Paperless Master Timetable Scheduling System

Paul Godwin Daniel

Dr. Alimi Olasunkanmi Maruf

Department of Computer Science
Faculty of Science
Kebbi State University
Kebbi, Nigeria

Dr. Bala Modi
Department of Computer Science
Faculty of Science
Gombe State University
Gombe, Nigeria

Abstract

Timetable generation is a very difficult task. It is a time consuming, and arduous process. To manually generate a timetable, takes a lot of time, effort, and manpower. However, a timetable scheduling system is designed for different purposes such as: organizing lectures in higher institutions, private organization, airlines, bus station, etc. This paper tries to minimize the difficulties in generating a timetable for academic purposes, using Logarithmic algorithm, to be precise the modified Quicksort algorithm for its design. The algorithm is designed to eradicate collisions on the timetable, and to run in parallel. This helps to minimize the human errors, improve the accuracy of the timetable scheduling process as well as the computation time. To this extent, a Paperless Master Timetable Scheduling System (PMTSS) was proposed and implemented tin this paper to achieve the said requirements. The system can be installed on any android or windows platform. The content of the design also provides exciting services such as: Live Chat for Administrators, Lecturers and Students, Timetable alertsetc. The paperless timetable scheduling system involves dynamic system utilization and modifications.

Keywords*: Timetable, Scheduling, Logarithmic Algorithm, Quicksort Approach, Constraints*.

Introduction

Timetable scheduling system approach is a general problem across the world, particularly using the local system of manually preparing timetable. The procedure for scheduling timetable appears to be a very hard and difficult task. Most academic institutions do produce timetables either once or twice in a year according to the administrators. In Nigeria, most universities, polytechnics, colleges and secondary schools schedule their various timetables manually using the conventional paper and pencil based approach. Such an approach is very tedious, complex and often leads to clashes in lecture time slots and venues.

However, in planning for timetables, considerations have to be made such that each lecture period must have at least one lecturer or professor, a time slot and venue. This is generally considered to be a highly constrained and hard problem to solve. Another challenge is how to generate an optimal timetable solution. To this end, approaches based on evolutionary algorithm using problem-specific domains, heuristics and context-based reasoning have been developed by various researchers. Parallel frameworks such as the PTMSS and genetic artificial immune networks have also been developed all in a bid to produce optimal solutions for timetable generation. The PMTSS implemented in this paper uses tools such as: XHTML, HTML6, CSS6, PHP, JAVASCRIPTS, JQUERY, MYSQL and Tomcat Apache. Furthermore, there are quite a number of approaches that have been used to create various systems with similar features, but not with exact functions as the PMTSS. The remainder of this paper is organized thus: A brief summary of related works is described in the section following the introduction. The section proceeding from the related works discusses the problem statement.

After the problem statement section is the section that gives a brief background of this paper. The proposed system under consideration follows immediately after the background section, while the methodology used in this paper is discussed afterwards. The result and discussion follows immediately after the methodology and the paper rounds up with the section for conclusion.

Related Works

A considerable number of studies have been carried out in the area of timetable scheduling. Various methods including the methods of operation research, human-machine interaction, and artificial intelligence (Andrianto, 2014). Shrinivasan et'al. (2002) explains thata feasible lecture timetable for a large university department is a challenging problem faced continually in educational establishments. The paper presented by Shrinivasan et'al. (2002) was an evolutionary algorithm (EA) based approach for solving heavily constrained timetabling problem. The approach uses a problem-specific chromosome representation. Also, Heuristics and context-based reasoning have been used before to obtain feasible timetables within a reasonable computing time. Abramson and Abela(1992) proposed a genetic algorithm based approach for solving the problem of time table scheduling.
However, the problem with the approach by Abramson and Abela (1992) is that it was too slow, but a good speed up was later achieved. This was made possible by exploiting parallel processing as a technique for solving complex problems and for searching large problem spaces. As such, there is a great requirement for an application that distributes courses evenly without collisions while exploiting such a process of execution.Saviniec et'al. (2018) showed how two different parallel frameworks are used to design parallel heuristics for a high school timetabling problem. The process deployed strong exchange of information among threads to improve the search processes.Kheiriet'al.(2016) used a hyper-heuristic search strategy for solving the computationally hard problem. A different approach was presented by Bhaduri (2009), where an evolutionary technique was shown to solve the time table scheduling with mixed success problem by using methods such as: Genetic Algorithms (GAs), Evolutionary Algorithms (EAs) etc.

The problem of lecture time table scheduling was well stated and solved with genetic algorithm using mimetic hybrid algorithm, and genetic artificial immune network (GAIN) and the result was compared with that obtained from that of a GA. Results showed that GAIN was able to reach an optimal feasible solution faster than that of GA. Also, a hybrid approach produced by combining the concept of Bee colony Optimization (BCO) and Firefly Algorithm (FA) collectively termed as BCFA was presented by Sahooet'al.(2017). GAIN was designed to findan optimal solution for a course time table schedule.

