NIPES

Journal of Science and Technology Research

Journal homepage: www.nipesjournals.org.ng

Development of a Hybrid Tabu Search and Genetic Algorithms for the Examination Timetabling Problem

Ezike J.O.J[a]*, Oyeleye C.A.[b], Olabiyisi S.O.[b], Omidiora E.O.[b]

[a]Dept. of Computer Science and Information Technology, Bells University of Technology, Ota, Nigeria
[b]Dept. of Computer Science and Engineering, Ladoke Akintola University of Technology, Ogbomoso, Nigeria
**Corresponding Author Email: joezike@bellsuniversity.edu.ng, 08035057550*

Article Info

***Keywords*:**
Genetic Algorithm, Tabu Search, Automated Examination Timetabling, Hybrid Algorithm, Combinatorial Optimization

Received 16 June 2020
Revised 22 June 2020
Accepted 25 June 2020
Available online 31 August 2020

https://doi.org/10.37933/nipes/2.3.2020.14

https://nipesjournals.org.ng

Abstract

Genetic Algorithms (GA) and Tabu Search Algorithm (TSA) are amongst the leading research approaches for solving the Examination Timetabling Problem (ETP), however, both algorithms are not optimal. GA returns poor solution, uses excessive memory, experience damage to solution during crossover while solving the ETP. TSA consumes much time, can easily miss some regions of the search space since it uses one solution, and may fail to generate some neighborhood candidate solution. TSA also selects best solution based on the current steps without taking future steps into consideration. This research developed a hybrid of GA and TSA, the GATS algorithm, with the aim of mitigating against the GA's and TS weaknesses to produce higher quality results when solving the ETP. The ETP was modeled as an optimization problem, implemented in Java for the three algorithms and experimented with dataset from Bells University of Technology, Ota. The algorithms' performances were evaluated using first Order Conflict Counts (OCC) and second OCC for students and invigilators respectively, as well as with space complexity. The GA, TSA and GATSA yielded average first Order Conflict Counts (OCC) of 0.0, 0.0 and 0.0 for both students and invigilators. They yielded average second OCC of 5228.5, 18.8 and 0.7 for students and, 0.0, 0.0 and 0.0 for invigilators respectively. The Developed GATSA produced higher quality timetables than TSA and GA, and consumes similar amount of memory as the TSA and has an empirical space complexity of O(n).

1. Introduction

Educational timetabling, one of the most widely studied of all variants of the timetabling problem [1, 2], has been noted to be very time-consuming, and the quality of the timetable produced have 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 [3, 4]. A summary of the three broad categories of educational timetabling is given in [1] and [5]. The focus of this research is on solving the Examination Timetabling Problem (ETP).

1.2 The Examination Timetabling Problem (ETP)

The ETP 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[6-8]. No classical operations research (OR) approach is directly applicable for solving the ETP [9]. Some of the approaches used over the years are listed in [10], [11] and [5]. Among the leading approaches are Tabu search (TS) and Genetic Algorithm (GA). However, the two algorithms are not optimal: GA returns poor quality result with increasing problem size; damage is also done to solution during cross-over, lastly, GA utilizes excessive memory before returning result [12, 13]. On the other hand, TS which consume much time and uses only one solution can easily miss some areas of the search space. TS with a larger set of parallel solutions does not exchange information [14]. TS also choose the best solution based on the current step and position without taking the future steps into consideration. It suffers in handling the solution search space diversity as some neighborhood candidate solution is not necessarily to be generated [15]. In [5], [16] and [17] it was noted that many of the successful methodologies that have appeared in resent timetabling literature represent hybridization of a number of techniques. These hybrid meta-heuristic algorithms are now used to find high quality solution to an ever-increasing number of complex, ill-defined real world combinatorial problems. They often perform substantially better than their "pure" counterparts when properly designed. It is currently believed that choosing an adequate combination of multiple algorithmic concepts is the key for achieving top performance in solving most difficult problems [17]. This research aim to develop a hybrid of Genetic and Tabu Search Algorithms for solving the ETP that through synergy, mitigate against the GA's and TS weaknesses and capitalizes on their strength to produce higher quality results while consuming less computing resources.

1.3 Genetic Algorithm

Genetic Algorithm (GA), an evolutionary algorithm, has been widely studied, particularly in hybridization with local search methods (sometimes called *memetic* algorithms) and have recorded some success [5]. Genetic Algorithm mimics the evolution in nature by manipulating and evolving a population of solutions within the search space. Solutions are coded as chromosomes and are evolved by a reproduction process using crossover and mutation operators. The aim is to obtain increasingly improving solutions through a number of generations. A template for GA is given in [18]. In [19], it was shown that direct representation in GA was incapable of dealing with certain problem structures in some specially generated graph coloring problems. Further evidence to this was given in [2]. Result generated by GA may not be as good as those generated by some local search-based algorithms like TS and simulated annealing as noted in [20, 21]. As such, hybridizing GA and local search-based techniques is often desirable.

1.4 Tabu Search

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 [22], as a result, TS have often been used to implement "hyperheuristics," a heuristics which choose between heuristics in order to solve a given optimization problem as seen in [23] and [24]. The goal of hyperheuristic applications is usually that of developing automated scheduling methods which

are not restricted to one problem. 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 [25], TS was noted to be amongst the most effective, if not the best in tackling difficult problems and finding good solutions to the large combinatorial problems encountered in many practical settings, resulting in TS popularity 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 [26]. Parameters, such as that for the tabu list, stopping criteria, usually need to be fine-tuned in line with the problem being solved to enable efficient and effective performance of the algorithm. The challenge with TS which use only one solution is that it can easily miss some areas of the search space, while TS with a larger set of parallel solutions does not exchange information [14]. A template for TS is given in [27],

1.5 Hybridization of GA and TS Algorithms

A hybrid algorithm incorporate concepts and optimization techniques from different algorithms in order to better solve an ever-growing number of complex, ill-defined real world combinatorial problems [28]. In fact, many of the successful and current methodologies that have appeared in resent timetabling literature represent hybridization of a number of techniques [5]. Well designed-hybrids often perform substantially better than their “pure” counterparts. As a result, a number of dedicated scientific events such as workshops on metahuristics and conferences are dedicated to hybridization techniques [29], [30], [31], [32], [33], [34], [35]. However, it should be noted that compared to the classical “pure” strategies, metaheuristic hybrids are significantly more complex, requiring more substantial efforts in development and tuning, and does not necessary automatically translates into better performance. Adequate design and appropriate tuning efforts, which increases with the system’s complexity, is often mandatory [17]. A classification of hybrid metaheuristics based on some basic characteristics and design templates used in implementing successful hybrid algorithms was given in [17].

