A Review of Optimization Algorithms for University Timetable Scheduling

Hayat Alghamdi
College of Computer and Information Systems
Umm Al Qura University
Makkah, Saudi Arabia
s43980248@uqu.edu.sa

Tahani Alsubait
College of Computer and Information Systems
Umm Al Qura University
Makkah, Saudi Arabia
tmsubait@uqu.edu.sa

Hosam Alhakami
College of Computer and Information Systems
Umm Al Qura University
Makkah, Saudi Arabia
hhhakam@uqu.edu.sa

Abdullah Baz
College of Computer and Information Systems
Umm Al Qura University
Makkah, Saudi Arabia
aobaz01@uqu.edu.sa

Abstract—**The university course timetabling problem looks for the best schedule, to satisfy given criteria as a set of given resources, which may contain lecturers, groups of students, classrooms, or laboratories. Developing a timetable is a fundamental requirement for the healthy functioning of all educational and administrative parts of an academic institution. However, factors such as the availability of hours, the number of subjects, and the allocation of teachers make the timetable problem very complex. This study intends to review several optimization algorithms that could be applied as possible solutions for the university student course timetable problem. The reviewed algorithms take into account the demands of institutional constraints for course timetable management.**

Keywords-timetabling; genetic algorithms; Particle Swarm Optimization (PSO)

I. INTRODUCTION

Many higher education institutions have teaching staff from various fields, who work together to meet the educational goals set. However, creating a timetable with no conflicts, which these lecturers can use is one of the challenges most universities face. Generally, a timetable is a table of various events and their schedule [1]. Therefore, in a university timetable, the institution assigns the courses taken by students and delivered by tutors to a defined finite set of resources, which include time slots and classrooms. This process presents many challenges. For example, in a typical school, there are several student groups who may or may not be taking the same course at the same time [2]. Thus, when scheduling lectures, one has to ensure that learners, lecturers, and lecture halls do not conflict. This need makes the timetabling of university courses to be a complex and time-consuming task to complete. University timetabling committees cannot perform this distribution randomly as they must consider several decisive factors. There are several constraints that guide the process of making an effective timetable. These constraints, which are rules, policies, and preferences of universities, lecturers, and students, can be categorized as hard, soft, or medium. Hard constraints are those guidelines that should not be violated or overlapped at all, soft constraints are desires of the involved stakeholders that can be ignored without any serious consequences, and medium constraints are preferences that they are preferred to not be violated [3]. The committee handling the timetabling must consider all these requirements to create an optimal outcome. Due to the complexity of this problem, timetable coordinators spend a considerable amount of time looking for the best solution. However, even if they have a lot of experience, the resolution found may not be optimal due to the large number of possible combinations. Thus, the distribution of workload among the various lecturers in an academic institution constitutes a problem of combinatorial nature. In general, resolving such challenges and obtaining exact optimal solutions is computationally intractable. Therefore, the university timetabling problem is an example of a Non-Polynomial (NP)-hard problem [4]. These are problems with no particular efficient solutions. In the case of timetabling where there is no specific algorithm that can be utilized to schedule classes since each institution has its special constraints [5, 6]. At the same time, when done manually, the obtained result depends on both the initial approach and the experience of the timetabling committee.

Public universities typically take days to manually schedule all the classes that students should take depending on the availability of lecturers and classrooms [7]. Thus, the proposal to automate this process intends to meet real needs, with the main aim to reduce the time taken to complete it efficiently. Describing the real problem by mathematical functions, the study seeks the values of such models with the use of optimization techniques. This way, it will provide a solution that maximizes the use of available human resources and satisfies the needs of all involved parties. This approach becomes very complex as more restrictions arise, such as the

Corresponding author: Hosam Alhakami

number of courses in the school and the availability of essential resources like lecturers, students, and classrooms. The presented solutions behave inefficiently when there is an expansion of the institution and their accuracy suffers as the number of courses and classes grow [8]. Thus, a personalized computational approach to address the issue becomes important and a priority. Due to the importance of the problem, several formalizations and solution methods have been proposed. The approaches based on meta-heuristics are considered to be particularly relevant. However, these optimization algorithms are not mutually exclusive. Instead, they combine ideas from different areas, with the main aim of providing different solutions to the timetabling issue [9]. One of the most popular approaches is acquiring solutions based on Genetic Algorithms (GAs) due to their high degree of computational parallelization and enhanced computational performance. Overall, university course timetabling can be treated as organizing a group of classes to meet the specifications of particular institutions. Over the years, various researchers have created various algorithms to find an optimal solution to this issue. For instance, authors in [9] created graph coloring schemes for timetable scheduling in 1967. Their work formed the foundation upon which more sophisticated techniques have been based. One of these models is linear programming, which was used to solve a labor scheduling problem in [10].

Moreover, a GA presents one of the best possible methods to address NP-complete problems. At its core, a GA is a parameter optimization strategy that iterates over a population, seeking the best fit for a problem until it obtains an optimal solution [11]. Several works have been completed with this algorithm for the treatment of timesheets. The GA borrows heavily from the field of natural sciences on the way it works. In particular, it allows observing events from the lens of natural phenomena, such as mutations and natural selection [12]. Hence, it helps developing computational mechanisms, which resemble these processes. Therefore, GAs use and specify biological terminologies, such as crossing-over and mutation, in a precise and specialized way [13]. In particular, through the process of cross-over, the fittest generations pass on their attributes to their offspring. In turn, mutation helps to prevent stagnation within a population and is essential for the proper functioning of the algorithm. Along this line, this study will use surveys to understand the general foundations for the problem of university lesson planning. This activity will assist in formulating the optimization model and describing the manner in which the algorithm can be adapted to make it applicable to the problem under investigation.

