Jurnal Ilmiah Widya Teknik
Volume 15 Nomor 2 2016
ISSN 1412-7350

TIMETABLE MANAGEMENT USING GENETIC ALGORITHMS

Limbran Sampebatu[1)], Aries Kamolan[2)]
Fakultas Teknik, Universitas Atma Jaya Makassar
[1)] elsampebatu@gmail.com, [2)] arieskamolan@yahoo.com

ABSTRACT

Scheduling course timetables for a large array of courses is a very complex problem which often has to be solved manually by the center staff even though results are not always fully optimal. Timetabling being a highly constrained combinatorial problem, this work attempts to put into play the effectiveness of evolutionary techniques based on Darwin's theories to solve the timetabling problem if not fully optimal but near optimal.

Genetic Algorithm is a popular meta-heuristic that has been successfully applied to many hard combinatorial optimization problems which includes timetabling and scheduling problems. In this work, the course sets, halls and time allocations are represented by a multidimensional array on which a local search is performed and a combination of the direct representation of the timetable with heuristic crossover is made to ensure that fundamental constraints are not violated.

Finally, the genetic algorithm was applied in the development of a viable timetabling system which was tested to demonstrate the variety of possible timetables that can be generated based on user specified constraint and requirements.

Keywords: Time table management, genetic algorithms

I. Introduction

Timetabling concerns all activities with regard to making a timetable that must be subjective to different constraints. According to Collins Concise Dictionary (4th Edition) "a timetable is a table of events arranged according to the time when they take place."

A critical factor in running a university or essentially an academic environment is the need for a well-planned and clash-free timetable. Back in the times when technology was not in wide use, academic timetables were manually created by the educational center staff.

Every school year, institutions of education face the rigorous task of drawing up timetables that satisfies the various courses and their respective examinations being offered by the different department. The difficulty is due to the great complexity of the construction of timetables for lectures and exams, due to the scheduling size of the lectures and examinations periods and the high number of constraints and criteria of allocation, usually circumvented with the use of little strict heuristics, based on solutions from previous years (Jose, 2008).

Nowadays, this process has been simplified by semi-automatic solutions based on timetable generation applications (e.g. Open Course Timetabler). A timetable management system is designed and created to handle as much course data as fed while ensuring the avoidance of redundancy. An educational timetable must meet a number of requirements and should satisfy the desires of all entities involved simultaneously as well as possible. The timing of events must be such that nobody has more than one event at the same time (Robertus, 2002).

The proposed timetabling system is designed to handle events of course lectures offered at a university (university course timetabling). Based on the above event, the system would have only one module which is the Course Lecture Timetable Module.

II. Literature Review

Solutions to timetabling problems have been proposed since the 1980s. Research in this area is still active as there are several recent related papers in operational research and artificial intelligence journals. This indicates that there are many problems in timetabling that need to be solved in view of the availability of more powerful computing facilities and advancement of information technology (Deris et.al, 1997). Since 1995, a large amount of timetabling research has been presented in the series of international conferences on Practice and Theory of Automated Timetabling (PATAT).

Papers on this research have been published in conference proceedings, see e.g., (Burke & Carter, 1997) and (Burke & Erben, 2000), and three volumes of selected papers in the Lecture Notes in Computer Science series, see (Burke & Ross, 1996), (Burke & Carter, 1998), and (Burke & Erben, 2001). Additionally, there is a EURO working group on automated timetabling (EURO-WATT) which meets once a year regularly sends out a digest via e-mail, and maintains a website with relevant information on timetabling problems, e.g., a bibliography and several benchmarks. There are two main problems in timetabling. The first one is related to the combinatorial nature of the problems, where it is difficult to find an optimal solution because it is impossible to enumerate all nodes in such a

large search space. The second one is related to the dynamic nature of the problems where variables and constraints are changing in accordance with the development of an organization (S.B. et. al., 1997).

Therefore, a timetabling system must be flexible, adaptable and portable, otherwise the users will not use the system optimally or even as decision aids such as for storing, retrieving, and printing timetables, when the timetable planning decisions are made manually. In addition, most of the universities adopting a semester system give freedom to students to choose subjects provided that all pre-requisites are satisfied. This situation further complicates the construction of a timetable.

Various techniques have been proposed to solve timetabling problems. These techniques are neural networks (Gianoglio, 1990), heuristics (Wright, 1996), graph coloring, integer programming, genetic algorithms (Burke et. al, 1994; Paechter et. al., 1994), knowledge-based, and constraint logic programming (Lajos, 1995). The models formulated by some of these techniques cannot be easily reformulated or customized to support changes, hence the selection of the genetic algorithm for the implementation of this project.

