International Journal of Computer Applications (0975 - 8887)
Volume 183 - No.50, February 2022

Using Random Scheduling Technique with Crowd-Sourcing to Solve Exam Timetable

Charles Roland Haruna
University of Cape Coast
Department of Computer Science and Information Technology
Cape Coast, Ghana

Elliot Attipoe
University of Cape Coast
Department of Computer Science and Information Technology
Cape Coast, Ghana

Isaac Armah Mensah
University of Cape Coast
Department of Computer Science and Information Technology
Cape Coast, Ghana

Kwame Opuni-Boachie Obour Agyekum
Kwame Nkrumah University of Science and Technology
Department of Telecommunication Engineering
Kumasi, Ghana

ABSTRACT

An examination timetable is arranging temporary, a set of meetings making sure that all given constraints are fulfilled. A university has many colleges, faculties and programs within departments, each having their plan about how, when and where their examination should be run. Students can offer courses from a combination of several programs in the same or different departments, from different faculties and or colleges. Thus, scheduling of an examination timetable is a large and complex assignment. Various university examination timetabling systems have been proposed and developed, but not all university examination timetables can be scheduled by machines only, for example in the case of the University of Cape Coast. Both machine and experts are required. In this paper, a case study is presented and a hybrid technique tailored to suit the University of Cape Coast examination timetable scheduling is proposed, where machines are first used then experts complete the scheduling. Real data set from the University is used in this research and the effectiveness of the proposed technique is presented by performing multiple experiments and the results of an examination timetable using real data sets from past academic years (semesters) from the University of Cape Coast are discussed.

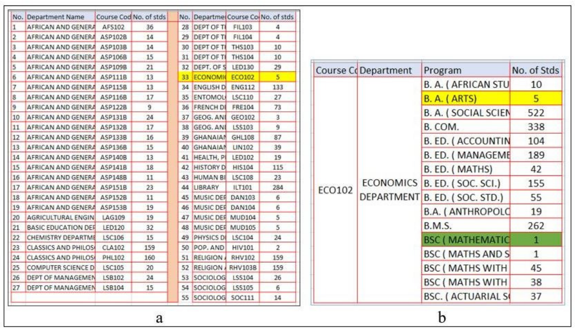
Keywords

Timetabling Problem, Examination Timetabling Scheduling, Random Scheduling Algorithm, Crowd-Sourcing

1. INTRODUCTION

In all universities, scheduling an examination timetable is a serious challenge with respect to concerns on assigning venues over a temporal period of time to examinations. The notable major challenges examination scheduling faces include; when students' population exceeds the available resources, when the university's examination venues within time periods are limited and to satisfy all other constraints. For decades, examination timetabling has become one of the most studied domains in the research communities, this is because of the importance of scheduling examination in numerous academic institutions across the world. In spite of this, researchers have aimed their works at developing methodologies that generate the best timetables in solving a single problem, Qu et al.[4].

Figure 1a is a sample list of records of all 55 courses, read by level 100 B.A Arts students only, with different course combinations. The columns, Department Name, Course Code and No. of stds are representing the departments the courses belong to, the codes of the courses and the number of students reading each course respectively. It can be seen that there are courses (course codes) from distinct departments, for example LSC106 from the chemistry department, LAG109 from the agricultural engineering department, which are read by the students as either electives or compulsory courses. Due to the complexities of the course selections for a group of students, the examination timetable scheduling for UCC is always a challenge. The algorithm must therefore take into consideration so many factors when scheduling. One of the courses read by some students of the group, record number 33 (Figure 1a), highlighted in yellow with a course code ECO102 is mounted by the Economics Department. Figure 1b shows additional details of the course code ECO102 such as; For that particular semester, a group of students with different class size numbers from different program backgrounds who are reading the same course. Highlighted in green and yellow are programs BSc Mathematics and BA Arts respectively.
Figure 2 shows a pictorial diagram depicting the relationship between BSc Mathematics and BA Arts. The diagram also shows the relationships among programs(level), departments, courses and students. The green and yellow rectangles depict programs BSc Mathematics(100) and BA Arts(100) respectively. The different colors of ellipses represent departments from which the two programs select their courses for example department of mathematics, department of computer science, department of music and so on. In each colored ellipse are unique courses for each department rep-



