Computing, Information Systems, Development Informatics & Allied Research Journal
Vol. 9 No. 1, March, 2018 - www.cisdijournal.org

Generating Feasible and Optimized Timetables using Tabu Search: The Case of Bells University of Technology, Ota.

Ezike J.O.J[1*], Oyeleye C.A.[2], Omidiora E.O.[2], Olabiyisi S.O.[2]& Oluwatosin, A.E. [1]
[1]Dept. of Computer Science and Information Technology,
Bells University of Technology, Ota, Nigeria
[2]Dept. of Computer Science and Engineering,
Ladoke Akintola University of Technology, Ogbomoso, Nigeria.

** Corresponding Author:*
E-mail: josephezike@gmail.com;
Phone: +2348035057550

ABSTRACT

The Examination timetabling problem is a well-known NP-Complete combinatorial problem present in universities. Manual timetabling is usually time consuming and produces infeasible and not satisfactory results. This research employed Tabu Search (TS) heuristic to solve the Examination Timetabling Problem using data from Bells University of Technology, Ota, Nigeria. Hard and soft constraints were elicited and used in the design of the objective function. The TS algorithm was then implemented in an application developed using Object-Oriented approach, and investigated on some parameters. The developed application automated the efficient generation of feasible and high quality real-world university timetables. Result obtained showed that the implemented TS algorithm generated high quality examination timetables and compares favorably on the Melbourne University benchmarked timetable data set.

Keywords: Automation, Optimization, Scheduling, Timetabling, Search, Examination

CISDI Journal Reference Format
Ezike J.O.J, Oyeleye, C.A., Omidiora, E.O., Olabiyisi, S.O. & Oluwatosin, A.E. (2018): Generating Feasible and Optimized Timetables using Tabu Search: The Case of Bells University of Technology, Ota.. Computing, Information Systems, Development Informatics & Allied Research Journal. Vol 8 No 1. Pp 83-103 Available online at www.cisdijournal.org

1. INTRODUCTION

A timetable is a time management tool composed of the times in which an event, task or activity will take place. Timetabling problems represent a challenging and important problem area for researchers across the fields of Operational Research, and Artificial Intelligence. It is basically a scheduling problem, which by their nature, are concerned with resource management (Pinedo, 2008) which generally arise in various forms, such as sport scheduling, crew scheduling, nurse scheduling and transportation (bus, vehicle, train, airline, et-cetera) and educational timetabling. Scheduling Problems are germane to human and organizational activities and productivities, as such, research in scheduling and timetabling have witness increasing interest from researchers globally, right from as early as the 1960s to date as noted in (Fowler, Kendall, & McCollum, 2011; Komar, Grbic, & Cupic, 2011; Pinedo, 2008; Qu, Burke, McCollum, Merlot, & Lee, 2009).

The manual university examination timetable scheduling process consumes lots of time and resources in its preparation, yet the output is usually not satisfactory to all the parties concern. Sometimes, students may be scheduled to sit for two or more examinations at the same time or to sit for two or more consecutive examinations in a given day. Such situation places such students in an unfair position when compared to their peers and usually affects their academic performance. Also, an invigilating staff may be scheduled to invigilate more times than his peers or may be schedule to invigilate at a time conflicting with other schedules. For this and similar reasons, much attention has been given to automated timetabling as noted in (Schaerf, 1999).

This research solve the Examination Timetabling Problem (ETP), using Tabu Search (TS). The ETP is essentially concerned with assigning examinations to timeslots and venues with adequate facilities and capacity for the examination while satisfying a set of constraints. The research is motivated by the desire to provide examination timetabling solution via automation while efficiently using system resources.

2. RELATED WORKS

In (Wren, 1996), a Timetabling problem was defined as "the allocation, subject to constraints, of given resources to objects being placed in space time, in such a way as to satisfy as nearly as possible a set of desirable objectives." (Ross, Hart, & Corne, 1997; Schaerf, 1999) noted that Educational Timetabling, which is one of the most widely studied of all the variants of the timetabling problems because it occurs most frequently in educational institutions, is very time-consuming, with the quality of timetables produced having a great impact on a broad range of different stake-holder. Variants of the timetabling problems discussed in literatures differ from each other based on the type of institution involved (University or high school) and the types of constraints. University timetabling can be categorized into course timetabling and examination timetabling as noted in (Burke, Elliman, & Weare, 1993; Carter, 1986). A summary of the three broad categories of Educational timetabling is given in (Schaerf, 1999) and (Qu, et al., 2009) as follows: ***School timetabling*** – the weekly scheduling for all the classes of a school, avoiding teachers meeting two classes at the same time, and vice versa; ***Course timetabling*** - the weekly scheduling for all the lectures of a set of university courses, minimizing the overlaps of lectures of courses having common students; ***Examination timetabling*** - the scheduling for the Examinations of a set of university courses, avoiding overlap of examinations of courses having common students, and spreading the examinations for the students as much as possible.

2.1 The Examination Timetabling Problem

A number of researchers such as (Burke, Elliman, Ford, & Weare, 1996; Cooper & Kingston, 1995; Garey & Johnson, 1979) have note that Examination timetabling problem is a well-known NP-Complete combinatorial problem present in universities with a large number of students and courses, especially if the courses are many and the student can choose from a wide range of electives. In (Boizumault, Delon, & Peridy, 1996), it was stated that Operations researchers has identified the Examination timetabling problem as a scheduling problem with disjunctive and cumulative conjunctive constraints, classified as NP-complete, for which no classical operations research (OR) approach is directly applicable.

In (Burke, et al., 1996) and (Carter, Laporte, & Chinneck, 1994.), it was noted that the complexities and the challenges posed by the timetabling problem arise from the fact that a large variety of constraints, differing from institution to institution, some of which contradict each other, need to be satisfied in different institutions. The constraints fall into one of two category**:** Hard and Soft. Fulfilling all hard constraint make the generated Timetable feasible. The more the soft constraints fulfilled the more desirable the timetable becomes. Examples of soft constraints are given in (Qu, et al., 2009) who also noted that the primary (or key) hard and soft constraints in examination timetable can be classified as *time related* or *resource related*. Determining the quality of timetables generated is usually done by using an objective function, usually formulated using the hard and soft constraints stipulated by the institution and implemented in the timetable generating algorithm.

2.2 Approaches to Solving the Timetabling Problem

From literature, many approaches exist for solving the ETP, such as Simulated annealing (SA), Fuzzy Logic, Neural Networks, Genetic Algorithm (GA), Knowledge-based Techniques, Case based Reasoning, Ant Colony Optimization (ACO), Tabu Search (TS), et-cetera. These approaches have been classified as Graph based technique, constraint-based technique, local search based techniques, Evolutionary/Population based Algorithms, Multi-criteria techniques, Hyper-heuristics, Decomposition/Clustering technique and Hybridization. A detailed review of these different techniques and their classification can be found in (Burke & Petrovic, 2002; Carter & Laporte, 1996; Qu, et al., 2009). Of all the above mentioned techniques, Tabu Search, a local search based technique, is amongst the leading paradigms for solving the ETP and is utilized in this research.

2.3 Tabu Search Algorithm

Tabu Search (TS) is a "higher level" heuristic procedure for solving optimization problems, designed to guide other methods (or their components) to escape the trap of local optimality (Glover, 1990). TS has been used in a wide range of applications ranging from scheduling to telecommunications, character recognition to neural networks, to obtain optimal and near optimal solutions in classical and practical problems. In (Gendreau & Potvin, 2010), TS was noted to be amongst the most effective technique in tackling difficult problems and finding good solutions to the large combinatorial problems encountered in many practical settings. As such, TS is popular among researchers.

