Available online at www.scientiaresearchlibrary.com

Scientia Research Library

ISSN 2348-0424
USA CODEN: JETRB4

Journal of Engineering And Technology Research, 2017, 5(5):1-14
(*http://www.scientiaresearchlibrary.com/arhcive.php*)

SOLVING UNIVERSITY TIMETABLING PROBLEMS BASED ON A MULTI-CONSTRAINT ENVIRONMENT

FERNANDO S. VIRAY JR.
Pangasinan State University
Correspondance email: skyfher@gmail.com

ABSTRACT

The key to finding an optimum solution for a gargantuan problem such as the University Timetabling Problem (UTP) which is considered an NP-hard problem is to implement a "divide and conquer" mechanism. In this research, the sub-problems of UTP are considered and input data, factors, and parameters were classified and organized to model several stages or phases of the solution process. Meta-heuristic approaches were implemented in finding an overall feasible solution to the UTP by initiating a specific AI-based local search and optimization algorithm with little human intervention in every phase of the solution process. Algorithms to be utilized as a part of the meta-heuristics include Tabu Search, Greedy Algorithm, Integer Linear Programming, and Bi-Partite Graph Approach. Simulation results revealed that the proposed multi-stage solution process model, by incorporating multi-constraint inputs, is a promising model when paired with any of the subject algorithms as it significantly aids in finding an optimum solution faster in terms of elapsed time and computation resources.

Keywords: Timetabling Problem (TP), University Class Scheduling Problem (UCSP), local search and optimization techniques

---

INTRODUCTION

Considered one of the most tiresome and recurring undertakings performed among academic institutions worldwide, the University Timetabling Problem (UTP), also commonly known as the University Class Scheduling Problem (UCSP)[1], has continued to be a burdensome enrollment preparation be it among small primary schools locally or large universities in the international level.[2] Although there are several variations among input variables under the UTP[3], the basic idea of the problem simply requires[4]: (1) several courses in which each course is a set of subjects; (2) students who belong to a certain course and are required to take all the subjects specified on a student's respective course curriculum; (3) teachers who are assigned based on their specializations; (4) and a pool of rooms categorized either as regular lecture classroom or specialized rooms such as laboratory, music center, gymnasiums, etc., where a specific set of equipment, tools or devices are placed to aid in teaching.

The main objective of the UTP is to assign a class of students belonging to the same course subjects

to teachers and classrooms without conflict while considering university policies and maximizing existing facilities.[3] In this regard, rules that should be strictly followed are referred to as hard constraints while other schedule considerations that can be set aside to achieve a conflict-free schedule are called soft constraints.[4] In this context, a feasible solution is easier to find rather than an optimum solution – as feasible solutions require that they should be conflict-free among hard constraints but sometimes do not guarantee full practical implementation while optimum solution calls for the best among feasible solutions[5] which entails higher human logic both in technical solution and practical implementation[6]. This held the timetabling algorithms as part of local search and optimization techniques which are one of the computational foundations of Artificial Intelligence (AI)[1] – techniques that are regarded as a nondeterministic polynomial time (NP) hard problem[5] which means that it is easier to verify in polynomial time that a solution, if one has been found, is, in fact, a solution of the problem rather than find the solution for the problem as there is no known efficient and expressway of solving the problem, and although it can be solved, it does not guarantee optimality.[7]

This research aims to determine the generic stages that will ease in finding a feasible solution to the Timetabling Problem by considering the inputs from each sub-problem involved under the UTP. After determining the problem stages, it seeks to determine the criteria to be set in which human intervention as part of a phase-by-phase heuristic can be provided in as little as possible. Finally, this study aspires to identify which AI-based local search and optimization algorithm is appropriate for the proposed multi-stage solution process.

Related Works

The University Timetabling Problem refers to the problem of finding a conflict-free class schedule in which a student is assigned a set of specific subjects from his respective course and assigning each subject to be taken for the current semester or period into a specific classroom and teacher.[8] UTP originated from the Job-Shop Scheduling Problem (JSP) in which a set of jobs are allocated to limited resources at distinct times – replacing jobs as subjects from course curricula to be taken by students and limited resources as the designated classrooms and teachers at specific timeslots.[9] The solution to the UTP involves local search and optimization processes as it needs to find the best solution among all possible candidate solution sets.[5] The use of machine learning to schedule problems recently as means of heuristics became an emerging technique [1] [10] in which artificial intelligence (AI) learns to find an optimum by mimicking human intelligence and intuition to perform human-level logic, comprehend complex calculations and causality verification.[10]

UTP is most of the time characterized as a problem with rule-based inputs, rules which control and dictate the computational complexity of the problem and are often called constraints which can be classified as either hard or soft constraints.[3] Hard constraints are the rules or factors in which a solution cannot be derived unless all these rules are fully accounted for. On the other hand, soft constraints are optional parameters or preferences which are used to improve the quality of the solution. Under these pretexts, a feasible solution can still be found even without satisfying any of the soft constraints [4] as long as all hard constraints are satisfied and not one of them is violated.[7]

The complexity of the University Timetabling Problem grows as the number of constraints grows larger and each solution is unique to a particular school as the number of students, offered programs, academic resources, and provision of the degree of stakeholder satisfaction vary from each academic institution which gives rise to customized university timetabling software[10]. The difficulty to develop a generalized model and a "one-size-fits-all" timetabling computer program given the regional, ethnic, curricular, and time zone differences factors serve as an inspiration among software developers to establish the backbones and foundations of school timetabling software.[11] Under this endeavor, the generic stages of the solution process can be determined by considering the sub-problems involved under the Timetabling Problem and the human interventions

which serve also as a heuristic.