No.	Department Name	Course Cod	No. of stds
1	AFRICAN AND GENERA	AFS102	36
2	AFRICAN AND GENERA	ASP102B	14
3	AFRICAN AND GENERA	ASP103B	14
4	AFRICAN AND GENERA	ASP106B	15
5	AFRICAN AND GENERA	ASP109B	21
6	AFRICAN AND GENERA	ASP111B	13
7	AFRICAN AND GENERA	ASP115B	13
8	AFRICAN AND GENERA	ASP116B	17
9	AFRICAN AND GENERA	ASP122B	9
10	AFRICAN AND GENERA	ASP131B	24
11	AFRICAN AND GENERA	ASP132B	17
12	AFRICAN AND GENERA	ASP133B	16
13	AFRICAN AND GENERA	ASP136B	15
14	AFRICAN AND GENERA	ASP140B	13
15	AFRICAN AND GENERA	ASP141B	18
16	AFRICAN AND GENERA	ASP148B	11
17	AFRICAN AND GENERA	ASP151B	23
18	AFRICAN AND GENERA	ASP152B	11
19	AFRICAN AND GENERA	ASP153B	19
20	AGRICULTURAL ENGIN	LAG109	19
21	BASIC EDUCATION DEF	LED120	32
22	CHEMISTRY DEPARTME	LSC106	15
23	CLASSICS AND PHILOS	CLA102	159
24	CLASSICS AND PHILOS	PHL102	160
25	COMPUTER SCIENCE D	LSC105	20
26	DEPT OF MANAGEMEN	LSB102	24
27	DEPT OF MANAGEMEN	LSB104	15

No.	Departme	Course Co	No. of stds
28	DEPT OF TI	FIL103	4
29	DEPT OF TI	FIL104	4
30	DEPT OF TI	THS103	10
31	DEPT OF TI	THS104	10
32	DEPT. OF S	LED130	29
33	ECONOMI	ECO102	5
34	ENGLISH D	ENG112	133
35	ENTOMOL	LSC110	27
36	FRENCH DI	FRE104	73
37	GEOG. ANI	GEO102	3
38	GEOG. ANI	LSS103	9
39	GHANAIAN	GHL108	87
40	GHANAIAN	LIN102	39
41	HEALTH, PI	LED102	19
42	HISTORY D	HIS104	115
43	HUMAN BI	LSC108	23
44	LIBRARY	ILT101	284
45	MUSIC DEF	DAN103	6
46	MUSIC DEF	DAN104	6
47	MUSIC DEF	MUD104	5
48	MUSIC DEF	MUD105	5
49	PHYSICS DI	LSC104	24
50	POP. AND	HIV101	2
51	RELIGION	RHV102	159
52	RELIGION	RHV103B	159
53	SOCIOLOG	LSS104	26
54	SOCIOLOG	LSS105	6
55	SOCIOLOG	SOC111	14

a

Course Co	Department	Program	No. of Stds
ECO102	ECONOMICS DEPARTMENT	B. A. (AFRICAN STU	10
		B. A. (ARTS)	5
		B. A. (SOCIAL SCIEN	522
		B. COM.	338
		B. ED. (ACCOUNTIN	104
		B. ED. (MANAGEME	189
		B. ED. (MATHS)	42
		B. ED. (SOC. SCI.)	155
		B. ED. (SOC. STD.)	55
		B.A. (ANTHROPOLC	19
		B.M.S.	262
		BSC (MATHEMATIC	1
		BSC (MATHS AND S	1
		BSC (MATHS WITH	45
		BSC (MATHS WITH	38
		BSC. (ACTUARIAL S	37

b

Fig. 1. A sample of a program and its course combinations (a) and an example of a course code with its different program backgrounds (b).

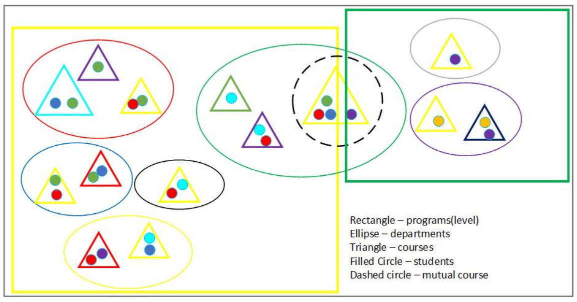


Fig. 2. A graphical example of course conflict.

resented by triangles like elementary programming from department of computer science and analytic geometry and calculus from the mathematics department. Finally, students who read different courses are represented by filled colored circles; for example, a circle filled with red implies the student is reading six courses, four from four different departments and two from the same department. The course highlighted with black dashed lines is representing ECO102, a mutual course which is read by level 100 students from both BSc Mathematics and BA Arts. Timetabling constraints studied and presented by Burke, et al.[1] were divided into two categories; hard and soft constraints. Hard constraints must always be satisfied.

1.1 Hard Constraints

A timetable which does not conform to a hard constraint is not a feasible solution. Examples of hard constraints are:

—In no period must a student be allowed to sit for more than one exam.

—Sufficient resources such as seats must be available to students for all the exams that have been scheduled for that period.

—Given a timeslot (timeslots explained later), each room capacity should not be exceeded by the number of students sitting a course examination.

—Practical examinations (exam A) must come before its theoretical examination (exam B).

1.2 Soft Constraints

In contrast, soft constraints vary from one university to another. These rules are desirable or requested but are not completely important; some of them can be violated. Some common soft constraints for university timetabling presented by Burke et al.[2] are as follows:

—Assignments of time: An exam may need to be scheduled in a particular time period.

—Spreading exams: As much as possible, students should not have exams in consecutive periods or two exams on the same day unless one is a re-sit examination for a continuing student (case of UCC).