TS uses flexible memory structures, classified as short term memory, intermediate term and long term memories, which allow search information to be exploited more thoroughly. TS also uses conditions for strategically constraining and freeing the search process (embodied in tabu restrictions and aspiration criteria), and memory functions of varying time spans for intensifying and diversifying the search, thereby reinforcing attributes historically found good and driving the search into new regions (Glover, 1989). Parameters, such as tabu list size and stopping criteria, usually need to be fine-tuned in line with the problem being solved to enable efficient and effective performance of the algorithm.

An application of TS is generally characterized by the following: (i) the initial solution, (ii) the neighborhood generation methods, i.e., set of possible moves applicable to the current solution, (iii) the definition of tabu moves with the tabu list size and (iv) the termination condition(s).

In (Gendreau, 2002), a template for TS is presented as follows:

Notation

- ***S***, the current solution,
- ***S****, the best-known solution,
- ***f****, value of *S**,
- ***N(S)***, the neighborhood of *S*,
- ***Ñ(S)***, the "admissible" subset of *N(S)* (i.e., non-tabu or allowed by aspiration).

Initialization

- Choose(construct) an initial solution S0,
- Set $S := S_0$, $f^* := f(S_0)$, $S^* := S_0$, $T := \emptyset$.

Search

while termination criterion not satisfied do

- Select *S* in *argmin* [*f(S')*];

$$S' \in \tilde{N}(S)$$

- if $f(S) < f^*$, then set $f^* := f(S)$, $S^* := S$;
- Record tabu for the current move in T (delete oldest entry if necessary);

endwhile

The scope of this research is the basically to solve the ETP, by using TS to automated the generation of university examination timetables

3. METHODOLOGY

In carrying out this research, theoretical foundation for the ETP was established via a comprehensive literature review, where approaches and implementation consideration for solving ETP using TS was noted. Using Bells University of Technology, Ota, as a case study, constraints for solving the ETP were then determined and used in the design of the objective function. Listings of the constraints and their associated penalty values are given in Table 1 below. The constraints penalty values used were determined from trial experimental runs and were found to be effective for guiding the TS algorithm during the search process.

Table 1: Constraint Types and Penalty Values

TYPE	CODE	DEFINITION	PENALTY VALUE
Hard	HC1	No student should write more than one exam at a time (that is, write two or more exams at a time)	1,000,000,000
	HC2	No teacher (staff) should be scheduled to be in more than one room at any time.	1,000,000,000
	HC3	No exam should be schedule more than one	1,000,000,000
	HC4	All scheduled venues must have adequate capacity to contain the students that enrolled for the exam.	1,000,000,000
Soft	SC1	No student should be scheduled to sit for two non-consecutive exams.	1
	SC2	No student should be scheduled to sit for two consecutive exams.	100
	SC3	No student should be scheduled to sit for three consecutive exams.	100,000
	SC4	No teacher should be scheduled to invigilate two non-consecutive exams.	1
	SC5	No teacher should be scheduled to invigilate two consecutive exams.	100
	SC6	No teacher should be scheduled to invigilate three consecutive exams.	100,000

3.1Mathematical Formulation of the ETP

Table 1 shows the main constraints considered in the ETP. These summarizes the basic Examination Timetabling constraints in literature, in timetabling completion and in most benchmarked timetabling problems as can be seen in (McCollum, 2006; Qu, et al., 2009). These constraints are grouped as hard and soft.

3.1.1 Resource Definition:

The resources used in solving the ETP are defined as follows:

P: A set of p periods (or time-slots), p_1, p_2, …, p_p.
D: A set of d days (i.e. examination duration), $d_1, d_2, d_3, \dots, d_d$: a day comprise 1 to 3 periods.
E: A set of e examinations, $e_1, e_2, e_3, \dots, e_e$
E_p: A set of β examinations scheduled in period p_p, that is, e_1p_n, e_2p_n, e_3p_n … $e_\beta p_p$
S: A set of s students, $s_1, s_2, s_3, \dots, s_s$, in the campus of the university
L: A set of l teachers, $l_1, l_2, l_3, \dots, l_l$ in campus c of the university.
R: A set of s course registration lists for all students in the campus, that is, $R_{s_1}, R_{s_2}, R_{s_3}, \dots, R_{s_s}$
V: A set of all y venues in the campus, that is, $v_1, v_2, v_3, \dots, v_y$

3.1.2 Decision Variables
All decision variables can have a value of 0 or 1.
$e_m p_n$: is the instance of an examination e_m scheduled in period p_n. $e_m p_n$= 1 if schedule or 0 otherwise.
$s_j e_m$: is the instance that student s_j enrolled for examination e_m. $s_j e_m$ = 1 if student enrolled or 0 otherwise.
$s_j e_m p_n$: is the instance that student s_j is to sit for examination e_m scheduled for period p_n. $s_j e_m p_n$ = 1 if student is scheduled or 0 otherwise.
$l_g v_y p_n$: is the instance of a teacher l_g scheduled to be in venue v_y at period p_n.

3.1.3 Notations Used
The notation $n(e_1)$ denote the number of students that enrolled for examination e_1, cap(v_y) denote the capacity of venue v_y, and duration (p_n) denote the number of hours in period p_n.

3.1.4 Assumptions
The following are the assumption made in carrying out the research experimentation:
i. The total venue capacity in any campus is at least 50% of total students on campus.
ii. Staff to Student ratio is assumed to be 1 to 30, that is, for every 1 to 30 students, there is an invigilating staff available.
iii. There are only three (3) periods in a day, that is,

$$\forall\, d_h \in D\,, \exists p_{n+i} \in P : i = 1, 2, 3.$$

Where p_n is the last period of the previous day and $n \in \mathbb{N}_0$.

iv. Each period is of a fixed 3-hour duration, that is,

$$\forall\, t_b \in T, duration\,(t_b) = 3.$$

v. All venues dedicated for use during examination are available during the entire examination period.

Each student course registration list R_{sj} for which examinations are to be taken is of the form:

$$R_{si} = c_1, c_2, c_3, \dots, c_\lambda \text{ where } j \in \mathbb{N}_1 \wedge j \geq 1 \text{ or } 1 \leq j \leq \lambda.$$

The value of λ is determined by the total units of all courses registered for in the given semester.
Secondly, each course has "unit" weight w as one of its property, such that:

$$w \in \mathbb{N}_1 \wedge 1 \leq w \leq a : a \in \mathbb{N}_1$$

where *a* is the maximum weight any course can have in the given university. Usually, there is a minimum and maximum amount of course-unit load a student is expected to register for every semester, say φ and ψ. As such, another property of each student registration will be the total course-unit load registered for in the given semester.

This can be computed from the unit of all the courses in the student's course registration list. Using the notation, c_{zwz} to refer to course z with its associated weight w_z, then a student's course registration list becomes: $R_{sj} : c_{1w1}, c_{2w2}, c_{3w3} \dots, c_{\lambda w\lambda}$ and the expression of the expression of the total registerable course-unit load per semester for student s_j becomes:

$$T_{usj} = \sum_{i=1}^{\lambda} w_i \; : \; \varphi \leq T_{usj} \leq \psi \; \wedge \; \varphi, \psi \in \mathbb{N}_1$$

3.1.5 Course and Examination Relations

Usually, examinations are written for all courses registered in a given semester, that is:

$$\forall \, c_z \in C, \exists \, e_z \in E : c_z \; \Leftrightarrow \; e_z$$

However, this condition may not hold in all cases as there are some courses (either practical in nature or otherwise) for which examinations are not to be conducted. In view of the foregoing, let E be the set of all examinable courses for which at least a student is in enrolment, that is:

$$\forall \, e_m \in E \, , \nexists \, e_m \in E : n_s(e_m) = 0 : \; E \subseteq C \; \wedge \; m = 1, 2, 3, \ldots, n$$

Where $n_s(e_m)$ represents the number of students that enrolled for examination e_m
If Se_m represent the set of students that enrolled for examination e_m, then $|Se_m| = n_s(e_m)$

3.2 Constraint Modelling

In this section, the constraints used in this research as well as the objective function formulated are modelled. The Local Hard and Soft Constraints are represented with the HC and SC codes respectively(see Table 3.1).