Bagulet'al.(2016) also presented a solution to the various problems of venue, class slots and teacher clashes, which was achieved by simply automating the manual processes and choosing the optimal solution from a set of possible solutions. This was achieved using a heuristic algorithm that takes values and manages the constraints and resource scheduling one by one. This paper was aimed at developing a simple, easy and collision free lecture timetable scheduling solution by implementing a slight variation of the solution provided by Saviniec et'al. (2018). Also, the solution provided in this paper is simpler, more efficient, and capable of automatically generating good quality timetable schedules. This is made possible by using a simple Logarithmic algorithm, precisely the modified quick sort algorithm for assembling parallel collision free frameworks (Nanda, Pai, and Gole, 2012). The algorithm is designed to make the process of search and assembly much faster, compared to most other approaches. This is especially useful in the process of generating the lecture slots based on the timetable indexes. The scheduling index process incorporates a unique collision avoidance algorithm that ensures that no repetitions are permitted during the generation of selection timetable slots indexes.

Problem Statement

The existing issues with traditional timetable generation includes: difficulty in execution, time consuming, and is considered an arduous process. When generating a manual timetable, lots of effort and man power is needed, and such timetables in most Nigeria Institutions are usually prone to human error. Furthermore, a major problem that is associated with the manual lecture timetable system is the high rate of clashes in lecture times and venues. Amending an already generated timetable requires the scheduler to recreate the schedule manually over and over again. This certainly creates a series of retracing which is usually difficult to figure out or resolve as the case maybe. To overcome these problems, concerned institutions need an efficient automated, feasible and competent timetable scheduling system. Such a system should be capable of satisfying all the soft and hard constraints and conditions previously highlighted.

For instance, the same faculty lecturer taking two more courses cannot be assigned the same room, venue and time slot for the same lectures. Concurrently, two different courses which are to be delivered to the same students or group of students should also not be allowed. As such, there is a major requirement for an application appropriate lecture timetable without variation, such that collision in the scheduling process is totally eliminated. To this effect, the PMTSS system overcomes these problems particularly by saving more processing time and also by eliminating the traditional error-prone manual processes involved.

Background

The Logarithmic algorithm used in implementing the said system is the modified Quick sort algorithm. The algorithm is based on the Divide-and Conquer approach. The process is split into 6 parallel processes that are each running simultaneously. However, since there are 6 parallel processes, the best-case performance is $\Theta(6n\log n)$, which reduces to just $\Theta(n\log n)$, while the worst-case performance is $\Theta(6n^2)$, which also reduces to $\Theta(n^2)$.

- **Definition**: Suppose there are *P* processes running in *n* time to generate the timetable. Then we can model the problem as follows:

Condition 1: $P_i \neq P_{i+1,}$ where $_{i=0,1,2,..n}$

Condition 2: $P_n \neq \emptyset$.

If and only if Condition 1 and Condition 2 are met, then we obtain Equation 1 as follows:

$P_n = \{P_{i-5,} P_{i-4}, P_{i-3}, P_{i-2}, P_{i-1}, P_i\}$ ……(1)

where, $_{i=0,1,2,..n}$

Each P_i is running independently and is generating specific days of the week having 8 separate periods between 8am – 4pm. To modify the algorithm, we first generate the timetable for both single and double lecture periods in order to eliminate clashes in both courses and venues to be allocated as seen in Algorithm 1. The algorithm stores the created slots and merges them together using the carefully designed function called *mergeTable(D, S)*.

- **Algorithm** 1: Generation of timetable

Create storage arrays *D and S.*
Create Arrays *Dp* and *Sp* for Double and Single periods of
lecture times respectively. Each array is of *p* periods (am-pm)

---

begin

a. create an array of *p* empty periods, $p = \emptyset$.
b. check if $P_i = P_{i+1}$ and $P_{i+1} < P_n$,
   $where_I = 0,1, 2, \ldots, n$.
   then:
   generate Dp and *Sp* and store in *D* and *S*
respectively.
c. Query from *D* and *S* to generate the timetable by calling the *merge Table (D, S)* function.
end

Proposed System

This section describes the proposed system structure, starting with the aim as follows:

Aim of the System

- Developing a paperless timetable system semester courses and examinations, together with other related scheduled administrative operations.
- To develop a fast, trendy, unique and easy to use application that is deployable and efficient.
- To avoid lecture and venue clashes.
- To provide an interface that supports other related activities required but not necessarily related to timetable scheduling processes such as: calendar of events, news feed etc.
- To provide an interactive chat forum for the administrators, lecturers and students alike in real-time.