Although any given solution to the timetabling problem can be verified quickly, there is no known efficient way to locate a solution in the first place; indeed, the most notable characteristic of NP-complete problems is that no fast solution to them is known. That is, the time required to solve the problem using any currently known algorithm increases very quickly as the size of the problem grows (Ossam, 2009).

When solving the timetabling problem, we are usually looking for some solution, which will be the best among others. The space of all feasible solutions (series of desired solutions with some more desirable than others) is called search space (also state space). Each point in the search space represents one feasible solution which can be "marked" by its value or fitness for the problem. The solution is usually one point in the search space (Ossam, 2009).

As a result of comparative fact finding and exhaustive study of existing systems, Genetic Algorithms have been the most prominently used in generating near-optimal solutions to timetabling problems, hence its usage in the implementation of this project.

The earliest instances of what might today be called genetic algorithms appeared in the late 1950s and early 1960s, programmed on computers by evolutionary biologists who were explicitly seeking to model aspects of natural evolution. It did not occur to any of them that this strategy might be more generally applicable to artificial problems, but that recognition was not long in coming: "Evolutionary computation was definitely in the air in the formative days of the electronic computer" (Mitchell, 1996). By 1962, researchers such as G.E.P. Box, G.J. Friedman, W.W. Bledsoe and H.J. Bremermann had all independently developed evolution-inspired algorithms for function optimization and machine learning, but their work attracted little follow-up. A more successful development in this area came in 1965, when Ingo Rechenberg, then of the Technical University of Berlin, introduced a technique he called evolution strategy, though it was more similar to hill-climbers than to genetic algorithms. In this technique, there was no population or crossover; one parent was mutated to produce one offspring, and the better of the two was kept and became the parent for the next round of mutation (Haupt et. al., 1998). Later versions introduced the idea of a population. Evolution strategies are still employed today by engineers and scientists, especially in Germany.

Having considered the basis for a genetic algorithm, the outline below highlights the applications of the proposed system in generating timetables (with Covenant University as case study). The timetabling problem is a combinatorial optimization problem (COP) and in order to find a very comprehensive mathematical framework to describe it (and also tackle its NP-hardness), hence the introduction of a highly abstract concept of heuristics (genetic algorithms). The basic property of the timetable problem is the attempt of the genetic algorithm to optimize a function over a discrete structure with many independent variables. The relation between the choices made in the discrete domain and the effects on the objective function value are usually complex and frequently not easy to trace. The unifying framework for COP's is the Constraint Satisfaction Problem (CSP) in conjunction with the optimization of an objective function (Kostuch, 2003). It is worthy of note that even though the timetabling problem is treated as an optimization problem, there is actually no fixed objective function, the function that exists is used as an arbitrary measure to check for optimized solutions and degree of constraints satisfaction (Abramson, 1992). Once the objectives and constraints are specified, genetic algorithms offer the ultimate scenarios of good timetable solutions through evolution processes even though the complexity of assignment is totally dependent on the number of instances and number of constraints.

Hence the algorithm considered for use in the proposed system is a scaled down version of the Hybrid Genetic algorithm for the construction of examination timetables developed for the University of Nottingham. The concept though developed for examination timetabling, can be adapted to fit the construction of course timetables. The genetic algorithm employed combines two heuristic algorithms, the first finding a non-conflicting set of exams and the second assigning the selected exam to rooms. The process is repeated until all exams have been scheduled without conflicts (Rupert, 1995).

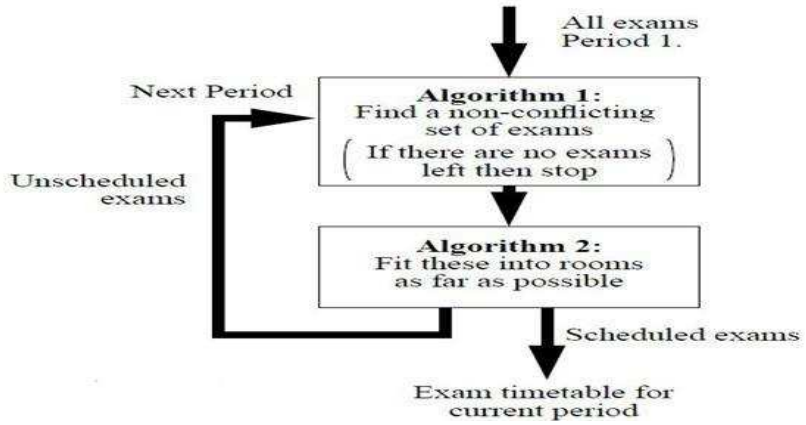

Figure 1. Diagram depicting the Hybrid Genetic Algorithm used in University of Nottingham (Adapted from Rupert Weare et. al. 1995).

