Federal Polytechnic Ilaro

Journal of Pure & Applied Sciences {FEPI- JOPAS}

Published by:

The School of Pure and Applied Science

The Federal Polytechnic Ilaro, Ogun State, Nigeria.

https://fepi-jopas.federalpolyilaro.edu.ng

E-mail:fepi.jopas@federalpolyilaro.edu.ng

ISSN: 2714-2531

FOREWORD

It is with great delight I welcome you to volume 4 issue 2 of Federal Polytechnic – Journal of Pure and Applied Sciences (FEPI-JOPAS). It is a peer-reviewed open-access multi-disciplinary Journal of global recognition which is referenced and indexed in African Journal Online (AJOL). It is a highly commendable Journal that publishes excellent research contributions and exhibiting also special attention to experience papers coming from the many application areas of pure and applied Sciences. FEPI-JOPAS publishes full-length research work, short communications, critical reviews and other review articles.

The aim of FEPI-JOPAS is to provide intellectual bedrock for both indigenous and international scholars with quality research outputs to express and communicate their research findings to a broader populace. It serves as a valuable platform for the dissemination of information to 21st Century researchers, professionals, policymakers, manufacturers, production staff, R & D personnel as well as governmental and non-governmental agencies. It also aimed to provide a platform for academics and industry practitioners to share cases on the application of management concepts to complex real-world situations in pure and applied sciences and related fields.

This volume 4 issue 2 of FEPI-JOPAS is loaded with quantum and well-featured diversity of trending topics in applied and basic research. These hot and trending topics are: Sustainable Art and Design: Activating Sighting as the Phenomenon of Representational Drawing; Assessment of Heavy Metals in Processed Meat (Tinko) Sold within Igbesa Community; The Hypoglycemic Effect of Musa Sapientum in Alloxan Induced Diabetic Albino Wistar Rat; Rainwater Quality Evaluation for Agricultural Use: Case Study of a Portland cement Producing Area; Analytical Approach to Investigating the Influence of Blood Group and Blood Genotype on the Performance of Students of Federal Polytechnic, Ilaro; Dough Mixing Time: Impact on Dough Properties, Bread-Baking Quality and Consumer Acceptability; Chemical Composition of Harvested Rainwater Around a Cement Factory in Ibeshe, Yewa North, Ogun State.

Furthermore, other topics to be encountered in this issue that have added colour and beauty to this edition are: Physicochemical properties and sensory evaluation of milk candy ‘toffee’ (a

NIGERIA candy) enrich with coconut, tigernut and groundnut; Informal Settlements in Developing Countries: Issues, Challenges and Prospects; Comparison of Sensory Properties of Meals Produced from Cowpea and Pigeon Pea; Automated Lecture Timetable Generation Using Genetic Algorithm; Septic Tanks Contamination in Groundwater Quality around Elementary Schools in Ibadan, Oyo State Nigeria; and Waste Disposal Systems in Some Selected Abattoirs Located in Ilaro Metropolis. FEPI-JOPAS has been centered on discerning the changing needs of the academic world and is committed to advancing research around the world by publishing the latest research in various academic fields and ensuring that the resources are accessible in print, digital, and online formats.

In addition, I would like to thank many people who worked so hard to ensure that publishing this issue 2 of volume 4 is a reality. I would like to thank the Editorial Board for their guidance and the publishing team for the continued support and effort in streamlining the publication process. I am grateful to the reviewers who provided timely and constructive reviews for the papers assigned to them. The authors are solely responsible for the information, date and authenticity of data provided in their articles submitted for publication in the Federal Polytechnic Ilaro – Journal of Pure and Applied Sciences (FEPI-JOPAS).

I am looking forward to receiving your manuscripts for the subsequent publications. You can visit our website (https://fepi-jopas.federalpolyilaro.edu.ng) for more information, or contact us via e-mail us at fepi.jopas@federalpolyilaro.edu.ng

Thank you and best regards.

Prof. Olayinka Oyewale AJANI

(Editor-in-Chief)

TABLE OF CONTENTS

https://fepi-jopas.federalpolyilaro.edu.ng Review

Automated Lecture Timetable Generation Using Genetic Algorithm

Ogunseye, J. O. and Ojuawo, O. O.

