Lecture Timetable Scheduling Software

BY

Ebieto, C. E

Department of Mechanical Engineering,
University of Port Harcourt, Port Harcourt, Nigeria.
Email: celestine.ebieto@uniport.edu.ng; Phone: 08037728846

DOI: 10.29322/IJSRP.8.9.2018.p8155
http://dx.doi.org/10.29322/IJSRP.8.9.2018.p8155

ABSTRACT

In this paper, computer software is designed for optimum scheduling of timetable for courses in educational institutions. This system uses Dynamic Slot Algorithm (DSA) with the application of Constraint Satisfaction Programming. The algorithm was transformed into a program using Microsoft Excel-visual basic for application (Excel-VBA). It has been used for the timetable of the Faculty of Engineering, University of Port Harcourt. Results obtained are better than those generated manually by the human timetabler as they show a much higher space and time utilization, apart from the great speed associated with computer productions.

Key words: Excel Add-Ins, Computer software, Excel-VBA, Constraint Satisfaction Programming, Timetable.

INTRODUCTION

Timetable scheduling is the problem of assigning courses or examinations to periods and to classrooms. Although the course and examination timetables are related to each other, they can be quite different. For example, generally, it is possible to have more than one examination on in a large examination hall at any particular time, whereas it would be extremely unlikely that lectures (courses) on more than one course would take place in the same hall, no matter how large that hall is. This means that, practically, the examination timetable scheduling process must be carried out centrally by the institution. Therefore the examination timetabling problem is a subset of the course timetabling problem.

The automatic timetable construction is within the sphere of combinatorial search algorithms [1]. Particular techniques which have been used to deal with the search problems include integer programming [2]; genetic or evolutionary algorithms [3,4]; and simulated annealing [5]. However, each of these techniques plagued by a shortcomings ranging from performance unreliability, blurred search landscapes, and high resource utilization [6,7,8].

There are currently many different solution generation algorithms in existence. Some are so well known that they are the subject of standard textbook material [9,10]. Linear and integer programming techniques [11,9] the first applied to timetabling were developed from the broader area of mathematical programming. In the 1970's IP was identified as being less efficient than linear programming (LP) for solving

optimization problems but was still considered to be a good problem solving technique [12].

Evolutionary algorithms (EAs) are a class of direct, probabilistic search and optimization algorithms gleaned from the model of organic evolution. It is a problem solving and optimization method that uses genetics as its model problem and applies the rules of reproduction, general crossover and mutation to pseudo-organizations so that those organizations can pass beneficial and survival-enhancing traits to the new generation [3]. It is claimed that Genetic Algorithms have a greater capacity than any other search method in finding the largest number of possible solutions and depict the best possible results [4]. However, GAs still fall short in the choice of control parameters, the exact roles of crossover and mutation, and the characterization of search landscapes for optimization and convergence characteristics [7].

Constraint Satisfaction Programming (CSP) is a relatively new technique for dealing with search problems [13,14]. It involves searching for the values of problem variables subject to constraints on their feasible combinations. In some cases, the satisfaction of the constraints leads to conflicts with the problem objectives. Thus, only the solution of the sub-problem (by satisfying a subset of the constraints) is possible, if the problem objectives must be met; this is the case of Partial Constraint Satisfaction Programming (PCSP). PCSP with dynamic slot algorithm (DSA) is a particular application of CSP which is an algorithm where the process of finding solution to a problem is dependent on the strategy chosen. Therefore, it involves the theory of multistage decision process where the overall problem with *n* variables is sub-divided into *n* sub-problems, each sub-problems being a decision making problem in one variable. The optimal solutions of these sub-problems can now be used to find an optimal solution to the overall problem. The major advantages are as follows: (i) constraint propagation reduces the search space so it takes less time to search thus minimizing backtracking; (ii) memory requirement is smaller since the search space has been reduced; (iii) all available resources are represented in the form of constraints and hence user preferences and requirements can be easily satisfied.

This paper, therefore presents an Excel-VBA based computer software for scheduling lecture timetable in educational institutions by partial constraint satisfaction programming with dynamic slot algorithm. It is then applied for the special case of lecture timetabling of the Faculty of Engineering lectures at the University of Port Harcourt.

