Web-based Student Time-Table Management System

[1]Omoregbee Otaninyenuwa Helen, [2]Ihama E. I., [3]Izogie L.E. and Otamere Friday Mike

[1,2&3]Department of Computer Science and Information Technology,
School of Applied Sciences, Edo State Polytechnic, Usen, Benin City, Nigeria.
+2347039404855, email:eyoski@yahoo.com

ABSTRACT

The scheduling of lectures and practical timetables for a large number of courses is a very complex problem that is frequently managed manually by the center's employees, despite the fact that the outcomes are not always completely satisfactory. Because timetabling is a highly constrained combinational problem (i.e., it concerns the combinations and arrangements of elements in sets), this work attempts to demonstrate the effectiveness of evolutionary techniques based on Darwin's theories in solving the timetabling problem, even if the solution is not fully optimal but close to optimal. An very popular meta-heuristic, the Genetic Algorithm has been successfully applied to a variety of difficult combinational optimization issues. When a local search is done on a multi-dimensional array representing course sets, halls, and time allocations, the results are combined with heuristic crossover to guarantee that fundamental restrictions are not broken. Once a workable timetable system had been developed and verified, the genetic algorithm was used to demonstrate the variety of different schedules that might be generated depending on user-specified limits and requirements. The application was created with PHP and MySQL as the programming languages.

Keywords: courses, Genetic Algorithm, Allocation, Timetable

INTRODUCTION

In computing, a timetable or schedule is a logically organised list of events, usually shown in tabular form, that provides information about a series of scheduled occurrences, including the time at which it is anticipated that these events will take place. This type of system can be used at any institution where activities must be completed by a group of people within a certain time range. Timetables have served as the organising framework for all school activities since the beginning of organised environments, which dates back thousands of years. Because of this, schools have committed significant resources in terms of time, energy, and human capital to the development of virtually ideal

timetables that must be able to meet all of the required limitations given by participating institutions (Robertus, 2018).

Traditionally, the problem is solved manually using the trial and error method, which does not guarantee the existence of a suitable solution. Even if a valid answer is discovered, it is conceivable that it will overlook even better alternatives. These uncertainties have prompted a scientific investigation into the topic, as well as the development of an automated solution technique for it. The topic has been under investigation for more than four decades, but no universal solution technique has yet been developed to address the issue (Datta et.al, 2015).

This project topic involves investigating the performance of Genetic Algorithms in automated timetabling lecture and practical scheduling. It is noteworthy that as educational institutions are challenged to grow in number and complexity, their resources and events are becoming more difficult to schedule, thus the choice of this project topic which involves investigating the performance of Genetic Algorithms in automated timetabling lecture and practical scheduling (Ossam, 2017).

A web-based students timetable system is being developed and implemented in this research to make the process of arranging courses and lectures easier.

EXAMINATION OF RELATED LITERATURE

Since the 1980s, various solutions to the problem of timetabling have been offered. Research in this area is currently ongoing, as evidenced by the publication of several recent relevant publications in the journals of operations research and artificial intelligence. This suggests that, in light of the availability of increasingly powerful computer facilities and the evolution of information technology, there are numerous scheduling issues that must be addressed (Deris et.al, 2013).

Gotlieb (2015) was the first to investigate the topic, in which he formulated it as a class-teacher timetabling problem under the assumption that each lecture had one group of students, one teacher, and any number of times that could be picked at random. Since then, the subject has been continuously investigated using a variety of methodologies and under a variety of settings. Initially, it was mostly used in educational institutions (de Gans, 1981; Tripathy,

2016). The challenge in schools is very straightforward due to the fact that the class structures are simple, which allows classical methods such as linear or integer programming approaches (Lawrie, 2012; Tripathy, 2014) to be applied with relative ease. Although the situation is becoming increasingly complex, the gradual consideration of instances involving higher secondary schools and universities, which involve a variety of different sorts of convoluted class structures, is making it even more so. Traditional methods were determined to be ineffective for dealing with the problem due to the large number of integer and/or real variables, discontinuous search space, and many objective functions, among other factors.

