NIPES

Journal of Science and Technology Research

Journal homepage: www.nipesjournals.org.ng

Tabu Search with Explicit Adaptive Guiding Heuristic for the Examination Timetabling Problem

Ezike J.O.J *

Dept. of Computer Science and Information Technology, Bells University of Technology, Ota, Nigeria

* Corresponding Author: josephezike@gmail.com; +2348035057550

Article Info

Received 05 June 2020
Revised 22 June 2020
Accepted 25 June 2020
Available online 31 August 2020

***Keywords*:**
Tabu Search, Examination Timetabling, Adaptive Guiding, Heuristic

https://doi.org/10.37933/nipes/2.3.2020.7

https://nipesjournals.org.ng

Abstract

Tabu Search algorithm (TS) is one of the leading methods for solving the Examination Timetabling Problem (ETP). The challenge with TS which uses only one solution in searching is that it can easily miss some areas of the search space and chose the best solution based on the current search step and position without taking the future steps into consideration. To address these problems, an explicit and adaptive guiding heuristic was proposed for the standard TS algorithm with the goal of enhancing its performance. The standard TS and the TS with the explicit adaptive guiding mechanism, the TS_G algorithm were implement using the Java programing language (version 8.92) with the MySQL database and experimented with data from Bells University of Technology, Ota. Performance of the two algorithms were evaluated using first Order Conflict Counts (OCC), Second OCC and third OCC for students and invigilators respectively, in addition to memory consumption. The TS and TS_G yielded an average first OCC 0.0 and 0.0 students and 0.0 and 0.0 for invigilators. Similarly, TS and TS_G yielded average second OCC of 18.8 and 10.5 for students and, 0.0 and 0.0 for invigilators respectively. Furthermore, TS and TS_G yielded average third OCC of 33375.0 and 9922.1 for students and 0.0 and 0.0 for invigilators, respectively. The TS_G algorithm outperformed the standard TS algorithm with lower conflicts violations. Both algorithms however recorded similar memory consumption.

1. Introduction

The Examination Timetabling Problem (ETP) is the problem of assigning examinations, candidates and invigilators to time periods and venues in a given university while satisfying relevant constraints. A number of researchers such as [1] have noted that examination timetabling problem (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. Tabu Search is one of the leading method for solving difficult problems and finding good solutions to large combinatorial problems encountered in many practical settings [2], such as the ETP; this has made TS popular among researchers. However, TS which uses only one solution, suffers from the following challenges: it can miss promising regions or search space [3]; it choses best solution based on current step and position without taking the future steps into consideration [4].

This paper aims to address the aforementioned challenges of TS by proposing an explicit adaptive guiding heuristic in addition to the native TS search guiding techniques, so as to enhance the TS performance in solving the ETP.

1.1. Related Works

1.1.1. The Examination Timetabling Problem

Operations researchers have "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."[5]. The complexities and the challenges posed by the ETP 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 [1, 6]. 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. Determining the quality of timetables generated is normally done by using an objective function, formulated using the hard and soft constraints stipulated by the institution and implemented in the timetable generating algorithm.

1.1.2. Tabu Search (TS)

TS is a local search-based "higher level" heuristic procedure for solving optimization problems. When implemented as a hyper-heuristic, TS can be used to guide other methods (or their components) to escape the trap of local optimality [7] or for developing a more general and non-problem specific automated scheduling methods as in [8] and [9]. 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. TS has been noted to be amongst the best and effective methods of tackling difficult problems and finding quality solutions in many practical setting where large combinatorial problems are encountered [2].

TS uses flexible memory structures (in form of lists), classified as short term memory, intermediate term and long term memories of varying time spans for intensifying and diversifying the search, thereby reinforcing attributes historically found good and driving the search into new region. This allows the search space to be exploited more thoroughly[10]. The adaptive memory structures used by TS to keep track of the information related to the search process give Tabu Search algorithm its intelligence [11, 12]. TS also uses conditions for strategically constraining and freeing the search process (embodied in tabu restrictions and aspiration criteria). Parameters, such as the 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.

TS uses a Neighborhood operator, N(S) to carry out a local search on the region of search space being investigated, via the generation of a candidate list of solutions on which atomic moves are applied. An application of TS is generally characterized by the following factors: (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). A template for TS is given in [13].

In [11] different findings were reported from experimentation with short and long term tabu list sizes. TS has been shown to be more effective when both short and long term memories were in use (increasing quality to about 34%) than the short term memory only [14]. In addition, the length of the long term memory need to be carefully chosen as it contributes to the list effectiveness. In [15]

it was shown that using a relaxation strategy to the tabu list, by emptying the tabu list when no improvement was recorded, produced a better solution when compared with the approach with only short term memory alone. Conventionally, fixed length tabu lists are usually implemented, but in [10], it was shown that fixed length list cannot always prevent cycling as such variable length list may also be considered. In [16], a fuzzy inference rule based system (FIRBS), based on frequency and inactivity, was developed and used to manage the tabu list length for examination timetabling, resulting in improved performance.