Department of Computer Science, Federal Polytechnic Ilaro Ogun state.
✉james.ogunseye@federalpolyilaro.edu.ng

Abstract

Making the perfect school schedule has been a major concern for both administrators and lecturers. Often, this is done manually, and many errors are discovered at the stages where a class is supposed to begin. A Genetic Algorithm approach is proposed in this study to aid the timetable scheduling process. The main reason for this is that there are numerous factors and decisions to be taken, such as audience, subjects, lecturers, student group, and time slot the need to take into account the difficulty of the classes they will take as well as their exam schedules and availability. This approach compares the quality of different schedules to decide which one is the best based on fitness costs. Then, we may decide on better schedules utilizing crossover, mutation, and elitism choices. The use of a genetic algorithm assists in the automatic generation of a timetable based on information. Human error and effort in the planning process are reduced by the genetic algorithm.

Keywords: Genetic Algorithm, Scheduling, Timetable

INTRODUCTION

The timetable task focuses on creating timetables that need to be flexible to different constraints. A schedule is usually described as an ordered list in tabular form, detailing a planned set of activities to take place at a particular location and their scheduled times.

One definition of a schedule is to optimize a set of tasks, activities, or events for a set of objects in the space-time matrix to fit a set of desired constraints. A schedule, according to the Collins Concise Dictionary (4th edition), is a collection of events planned at a scheduled time.

Even if the solutions are often sub-optimal, scheduling the most diverse courses is a complex task, usually done manually by the center's employees. In an academic environment, a well-or Genetic algorithmized, structured and conflict-free class schedule is one of the most important factors in effectively conducting activities. Universities must carefully create a curriculum for each academic year, taking into account subjects and related exams for each department.

- courses for a particular semester,
- lecture days (usually depending on school hours),
- Set the lecture time,
- The location of a particular lecture. Information commonly used to plan timetables.

Four main parameters make up the scheduling problem: T (Time), R (Available Resources), M (Planned Contacts), and C (Constraints). Academic timetables are important, but they are often used in educational institutions and require a lot of work.

Another factor contributing to the challenge is the length of the lecture and exam, the number of constraints, and the complexity of structuring the size of the lecture and exam with the assignment criteria. Responses from the previous semester or previous year (Moreira, 2008). We need to time our events so that we do not experience more than one event at a time (Robertus, 2002).

Literature Review

To solve the problem to some extent, a family of unconventional methods known as genetic algorithms (Genetic Algorithm) can handle complex and large search fields (George et al., 2010; Guo et al., 2011). (Deb, 1999). The search space is divided into subspaces using various genetic algorithm techniques to increase the likelihood that the starting population contains healthy chromosomes. This method improves the chances of finding a near-ideal solution and typically speeds up computation (Chen et al., 2002; Pezzella et al., 2008), (Carvalho et al. 2012). In a genetic algorithm, the best solutions from previous generations are developed according to the survival-of-the-fittest principle until a nearly optimal solution is reached. It shows self-or Genetic algorithmization and adaptation. This is the same way the best or Genetic Algorithmnisms survive and reproduce. It uses an iterative process and displays possible solutions as long strings of genes called chromosomes. This usually applies to very large spaces. Evaluate how learned to produce better offspring using a fitness function that measures either the maximum or minimum goal to be achieved. There is no doubt that the lesson planning process is a common theme in all educational institutions.

Timetabling as an Np-Complete Problem

The NP-complete (NP-C, NP stands for Nondeterministic Polynomial Time). A complexity class is a class of problems with two features in computational complexity theory, which is:

- A set of problems with this property is called NP, and the solution of a given problem can be verified quickly (polynomial time).
- If the problem can be solved quickly (in polynomial time), so do other NP problems.

The solutions given are readily verifiable, but there is still no reliable and effective way to find them. The fact that there are no known fast solutions to NP-complete problems is actually its most obvious feature. This means that the time it takes to solve a problem using currently available algorithms increases with the size of the problem (Chohan, 2009). When solving scheduling problems, we always look for the best option out of all possible options. The search space, also called the state space, is the set of all feasible solutions (the set of desirable solutions, some of which are more desirable than others).

Basis for a Genetic Algorithm

- A set or population of hypotheses for solving a problem.
- How to assess the quality or performance of each solution in the population, or how to assess the health of the generated solutions.
- Techniques for combining parts of good solutions to create new, generally better solutions.
- A mutation operator to prevent internal differences from being lost over time.