Sub-Problems of the Timetabling Problem

a. Teacher Assignment

The teacher assignment sub-problem involves assigning and scheduling teachers to specific courses by taking some factors, such as the abilities of the teachers and the number of courses offered. The aim is to arrive at balanced faculty loading for both the part-time and full-time teachers, the latter being prioritized and where loading refers to the total number of credits assigned to each teacher, a credit being equivalent to an hour of teaching (i.e. a 3-credit subject is equivalent to 3 hours of teaching load).[2]

The factors considered under the teacher assignment sub-problem are as follows: (1) Each course section has a minimum and maximum number of teachers to teach; (2) Part-time teachers are required to teach courses that cannot be taught by full-time teachers due to overloading or out of specialization scope; (3) A maximum and minimum number of teachers can be assigned to a particular course depending on the number of course sections; (4) Each teacher should be given ample time for class and lesson preparation so the maximum number of hours per teacher should not exceed or be equal to 8 hours; and (5) Full-time teachers are prioritized to be given faculty loading;

The first four factors are deemed as hard constraints while the fifth is treated as a preference to minimize the total weighted variance among teachers' load.[2]

b. Course Scheduling

The course scheduling sub-problem involves the scheduling of subjects on a specific period, year level, and class sections for a particular course program approved by government agencies to be offered by the school for a particular school year. The main goal is to create a conflict-free initial schedule of subjects as specified by each course curriculum per year level regardless of enrolled students.[2]

The Course Scheduling sub-problem has the following factors and constraints:[2] (1) Each student strictly follows a specific course curriculum; (2) A course curriculum consists of a set of subjects to be taken for a particular semester or year; (3) Each subject has its corresponding credit units equivalent to several hours to be taught in a week (i.e. a 3 unit subject should be provided 3 hours in a week to be taught) and can be divided into hourly length, whole length or half-length; (4) Each class day consists at most of two half days (i.e., morning and afternoon) with a fixed lunchtime in-between. Each half-day has several consecutive periods or time slots corresponding to the length of each unit a subject is divided to be taught in a week; (5) Students are grouped according to their courses and year level which entails the number of subjects the student has finished in following his curriculum; (6) Large courses on which numerous students are projected to enroll have to be divided into course sections; (7) Sections of a course need not be taught by the same teacher, hence, sections of a course may have periods and subjects to be taken in common but being taught by different teachers. (8) The number of teachers as well as the set of course sections are determined in advance based on the approved course offering and previously enrolled students.[3]

c. Class-Teacher Timetabling

The class-teacher timetabling sub-problem is about creating a schedule of teachers and class sections over a set of periods, typically covering five or six weekdays depending upon the institutional policy. The basic constraints that must be satisfied are: i) professors can only be in one class at any given time, and ii) students belonging to a section or class can only attend a teacher's lecture at any given time. This sub-problem also focuses on a) assigning a substitute teacher to a

particular class in the absence of the regularly assigned teacher, and b) merging classes/sections having the same subject to take under the same curriculum.[12]

The class-teacher timetabling sub-problem is different from the teacher assignment problem as this specifically sets the class section in which a specific teacher will handle for a given period. Additionally, this problem is different from the course scheduling problem as it focuses more on opening or closing subject offerings based on enrolled students and available teachers as per curriculum standards.[12]

d. Student Scheduling

The student scheduling sub-problem is primarily involved in creating balanced, almost gap-free, or consecutive subject schedules for a specific period within the week based on the course curriculum the student is under. A balance and ideal student schedule relate to an even number of subject hours per day for the student to be taken in a week while gap-free speaks about subjects being contiguously scheduled (except for lunch break, morning and afternoon breaks, or recess times) as much as possible to maximize the usual 6 to 8 hours whole day class period.[13]

The hard constraints of this sub-problem are the prescribed subjects that a student must take within the semester or year concerning his course curriculum and the master schedule of course offerings with their multiple sections, each with possibly several meeting times, rooms, and instructors.[13]

e. Room Assignment

The room assignment sub-problem covers allocating classes to classrooms which is mathematically similar to the problems of hotel room assignments, shipyard scheduling, job scheduling on machines with varying capabilities, the assigning of airport gates to flights, etc. The constraints of this problem are: (1) lecture subjects should be taught inside regular lecture classrooms; (2) laboratory or elective subjects should be taught inside a specialized facility or laboratory room where tools, devices, and equipment aid in learning; (4) a particular subject is taught in a classroom by a specific teacher during a specified timeslot; and (3) each room, regardless of type, has a maximum seating capacity.