2. Methodology

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 [17, 18]. These constraints are grouped as hard and soft. These constraints were attached penalty cost in view of their severity if violated; very high penalty cost is attached to constraints with very severe implication when violated. This practice has been recommended in literature and found to be true empirically.

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 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 and HC2 violations (see Table 1) are referred to as first-order conflicts, a situation where two or more examinations were scheduled at the same time for at least a student (or invigilation for invigilator(s))[9, 14]. In this paper, second order conflicts refer to the soft constraint of a student having to sit for three consecutive examinations, that is, back-to-back. This refers to violation of SC3 and SC6 in Table 1. Third order conflict refers to cases of two examinations (or invigilation), back-to-back (SC2 or SC5), while fourth-order conflict refers to sitting for two examinations (or invigilation), non-back-to-back, i.e., separated by a period (SC1 and SC4).

2.1. Mathematical Formulation of the ETP

From constraints in Table 1, the ETP is formulated as shown in subsequent sections.

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), $d_1, d_2, d_3, \dots, d_d$: 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, $v_1, v_2, v_3, \dots, v_y$

2.1.2. Decision Variables

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

e_mp_n: is the instance of an examination e_m scheduled in period p_n. e_mp_n= 1 if schedule or 0 otherwise.

s_je_m: is the instance that student s_j enrolled for examination e_m. s_je_m = 1 if student enrolled or 0 otherwise.

$s_je_mp_n$: is the instance that student s_j is to sit for examination e_m scheduled for period p_n. $s_je_mp_n$ = 1 if student is scheduled or 0 otherwise.

$l_gv_yp_n$: is the instance of a teacher l_g scheduled to be in venue v_y at period p_n.

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_n) denote the number of hours in period p_n.

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,

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

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

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

$R_{si} = c_1, c_2, c_3, \dots, c_\lambda$ *where* $j \in \mathbb{N}_1 \wedge j \geq 1$ *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:

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

2.1.5. 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:

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

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 $|Se_m| = n_s(e_m)$

2.2. Constraint and Objective Function Modeling

Details of the constrain modeling is given in [19]. The objective function formulated using the constraint is given as follows:

$$f_p = w_h \sum_{i=1}^{i=4} HC_i + \sum_{j,k=1}^{j,k=3} w_j (SC_j + SC_k) \quad (2)$$

With HC_i, SC_j, and SC_k being 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 choices 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), subject to the following constraints:

$$\sum_{j=1}^{e} s_i e_j p_k \leq 1 \quad (3)$$

$$\sum_{a=1}^{j} l_g v_a p_n \leq 1 \quad (4)$$

$$\sum_{i=1}^{k} s_i e_m v_y = n_s(e_m) : n_s(e_m) \leq cap(v_y) \quad (5)$$

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

$$\sum_{k=a}^{k=a+1} s_i p_{kd_h} < 2 \quad (7)$$

$$\sum_{k=1}^{k=3} s_i p_{kd_h} < 3 \quad (8)$$

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

$$\sum_{k=a}^{k=a+1} l_g e_m p_{kd_h} < 2 \quad (10)$$

$$\sum_{k=1}^{k=3} l_g e_m p_{kd_h} < 3 \quad (11)$$

2.3. Tabu Search for Examination Timetables with Fixed Structures

Regular institutions of higher learning usually have examination periods lasting for a certain number of weeks, with examinations taking place in a three-periods-per-day schedules or a two-periods-per-day schedule with each period lasting for a maximum of three hours. In such cases, examinations duration would not exceed three hours but could be less. Figure 1 illustrate this scenario.