A programming technique called a genetic algorithm mimics biological evolution as a solution to a problem. Given a problem to solve, the genetic algorithm is given a set of possible solutions encoded in some way, along with a fitness function that can be used to quantify each candidate. These candidates can be tested for improved variants by genetic algorithms, but are often randomly selected. In the next step, a genetic algorithm evaluates each candidate based on its fitness function. In a random pool of candidates, the majority will not do well at all and will be eliminated. However, some of them may prove promising. It can show activity even if it is weak and ineffective in solving the problem. These viable candidates are retained and clones are allowed. Several copies are then made, but these are incomplete because incorrect changes were made during copying. The digital descendants are then passed on to the next generation, creating a new pool of possible solutions and undergoing a second suitability assessment. Genetic algorithms introduced into populations, purely random, random variability allow some individuals to find better, more complete, or more effective solutions to the problems at hand could have been improved. Candidate solutions that have been degraded by code changes are eliminated by the genetic algorithm. Winners are re-selected, randomized, copied to the next generation, and the cycle repeats. By repeating this process for hundreds or thousands of rounds, we can find a very effective solution to the problem, since each round should, on average, show an increase in the fitness of the population. When it comes to problem-solving methods, genetic algorithms have proven to be incredibly efficient and powerful, demonstrating the power of evolutionary principles. Solutions created by genetic algorithms tend to be more efficient, sophisticated, and complex than those created by human engineers. In most cases, genetic algorithms have answers that the programmer who originally created the algorithm did not understand (Marczyk, 2004).

Fitness Function

Limits are quantified to determine suitability. He has two categories of limits: hard limits and soft limits. A hard constraint is one that invalidates the curriculum if the solution is not met. For example, one instructor is teaching two of her courses at the same time. Therefore, all strict constraints must be met for a schedule to be valid. Soft constraints are strongly encouraged to be met but do not invalidate a solution if they are not. For example, some instructors prefer to give lectures in the morning, while others prefer to give lectures. The fitness function gives a positive value (+2) when the condition is met and a new Genetic Algorithmtive value (-2) when the condition is not met (-2). However, satisfied soft constraints also contribute positively (=1) to fitness, while unsatisfied constraints have no effect on the fitness function.

For Hard Constraints;

HCi = 0. if hard constraint i is achieved.

HCi= 2. if hard constraint i is not achieved. (1)

For soft Constraints:

SCi= 0. If soft constraint i is achieved.

SCi=1. if soft constraint i is not achieved. (2)

As a result, each constraint is included in the fitness function. The ultimate goal of this algorithm is to find chromosomes with zero fit values. Genetic Algorithm does not guarantee detection of such chromosomes, and even if detected, it is generally computationally intensive (Mohammed et al., 2017).

Method of Representation

Before we can use genetic algorithms to tackle a problem, we need a way to encode potential solutions to the problem into a form that computers can understand. A common practice is to encode the solution as a binary string, which is a sequence of 1's and 0's. Each digit corresponds to a value for a different aspect of the solution (Fleming et al., 2002).

Another similar approach is to encode the solution as an array of integers or fractions, with each position representing a different aspect of the solution.

Compared to more constrained approaches using binary numbers, this method promotes higher precision and complexity, and rarely "intuitively approaches the problem space." (2002) Fleming et al.

Another strategy is to represent the Genetic Algorithm person as a series of letters. Each letter represents a different component of the solution. In Hiroaki Kitano's "Grammar Coding Method", Genetic Algorithm was tasked with developing a simple set of rules known as context-free grammars, which were then used to create neural networks for various problems (Mitchell, 1996).