The usual soft constraints of this sub-problem most of the time involve the use of lecture rooms by respective colleges or departments and class sections or blocks can be merged to attend the same subject in a room as long as the seating capacity allows.

Stages of the University Timetabling Solution Process

Several studies have been conducted to solve the Timetabling Problem by careful decomposition of the whole problem into phases or stages iterating through the construction phase and local search phase until a feasible solution is found[14]. This procedure is similar to the GRASP Model or Greedy Randomized Adaptive Search Procedure, a meta-heuristic algorithm for combinatorial optimization[15].

A two-stage decomposition technique has been used for timetabling problems through Integer Linear Programming and has shown significantly better performance than solving the problem using the original Integer Programming algorithm in terms of both found solutions and bounds[16]. The technique is comprised of assigning timeslots (the first stage) and assigning rooms (the second stage) where timeslots and available rooms served as hard constraints. This procedure has shown that input constraints can serve as stages or phases of the timetabling solution process. Another similar multi-stage technique used in solving examination timetabling problems is composed of: Stage 1 – Achieving Feasibility where a feasible solution is constructed by adding to the schedule all the exams sequentially; Stage 2 – Local Search of the neighborhood structure is defined by randomly selecting a pair of periods and moving certain exams from one period to the other; and

Stage 3 – Improvements per Period by using cost scoring on each room and using only those rooms with low-cost score[14].

Human Intervention as Heuristics in Solving the

University Timetabling Problem

Regardless of how advanced AI-based computer systems and scheduling applications we have today, the high degree of logic, comprehension, and prediction set by human intelligence is still unmatched.[6]. Human intervention during stages or phases in finding optimized solutions remains the highest form of heuristic as it is normal for humans to see whether a solution is logical and practical in all of its senses and aspect, an area where computer technologies are still far away[2].

Among the key human abilities where interventions can be introduced to help in finding an optimum solution for University Timetabling problem includes: (1) immediate determination of complex computation and computational resources that would be involve in a solution – with this human ability, human intervention can be introduced by immediately cancelling the ongoing optimization or local search task; (2) spontaneously comprehending the root cause of the problem, any impending effect, and the candidate solution to apply – humans can easily discern whether an algorithm or set or process would result to a better solution, pinpoint the strength and weakness of the procedure, and introduce alternatives or patches to resolve issues; and (3) high degree of logical reasoning that can gauge whether a solution is practical to implement or not – this characteristic of human capability sets a huge gap between humans and computers as humans can make immediate assessment whether a solution is optimum or not by logic and practicality[17]. These three human abilities can be used as a heuristic in the form of human intervention to find an optimum solution to the timetabling problem[18].

Commonly Used AI-based Timetabling Algorithms

A lot of works and heuristic algorithms have been proposed to solve the Timetabling Problem which is based on local search techniques.[5] This section provides different analytical timetabling algorithms based on local search optimizations.

a. Tabu Search

The Tabu Search algorithm was created in 1986 by Fred W. Glover and was formalized three years thereafter. It is a metaheuristic search method that employs mathematical local search optimizations schemes.[19] In Tabu Search, local searches mean taking a candidate solution by checking its immediate solutions (called neighbors) from a solution set that is similar but has very minute details hoping to find an enhanced solution by easing its basic rules (aspiration criteria). However, when there are many similar solutions, called suboptimal regions, that are a fit to the possible solution being compared, this method becomes stuck as each candidate solution matches the relaxed criteria it is looking for. If no possible similar solution among neighbors is present and that neighbors do not satisfy the strict local minimum, the neighborhood is marked and remembered and prohibitions are introduced (appending its address and values in the Tabu List), thus the term “tabu”, to prevent searching again the visited and marked neighborhood.[10]

The implementation of the Tabu Search algorithm to solve the Timetabling problem in the hope to ensure that hard constraints are satisfied and obtaining more feasible solutions through swapping of lectures to avoid conflict among lecture instructors has been noted that although the Tabu Search algorithm is more time consuming as complexity increases with a directly proportional increase in constraints, it is an efficient algorithm in terms of using limited computing resources and has effectively avoided loops in cycling back to previously visited solutions through memory structure called Tabu List.[20]

b. Greedy Algorithm

A greedy algorithm is an algorithmic model that adheres to the problem-solving heuristic of making the locally optimal choice at each phase with hoping to find an overall optimum. Generally, a greedy technique does not search for an optimal solution, however, it may produce optimal solutions from each local that estimates an overall optimum solution at a significant period.[6] The greedy algorithm finds the optimum solution in timetabling problems by 1) choosing the interval x that will finish first; 2) removing the interval x and all other intervals that intersect with the interval x from a set of candidate intervals, and 3) repeating until a set of candidate intervals is emptied. When choosing an interval at step 1, it is opted to remove a lot of intervals in step 2. However, all given intervals may intersect at the time when x will be finished, and as such, these intervals will intersect each other at the same period. Thus, at least 1 of these candidate intervals can be considered as an optimum solution, which is proof that this algorithm will indeed search for an optimum solution, hence, the term "greedy".[7]