Researchers have begun to pay attention to heuristic-based non-classical techniques as a result of the insufficiency of classical methods. Genetic algorithms, for example, are non-classical techniques that are being applied to the problem and are worth mentioning (Alberto et. al., 2011).

Abramson et al. (2010), Piola R. (1994), and Bufe et al. (2010) are only a few of the EAs that have been utilised to solve the school timetabling problem that are worth mentioning (2001). Similarly, EAs that have been utilised to solve the university class scheduling problem include those developed by Carrasco et al. (2016), Srinivasan et al. (2013), and Datta et al. (2015). When considering diverse class structures, such as single-slot classes, multi-slot classes, split classes, combined classes, open classrooms, and group classes, Datta et al. (2015) characterised the university class scheduling problem as a multi-objective optimization problem. NSGA-II-UCTO is an enhanced version of the NSGA-II multi-objective optimizer that is based on evolutionary algorithms (Deb, 2012; Deb et. al., 2012).

During the series of international conferences on Practice and Theory of Automated Timetabling, which have been held annually since 1995, a significant amount of timetabling research for both lecture and practical has been presented (PATAT). Papers relating to this study have been published in conference proceedings, for example, see, for example, Burke and Carter (2011) developed a for formalize.

Fang (2017) studies the application of evolutionary algorithms to the solution of a series of timetabling problems in his doctoral dissertation. He proposes a framework for the application of evolutionary algorithms in the solution of

timetabling problems in the setting of educational institutions and universities. The following are the most essential aspects of this framework, which provide you with a great deal of flexibility: a declaration of the specific restrictions of the problem and the usage of a function for the evaluation of the solutions, with the recommendation that a genetic algorithm be used for the resolution of the problem because it is independent of the problem

Oliveira (Oliveira and Reis, 2014) introduces the UniLang, a language for representing the timetabling problem that was developed by Oliveira. In order to be a standard that may be used as an input language for any timetabling system for lecture and practical, UniLang must first be approved by the International Organization for Standardization. It permits the clear and natural representation of data and constraints as well as the development of quality measures and solutions for various timetabling (and related) problems, including school timetabling, polytechnic/university lecture and practical timetabling, and examination scheduling, among others.

Fernandes (2014) divided the constraints of the class-teacher timetabling problem into two categories: strong constraints and weak constraints. When stringent constraints are violated (for example, scheduling a teacher in two classes at the same time), an invalid timetable is produced. Breaking weak restrictions results in a valid timeline, but it has a negative impact on the quality of the solution (for example, the preference of teachers for certain hours). The evolutionary algorithm that has been suggested has been tested in a university with 109 teachers, 37 rooms, 1131 one-hour time intervals between classes, and 472 classes. In 30 percent of the executions, the algorithm proposed was successful in resolving the scheduling problem without breaching the stringent constraints.

For the purpose of solving timetabling challenges, various solutions have been presented. Neuronal networks (Gianoglio, 2015), heuristics (Wright, 2014), graph colouring, integer programming, genetic algorithms (Burke et al., 2012; Paechter et al., 2012), knowledge-based, and constraint logic programming are examples of such techniques (Lajos, 2010).

ALGORITHMS DERIVED FROM GENETIC INFORMATION

The earliest examples of what we now refer to as genetic algorithms appeared in the late 1950s and early 1960s, when evolutionary biologists wrote

computer programmes to simulate parts of natural evolution with the explicit goal of simulating the process of evolution. It did not occur to any of them that this strategy might be more broadly applicable to artificial problems at the time, but this realisation did not take long to arrive: "Evolutionary computation was definitely in the air during the formative days of the electronic computer," writes the historian David McCullough (Mitchell, 2010). Despite the fact that academics such as GEP Box, GJ Friedman, W.W. Bledsoe, and H.J. Bremermann had all separately created evolution-inspired algorithms for function optimization and machine learning by 1962, their work received little attention and was not further explored further. Ingo Rechenberg, then of the Technical University of Berlin, made a breakthrough in this field in 1965 when he proposed an approach he called evolution strategy, which was more closely related to hill-climbers than to genetic algorithms. There was no population or crossover in this procedure; instead, one parent was altered to generate one child, and the better of the two offspring was preserved and used as the parent for the next round of mutagenesis (Haupt et. al., 2014).