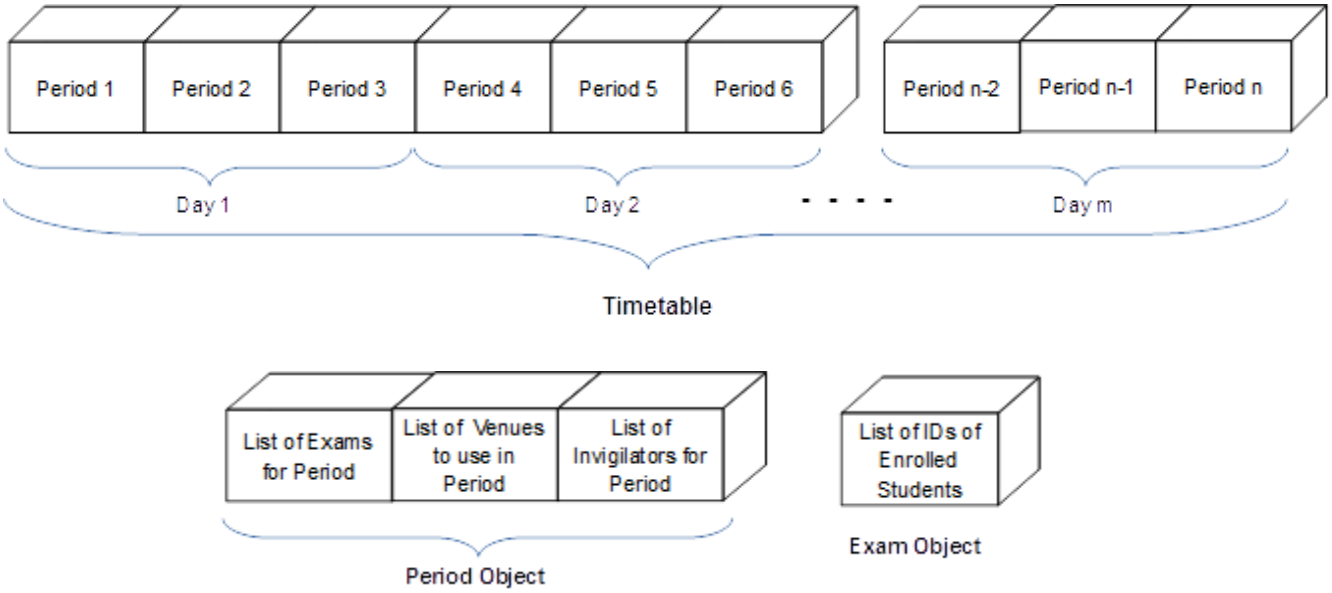

Figure 1: The Timetable structure showing the different components

This research assumes a timetable with a three-periods-per-day examination schedules with each period lasting for a maximum of three hours. This assumption allows the examination timetabling search space to be segmented or divided to the numbers of days of the examination duration; daily penalty cost can then be computed. The total penalty cost of the timetable is then the sum of penalty cost for each of the timetable days as illustrated in Figure 2.

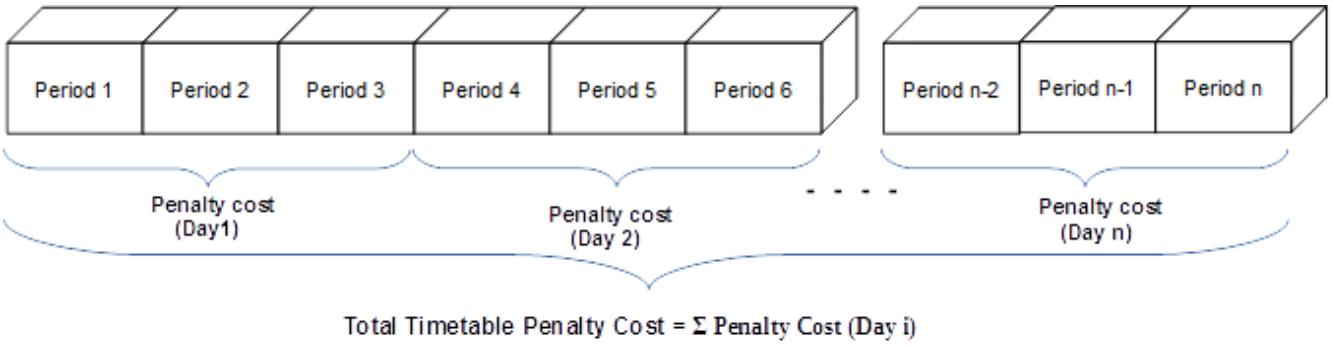

Figure 2: Illustrates this Timetable Penalty Cost Computation

Days with higher penalty costs computation then represent regions of more profitable search spaces. It then become possible to explicitly direct the TS algorithm to these days (areas of the search spaces) when diversifying, thereby explicitly guiding the TS algorithm; incorporating this explicit and adaptive guiding heuristic to the standard TS algorithm resulted in the TS with explicit Guided (TS_G) algorithm developed in this research.

2.4. Description and Implementation of the TS and TS_G Algorithms

The description of the implemented TS and the TS_G algorithms follow in the subsequent sections.

2.4.1. Description of the TS Algorithm

The description of the implemented Tabu Search algorithm is given in Table 2.

Table 2: The implemented standard Tabu Search algorithm.