METHODOLOGY

The Timetabling Problem

The timetabling process consists of two distinct phases:

- The curricula, that is, lists of courses, are defined for each class and various resources (manpower and/or equipment) are assigned to them.
- A detailed timetable is composed which is compatible with the previously defined requirements and with some additional constraints as well.

The second phase is the one which is usually referred to as the *timetabling problem*. Due to the complexity of this problem, it is worthwhile trying to solve it by a computer instead of tackling it manually.

The timetabling problem involves scheduling some relations that can be obtained between finite specific resources such as lecturers, students, classrooms, hours, and courses with the respect to some *hard* and *soft* constraints. The main differences between the *hard* and *soft* constraints is that Hard constraints are those constraints we cannot violate, while Soft constraints are constraints that we can pass sometimes but it is preferable not to do so and every time we violate them, it reflect in our result.

Some examples of hard constraints are:

- A lecturer can only teach in a single place at a time.
- A lecturer can only give one lecture at a time.
- A venue can only host one lecture at a time.
- A student can only attend one lecture at a time.
- Class room capacity must be respected.
- Not more than one lecturer is scheduled to hold in a classroom at a time.
- Each course is scheduled in a proper room (for example, a laboratory course needs proper laboratory).

Some examples of soft constraints are:

- A lecturer should not teach more than 6 hours a day.
- A student should not have more than 8 hours a day.
- There should not be unnecessary gaps in the activity of the lecturers.
- There should not be unnecessary gaps in the activity of the students.
- The courses should be scheduled in the morning and laboratories in the after-noon.
- Some lectures are scheduled *a priori.*
- As much as possible the preferences of the lecturers and the ones of the students should be respected.
- Launch break should be respected.

LTSS – A Timetable Construction System

LTSS (Lecture Timetable Scheduling Software) uses a Dynamic Slot Algorithm (DSA) which is a method for reducing the runtime of algorithms exhibiting the properties of overlapping sub-problems (that is the same sub-problems are used to solve many different larger problems) and optimal substructures (that is, the optimal solutions of sub-problems can be used to find the optimal solutions of the overall problem). DSA also apply memoization which involve saving the solution to problems that have already been solved so that if we need to solve the same problem later, we can retrieve and reuse our already computed solution. A number of assumptions and constraints have been taken into consideration to meet the requirement of the Faculty of Engineering University of Port Harcourt.

More specifically, four main assumptions have been considered:

- Each course is split into two or more lecture time depending on its credit unit
- Each course can be lecture, practical or both; lecture Courses are taught in class rooms, whereas practical courses are performed in laboratories or workshops. Each course (lecture or practical) assumed to have a fixed class size.
- Each course belongs to a specific level of study (year) and falls under one out of five separate levels (year 1, year 2, year 3, year 4, and year 5); and the courses in each year are further classified as General, Interfaculty, Faculty, or Departmental in that order of priority.
- General courses are done by only year 1 students; and all Departments take the same courses in year 1.

Apart from these conditions, five main constraints have been taken into consideration:

- No two lectures can be scheduled the same day and period of time in the same venue.
- No two courses belonging to the same level of study should overlap if they are for the same Department or the entire Faculty.
- No two lectures should be given by the same lecturer at the same time. Moreover no lecturer should be assigned a lecture immediately after a lecture if the two lectures are at different campuses.

- The lecture venues allocated for a particular course should be large enough to fit the student class size.
- If a faculty course has been scheduled in a particular time, no other departmental course of the same level should be scheduled at that time.

Three basic criteria have been set in LTSS, which of course can be easily expanded and improved:

- In other to achieve the best utilization of each venue, it is desirable that each course is scheduled in a venue with enough capacity to accommodate the number of students attending this course.
- It is desirable that all courses belonging to the same level, if scheduled in the same day, should be taught in periods of time with minimum gaps between them as possible.

Solution Generation

The problem depicts each department as offering several courses per semester and each course having a specific period/duration in hours per week (corresponding to the credit units). Each of the specified periods can thus be defined as the contact hours between lecturers and students from a specified time and at a particular venue.

The timetabling problem can thus be defined as an assignment of time t_j, $1 \le j \le m$ and venues v_k, $1 \le k \le p$ to courses $S(i)$, $1 \le i \le n$ taught by lecturers $L[S(i)]$ such that all constraints $C[S(i)]$ are satisfied; where $L[S(i)]$ and $C[S(i)]$ are lecturers of and constraints on courses, respectively; m, p, and n are integer variables. Thus, generally, the CSP for a timetabling problem is as follows:

1. A finite set of variables, X_1-------X_n
2. For each variable X_i a set of domains D_1--------D_n containing possible values of X_i.
3. A finite set of constraints, C_1-------- C_q representing relations between variables.