Walsh and his colleagues introduced a technology known as evolutionary programming to the United States. Candidate solutions to issues were represented as simple finite-state machines in this manner; similarly to Rechenberg's evolution strategy, their algorithm functioned by randomly mutating one of these simulated machines and keeping the better of the two solutions (Mitchell, 2015; Goldberg, 2014).

This book, written by Holland and colleagues at the University of Michigan, was the first to systematically and rigorously present the concept of adaptive digital systems, which uses mutation, selection, and crossover to simulate the processes of biological evolution as a problem-solving strategy, as a systematically and rigorously presented concept. The concept of schemata was introduced in the book, which aimed to establish genetic algorithms on a theoretically sound foundation (Mitchell, 2012; Haupt et. al., 2000). Ken De Jong's groundbreaking dissertation from the same year demonstrated the promise of GAs by demonstrating that they could perform well on a broad variety of test functions, including noisy and discontinuous search landscapes and multimodal search landscapes (Goldberg, 2011).

Initially, these applications were primarily theoretical in nature. Although genetic algorithms have been around for a while, they have only just made

their way into the business sector, thanks to the rapid increase in computational capacity as well as the creation of the Internet. The field of evolutionary computation is flourishing today, with genetic algorithms "solving problems of everyday interest" (Haupt et al., 2011) in a wide range of fields of study, including stock market prediction and portfolio planning, aerospace engineering and microchip design, biochemistry and molecular biology, scheduling at airports and assembly lines, and scheduling at manufacturing plants. The power of evolution has impacted practically every subject that can be thought of, affecting the world around us in many ways that are invisible to the naked eye, and new applications are being discovered all the time as research continues. There is nothing more fundamental than Charles Darwin's simple, powerful insight: that the random chance of variety, combined with the law of selection, is a problem-solving technique with enormous power and practically limitless application.

He detailed this concept in his 1975 book, Adaptation in Natural and Artificial Systems (which has since been re-issued with additional material), which is particularly noteworthy for its forward-looking perspective. Due to its extensive use in real-world applications, it has shown to be much more than just a reliable approach for inferring a sequence of unknown parameters from within a physical system model (David, 2013).

However, its resilience is applicable to a wide range of real optimization issues, particularly ones that are of particular interest to us, such as the scheduling problem in the context of this project.

MODELS OF REPRESENTATION METHODS OF REPRESENTATION

1. Prior to putting a genetic algorithm to work on a problem, a method for encoding potential solutions to that problem in a form that a computer can understand is required. For example, one typical way is to encode solutions as binary strings, which are strings of one-digit one-digit ones and zeros, where each digit at each place reflects a different part of the solution's value (Fleming et. al., 2002).
2. Another option that is comparable is to encode solutions as arrays of integers or decimal numbers, with each place indicating a different feature of the solution, as in the previous approach. When compared to the more restrictive way of employing only binary integers, this approach allows for

higher precision and complexity, and it is frequently "intuitively closer to the issue space" (Fleming et. al., 2002).

Another way is to represent individuals in a GA as strings of letters, with each letter representing a different component of the solution. Among the techniques utilised in this manner is Hiroaki Kitano's "grammatical encoding" approach, in which a GA was assigned to the task of generating an extremely simple set of rules known as a context-free grammar, which was then used to create neural networks for a wide range of issues (Mitchell, 1996).

There is an advantage to using the three methods described above in that they make it simple to define operators that cause random changes in the selected candidates. Examples of such operators include flipping one digit to another, adding or subtracting from the value of a number by a randomly selected amount, and changing one letter to another.