There are a number of advantages and disadvantages considered while building the PMTSS scheduling system are as follows:

Advantages

- The paperless timetable system reduces the human stress attributed to the manual process of creating timetables.
- The automation provides Subjects (Course Title), Course Code, Department, Faculty, Lecturer, School Years, Semester's, Room, Venue, Time, and Live Chats (for Administrators, Lecturers and Students). In addition, Timetable Alerts for Students and Lecturers is provided by the PMTSS.
- The system eliminates all manual paperwork.
- The system also drastically reduces the man power and time consumed while executing the timetable scheduling tasks.
- The PMTSS can be used either as a stand-alone or real-time system.

Disadvantages

- Lack of acceptance and awareness of the scheduling systems such as the PMTSS at most Nigeria Universities, Polytechnics, and Colleges is partly attributed to ICT phobia and other related causes.
- Increased running costs of ICT infrastructure such as bandwidth and power is a real source of concern.
- The Applications are usually customized and copyrighted, making distribution and reuse difficult.

Components Functionality

The PMTSS allows for functionalities considered to be appropriate for implementation especially in cases where the description of the functionality is not adequate. In such cases, generally appropriate assumptions are made to the following rules as follows:

Student's Classification Rules

The rules are applied for each semester in each academic year thus:

- Each class may have an interval of a maximum of 2hrs per slot.
- At least a maximum of 7 to 8 hours of lectures is permitted in a day.
- Time for practical work is also inclusive.
- Lecture time slots can be allotted and changed with ease by each course Head of Departments (HODs) often based on request by either lecturers or students.
- Courses are to be allocated in any of the following categories:
  - Core (mandatory) courses.
  - Electives (optional) courses.
- Maximum and minimum number of courses to be offered by students should be specified.

Lecturer Classification Rules

Lecturer requirements are applied thus:

- Lecturers may reschedule already allotted time slots by agreeing and arranging with their students.
- A lecturer is restricted to no more than 3 lectures in a day, and no more than 2 hours at a stretch for every taught course the deliver. The extra hour should be allotted a separate time slot. However, free slots can still be utilized for extra lectures by a willing lecturer as agreed with their students.

Administrator Classification Rules

These rules are determined by both HODs and administrators thus:

- A student may produce a copy of the timetable.
- Emails are to be sent to both students and lecturers containing the timetable.
- Break periods are to be properly captured.
- The timetable should not contain clashes.

Methodology

The architecture of the PMTSS consists of data entering processes, data structure of the Course Title, Course Code, Department, Faculty, Lecturers, School Years, Semester, Room, Venue, Time Intervals, and Time Slots etc. The data input method is defined by the following processes:

Data Input Process

This defines the type of data that is inserted, retrieved and updated from a database as follows:

- Lecturer: define details of information that describe the lecturers involved.
- Course Title: Title that describes the course.
- Course Code: Code that describes the course.
- Department: Is the name of a given department.
- Faculty: Is a specified faculty for each department.
- School Years: The duration of the course of study such as: 4, 5, 6 or 7yrs.
- Semester: The semester in question i.e. first semester and second semester.
- Room: Is the lecture room or hall name or any related description.
- Venue: Describes building or location where the room is situated.
- Time Slot: The time allotted for each lecture.
- Time Interval: It is the time that a lecture is expected to last.

System Constraints

A constraint is a condition that a solution to a problem must satisfy. Two major constraints are stated in this paper namely: hard and soft constraints.

- **Hard Constraints:**

Hard constraints are those constraints which set conditions for the variables that are required to be satisfied.

- Duplicate lectures must be eliminated.
- Experiments must be held in a Laboratory and not in lecture classes.
- Lecture rooms must not be booked twice at the same time.
- Venue of lectures should not be doubly-booked during the same period to avoid clashes.
- All lecture venues and rooms must be scheduled once not twice.
- A lecture room must be large enough to contain all students before allocation.

- **Soft Constraints:**

Soft constraints are constraints that are easier to adjust as follows:

- Lectures for each given course should be evenly spread within the week.
- Departmental courses borrowed from different departments must be evenly distributed.
- Break periods must be allocated slots first before other courses.
- Faculty general courses must be allocated slots first before departmental courses.

Violation of Validated Constraints

These are constraints that are required to be satisfied, so that there will be assurance of validated timetables. However, this defines process such as;

- No more than 2 consecutive lectures by the same lecturer in any given period.
- The vital constraint is that both lecturers and their students can not appear in more than one lecture venue at the same time.