II. RELATED WORDK

Because university course timetabling is an NP-hard problem, there are several methods that have been explored in attempts to find optimal solutions. The manual approach requires several days of work and often results in inefficient outcomes [4, 14]. However, there are several other procedures that can be used to schedule academic activities and in which restrictions related to the type of problem considered must be respected. Nonetheless, the objective often involves distributing teachers and students in a particular course depending on time and resource availability, while also respecting their desires. Educational timetables are of three types: school, course, and exam timetables [1]. Scheduling the course timetable presents the most difficulties among the three as it is utilized frequently through a typical university semester and it covers only particular days within a week. University timetabling is not simple due to various constraints that would be applied to produce effective course timetabling [15-17]. Eventually, the timetabling constraints can be classified as hard and soft; the hard constraints are considered as mandatory while the soft constrains are optional [18-20]. Table I summarizes the most important soft and hard constraints of course timetabling.

TABLE I. SOFT AND HARD CONSTRAINTS

Ref.	Soft constraints	Hard constraints
[15]	Honours and general courses need to be scheduled in non-overlapping time-slots.	- Courses having common students cannot be allotted at the same time slot on the same day. - The total number of available periods of the daily timetable is 8 hours (maximum)
[16]	- The number of students taking a course must fit into the assigned room. - Each additional student over the capacity of the room counts as a violation. - Lectures for a given curriculum must be consecutive.	- A room cannot be assigned to more than one lecture in a given period. - Courses belonging to a curriculum must be assigned to different periods. -Courses taught by a lecturer must be assigned to different periods.
[17]	To open the course, a minimum number of students should be registered	The timetable should be scheduled based on the university calendar
[18]		- Considers the workload of the lecturers. - Takes into account the free time (i.e. office hours and resets) of the lecturers.
[19]		- Every course should be assigned to a venue at a particular timeslot. - The scheduled courses must not exceed the venue capacity.
[20]	- Break periods must be allocated before other courses. - Faculty general courses must be allocated to slots before departmental courses.	- Lecture rooms must not be booked twice at the same time. - All lecture venues and rooms must be scheduled once not twice.
[21]		- No student group has two events at the same time. - No lecturer has two events at the same time. - No event is in a room with less capacity than the number of students at the event.

Hard and soft constrains are related to many factors such as courses, lecturers, students, classrooms, and operational dates and hours [16, 18, 19, 21]. Thus, it is important to apply optimization algorithms for effective scheduling of the course timetable. The use of optimization techniques in problem-solving dates back several decades. Nonetheless, it has recently been boosted by the advent of computers and the increase in their processing capacity, which has led to a rise of the number of scheduling approaches. However, the application of these methods to solve practical problems has not been verified at the same proportion as the rate of their development. One of the reasons is the complexities of the real behavior and the model generated, which are described by non-linear behavior functions and whose solution space may be non-convex [14]. Heuristic methods have played an important role in resolving

these types of problems. Primarily, they involve values of the functions in the process, regardless of whether there is unimodality of the function or continuity in their derivatives. On the other hand, they demand a large number of calculations as stated by authors in [22], who argue that instead of performing many additional computations to determine a gradient with mathematical programming techniques, it may be more beneficial to use this time and resources to explore the search space more intensively. Among the heuristic optimization techniques, there are metaheuristic methods, which promise accurate and optimal solutions to the timetable scheduling issue. Notable approaches of this nature include Greedy Randomized Adaptive Search (GRASP), simulated annealing, tabu search, GAs, microchanonic annealing, and microchanonic optimization [23]. Scheduling problems are usually addressed by heuristic techniques due to their structure and complexity [22]. For instance, authors in [23] state that the simulated annealing scheme provides an efficient solution when applied to scenarios that require some form of programming. Several authors present comparisons and evaluations of the performance of different metaheuristics in solving scheduling problems [24]. Some of the common metaheuristic strategies include evolutionary algorithms, ant colony, local search, simulated annealing, and tabu search, but none of them can be singled out as the best. Table II shows a summary of some of these algorithms, including their advantages, disadvantages, and tools that can be applied to develop them. Using this Table, it is possible to identify the methodologies that can be applied to develop a solution to the university timetabling problem effectively. The problem of scheduling can be found in many areas, e.g. in the assignment of drivers to vehicles in a public transport company or in the development of school timetables. However, conflicts are common due to competing interests among the stakeholders involved and the existence of constraints that should be satisfied. In order to find the best possible solution to such issues, a swarm-intelligent algorithm, named Particle Swarm Optimization (PSO), can be used. This meta-heuristic was developed to find and optimize solutions for continuous problems. Nonetheless, it turned out to be a very suitable method for achieving quick and good solutions within these applications. At this point, however, the process must be adapted to the given circumstances for this discrete problem.