Another technique, genetic programming, which was developed primarily by John Koza of Stanford University and is based on the representation of programmes as branching data structures known as trees (Koza et. al., 2003). Random changes can be introduced into a tree using this strategy by changing the operator or adjusting the value at a specific node in the tree, or by replacing one subtree with another.

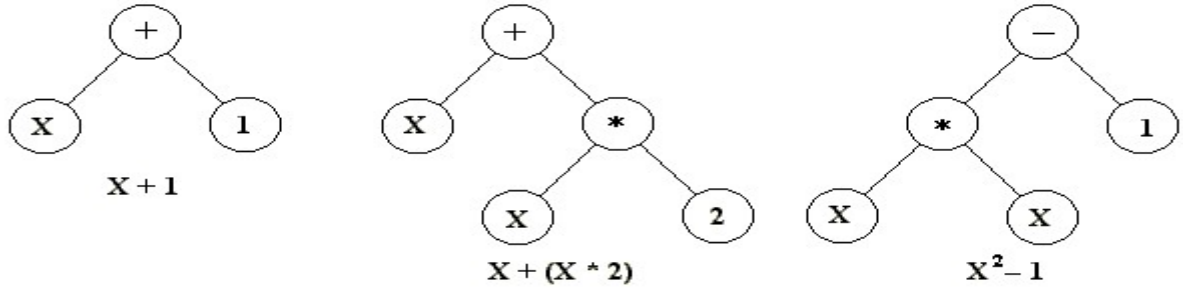

Figure 1. Three simple program trees of the kind normally used in genetic programming.
The mathematical expression that each one represents is given underneath it (Adapted from Adam Marczyk 2004).

It is important to note that evolutionary algorithms do not necessarily represent candidate solutions as data strings of fixed length. Though some represent them this way, but others do not; e.g. Kitano's grammatical encoding discussed above can be efficiently scaled to create large and complex neural

networks, and Koza's genetic programming trees can grow arbitrarily large as necessary to solve whatever problem they are applied to.

METHODS OF SELECTION

There are a variety of approaches that a genetic algorithm can employ to pick the individuals who will be passed down to the next generation, but the techniques described below are some of the most commonly used techniques. Some of these techniques are mutually incompatible, but others can and frequently are used in conjunction with one another.

1. Elitist selection ensures that only the most fit members of each generation are selected, regardless of their background. Although most GAs do not use pure elitism, they do use some variation of it where the single best or a few of the best individuals from each generation are replicated into the next generation in case no better individuals become available.
2. Fitness-proportionate selection: More physically fit persons are more likely to be picked, but this is not a guarantee.
3. Roulette-wheel selection: A type of fitness-proportionate selection in which the likelihood of an individual being selected is proportional to the degree by which his or her fitness exceeds or falls short of that of its opponents. (Conceptually, this can be compared to a game of roulette in which each individual receives a piece of the wheel, with more fit individuals receiving greater slices than less fit individuals.) Afterwards, the wheel is spun, and the one who "owns" the area where it lands each time is picked.)

The strength of the selective pressure increases as the average fitness of the population increases, resulting in a fitness function that is more discriminating as the average fitness of the population increases. This strategy can be useful in choosing the optimal pick later on when all individuals have a high level of fitness and only minor differences in fitness distinguish one individual from another, as is the case in this situation.

5. Tournament selection: Subgroups of individuals are selected from the wider population, and members of each subgroup compete against the other members of their subgroup. There is just one individual each subgroup who is selected for reproduction.

Selection is based on a numerical ranking based on fitness for each individual in the population, rather than on an absolute difference in fitness, in order to maximise fitness gains. In addition to preventing exceptionally fit people from establishing dominance early at the expense of less fit individuals, this

technique can also prevent genetic diversity from being reduced, which could impede attempts to discover an acceptable solution in the long run.