System Requirement and Specification

The system requirement and specification (SRS) document is summarized and laid out in a template as seen in Table 1. Some essential flow requirements are entered into it to show how to use the template. Care must be taken to ensure that even the smallest and most trivial requirements are written. Such requirements would help in validating the system during testing. The following are the specific software and hardware requirement:

- ***Software Specification Requirements***:

User Interface: PHP, XHTML, CSS, JQUERY
Client-side Scripting: JavaScript, PHP Scripting
Programming Language: PHP, ASP
IDE/Workbench/Tools: Adobe Dreamweaver CS6,

NetBeans IDE.
Database: MySQL (MySQLite, Optional, Oracle 10g)
Server Deployment: Apache/2.2.4, Tomcat Apache.

- ***Hardware Specification Requirements***:

Monitor: 17 inch LCD Screen (optional).
Processor: Pentium 3, 4, dual-core, Intel, Core i7.
Hard Disk: 500GB or 1, 2 or 4 Terabyte.
RAM: 4GB or more.

- ***The* System Design**

The system has a major component that forms the basis for the design. These components are: admin account panel (i.e. for registration and login authentication), user login panel, and timetable data entry with full details of lecturers for the scheduling process. The workflow of the system enables the user to have easy understanding of the process for creating the timetable. However, the system provides a user with a robust graphical user interface (GUI). This is a simple interactive interface for entering the details concerning a course such as: course title, course code, venue, rooms, department, and faculty, lecturer, school years, semesters, time slots, and lecturer information. The system administrator provides the users access to get registered, and have access to the system. Note that the scheduler is the assigned officer who collates all data for entries, from various sources and afterwards generate the timetable.

Table 1: Detailed Design Breakdown

S/N	Requirement	Required or Essential	Description of the Requirement	Remarks
SRS1	Admin account panel	Required	Registration and login authentication.	Admin access only.
SRS2	System user login interface	Required	A login detail access point.	The login access is assigned by the admin.
SRS3	The timetable scheduling process will contain all the datasets stated for the design process	Required and Essential	The timetable will provide full details on how the timetable will be designed and its functionality.	Timetable will be generated automatically without any conflicts or clashes.
SRS4	The scheduling process provides a list of details about courses offered, rooms venue and time slots for the current and next semester.	Essential and Required	The list of courses offered in all the departments in the current semester should be available for the students to select from and register.	Admin Interface is required for this field, to provide an update to the list of courses.
SRS5	Live chat feeds	Essential	An interactive feed where an admin can chat with lecturers and other staff and students concerning the timetable and other requirements.	A unique UI (user Interface) will have to be provided for this feed.
SRS6	The system provides help screens.	Essential	Help about the various components and features of the system should be provided with Q&A format.	The timetable policy should also be part of the help.

.

- ***Data* Flow Diagram**

The flow diagram shows the user interface design of the **Timetable Scheduling Process** and how the timetable is to be generated.