HC1: No student should write more than one examination at a time (period) at any time.

Using the decision variable $s_i e_j p_k$ which represent the instance that student s_i is to sit for examination e_j scheduled for period p_k: $s_i e_j p_k$= 1 if student s_i is scheduled or 0 otherwise. For a given period p_k for any student s_i and for all examination scheduled, this becomes:

$$\sum_{j=1}^{e} s_i e_j p_k \leq 1 \qquad 2.1$$

HC2: No Teacher should be scheduled to be in more than one venue at the same time (period). This is the same as saying, a teacher should not be booked twice at the same time or in a given period. Since $l_g v_a p_n$ is the instance of a teacher l_g scheduled to be in venue v_a at period p_n, then, for teacher l_g in period p_n this can be stated as:

$$\sum_{a=1}^{j} l_g v_a p_n \leq 1 \qquad 2.2$$

HC3: All scheduled venues must have adequate capacity to contain the students that enrolled for the examinations scheduled in them. For any venue v_y where examination e_m is scheduled to hold and using the decision variable $s_i e_m v_y$ to be the instance that student s_i enrolled for examination $e_{m,}$ scheduled in any venue v_y, HC3 can then be expressed as:

$$\sum_{i=1}^{k} s_i e_m v_y \; = \; n_s(e_m) : n_s(e_m) \leq cap(v_y) \qquad 2.3$$

In a situation where more than one examination, say examinations e_1 to e_z, can be held in a given venue v_y, in the campus with *k* students, provided the venue capacity constraint is not exceeded, equation (3.5) becomes:

$$\sum_{i=1}^{k}\sum_{j=1}^{z} s_i e_j = \sum_{j=1}^{z} n_s(e_j) \ : \sum_{j=1}^{z} n_s(e_j) \leq cap(v_y) \qquad 2.4$$

SC1**:** No student should be scheduled to sit for two non-consecutive (or more than one) examination in a day. Using the decision variable $s_i e_m p_n$ indicating that student s_i is schedule to write examination e_m at period p_k, in a given day d_h (recall, each day comprises three periods), then SC1 can be stated as:

$$\sum_{k=1}^{k=3} s_i p_{kd_h} \leq 1 \qquad 2.5$$

SC2: No student should be scheduled to sit for two consecutive examinations in a given day, that is, no student should write examinations in two consecutive periods. Using the decision variable $s_i e_m p_n$ indicating that student s_i is schedule to write examination e_m at period p_k, in a given day d_h, SC2 can then be stated as:

$$\sum_{k=a}^{k=a+1} s_i p_{kd_h} < 2 \qquad 2.6$$

where *a* = 1 *or* 2 : a + 1 $\leq$ 3.

SC3: No student should be scheduled to sit for three consecutive examinations in a day. Using the decision variable $s_i e_m p_n$ indicating that student s_i is schedule to write examination e_m at period p_k, in a given day d_h, then SC3 can be stated as:

$$\sum_{k=1}^{k=3} s_i p_{kd_h} < 3 \qquad 2.7$$

SC4: No teacher should be scheduled to invigilate two non-consecutive examinations in a day. Using the decision variable $l_g e_m p_k$ to be the instance of a teacher l_g scheduled to invigilate examination e_m in period p_k in day d_h Then SC4 can be stated as:

$$\sum_{k=1}^{k=3} l_g e_m p_{kd_h} \leq 1 \qquad 2.8$$

SC5: No teacher should be scheduled to invigilate in two consecutive periods. Using the decision variable $l_g e_m p_{kd_h}$ to be the instance of a teacher l_g is scheduled to invigilate examination e_m in period p_k in day d_h, SC5 can be stated as:

$$\sum_{k=a}^{k=a+1} l_g e_m p_{kd_h} < 2 \qquad 2.9$$

where a = 1 *or* 2 : a + 1 $\leq$ 3.

SC6: No teacher should be scheduled to invigilate in three consecutive periods. Using the decision variable $l_g e_m p_{kd_h}$ to be the instance of a teacher l_g scheduled to invigilate examination e_m in period p_k in day d_h, SC6 can be stated as:

$$\sum_{k=1}^{k=3} l_g e_m p_{kd_h} < 3 \qquad 2.10$$

3.3 The ETP Objective Function

In this research, the objective f_o is defined in terms of the penalty function f_p as:

$$f_o = f_p = f_{p\,(hard)} + f_{p\,(Students)} + f_{p\,(invigilators)} \qquad 2.11$$

where $f_{p\,(hard)}$, $f_{p\,(Students)}$ and $f_{p\,(invigilators)}$ represent the three components of the penalty function as can be deduced from Table 1. Defining the objective function in terms of the hard and soft constraints is the practice in literature when the timetabling problem is formulated as an optimization problem and not a search problem as noted in (Schaerf, 1999). Representing the objective function in terms of the penalty function (see equation 2.11) indicates that the problem would be treated as a minimization problem, however, to treat same as a maximization problem, the objective function would be written as an inverse of the penalty function as stated in Equation 2.12.

$$f_o = \frac{1}{1+f_p} = \frac{1}{1+f_{p\,(Students)} + f_{p\,(invigilators)}} \qquad 2.12$$

In this research, the objective function is defined in terms of the penalty function, as a minimization problem, so equation 2.11 can be written as:

$$f_o = f_p = w_h \sum_{i=1}^{i=4} HC_i + \sum_{j=1}^{j=3} w_j SC_j + \sum_{k=1}^{k=3} w_k SC_k \qquad 2.13$$

Where $w_j SC_j$ and $w_k SC_k$ represent the students and invigilator-related constraints respectively. If the number of student-related constraints is pairwise comparable with that of the invigilator constraints and the assigned weights (penalty) are same for each pair as in the case in this work, that is,

$$\sum_{j=1}^{q} w_j SC_j \equiv w_k \sum_{k=1}^{z} SC_k \quad : q = z \text{ and } w_j = w_k;$$

then equation 2.13 can be written as:

$$f_p = w_h \sum_{i=1}^{i=4} HC_i + w_1\left(SC_{j=1} + SC_{k=1}\right) + w_2\left(SC_{j=2} + SC_{k=2}\right) + w_3\left(SC_{j=3} + SC_{k=3}\right) \qquad 2.14$$

and simplified further as:

$$f_p = w_h \sum_{i=1}^{i=4} HC_i + \sum_{j,k=1}^{j,k=3} w_j\left(SC_j + SC_k\right) \qquad 2.15$$

Considering that HC_i, SC_j, and SC_k are hard and soft constraints for which the consequence of their violation varies, the weights $w_h, w_{j=1}, w_{j=2}$ and $w_{j=3}$ are chosen such that $w_h \gg w_{j=1} \gg w_{j=2} \gg w_{j=3}$. These choice of weights values enable the search algorithms to be effectively guided.

From the foregoing, the ETP can now be stated as an optimization problem as follows:

Minimize equation (2.15), subject to the following constraints:

$$\sum_{j=1}^{e} s_i e_j p_k \leq 1 \qquad 2.1$$

$$\sum_{a=1}^{j} l_g v_a p_n \leq 1 \qquad 2.2$$

$$\sum_{i=1}^{k} s_i e_m v_y = n_s(e_m) : n_s(e_m) \leq cap(v_y) \qquad 2.3$$

$$\sum_{i=1}^{k}\sum_{j=1}^{z} s_i e_j = \sum_{j=1}^{z} n_s(e_j) : \sum_{j=1}^{z} n_s(e_j) \leq cap(v_y) \qquad 2.4$$

$$\sum_{k=1}^{k=3} s_i p_{kd_h} \leq 1 \qquad 2.5$$

$$\sum_{k=a}^{k=a+1} s_i p_{kd_h} < 2 \qquad 2.6$$

$$\sum_{k=1}^{k=3} s_i p_{kd_h} < 3 \qquad 2.7$$

$$\sum_{k=1}^{k=3} l_g e_m p_{kd_h} \leq 1 \qquad 2.8$$

$$\sum_{k=a}^{k=a+1} l_g e_m p_{kd_h} < 2 \qquad 2.9$$

$$\sum_{k=1}^{k=3} l_g e_m p_{kd_h} < 3 \qquad 2.10$$

3.4 Data Gathering and Generation

The data used for solving the ETP was gathered from Bells University of Technology, Ota, as at the end of 1st Semester 2012/2013 Sessions.