The utilization of the greedy algorithm to solve the timetabling problem by considering complicated multiple hard constraint conditions such as 1) uniform teaching resources, which include teachers, students, and classrooms, with practicable timeslots, and 2) a balance for all the curricular loads of students and teachers' satisfaction of preferred lecture schedule, has shown that the greedy algorithm is indeed effective and that the runtime is significantly shortened as compared to other techniques used in the timetabling system.[8]

c. Integer Linear Programming Algorithm

Linear programming (LP), which is also called linear optimization, is a technique to attain the best result in a mathematical model that requires to be characterized through linear relationships.[7] It is a special case of mathematical optimization with a feasible region called a convex polytope composed of a set of finitely many intersecting half spaces derived using linear equality and inequality constraints. The main goal of this algorithm is to search for a point in the polyhedron's function where the largest or smallest value exists. If the value being searched for and values to choose from the feasible region needs to be integers, then the problem is referred to as an integer programming (IP) or integer linear programming (ILP) problem.[21]

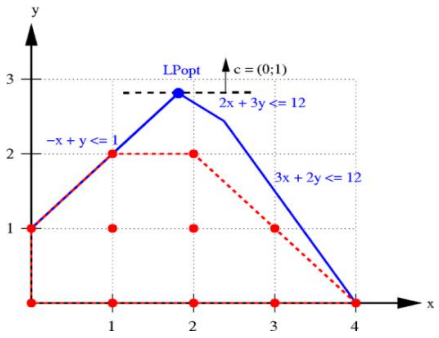

Figure 1. A graph showing Integer Programming Polytope from a relaxed Linear Programming Polyhedron derived from multiple inequality formulae (image source*: www.mathcs.emory.edu*).

Figure 1 illustrates how the best result (largest value) is derived using integer programming. Given a polyhedron (blue line) defined by several inequality formulae, the goal (black dotted line above the polyhedron) is to find the biggest integer values when the highest point (LPopt) of the polyhedron is extended upwards. Relaxing the polyhedron to attain integer values is done by creating a polytope (red dots) by marking all the integers inside the original polyhedron which will form the smallest polyhedron (red dotted line) possible inside the original polyhedron. Among the points in the polytope, the highest integers will be selected as a base for the goal.

There are various fields of study where linear programming can be utilized such as vehicle service scheduling which then can be adapted to university class timetabling by modifying the routes to courses/subjects, vehicles to classrooms, vehicle drivers to teachers, and passengers to students.[21] IP techniques can be used to model university class schedules which can integrate several academic operational requirements and constraints and generate a feasible outcome that can be used for practical use.[9]

d. Bi-Partite Graph Approach

A bipartite graph, also called a bigraph, is a graph that showcases the relative connections (relations) of the elements of two or more sets. Each set is called a vertex set while elements of the set are called vertices. The graph coloring approach, an adaptation of the bipartite graph, use a unique color for each vertex set for easy visualizations.[6] In a bipartite graph, an element from a set is linked through a line to one or all of the elements of another set, thus, forming bi-partite relations.[7] Figure 2 exhibits two vertex sets (U and V) in which each element from both sets are related to all elements (vertices) from the other set.

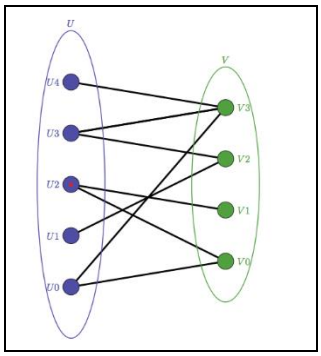

Figure *2:* A simple bipartite graph of vertex sets U and V showing the relationships of their corresponding elements as vertices. (image source*: stackexchange.com*).

Bi-partite graph approach can be used in timetabling problems by converting the list of teachers, classrooms, subjects, students, courses into vertex sets where each element in the list will become an element of a vertex set. Relations will be linked from corresponding vertices and then vertex value matching, collisions, non-collisions, and possible optimizations can be performed.[10] Results of experimental studies demonstrated that utilizing the bi-partite graph approach can solve university timetabling problems.[22]

MATERIALS AND METHODS

The evaluations in this paper were conducted in two ways: input constraint evaluation and computational hardness evaluation. In the input constraint evaluation, all required input data on all the five sub-problems of the Timetabling problem were listed and assessed which are hard constraints that must be positioned in the highest hierarchy and which should be its dependents. Meanwhile, the computational hardness evaluation calculated the total time and computer resources (memory and processor) consumed by each algorithm until it finds a feasible solution by implementing first the proposed 5-stage UTP solution process and comparing it with the simulation results of direct algorithm processing without the using the model.