Seventh, generational selection: The children of the person selected from each generation become the entire next generation. Individuals are not passed on from generation to generation.
In the eighth generation, individuals picked from the previous generation are reintroduced into the pre-existing gene pool, where they replace some of the less fit members of the previous generation, a process known as steady-state selection. Some persons are passed down from generation to generation.
9. Selection in a hierarchical structure: Individuals are subjected to many rounds of selection in each generation. Those that survive to lower stages of review are evaluated more quickly and with less discrimination, whereas those who survive to higher levels are evaluated more rigorously. Because it uses a more rapid and less selective evaluation to weed out the majority of individuals who show little or no promise, this method has the advantage of reducing overall computation time. Only those who survive this initial test are subjected to a more rigorous and computationally expensive fitness evaluation.

As a result, a GA that explicitly evaluates a small number of individuals is implicitly evaluating a much larger group of individuals - just as a pollster who asks questions of a specific member of an ethnic, religious, or social group hopes to learn something about the opinions of all members of that group and, as a result, can reliably predict national opinion while sampling only a small percentage of the general population In the same way, the GA can "home in" on the space that contains the most physically fit individuals and select the individual who is the best overall from that group. According to the Schema Theorem, evolutionary algorithms outperform alternative problem-solving approaches in terms of speed, accuracy, and generality, respectively (John, 2011; Mitchell, 2011; Goldberg, 2013).

A good illustration of this issue may be found in Koza et al. (2011), who tackle the difficulty of synthesising a low pass filter via genetic programming. For example, in one generation, two parent circuits were selected to undergo crossover; one parent had good topology (components such as inductors and capacitors in the appropriate positions), but poor sizing; the other parent had acceptable topology but poor sizing (values of inductance and capacitance for

its components that were far too low). The other parent has a poor topology, but a fair size distribution. Following their cross-pollination, an offspring was produced that possessed both the good topology and size of one parent and so shown a significant improvement in fitness over both of its parents.

The advantage of this technique is that it allows genetic algorithms to begin with an open mind, so to speak, which is advantageous. As a result of its decisions being based on randomness, a GA is theoretically capable of exploring all possible search pathways; in contrast, any problem-solving strategy that relies on prior knowledge must inevitably begin by ruling out many possible search pathways a priori, thereby missing any novel solutions that may exist in those pathways (Koza et. al., 2015).