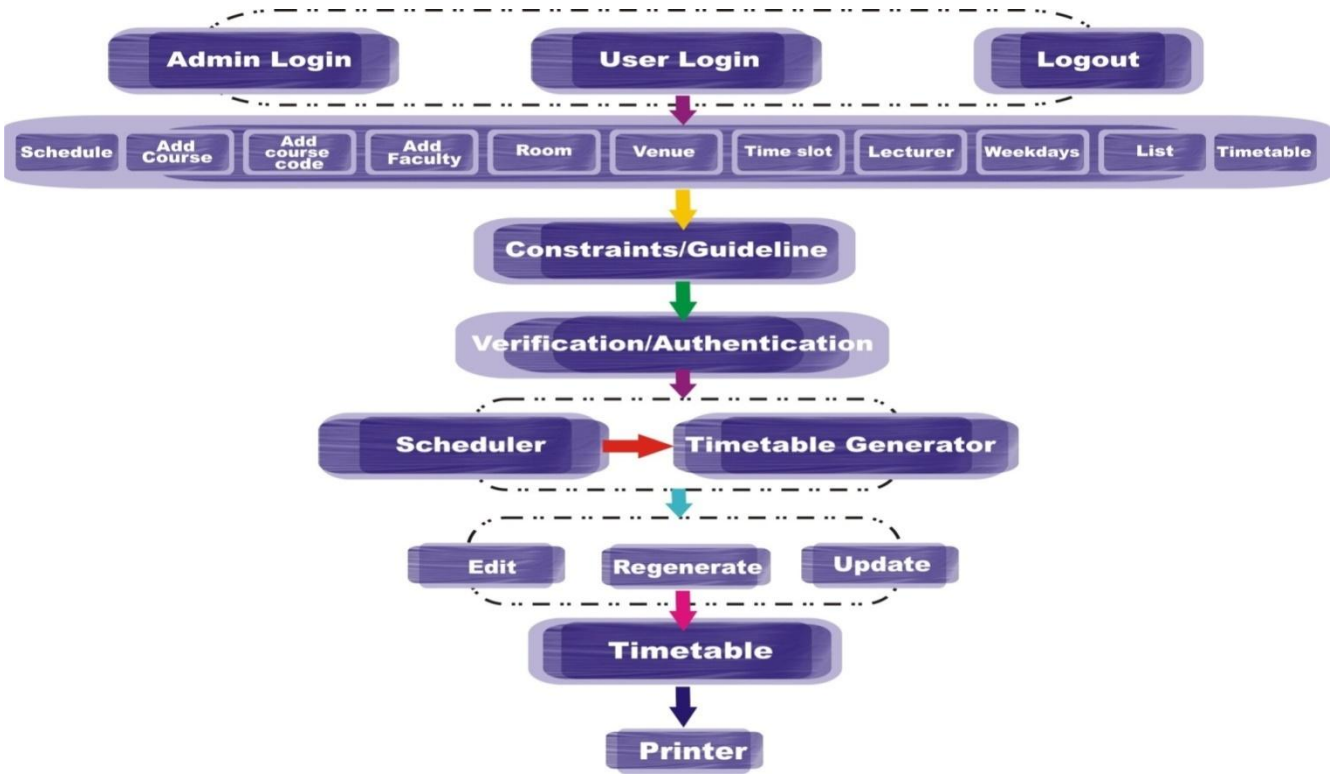

Figure 1: Timetable SchedulingSystem.

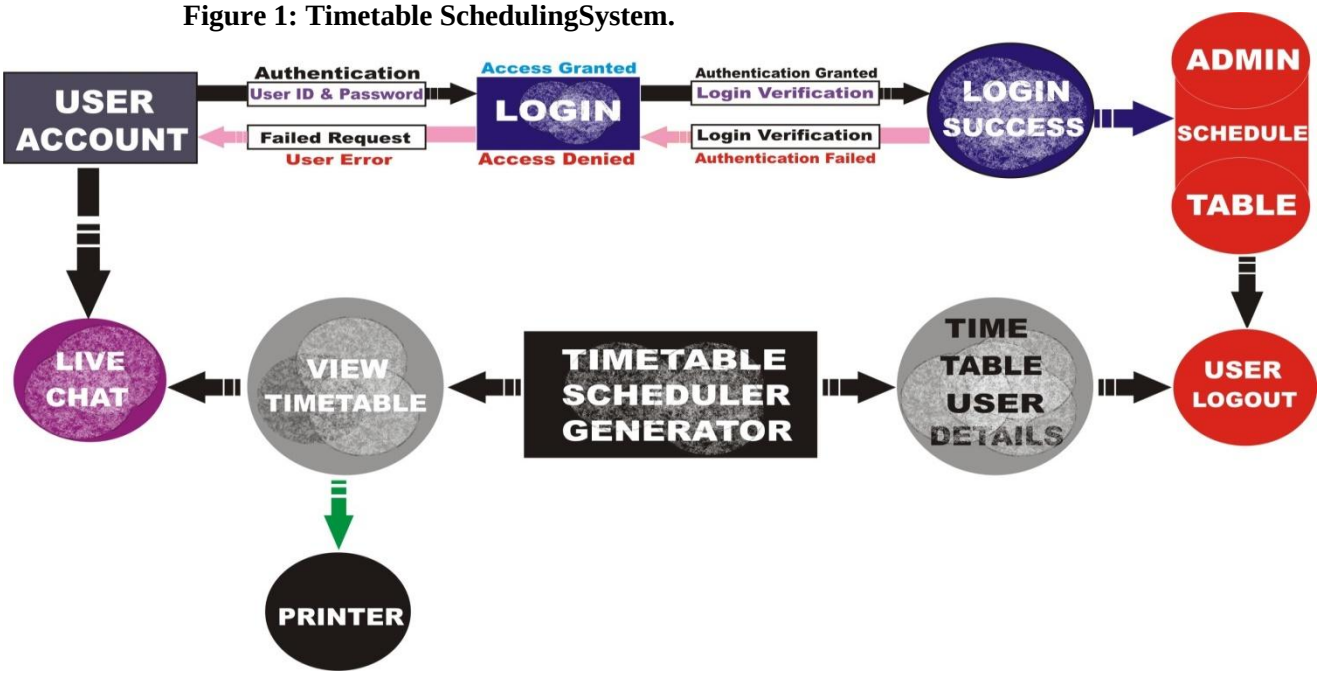

Figure 2: Data Flow Diagram.