TABLE II. COMPARISON OF DIFFERENT SCHEDULING AND OPTIMIZATION ALGORITHMS

Reference	Year	Technique	Tools	Advantages	Disadvantages
[25]	2015	PSO	Programming languages, such as Python. Supported by Gaussian 09 program package.	Simple to implement.	Does not converge fast.
[26]	2017	Integer programming	Various solver packages for global optimization.	Few parameters that need to be adjusted.	Difficult to define the initial design parameters.
[27]	2016	Multi-objective optimization	Can be implemented using various programming languages.	Able to run parallel computation.	Can converge prematurely.
[28]	2016	Hyper-heuristics	Python or R programming languages.	Can be robust.	Complex to implement for inexperienced programmers.
[29]	2017	Integer programming	Solver packages, such as BARON.	Have higher probability and efficiency in finding the highest optimization	Cannot work out problems of scattering.
[20]	2015	Metaheuristic techniques	Python or R programming languages.	Can converge fast	Dos not have short computational time.
[32]	2018	Fix-and-optimize metaheuristic	Python or R programming languages.	Does not overlap and mutate	Not efficient in working problems linked with scattering.
[32]	2015	Great deluge algorithm	Matlab	Has short computational time.	Complex to implement for inexperienced programmers.
[33]	2016	Linear integer model	Solver packages, such as GAMS library.	Efficient in solving problems that do not have accurate mathematical models.	Complex to implement for inexperienced programmers.
[34]	2017	Artificial bee colony algorithm	Matlab	It is easy to use and can interface with other algorithms efficiently.	Converges prematurely, resulting in no solutions in some instances.
[35]	2017	Stochastic gradient descent	Stochastic simulation toolkit (StochKit)	Simple to implement and works fast when applied to small datasets.	Hyperparameters need to be tuned manually. Hence, people not familiar with it find it challenging to use.
[36]	2018	Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm	The libLBFGS library written in the C++.	Has better convergence than most algorithms	Performs poorly in the context of non-smooth optimization.
[37]	2020	Simulated annealing algorithm.	Software tools, such as KDSimStudio and Simulated Conversation Development tool.	Gradually converges to a global optimal solution and, thereby, escapes from a local solution.	Requires enhancements to function effectively.
[38]	2020	Tabu search algorithm	Available frameworks, such as the Tabu Search Framework written in C++.	Together with a memory concept, it can be utilized to explore iteration problems more precisely than other algorithms.	It performs poorly in large dimension problems.
[39]	2020	Tabu search algorithm	Frameworks, such as the Tabu Search Framework.	Ignores recently explored neighborhoods to avoid settling on a local optimum solution.	As it does not have memory, it requires other algorithms or additional components to function effectively.
[40]	2020	Elite immune ant colony optimization algorithm	Matlab	Combines the strengths of both the ant colony and immune theory to ensure its efficiency.	Complex to implement.

A. Genetic Algorithms

The main task of scheduling algorithms in solving the university course timetabling problem is to find a solution that satisfies a set of restrictions. These limitations are either essential or nonessential. Essential restrictions are those that generate an unworkable timetable if they are not satisfied. On the other hand, the nonessential ones improve the quality of the framework. GAs are a strategy class that can be employed to resolve these conditions efficiently. They are based on the principles of genetics and natural selection, with the central idea being the selection of the fittest individuals in a population, who then recombine with others or mutate into new forms to generate new groups [41]. GAs are particularly applied to complex optimization problems, which are challenges that have different parameters or characteristics that need to be combined in search of the best solution and, at the same time, cannot be represented mathematically [42, 43]. Figure 1 illustrates the flowchart of the GA algorithm, from the initial population to when a suitable result is obtained.

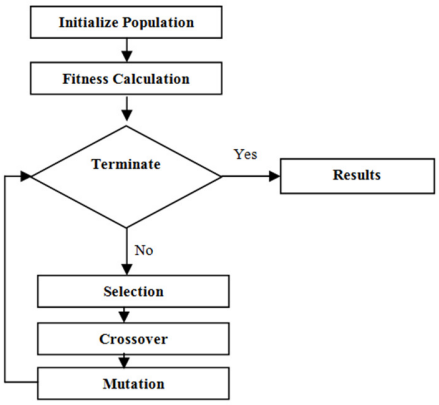

Fig. 1. GA flowchart [41].

Over the years, several researchers have applied GAs to solve such problems. For instance, authors in [44] dealt applied the GA to address the issue of scheduling class timetables. The differential feature in this work is the use of crossover operators only between rows and, later, columns. Many authors also utilize several methods to treat infective solutions, which are those that do not meet the essential restrictions. In [45], the algorithm was applied to resolve the problem in two phases. First, a GA was applied to generate solutions that only meet the essential restrictions, and, in the second stage, based on the initial population generated in the initial step, GAs were employed to satisfy the non-essential conditions. In [46], a greedy approach was proposed to generate a robust initial population. Subsequently, the authors applied the GA to optimize this solution further. Other works, such as [47], propose the application of a repair function, which is introduced after crossover and mutation operations, to deal with infactible solutions. In [48], it is proposed to use local search methods to improve the solution, which, consequently, reduces the number of violations of non-essential restrictions. These methods are usually applied after the generation of each population in the evolution process. Often, GAs are implemented in the Delphi 5.0 programming environment and the problem input data are imported from the Saber Software Database.

The problem of timetabling consists of scheduling a sequence of classes between teachers and students in a limited period of time, while satisfying a set of restrictions. This problem exists in any educational institution, but it becomes increasingly complex as the size of the establishments grows. Primarily, this phenomenon arises because the timetabling of courses belongs to the NP-hard class, for which polynomial-time algorithms have been unable to obtain optimal solutions [49]. The timetabling problem has been studied for several years, with pioneering solutions forming the foundation for more effective techniques. The literature reveals several types of solutions, such as graph-based techniques and other approaches that find a maximal sequence of pairings in the bipartite graph composed of teachers and classes. Authors in [50] used programming by applying a Lagrangian relaxation technique, while authors in [23] formulated and solved the problem as an allocation challenge. Authors in [51] used flow resolution techniques in networks. As mentioned above, many authors divided the restrictions between hard and soft, where the hard restrictions define the feasible solutions, and the soft restrictions are included in the objective function. While satisfying the hard conditions means that the solution addresses the problem successfully, its quality is dependent on the procedure to blocks of different durations. Another popular approach is the employment of a multiobjective GA, which combines two points of view: student-oriented and teacher-oriented. Moreover, one can use a two-stage GRASP algorithm. Others formulated another model for the problem, also using soft constraints in the objective function. One can explore a hyperheuristics framework based on graphs, researching local search algorithms based on low-level coloring algorithms. Other approaches can also apply a tabu search, using domain-independent neighborhood structures to penalize neighborhoods unable to generate better solutions. A combination of the bee colony meta-heuristic with an assimilation policy to guide the search process has also been proposed. Therefore, there are several approaches that can be followed to solve the university course timetabling issue, although they all have their advantages and disadvantages.