—An exam, such as practical courses or oral language courses must be conducted in a particular room (as it requires special resources only available in that room).

Since students have the freedom in their choice of courses, there always is a problem irrespective of whether the constraints are taken into considerations or not. Scheduling an examination to fit into a limited time interval without creating conflicts for some students is always a difficulty, Laporte et al.[3]. Thus, this research presents a hybrid-based approach to cater for such occurrences.

1.3 Intermediate Constraints

In this paper, another set of constraints are presented and described, which are called intermediate constraints in relation to University of Cape Coast examination timetable scheduling. Though these rules can be broken, they are not. If broken, the whole system will not function as desired. Examples of University of Cape Coast intermediate constraints are:

—There exist some level 100 compulsory courses that must be scheduled within the first three (3) days of each end of semester's examination.

—Different courses of different programs can be allocated the same room, irrespective of the students' levels, as long as the room is not filled up.

—Care must be taken when scheduling continuing students' (levels 300 and 400) courses, to avoid such students writing two papers in the same period due to the possibility of re-sit examination.

Note that a re-sit examination occurs when a student in either level 300 or 400 has to rewrite a failed examination from a lower level (200 0r 300).

2. MOTIVATION AND OBJECTIVE

Implementing existing works in scheduling the examination timetable of the university have performance limitations. Owing to the fact that at UCC, a department may have a wide range of majors where students can read courses from different departments. Most of the existing works as well as the current technique being used find it difficult scheduling this scenario. Another reason for choosing UCC as the case study is the nearness of on-field testing and availability of data set. The inter-departmental courses, where students belonging to one department have the opportunity to read courses from N number of other departments to satisfy the total credits hours made this case study an interesting choice. In this work a human-machine based approach is proposed where firstly algorithms are executed by machines then experts are used to complete the scheduling, which is not common in this field of research.

3. RELATED WORK

There exists so many automated works that have been proposed to solve the problems that academic institutions encounter when scheduling examination timetable. Abayomi-Alli et al.[5] proposed enhanced particle swarm optimization (PSO) for solving the examination timetable scheduling problems. PSO was used for resolving the problems at the Federal University of Agriculture, Abeokuta, Nigeria. The technique made a combinatory approach using PSO with local search mechanism to improve on the effectiveness of the algorithm against the manual timetabling method. Abdul-Rahman et al.[6] in their work, employed graph coloring heuristics at each step of the examination timetable scheduling. The authors in assigning examinations to rooms based on capacity, made use of the bin packing heuristics concept.

In [7] by Mauritsius et al., the authors published a detailed report using three unique neighborhood structures to conduct an investigation on the implementation of a heuristic hybridized in conjunction with a simulated annealing-based heuristic. The research focused on using graph coloring to significantly reduce problems associated with examination timetabling. Chu et al. in paper [8] employed particle swam optimization to solve distinct timetable scheduling problems and experimentally proved that as well. Practical Solutions to the problems of the examination timetable scheduling problems of the University of Technology of Compi'egne were developed by Boufflet et al.[9] presenting three tools in the process; the first being the approximate technique centered on the tree search, the next tool also centered on tabu technique and the last is based on a computer aided design system.The authors realized the best being the first tool. In generating the extent of the value of each examination's difficulty degree, Abdul et al.[10] used the non-linear heuristic aspect of two graph coloring; the largest and saturation degrees, to solve the examination timetable challenges.

Saat et al.[11] developed an application using visual basic, a graphical user interface and artificial intelligence called eMaS that offer valuable attributes that presents a solution to solving the examination timetable problem in precise and a fast manner. Mandal et al.[12] presented a feasible solution after investigating performances of the key trajectory metaheuristics and graph heuristics employing six graph heuristics namely; largest degree (LD), largest weighted degree (LWD), largest enrolment degree (LE), and three hybrid heuristic with saturation degree (SD) being SD-LD, SD-LE, and SD-LWD. The sole objective by Kahar et al.[13] was to formulate a mathematical model while testing it to produce an examination timetable that outperformed some existing techniques. A doctoral research dissertation by Samaya[14] presented findings from the work, to find techniques to overcome the scheduling challenges of examination timetable. Aminu et al.[15] from the Yusuf Maitama Sule University Kano, Nigeria presented a method using genetic algorithms for the problem of scheduling examination timetable and invigilation, that provided an optimal solution. Elsaka[16] modeled the examination timetable applying CSP with Optimization Programming Language and in so doing, automatically generating a schedule which is free of conflicts.