A summary of this is given as follows:

1	Total number of Students	1896
2	Total number of Registrations	16938 (~ 9 Examination/student)
3	Total number of Examinations	910 (501 for 1st Semester)
4	Total number of Venues	25 (capacities from 20 – 240)
5	Total Number of Invigilating Staff	170

In order to be able to investigate the impact of increasing student's population on the two algorithms, the data gathered was used to generate databases for a student population of 25,000, 50,000, 75,000 and 100,000. Invigilators or staff to student's ratio was set at 1 to 30. For the generated databases, the total available venue capacity was set at 50% of the student population.

3.5 Timetable Representation

In this research, the timetable was represented as an object, modelled using the following classes: Staff, Student, Examination, Registration, Venue, Period and Timetable. The relationship between these classes is shown in Figure 1.

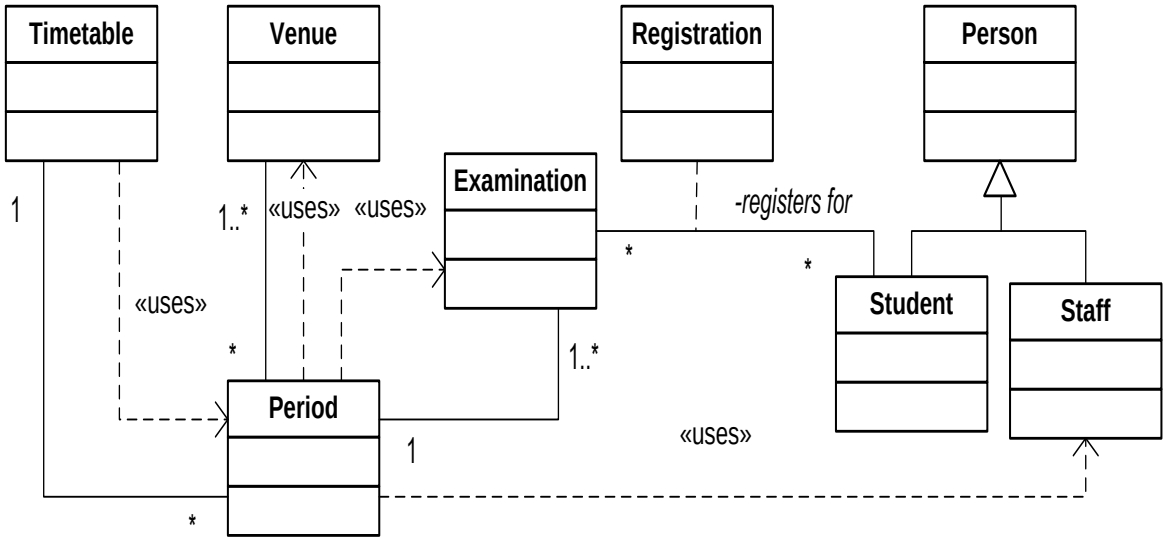

Figure 1: Classes used in the object-oriented design of the timetable object.

The timetable object contains an arraylist of period objects. Period objects in turn contains three arraylists for examinations, venues and staff (invigilator) objects. Each examination object contains an arraylist of matriculation numbers of students in enrolment. Figures 2 illustrates the different components of the timetable object.

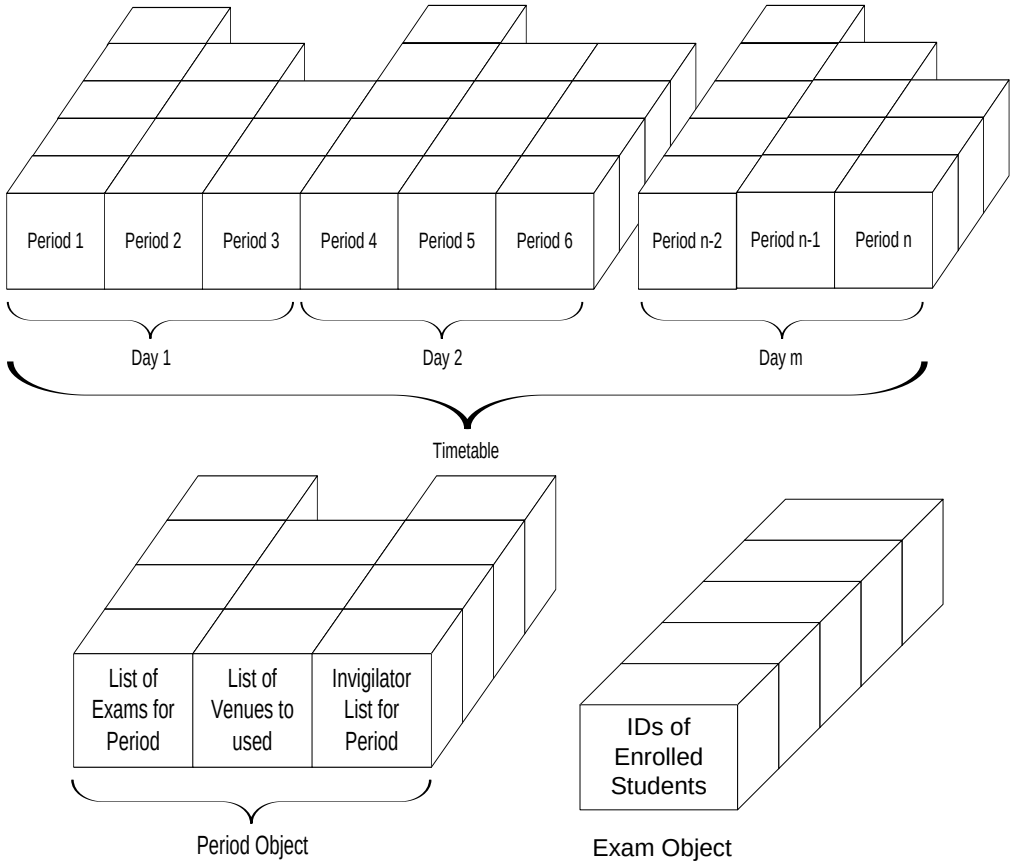

Figure 2: Different components contained in the timetable object.

3.6 Implemented Algorithms

To carry out the research, an application was develop that implemented the TS algorithm used in this research, including the algorithm for generating the initial solution before the optimization process.

	Pseudo Code for the Timetabling Application
1	Start
2	Declare and Initialize working variables
3	Load Data from Database (Venues, Courses, Registrations, Students, Staff)
4	Extract Course Registration List for each student in semester
5	Extract student's list for each enrolled course
6	Create Initial Solution
7	Optimize Population Solution with TS
8	Schedule Invigilators
9	Display Timetable
10	End.

Figure 3: Pseudo Code for the Implemented Timetabling Application.

3.6.1 Generation of Initial Timetable Solution

Line 6 of the pseudo code of the developed timetabling application (Figure 3) capture the generation of the initial timetable solution. This algorithm for "Create Initial Solution is given in Figure 4.