B. Particle Swarm Optimization

PSO was originally developed in 1995 to solve continuous problems [52]. The idea of simulating a swarm of bees has evolved into a meta-heuristic and presented an optimization procedure with some basic extensions. Based on this premise, the PSO procedure can be further modified to solve combinatorial problems [53, 54]. The boids (bird-like objects) model has been introduced, whereby each boid represents an individual who obeys three implemented rules in order to simulate swarm behavior for computer graphics and films, since it was found that the choreography of flocks of birds was aesthetic. The alignment ensures that individuals move in the same direction as their neighbors. Merging several boids enables cohesion, in which a single boid moves in the middle of a defined group, whereas separation causes boids to diverge in a small space. Depending on a given rule, a marked boid is aligned with the other boids. Later on, rest areas were added to

the boids model [55]. The individuals also have a need to remain within the swarm. The more a boid approaches a resting place, the greater the desire to move towards it, pulling more boids in this direction. Later, Kennedy and Eberhart developed the Swarm Optimization [55]. Theoretically, individuals can benefit from the discoveries and past experiences of their colleagues when searching for food [56]. This advantage can become critical and outweigh the disadvantages of food competition if the resource is unpredictably patched [57]. Kennedy and Eberhart followed this hypothesis that information exchange between group members offers an evolutionary advantage in the further development of the PSO by implementing social behavior in the birds and, hence, making them mass- and collision-free particles. They also extended the model by not only having them look for a rest area but also using a "cornfield vector" on which the best feeding place exists. These particles now have a memory of their best position in relation to such a feeding place and a knowledge of the best position so far within the entire swarm [57]. Based on these considerations, the classic PSO for continuous problems was developed. Figure 2 shows the flowchart of the PSO algorithm.

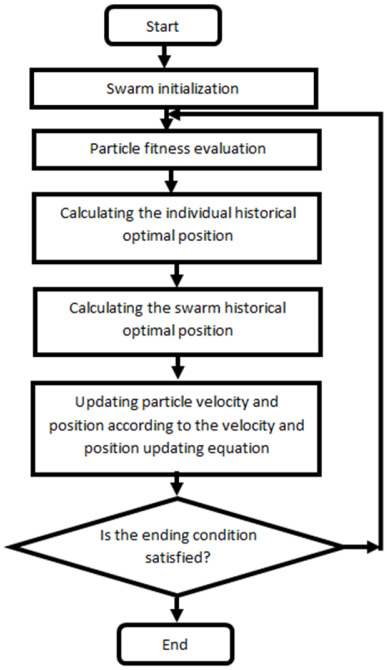

Fig. 2. Flowchart of the PSO algorithm.

To optimize the solution, the PSO developed for continuous problems was modified in such a way that it was suitable for solving combinatorial problems. Various compositions of parameters were examined for their performance and it was observed that the attempt to modify the PSO in such a way that it was able to solve a combinatorial problem without the characteristics of swarm-intelligence [59]. The PSO significantly improved the quality of the solution of the manually created timetable with reasonable computing effort. In particular, the influence of an individual event can be emphasized in combination with swapping a random time window of the current solution. This exercise suggests a good current solution [60]. This exercise suggests a good balance between exploration and exploration of the solution space. Since the method presented primarily deals with continuous problems, further research is needed to determine its suitability in addressing the university timetable problem.

While formulating a time table, it is important to consider the context of the problem within the educational institution in question. The problem may vary according to the institution and its business model, as, for example, in public and private educational establishments. Public schools usually have an open timetable model, in which subjects are offered and courses are built around them [60]. Thus, students are free to choose the subjects and they want to take and at what times. As the subjects are free, there is no requirement for rooms where it should be taught, and this allocation is also part of the process as a whole [59, 61]. Thus, it is possible for a course to be taught in several different classrooms throughout the week. In contrast, most private institutions follow a closed model. In this case, a course coordinator allocates the subjects that a student will take. Eventually, the learner has to approve the given set of units according to the level of study. In addition, the subjects are linked to courses they are being offered, and they are completely free as in the case of public institutions. As a result, subjects should not be allocated to classrooms, as they have a fixed location. That is, the classroom to which they are assigned. Thus, lecturers can switch rooms but not the student since teachers in private institutions can teach several subjects, although there is no requirement that a subject should always be taught by the same instructor. However, in public institutions, it is common for a teacher to be hired for one or more specific subjects.

III. DISCUSSION

Lesson planning deals with the scheduling of classes at a specific time in a given location, with particular participants that have to attend. In the best case, the solution of the problem results in plans that enable all students to take part in the meetings of interest at a certain location and time. There are, however, restrictions to the allocation of resources, which include time, space, and the availability of both teaching staff and the students. No other class should happen at the same time and venue with an ongoing one, and neither should two lessons demand the presence of the same group of students concurrently. Double occupancy of rooms has the same potential for conflict. As mentioned above, creating an efficient timetable is a typical NP-hard problem, which does not have a specific methodology of finding an optimal solution [62, 63]. Moreover, course timetables do not consider the preferences of students, as is generally the case in appointment requests.

In the case of university lesson planning, this phenomenon means that lecturers play the role of active agents and students that of passive agents. At the same time, a lecture hall has to be of appropriate size, should possess the required equipment, and be available at an acceptable time for all participants without causing violations of both essential and nonessential constraints [63]. Therefore, the dates of a resulting timetable correspond to

assignments of a previously defined matrix of possible time windows.