Like every other genetic algorithms, this algorithm can quickly produce large populations of random feasible exam timetables. Uniquely, the process takes each member of the course population and assigns it to the first period in which the exam may be placed without conflict. The mutation and crossover procedures are then applied to the population so that constraints associated with each course in the assignment are satisfied. The timetables generated by the algorithm with a starting population size of 200 had an average fitness of 0.986 (Rupert Weare et. al., 1995).

III. Research Method

The proposed systems was developed to solve the timetabling problem being faced by universities every academic year and reduce high cost and slow turnaround involved in the generation of near-optimal timetables.
The system has capabilities for input of the various courses, halls of lectures, departments, programs, buildings, lecturers and the specification of a few constraints from which the timetable is constructed. The proposed timetabling system for this project seeks to generate near optimal timetables using the principles of genetic algorithm (selection and crossover).
A file is a collection of similar or related records As a result of this programs complexity, a file processing system is used rather than a database. The timetabling system uses 6 files consisting of:

a. Halls File: contains the name of the Halls/Rooms that are used in the course allocations within the program.
b. Programs File: contains the different programs in the university entered at input.
c. Buildings File: contains the names of the building existing in the school for which the timetable is to be generated.
d. Lecturers File: contains the names of the Lecturers in the school.
e. Departments File: contains all the department names in the university.
f. Course Codes File: contains all the course codes being offered in the school including their titles and units.

IV. Results

The system implementation defines the construction, installation, testing and delivery of the proposed system. After thorough analysis and design of the system, the system implementation incorporates all other development phases to produce a functional system.

The Delphi (Object Pascal) is the language of choice for this project because of its high speed and low memory usage. The timetabling problem is combinatorial in nature hence the need for a programming language that has enhanced CPU optimizing capabilities for the development of and algorithm like the genetic algorithm which optimizes search space and avoids local optima. Delphi has an intuitive Integrated Development Environment (IDE) for Rapid Application Development (RAD) and also cross-platform compatible.

The software implementation contains two major modules:

a. Crossover Module: has a functionality of simulating the crossover of course population whose constraints are violated. Crossing over individual courses in the population attempts to reduce and or eliminate constraints violations.
b. Random Population Generation Module: generates the initial random course population from the input supplied by user.

PROGRAM MODULES AND INTERFACE

a. BUILDING AND HALLS INPUT SECTION: This section of the timetable system form is the input module for the buildings and their respective halls/rooms with their capacities. This section also involves

the software reading the building names and hall names from external files rather than manual input and writing to an external file to save building input.

b. DEPARTMENT INPUT SECTION: This section of the timetable system form is the input module for the departments and their timing constraints. This section also involves the software reading the department names from external files rather than manual input and writing to an external file to save department input. See Figure 3.

Figure 2. Building and Halls input section

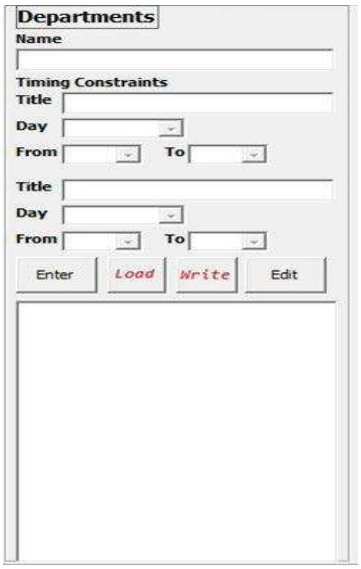

Figure 3. Departments input section

c. PROGRAM INPUT SECTION: This section of the timetable system form is the input module for the programs and their timing constraints. This section also involves the software reading the programs names from external files rather than manual input and writing to an external file to save input.

d. LECTURER INPUT SECTION: This section of the timetable system form is the input module for the lecturer and their timing constraints. This section also involves the software reading the lecturers names from external files rather than manual input and writing to an external file to save input.