Algorithm: CreateInitialSolutionForTS( )

1 Input: VL, eCL, StudCRL

// VL is VenueList; eCL is examinableCourseList; StudCRL is StudentCourseRegistrationList

Output: t_0'

2 ***CreateTimetableforSelCourseRand( )*** *// This create the Timetable object and*

// populate it

3 Declare and Initialize variables and List

4 $t_0 \leftarrow$ Create Timetable Object ()

5 **While** (examinableCourseListNotEmpy)

6 $p_i \leftarrow$ Create Period Object () // i goes from 1 to n

7 Initialize Period's Total Venue Capacity(p_i)

8 ScheduleExamsToPeriod(pi, eCL) //courses are selected randomly

9 AddPeriodObjectToTimetable(t_0, p_i)

10 increment i.

11 **Endwhile**

12 AddAdditionalEmptyPeriodObjectsToTtimetableIFNecessary (t_0) // i.e. "normalize" Timetable

13 SpreadScheduledExaminationInTimetableToNewlyAddedPeriods(t_0, ,eCL, StudCRL)

14 ComputeTimetableConflictsAndFitness (t_0, eCL, StudCRL)

15 Return Solution(t_0)

16 **End**

Figure 4: Algorithm for "Create Initial Solution for the TS Algorithm.

3.6.2 Description of the TS Algorithm

The description of the implemented Tabu Search algorithm is given below:

```
    Algorithm Tabu Search
1   Input: t0, StudCRL  // t0 is the initial solution,
2                       // StudCRL is the list of courses registration list of all the students
3   Output: tbest
4   t ← t0
5   tbest ← t0
6   TL ← Ø  // TL is the tabu list
7   while (not stoppingCondition)
8           CL ← Ø  //CL, the list for all candidate solution
9           CL ← N(t)  // N, the neighborhood operator, generates all candidate solution of t
10          t ←applyBestMove(t, CL)
11          if (fitness(t) > fitness(tbest))
12              tbest← t
13          endif
14          TL ← tabuAppliedMove(CL)
15          if((size(TL) > maxSize(TL))
16              removeFirst(TL)
17          endif
18          if(DiversificationCcondition)
19                  diversify()
20          endif
21  endwhile
22  return tbest
```

Figure 5: Description of the TS Algorithm.

3.6.3 TS Neighborhood Operator Description

In the TS Algorithm implementation, the Neighborhood operator (line 9) generated all the possible "moves" to different new timetable solutions, that is, candidate solutions (see Figure 6). The technique applies an atomic move and produce a resulting timetable solution, evaluate its fitness and then reverse the move. This was done with all the generated moves, held in the generated moves list (representing CL). This ensured the evaluation of all candidate solutions. The acceptance of the best candidate solution results in the implementation of the move that resulted in that candidate solution.

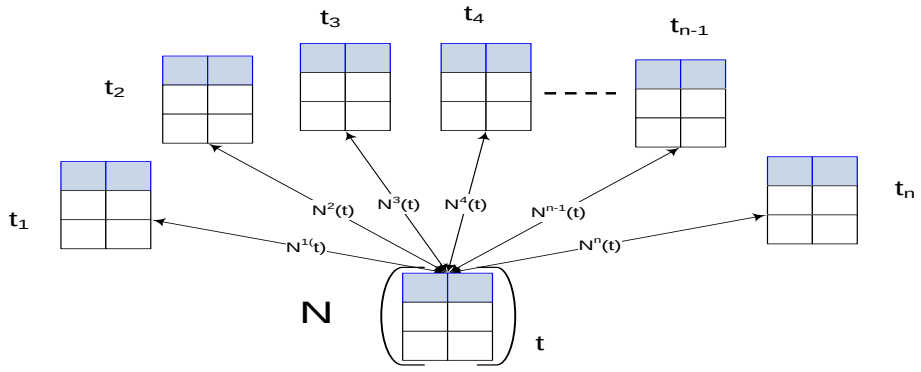

Figure 6: The operation of the Neighborhood operator N on timetable t to produce its neighbors t1 to tn.

3.6.4 TS Parameter Tuning

Trial experimental runs were carried out with the TS algorithm with tabu-list sizes of 5, 7, 8, 9, 10, 11, 12, 15, 20. Tabu-list size of 10 was found to be more effective in guiding the search. Different other conditions such as when the generated moves list was empty because all generated moves were not admissible, were monitored and used to determined appropriate time to diversify to other search region.

3.7 Experimental Environment

The algorithms were implemented in Java (JDK8u54) on Windows 7 64-bit Operating system on a Dell Inspiron N5110 Laptop with Intel core i5 (quad core) processor, 6 GB RAM, 700 GB HDD (Hitachi) at 5400 RPM. NetBeans IDE 8.02 was used for the application development with XAMPP version 3.2., which incorporates MySQL Database and phpMyAdmin for administering the database.

3.8 Experimental Procedures

The experiments commenced with the loading of data from the database and preprocessed accordingly. Initial timetable solution was then generated by randomly selecting examinable courses from list and scheduled to periods by a heuristic that ensure the solution was feasible. As a result of the random selection of courses, the number of periods in different initial solutions varied from about 21 to about 28. The initial solutions where then normalized to 30 periods so as to give a common bases for comparing all final timetables generated using the TS algorithm. This normalization process also improves the initial timetable quality as courses were spread into additional periods. The 30 period used implies the Examination duration will last 10 day, or two weeks of five working days each. As a given institution may desire, the duration of the examination (in terms of period or days) can be reduced to a pre-determined minimum before the produced timetable becomes infeasible, or increased to above 10 days for better spread and by implication, increased the possibility of generating much higher quality timetable.

3.9 Performed Experiments

Experiments were performed to investigate the following: (i) Effect of Varying TS epoch on Timetable Quality (ii) Effect of varying Student Population on Timetable Quality (iii) the empirical Time Complexity of the Implemented TS algorithm (iv) the empirical Space Complexity of the Implemented TS algorithm. Experimental runs were conducted three (3) times on the parameter being investigated. The results of the experimental runs were harvested for analytical purposes and reported as Best result (in terms of timetable quality) of the three experimental run (BO3) and Average of the three runs (AO3). Results from these experiments are report in the next section.

4. RESULTS AND DISCUSSION

The results obtained from the conducted experiments are here presented and discussed. Appendix A is an extracted page of one of the generated timetable using actual data from Bells University of Technology. Appendix B is a list of used venues with their capacities from Bells University of Technology, Ota.

4.1 The Effect of Varying TS Epoch on Timetable Quality

Figure 1 showed the effect of varying the TS epoch from 0 to 3000 for a student population of 100,000. The drop in the Timetable penalty values can be seen as the epoch count progresses, implying an improvement in the timetables qualities. Figure 2 is a magnified view of values at epoch of 1200 to 3000, showing that the penalty reduction is still continuing with increasing epoch count. At the end of 3000 epoch, the TS (BO3) and the TS (AO3) had both improved by 99.96% and 99.90%. This implies that the TS algorithms rapidly improved its individual in all the three experiments. It is clearly obvious that the TS algorithm produced high quality examination timetables.