Table 2: Test Plan Description

No.	Test Case Title	Description	Expected Outcome
1	Successful User Authentication	Login into the system using the details assigned by the admi.	Login should be successful and the user should enter in to the system
2	Unsuccessful User Verification due to wrong password	Login to the system with a wrong password	Login should fail with an error message 'Invalid Password'
3	Unsuccessful User Verification due to invalid login id	Login to the system with a valid id	Login should fail with an error 'Invalid user id'
4	Trials for generating both clash-prone and modified algorithm for generating timetable	The 5 trials are for the unmodified Quicksort algorithm as seen in Tables 3-7.	The data analysis as seen in Figures 3-7

- ***System Architure Design***

The architecture of the system is structurally designed to constitute three essential parts which includes: **GUI, Front End ((FE) and Back End (BE).** A description of the parts is as follows:

- The GUI defines the structural design regarding how the system will look like after implementation. This has a unique interactive platform that suite the user needs.
- The FE comprises of everything including the design and types of languages used in the design of the system. They include: PHP, PHPMYQL, HTML AND CSS etc.
- The BE otherwise called the server-side, deals with the system inputs, retrieval, editing and updates. This refers to everything the user cannot see in the FE such as the database and servers used.

Result and Discussion

The test-plan is basically a list of test cases that need to be run on the system. Some of the test cases can be run independently for some components such as report generation from the database which is tested independently. However, some of the test cases require the whole system to be ready before execution. It is better to test each component as at when it is ready before integrating the components, which is a unit test before the system test.
It is important to note that the test cases cover all the aspects of the system (i.e. all the requirements stated in the SRS template in Table 1). Table 2 contains a sample authentication test plan.

Test Plan Description for Table 2

Tables3-7 and Figures 3-7demonstrate the results of the 5 test trials carried out by separate administrators as mentioned in item 4 of Table 2. The tables and figures are for the unmodified Quicksort algorithm used in generating the timetables based on Algorithm 1. The values in the various fields show the number of clashes found in slots allocated for each given lecture periods, rooms and venues.

Table 3: Trial 1 for unmodified Quicksort Algorithm Implementation

Days	8-9am	9-10am	10-11am	11-12pm	Break	1-2pm	2-3pm	3-4pm
Mon	3	1	0	2	0	0	2	0
Tue	0	0	0	0	0	6	0	0
Wed	4	0	3	0	0	0	0	0
Thur	0	0	0	2	0	3	0	0
Fri	1	0	2	1	0	0	0	0
Sat	2	0	0	0	0	4	0	3

Table 4: Trial 2 for unmodified Quicksort Algorithm Implementation

Days	8-9am	9-10am	10-11am	11-12pm	Break	1-2pm	2-3pm	3-4pm
Mon	1	0	0	1	0	2	0	0
Tue	0	1	2	0	0	0	0	2
Wed	0	0	0	0	0	0	2	0
Thur	2	0	0	3	0	1	0	0
Fri	0	2	2	0	0	0	0	0
Sat	4	0	0	0	0	0	3	0

Table 5: Trial 3 for unmodified Quicksort Algorithm Implementation

Days	8-9am	9-10am	10-11am	11-12pm	Break	1-2pm	2-3pm	3-4pm
Mon	0	1	0	2	0	0	2	0
Tue	1	0	0	0	0	3	0	0
Wed	0	0	3	0	0	0	0	0
Thur	0	2	0	2	0	3	1	1
Fri	3	0	0	1	0	1	0	0
Sat	0	0	0	0	0	0	0	0

Table 6: Trial 4 for unmodified Quicksort Algorithm Implementation

Days	8-9am	9-10am	10-11am	11-12pm	Break	1-2pm	2-3pm	3-4pm
Mon	1	0	1	0	0	3	0	1
Tue	0	0	0	0	0	0	0	0
Wed	2	2	0	0	0	0	1	0
Thur	0	0	0	3	0	0	0	3
Fri	0	0	1	0	0	0	0	0
Sat	1	3	1	0	0	1	2	0

Table 7: Trial 5 for unmodified Quicksort Algorithm Implementation

Days	8-9am	9-10am	10-11am	11-12pm	Break	1-2pm	2-3pm	3-4pm
Mon	0	0	1	0	0	2	0	0
Tue	1	0	0	0	0	0	3	1
Wed	0	3	0	4	0	0	0	0
Thur	3	0	0	0	0	1	0	2
Fri	0	1	1	3	0	0	1	0
Sat	0	0	1	0	0	0	0	0

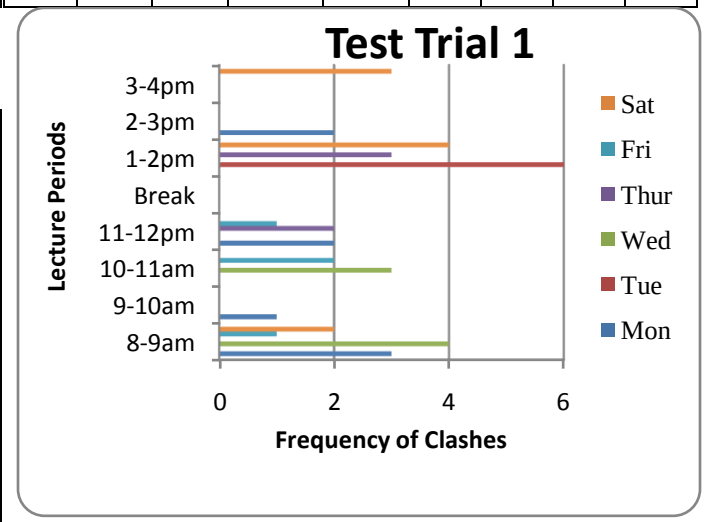

Figure 3: Chart for Table 3 trials.