To conduct the evaluations, the researchers created a VisualBasic.NET tool (Figure 3) using Microsoft Visual Studio 2010 to simulate each algorithm. Requisites, factors, and guidelines discussed in each sub-problem of UTP were incorporated and test data were read from an input xml file.

A dummy set of hard and soft constraints, a list of teachers, classrooms, subjects, block sections,

and students were prepared and served as test data for the simulation. The test data is equivalent to enrollment scheduling data for a semester of

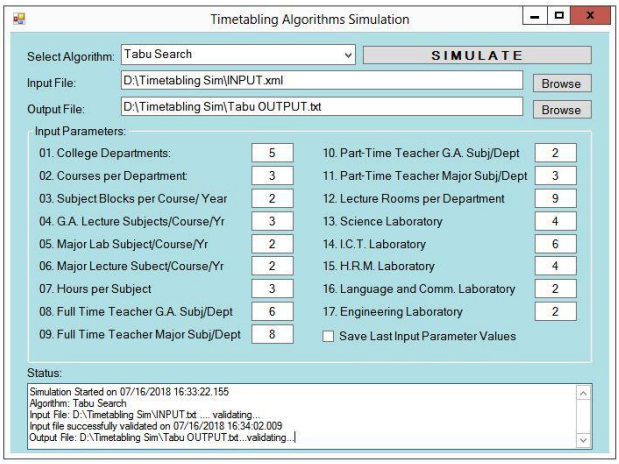

Figure 3*:* Simulation of AI-based local search and optimization algorithms.

a university college with 5 departments, each with four 4-year degree courses for the morning shift. Tables 1 and 2 display the details of constraints and input parameters used in this study.

Table 1: Input Constraints

#	**Constraint**	**Type**
1	Subjects to follow are based on the curriculum	Hard
2	The number of hours per subject per week is equal to the corresponding subject's number of units	Hard
3	Laboratory subjects are to be held in respective laboratory rooms	Hard
4	Full-time teachers' daily teaching load minimum is 5 and a maximum of 6	Hard
5	Daily student lesson hours' minimum is 4 and a maximum of 8	Hard
6	Teachers should teach a subject based on expertise/ specialization	Hard
7	Classes are to be held from Monday to Friday	Hard
8	Part-time teacher's daily minimum teaching load is 2 and a maximum of 4	Soft

9	Lecture classes are prioritized to be held in respective departments	Soft
10	Each subject is to be taught at least twice a week	Soft

Listing of Hard and soft constraints used in the simulation of the four AI-based local search and optimization algorithms.

Table 2: Input Parameters

#	**Parameter**	**Count**
1	College Departments	5
2	Courses per College Department (4-Year)	3
3	Subject Blocks per course per year	2
4	General Academic Lecture Subject per course/yr	3
5	Specialization Laboratory Subject per course/yr	2
6	Specialization Lecturer Subject per course/year	2
7	Hours per subject (Laboratory and Lecture)	3
8	General Academic Subject Fulltime Teacher per Dept.	6
9	Specialized (Major) Subject Fulltime Teacher per Dept.	8
10	General Academic Subject Part-time Teacher per Dept.	2
11	Specialized (Major) Subject Part-time Teacher per Dept.	3
12	Lecture Classrooms per Department	9

13	Science Laboratory	4
14	I.C.T. Laboratory	6
15	H.R.M. Laboratory	4
16	Language and Communication Laboratory	2
17	Engineering Laboratory	2

Listing of input parameters used in the simulation of the four AI-based local search and optimization algorithms.

The following formulae were devised and used to check the validity of the count of each input parameter in connection with the given constraints. Equation 1 pertains to the Subject Hours

Sh formula:

$$Sh = \mathrm{C} \times \mathrm{Y} \times \mathrm{S} \times \mathrm{D} \times \mathrm{B} \times \mathrm{H} \quad \textbf{(1)}$$

to get the number of subject hours in a week where C equals the number of Courses per Department, Y equals the number of Year Level per Course, S equal s the number of Subjects (Lab/Lec) per Course, D equals the number of Departments, B equals the Number of Blocks per Year Level, and H is the allotted Hours per Subject.

Equation 2, the Rooms formula, is devised to obtain the minimum number of rooms required:

$$Rooms = \frac{Sh}{Rh} \quad \textbf{(2)}$$

where Sh refers to the result of the Subject Hours formula and

Rh is the allotted hours to a room per week.

To get the number of weekly teaching hours per teacher ITH, Equation 3, the Individual Teacher Hours formula, is devised:

$$ITH = Th \times d \quad \textbf{(3)}$$

where Th refers to the minimum or a maximum number of teaching hours per day and d as the number of days per week. Meanwhile, to get the minimum required number of teachers, Equation 4 is used:

$$Teachers = \frac{Sh}{ITH} \quad \textbf{(4)}$$

where Sh is the result of Equation 1 and ITH equals the result of Equation 3.