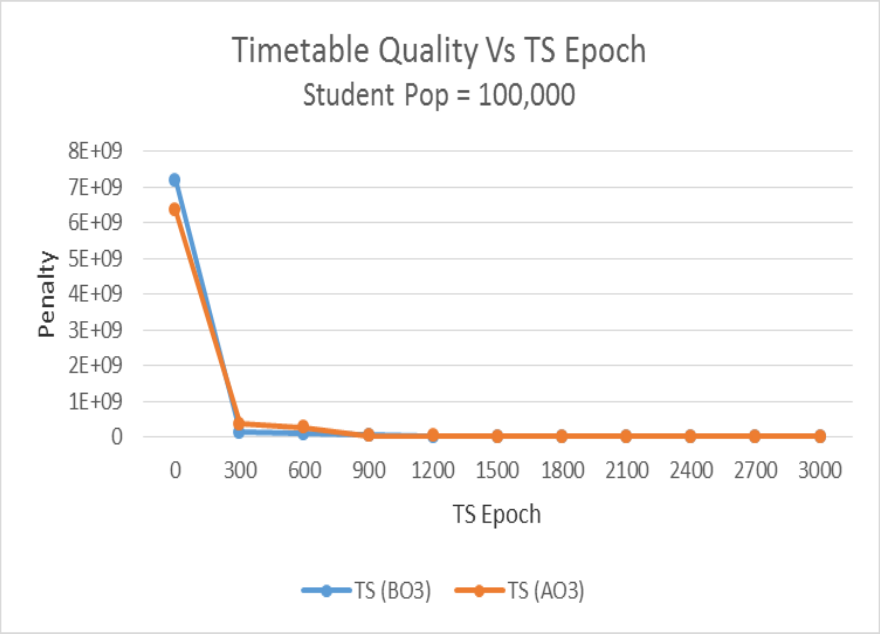

Figure 1: The Effect of varying TS epoch on generated Timetable Quality

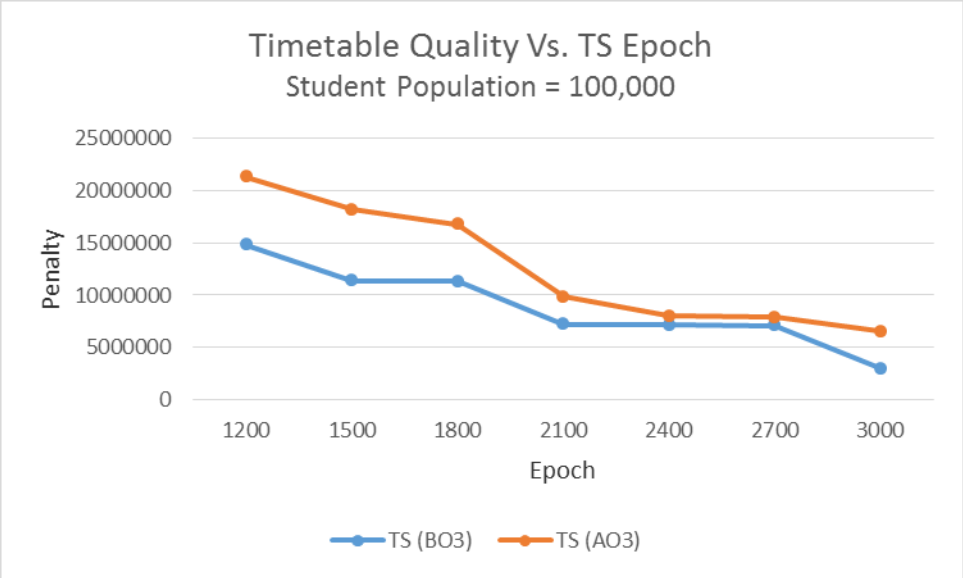

Figure 2: A magnified view of Figure 1 (from epoch of 1200 to 3000)

Timetable Quality vs. Number of Students Having 3 Consecutive Exams

It should be noted that cases of students being scheduled to sit for three consecutive (3C) examinations, with only 30 minutes break in between can be terrible for such students, definitely affecting the student's performance in all the exams. As such, this soft constraint was awarded a penalty value of 100,000 as can be seen in Table 1. Since no hard constraints is violated, thereby making the timetable feasible, it is primarily the 3C cases (SC3) that contributes the most to the penalty values of the generated timetables.

The large penalty value of the SC3 constraint compared to SC2 in Table 1 for cases of two consecutive examinations (2C) drives the TS algorithm to give preference to addressing the SC3 constraint as against the SC2 constraint. Table 2 shows the performance of the TS algorithm in eliminating the SC3 constraints for the results reported in Figure 1 and Figure 2 above.

Table 2: TS Progress in Ensuring the Soft Constraint SC3 (i.e. 3C)

Epoch	Exp1	Exp2	Exp3	Avg Pen
0	65141	71804	53371	63438.67
300	5757	1219	3457	3477.67
600	3831	671	2911	2471.00
900	151	447	222	273.33
1200	143	86	222	150.33
1500	86	86	220	130.67
1800	48	86	220	118.00
2100	0	45	99	48.00
2400	0	45	46	30.33
2700	0	45	46	30.33
3000	0	0	46	15.33

At epoch 0, the total count of cases of students scheduled to have 3C examinations in a given day were listed for all three experimental runs. Only Exp3 still haves 46 cases at the end of epoch 3000. The second experimental run (Exp2) however turned out the best result.

4.2 Effect of varying Student Population on Timetable Quality

Figure 3 showed the effect of varying student's population from 25,000 to 100,000, in steps of 25,000, while holding the TS epoch at 3000. It can be said that increasing student population did not seriously impact negatively on Timetable quality for the TS (BO3), though there appears to be a very marginal increment. The TS (AO3) showed almost a linear increase save for the value at 75,000 student population, where two of the three experiments recorded a significant high penalty value, resulting in the overall high aggregate. It is therefore advice that experiments of this kind containing stochastic components be done more than once to obtain a more likely real world result and give a better view of the algorithms performance.

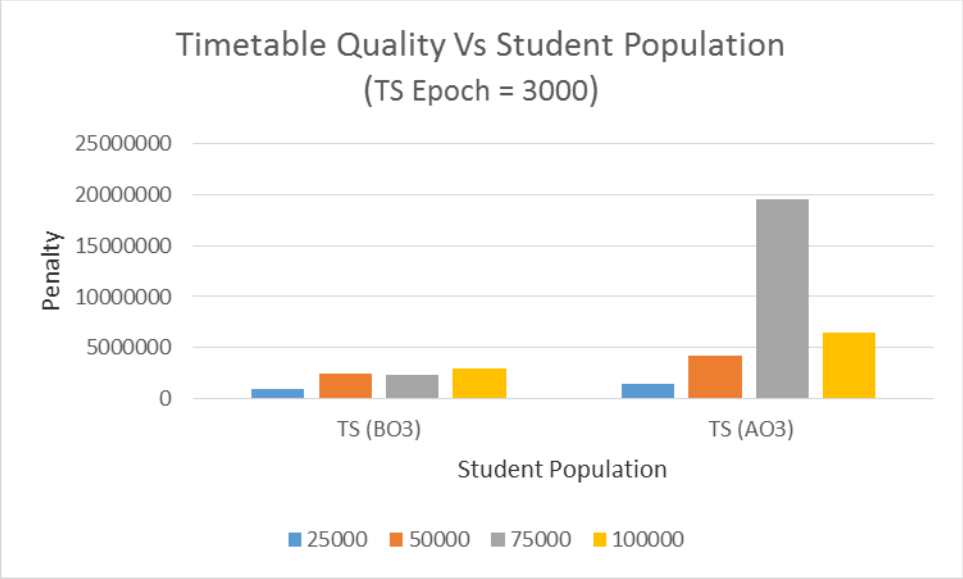

Figure 3: The Effect of varying student population on the Timetable Quality

4.3 Empirical Time Complexity

By varying the number of student population from 25,000 to 100,000, while keeping the TS epoch at 3000, the time and space complexity of the TS algorithm were determined from the captured data. Figure 4 showed that the implemented TS algorithm is linearly increasing with increase in student population, implying a time complexity of O(n).