A solution to the constraint satisfaction programming (CSP) involves assigning values from domains of all variables such that all constraints are satisfied.

These values for the domains are generated using a Random function as shown below:

$$X = ((v * rnd) + 0) \quad (1)$$

where:

v is the number of venues and the days available

rnd is the generated random number

The following function was constructed for this work to test the value of the timetable. The value of a timetable $V(t)$ is given by:

$$V(t) = N - P - Q \quad (2)$$

where:

N is the number of classes assigned to the available venues (excluding the unconstraint venues).

P is the number of classes with conflict.

Q is the number of classes that could not be schedule.

This function, although discrete, will have a maximum value equal to C, the number of lecture scheduling that need to be made. Its minimum value must necessarily be greater than -C. when the function $V(t)$ is applied to the timetable scheduling; the value of the function continually drops as C increases.

The implementation described in the work, being that we are using a constraint satisfaction programming algorithm, never generated any value for P (by design, no classes will be schedule that will result to a clash). However, as the number of classes to schedule increased, the algorithm took longer time to generate solution.

Finally, it was interesting to note that even though large values of C often resulted in the algorithm not finding a complete solution, the algorithm was always able to schedule more

than 50% of the available courses (at 50%, $V(t)=0$).

Implementation

The software LTSS (Lecture Timetable Scheduling Software) is quite flexible and it provides the user with a windows environment interface which is designed so that it can be easily handled by a non-expert. The user create the database for courses, lecture venues and laboratories that will be included in the timetable or create his/her own data by modifying (i.e. adding, removing or changing) some of the data's that is already in the database. He/she is also required to set the constraints concerning the availability of lecture venues (i.e. define the days and periods of time in these days in which they are available) if applicable to any of the venues while venues that are not constraints are given all the days of the week. As soon as the creating of the database stage has been completed, you will be required to choose if there is any preferential course you will want to schedule (i.e. these are General or Inter-Faculty courses with a fixed time and days of the week. These courses can not be altered).when you are true if any preferential course, the main process starts which includes the following:

- You select a particular Department (according to priority) to schedule which automatically displays the courses offered by that department then you select a course to schedule and enter the class size.
- The system checks with the available venues for that department, the venues that meet that class size and select a venue from the list and also checks if that selected venue is a constraint venue or not.
- The program traverse through the search spaces via recursion with backtracking upon constraint failure until there is an optimum solution. However, if it cannot find a solution due to lack of venue, the program terminates notifying the user that there is no venue for that course.

When all these steps have been successfully completed for all unscheduled courses, the solution is displayed on an Excel spreadsheet and printed on a printer device.

RESULTS AND DISCUSSION

For a timetable scheduling software to be complete, a flexible user interface should be provided, so that the specific requirement of the problem can be stated (courses, venue available, etc.). LTSS provides such an interface.
The LTSS interface has been developed using Microsoft excel-visual basic for application (Excel-VBA). The interface allows the user to define the software parameters as preferred. Not only does it make it possible for the user to enter the particular courses, etc. that are to be Scheduled, it also enables the dynamic slotting of the course.
After successful installation of the software as an Excel-Add-Ins, click on the Time-Table menu on the menu bar and then click on the Start button of the Time-Table menu as shown in figure 1.

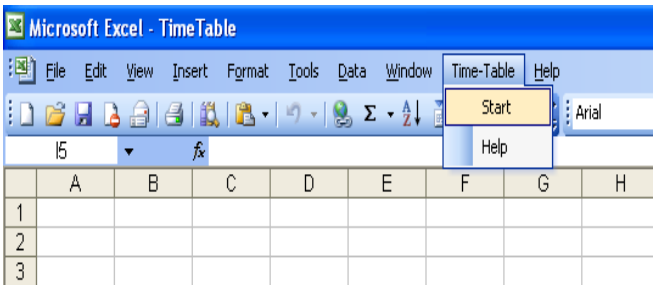

Figure 1: Excel worksheet showing the start drop menu

By clicking on the start button of the Time-Table menu we see the window of figure 2. click proceed to move to the next window (figure 3) where you will be required to enter the session, semester, and faculty (this is because we intend to expand the scope to the university at large but for this case you click on Engineering) you are scheduling the timetable for. You first click on Database (figure 3) to build your database (figure 4) if there is no existing database or if you want to make any changes to an existing database. When through with building the database, you click previous to return to the formal environment (figure 2) and click Build timetable to move to the final stage (figure 5) where you carry out the scheduling. However if there are preferential courses to schedule, you can either move from figure 2, depending on how you responded to the questions on clicking Build Timetable or you can move from figure 4 by clicking Go to preference on the environment to get to the preference Environment (figure 6).