The conditions, which a timetable should satisfy can be divided into hard and soft constraints. Violations of hard constraints are not permitted because they could lead to the unavailability of lecturers, e.g. they might be scheduled to attend two concurrent classes. On the other hand, soft constraints can be left unsatisfied since such a case does not affect the admissibility of the timetable. The punishment for non-compliance with soft constraints is included in the assessment of the timetable [14]. Soft constraints occur, for example, when deadline requests for events cannot be met. In this case, these incidents can be shifted from their original desired dates to another time window without affecting the effectiveness of the timetable.

The lesson planning activity in schools is further made difficult by the options that students have while selecting the classes to attend. For instance, one does not have to stick to given subject combinations, but can largely plan study programs independently. The attendance of individual courses does not have to take place in a particular semester, but can usually be integrated into the study period at will. The aim of university timetable scheduling is to use the resulting multitude of subject combinations to create a program, which is as free of overlaps as possible for all [64, 65]. Although the problems of school and course timetabling may appear similar at a first glance, they present significantly different problem sets, with the latter's being more complex than those of the former.

While formulating a timetable, it is important to consider the context of the problem faced by the educational institution in question. The issue may vary depending on the establishment and its business model. For example, while they are both institutions of higher learning, public and private universities have different funding models, which affect how they offer their classes and the types of services their students enjoy. Public schools usually have an open structure, in which subjects are offered and courses are built around them, while students are free to choose which subjects they want to take and at what time. As the subjects are free, there is no requirement for rooms where they should be taught, and this allocation is also part of the process as a whole [66]. Thus, it is possible for a course to be taught in several different classrooms throughout the week. On the other hand, most private institutions follow a closed model. In this case, a course coordinator allocates the subjects that a student will take. Eventually, the student has to approve the given set of units according to the level of study. In addition, the subjects are linked to courses in which they are being offered, and they are not completely free as in the case of public institutions. As a result, subjects should not be allocated to classrooms, as they have a fixed location. That is, the classroom to which they are assigned. Thus, lecturers can switch rooms but not the student since teachers in private institutions can teach several subjects, although there is no requirement that a subject should always be taught by the same instructor. However, in public institutions, it is common for a teacher to be hired for one or more specific subjects.

University course timetabling solves the problem of creating assignments given specific time slots, classrooms, teachers, and subjects. The goal is to prevent a room from being occupied by two subjects simultaneously, or two teachers teaching the same class, or a teacher to have to teach two classes at the same time. The quality indicator of a solution varies, with some researchers using the sum of the preferences of a given discipline, such as a teacher allocation, to measure the effectiveness of a timetable [66, 67]. Others adopt soft restrictions and utilize them to assess the quality of the solution, while there are instances where student preferences or the spacing between classes has been employed to achieve the same goal. Authors in [4] consider a matrix of conflicts between learners taking two courses i and j simultaneously. The objective function considers the minimization of the number of cases where i and j are scheduled at the same time. Therefore, just as there are several methods of creating an optimal solution to the course timetabling problem, the quality of the resulting solutions can be evaluated in various ways.

IV. CONCLUSION

The problem of planning school hours is described as having to schedule a series of meetings between teachers and students over a set period of time and meet a number of different types of constraints. Since the types of restrictions generally change from one institution to another, different solutions to this problem have been proposed. The problem of school hours, it is a problem of combinatorial optimization with a vast search space and with a generally large number of restrictions. It is considered as an NP-hard problem. For problems like these, there is still no algorithm that tests all the possibilities to find the optimum solution in a timely manner. Therefore, this problem has been approached through heuristic techniques, and more recently by meta-heuristics.

REFERENCES

[1] N. L. A. Aziz and N. A. H. Aizam, "A brief review on the features of university course timetabling problem," *AIP Conference Proceedings*, vol. 2016, no. 1, Sep. 2018, Art. no. 020001, https://doi.org/10.1063/1.5055403.

[2] D. M. Premasiril, "University Timetable Scheduling Using Genetic Algorithm Approach Case Study: Rajarata University OF Sri Lanka," *Journal of Engineering Research and Application*, vol. 8, no. 12, pp. 30–35, 2018.

[3] E. A. Okonji, Y. M. Oluwatoyin, and O. I. Patricia, "Intelligence Classification of the Timetable Problem: A Memetic Approach," *International Journal on Data Science and Technology*, vol. 3, no. 2, pp. 24–33, 2017.

[4] R. R. Iwankowicz and M. Taraska, "Self-classification of assembly database using evolutionary method," *Assembly Automation*, vol. 38, no. 3, pp. 268–281, Jan. 2018, https://doi.org/10.1108/AA-06-2017-071.

[5] W. Tian, W. Guo, and M. He, "On the classification of NP complete problems and their duality feature," *International Journal of Computer Science & Information Technology*, vol. 10, no. 1, pp. 67–78, 2018.

[6] J. Soyemi, J. L. Akinode, and S. A. Oloruntoba, "Automated Lecture Time-tabling System for Tertiary Institutions," *International Journal of Applied Information Systems*, vol. 12, no. 5, pp. 20–27, 2017.

[7] A. E. Phillips, C. G. Walker, M. Ehrgott, and D. M. Ryan, "Integer programming for minimal perturbation problems in university course timetabling," *Annals of Operations Research*, vol. 252, no. 2, pp. 283–304, May 2017, https://doi.org/10.1007/s10479-015-2094-z.

[8] M. F. Syahputra *et al.*, "Genetic algorithm to solve the problems of lectures and practicums scheduling," *IOP Conference Series: Materials*

Science and Engineering, vol. 308, Feb. 2018, Art. no. 012046, https://doi.org/10.1088/1757-899X/308/1/012046.