	Algorithm Tabu Search
	Input: t_0, StudSemRegL // List of each student exam registration of in semester
	Output: t_{best}
1	$t \leftarrow t_0$
2	$t_{best} \leftarrow t_0$
3	$TL \leftarrow \emptyset$
4	while (not stopping Condition)
5	$CL \leftarrow \emptyset$ // CL, the list for all candidate solution
6	$CL \leftarrow N(t)$ // N, the neighborhood operator, generates all candidate solution of t
7	t ←applyBest(CL)
8	if (fitness(t) > fitness(t_{best}))
9	$t_{best} \leftarrow t$
10	endif
11	$TL \leftarrow t$
12	if((size(TL) > maxSize(TL))
13	removeFirst(TL)
14	endif
15	if(DiversificationCcondition)
16	diversify()
17	endif
18	endwhile
19	return t_{best}

2.4.1.1. 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 produces 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.

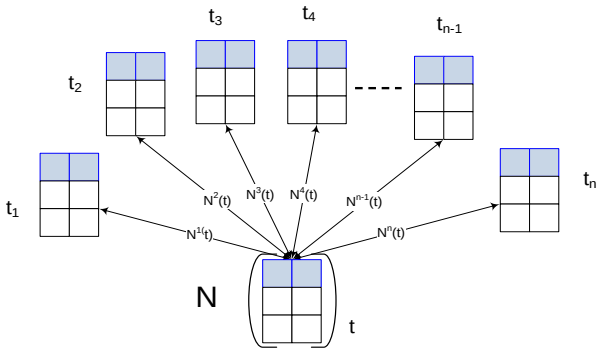

Figure. 3: The operation of the Neighborhood operator N on timetable t to produce its neighbors t_1 to t_n.

2.4.2. Description of the TS _G Algorithm

The description of the implemented TS_G algorithm is identical to that of the TS algorithm given in Figure 1; the only difference in line 15 to line 17. The modification is captured in Table 3.

Table 3: The Implemented TS-G Search algorithm.

```
15  if(DiversificationCcondition)
16  diversify()
17          else if(ExplicitGuidingHeuristicDiversificationCondition)
18                  adaptiveHeuristicDiversify()
19          endif
20  endwhile
21  return t_best
```

2.4.2.1. The Explicit Adaptive Guiding Heuristic of the TS_G Algorithm

The adaptation in the explicit guiding heuristic was accomplished by using the following probability function:

$$P = 1000/exp\left(\frac{1}{0.01 * (SC3\ violation\ counts)}\right) \quad (12)$$

Equation 3.12 was curled from the Boltzmann-Gibbs distribution used in Simulated Annealing [20]:

$$P = \exp\left(\frac{-\Delta E}{k_b T}\right) \quad (13)$$

Where k_b the Boltzmann constant and T is the current annealing temperature. The value of probability function P (Equation 12) is periodically computed and compared with a randomly generated real number in the range [0,1), with the result used in adapting the search. As the SC3 violation reduces to a certain minimum, the search then relies more on the native TS diversification mechanism.

2.5. Data Gathering and Generation

The data used for generating the University Examination Timetables was gathered from Bells University of Technology, Ota, as at the end of 1st Semester 2012/2013 Sessions (see Table 4).

Table 4: Summary of 2012/2013 data gathered from Bells University of Technology, Ota

Total number of students	1896
Total number of Registrations	16866
Total Number of Examinations	443
Total Number of Venues	25 (see Appendix B) total capacity: 1436)
Total number of Invigilating staff	170

2.6. 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.

2.7. Experimental Procedures

The procedure commenced with the loading of the ETP data from the database. Initial timetable solution (without invigilators scheduling) was then generated by random selection of examinations from the examination pool and scheduling into the available venue spaces, while ensuring feasibility. No of periods in generated initial solutions varies from about 21 to 28. For uniformity, initial solutions where normalized to 30 period by adding additional periods and spreading scheduled examinations into these added periods. The 30 period used implies the examination duration will last 10 day, or two weeks of five working days each for a three period in a day examination schedule. The generated initial solutions were then optimized using the TS and the TS_G algorithms, after which the scheduling of invigilators was done.

The TS algorithm was tuned; Tabu list of different sizes (5, 7, 8, 9, 10, 11, 12, 15, 20), were tried. 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 determine the appropriate time to diversify to other search region. Since the two algorithm are identical save for the explicit guiding heuristics of the TS-G algorithm, the tuning was also identical. However, the guiding heuristics was itself turned for effective performance. The TS and TS-G algorithms where then run 10 times each for a duration of 5400s (i.e. 1 hour 30 min). The results of the 10 runs were harvested for analytical purposes. From these results, the best as well as the average performance of each of the two algorithms were determined 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.