The goal of the evaluation is for algorithms to prepare a conflict-free schedule for 95 teachers, 53 rooms, and 840 class blocks by observing the proposed stage-by-stage process model and preparing the same schedule without following the model.

During each scheduling algorithm simulation, the VB.NET tool that the researchers created records the start, end, and elapsed times in ticks to monitor which among the algorithms performed well following time. A single tick represents one ten-millionth of a second or one hundred nanoseconds, this means that there are 10,000 ticks in a millisecond or 10 million ticks in a second[23]. To convert

the number of ticks covered by the hash function in generating hash values into seconds, Equation 5 is used:

$$time_{seconds} = \frac{time_{ticks}}{10{,}000 \text{ ticks/second}} \quad (5)$$

Meanwhile, Microsoft's Process Monitor version 3.50 was used to monitor and log the real-time effect of the VB.NET tool created on computer memory and CPU usage.[24] Microsoft's Process Monitor version 3.50 is part of Microsoft's Windows Sysinternals utilities that help users in managing, diagnosing, and troubleshooting Windows systems and applications in real-time.[25] The simulation was run on a regular desktop computer with 2.4 Gigahertz Intel Core i5 processor, 4 gigabytes of random access memory, 256 gigabytes of hard disk space, and installed with Microsoft Windows 8 64-bit operating system. Results of the simulation are saved via a regular text output file to allow a manual review of each schedule created.

RESULT AND DISCUSSION

Figure 4 shows the results of the input constraints evaluation performed by the researchers after considering all the input data, factors, and required parameters of each sub-problem of the Timetabling Problem. It exhibits the proposed

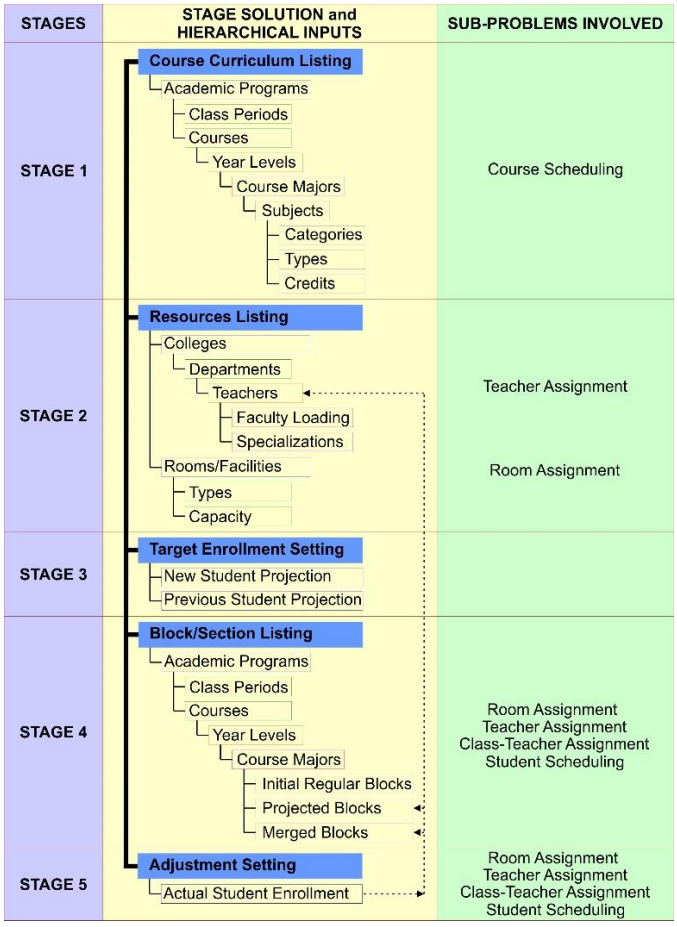

Figure 4: Input Constraints Evaluation Results depicting the proposed 5-Stage University Timetabling Solution Process Model.

the generic solution in solving the UTP by dividing the problem into five stages and solving a sub-problem per solution stage after providing the required input data constraints. The figure shows that among all input constraints, input data about the course curriculum is positioned on the highest hierarchy and the rest of the input data form its subset. The course curriculum listing requires to list first the following in order: academic programs that the school is offering; class periods (either morning or evening classes); list of courses for each academic program; each course would have several year levels and on each year level, course majors should be listed; after listing all course majors per year level, list all prescribed subjects together with the subject's category (either major or minor subject), type (either lecture or laboratory/elective type of subject), and credits or the number of prescribed units to be eared after taking the subject. The course curriculum listing stage involves the full and final completion of the solution for the course scheduling sub-problem.

