Overview of Genetic Algorithms in Educational Timetabling

Luca Quaer
luca@quaer.net
Wilhelm Büchner Hochschule
Darmstadt, Baden-Württemberg, Germany

ABSTRACT

Empty

KEYWORDS

Genetic Algorithms, Educational Timetabling, Metaheuristics

1 INTRODUCTION

Educational timetabling involves creating schedules for educational institutions such as schools, colleges, and universities. The problem domain can be divided into the following three main problems [7, 9]: High-School Timetabling (HTT), University Course Timetabling (CTT) and University Examination Timetabling (ETT). Although a clear distinction between these three problems is not always possible, they generally differ significantly from one another [2]. However, each of these problems essentially is a resource allocation problem with the goal of assigning classrooms, instructors, and students to specific time slots for various courses or activities, ensuring that all constraints and requirements are met. This includes avoiding conflicts (e.g., a student being scheduled for two classes at the same time), adhering to institutional policies, and maximizing the efficient use of resources.

The difficulty in finding a valid and effective solution to such a problem lies in meeting the diverse requirements of different stakeholders (e.g. students, teachers, administration), multiple constraints and resolving resource conflicts in a combinatorial complex solution space caused by the numerous constraints. Timetabling problems like these are therefore known to be NP-complete in their general form, meaning that the difficulty of finding a solution increases exponentially with the problem size, which in turn makes it impossible to find a deterministic algorithm providing an acceptable solution in polynomial time [2]. One popular approach to addressing the complexity of timetabling problems is the use of metaheuristic algorithms [2]. This class of algorithms leverages a non-deterministic search approach which compromises on finding an optimal solution in favor of better runtime performance. Consequently, such algorithms are not guaranteed to find the best solution for a given problem, but a near optimal one [1]. Despite this limitation, metaheuristic algorithms are widely used in educational timetabling due to their ability to provide high-quality solutions within a reasonable timeframe. These algorithms can be broadly classified into two categories: single-solution and population-based metaheuristics [6]. Single-solution based algorithms use a single candidate solution and iteratively improve it by using local search, but are prone to get stuck in local maxima [6]. Population-based metaheuristics on the other hand work on multiple candidate solutions during the search process, avoiding the risk of getting stuck in a local maximum by maintaining diversity among the solution candidates [6]. Popular single-solution based algorithms in the timetabling domain are simulated annealing, local search and Tabu search [4, 6]. Well-known population based metaheuristics are genetic algorithms, particle swarm optimization and ant colony systems [2, 6].

Among these methods, genetic algorithms are known for their versatility and application in a variety of use cases with the need of searching for solutions of a combinatorial problem in a large solution space. Therefore, this paper specifically focuses on genetic algorithms and how they are used in the domain of educational timetabling.

Genetic algorithms (abbr. *GA*) are a heuristic search method inspired by the process of natural selection in biological evolution and thus belong to the group of evolutionary algorithms [6]. As mentioned previously, genetic algorithms utilize a population based approach, meaning multiple solution candidates are iteratively evolved through numerous generations imitating the Darwinian theory of survival of the fittest [6].

2 METHODS

This paper specifically focuses on the application of genetic algorithms in the domain of educational timetabling and not timetabling or scheduling problems in general. To accomplish this, a thorough search among recent and early publications on this topic has been conducted, to create a representative overview of the most important concepts of genetic search used in the field of timetabling. After an introduction of the basic and some advanced concepts of genetic algorithms, the research results will be visualized and discussed.

3 BASIC CONCEPTS

Genetic algorithms are a type of search and optimization algorithm inspired by the biological process of reproduction and natural selection and represent one branch in the field of evolutionary computing [3, 5].

In the search for a solution to an optimization problem, the set of possible solutions – the so-called solution space – must first be determined and made comprehensible for an algorithm in form of a data structure, which is suitable for representing a solution [1]. This representation of the solution is also called *encoding* and contains the data of a possible solution to the problem to be solved. In nature, this data is encoded on chromosomes. Similarly, in genetic algorithms the possible solution in coded form is also called *chromosome* or *individual* [1].

In addition, genetic algorithms employ a population based search approach, whereby instead of a single potential solution a whole set of solutions is iteratively improved. Such a set of solutions is called *population* and consists of multiple chromosomes. The stages of iterative improvements are called *generations* [1].