[9] R. A. Oude Vrielink, E. A. Jansen, E. W. Hans, and J. van Hillegersberg, "Practices in timetabling in higher education institutions: a systematic review," *Annals of Operations Research*, vol. 275, no. 1, pp. 145–160, Apr. 2019, https://doi.org/10.1007/s10479-017-2688-8.

[10] O. Y. M. Al-Rawi and T. Mukherjee, "Application of Linear Programming in Optimizing Labour Scheduling," *Journal of Mathematical Finance*, vol. 9, no. 3, pp. 272–285, Jun. 2019, https://doi.org/10.4236/jmf.2019.93016.

[11] I. R. Cassar, N. D. Titus, and W. M. Grill, "An improved genetic algorithm for designing optimal temporal patterns of neural stimulation," *Journal of Neural Engineering*, vol. 14, no. 6, Nov. 2017, Art. No. 066013, https://doi.org/10.1088/1741-2552/aa8270.

[12] A. Hussain, Y. S. Muhammad, and A. Nawaz, "Optimization through genetic algorithm with a new and efficient crossover operator," *International Journal of Advances in Mathematics*, vol. 2018, no. 1, pp. 1–14, 2018.

[13] S. Muniyappan and P. Rajendran, "Contrast Enhancement of Medical Images through Adaptive Genetic Algorithm (AGA) over Genetic Algorithm (GA) and Particle Swarm Optimization (PSO)," *Multimedia Tools and Applications*, vol. 78, no. 6, pp. 6487–6511, Mar. 2019, https://doi.org/10.1007/s11042-018-6355-0.

[14] M. Posada, H. Andersson, and C. H. Hall, "The integrated dial-a-ride problem with timetabled fixed route service," *Public Transport*, vol. 9, no. 1, pp. 217–241, Jul. 2017, https://doi.org/10.1007/s12469-016-0128-9.

[15] R. Ganguli and S. Roy, "A Study on Course Timetable Scheduling using Graph Coloring Approach," *International Journal of Computational and Applied Mathematics*, vol. 12, no. 2, pp. 469–485, 2017.

[16] P. Kenekayoro, "Incorporating Machine Learning to Evaluate Solutions to the University Course Timetabling Problem," *Covenant Journal of Informatics and Communication Technology*, vol. 7, no. 2, pp. 18–35, Dec. 2019.

[17] A. Malikov, V. Voronkin, and S. Pevchenko, "Software and hardware infrastructure for timetables scheduling in university," *CEUR Workshop Proceedings*, vol. 2254, pp. 191–198, 2018.

[18] C. Kalu, S. Ozuomba, and S. I. Umana, "Development of Mechanism for Handling Conflicts and Constraints in University Timetable Management System," in *Communications on Applied Electronics*, vol. 7, 24 vols., New York, USA, 2018, pp. 22–32.

[19] O. Abayomi-Alli, A. Abayomi-Alli, S. Misra, R. Damasevicius, and R. Maskeliunas, "Automatic Examination Timetable Scheduling Using Particle Swarm Optimization and Local Search Algorithm," in *Data, Engineering and Applications*, R. K. Shukla, J. Agrawal, S. Sharma, and G. Singh Tomer, Eds. Singapore: Springer, 2019, pp. 119–130.

[20] P. G. Daniel, A. O. Maruf, and B. Modi, "Paperless Master Timetable Scheduling System," *International Journal of Applied Science and Technology*, vol. 8, no. 2, pp. 58–68, Jun. 2018, https://doi.org/10.30845/ijast.v8n2p7.

[21] A. M. Hambali, Y. A. Olasupo, and M. Dalhatu, "Automated university lecture timetable using Heuristic Approach," *Nigerian Journal of Technology*, vol. 39, no. 1, pp. 1–14, Apr. 2020, https://doi.org/10.4314/njt.v39i1.1.

[22] K. Yonekura and Y. Kanno, "A Heuristic Method Using Hessian Matrix for Fast Flow Topology Optimization," *Journal of Optimization Theory and Applications*, vol. 180, no. 2, pp. 671–681, Feb. 2019, https://doi.org/10.1007/s10957-018-1404-4.

[23] E. D. Taillard, "Heuristic Methods for Large Centroid Clustering Problems," *Journal of Heuristics*, vol. 9, no. 1, pp. 51–73, Jan. 2003, https://doi.org/10.1023/A:1021841728075.

[24] E.-G. Talbi, "Combining metaheuristics with mathematical programming, constraint programming and machine learning," *4OR*, vol. 11, no. 2, pp. 101–150, 2013.

[25] S. Larabi Marie-Sainte, "A survey of Particle Swarm Optimization techniques for solving university Examination Timetabling Problem," *Artificial Intelligence Review*, vol. 44, no. 4, pp. 537–546, Dec. 2015, https://doi.org/10.1007/s10462-015-9437-7.

[26] A. E. Phillips, C. G. Walker, M. Ehrgott, and D. M. Ryan, "Integer programming for minimal perturbation problems in university course timetabling," *Annals of Operations Research*, vol. 252, no. 2, pp. 283–304, May 2017, https://doi.org/10.1007/s10479-015-2094-z.

[27] K. S. Abdallah, "Multi-objective Optimization for Exam Scheduling to Enhance the Educational Service Performance," *The Journal of Management and Engineering Integration*, vol. 9, no. 1, pp. 14–23, 2016.

[28] N. Pillay, "A review of hyper-heuristics for educational timetabling," *Annals of Operations Research*, vol. 239, no. 1, pp. 3–38, Apr. 2016, https://doi.org/10.1007/s10479-014-1688-1.