Stage 2 refers the Resources Listing, a resource can be either (1) a teacher belonging to a particular college and department with a specified number of minimum and maximum hours of teaching load and field of specialization the teacher can teach; and (2) a classroom or academic facility with a maximum capacity which is either a laboratory room or regular classroom; Stage 2 partially solves the Teacher Assignment and Rooms Assignment sub-problem of UTP. Meanwhile, Stage 3, the Target Enrollment Setting Stage, does not solve any sub-problem of UTP but is a key task to perform wherein the projected number of new students and previous students who would enroll during the term will be inputted to be used by each sub-problem of UTP and solve UTP as a whole.

In Stage 4, initially projected and merged class blocks or sections are to be inputted according to their respective academic programs, periods, courses, year level, and course majors. Under this stage, partial or complete solutions for Room Assignment, Teacher Assignment, Class-Teacher Assignment, and Student Scheduling sub-problems of the UTP can be achieved. Finally, Stage 5 relates to the Adjustment Setting Stage with which the actual number of enrolled students is inputted. If enrollment turnout is either higher or lower than expected, it will result in the adjustment of the number of teachers who will teach during the term and the number of class blocks that will be opened. At this stage, a complete solution for the Timetabling problem can be achieved.

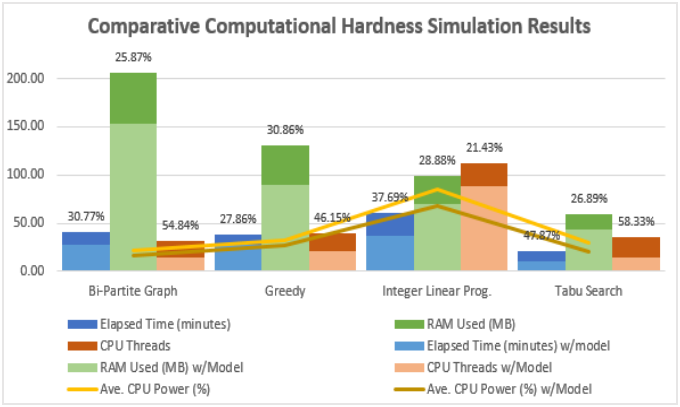

Figure 5*:* A Comparative Computational Hardness Evaluation Results of the 4 subject AI-based Timetabling Algorithms in terms of speed and computational resources in the use and non-use of the proposed 5-Stage University Timetabling Solution Process Model.

Figure 5 highlights the comparative computational hardness evaluation regarding the total time and computer resources (memory and processor) consumed by each algorithm until it finds a feasible solution in the usage and non-usage of the proposed 5-Stage University Timetabling Solution Process Model. Utilizing the proposed model, a significant decrease in elapsed time and computation resources has been achieved in processing test data and input constraints with the 4 subject algorithms. An average difference of 45.19% in the use of CPU Threads was achieved among the 4 algorithms in which Tabu Search benefited a 58.33% CPU Thread usage decrease,

followed by Bi-Partite Graph (54.84% decrease), Greedy Algorithm (46.15% decrease) and Integer Linear Programming (21.43% decrease). The next significant difference in using the model can be seen in the elapsed time with an average decrease of 36.05% among the subject algorithms with the highest time reduction in Tabu Search (47.87% decrease) followed by ILP (37.69% decrease), Bi-Partite Graph (30.77% decrease) and Greedy Algorithm (27.86% decrease). Meanwhile, in terms of RAM usage, a substantial average decrease of 28.13% has been achieved in using the model in which the Greedy Algorithm saved 30.86% RAM usage followed by ILP (28.88%), Tabu Search (26.89%), and Bi-Partite Graph (25.87%).

CONCLUSION

The implementation of the proposed multi-stage solution process in hierarchically setting the multi-constraint inputs and environment of the University Timetabling Problem and dealing with all the sub-problems of UTP is a promising model when paired with any of the subject algorithms as it significantly aids in finding an optimum solution faster in terms of time and computer resources. The model is also usable in identifying criteria to be set forth as the basis for human intervention as a heuristic to be performed in as little as possible in every stage of the solution process. Being able to provide organized input constraints, a feasible solution in finding a conflict-free schedule for 95 teachers, 53 rooms, and 840 block classes is established faster using the four subject AI-based local search and optimization algorithms. Simulation results proved that the model can significantly reduce CPU Threads Usage down to an average of 45.19%, find the solution faster by an average elapsed time difference of 36.05%, and decrease computer RAM usage down to an average of 28.13%, respectively when implemented with Tabu Search, Greedy Algorithm, Bi-Partite Graph Approach, and Integer Linear Programming.

REFERENCE

[1] Domenech, B. & Lusa, A. (**2016**). *A MILP model for the teacher assignment problem considering teachers' preferences.* European Journal of Operational Research Volume 249, Issue 3, 16 March 2016, Pages 1153-1160. Elsevier.

[2] Poole, David; Mackworth, Alan (**2017**). *Artificial Intelligence: Foundations of Computational Agents (2nd ed.).* Cambridge University Press. ISBN 9781107195394.