DATA FLOW DIAGRAM OF THE NEW SYSTEM

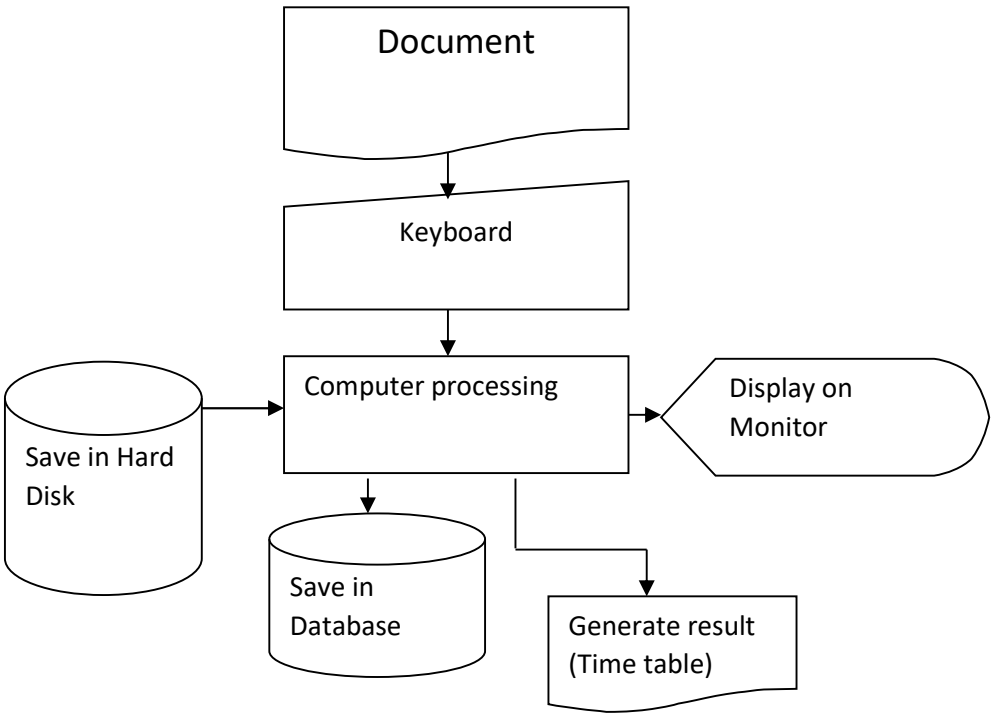

Figure 2.: Diagram of the new system

SYSTEM FLOWCHART

A system flowchart is a diagram that shows the steps necessary to solve a problem. This diagram is used to illustrate the order in which a variety of decisions are to be made and activities performed. It emphasizes how data moves in various forms through the stages of input, processing, output and storage.

OVERVIEW OF THE SYSTEM DESIGN

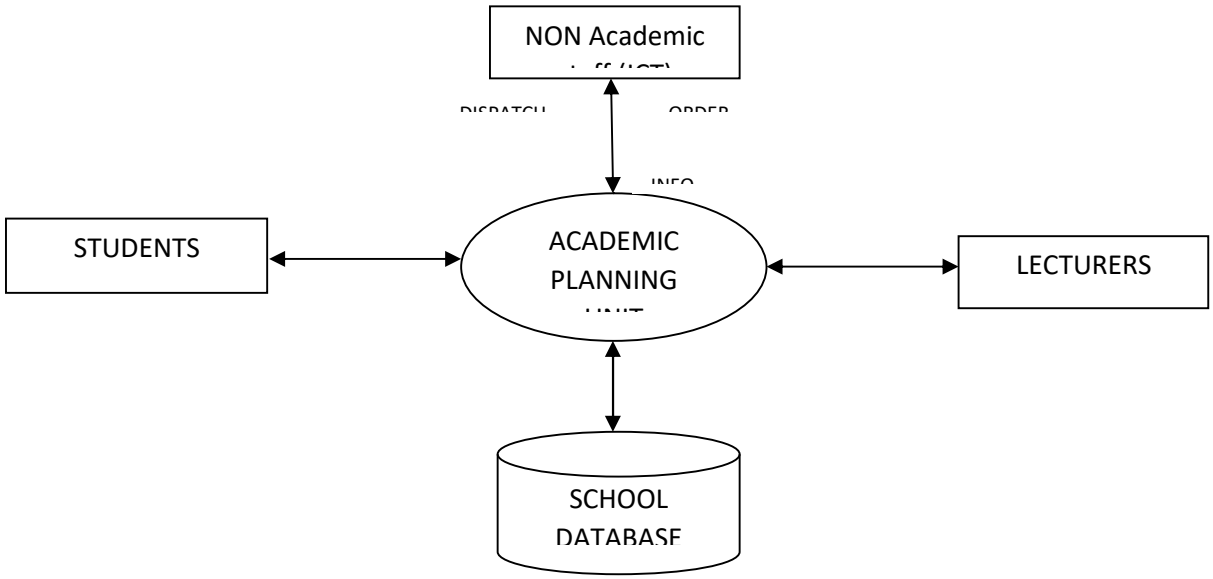

Figure 3. Diagram showing the overview of the system design

CLASS DIAGRAM

A class diagram is an organization of related objects. It gives an overview of a system by showing its classes and the relationships among them. Class diagrams only display what interacts but not what happens during the interaction hence they are static diagrams.