2. Methodology

Table 1 summarizes the constrained used in Bells University of Technology, Nigeria, as well as key constrains considered in literature as can be seen in [5, 36]. These constraints are grouped as hard and soft with penalties attached in view of their severity if violated. The choice of penalty values follows recommendations in literature and determined from trial experimental runs; they were found to be effective in guiding the search, TS in particular.

Table 1: Constraint Types and Penalty Values

Type	Code	Definition	Penalty Cost
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 in a given day.	1
	SC2	No student should be scheduled to sit for two consecutive exams in a day.	100
	SC3	No student should be scheduled to sit for three consecutive exams in a day.	100,000
	SC4	No teacher should be scheduled to invigilate two non-consecutive exams in a day.	1
	SC5	No teacher should be scheduled to invigilate two consecutive exams in a day.	100
	SC6	No teacher should be scheduled to invigilate three consecutive exams in a day	100,000

In literature, HC1, HC2 and HC3 violations (see Table 1) are referred to as first-order conflicts [24, 37] In this paper, second order conflicts refers SC3 and SC6 violations, third order conflict to SC2 and SC5, while fourth-order conflicts to SC1 and SC4.

2.1 Mathematical Formulation of the ETP

From constraints in Table 1, the ETP is formulated as follows:

2.1.1 Resource Definition

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

P: A set of p periods (or time-slots), $p_1, p_2, \ldots, p_p$.

D: A set of d days (i.e. examination duration), $d_1, d_2, d_3, \ldots, d_d$: a day comprise 1 to 3 periods.

E: A set of e examinations, $e_1, e_2, e_3, \ldots, e_e$

E_p: A set of β examinations scheduled in period p_p, that is, e_1p_p, e_2p_p, e_3p_p ... $e_\beta p_p$

S: A set of s students, $s_1, s_2, s_3, \ldots, s_s$, in the campus of the university

L: A set of l teachers, $l_1, l_2, l_3, \ldots, l_l$ in the university.

R: A set of s course registration lists for all students in the campus, that is, $R_{s_1}, R_{s_2}, R_{s_3}, \ldots, R_{s_s}$

V: A set of all v venues in the campus, that is, $v_1, v_2, v_3, \ldots, v_v$

2.1.2 Decision Variables

All decision variables can have a value of 0 or 1.

i. $e_m p_{kd_h}$: is the instance of an examination e_m scheduled in period p_k of day d_h. $e_m p_{kd_h}$= 1 if schedule or 0 otherwise.
ii. $s_j e_m$: is the instance that student s_j enrolled for examination e_m. $s_j e_m$ = 1 if student enrolled or 0 otherwise.

iii. $s_i e_m v_y$: is the instance that student s_j who enrolled for examination e_m is scheduled for venue v_y . $s_i e_m v_y$= 1 if scheduled or 0 otherwise.
iv. $s_j e_m p_{kd_h}$: is the instance that student s_j is to sit for examination e_m scheduled for period p_k of day d_h. $s_j e_m p_{kd_h}$= 1 if student s_j is scheduled or 0 otherwise.
v. $l_g v_y p_{kd_h}$: is the instance of a teacher l_g scheduled to be in venue v_y at period p_k of day d_h.

2.1.3 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 $duration\ (p_k)$ denote the number of hours in period p_k.

2.1.4 Assumptions

The following are the assumption made in carrying out the research experimentation:
i. There are only three (3) periods in a day, that is,

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

Where p_k is the last period of the previous day and $k\ \in \mathbb{N}_0$.
ii. Each period is of a fixed 3-hour duration, that is,

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

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

2.2 Constraint and Objective Function Modelling

Using the decision variables defined, constraints used in this research (see Table 1) are modelled as follows:

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

$$HC1 = \sum_{j=1}^{e} s_i e_j p_{kd_h} \leq 1 \qquad (1)$$

HC2: No Teacher should be scheduled to be in more than one venue at the same time (period) in any given day.

$$HC2 = \sum_{k=1}^{p} l_g v_y p_{kd_h} \leq 1 \qquad (2)$$

HC3: All scheduled venues must have adequate capacity to contain the students that enrolled for the examinations scheduled in them. If x is the number of students that enrolled for the examination e_m, then:

$$HC3 = \sum_{i=1}^{x} s_i e_m v_y\ =\ n_s(e_m) :\ n_s(e_m) \leq cap(v_y) \qquad (3)$$

SC1**:** No student should be scheduled to sit for two non-consecutive (or more than one) examination in a day.

$$SC1 = \sum_{k=1}^{k=3} s_i e_m p_{kd_h} \leq\ 1 \qquad (4)$$

SC2: No student should be scheduled to sit for two consecutive examinations in a given day.

$$SC2 = \sum_{k=a}^{k=a+1} s_i e_m p_{kd_h} < 2 \quad (5)$$

where a = 1 *or* 2: a + 1 ≤ 3.

SC3: No student should be scheduled to sit for three consecutive examinations in a day.

$$SC3 = \sum_{k=1}^{k=3} s_i e_m p_{kd_h} < 3 \quad (6)$$

SC4: No teacher should be scheduled to invigilate two non-consecutive examinations in a day.

$$SC4 = \sum_{k=1}^{k=3} l_g e_m p_{kd_h} \leq 1 \quad (7)$$

SC5: No teacher should be scheduled to invigilate in two consecutive periods.

$$SC5 = \sum_{k=a}^{k=a+1} l_g e_m p_{kd_h} < 2 \quad (8)$$

where a = 1 *or* 2 : a + 1 ≤ 3.

SC6: No teacher should be scheduled to invigilate in three consecutive periods.

$$SC6 = \sum_{k=1}^{k=3} l_g e_m p_{kd_h} < 3 \quad (9)$$

2.3 The ETP Objective Function

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

$$f_o = f_p = f_{p\,(hard)} + f_{p\,(Students)} + f_{p\,(invigilators)} \quad (10)$$

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. Equation (10) can be written as:

$$f_o = f_p = w_h \sum_{i=1}^{i=4} HC_i + \sum_{j=1}^{j=3} w_j SC_j + \sum_{k=1}^{k=3} w_k SC_k \quad (11)$$

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 (see Table 1), that is,

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

With the objective function stated in terms of the penalty function as a minimization problem, Equation 11 can be simplified to:

$$f_p = w_h \sum_{i=1}^{i=4} HC_i + \sum_{j,k=1}^{q=z=3} w_j \left(SC_j + SC_k\right) \quad (12)$$