[3] Pereira, Valdecy and Costa, Helder Gomes (**2016**). *Linear Integer Model for the Course Timetabling Problem of a Faculty in Rio de Janeiro.* Advances in Operations Research Volume 2016. doi: http://dx.doi.org/10.1155/2016/7597062.

[4] Almeida, Maria Weslane S., Medeiros, João Paulo S. and Oliveira, Patrícia R. (**2015**). *Solving the Academic Timetable Problem Thinking on Student Needs.* 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA). Florida: IEEE. ISBN: 978-1-5090-0287-0.

[5] Babaei, Hamed; Karimpour, Jaber; and Hadidi, Amin (**2015**). *A Survey of Approaches for University Course Timetabling Problem.* Computers and Industrial Engineering Volume 86. Elsevier.

[6] Christian, Brian and Griffiths, Tom (**2016**). *Algorithms to Live By: The Computer Science of Human Decisions.* Henry Holt and Company. ISBN: 1627790373.

[7] Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. (**2014**). *Introduction to Algorithms (3rd ed.).* MIT Press and McGraw-Hill. pp. 966–1021. ISBN 0-262-03293-7

[8] Zhang, D., Guo, S., Zhang, W. & Yan, S. (**2014**). *A novel greedy heuristic algorithm for university course timetabling problem.* Intelligent Control and Automation (WCICA), 2014 11th

World Congress On Intelligent Control and Automation. IEEE. Shenyang, China

[9] Phillips, Antony, Waterer, Hamish, Ehrgott, Matthias, and Ryan, David (**2014).** *Integer Programming Methods for Large Scale Practical Classroom Assignment Problems.* Lancaster EPrints 2015. Accessed February 12, 2018.

[10] Russell, Stuart J.; Norvig, Peter (**2015**), *Artificial Intelligence: A Modern Approach 3rd Edition* (Paperback). New Jersey: Prentice-Hall, ISBN: 9332543518, 978-9332543515.

[11] Fonseca, George H.G., Santos, Haroldo G., Carrano, Eduardo G., Stidsen, Thomas J.R. (**2017**). *Integer programming techniques for educational timetabling.* European Journal of Operational Research Volume 262, Issue 1, 1 October 2017, Pages 28-39. Elsevier.

[12] Bahel, M. & Thomas A. (**2017**). *Innovative Evolutionary Algorithm Approach for Class-Teacher Timetabling Problem.* Bhilai Institute of Technology, Durg District, India.

[13] Cheng, E., Kruk, S., & Lipman, M.J. (**2013**). *Flow Formulations for the Student Scheduling Problem.* Practice and Theory of Automated Timetabling IV, 4th International Conference, PATAT 2013, Gent, Belgium.

[14] Gogos, C., Alefragis, P., & Housos, E. (**2007**). *A Multi-Staged Algorithmic Process for the Solution of the Examination Timetabling Problem*. Dept. of Electrical and Computer Engineering. The University of Patras-Greece. PDF. Retrieved from

http://www.cs.qub.ac.uk/itc2007/winner/bestexamsolutions/cgogos.pdf. Date Accessed: 01-05-2019.

[15] Festa, P. and Resende, M. G. C. (**2002**). *GRASP: An Annotated Bibliography.* Essays and Surveys on Metaheuristics, pp. 325–367, Kluwer Academic Publishers.

[16] Sørensen, M. & Dahms, F. H.W. (**2014**). *A Two-Stage Decomposition of High School Timetabling Applied to Cases in Denmark.* Computers & Operations Research, 43, 36–49. DOI: 10.1016/j.cor.2013.08.025

[17] Müller, T., & Barták, R. (**2015**). *Interactive Timetabling.* Charles University, Czech Republic.

[18] R. Barták (**2000**). *Dynamic Constraint Models for Planning and Scheduling Problems*. In New Trends in Constraints, LNAI 1865, pp. 237-255, Springer.

[19] Rovatsos, Michael, Vouros, George & Julian, Vicente (**2015**). *Multi-Agent Systems and Agreement Technologies.* 13th European Conference, EUMAS 2015, and Third International Conference, AT 2015, Athens, Greece. Springer. ISBN: 9783319335094.

[20] Bindra, Richa, Modi, Nishank, Shah, Rahil and Puri , Simran (**2014**). *University Course Scheduling using Tabu Search Algorithm.* Thesis: University of Waterloo. Retrieved 20 March 2018.

[21] Ganguli, Runa and Roy, Siddhartha (**2017**). *A Study on Course Timetable Scheduling using Graph Coloring Approach.* International Journal of Computational and Applied Mathematics Volume 12, Number 2 (2017), pp. 469-485. Research India Publications. ISSN 1819-4966.

[22] Microsoft Developer Network. Article: *DateTime.Ticks Property*

.https://msdn.microsoft.com/en-us/library/system.datetime. ticks(v=vs.110).aspx, accessed **2017**-03-09

[23] Russinovich, M., Kim, L., and Sharkey, K. (**2017**) *Windows Sysinternals*. Retrieved from https://docs. microsoft.com/en-us/sysinternals/