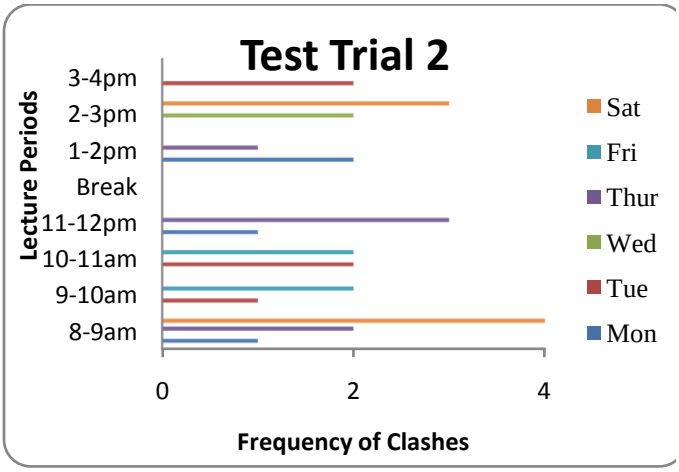

Figure 4: Chart for Table 4 trials.

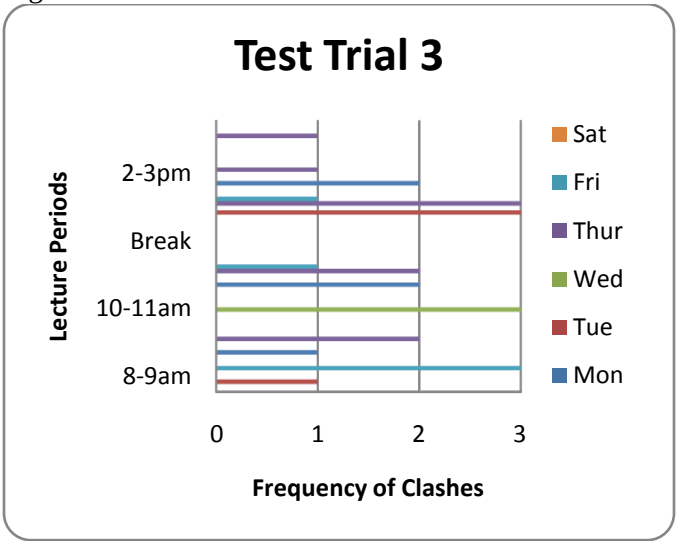

Figure 5: Chart for Table 5

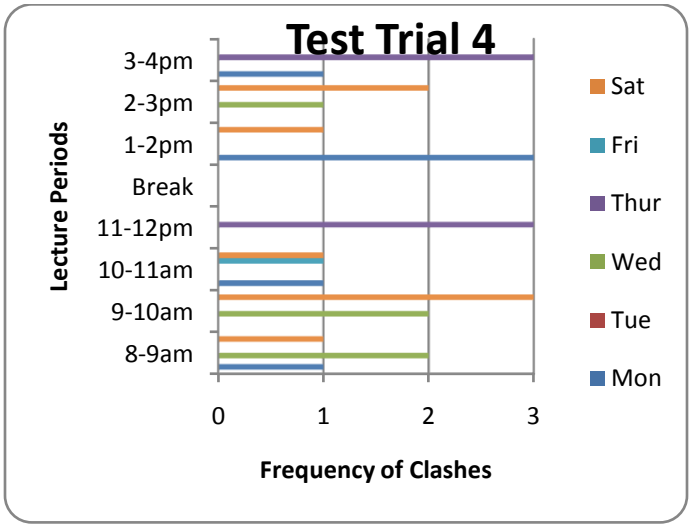

Figure 6: Chart for Table 6

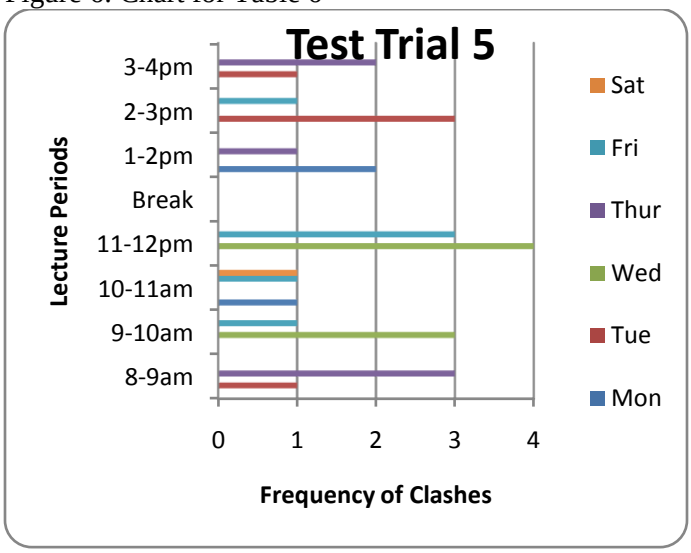

Figure 7: Chart for Table 7

Looking at Table 3 for instance, there are up to 3 clashes on Monday by 8-9am. This was reflected in Figure 3 for emphasis. After 5 trials as seen in Figures 3-7, it is clear that modifying the Quicksort algorithm in other to eliminate clashes in timeslots and venues allocated to departments was inevitable. That was why Algorithm 1 was created to check for double and single periods in parallel, which further simplified the process of the course allocation.

The randomly generated venues and time slots was particularly responsible for the clashes. However, Algorithm 1 ensured that by comparing the periods in parallel and applying the collision avoidance technique, no single or double period could be allocated in two or more consecutive days before the days are merged to form the generated time table. The timetable is generated separately for each course and course level.

Conclusion

The workflow of this proposed system makes use of collision avoidance technique and process in scheduling an automated timetable system. This made it easier and faster to completely eliminate the manual process of generating timetable. This paper presented a Logarithmic Quicksort algorithm for solving a highly constrained timetable generation problem. The approach used a problem-specific domain representation context-based reasoning for obtaining feasible solution ata reasonable computing time. The future work will entail the use of a real-time generation of timetables, with content based analysis and reports to be generated through an embedded management information system (MIS).

Acknowledgement

Glory is to God almighty for helping to oversee the writing of this article. All our family, friends, together with colleagues at the Department of Computer Science, M.Sc. Programme, Kebbi State University of Science and Technology Aliero, Kebbi State, Nigeria are very much appreciated. Appreciation also goes to Dr. Yakubu Abubakar, a Senior Lecturer in the Physics Department, Kebbi State University of Science and Technology Aliero, Kebbi State. Lastly, a profound appreciation goes to Miss. Anastesia Chinenye Paul for her care and moral support.