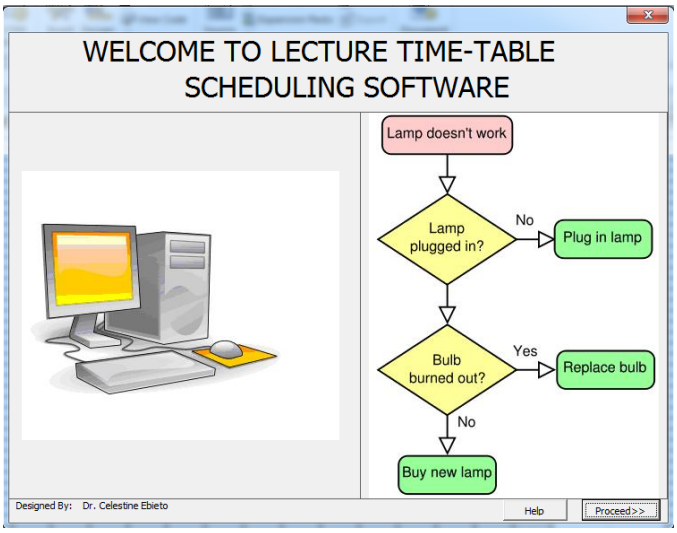

Figure 2: LTSS Welcome Interface

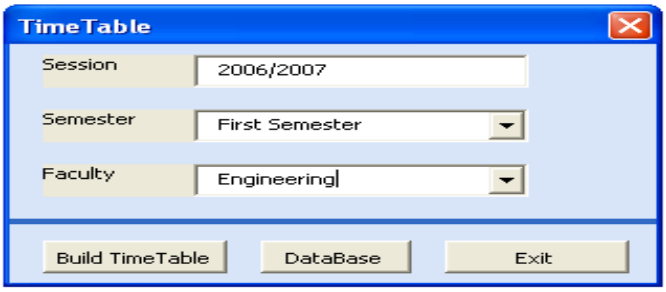

Figure 3: LTSS Initializing Interface

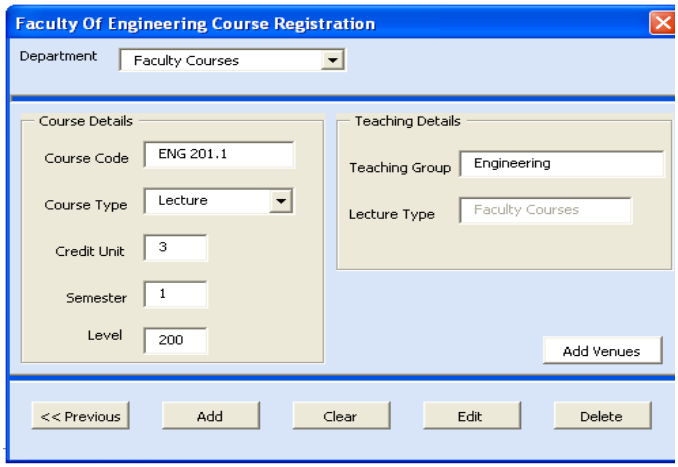

Figure 4: Database Interface

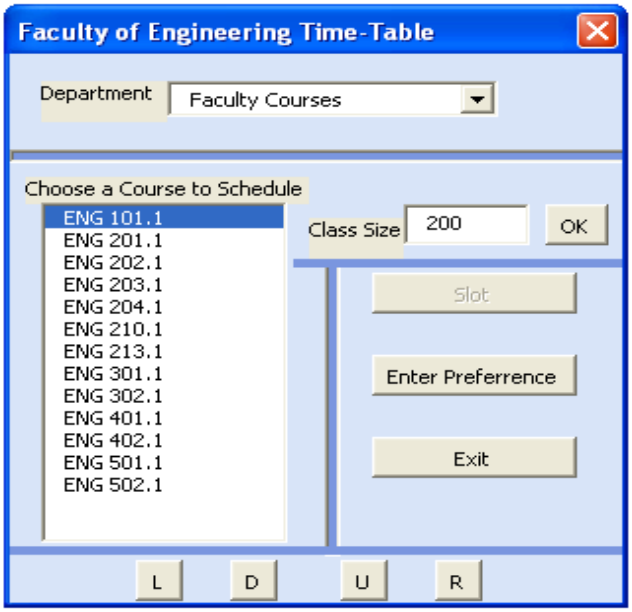

Figure 5: LTSS Interface

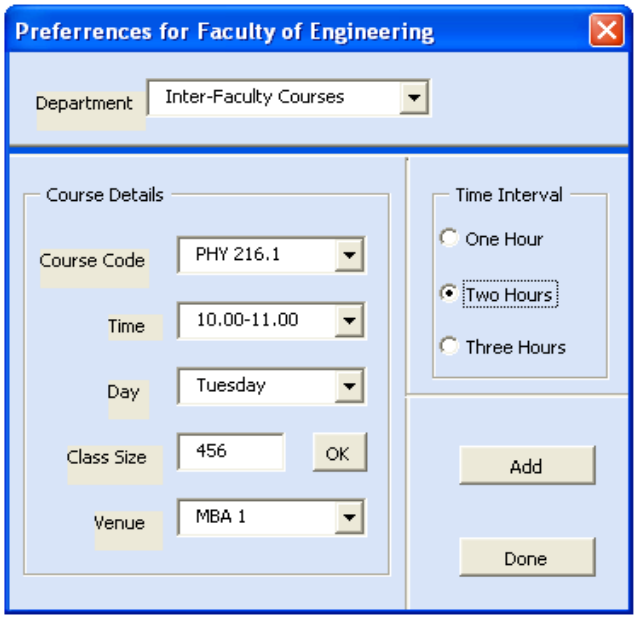

Figure 6: Preference Interface

Data set from the Faculty of Engineering University of Port Harcourt was used for scheduling the Timetable. With regard to the performance of the software at run-time, LTSS has been tested and has yielded most satisfactory results, not only at respecting the user preferences that had been entered but also at reaching an optimum solution to the specific timetabling problem. Figure 7 shows an example timetable generated for 2006/2007 academic session.

The constraint satisfaction programming (CSP) approach adopted for the development of LTSS is a programming framework that fits perfectly with the timetabling problem faced by the Faculty of Engineering University of Port Harcourt, as well as with any other timetabling problems. The software is quite flexible both from the implementation and the user points of view. The flexibility in the implementation, which is actually based on the declarative style of programming supported by CSP, offers the possibility to easily add more constraints, which might be very different in nature from the existing once. All the information that describes the data for the specific problem is user defined (or may be taken from an existing database).