3.1. The Effect of Increasing Processing Time on Timetable Quality

Figure 4 showed the improvement of the initial timetable solution as the processing time increases. The best results obtained and the average results of the two algorithms for the 10 experimental runs were illustrated. Both algorithms showed similar performance, however, as can be seen in Figure 5, a magnified vies of Figure 4, the performance of the TS-G algorithm was superior both for the best generated solution and for the average of all 10 solutions. This implies that the TS-G algorithm generated more quality timetables that the TS algorithm. The actual computed penalty costs for the timetables generated are given as follows: TS (best) 916,323, TS-G (best) 335,821; TS (avg) 2,891,433.00, TS-G (avg) 2,045,567.40

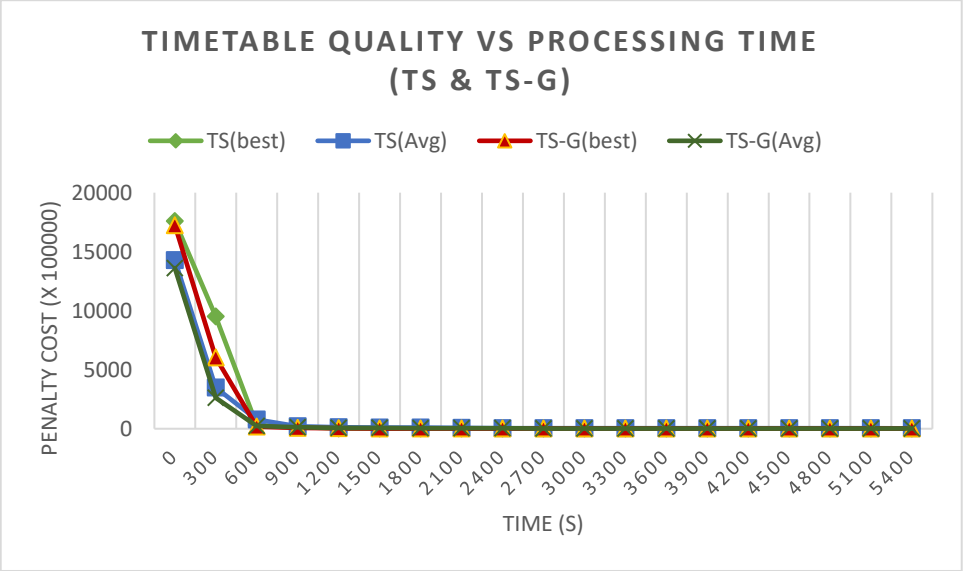

Figure 4: Improvement of the initial timetable solutions with increase in processing time

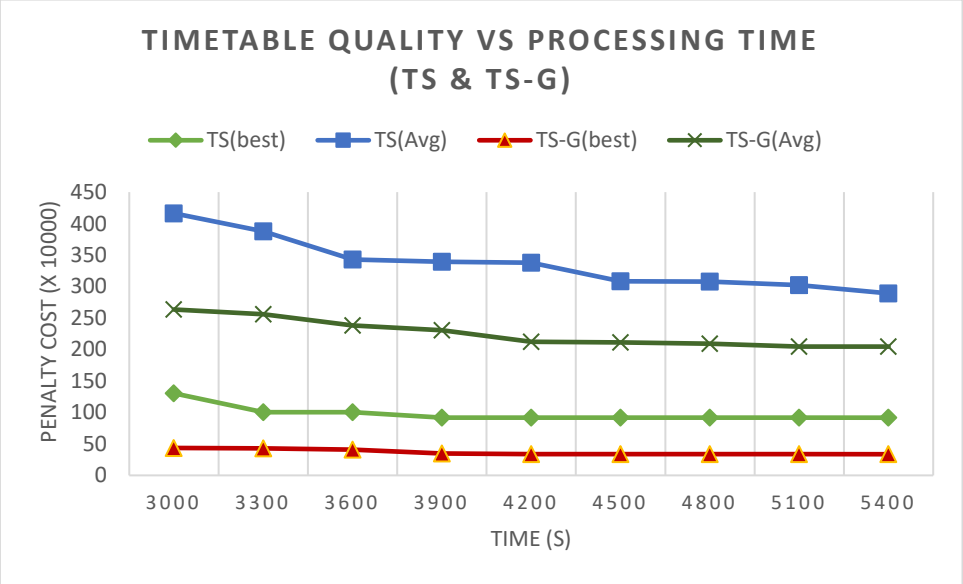

Figure 5: Improvement of the timetable solutions with increase in processing time (a closer look)

3.2. Timetable Quality vs. Number of Students Having 3 Consecutive Examinations

Violating the SC3 constraint (see Table 1), a second order conflict, implies one or more students will have to sit for three examination in succession in a given day. This could be very stressful and would likely lead to such student(s) performing poorly in the examinations. As a result, the SC3 constraint is assigned the most penalty cost next to the hard (HC) constraint. Since no HC constrains are violated, the SC3 constraint constitutes the most to the timetable penalty cost if it exist.