4. PROBLEM DESCRIPTION

In this paper, the authors aim at automating an examination timetable schedule for UCC. Nonetheless, the technique proposed can be applied to solve the exams timetable problems for other universities. The timetable to be treated in this work is constructed for all levels of the university degrees; undergraduate, masters and PhD courses, so far as there are examinations to be sat for the courses. The major problem of the University is with the conflicts that arise when scheduling undergraduate examination timetable, thus this research focuses on undergraduate examination timetable scheduling. Once the problem is solved for the undergraduate exam timetable scheduling, the technique can be used at all levels. The University of Cape Coast examination is scheduled within a period of two weeks (10 weekdays), and even though the length of period may vary, the algorithm can still schedule the timetable. There are three time periods in a day, and each consists of 120 minutes; 8:30 – 10:30, 12:00-14:00 and 15:30-17:30. Five main entities are used to define the problem; Student Number, Venue, Time, Course and Date. It is worth noting that the algorithm is not affected by the number of days and duration of time for the exams. One notable rule of the university is to, within the first days, schedule all level 100 elective courses which are currently done manually by experts. Due to the huge number of students (an entire year group of students, making thousands of them) reading each level 100 compulsory course, the students are divided into groups after registration. The remainder of the courses are then also scheduled.

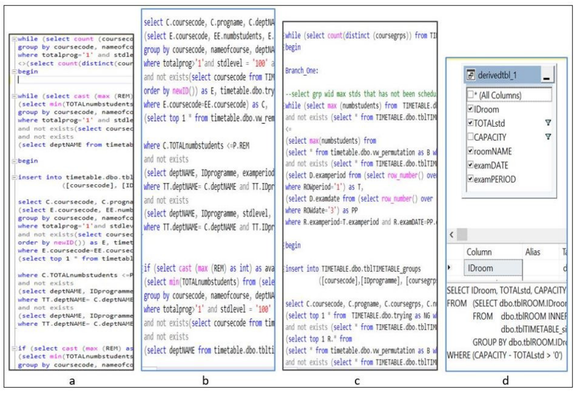
4.1 PROBLEM DEFINITION

The examination timetable is formulated as follows: A set of Problem = $\{C, T, S\}$, where T = $\{T_1, T_2, ..., T_t\}$ are the timeslots for examinations and each consists of a date, a time (duration) and a venue code. For example, the set of T_t may be any of the following sets; $\{(25-May-2021; 8:30-10:30; LLT); (25-May-2021; 12:00-14:00; LLT); (25-May-021; 15:30-17:30; LLT); (25-May-2021; 8:30-10:30; LT12); (25-May-2021; 8:30-10:30; LT5); (25-May-2021; 8:30-10:30; LT22)\}$. C = $\{C_1, C_2, ..., C_c\}$ is the set of examination courses. S = $\{S_1, S_2, ..., S_s\}$ is the number of students who will participate in C_c.

4.1.1 Machine-Based Random Local Search. In this paper, a random local search method is designed and implemented on the University of Cape Coast timetable scheduling. The machine-based approach schedules the timetable in three phases, with the end of a phase signifying the beginning of another, in this order:

(1) Randomly schedules exams for the level 100s with respect to the compulsory courses within the first three days. Presently, this phase of scheduling is done manually by user experts. By automating this phase, the technique eradicates the manual way significantly. Exams for all other levels but level 100s can also be scheduled into an unfilled venue of timeslots, T_t, within the first three days.

The pseudocode of the random scheduling of the first phase is as follows. The algorithm randomly selects a group related to a level 100 compulsory course. It then randomly selects a timeslot within the first three days, and schedules the group, irrespective of the capacity of venue. It checks if the group's students have all been scheduled. If not, it picks another timeslot with similar date and time and nearby venue as an already scheduled timeslot and same course and schedules the remainder of the students. It does this iteration till a group is exhausted. If all group's students have been scheduled, it then selects another group of the same elective course. Now, if an already scheduled timeslot has seats available, some students of the new group are scheduled, otherwise a new timeslot is selected taking into consideration the location of the venues. The algorithm iterates till all groups of an elective course have been scheduled. Then it selects another elective course and the whole scheduling iterations begin. The algorithm executes



```
while (select count (coursecc
group by coursecode, nameofcc
where totalprog='1' and stdle
<>(select count(distinct(cour
begin

while (select cast (max (REM)
(select min(TOTALnumbstudents
group by coursecode, nameofcc
where totalprog='1' and stdle
and not exists(select coursec
and not exists
(select deptNAME from timetab

begin

insert into timetable.dbo.tbl
        ([coursecode], [ID

select C.coursecode, C.progna
(select E.coursecode, EE.numb
group by coursecode, nameofcc
where totalprog='1'and stdlev
and not exists(select coursec
order by newID()) as E, timet
where E.coursecode=EE.coursec
(select top 1 * from timetabl

where C.TOTALnumbstudents <=P
and not exists
(select deptNAME, IDprogramme
where TT.deptNAME= C.deptNAME
and not exists
(select deptNAME, IDprogramme
where TT.deptNAME= C.deptNAME


if (select cast (max (REM) as
(select min(TOTALnumbstudents
group by coursecode, nameofcc
```

```
select C.coursecode, C.progname, C.deptNA
(select E.coursecode, EE.numbstudents, E.
group by coursecode, nameofcourse, deptNA
where totalprog>'1'and stdlevel = '100' a
and not exists(select coursecode from TIM
order by newID()) as E, timetable.dbo.try
where E.coursecode=EE.coursecode) as C,
(select top 1 * from timetable.dbo.vw_rem

where C.TOTALnumbstudents <=P.REM
and not exists
(select deptNAME, IDprogramme, exameperiod
where TT.deptNAME= C.deptNAME and TT.IDpr
and not exists
(select deptNAME, IDprogramme, stdlevel,
where TT.deptNAME= C.deptNAME and TT.IDpr


if (select cast (max (REM) as int) as ava
(select min(TOTALnumbstudents) from (sele
group by coursecode, nameofcourse, deptNA
where totalprog>'1' and stdlevel = '100'
and not exists(select coursecode from tim
and not exists
(select deptNAME from timetable.dbo.tblti
```

```
while (select count(distinct (coursegrps)) from TI
begin

Branch_One:

--select grp wid max stds that has not been schedu
while (select max (numbstudents) from  TIMETABLE.d
and not exists (select * from TIMETABLE.dbo.tblTIM
<=
(select max(numbstudents) from
(select * from timetable.dbo.vw_permutation as B w
and not exists (select * from TIMETABLE.dbo.tblTIM
(select D.examperiod from (select row_number() ove
where ROWperiod='1') as T,
(select D.examdate from (select row_number() over
where ROWdate='3') as PP
where R.examperiod=T.examperiod and R.examDATE=PP.

begin

insert into TIMETABLE.dbo.tblTIMETABLE_groups
        ([coursecode],[IDprogramme], [coursegrp

select C.coursecode, C.progname, C.coursegrps, C.n
(select top 1 * from  TIMETABLE.dbo.trying as NG w
and not exists (select * from TIMETABLE.dbo.tblTIM
(select top 1 R.* from
(select * from timetable.dbo.vw_permutation as B w
and not exists (select * from TIMETABLE.dbo.tblTIM
```