Advantages of the above method include the ease of defining an operator that randomly modifies the selected candidate. For example, convert 0 to 1 and vice versa, subtract or add a specific amount to a number, replace one character with another, and so on. Another approach, called genetic programming, was developed by John Koza at Stanford University to represent programs as branching data structures called trees (John et al., 2003). Using this technique, you can change operators or values at specific nodes in the tree, or swap one subtree for another to make random changes

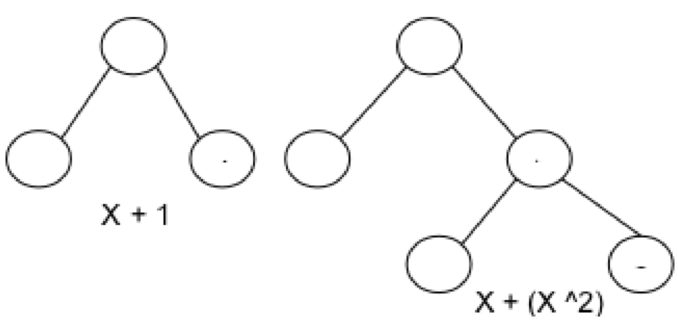

Fig 1: Three simple programs typically used in genetic programming trees. Each one is followed by the mathematical phrase that it represents.

Source: Adam Marczyk (2004). "Genetic Algorithms and Evolutionary Computation ". Available online at http://www.talkorigins.org/faqs/genalg/genalg.html

Figure 1 - Program trees

It is important to remember that evolutionary algorithms do not always represent potential solutions as fixed-length data strings. Some systems show it that way, some don't. For example, the goat's genetic programming tree can grow to any size needed to solve a particular problem. In addition, the Kitano grammar encoding described above can be efficiently scaled to produce large and complex neural networks.

Methods of Change- mutations and crossovers

After selection selects the most suitable individuals, they should be randomly changed in order to improve the fitness of the next generation. There are two basic ways to accomplish this. Mutation is the first and easiest. Mutations in genetic algorithms result in subtle changes at specific points in an individual's code, much like mutations in an or Genetic Algorithm cause one gene to become another.

Table 1. Effect of mutation on an individual.

0	0	1	0	1	1	0	1
0	0	1	1	1	1	0	1

Table illustrating how a mutation at position 4 in an individual's DNA converts the 0 at that location into a 1 in a population of 8-bit strings.

Table Source: Adam Marczyk (2004). "Genetic Algorithms and Evolutionary Computation ". Available online at http://www.talkorigins.org/faqs/genalg/genalg.html

The second method, known as crossing, takes two individuals and swaps parts of their genetic code to create a synthetic "offspring" that is a combination of their parents. This procedure aims to mimic similar recombination that occurs in chromosomes during sexual reproduction (Marczyk, 2004).

The two most common types of crossovers are single-point crossovers and unitary crossovers. In the single-point crossover, exchange points are placed at random locations in two people's genomes. One gives all code up to that point, the other gives all code after that point. Sponsored. An offer to the offspring of the father. In the uniform crossover, the genome score of the offspring at a given point in time is equal to either the genome score of one parent or the genome score of the other parent at that point in time.

Table 2. Single-point crossover

0	0	1	0	1	1	0	1
1	0	1	1	0	0	1	1
1	0	1	1	0	1	0	1

Table shows three straightforward program trees similar to those used in genetic programming.

Each one is followed by the mathematical phrase that it represents.

Table Source: Adam Marczyk (2004). "Genetic Algorithms and Evolutionary Computation ". Available online at http://www.talkorigins.org/faqs/genalg/genalg.html.

Strengths of Genetic Algorithms

The key point is the fact that genetic algorithms are inherently parallel. Most other algorithms are sequential and can only explore one direction of the solution space at a time. If the solution you find isn't perfect, you have no choice but to start over. However, the genetic algorithm has many children, so it is possible to examine different parts of the decision space at the same time. If a component turns out to be a dead end, it can simply be discarded and moved in a more promising direction, increasing the likelihood of finding the perfect solution with each iteration. (Marchik, 2004; Holland, 1992).

Such problems lend themselves particularly well to genetic algorithms since the space of all possible solutions is too large to be exhaustively explored in a reasonable amount of time. The key feature that differentiates genetic algorithms from other approaches such as simulated annealing and hill climbing is the intersection. Because there is no intersection, each solution explores the immediate search space without considering what others have found.

Limitations of Genetic Algorithms

Genetic algorithms have proven to be an effective and efficient approach to problem-solving, but genetic algorithms are not a panacea. The genetic algorithm has some limitations: creating a representation of the problem is the most important factor to consider when developing a genetic algorithm. The language used to identify possible solutions must be robust. This means it must be able to withstand unpredictable changes without constantly spawning deadly bugs and nonsense.

The actual genetic programming code cannot be altered by some methods. As discussed in the Representation section, GP models a person as a tree of executable code that can be modified by switching and rearranging subtrees.