The large penalty value of the SC3 constraint compared to SC2 in Table 1 guide the search algorithms to eliminate cases of the SC3 constraint violation from the timetable in preference to the SC2 violation where possible. From the results obtained, the TS algorithm eliminated the SC3 constraint in 3 of the 10 runs while the TS-G algorithm did same in 4 of the 10 runs. The average of the total SC3 violation is 18.8 and 10.5 with the TS and TS-G algorithms respectively.

3.3. Memory Consumption during Simulation

The TS and the TS_G Algorithms showed similar memory consumption during experimental runs for the best and average results of 10 experimental runs. This is applicable for the two algorithms before and after the java garbage collection utility (GC) is called. Figure 6 illustrates this. The two algorithms are identical in all aspect except for the explicit and adaptive guiding heuristics incorporated in TS_G.

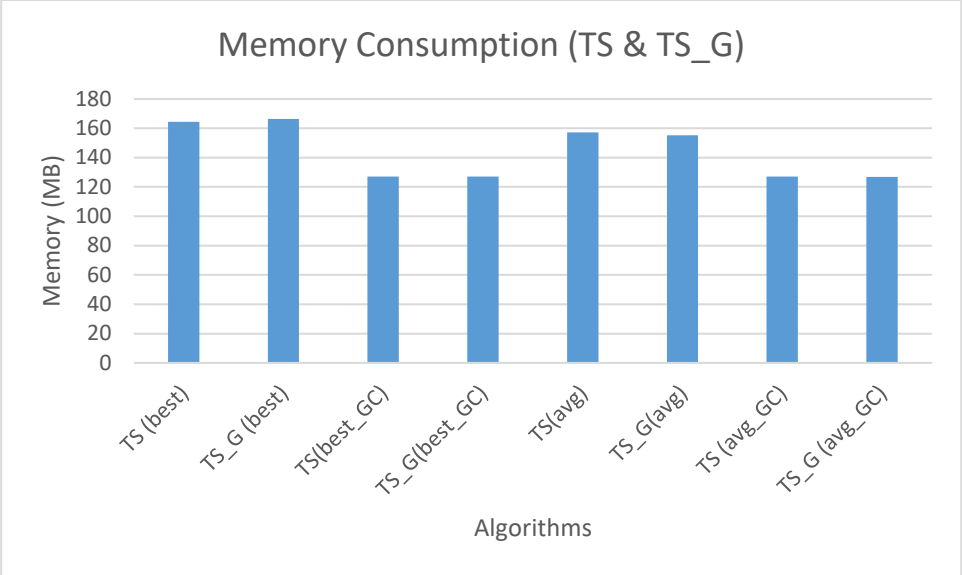

Figure 6: Memory consumption for the TS and TS_G Algorithms

3.4. Result Consideration in view of Two Related Tabu Search Applications

It is noteworthy that the developed TSA with explicit guiding heuristic was applied as a metaheuristic to solve the ETP using dataset from Bells University of Technology. In [8] and [9], Tabu Search was applied as a hyperheuristic that chooses among lower level heuristics in solving different problem instances at a more general level. The concept of explicit guidance in this paper, which relies on the knowledge of the penalty incurred for each day in the timetable, however, can be related to the ranking of the competing lower heuristics used in [8] and [9]. Though the timetabling dataset used in [9] was not experimented with, it is expected that the developed TS algorithm with explicit guidance will compete favorable when applied.

4. Conclusion

This paper has shown for ETP with known examination schedules, the TS can be explicitly and adaptively guided to search regions that are more profitable, thereby reducing the chances of missing promising search spaces or regions. This is in addition to making some consideration for further steps through adaptive guidance thereby enhancing the performance of the TS algorithm. The modified TS algorithm with the guiding mechanism incorporated resulted in the TS_G algorithm.

After the implementation of the standard TS algorithm and the TS algorithm with the explicit adaptive guiding mechanism, 10 experiments runs conducted for each algorithms and results were harvested. From the results obtained, the TS and TS_G yielded an average first OCC 0.0 and 0.0 students and 0.0 and 0.0 for invigilators. Similarly, TS and TS_G yielded average second OCC of 18.8 and 10.5 for students and, 0.0 and 0.0 for invigilators respectively. Furthermore, TS and TS_G yielded average third OCC of 33375.0 and 9922.1 for students and 0.0 and 0.0 for invigilators, respectively. From the foregoing, the TS_G algorithm outperformed the standard TS algorithm with lower second OC conflicts violations, thereby producing timetables with higher quality. Both algorithms however recorded similar memory consumption.