In order for the algorithm to optimize towards a desired solution, it is necessary to have a measure in place to evaluate and compare the chromosomes. This value referred to as *fitness* and is provided by the *fitness function* (also called *objective function*) [1].

With these basic terms defined, the general process of genetic algorithms can now be described as follows: First, an *initial population* must be created and the fitness of its chromosomes must be evaluated [1]. Pairs (or triples, quadruples, etc.) are then selected (*selection* phase) from this population in order to reproduce (known as *crossover*) [1]. The resulting offspring may undergo one or more mutations (*mutation* phase) with a defined probability before the fitness of these new chromosomes is determined (*evaluation* phase) [1]. Based on certain criteria chromosomes from the current generation and their offspring are now selected, to replace the population with a new one (*replacement* phase) [1]. The result of this step is a new generation of chromosomes forming a new (usually fitter) population [1]. From this population, chromosomes are once again selected for reproduction, starting the process all over again [1]. The genetic algorithm could theoretically continue indefinitely according to this pattern, with termination conditions serving as the only means of halting the process [2]. The forementioned phases of genetic algorithms will be explained in the following chapters.

3.1 Encoding

Genetic algorithms require two essential components: an encoding and a fitness function [1]. The encoding plays a pivotal role in the design of a genetic algorithm [6]. Its most significant property is that it can completely represent the solution space of the problem at hand, thereby deriving the potential solution to the problem from a given chromosome [1]. Moreover, the encoding must be designed in consideration of the data processing of other genetic algorithm components, such as the fitness function and the crossover operator [1]. The fitness function must calculate the fitness value based on this representation of a solution candidate, and the crossover operator should generate offspring that are as valid as possible [1]. In particular, the latter aspect can often only be fulfilled by an adapted, domain-specific representation [1]. The following paragraphs present some well-known encodings.

3.1.1 Binary Encoding.

3.1.2 Permutation Encoding.

3.1.3 Value Encoding.

3.1.4 List Encoding.

3.1.5 Matrix Encoding.

3.2 Fitness Function

The fitness function of a genetic algorithm is arguably its most crucial component [2]. "It is the only chance that you have to communicate your intentions to the powerful process that genetic programming represents. Make sure that it communicates precisely what you desire"[8]. Especially in the context of constrained optimization problems with multiple objectives (e.g. hard and soft constraints), the fitness function must be carefully designed to convey the correct optimization target for solving the problem at hand [2, 3].

3.3 Initial Population

Once an appropriate encoding and a fitness function have been identifier, the first step in the actual execution of the algorithm is to generate an initial population [1].

3.4 Selection

asdf

3.5 Crossover

asdf (und [2])

3.6 Mutation

asdf

3.7 Evaluation

asdf

3.8 Replacement

asdf

3.9 Termination

asdf [2]

4 ADVANCED TECHNIQUES

To do.

5 DISCUSSION

To do.

6 CONCLUSION

To do.

7 FUTURE WORK

To do.

REFERENCES

[1] Michael Affenzeller, Stephan Winkler, Stefan Wagner, and Andreas Beham. 2009. Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications.

[2] G. N. Beligiannis, C. Moschopoulos, and S. D. Likothanassis. 2009. A genetic algorithm approach to school timetabling. *Journal of the Operational Research Society* 60, 23–42. Issue 1. https://doi.org/10.1057/palgrave.jors.2602525

[3] Jenna Carr. 2014. An Introduction to Genetic Algorithms.

[4] Sara Ceschia, Luca Di Gaspero, and Andrea Schaerf. 2023. Educational timetabling: Problems, benchmarks, and state-of-the-art results. , 18 pages. Issue 1. https://doi.org/10.1016/j.ejor.2022.07.011

[5] David E. Goldberg. 1989. *Genetic Algorithms in Search, Optimization & Machine Learning*. https://doi.org/10.1023/A:1022602019183

[6] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. 2021. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications* 80 (2 2021), 8091–8126. Issue 5. https://doi.org/10.1007/s11042-020-10139-6

[7] Jeffrey H Kingston. 2013. Educational timetabling. In *Automated Scheduling and Planning: From Theory to Practice*. Springer, 91–108.

[8] Kenneth E Kinnear Jr. 1994. A perspective on the work in this book. *Advances in genetic programming* (1994), 3–19.

[9] Andrea Schaerf. 1999. A survey of automated timetabling. *Artificial intelligence review* 13 (1999), 87–127.