[29] A. Cataldo, J.-C. Ferrer, J. Miranda, P. A. Rey, and A. Saure, "An integer programming approach to curriculum-based examination timetabling," *Annals of Operations Research*, vol. 258, no. 2, pp. 369–393, Nov. 2017, https://doi.org/10.1007/s10479-016-2321-2.

[30] C. K. Teoh, A. Wibowo, and M. S. Ngadiman, "Review of state of the art for metaheuristic techniques in Academic Scheduling Problems," *Artificial Intelligence Review*, vol. 44, no. 1, pp. 1–21, Jun. 2015, https://doi.org/10.1007/s10462-013-9399-6.

[31] M. Lindahl, M. Sørensen, and T. R. Stidsen, "A fix-and-optimize matheuristic for university timetabling," *Journal of Heuristics*, vol. 24, no. 4, pp. 645–665, Aug. 2018, https://doi.org/10.1007/s10732-018-9371-3.

[32] M. N. Mohmad Kahar and G. Kendall, "A great deluge algorithm for a real-world examination timetabling problem," *Journal of the Operational Research Society*, vol. 66, no. 1, pp. 116–133, Jan. 2015, https://doi.org/10.1057/jors.2012.169.

[33] V. Pereira and G. C. Helder, "Linear Integer Model for the Course Timetabling Problem of a Faculty in Rio de Janeiro," *Advances in Operations Research*, vol. 2016, no. 1–9, 2016, Art. no. 7597062.

[34] S. Anam, "Multimodal optimization by using hybrid of artificial bee colony algorithm and BFGS algorithm," *Journal of Physics: Conference Series*, vol. 893, Oct. 2017, Art. no. 012068, https://doi.org/10.1088/1742-6596/893/1/012068.

[35] M. M. Najafabadi, T. M. Khoshgoftaar, F. Villanustre, and J. Holt, "Large-scale distributed L-BFGS," *Journal of Big Data*, vol. 4, no. 1, Jul. 2017, Art. no. 22, https://doi.org/10.1186/s40537-017-0084-5.

[36] J. Guo and A.S. Lewis. "Rescaling nonsmooth optimization using BFGS and Shor updates." 2018.

[37] L. Su, Y. Xu, Y. Yuan, and J. Yang, "Combining Pixel Swapping and Simulated Annealing for Land Cover Mapping," *Sensors*, vol. 20, no. 5, Jan. 2020, Art. no. 1503, https://doi.org/10.3390/s20051503.

[38] A.-R. Hedar, W. Deabes, M. Almaraashi, and H. H. Amin, "Evolutionary Algorithms Enhanced with Quadratic Coding and Sensing Search for Global Optimization," *Mathematical and Computational Applications*, vol. 25, no. 1, Mar. 2020, Art. no. 7, https://doi.org/10.3390/mca25010007.

[39] J. Tassone, G. Pond, and S. Choudhury, "Algorithms for optimizing fleet staging of air ambulances," *Array*, vol. 7, Sep. 2020, Art. no. 100031, https://doi.org/10.1016/j.array.2020.100031.

[40] M. Xu and J. Zhou, "Elite Immune Ant Colony Optimization-Based Task Allocation for Maximizing Task Execution Efficiency in Agricultural Wireless Sensor Networks," *Journal of Sensors*, vol. 2020, Jan. 2020, Art. no. 3231864, https://doi.org/10.1155/2020/3231864.

[41] M. Jalal, A. K. Mukhopadhyay, and Z. Grasley, "Design, manufacturing, and structural optimization of a composite float using particle swarm optimization and genetic algorithm," *Proceedings of the Institution of Mechanical Engineers, Part L: Journal of Materials: Design and Applications*, vol. 233, no. 7, pp. 1404–1418, Jul. 2019, https://doi.org/10.1177/1464420718755546.

[42] M. Rezaei, M. Akbarpour Shirazi, and B. Karimi, "IoT-based framework for performance measurement: A real-time supply chain decision alignment," *Industrial Management & Data Systems*, vol. 117, no. 4, pp. 688–712, Jan. 2017, https://doi.org/10.1108/IMDS-08-2016-0331.

[43] H. Jafarzadeh, N. Moradinasab, and M. Elyasi, "An Enhanced Genetic Algorithm for the Generalized Traveling Salesman Problem," *Engineering, Technology & Applied Science Research*, vol. 7, no. 6, pp. 2260–2265, Dec. 2017, https://doi.org/10.48084/etasr.1570.

[44] R. Perez-Rodriguez and A. Hernandez-Aguirre, "A hybrid estimation of distribution algorithm for flexible job-shop scheduling problems with process plan flexibility," *Applied Intelligence*, vol. 48, no. 10, pp. 3707–3734, Oct. 2018, https://doi.org/10.1007/s10489-018-1160-z.

[45] C. B. Kalayci, O. Polat, and S. M. Gupta, "A hybrid genetic algorithm for sequence-dependent disassembly line balancing problem," *Annals of Operations Research*, vol. 242, no. 2, pp. 321–354, Jul. 2016, https://doi.org/10.1007/s10479-014-1641-3.

[46] S. Acid, L. M. de Campos, and M. Fernandez, "Score-based methods for learning Markov boundaries by searching in constrained spaces," *Data Mining and Knowledge Discovery*, vol. 26, no. 1, pp. 174–212, Jan. 2013, https://doi.org/10.1007/s10618-011-0247-5.

[47] R. L. Burdett and E. Kozan, "A sequencing approach for creating new train timetables," *OR Spectrum*, vol. 32, no. 1, pp. 163–193, 2010, https://doi.org/10.1007/s00291-008-0143-6.