UNIVERSITY OF PORTHARCOURT

FACULTY OF ENGINEERING

FIRST SEMESTER TIME TABLE FOR 2006/2007 SESSION

DAY	8.00-9.00		9.00-10.00		10.00-11.00		11.00-12.00		12.00-1.00		1.00-2.00		2.00-3.00		3.00-4.00		4.00-5.00		5.00-6.00	
	C-CODE	VENUE	C-CODE	VENUE	C-CODE	VENUE	C-CODE	VENUE	C-CODE	VENUE	C-CODE	VENUE	C-CODE	VENUE	C-CODE	VENUE	C-CODE	VENUE	C-CODE	VENUE
	GES 100.1	PS HALL	GES 100.1	PS HALL	ENG 213.1	ULCH 1	ENG 101.1	EDS	ENG 401.1	ULCH 1	ENG 401.1	ULCH 1	CHE 313.1	ULCH 1	CHE 413.1	ULCH 1	ENG 502.1	LH II	ENG 203.1	LH II
M	CHE 511.1	ULCH 1	ENG 202.1	ETF	ENG 213.1	EDS	ENG 101.1	ETF	ENG 401.1	EDS	ENG 401.1	EDS	MEG 307.1	EDS	MEG 405.1	EDS	ENG 502.1	EDS	ENG 203.1	EDS
O	MEG 503.1	EDS	ENG 202.1	EDS	CHE 311.1	RM B3	CHE 513.1	ULCH 1	MEG 551.1	MEG LAB	CEG 381.1	CEG LAB	CEG 446.1	HYDR	EEE 302.1	RM B2	CEG 461.1	PLE	CHE 515.1	ULCH 1
N	CEG 562.1	RM A5	CEG 446.1	RM A5	CEG 323.1	RM B4	EEE 401.1	RM B2	CEG 547.1	RM A4	PNG 301.1	PTDF-N	EEE 401.1	RM B3	PNG 501.1	PTDF-N	EEE 304.1	ULCH 1	EEE 402.1	RM B1
	EEE 404.1	RM B3	EEE 501.1	ULCH 1	PNG 301.1	PTDF-A	EVE 401.1	RM B3	PNG 302.1	PTDF-W	CHE 315.1	RM B3	PNG 303.1	PTDF-LAB	CEG 461.1	PLE	CHE 315.1	RM A4	PNG 501.1	PTDF-A
	PNG 502.1	PTDF-A	CHE 415.1	RM B3	EEE 304.1	RM B1	PNG 502.1	PTDF-W	CHE 517.1	RM B3	EEE 506.1	RM B1	GNG 501.1	PTDF-N			PNG 403.1	PTDF-W	GNG 502.1	PTDF-N
	MTH 110.1	MBA 1	GNG 402.1	ULCH 1	MEG 307.1	EDS	CHM 130.1	MBA 1	ENG 101.1	EDS	ENG 301.1	ULCH 1	ENG 201.1	LH II	FSB 101.1	MBS 13	EEE 303.1	EDS	CHE 313.1	RM B3
T	CHE 421.1	CHE LAB	MEG 307.1	EDS	CEG 443.1	HYDR	ENG 202.1	ETF	ENG 101.1	ULCH 1	ENG 301.1	EDS	ENG 201.1	EDS	MEG 309.1	ULCH 1	CEG 514.1	HYDR	MEG 407.1	ULCH 1
U	MEG 451.1	MEG LAB	CEG 548.1	HYDR	EEE 305.1	EEE LAB	ENG 202.1	EDS	EVE 501.1	HYDR	CEG 444.1	RM B1	MEG 507.1	ULCH 1	EEE 303.1	EDS	EVE 502.1	PLE	CEG 321.1	RM B4