- [ ] * (All Columns)
- [x] IDroom
- [x] TOTALstd
- [ ] CAPACITY
- [x] roomNAME
- [x] examDATE
- [x] examPERIOD

Column	Alias
IDroom	

```
SELECT IDroom, TOTALstd, CAPACITY
FROM   (SELECT dbo.tblROOM.IDroor
        FROM    dbo.tblROOM INNEF
                dbo.tblTIMETABLE_si
        GROUP BY dbo.tblROOM.IDr
WHERE (CAPACITY - TOTALstd > '0')
```

Fig. 3. Code Snippets.

satisfying all the constraints.

(2) The technique schedules the single-program courses. These courses are read by students of the same program background only.

The algorithm responsible for the single-program courses is implemented. The pseudocode of this algorithm first checks if there are, within the first three days unallocated timeslots, including available seats in an already scheduled timeslot. If yes and a randomly selected non-level 100 single-program course can be scheduled, it is executed. Otherwise, the course is scheduled in a new timeslot selected randomly. If a level 100 single-program course is selected, the algorithm then selects a timeslot not in the first three days and schedules. The algorithm iterates till, if possible, all single-program courses are scheduled not violating any of the rules of the constraints; hard, soft and intermediate.

(3) Finally, the automation schedules multi-program courses. Courses that are read by students with different program backgrounds. These courses usually pose challenges when scheduling at the university due to conflicts encountered while processing.

The third phase of the algorithm responsible for the multiprogram courses is implemented. The pseudocode of this algorithm first checks if there are, within the first three days unallocated timeslots, including available seats in an already scheduled timeslot. If yes and a randomly selected non-level 100 multi-program course can be scheduled, it is executed. Otherwise, the course is scheduled in a new timeslot selected randomly. If a level 100 multi-program course is selected, the algorithm then selects a timeslot not in the first three days and schedules. The algorithm iterates till, if possible, all multiprogram courses are scheduled not violating any of the rules of the constraints; hard, soft and intermediate.

4.2 Human/Expert Input

After the machine-based random scheduling has been executed successfully, the unscheduled courses are submitted to the expert for completion. The expert uses a graphical user interface (GUI) to update the scheduled timetable by scheduling the remaining courses if any.

5. EXPERIMENTS AND DISCUSSIONS

The proposed algorithm was implemented using mainly SQL server for the random scheduling and the back-end while python language was used for the front-end to produce a GUI. The experiments were conducted using a computer with Intel Core i5-8265U CPU @ 1.60GHz (8 CPUs), 1.8GHz, RAM of 8192MB and MS windows 10. The aim of the experiments was to evaluate the results with respect to the efficiency of the proposed algorithm against the technique currently being used on the same data. To have a basis for comparisons of the proposed technique, two different automated techniques were used in the experiments and compared against the current system of scheduling. The three methods for comparisons in the experiments were:

(1) For the first automated algorithm used in the experiments, the authors, subjected the data sets to an algorithm where the remainder of courses were not split into multi and single programs (fused multi and single program courses), after the level 100 compulsory algorithm had been executed.

(2) Then the second automated technique (pseudo codes described *4.1.1*) where the remainder of courses were split into multi and single programs, was subjected to the data.

(3) Current exam timetable scheduling system of UCC where the first three days are done manually and the rest semi-automated.