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 (12), subject to the following constraints:

$$HC1 = \sum_{j=1}^{e} s_i e_j p_{kd_h} \leq 1 \quad (1)$$

$$HC2 = \sum_{k=1}^{p} l_g v_y p_{kd_h} \leq 1 \quad (2)$$

$$HC3 = \sum_{i=1}^{x} s_i e_m v_y \; = \; n_s(e_m) : \; n_s(e_m) \leq cap(v_y) \quad (3)$$

$$SC1 = \sum_{k=1}^{k=3} s_i e_m p_{kd_h} \leq \; 1 \quad (4)$$

$$SC2 = \sum_{k=a}^{k=a+1} s_i e_m p_{kd_h} < \; 2 \quad (5)$$

$$SC3 = \sum_{k=1}^{k=3} s_i e_m p_{kd_h} < \; 3 \quad (6)$$

$$SC4 = \sum_{k=1}^{k=3} l_g e_m p_{kd_h} \leq \; 1 \quad (7)$$

$$SC5 = \sum_{k=a}^{k=a+1} l_g e_m p_{kd_h} < \; 2 \quad (8)$$

$$SC6 = \sum_{k=1}^{k=3} l_g e_m p_{kd_h} < \; 3 \quad (9)$$

2.4 Timetable Representation

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

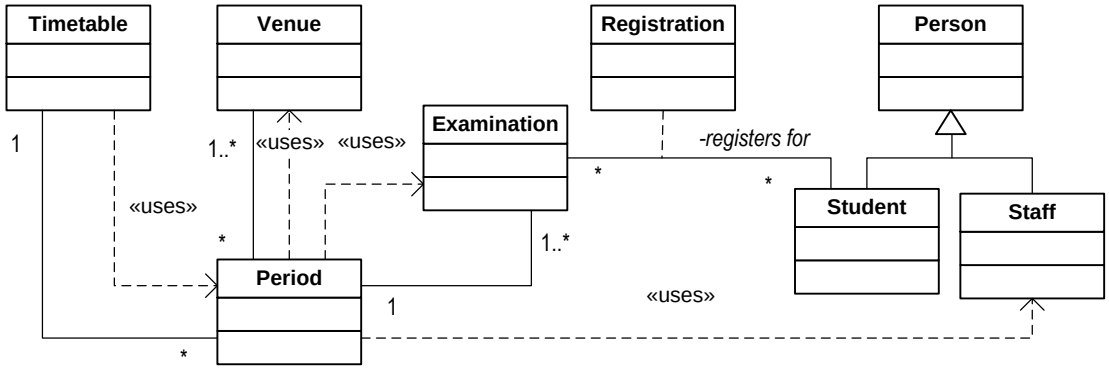

Figure 1: Classes used in the object-oriented design of the timetable object

The timetable solution was implemented using Java objects containing list of period objects, which in turn contains list of examinations, list of venue and list of invigilators. For the GA, the population of individual solution was contained in a list for processing.

2.5 Implemented Algorithms

The pseudo code for the developed application that implemented the GA, TS and GATS algorithms is given below.

Pseudo Code for Timetabling Application

1 Start
2 Declare and Initialize working variables
3 Load Data from Database (Venues, Courses, Registrations, Students)
4 Extract Course Registration List for each student in semester
5 Extract student's list for each enrolled course
6 Create Initial Population (or Solution)
7 Optimize Solution (Population) (with GA, TS or hybrid algorithms)
8 Allocate examinations & Students to actual venues
9 Schedule Invigilators to venues
10 Display Timetable
11 End.

Section 2.5.1 presents a formal description of the implemented GA algorithm as indicated in line 7 of the generic application pseudo code.

2.5.1 Description of Genetic Algorithm (GA)

Using the following declared variables, the GA algorithm is illustrated in Table 2.

H = List of n individual forming initial GA population
H' = List of n individual forming final pupation
P = List of two selected individuals (parent)
T = List of individuals selected from H for tournament
t = an individual (that is, a single timetable solution)
StudSemRegL = the list of all examinations registered for by each student in the semester.

Table 2: Description of the Implemented Genetic Algorithm

	Algorithm GA
	Input: H, studSemRegL
	Output: H'
1	T ← ∅ // Tournament Individuals List
2	P ← ∅
3	t ← $null$
4	x ← $f(H)$ // the number of competing individuals, a function of H
5	**for**(i = 1 to N, do)
6	evaluateTitness(H_i)
7	**endfor**
8	**while**(not stopCondition)
9	T ← selectIndividualsForTournament(H,x)
10	P ← peformTornamentSelection(T)
11	t ← peformCrossOverWithMutation(P, studSemRegL)
13	evaluateFitness(t)
14	normalisePopulation(H, t)
15	**endwhile**
16	H' ← sort(H)
17	**return** H'

(a) The Crossover Operator

The GA implements a heuristic crossover operator identical to that shown in Figure 2 as described in [38]. The crossover process elicit all common exams in both periods and introduces some other examinations in the pool or from that left over in previous crossing, while ensuring feasibility.

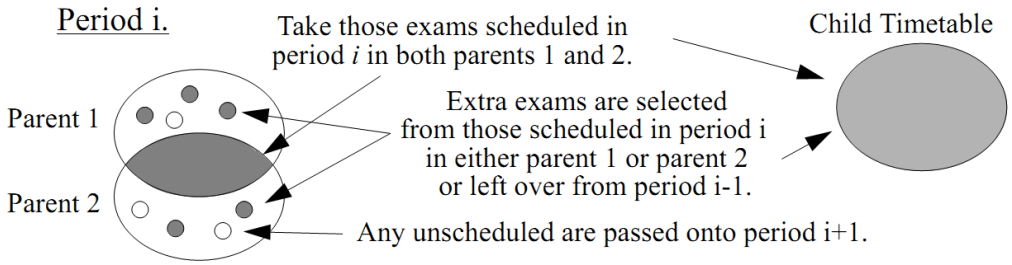

Figure 2: A heuristic hybrid crossover operator [38]

(b) The GA's Mutation Operator

The mutation operator was incorporated into the crossover function as done in [38], i.e. by adding examination into the current search that would otherwise not be considered until a later period. This was necessary because mutation by randomly picking two examination from different period and exchanging them may result in an infeasible examination timetable.

2.5.2 Description of the TS Algorithm

The description of the Tabu Search algorithm is given Table 3.