[48] N. Liu, D. Plets, S. K. Goudos, L. Martens, and W. Joseph, "Multi-objective network planning optimization algorithm: human exposure, power consumption, cost, and capacity," *Wireless Networks*, vol. 21, no. 3, pp. 841–857, Apr. 2015, https://doi.org/10.1007/s11276-014-0822-y.

[49] M. Ehrgott, C. Guler, H. W. Hamacher, and L. Shao, "Mathematical optimization in intensity modulated radiation therapy," *Annals of Operations Research*, vol. 175, no. 1, pp. 309–365, Mar. 2010, https://doi.org/10.1007/s10479-009-0659-4.

[50] V. Grout, J. McGinn, and J. Davies, "Real-time optimisation of access control lists for efficient Internet packet filtering," *Journal of Heuristics*, vol. 13, no. 5, pp. 435–454, Oct. 2007, https://doi.org/10.1007/s10732-007-9019-1.

[51] V. K. Kaishev, D. S. Dimitrova, S. Haberman, and R. J. Verrall, "Geometrically designed, variable knot regression splines," *Computational Statistics*, vol. 31, no. 3, pp. 1079–1105, Sep. 2016, https://doi.org/10.1007/s00180-015-0621-7.

[52] E. Elbeltagi, M. Ammar, H. Sanad, and M. Kassab, "Overall multiobjective optimization of construction projects scheduling using particle swarm," *Engineering, Construction and Architectural Management*, vol. 23, no. 3, pp. 265–282, Jan. 2016, https://doi.org/10.1108/ECAM-11-2014-0135.

[53] M. Spiliotis, L. Mediero, and L. Garrote, "Optimization of Hedging Rules for Reservoir Operation During Droughts Based on Particle Swarm Optimization," *Water Resources Management*, vol. 30, no. 15, pp. 5759–5778, Dec. 2016, https://doi.org/10.1007/s11269-016-1285-y.

[54] T. M. Kumar and N. A. Singh, "Environmental Economic Dispatch with the use of Particle Swarm Optimization Technique based on Space Reduction Strategy," *Engineering, Technology & Applied Science Research*, vol. 9, no. 5, pp. 4605–4611, Oct. 2019, https://doi.org/10.48084/etasr.2969.

[55] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *International Conference on Neural Networks*, Perth, WA, Australia, Dec. 1995.

[56] J.-S. Pan, Z. Meng, S.-C. Chu, and H.-R. Xu, "Monkey King Evolution: an enhanced ebb-tide-fish algorithm for global optimization and its application in vehicle navigation under wireless sensor network environment," *Telecommunication Systems*, vol. 65, no. 3, pp. 351–364, Jul. 2017, https://doi.org/10.1007/s11235-016-0237-4.

[57] Z. Jiang, Q. Lin, K. Shi, and W. Pan, "A novel PGSA–PSO hybrid algorithm for structural optimization," *Engineering Computations*, vol. 37, no. 1, pp. 144–160, Jan. 2019, https://doi.org/10.1108/EC-01-2019-0025.

[58] G. Lin and J. Guan, "A Binary Particle Swarm Optimization for the Minimum Weight Dominating Set Problem," *Journal of Computer Science and Technology*, vol. 33, no. 2, pp. 305–322, Mar. 2018, https://doi.org/10.1007/s11390-017-1781-4.

[59] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: an overview," *Soft Computing*, vol. 22, no. 2, pp. 387–408, Jan. 2018, https://doi.org/10.1007/s00500-016-2474-6.

[60] Y. Ning, Z. Peng, Y. Dai, D. Bi, and J. Wang, "Enhanced particle swarm optimization with multi-swarm and multi-velocity for optimizing high-dimensional problems," *Applied Intelligence*, vol. 49, no. 2, pp. 335–351, Feb. 2019, https://doi.org/10.1007/s10489-018-1258-3.

[61] U. Nagalingam, B. Mahadevan, K. Vijayarajan, and A. P. Loganathan, "Design optimization for cogging torque mitigation in brushless DC motor using multi-objective particle swarm optimization algorithm," *COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, vol. 34, no. 4, pp. 1302–1318, Jan. 2015, https://doi.org/10.1108/COMPEL-07-2014-0162.

[62] M. Nouiri, A. Bekrar, A. Jemai, S. Niar, and A. C. Ammari, "An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem," *Journal of Intelligent Manufacturing*, vol. 29, no. 3, pp. 603–615, Mar. 2018, https://doi.org/10.1007/s10845-015-1039-3.

[63] R. J. Acosta-Amado, M. Villa-Marulanda, A. Garcia-Diaz, I. Atuahene, and I. C. Lacera-Cortes, "Planning Personnel Timetable: A Network Model for an Academic Application," in *Industrial & Systems Engineering Research Conference*, Nashville,Tennessee, Jun. 2015, pp. 2656–2663.

[64] A. (Avi) Ceder, S. Hassold, and B. Dano, "Approaching even-load and even-headway transit timetables using different bus sizes," *Public Transport*, vol. 5, no. 3, pp. 193–217, Oct. 2013, https://doi.org/10.1007/s12469-013-0062-z.

[65] J. Joy and T. Nambirajan, "Automating Time Table Planning Process: Participatory Action Research," *SCMS Journal of Indian Management*, vol. 15, no. 4, pp. 32–51, 2018.

[66] O. Ullrich, D. Lückerath, and E. Speckenmeyer, "Do regular timetables help to reduce delays in tram networks? It depends!," *Public Transport*, vol. 8, no. 1, pp. 39–56, Mar. 2016, https://doi.org/10.1007/s12469-015-0115-6.

[67] M. A. Al-Betar and A. T. Khader, "A harmony search algorithm for university course timetabling," *Annals of Operations Research*, vol. 194, no. 1, pp. 3–31, Apr. 2012, https://doi.org/10.1007/s10479-010-0769-z.