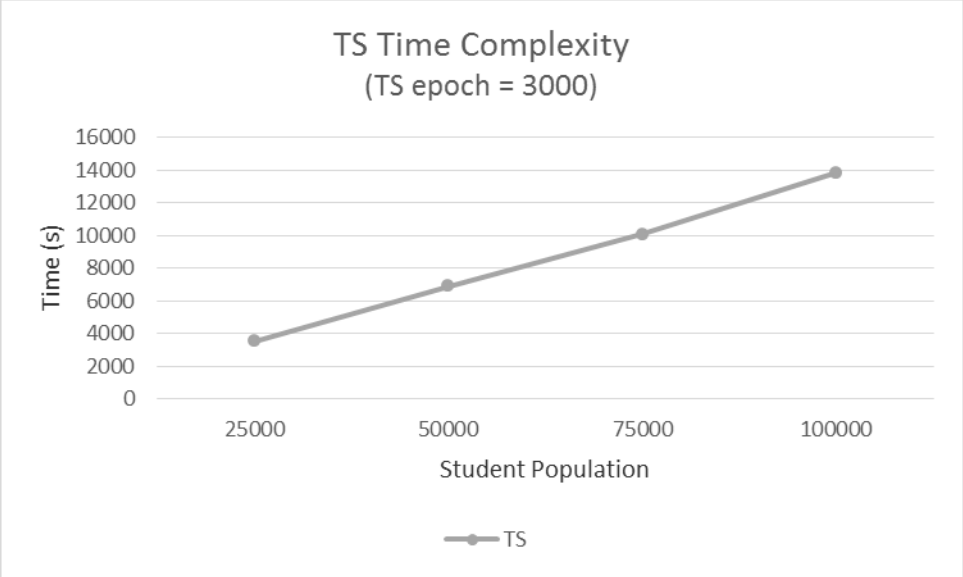

Figure 4: Empirical Time Complexity of the TS Algorithms (Avg. of 3 runs)

4.5.1 The Issue of Algorithm Design

The initial design of the TS algorithm resulted in a time complexity of a quadratic order, i.e. $O(n^2)$. In a bid to improve the algorithm's time complexity, all cases where linear searching was used were identified (such as when checking through the list of all students to determine those that enrolled for a particular examination, checking to determine if any student who enrolled for an examination about to be scheduled also enrolled for any of the examinations already scheduled for the same period, etc.) and replaced with binary searching, that is done in logarithmic time ($log_2 n$), where *n* is the input size.

4.4 Empirical Space Complexity

From experiments conducted, it was observed that in the Java Programming environment, the memory consumed by the implemented algorithms differs before garbage collection (GC) and after GC. Data was captured for the two scenarios. Figure 5 shows the space complexity of the TS algorithm without GC and with GC. It can be seen that the TS algorithm exhibited a linear relationship between memory consumed and increase in student population. Secondly, more memory was consumed before GC than after GC. For resource projection purposes, it is recommended that the consumption before GC be used, as such amount of memory was required for the running of the application. The GC only gather memory spaces from objects existing in the memory, which were once used but no longer referenced.

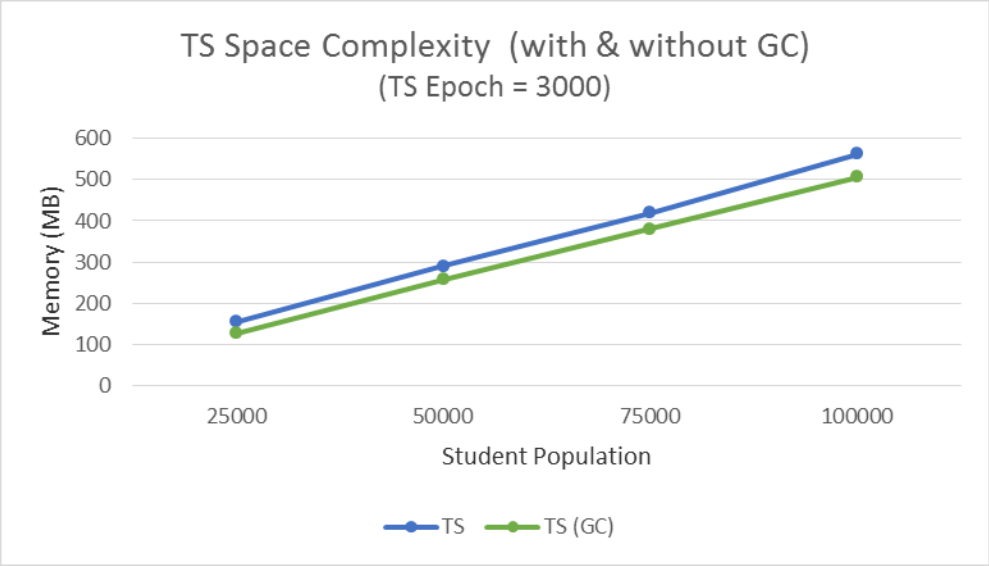

Table 5: Empirical Space Complexity of the TS Algorithm – without and with GC (Avg. of 3 Runs)

4.5 Validation of Result using the University of Melbourne Benchmark Data

The University of Melbourne datasets was introduced by Merlot, Boland, Hughes & Stuckey (2002) at the PATAT conference in 2002 with the objective to minimize the number of occasions of student having two exams consecutively either on the same day or overnight with only two sessions or periods in each day. Two data sets were introduced; the first with 521 exams, 28 sessions, 20656 students and 62248 enrolments and the second with 562 exams, 31 sessions, 19816 students and 60637 enrolments. The best result obtained so far are given in The University of Melbourne (2002) and Qu et al. (2009).

From the experiment conducted using the retrofitted and adapted TS algorithm, with the tabu epoch set at 5000 and tabu list size of 10 for each of the three conducted experimental runs on each data set, the following best TS results were obtained for the data sets Mel-F-01 (set 1) and Mel-S-01b (set 2) are 2337 and 1656, as against the reported best results of 1027 and 1115.

5. CONCLUSION

This paper has demonstrated the ability and effectiveness of the Tabu Search algorithm to solve the University Examination Timetabling Problem. The developed application was used to generate feasible high quality university examination timetables using real-life data, for Bells University of Technology, Ota, For the purpose of algorithm complexity analysis, the data from Bells University was used as a seed to develop databases of 25,000, 50,000, 75,000 and 100,000 students, thereby allowing for the time and space complexities of the algorithm to be investigate.

A high TS epoch usually leads to a better optimized timetable. Increasing the number of student to be examined did not translate to a commensurate decline in quality of timetables; the algorithm is therefore effective for universities with small or large student populations, exhibiting an empirical time and space complexities of order O(n) with increasing student population and in the utilization of systems resources.

It is recommended that for evaluating algorithms with stochastic components, experiments be conducted more than once for each step of the parameter being investigate thereby allowing a realistic evaluation of the algorithm. It is also recommended that this research be adopted by universities for the generation of feasible and optimized university examination timetables.