Table 3: Description of the Implemented Tabu Search Algorithm

```
     Algorithm Tabu Search
     Input: t0, StudSemRegL  // StudSemRegL is the list of all examinations registered
                                for by each student in the semester.
     Output: tbest
1    t ← t0
2    tbest ← t0
3    TL ← ∅
4    while (not stoppingCondition)
5         CL ← ∅//CL, the list for all candidate solution
6         CL ← Nt   // N, the neighborhood operator, generates all candidate solution of t
7         t ←applyBest(CL)
8         if (fitness(t) > fitness(tbest))
9              tbest← t
10   endif
11        TL ← t
12        if((size(TL) > maxSize(TL))
13             removeFirst(TL)
14        endif
15        if(DiversificationCondition)
16             diversify()
17        endif
18   endwhile
19   return tbest
```

TS Neighborhood Operator Description

In the TS Algorithm implementation, the neighborhood operator (line 6) generated all the possible "moves" to different new timetable solutions, that is, candidate solutions (see Figure 3) and held these in the generated moves list ((represented by CL). The technique applies an atomic move and produce a resulting candidate timetable solution, evaluate its fitness and then reverse the move. This was done with all the generated moves. The acceptance of the best candidate solution results in the implementation of the move that resulted in that candidate solution.

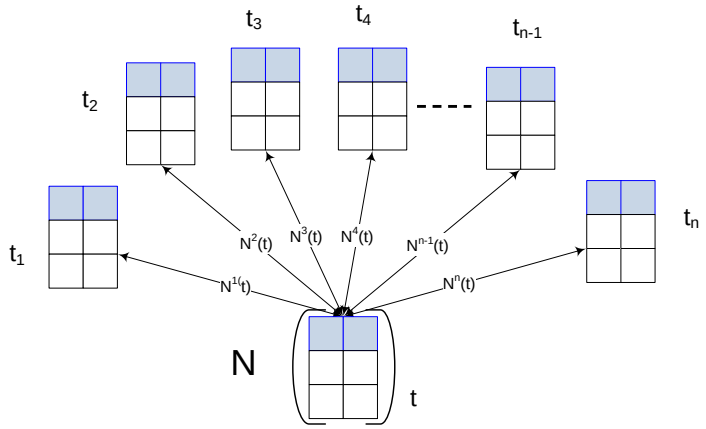

Figure 3: The operation of the Neighborhood operator N on timetable t to produce its neighbors t_1 to t_n.

TS Parameter Tuning

Trial experimental runs were carried out on 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.

2.6 Description of the Developed Hybrid Algorithm (GATS)

The multi-stage approaches listed in [17], which improves solution by embedded methods was the design template used in this work. An enhanced crossover operator was used with the GA employed in the GATS algorithm. In producing a child, the operator searches Parent 1 and extracts all examinations from the day with lowest penalty into a pool; all examinations from the day with highest penalty in Parent 2 were also extracted into the pool while preventing duplicate examinations. Then all examinations that were previously scheduled in any period in the child were removed from the pool; the remaining exams are then scheduled into day 1 of child timetable while ensuring feasibility and avoiding 2nd order (SC3) conflicts where possible. Scheduling for day 2 involve searching for the day in Parent 1 that is next in order of lowest penalty while for Parent 2, day that is next in order of highest penalty is used, etc. Figure 4 illustrates this.

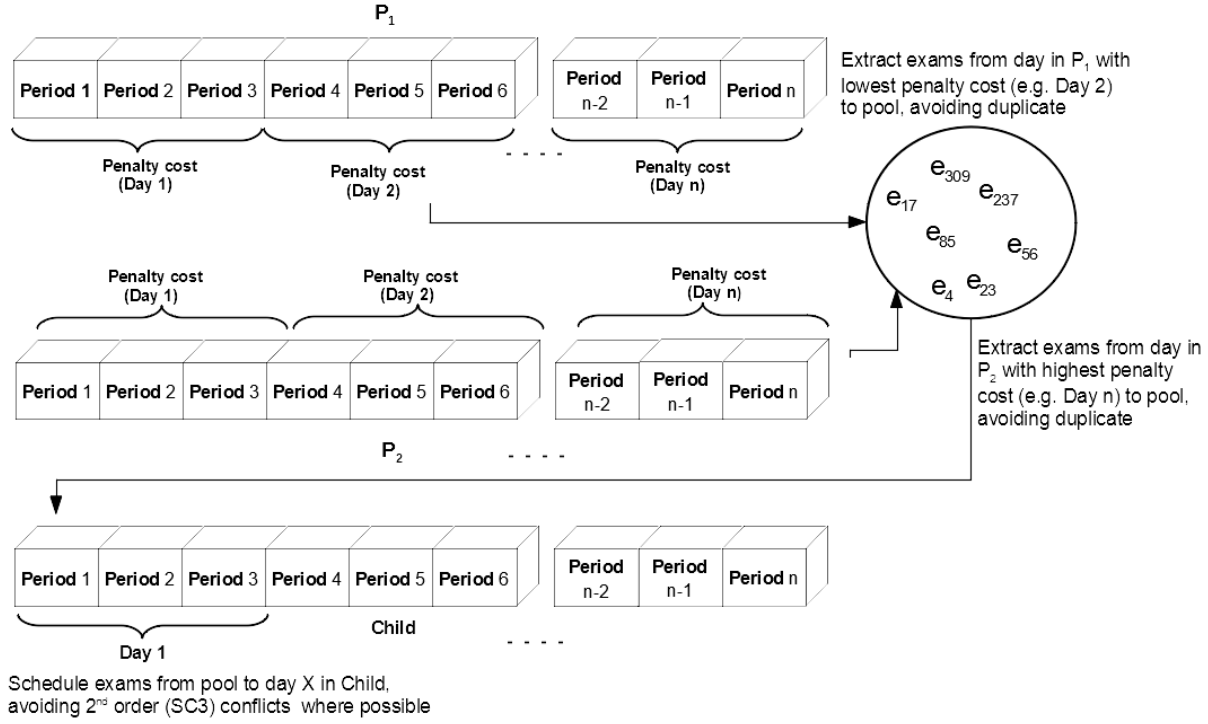

Figure 4: An enhanced Crossover for the GA component of the GATS hybrid algorithm

The GATS algorithm also employed a modified and improved TS algorithm that was used to improve every child solution produced by the GA in each generation. As such, the developed algorithm, the GATS, incorporates features of the GA and TS algorithms. The GATS algorithm is described in Table 4.