References

[1] 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.

[2] 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.

[3] Zdansky, M. and J. Pozivil. *Combination Genetic/Tabu Search Algorithm for Hybrid Flow Shops Optimization*. in *ALGORITMY 2002 Conference on Scientic Computing*. 2002.

[4] Nayak, S.K., S.K. Padhy, and P. S.P., *A novel algorithm for dynamic task scheduling.* . Future Generation Computer Systems., 2012. **28**(5): p. 709-717.

[5] Boizumault, P., Y. Delon, and L. Peridy, *Constraint logic programming for examination timetabling.* The Journal Of Logic Programming, 1996.

[6] Carter, M.W., G. Laporte, and J.W. Chinneck, *A general examination scheduling system.* Interfaces, 1994. **24**: p. 109-120.

[7] Glover, F., *Tabu Search: A Tutorial.* Interfaces, 1990. **20**: p. 74-94.

[8] Burke, E.K., G. Kendall, and E. Soubeiga, *A Tabu-Search Hyperheuristics for Timetabling and Rostering.* Journal of Heuristics, 2003. **9**: p. 451-470.

[9] 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.

[10] Glover, F., *Tabu Search - Part I.* ORSA Journal on Computing 1989. **1**(3).

[11] Khader, A.T. and A.S. See, *Improving Exam Timetabling Solution Using Tabu Search.* Journal of Digital Information Management (JDIM), 2005. **3**(4): p. 250-253 . .

[12] Abdul Rahman, S., *Search methodologies for examination timetabling*, 2012, University of Nottingham.

[13] Gendreau, M. *An Introduction to Tabu Search*. 2002.

[14] 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.

[15] White, G.M., B.S. Xie, and S. Zonjic, *Using tabu search with longer-term memory and relaxation to create examination timetables.* European Journal of Operational Research, 2004. **153**(1): p. 80-91.

[16] Pais, T.C. and P. Amaral, *Managing the tabu list length using a fuzzy inference system: an application to exams timetabling.* Annals of Operations Research 2012. **194**(1): p. 341-363.

[17] 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.

[18] 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.

[19] Ezike, J.O.J., et al., *Generating Feasible and Optimized Timetables using Tabu Search: The Case of Bells University of Technology, Ota.* Computing, Information Systems & Development Informatics Journal. , 2018. Vol 9,(1): p. 84-104.

[20] Dall'igna Junior, A., et al. *Performance and parameterization of the algorithm Simplified Generalized Simulated Annealing*. Genetc and Molecular Biology [online], 2004. **27**, 616-622 DOI: Https://doi.org/10.1590/S1415-47572004000400024.

APPENDIX A

BELLS UNIVERSITY OF TECHNOLOGY, OTA
EXAMINATION TIMETABLE

Days	Courses/No of Students	Venues(Count)	Invigilators' IDs
Mon	GES101(589)-MPH(240);HallD(230);B11(56);B10(56);BioCLab(7)	MPH(240)	221; 185; 121; 104; 103
9.00am-12.00pm	MEE203(172)-B9(56);Rm6(56);Rm5(56);BioCLab(4)	HallD(230)	578; 159; 555; 581; 582
	ITP401(71)-Rm1(56);BioLab3(15)	B11(56)	274; 135
	HRM303(19)-BioLab1(19)	B10(56)	302; 200
	MEE403(15)-ChemLab2(15)	B9(56)	296; 272
	BIO203(15)-ChemLab2(15)	Rm6(56)	171; 276
	GES313(1)-BioLab3(1)	Rm5(56)	99; 586
		Rm1(56)	89; 236
		ChemLab2(30)	38
		BioLab1(19)	254
		BioLab3(16)	253
		BioCLab(11)	178
Mon	ACC303(55)-Rm1(55)	HallD(133)	39; 329; 332
12:30pm-3.00pm	AMS427(1)-Rm1(1)	Rm1(56)	381; 351
	BDT309(4)-BioCLab(4)	Rm5(56)	365; 238
	ARC405(13)-BioCLab(13)	B6(40)	362
	MKT303(6)-BioLab3(6)	PhyLab1(30)	123
	NUD311(2)-BioCLab(2)	BioLab1(30)	240
	TML501(1)-BioCLab(1)	BioLab2(30)	508
	FSB505(2)-BioLab3(2)	PhyLab2(30)	552
	ITP303(51)-Rm5(51)	ChemLab2(30)	506
	PMT307(4)-Rm5(4)	ChemLab1(30)	473
	PHY309(5)-BioLab3(5)	BioCLab(20)	505
	FDT201(5)-BioLab3(5)	BioLab3(20)	551
	ECO303(39)-B6(39)	B5(14)	550
	BDT413(2)-BioLab3(2)		
	BTE403(2)-BioLab1(2)		
	MIC403(10)-BioLab1(10)		
	MIC303(14)-BioLab1(14)		