References

Abramson, D. and Abela, J. (1992).A Parallel Genetic Algorithm for Solving the School Timetabling Problem. Division of Information Technology, C.S.I.R.O, 15 Australian Computer Science Conference, Hobart, Feb 1992, Carlton, 3053 Australia, pp.1-9.

Andrianto D. (2014). Comparison using Particle Swarm Optimization and Genetic Algorithm for Timetable Scheduling. Journal of Computer Science, 10 (2): 341-346, 2014.

Bagul, M., Chaudhari, S., Nagare, S. and Patil , P. (2016). A Novel Approach for Automatic Timetable Generation. International Journal of Advanced Engineering and Science, 4(3): 357-367, March, 2016.

Bhaduri, A. (2009). University Timetable Scheduling using Genetic Artificial Immune Network. International Conference On Advances in Recent Technologies in Communication and Computing, 2009. ARTCom '09, 27-28 October, 2009. Kottayam, Kerala, India.IEEE.

Kheiri, A., Ozcan, E., Parks, A.J. (2016). A Stochastic Local Search Algorithm with Adaptive Acceptance for High-School Timetabling. Annals of Operations Research, 239(1):135–151, April 2016, Springer, U.S.A.

Nanda, A., Pai, M.P., and Gole, A. (2012).An Algorithm to Automatically Generate Schedule for School Lectures using a Heuristic Approach.International. Journal of Machine Learning and Computing, 2(4): 492-495, August 2012.

Sahoo, R.K,.Ojha, D., Mohapatra D.P., and Patra, M.R. (2017).Automatic Generation and Optimization of Course Timetable using a Hybrid Approach. Journal of Theoretical and Applied Information Technology, 95(1):68-77, 15th January 2017.

Saviniec, L., Santos, M.O., Costa, A.M. (2018).Parallel Local Search Algorithms for High School Timetabling Problems. European Journal of Operational Research, 265(1): 81-98, 16 February 2018.

Shrinivasan, D., Seown, T.H., and Ju, J.X. (2002).Automated Time Table Generation using Multiple Context Reasoning for University Modules. Proceedings of the 2002 congress on evolutionary computation, cec'02, volume: 2,12-17 May, 2002, Honolulu, HI, USA, USA.