5.1 Data Sets of Real Problem

In the experiments, data sets of regular undergraduate courses for five semesters from 2019 to 2021 of the University of Cape Coast were used. They are 2019 academic year semesters one and two, 2020 academic year semesters one and two and semester one only of 2021 academic year. All the examination timetable scheduling experiments were done over a period of 10 working days. Table 1 shows details of data sets used in testing the algorithms including records of that of masters and doctorate. However, the algorithms extract and execute records of bachelors degree only. Data sets of five previous semesters were used; Captured in column 1 are the academic years and respective semesters. Column 2 shows the total number of records in each data set and unique courses (in brackets) for each semester to be scheduled. Column 3 represents in each data set the total number of records, the distinct course codes and the percentage of compulsory courses for level 100 that need to be scheduled within the first 3 days of the exam period. The percentages in the third column represent the minimum percentage of manual work that were input. The last column represents the percentage of the courses that the current system automatically schedules. The proposed work aims at eliminating the percentages in the third column while improving the percentages in the last column.

5.2 Performance and Evaluation

The effectiveness of the algorithm was measured by the time of execution and its efficiency against the current system. The algorithm implemented on each data set was executed 5 times. In all the experiments, for each data set, the same scheduled timetable was recorded with different execution times recorded. The varying times recorded were as a result of the random scheduling nature of the algorithm. The process of scheduling the courses implies that each course code is visited and selected at most once. The algorithm randomly selects a course then iterates comparing with courses in the scheduled timetable set.

Table 1. Data Sets.

Academic Year (Semester)	Number of records (Distinct Courses to Schedule)	No. of Records of Level 100 First 3 days compulsory Courses (Distinct Courses) - Percentage	Percentage Efficiency of Current System
2021 (1)	1513 (268)	421 (49) – 18.28%	81.72%
2020 (2)	1635 (276)	466 (60) – 21.74%	78.26%
2020 (1)	1150 (265)	441 (50) – 18.87%	81.13%
2019 (2)	1965 (266)	582 (40) – 15.04%	84.96%
2019 (1)	848 (242)	189 (47) – 19.42%	80.58%

Figure 3 shows snippets of the codes used in executing the algorithm. SQL view like in Figure 3d was created to cater for scenarios such as; calculating the remaining seats available in a scheduled venue to allow another course to be scheduled if possible. Also, if a course is scheduled in a room where the number of students exceeded the room's capacity, the remaining students of that course can be calculated using a view. In the first phase of the scheduling, the timetables were generated for the level 100 compulsory courses that have to be scheduled during the first 3 days of the entire exam period. Generation of timetables for this phase causes problems due to their complexities and are done manually currently at the university. In Table 1, column 3 shows the percentages of work done manually currently. The first phase of the algorithm successfully scheduled the first three days level 100 courses, by eliminating all the manual input. If no violations on hard and intermediate constraints were not encountered, the scheduling was generated quickly and successfully.

Table 2 shows results of the efficiency of the algorithm (fused multi and single program courses). Column 2 shows the number of distinct courses the algorithm needed to schedule for each semester (column 1). Captured in column 3 are number of unscheduled courses and their percentages. Implying the amount of manual work needed to complete the timetable scheduling. In 2021 first semester, 7.76% representing 17 courses out of 219 distinct courses that the algorithm could not schedule. 18 courses out of 216 unique courses for 2020 second semester were not scheduled represented by 8.33%. For 2020 first semester data set, 16 out of 215 courses making 7.44% were not scheduled. The data set for 2019 second semester had 8.85%, representing 20 out of 266 unique courses that were not scheduled. Finally, 10.26% (that is 20 out of 195 unique courses) was not scheduled for 2019 first semester data set. The final column shows the percentages of the scheduled courses. This algorithm on the average, randomly schedules 91.48% of the unique courses. While on the average, 8.53% of the unique courses are scheduled manually. In contrast, the current system on the average automates 81.33% and at the same time manually input 18.67% of the unique courses. The algorithm (fused multi and single program courses) significantly improved the efficiency of the old and current system by about 10%.

In the experiments with respect to splitting the remainder of the course, the algorithm successfully scheduled all single-program courses in each data sets. With no violations on hard and intermediate constraints encountered, the scheduling was generated quickly.

Table 3 shows the algorithm's efficiency results based on multi-program courses after splitting the remainder of courses. Column 4 shows the number of distinct multi-program courses and courses scheduled in parenthesis. Captured in column 5 are number of unscheduled courses and their percentages. Implying the amount of manual work needed to complete the timetable scheduling. Column 6 shows the efficiency percentage of the algorithm on multi-program courses.