Table 4: Description of the developed GATS hybrid Algorithm

	Algorithm GATS
	Input: H , studSemRegL // StudSemRegL is the list of all examinations registered for by each student in the semester.
	Output: H'
1	T ← ∅// Tournament Individuals List
2	P ← ∅ // selected parent's List
3	t ← $null$ // individual solution
4	t' ← $null$ // improved t
5	x ← $f(H)$ // the number of competing individuals, a function of H
6	**for** (i = 1 to N, do)
7	evaluateTitness(H_i)
8	**endfor**
9	**while** (not stopCondition)
10	T ← selectIndividualsForTournament(H,x)
11	P ← peformTornamentSelection(T)
12	t ← peformEnhancedCrossOverWithMutation (P, studSemRegL)
13	t' ←TS-MMG(t) // apply enhanced TS-based optimization technique
14	evaluateFitness(t')
15	normalisePopulation(H, t') // using steady state GA mechanism
16	**endwhile**
17	H' ← sort(H)
18	**return** H'

2.7 Data Gathering

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
3	Total number of Examinations	501
4	Total number of Venues	25 (Total capacity is 1436)
5	Total Number of Invigilating Staff	170

2.8 Experimental Environment

The algorithms were implemented in Java (JDK8u54) on Windows 10 Pro Operating System (64 bits) on a HP Elite book 2560p having Intel ® Core ™ i5-2520M CPU. 8.00GB installed RAM and 500GB HDD (Toshiba) at 540 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.

2.9 Experimental Procedures

The ETP data was loaded from the database and preprocessed. Initial timetable solution (or population) was then generated, as required by the algorithm under consideration: examinations were randomly picked from the examination pool and scheduled into the available venue spaces, while avoiding HC constraints violation (see Table 1). The process resulted in the number of periods in the initial solutions varying from 21 to 28. The initial solutions were then normalized to 30 periods

for all algorithms so as to give a common base for final timetables comparison. The normalization improves the initial timetable quality as courses were spread into additional periods.

The 30 period used implies examination duration of 10 day, or two weeks of five working days each for a three-period-in-a-day examination schedule. A database of 25,000, 50,000, 75,000 and 100,000 students were also generated using the Bells University of Technology dataset as seed. This was used to test the performance of the algorithms for a larger student's population and to determine their space complexities. Two sets of experimental runs were conducted 10 times each for the GA, TS and the GATS hybrid algorithms. The 1st set uses the Bells University dataset of 1896 students while the 2nd set uses the 25,000 student population dataset. The results captured are analyzed and reported in the next section.

3. 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. Table 5 summaries the result and salient data on the performance of the three algorithms.

Table 5: Result Summary from 1st Set of Experimental Runs using Bells University Dataset
(Parameters: Student Population = 1896, TS Epoch-Time bound (300s), GA Population = 100, GA Gen = Time-bound (300s), GATS Pop = 10, GATS Gen/Epoch-Time bound (300s)).

Descriptions	**GA**	**TS**	**GATS**
Constraints Violation	**Best of 10 Runs**		
First Order Conflict Count (OCC)	0	0	0
2nd OCC	356	0	0
3rd OCC	2402	601	336
4th OCC	1139	2256	1600
Penalties of Best Generated Timetable	35841339	62356	35200
% Improvement (GATS comparison)	99.9%	43.5%	0.0%
Average of 10 Runs			
1st OCC	0	0	0
2nd OCC	454.78	1.5	0.1
3rd OCC	2584.89	865.5	824.9
4th OCC	1172.44	1599.1	1438.6
Penalties of Best Generated Timetable (Average)	45737439	238149.1	93928.6
Average % Improvement (GATS comparison)	99.8	60.6%	0.0%
Invigilators			
1st OCC	0	0	0
2nd OCC	0	0	0
3rd OCC	0	0	0
4th OCC	0	0	0

Table 6: Result Summary from 2nd Set of Experimental Runs using Generated Dataset
(Parameters: Student Population = 25,000, TS Epoch-Time bound (5400 s), GA Population = 100, GA Gen = Time-bound (5400 sec), GATS Pop = 10, GATS Gen/Epoch-Time bound (5400 s)).

Descriptions (25k)	GA	TS	GATS
Constraints Violations	**Best of 10 Runs**		
First Order Conflict Count (OCC)	0	0	0
2nd OCC	4801	0	0
3rd OCC	39624	8952	5085
4th OCC	10110	21123	24126
Final Timetable Penalties (Best result)	484072510	916323	532626
% Improvement (GATS)	99.9%	41.9 %	0.0%
Time (s) to eliminate 2nd OCC	Nil	1120s	181s
Constraints Violations	**Average of 10 Runs**		
First OCC	0	0	0
2nd OCC	5228.5	18.8	0.7
3rd OCC	33375.0	19222.7	11114.5
4th OCC	19341.3	19222.7	19792.4
Final Timetable Penalties (Best result)	526206841.3	2891433.0	1201242.4
% Improvement (GATS)	99.8%	58.5%	0.0%
Time (s) to eliminate 2nd OCC	Nil	1707.7s	509.4s

3.1 The Effect of Processing Time (GA, TS and GATS) on Timetable Quality

Figure 5 showed the effect of varying the processing time for the three algorithms from 0 to 5,400 seconds (i.e. 1½ hours), while Figure 6 illustrate the quality of the Final Timetable generated by each of the algorithm.