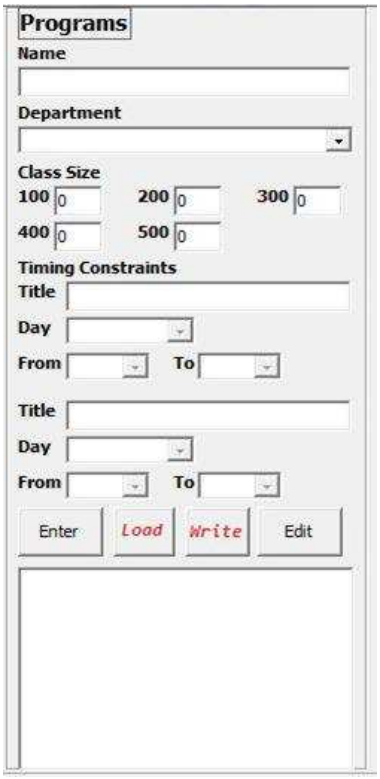

Figure 4. Programs input section

Figure 5. Lectures input section

e. LEVEL INPUT SECTION: This section of the timetable system form is the input module for specifying each predefined levels timing constraints. See Figure 6.

f. COURSE INPUT SECTION: This section of the timetable system form is the input module for the courses with their hall/room constraints and timing constraints. This section also involves the software reading the courses names from external files rather than manual input and writing to an external file to save input. See Figure 7.

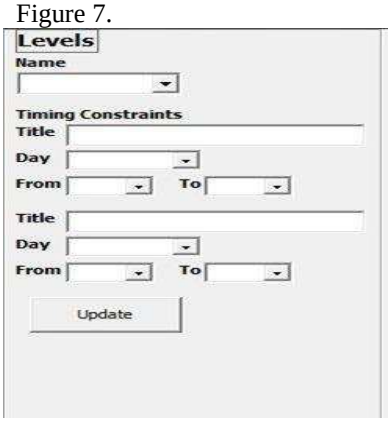

Figure 6. Level input section

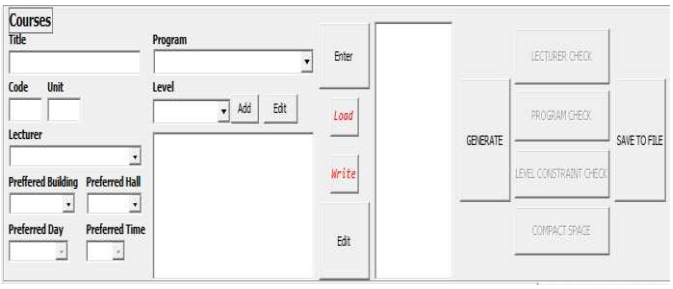

Figure 7. Course input section

g. REPORT SECTION: This section of the timetable system generates reports for timetable generate. It indicates the constraints which the generated timetable has violated. See Figure 8.

Figure7. Report input section

V. Conclusion

Timetabling problem being the hard combinatorial problem that it is would take more than just the application of only one principle. The timetabling problem may only be solved when the constraints and allocations are clearly defined and simplified thoroughly and more than one principle is applied to it i.e. a hybrid solution (a combination of different solution techniques).

This research has been able to actualize a sub-implementation of a genetic algorithm which can be applied to input of 2-units courses.

References

1. A. Cornelissen, M.J. Sprengers and B.Mader (2010). "OPUS-College Timetable Module Design Document" Journal of Computer Science 1(1), 1-7.
2. Abramson D. & Abela J. (1992). "A parallel genetic algorithm for solving the school timetabling problem." In Proceedings of the 15th Australian Computer Science Conference, Hobart, 1-11.
3. Adam Marczyk (2004). "Genetic Algorithms and Evolutionary Computation ". Available online at http://www.talkorigins.org/faqs/genalg/genalg.html.
4. Al-Attar A. (1994). White Paper: "A hybrid GA-heuristic search strategy." AI Expert, USA.
5. Alberto Colorni, Marco Dorigo, Vittorio Manniezzo (1992). "A Genetic Algorithm to Solve the Timetable Problem" Journal of Computational Optimization and Applications, 1, 90-92.
6. Bufe M., Fischer T., Gubbels H., Hacker C., Hasprich O., Scheibel C., Weicker K., Weicker N., Wenig M., & Wolfangel C. (2001). Automated solution of a highly constrained school timetabling problem - preliminary results. EvoWorkshops, Como-Italy.
7. Burke E, Elliman D and Weare R (1994)."A genetic algorithm for university timetabling system." Presented at the East-West Conference on Computer Technologies in Education, Crimea, Ukraine.
8. Carrasco M.P. & Pato M.V. (2001). "A multiobjective genetic algorithm for the class/teacher timetabling problem." In Proceedings of the Practice and Theory of Automated Timetabling (PATAT'00), Lecture Notes in Computer Science, Springer, 2079, 3-17.
9. Chan H. W. (1997). "School Timetabling Using Genetic Search." 2th International Conference on the Practice and Theory of Automated Timetabling, PATAT'97.
10. Coello Carlos (2000). "An updated survey of GA-based multiobjective optimization techniques." ACM Computing Surveys, 32(2), 109-143.
11. Costa D. (1994). "A tabu search algorithm for computing an operational timetable." European Journal of Operational Research, 76(1), 98-110.