E	EEE 402.1	EDS	EEE 501.1	RM A5	PNG 302.1	ULCH 1	CEG 443.1	RM A4	EEE 405.1	RM A5	EEE 502.1	RM A5	EEE 406.1	EEE LAB	PNG 303.1	PTDF-LAB	GNG 401.1	ULCH 1	EEE 303.1	RM A4
	GNG 402.1	ULCH 1	PNG 405.1	PTDF-LAB	CHE 317.1	RM B3	EEE 503.1	RM A5	CHE 417.1	RM B3	CHE 515.1	RM B3	CHE 417.1	RM B3	CHE 517.1	RM B3	CHE 317.1	RM B3	PNG 503.1	PTDF-N
	PNG 503.1	PTDF-A	CHE 517.1	RM B3			PNG 402.1	ULCH 1	MEG 403.1	LH II	MEG 561.1	LH II	PNG 504.1	PTDF-N			MEG 507.1	LH II		
	GES 102.1	PS HALL	MEG 311.1	ULCH 1	ENG 501.1	ULCH 1	ENG 201.1	MBA 1	MTH 120.1	MBA 2	MTH 120.1	MBA 2	CHE 211.1	ULCH 1						
W	ENG 213.1	LH II	ENG 401.1	LH II	ENG 501.1	EDS	ENG 201.1	EDS	CHE 421.1	CHE LAB	MEG 303.1	ULCH 1	MEG 403.1	EDS						
E	ENG 213.1	EDS	ENG 401.1	EDS	CEG 321.1	PLE	CHE 411.1	RM B1	MEG 309.1	ULCH 1	CEG 534.1	HYDR	EEE 406.1	EEE LAB	SPORT		SPORT		SPORT	
D	MEG 311.1	ULCH 1	CEG 321.1	RM A4			MEG 401.1	ULCH 1	CEG 351.1	RM B4	EEE 403.1	EDS	CEG 547.1	HYDR						
	CEG 413.1	RM A4	EVE 501.1	RM B4			CEG 444.1	RM A4	EEE 404.1	EDS	GNG 401.1	PTDF-A	GNG 501.1	PTDF-A						
	GNG 402.1	PTDF-A	GNG 502.1	PTDF-A	GNG 403.1	PTDF-LAB	GNG 503.1	PTDF-A	PNG 401.1	PTDF-W	PNG 501.1	PTDF-A/N	PNG 401.1	PTDF-A/N						
	ENG 203.1	ULCH 1	ENG 203.1	ULCH 1	PHY 216.1	MBA 1	PHY 216.1	MBA 1	ENG 502.1	ULCH 1	MEG 301.1	EDS	MEG 301.1	EDS	MEG 401.1	EDS	MEG 305.1	EDS	MEG 305.1	EDS
T	ENG 203.1	EDS	ENG 203.1	EDS	ENG 301.1	ETF	ENG 301.1	ETF	ENG 502.1	EDS	CEG 332.1	PLE	CEG 413.1	PLE	CEG 351.1	PLE	CEG 483.1	CEG LAB	CEG 352.1	PLE
H	MEG 451.1	MEG LAB	CEG 534.1	HYDR	ENG 301.1	EDS	ENG 301.1	EDS	CEG 352.1	HYDR	EEE 305.1	EEE LAB	EEE 403.1	RM B4	EEE 501.1	RM B4	EEE 404.1	RM B4	EEE 502.1	RM B4
U	CEG 548.1	HYDR	EEE 405.1	RM B4	CEG 514.1	RM B3	CEG 461.1	RM B4	EEE 406.1	EEE LAB	EVE 401.1	HYDR	EVE 504.1	HYDR	GNG 401.1	PTDF-A/N	PNG 301.1	PTDF-A/N	EVE 501.1	RM A5