In 2021 first semester, 2.28% representing 5 courses out of 114 distinct multi-program courses (out of 219 distinct distinct courses) that the algorithm could not schedule. 6 courses out of 117 distinct multi-program courses (out of 216 distinct distinct courses) for 2020 second semester were not scheduled represented by 2.78%. For 2020 first semester data set, 4 out of 114 distinct multi-program courses (out of 215 distinct distinct courses) making 1.86% were not scheduled. The data set for 2019 second semester had 2.21%, representing 5 out of 116 distinct multi-program courses (out of 226 distinct distinct courses) that were not scheduled. Finally, 4.10%, that is 8 out of 109 distinct multi-program courses (of 195 distinct distinct courses) were not scheduled for 2019 first semester data set. The final column shows the percentages of the scheduled courses.

It was realized that the unscheduled courses were as a result of incredibly huge number of students from vast different program backgrounds reading a course, thus the algorithm does not schedule such courses. The percentages representing a handful of courses were unscheduled, implying experts will manually schedule them using a GUI provided to them. With respect to the 5 data sets used, the proposed algorithm (splitting the remainder of courses) on the average, randomly schedules 97.35% of the unique courses. While on the average, 2.65% of the unique courses are scheduled manually. In contrast, the current system on the average automates 81.33% and at the same time manually input 18.67% of the unique courses. The proposed algorithm significantly improved the efficiency of the old and current system by about 16%.

5.2.1 Efficiency of Proposed Algorithm. Figure 4 shows a graph comparing the efficiency percentages of the proposed system against the current system and the algorithm on the fused multi and single program courses, relative to the five data sets used. It can clearly be seen that the proposed technique in this work represented by the blue bars, outperformed the current system and the fused multi and single course algorithm represented by red and yellow bars respectively, and in all the five data sets used. In splitting the remainder of the courses into multi-program and single-program as proposed in this work, the random scheduling tends to allocate a higher percentage of the courses than not splitting the remainder of the courses. It can be concluded that the random scheduling technique proposed in this work can be implemented at the university

5.2.2 Execution Time of Proposed Algorithm. Figure 5 is a graph showing the execution time it took the algorithm to execute each data set five times. The varying execution times are as a result of the random selection nature of the algorithm. Implying for each data

Table 2. Fused Multi and Single Program Courses Experimental Results.

Academic Year (Semester)	Distinct Courses (Both Multi and Single Program Courses)	Number of Unscheduled courses (Percentage)	Percentage Efficiency
2021 (1)	219	17(7.76%)	92.24%
2020 (2)	216	18(8.33%)	91.67%
2020 (1)	215	16(7.44%)	92.56%
2019 (2)	226	20(8.85%)	91.18%
2019 (1)	195	20(10.26%)	89.74%

Table 3. Multi-Program Courses Experimental Results.

Academic Year (Semester)	Distinct Remainder Courses	Single-Program Courses	Multi-Program Courses (Scheduled Courses)	Number of Unscheduled courses (Percentage)	Percentage Efficiency
2021 (1)	219	104	114 (109)	5 (2.28%)	97.72%
2020 (2)	216	99	117 (112)	6 (2.78%)	97.22%
2020 (1)	215	101	114 (110)	4 (1.86%)	98.14%
2019 (2)	226	110	116 (111)	5 (2.21%)	97.79%
2019 (1)	195	86	109 (101)	8 (4.10%)	95.90%

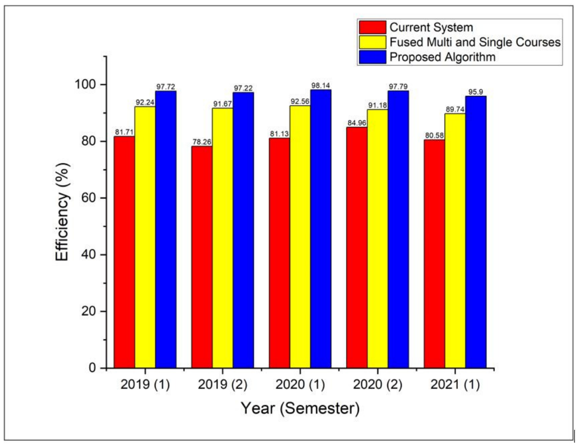


Fig. 4. Efficiency of the Proposed Algorithm.

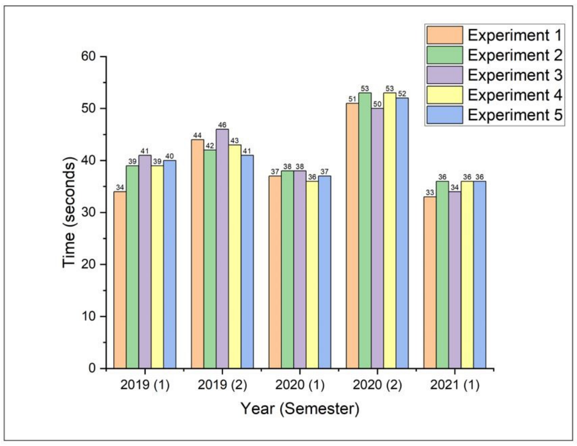


Fig. 5. Execution Time of Proposed Algorithm.

set, in each experiment, the algorithm does not follow any sequence in scheduling, thus for N number of experiments, there would be discrete execution times. It can also be noted that, it takes more time to execute data sets with larger number of records compared to those with smaller numbers of records.