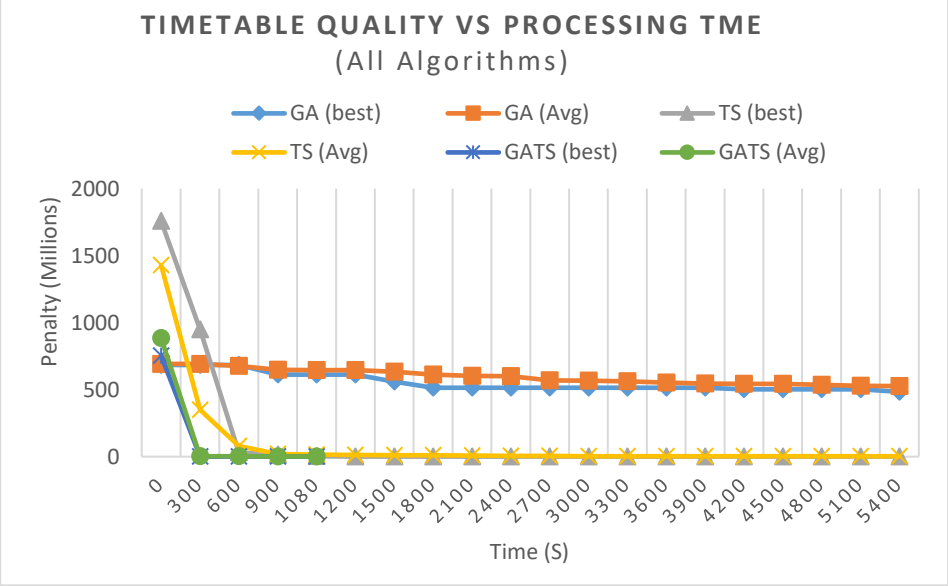

Figure 5: Timetable penalty reduction with time

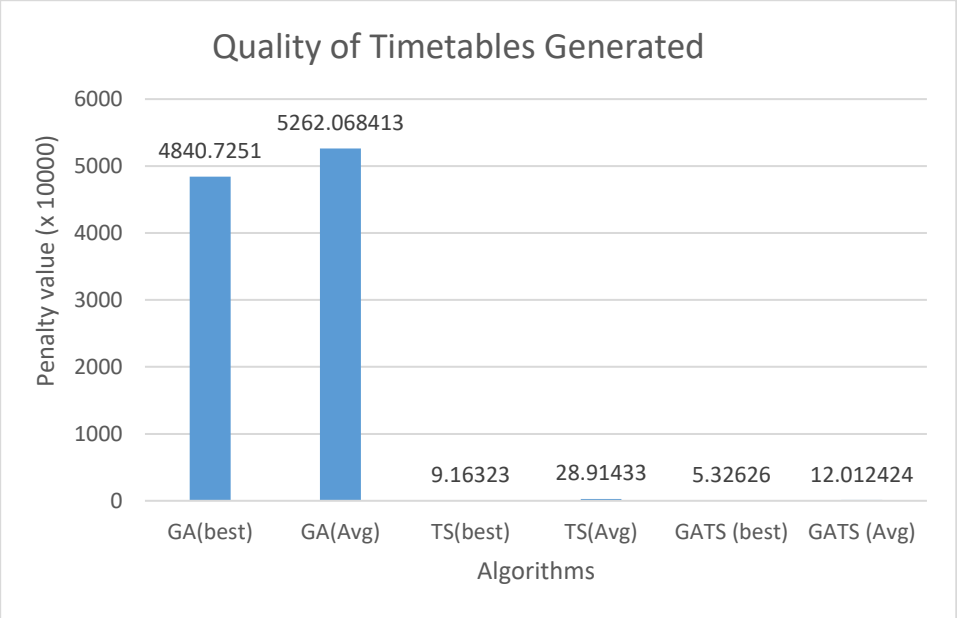

Figure 6: Comparison of Generated Timetables Qualities- All Algorithms (Best and Average)

The GATS hybrid algorithm produced more quality timetables than the GA and TS algorithms. For the 1st set of experimental run, the quality of result produced by the GATS algorithm in the best case was 99.9% better than the GA and 43.5% than the TS algorithms. For the second set, the GATS's best result was again 99.9% better than that of GA and 41.9 % than that of TS. The GA performance was poor; this was due to limitation of the crossover operator, the key search operator in GA. GA may indeed not be very effective in solving highly constrained problems like the ETP as noted in [2].

3.2 Eliminated 2nd Order Conflict Violation

This soft constraint SC3, the constraint of student not sitting for three consecutive examinations in a day (see Table 1) is the most costly as no hard (HC) constraints were broken. The TS algorithm returned timetable solutions with the SC3 constraints eliminated in three out of the 10 experimental runs while the GA algorithm returned no such solution. The developed GATS hybrid algorithm returned timetable solution with the SC3 constraints eliminated in nine out of the 10 experimental runs. In actual fact, a total of 18 of such solutions with SC3 constraint violation removed were actually returned by the GATS algorithm with its population of 10 solutions over five generations within the specified time of 5400s (see Figure 7). Figure 8 is an illustration of the time take by the TS and GATS algorithms to achieve the elimination of the SC3 constraint violation.

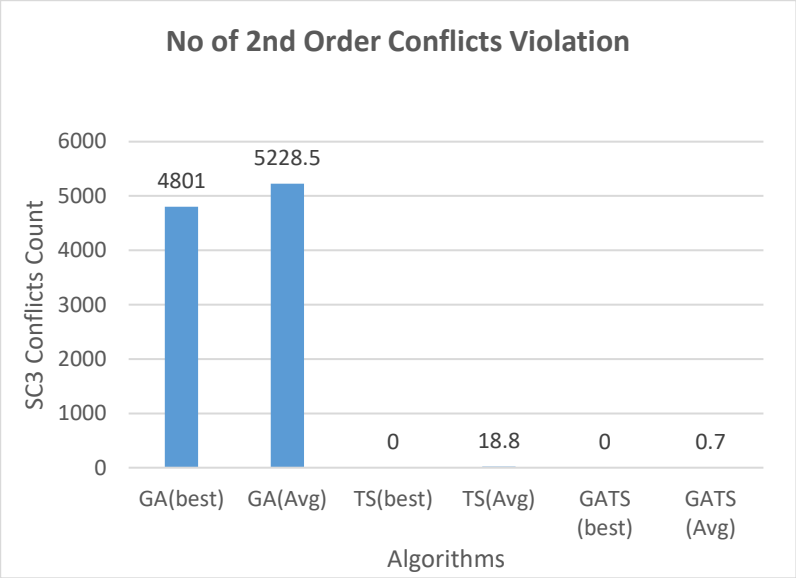

Figure 7: Comparison of All Algorithms based on Number of Students Having 3 Consecutive Exams (Best and Avg. of 10 Runs)

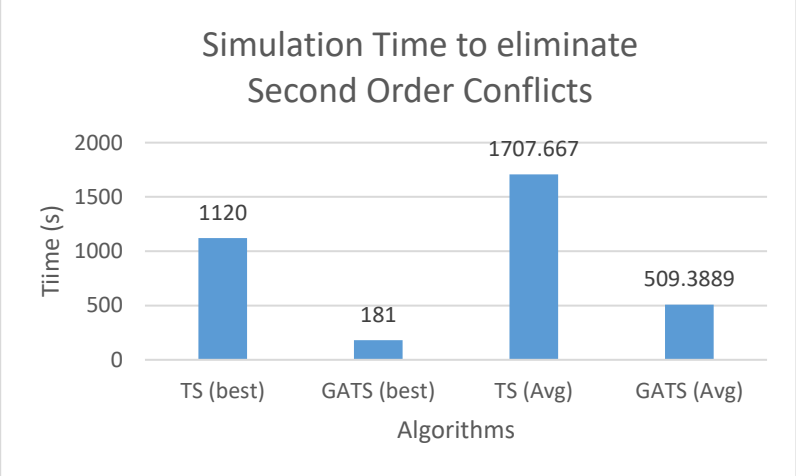

Figure 8: Time taken to eliminate the Second Order Conflicts (Best and Avg. of 10 Runs)

3.3. Evaluation based on Empirical Space Complexity

The empirical Space Complexity of the three Algorithms was determined using the generated databases of students' populations varying from 25,000 to 100,000 in steps of 25,000. The memory consumed by the GA, TS and the GATS algorithms before and after garbage collection differs. Figure 9 shows the space complexity without Garbage Collection (GC). The graph shows a linear plot for the GA and TS algorithm and approximately linear for the GATS algorithm. From Figure 9, the GA used more memory during program execution compared to the TS algorithm. Memory consumption of the GATS algorithm is comparable to that of the TS algorithm. Figure 10 shows the complexity with GC, and shows a linear relationship with population increase. All three algorithms showed identical complexity of order $O(n)$ without and with garbage collection done.