REFERENCES

1. Boizumault, P., Delon, Y., & Peridy, L. (1996). Constraint logic programming for examination timetabling. The Journal Of Logic Programming.
2. Burke, E. K., Elliman, D. G., Ford, P. H., & Weare, R. F. (1996). Examination timetabling in British universities: A survey. In B. E.K. & R. P. (Eds.), Practice and Theory of Automated Timetabling: Selected Papers from the 1st International Conference. LNCS 1153. (pp. 76-90). Berlin, Heidelberg: Springer-Verlag.
3. Burke, E. K., Elliman, D. G., & Weare, R. (1993). A University Timetabling System based n Graph colouring and Constraint Manipulation. . Journal of Research on Computing in Education., 26.
4. Burke, E. K., & Petrovic, S. (2002). Recent Research Directions in Automated Timetabling. European Journal of Operational Research - EJOR, 140(2), 266-280.
5. Carter, M. W. (1986). A survey of practical applications of examination timetabling algorithms. Operations Research, 34.(2.), 193-202.
6. Carter, M. W., & Laporte, G. (1996). Recent developments in practical examination timetabling. In: E.K. Burke and P. Ross (eds) (1996) 3-21.
7. Carter, M. W., Laporte, G., & Chinneck, J. W. (1994.). A general examination scheduling system. Interfaces, 24, 109-120.
8. Cooper, T. B., & Kingston, J. H. (1995). The complexity of timetable construction problems. In (Burke & Ross, 1996). 283-295.
9. Fowler, J., Kendall, G., & McCollum, B. (Eds.). (2011). Proceedings of the 5th Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2011). (Vol. 2014). Phoenix, Arizona, USA.
10. Garey, M. R., & Johnson, D. S. (1979). Computers and intractability. A guide to the theory of NP-completeness. A Series of Books in the Mathematical Sciences. San Francisco, Calif: WH Freeman and Company.
11. Gendreau, M. (2002). An Introduction to Tabu Search. Retrieved from http://home.ifi.uio.no/infheur/Bakgrunn/Intro_to_TS_Gendreau.htm
12. Gendreau, M., & Potvin, J. (2010). Tabu Search. In M. Gendreau & J. Potvin (Eds.), Handbook of Metaheuristics. International Series in Operations Research and Management Sciences 146, New York Dordrecht Heidelberg London: Springer Science+Business Media, LLC 2010.
13. Glover, F. (1989). Tabu Search - Part I. ORSA Journal on Computing 1(3).
14. Glover, F. (1990). Tabu Search: A Tutorial. Interfaces, 20, 74-94.
15. Komar, M., Grbic, D., & Cupic, M. (2011). Solving Exam Timetabling Using Distributed Evolutionary Computation. Paper presented at the The ITI 2011 33rd Int. Conference on Information Technology Interfaces, Cavtat, Croatia, June 27-30.
16. McCollum, B. (2006). University Timetabling: Bridging the Gap between Research and Practice. Paper presented at the The Proceedings of the 6 th International Conference on the Practice and Theory of Automated Timetabling (PATAT).
17. Pinedo, M. L. (2008). Scheduling: Theory, Algorithms, and Systems (3 ed.). 233 Spring Street, NewYork, NY 10013, USA: Springer Science+Business Media, LLC.
18. Qu, R., Burke, E. K., McCollum, B., Merlot, L. T. G., & Lee, S. Y. (2009). A Survey of Search Methodologies and Automated System Development for Examination Timetabling. Journal of Scheduling, 12(1), 55 - 89.
19. Ross, P., Hart, E., & Corne, D. (Eds.). (1997). Some observations about GA-based exam timetabling. (Vol. 1408). Toronto, Canada: Springer-Verlag.
20. Schaerf, A. (1999). A survey of automated timetabling. . Artificial Intelligence Review., 13(2), 87-127.
21. Wren, A. (1996). Scheduling, timetabling and rostering - A special relationship? In E. K. Burke & P. Ross (Eds.), Practice and Theory of Automated Timetabling: Selected Papers from the 1st International Conference. LNCS 1153. (pp. 46-75.). Berlin, Heidelberg: Springer-Verlag, .

APPENDIX A

BELLS UNIVERSITY OF TECHNOLOGY, OTA
2014/2015 EXAMINATION TIMETABLE

Days	Courses/No of Students	Venues(Count)	Invigilators' IDs
Mon 9.00am-12.00pm	MEE207(157)-HD(157)	MPH(240)	38; 39; 50; 64; 65
	BUS101(133)-MPH(133)	HD(230)	66; 67; 68; 89; 99
	ECO309(107)-MPH(107)	Rm5(56)	103; 104
	ARC101(68)-HD(68)	MScStu 2(43)	105
	ARC209(54)-Rm5(54)	MScStd1(40)	108
	CSC505(41)-MScStu 2(41)	Adenuga 2(34)	110
	MEE307(38)-MScStd1(38)	BioCLab(30)	120
	BUS411(21)-BioCLab(21)	SoftWLab 2(30)	121
	HRM305(20)-DigitalLab(20)	BioLab3(25)	123
	CHM207(19)-TRLab(19)	DigitalLab(20)	124
	BIO203(15)-FoodPLab(15)	CtrlMicLab(20)	134
	MEE409(12)-CtrlMicLab(12)	FoodPLab(20)	135
	CEN403(12)-SoftWLab 2(12)	TRLab(20)	137
	PMT205(11)-SoftWLab 2(11)	AnalytLab(10)	147
	EST207(11)-Adenuga 2(11)	BuildgTech(10)	155
	PMT405(10)-AnalytLab(10)	ButechLab(10)	159
	ARC403(10)-BuildgTech(10)		
	BIC401(9)-BioCLab(9)		
	BIC311(8)-CtrlMicLab(8)		
	SGF205(7)-SoftWLab 2(7)		
	FDT405(6)-ButechLab(6)		
	BTE303(6)-Adenuga 2(6)		
	BUS405(6)-Adenuga 2(6)		
	BME303(6)-Adenuga 2(6)		
	CHM307(6)-BioLab3(6)		
	MKT303(6)-BioLab3(6)		
	PHY309(5)-HD(5)		
	BDT301(4)-ButechLab(4)		
	CHM411(3)-FoodPLab(3)		
	ECO411(3)-Adenuga 2(3)		
	TCE403(2)-Rm5(2)		
	EST403(2)-MScStu 2(2)		
	BTE403(2)-MScStd1(2)		
	URP313(2)-FoodPLab(2)		
	URP415(2)-Adenuga 2(2)		
	NUD311(2)-BioLab3(2)		
	MCT407(2)-BioLab3(2)		
	BDT403(2)-BioLab3(2)		
	BTE507(1)-TRLab(1)		
	SGF307(1)-BioLab3(1)		
	AMS421(1)-BioLab3(1)		
	QTS403(1)-BioLab3(1)		
	TML505(1)-BioLab3(1)		
	NUD411(1)-BioLab3(1)		
	NUD203(1)-BioLab3(1)		
	QTS305(1)-BioLab3(1)		
Mon 12:30pm-3.00pm	ARC413(11)-DigitalLab(11)	Adenuga 3(80)	171; 177
	BIC305(10)-AnalytLab(10)	E-Lib(80)	178; 179
	FDT303(4)-DigitalLab(4)	StrOfMLab(60)	181; 185
	BME405(1)-DigitalLab(1)	Rm5(56)	194; 196
	ACC403(41)-MScStu 2(41)	MScStu 2(43)	200

	SGF201(7)-BuildgTech(7)	Adenuga 4(42)	218	
	CSC307(59)-StrOfMLab(59)	MScStd1(40)	219	
	TCE401(2)-MScStu 2(2)	Adenuga 2(34)	221	
	ECO303(39)-MScStd1(39)	SoftWLab 2(30)	236	
	AMS417(1)-StrOfMLab(1)	BioCLab(30)	238	
	PHY307(6)-ButechLab(6)	BioLab3(30)	240	
	MEE313(29)-BioCLab(29)	DigitalLab(20)	243	
	PMT501(2)-BuildgTech(2)	TRLab(20)	246	
	MIC307(9)-TRLab(9)	FoodPLab(20)	252	
	EEE411(30)-SoftWLab 2(30)	CtrlMicLab(11)	253	
	EST407(2)-DigitalLab(2)	AnalytLab(10)	254	
	EEE415(1)-MScStd1(1)	BuildgTech(10)	255	
	NUD305(2)-DigitalLab(2)	ButechLab(10)	258	
	BIC421(1)-BioCLab(1)			
	MKT301(7)-TRLab(7)			
	SGF301(1)-BuildgTech(1)			
	QTS401(1)-ButechLab(1)			
	ITP505(12)-FoodPLab(12)			
	BDT413(2)-ButechLab(2)			
	EEE309(78)-Adenuga 3(78)			
	BME307(6)-FoodPLab(6)			
	BDT303(4)-TRLab(4)			
	FNB401(4)-CtrlMicLab(4)			

APPENDIX B: Bell University of Technology Data (Venue Details)

Venues and Their Corresponding Capacity

ID	Name	Capacity
1	Room6	56
2	Room5	56
3	Room4	48
4	Room3	48
5	Room2	48
6	Room1	56
7	PhyLab2	30
8	PhyLab1	30
9	Mph	240
10	HallD	230
11	ChemLab2	30
12	ChemLab1	30
13	BioLab3	20
14	BioLab2	30
15	BioLab1	30
16	BioChemLab	20
17	B9	56
18	B8	50
19	B7	40
20	B6	40
21	B5	40
22	B4	48
23	B12	48
24	B11	56
25	B10	56