6. CONCLUSION AND FUTURE WORK

In this research a random scheduling technique was used to schedule the University of Cape Coast Examination Timetable taking into account, records of undergraduate courses only. The technique first uses the machine to accept the data sets and randomly schedules the unique courses, then human experts are used on the courses that the algorithm did not schedule. The proposed algorithm on the average, randomly schedules 97.35% of the unique courses. While 2.65% of the unique courses are scheduled manually. Meanwhile, the current system on the average automates 81.33% while 18.67% of the unique courses are manually input. Implying the proposed algorithm significantly improved the efficiency of the old and current system by about 16%.The execution times of the algorithm were also captured in the experiments. To completely solve the full-scale problem, further algorithm improvements will have to be made which are presently being worked on. Major improvements of the user interfaces will also be done.

ACKNOWLEDGMENT

Thank you Professor Moses Jojo Eghan, for providing insightful comments that helped improve the paper. We appreciate the confidence reposed in us to write this scientific paper for the body of knowledge and particularly for the University of Cape Coast. Our gratitude is boundless.

7. REFERENCES

[1] Burke, E., Kingston, J., Jackson, K., Weare, R., "Automated university timetabling: The state of the art", The Computer Journal 40 (9), pp. 565–571 (1997).

[2] Burke, E., Petrovic, E.K., "Recent research directions in automated timetabling", European Journal of Operational Re-

search 140, pp 266–280 (2002)

[3] Laporte, G., Desroches, S., “Examination Timetabling By Computer”, Computer and Operation Research 11 (4), pp 351-360, 1984.

[4] Qu, R., Burke, E.K. and McCollum, B., 2009. Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems. European Journal of Operational Research, 198(2), pp.392-404.

[5] Abayomi-Alli, O., Abayomi-Alli, A., Misra, S., Damasevicius, R. and Maskeliunas, R., 2019. Automatic examination timetable scheduling using particle swarm optimization and local search algorithm. In Data, Engineering and Applications (pp. 119-130). Springer, Singapore.

[6] Abdul-Rahman, S., Sobri, N.S., Omar, M.F., Benjamin, A.M. and Ramli, R., 2014, December. Graph coloring heuristics for solving examination timetabling problem at Universiti Utara Malaysia. In AIP Conference Proceedings (Vol. 1635, No. 1, pp. 491-496). American Institute of Physics.

[7] Mauritsius, T., Legowo, N. and Gunawan, F.E., 2018, September. Reducing the Timeslot Used in Examination Timetable Problem. In 2018 International Conference on Information Management and Technology (ICIMTech) (pp. 211-216). IEEE.

[8] Chu, S.C., Chen, Y.T. and Ho, J.H., 2006, August. Timetable scheduling using particle swarm optimization. In First International Conference on Innovative Computing, Information and Control-Volume I (ICICIC’06) (Vol. 3, pp. 324-327). IEEE.

[9] Boufflet, J.P. and Negre, S., 1995, August. Three methods used to solve an examination timetable problem. In International Conference on the Practice and Theory of Automated Timetabling (pp. 325-344). Springer, Berlin, Heidelberg.

[10] Abdul Rahman, S., Syed Abdullah, S.S. and Benjamin, A.M., 2017. A nonlinear heuristic modifier for constructing examination timetable. Journal of Theoretical and Applied Information Technology, 95(20), pp.5642-5653.

[11] Saat, E.H.M., Ilham, N.I., Othman, N., Bakar, Z.A., Yusof, Y. and Abd Rahman, N.H., 2019, August. The examination timetabling problem based on expert system: a case study in malaysia. In 2019 IEEE 10th Control and System Graduate Research Colloquium (ICSGRC) (pp. 121-126). IEEE.

[12] Mandal, A.K. and Kahar, M.N.M., 2020. Performance Analyses of Graph Heuristics and Selected Trajectory Metaheuristics on Examination Timetable Problem. Indonesian Journal of Electrical Engineering and Informatics (IJEEI), 8(1), pp.163-177.

[13] Kahar, M.N., Bakar, S.A., Shing, L.C. and Mandal, A.K., 2018. Solving kolej poly-tech mara examination timetabling problem. Advanced Science Letters, 24(10), pp.7577-7581.

[14] Samaya, R., 2018. Examination timetable system (Doctoral dissertation, BUSE).

[15] Aminu, A., Caesarendra, W., Haruna, U.S., Sani, A., Sa’id, M., Pamungkas, D.S., Kurniawan, S.R. and Kurniawan, E., 2019, October. Design and Implementation of An Automatic Examination Timetable Generation and Invigilation Scheduling System Using Genetic Algorithm. In 2019 2nd International Conference on Applied Engineering (ICAE) (pp. 1-5). IEEE.

[16] Elsaka, T., 2017, September. Autonomous generation of conflict-free examination timetable using constraint satisfaction modelling. In 2017 International Artificial Intelligence and Data Processing Symposium (IDAP) (pp. 1-10). IEEE.