CLASSES

1. Lecturers
2. Buildings
3. Halls
4. Programs
5. Courses
6. Levels
7. Allocations
8. Front End Staff
9. Generator Module
10. File Writer

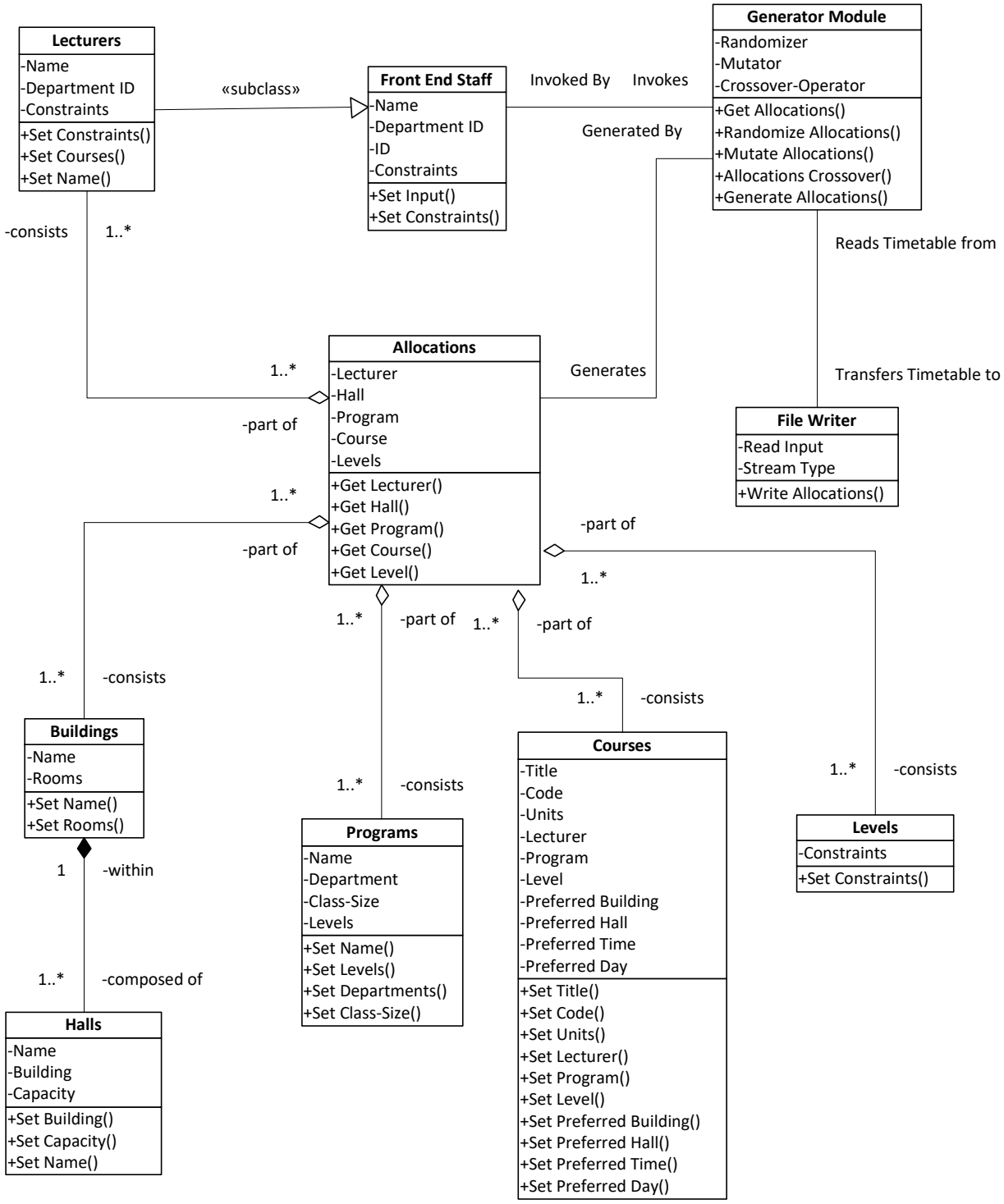

Figure 4 Class Diagram to show the relationships between the different classes associated with the system

SYSTEM IMPLEMENTATION

The PHP (Hypertext Preprocessor) and MYSQL programming languages were utilised in the development of this project's software implementation. PHP is a general-purpose server-side scripting language that was originally created for web development in order to generate dynamic web pages. PHP is also known as PHP scripting language. It has also progressed to incorporate a command-line interface capabilities and the ability to be utilised in stand-alone graphical programmes.