	EEE 503.1	RM B4	EVE 502.1	PLE	EEE 504.1	RM B4	PNG 510.1	PTDF-A/N	EVE 401.1	RM B4	GNG 403.1	PTDF-LAB	PNG 302.1	PTDF-A/N	PNG 405.1	PTDF-LAB			PNG 401.1	PTDF-A/N
	PNG 402.1	PTDF-A/N	PNG 510.1	PTDF-W	PNG 403.1	PTDF-A/N			PNG 404.1	PTDF-A/N	PNG 510.1	PTDF-N								
	ENG 201.1	LH II	ENG 302.1	ULCH 1	PHY 101.1	MBA 2	PHY 101.1	MBA 2	ENG 204.1	ULCH 1	ENG 204.1	ULCH 1	CHM 250.1	MBA 2	CHM 250.1	MBA 2	ENG 402.1	ULCH 1	ENG 402.1	ULCH 1
F	ENG 201.1	EDS	ENG 302.1	EDS	ENG 210.1	ETF	ENG 210.1	ETF	ENG 204.1	EDS	ENG 204.1	EDS	CHE 421.1	CHE LAB	ENG 501.1	LH II	ENG 402.1	EDS	ENG 402.1	EDS
R	CHE 511.1	ULCH 1	CEG 443.1	PLE	ENG 210.1	EDS	ENG 210.1	EDS	CEG 332.1	RM A4	CEG 445.1	PLE	MEG 305.1	ULCH 1	ENG 501.1	EDS	CEG 562.1	HYDR	CHE 513.1	RM A4
I	CEG 445.1	RM A1	CHE 55X.1	RM B3	MEG 503.1	ULCH 1	CHE 411.1	LH II	EEE 305.1	EEE LAB	EVE 502.1	HYDR	CEG 483.1	CEG LAB	EEE 304.1	ULCH 1	CHE 317.1	RM A4	CEG 591.1	HYDR
	EVE 503.1	PLE	PNG 503.1	PTDF-N	CEG 591.1	RM B4	MEG 505.1	ULCH 1	EVE 503.1	RM A5	PNG 515.1	PTDF-N	EEE 302.1	EDS			MEG 5XX.1	LH II	EEE 502.1	RM A2
			MEG 5XX.1	LH II	CHE 55X.1	RM A4			CHE 55X.1	LH II	MEG 5XX.1	LH II								
	CHE 311.1	ULCH 1	CHE 411.1	ULCH 1	CHE 315.1	RM B1	ENG 302.1	PS HALL	ENG 210.1	PS HALL	CHE 313.1	RM A1	MEG 303.1	EDS	MEG 405.1	EDS	CHE 311.1	ULCH 1	MEG 311.1	EDS
S	MEG 407.1	EDS	MEG 505.1	EDS	MEG 251.1	EDS	ENG 302.1	EDS	ENG 210.1	EDS	MEG 453.1	MEG LAB	EEE 302.1	RM A4	EEE 401.1	RM A1	MEG 507.1	EDS	EEE 301.1	ULCH 1
A	EEE 405.1	RM A4	CEG 323.1	RM B1	CEG 562.1	RM A1	CHE 413.1	ULCH 1	CHE 513.1	ULCH 1	EEE 301.1	EDS	CHE 415.1	ULCH 1	GNG 501.1	PTDF-W	EEE 301.1	RM A1	CHE 515.1	RM A2
T	GNG 502.1	PTDF-A	EEE 503.1	RM A2	PNG 504.1	PTDF-A/N	MEG 407.1	LH II	MEG 551.1	MEG LAB	PNG 403.1	ULCH 1	PNG 505.1	PTDF-N	CHE 417.1	ULCH 1	PNG 402.1	PTDF-A	PNG 515.1	PTDF-A
	PNG 404.1	PTDF-W	PNG 515.1	PTDF-A/N	EEE 504.1	ULCH 1			EEE 505.1	RM B1										
							EEE 505.1	RM B1												