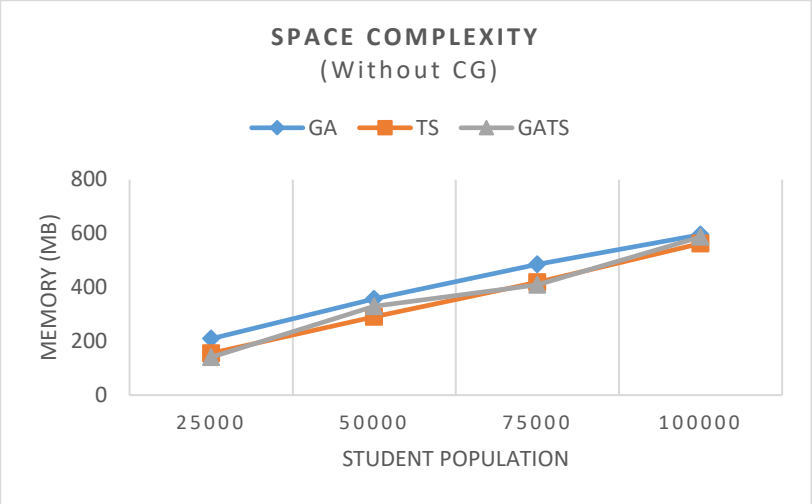

Figure 9: A Comparison of Space Complexity of GA, TS and GATS Algorithms (without GC)

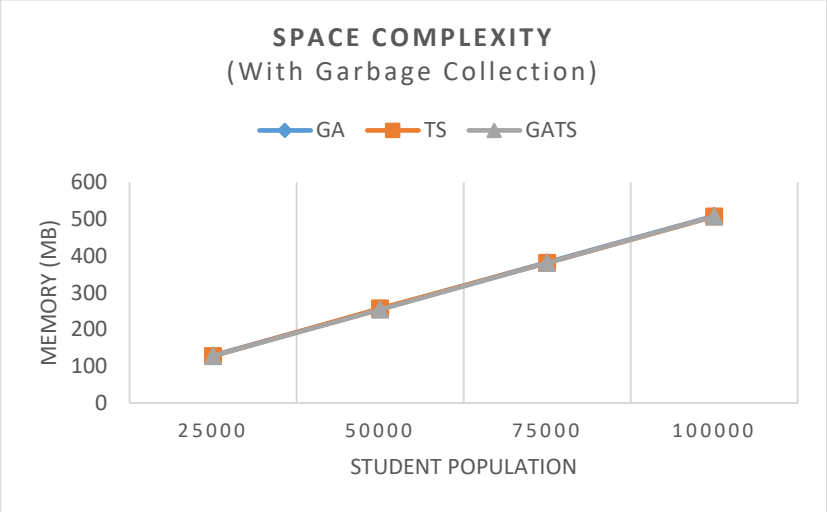

Figure 10: A Comparison of Space Complexity of GA, TS and GATS Algorithms (without GC)

4. Conclusion

In this research, a hybrid Genetic and Tabu Search algorithms, the GATS algorithm, was developed for solving the Examination Timetabling Problem. GATS algorithm incorporated features of both the GA and TS algorithms. The GA, TS and the developed GATS algorithms were tested using data from Bells University of Technology, Ota. The GATS algorithm exhibited superior performance when compared to both the GA and TS algorithms in terms of quality of timetable results returned and time required for such returns. Its memory consumption is similar to that of the TS algorithm and its space complexity is of order $O(n)$. The result showed that hybridization of two or more search algorithms, when properly done can benefit from synergy and outperform the individual component.

Nomenclature

GA	Genetic Algorithm
TS	Tabu Search
GATS	Hybrid GA-TS algorithm
1st OCC	First Order Conflict Counts
2nd, 3rd, .. OCC	Second, third, … Order Conflict Count
HC	Hard Constraint
SC	Soft Constrating
f_o	Objective Function
f_p	Penalty Function
w_h	Penalty weight for constraint h
$\mathbb{N}_0$	The set of Natural numbers (counting from zero)

References

[1] Schaerf, A., *A survey of automated timetabling.* . Artificial Intelligence Review., 1999. **13**(2): p. 87-127.
[2] Ross, P., E. Hart, and D. Corne, eds. *Some observations about GA-based exam timetabling.* LNCS 1408, Practice and Theory of Automated Timetabling II: Selected Papers from the 2nd International Conference. II : Second International Conference, PATAT 1997, ed. E.K. Burke and M.W. Carter. Vol. 1408. 1997, Springer-Verlag: Toronto, Canada. 115-129.
[3] Carter, M.W., *A survey of practical applications of examination timetabling algorithms.* Operations Research, 1986. **34.**(2.): p. 193-202.
[4] Burke, E.K., D.G. Elliman, and R. Weare, *A University Timetabling System based n Graph colouring and Constraint Manipulation.* . Journal of Research on Computing in Education., 1993: p. 26.
[5] Qu, R., et al., *A Survey of Search Methodologies and Automated System Development for Examination Timetabling.* Journal of Scheduling, 2009. **12**(1): p. 55 - 89.
[6] Garey, M.R. and D.S. Johnson, *Computers and intractability. A guide to the theory of NP-completeness. A Series of Books in the Mathematical Sciences.* 1979, San Francisco, Calif: WH Freeman and Company.
[7] Cooper, T.B. and J.H. Kingston, *The complexity of timetable construction problems. In (Burke & Ross, 1996).* 1995: p. 283-295.
[8] Burke, E.K., et al., *Examination timetabling in British universities: A survey.*, in *Practice and Theory of Automated Timetabling: Selected Papers from the 1st International Conference. LNCS 1153.* , B. E.K. and R. P., Editors. 1996, Springer-Verlag: Berlin, Heidelberg. p. 76-90.
[9] Boizumault, P., Y. Delon, and L. Peridy, *Constraint logic programming for examination timetabling.* The Journal Of Logic Programming, 1996.
[10] Carter, M.W. and G. Laporte, *Recent developments in practical examination timetabling. In: E.K. Burke and P. Ross (eds) (1996)* 1996: p. 3-21.
[11] Burke, E.K. and S. Petrovic, *Recent Research Directions in Automated Timetabling.* European Journal of Operational Research - EJOR, 2002. **140**(2): p. 266-280.
[12] Zahra Beheshti, Z., S. Mariyam, and H. Shamsuddin, *A Review of Population-based Meta-Heuristic Algorithms* International Journal of Advance. Soft Comput. Applications, 2013. **5**(1).
[13] Oyeleye, C.A., *Development of a Hybrid Model for Solving Examination Timetabling Problem*, in *Department of Computer Science and Engineering*2011, Department of Computer Science and Engineering, Ladoke Akintola University of Technology, Ogbomoso, Nigeria: Ogbomoso.
[14] Zdansky, M. and J. Pozivil. *Combination Genetic/Tabu Search Algorithm for Hybrid Flow Shops Optimization*. in *ALGORITMY 2002 Conference on Scientic Computing*. 2002.
[15] Nayak, S.K., S.K. Padhy, and S.P. Panigrahi, *A novel algorithm for dynamic task scheduling.* Future Generation Computer Systems, 2012.
[16] Gendreau, M. and J. Potvin, *Handbook of Metaheuristics*. 2 ed. International Series in Operations Research & Management Science, ed. F.S. Hillier. Vol. 146. 2010, New York Dordrecht Heidelberg London: Springer
[17] Raidl, G.R., J. Puchinger, and C. Blum, *Metaheuristic Hybrids*, in *Handbook of Metaheuristics*, M. Gendreau and J. Potvin, Editors. 2010, Springer New York Dordrecht Heidelberg London. p. 469-496.
[18] Hassani, A. and J. Treijs, *An Overview of Standard and Parallel Genetic Algorithms*, in *IDT Workshop on Interesting Results in Computer Science and Engineering (IRCSE '09)*2009: Mälardalen University, Sweden
[19] Corne, D., P. Ross, and H. Fang, *Evolutionary timetabling: Practice, prospects and work in progress.* , in *Proceedings of UK Planning and Scheduling SIG Workshop.*, P. Prosser, Editor. 1994.