PHP is the ideal implementation language for this project because of the following characteristics:

i. PHP has its origins in the programming languages C and C++. PHP grammar is the closest to the syntax of the C and C++ programming languages, making it simple for programmers to learn and operate.

ii. PHP is capable of running on both UNIX and Windows systems. As a result, it is compatible with a wide range of operating systems.

In addition, PHP includes robust output buffering, which increases in size as the output flow increases. PHP internally rearranges the buffer so that the header is displayed before the remainder of the document.

The fact that PHP is parsed by the web browser means that when code written in PHP is moved to a different platform, there are no compatibility issues to contend with. v

v. PHP is compatible with a wide range of relational database management systems, and it runs on all of the most popular web servers, as well as on a wide range of operating systems.

The PHP programming language is a fully object-oriented programming language, and its platform independence and performance on LINUX servers make it a good choice for developing big and complicated online applications.

The creation of several frameworks that provide building blocks and design structure to enable Rapid Application Development has also been stimulated by PHP (RAD). Cake PHP, code igniter, Yii framework, and Zend framework are just a few of the frameworks available.

PHP IDS (Infiltration Detection System) adds security to any PHP programme by defending against intrusion. PHPIDS is capable of detecting cross-site scripting (XSS), SQL injection, header injection, directory traversal, remote file execution, local file execution, and Denial of Service (DoS) vulnerabilities (DOS).

CONCLUSION

A huge number of courses require the scheduling of lectures and practical timetables, which is a very complex challenge that is frequently managed manually by the center's personnel, resulting in outcomes that are not always ideal. This project is being undertaken to address this problem. To test the efficacy of evolutionary strategies based on Darwin's theories in solving the timetabling problem, this work uses a multi-dimensional array of course sets, halls, and time allocations, on which a local search is done to find the best solution possible. Once a workable timetable system had been developed and verified, the genetic algorithm was used to demonstrate the variety of different schedules that might be generated depending on user-specified limits and requirements. The application was created with PHP and MySQL as the programming languages.

Following an examination of the prior manual system in the context of my case study, I am hopeful that the proposed system will be implemented. I would like to continue by saying that the management, with the support of the new system, will be able to compile and publish computerised lectures and practical timetables more quickly, which may occasionally be necessary.

REFERENCES

A. Cornelissen, M.J. Sprengers and B.Mader (2010). "OPUS-College Timetable Module Design Document" Journal of Computer Science

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.

Adam Marczyk (2004). "Genetic Algorithms and Evolutionary Computation".

Al-Attar A. (1994). White Paper: "A hybrid GA-heuristic search strategy." AI Expert, USA.

Alberto Colorni, Marco Dorigo, Vittorio Manniezzo (1992). "A Genetic Algorithm to Solve the Timetable Problem" Journal of Computational Optimization and Applications.

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.

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.

Carrasco M.P. & Pato M.V. (2001). "A multi objective 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.

Chan H. W. (1997). "School Timetabling Using Genetic Search." 2th International Conference on the Practice and Theory of Automated Timetabling, PATAT'97.

Coello Carlos (2000). "An updated survey of GA-based multi objective optimization techniques." ACM Computing Surveys.

Costa D. (1994). "A tabu search algorithm for computing an operational timetable." European Journal of Operational Research.

Datta D., Deb K., & Fonseca, C.M. (2006). Multi-objective evolutionary algorithm for university class timetabling problem, In Evolutionary Scheduling, Springer-Verlag Press.

David A Coley (1999). An Introduction to Genetic Algorithms for Scientists and Engineers, 1st ed. World Scientific Publishing Co. Pte. Ltd.

Dawkins Richard (1996). The Blind Watchmaker: Why the Evidence of Evolution Reveals a Universe Without Design. W.W. Norton.