	BME407(1)-Rm5(1)			
	ARC305(25)-ChemLab2(25)			
	BME305(6)-BioLab2(6)			
	EEE413(28)-PhyLab1(28)			
	BUS101(133)-HallD(133)			
	URP409(2)-PhyLab1(2)			
	BIC407(10)-BioLab2(10)			
	CHM315(8)-BioLab2(8)			
	FDT409(7)-PhyLab2(7)			
	QTS403(1)-B6(1)			
	URP315(2)-BioLab1(2)			
	EST307(8)-PhyLab2(8)			
	TML401(1)-BioLab1(1)			
	ITP501(13)-PhyLab2(13)			
	BTE501(1)-BioLab1(1)			
	HRM403(18)-ChemLab1(18)			
	SGF201(7)-ChemLab1(7)			
	MCT313(6)-BioLab2(6)			
	SGF305(1)-PhyLab2(1)			
	BIC303(10)-B5(10)			
	QTS305(1)-PhyLab2(1)			
	FDT309(4)-ChemLab2(4)			
	PMT501(2)-ChemLab1(2)			
	FNB401(4)-B5(4)			
	PHY203(1)-ChemLab2(1)			
	FNB305(1)-ChemLab1(1)			
	TCE403(2)-ChemLab1(2)			
Mon 4.00pm-07.00pm	MEE201(180)-HallD(180)	HallD(180)	553; 530; 589; 590	
	ECO201(180)-MPH(180)	MPH(180)	516; 587; 557; 414	
	MCT407(2)-BioCLab(2)	Rm1(53)	485; 481	
	PHY105(17)-BioCLab(17)	BioCLab(20)	580	
	ARC207(53)-Rm1(53)	BioLab3(4)	486	
	EST205(4)-BioLab3(4)			
	EEE415(1)-BioCLab(1)			
Tue 9.00am-12.00pm	EEE409(30)-BioLab1(30)	HallD(230)	482; 417; 420; 423; 429	
	PHY111(145)-HallD(145)	MPH(78)	440; 452	
	CHM411(3)-BioCLab(3)	Rm1(56)	453; 454	
	PMT505(2)-BioCLab(2)	Rm4(48)	455	

BTE301(9)-BioCLab(9)	B4(48)	456	
BIC411(9)-BioLab3(9)	Rm3(47)	461	
MCT405(1)-BioCLab(1)	B6(40)	568	
PHY303(4)-BioCLab(4)	BioLab1(30)	487	
BIO205(12)-ChemLab2(12)	ChemLab2(30)	510	
URP305(3)-BioLab3(3)	BioLab2(30)	538	
FDT405(6)-BioLab3(6)	PhyLab1(30)	513	
BME301(7)-ChemLab2(7)	PhyLab2(25)	514	
ITP503(14)-BioLab2(14)	BioCLab(20)	515	
PMT405(10)-ChemLab2(10)	BioLab3(20)	533	
ECO307(41)-B4(41)			
ACC403(41)-Rm4(41)			
TML505(1)-BioCLab(1)			
URP203(3)-B4(3)			
SGF301(1)-ChemLab2(1)			
EST303(8)-BioLab2(8)			
PHY409(7)-Rm4(7)			
URP415(2)-BioLab3(2)			
ARC203(56)-Rm1(56)			
BDT409(2)-B4(2)			
BME409(1)-B4(1)			
HRM305(20)-PhyLab1(20)			
BDT301(4)-BioLab2(4)			
CHM309(8)-PhyLab1(8)			
AMS425(1)-B4(1)			
CSC307(59)-HallD(59)			
EST203(4)-BioLab2(4)			
EST407(2)-PhyLab1(2)			
BTE505(1)-HallD(1)			
ARC403(10)-HallD(10)			
EEE309(78)-MPH(78)			
FSB507(2)-HallD(2)			
BIC305(10)-HallD(10)			
TCE405(1)-HallD(1)			
MEE305(34)-B6(34)			
BDT203(2)-HallD(2)			
CEN401(12)-PhyLab2(12)			
CSC509(41)-Rm3(41)			
NUD303(3)-B6(3)			
MIC307(9)-PhyLab2(9)			

	BTE405(2)-B6(2)			
	SGF207(6)-Rm3(6)			
	FNB405(4)-PhyLab2(4)			
	QTS401(1)-B6(1)			

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