[20] Oyeleye, C.A., et al., *Performance evaluation of simulated annealing and genetic algorithm in solving examination timetabling problem.* Scientific Research and Essays, 2012. **7**(17): p. 1727-1733.
[21] Hou, J.H., J.M. Wang, and X.J. Xu, *A Comparison of Three Heuristic Algorithms for Molecular Docking,.* Chinese Chemical Letters Vol., 1999. **10**(7): p. 615-618.
[22] Glover, F., *Tabu Search: A Tutorial.* Interfaces, 1990. **20**: p. 74-94.
[23] Burke, E.K., G. Kendall, and E. Soubeiga, *A Tabu-Search Hyperheuristics for Timetabling and Rostering.* Journal of Heuristics, 2003. **9**: p. 451-470.
[24] Kendall, G. and N.M. Hussin, *Tabu Search Hyper-heuristic Approach to the Examination Timetabling Problem at University of Technology MARA]*, in *PATAT 2004, Selected Papers from the 5th International Conference. Lecture Notes in Computer (LNCS) 3616,* , E.K. Burke and M. Trick, Editors. 2005, Springer-Verlag Berlin Heidelberg (2005). p. 199-218.
[25] Gendreau, M. and J. Potvin, *Tabu Search*, in *Handbook of Metaheuristics.*, M. Gendreau and J. Potvin, Editors. 2010, Springer Science+Business Media, LLC 2010: International Series in Operations Research and Management Sciences 146, New York Dordrecht Heidelberg London.
[26] Glover, F., *Tabu Search - Part I.* ORSA Journal on Computing 1989. **1**(3).
[27] Gendreau, M. *An Introduction to Tabu Search*. 2002.
[28] Malek, M., et al., *A Hybrid Algorithm Technique*, 1989, University of Texas: Austin, TX, USA ©1989.
[29] Almeida, F., et al., eds. *Proceedings of HM 2006 – Third International Workshop on Hybrid Metaheuristics, Lecture Notes in Computer Science*. Vol. 4030. 2006, Springer: Berlin
[30] Bartz-Beielstein, T., et al., eds. *Proceedings of HM 2007 – Fourth International Workshop on Hybrid Metaheuristics, Lecture Notes in Computer Science*. Vol. 4771. 2007, Springer: Berlin.
[31] Blesa Aguilera, M.J., et al., eds. *Proceedings of HM 2005 – Second International Workshop on Hybrid Metaheuristics. Lecture Notes in Computer Science.* Vol. 3636. 2005, Springer: Berlin.
[32] Blum, C., A. Roli, and M. Sampels, eds. *Proceedings of HM 2004 – First International Workshop on Hybrid Metaheuristics* 2004: Valencia, Spain.
[33] Maniezzo, V., P. Hansen, and S. Voss, eds. *Proceedings of Matheuristics 2006: First International Workshop on Mathematical Contributions to Metaheuristics*. 2006: Bertinoro, Italy.
[34] Hansen, P., et al., eds. *Proceedings of Matheuristics 2008: Second International Workshop on Model Based Metaheuristics*. 2008: Bertinoro, Italy.
[35] Perron, L. and M.A. Trick, eds. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems – CPAIOR 2008, 5th International Conference: Lecture Notes in Computer Science.* Lecture Notes in Computer Science. Vol. 5015. 2008, Springer: Berlin (2008).
[36] McCollum, B. *University Timetabling: Bridging the Gap between Research and Practice*. in *The Proceedings of the 6 th International Conference on the Practice and Theory of Automated Timetabling (PATAT)*. 2006.
[37] White, G.M. and B.S. Xie, *Examination Timetables and Tabu Search with Longer-Term Memory*, in *PATAT 2000, Lecture Notes in Computer Sciences (LNCS) 2079*, E. Burke and W. Erben, Editors. 2001 Springer-Verlag Berlin Heidelberg. p. 85-103.
[38] Burke, E.K., D.G. Elliman, and R. Weare, *The automated timetabling of university exams using a hybrid genetic algorithm.*, in *AISB Workshop on Evolutionary Computing. 1995*1995, Society for the Study of Artificial Intelligence and Simulation of Behaviour (SSAISB), 1995: University of Leeds, UK, 3-7 April 1995.

APPENDIX A
Extracted Pages of a Generated Examination Timetables

BELLS UNIVERSITY OF TECHNOLOGY, OTA
EXAMINATION TIMETABLE

Days	Courses/No of Students	Venues(Count)	Invigilators' IDs
Mon	MEE207(157)-HD(157)	MPH(240)	38; 39; 50; 64; 65
9.00am-12.00pm	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	