CONCLUSION

Due to the complexity of the scheduling problem, applying just one principle is not enough. Only when the mapping and boundary conditions are clearly defined and greatly simplified, and some principles are applied, can the planning problem be solved, leading to an increase in hybrid solutions (a combination of different solution methods). The steps of the proposed method are as follows. The first step is to enter a list of courses, the number of each section, faculty limits, available rooms, hard and soft limits for those rooms, and lab resources. In the second phase, an initial chromosome population (a set of initial solutions) is randomly generated. In the third stage, the operations of an iterative genetic algorithm (crossover and mutation) are performed on the original solution until a near-ideal solution that satisfies the given fitness function is obtained. The better the solution, the closer to zero.

In order to fully implement the genetic algorithm and have a fully functional system, we need to achieve the following goals:

- The capabilities of the genetic algorithm for mutations should be built into the system.
- The current crossover module can be rearranged to dynamically handle different course units.

When completely developed, it helps create a genetic algorithm that can accept input from several courses but still has to include an error-handling mechanism.

REFERENCES

Adam Marczyk (2004). "Genetic Algorithms and Evolutionary Computation ". Available online at_ http://www.talkorigins.org/faqs/genalg/genalg.html

Adejuwon, O. S. (2012, May). *Development of a University Timetable Automated System*.

Al- Jarrah, M. A., Al- Sawalqah, A. A., & Al-Hamdan, S. F. (2017). Developing a Course Timetable System For Academic Departments Using Genetic Algorithm. *Jordanian Journal of Computers and Information Technology*, 1. https://doi.org/10.5455/jjcit.71-1465735040

de Carvalho, G., Laender, H. F., Goncalves, & da Silva, S. (2010). A Genetic Programming Approach to Record Deduplication. *IEEE Transactions on Knowledge and Data Engineering*, *24*(3), 399–412. https://doi.org/10.1109/TKDE.2010.234

Chen, J., Goldberg, D.E., Ho, S., & Sastry, K. (2002). Fitness Inheritance In Multi-objective Optimization. *Annual Conference on Genetic and Evolutionary Computation*.

Chohan, O. (2009). Scheduling using Genetic Algorithm.

Deb K. (1999). Multi-objective genetic algorithms: problem difficulties and construction of test problems. *Evolutionary computation*, *7*(3), 205–230. https://doi.org/10.1162/evco.1999.7.3.205

Anderson, G., Marwala, T., & Nelwamondo, F.V. (2010). Use of Data Mining in Scheduler Optimization. *ArXiv, abs/1011.1735*..

Guo, P., & Zhu, L. (2011). The Design and Implementation of Timetable System Based on GA's. *IEEE International Conference on Mechatronics Science, Electric Engineering and Computer (MEC)*, pg 1497-1500.

Holland, J. H. (1992). Genetic Algorithms. *Scientific American, 267(1), 66–73.* http://www.jstor.org/stable/24939139

Koza, John & Bennett, F.H. & Andre, David & Keane, Martin. (1999). Genetic programming III: Darwinian invention and problem solving [Book Review]. *Evolutionary Computation, IEEE Transactions on*. 3. 251 - 253. 10.1109/TEVC.1999.788530..

John, K., Keane, M., Streeter, M., Mydlowec, W., Yu, J., & Lanza, G. (2003). *Genetic Programming IV. Kluwer Academic Publishers.*

Marczyk, A., 2004. Genetic algorithms and evolutionary computation. *The Talk Origins Archive*: http://www. talkorigins/faqs/genalg/genalg. html

Moreira, J. J. (2008). A system for Automatic Construction of Exam Timetable Using Genetic Algorithms. *Journal of Algorithm and Computations*, *6*(9), 1-18.

Peter, F., & Purshouse, R. C. (2002). Evolutionary algorithms in control systems engineering: a survey. *Control Engineering Practice*, (10), 1223-1241.

Pezzella, F., Morganti, G., & Ciaschetti, G. (2008). A Genetic Algorithm for the Flexible Job-shop Timetabling Problem. *Computers & Operations Research*, *35*(10), 3202-3212.

Willemen, R. J. (2002). *School timetable construction-Algorithm and Complexity* (1st ed ed.). Netherlands: University of Eindhoven Press.