Figure 7: An example result generated with the software

CONCLUSIONS

This paper has examined the timetable scheduling problem and a specific system called LTSS, which constructs optimum timetables for university courses, is presented. The timetabling problem being a constraint satisfaction one is mapped naturally to the constraints provided by the faculty of Engineering. moreover, the modeling of the problem under consideration profits a lot from the declarative style of programming which is supported by Excel Visual basic for application
The performance of LTSS is quite satisfactory, considering that it is a program that has to run once or twice a year for the construction of the timetable of an educational organization. Some improvements which are currently under development will certainly enhance the system's functionality. On a positive final note, there would appear to be almost no training issues to be addressed.

ACKNOWLEDGMENT

My sincere gratitude goes to my undergraduate project supervisor, Late Prof. C.O.C. Oko for his support and guidance during this work.

REFERENCES

[1] de Werra, D. (1985), *An Introduction to Timetabling*. European Journal of Operational Research, 19:151-162.

[2] Makower, M. S. and Williamson, E. (1975), *Operational Research – Teach Yourself Book*

[3] Chambers L. (1995), Practical Handbook of Genetic Algorithms: Applications

[4] Maxfield, C. (1997), *Genetic Algorithms: programs that boggle the mind*. EDN.

[5] Abramson, D. (1991), *Constructing School Timetable Using Simulated Annealing*, Sequential and Parallel Algorithms Management-Science.

[6] Taha, T. R. (1987), *Numerical schemes for nonlinear evolution equations*, The College Journal of Science and Technology (Jerusalem).

[7] Srinivas, M. and Patnik, L. M. (1994), *Genetic Algoriths: a survey*. Dordrecht Kluwer Academic Publishers.

[8] Dikmann, R., Luling, R. and Simon, J. (1993), *Problem independent distributed simulated annealing and its applications.* Technical Report – Paderborn Center for Parallel Computing.

[9] Schaerf, A. (1996), *Tabu search techniques for large high-school timetabling problems*. Proceedings of the Thirteenth national Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference. MIT Press.

[10] Osman, I. H. and Kelly, J. P. (1996), Meta-Heuristics: *Theory and Applications*. Kluwer Academic Publishers.

[11] Wren, A. (1996), *Scheduling, timetabling and rostering - a special relationship? Practice and Theory of Automated Timetabling*. First International Conference. Selected Papers. Springer-Verlag, Berlin, Germany.

[12] Zoints, S. (1974), *Linear and Integer Programming*. Prentice-Hall Inc.

[13] Tsang, E. P. K., Mills, P., Williams, R., Ford, J. and Borrett, J. (1999), *A computer aided constraint programming system*, The First International Conference on The practical Application of Constraint Technologies and Logic Programming (PACLP), London.

[14] Deris, S., Omatu, S., Ohta, H. and Abd-Samai, P. (1997), *Object-oriented constraint logic programming for timetable planning*, International Journal of Systems Science.