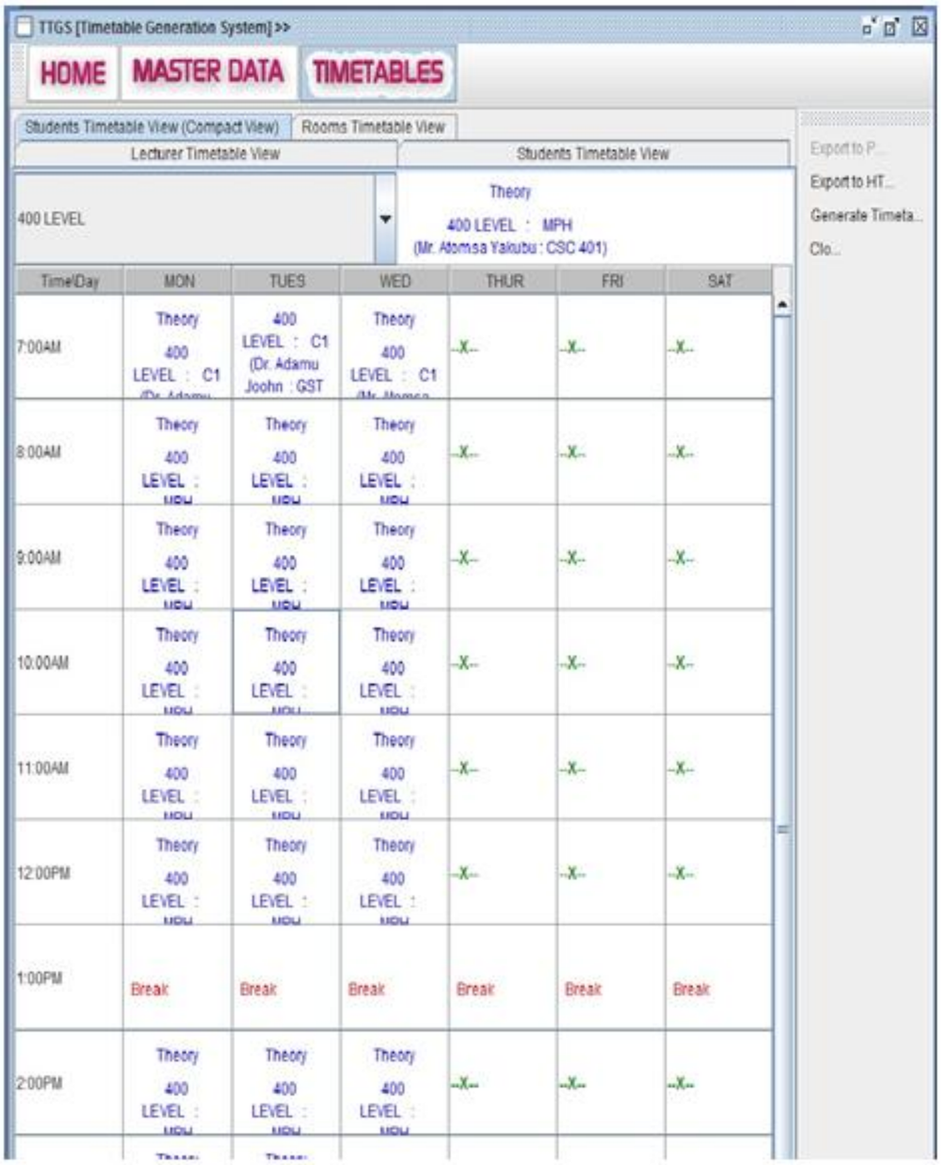


User Name : admin
Password :

Welcome to TTGS >> Administrator : Admin

Figure 3: The Home Screen View

The table below shows a sampled generated timetable using the algorithm developed, compilation took approximately 5 seconds. Execution time varies considerably with the difficulty of finding a possible feasible solution.


TTGS [Timetable Generation System] >>

HOME | MASTER DATA | TIMETABLES

Students Timetable View (Compact View) | Rooms Timetable View

Lecturer Timetable View | Students Timetable View

400 LEVEL

Theory
400 LEVEL : MPH
(Mr. Atomsa Yakubu : CSC 401)

Export to P... | Export to HT... | Generate Timeta... | Clo...

Time\Day	MON	TUES	WED	THUR	FRI	SAT
7:00AM	Theory 400 LEVEL : C1	400 LEVEL : C1 (Dr. Adamu Joohn : GST	Theory 400 LEVEL : C1	--X--	--X--	--X--
8:00AM	Theory 400 LEVEL :	Theory 400 LEVEL :	Theory 400 LEVEL :	--X--	--X--	--X--
9:00AM	Theory 400 LEVEL :	Theory 400 LEVEL :	Theory 400 LEVEL :	--X--	--X--	--X--
10:00AM	Theory 400 LEVEL :	Theory 400 LEVEL :	Theory 400 LEVEL :	--X--	--X--	--X--
11:00AM	Theory 400 LEVEL :	Theory 400 LEVEL :	Theory 400 LEVEL :	--X--	--X--	--X--
12:00PM	Theory 400 LEVEL :	Theory 400 LEVEL :	Theory 400 LEVEL :	--X--	--X--	--X--
1:00PM	Break	Break	Break	Break	Break	Break
2:00PM	Theory 400 LEVEL :	Theory 400 LEVEL :	Theory 400 LEVEL :	--X--	--X--	--X--

Table 1: A generated timetable

The generated timetable has different views for easy access to a particular view, ranging from lecturer view, students view, lecture hall view and the general view.

VII. Conclusion

In this paper, a presentation is made of a blend of Constraint Programming and Local Search Procedures for the solution of timetabling problems. The aim was to create an algorithm capable of generating an initial feasible solution whilst taking into consideration both weak and strong constraints for generating universities timetable. The timetables generated are in HTML/PDF formats which can be easily uploaded to the school's website. The proposed blend of Constraints Programming and Local Search Procedures were also found to perform better than most existing methods in terms of considering weak and strong constraints respectively.

References

[1]. S. Abdennadher, & M. Marte (1999). University timetabling using constraint handling rules. *Journal of Applied Artificial Intelligence*, 12-22.

[2]. G. Brassard, & P. Bratley (1996). *Fundamentals of Algorithmics.* New York: Prentice Hall.

[3]. E. Burke, D. Elliman, & R. Weare (1993). *Extension to a University Exam Timetabling System.* Chicago: Adventure Works Press.

[4]. J. Chen (2014, January 5). *Unitime Corporation* . Retrieved from Unitime Corporation web site : http://www.unitime.org

[5]. R. Doug (2003). *Extreme Programming Refactored.* Chicago: Apress.

[6]. S. Fred (1997). *Concurrent Programming.* New York: Springler.

[7]. M. Garey & G. Johnson (1979). *Computers and intractability, a guide to the theory of np-completeness* . New York : New York Press .

[8]. A. Hertz & D. Werra (1990). The Tabu search meta heuristic: How we use it. *Annals of Mathematics and Artificial Intelligence,* 111-121.

[9]. E. Horowitz & S. Sahni, S. (1984). Fundamentals of Computer Algorithms. *Computer Science Press*, 4-17.

[10]. R. Karp (1976). *The probabilistic analysis of some combinatorial search algorithms.* New York: New York Times .

[11]. B. Romero (1982). *Examination Scheduling in a Large Engineering School.* Madrid, Spain : A Computer-Assisted Participative Procedure.

[12]. A. Schaerf (2006). *Tabu search technique for large high-school timetabling problems.* Netherland: The Netherlands.