Multi-objective fuzzy-based adaptive memetic algorithm with hyper-heuristics to solve university course timetabling problem

Abdul Ghaffar[1], Mian Usman Sattar[2,*], Mubbasher Munir[3], and Zarmeen Qureshi[4]

[1] Department of Information Systems, University of Management and Technology, Lahore, Pakistan.
Email: abdul.ghafar@umt.edu.pk
[2]Assistant Professor, Department of Management Sciences, Beaconhouse National University, Lahore, Pakistan.
Email: usman.sattar@bnu.edu.pk
[3] Department of Economics and Statistics, University of Management and Technology, Lahore, Pakistan.
Email: mubbasher.munir@umt.edu.pk
[4]Department of Management Sciences, School of Business, Beaconhouse National University, Lahore, Pakistan.
Email: zarmeen.qureshi25@gmail.com

Abstract

The university course timetabling is an NP-hard (non-deterministic polynomial-time hard) optimization problem to create a course timetable without conflict. It must assign a set of subject classes to a fixed number of timeslots with physical resources, including rooms and teachers. Avoiding hard constraints creates an executable timetable, whereas the removal of different soft constraints creates a satisfactory timetable. The most common way to resolve this problem is through the use of a hybrid genetic algorithm. The multi-objective fuzzy-based adaptive memetic algorithm, a population-based hybrid genetic approach, is proposed by combining genetic algorithm with local search with tabu search and various artificial intelligence techniques. It starts with generating a random population by using the hyper-heuristics and initial repairing method. By using the hill-climbing algorithm, it iteratively generates new offspring from the population by applying fuzzy-based adaptive crossover and mutation operations. If the solution still contains some conflicts, then the tabu search improves it by applying the most appropriate candidate repeatedly. While getting the workable solution, the algorithm tries to maximize multiple objective functions to get manageable solutions with different perspectives. It efficiently allocates all the required resources to subject classes and generates optimal solutions for the datasets provided by the University of Management & Technology, Lahore. It shows 96.29% accuracy in resolving conflicts compare with that of the simple and hybrid genetic algorithms. A web-based dynamic timetable manager visually represents a timetable and also provides options to adjust conflicts manually.

Keywords: Timetabling, Memetic Algorithm, Hybrid Genetic Algorithm, Hyper Heuristics, Tabu Search, Fuzzy Logic.

Received on 08 July 2021, accepted on 28 November 2021, published on 16 December 2021

doi: 10.4108/eai.16-12-2021.172435

1. Introduction

The timetable is an essential and crucial document in any educational institute to manage classes of various subjects by allocating resources at the maximum level with minimum conflicts among them. It is a timetabling problem that has been defined first by Gotlib [1] as allocating time and space to settle a meeting between student and teacher for fixed timeslots while satisfying different constraints. Constraints, a set of rules, must be satisfied to create an optimal and workable solution for the university course timetabling problem [2]. He also describes that the optimal solution must satisfy all the hard constraints and minimize all medium and soft constraints.

[*]Corresponding author: usman.sattar@bnu.edu.pk

Most researchers categorize the timetabling problem into the school timetabling problem and university timetabling problems. The university timetabling problem is further categorized into the course timetabling problem and examination timetabling problem [1]. The university course timetabling problem is considered as an NP-hard (non-deterministic polynomial-time hard) optimization problem that does not solve in polynomial time [2]. It is simply to allocate and distribute resources such as timeslot, room, and teacher by satisfying different hard and soft constraints to accommodate many subject classes [3]. It has more hard and soft constraints compared with those of the university examination timetabling problem. An optimized solution must satisfy these constraints by allocating limited resources to a fixed number of timeslots to accommodate many subject classes. Various searches apply many techniques to locate a workable and optimal solution to execute a timetable without conflict. However, it is not possible to get a fully optimized solution for this problem, as described by [4] .

Heuristic approaches try to find the optimal solution by using some defined simple rules to search sequentially, but they cannot find the best solution to a problem [5]. These algorithms depend upon the defined heuristics and always return the same result and performance, as they find out every solution with those defined heuristics [2]. Optimization approaches such as graph theory[6], integer programming/linear programming [7], and constraint satisfaction programming [8], solve the university course timetabling problem by applying various heuristics. These approaches never return an efficient solution because of the complexity of satisfying various constraints in the timetabling problem [3]. So different intelligent algorithms reduce the computation of work and return a solution closer to a global optimal solution for complex combinational optimization and NP-hard problems.

Meta-heuristics approaches are more efficient and intelligent than the heuristic approaches during selecting a suitable process to solve course timetabling problems [2]. These approaches are categorized into local search and population-based approaches [9]. The local search normally refines the single solution repeatedly based on a pre-defined set of constraints and optimizes the solution. The population-based approaches perform various processes iteratively to get an optimal candidate from a set of candidates. He also describes single solution algorithm that lead toward local optima. Tabu search [10] , simulated annealing [11], and variable neighborhood search [12] are common single solution-based approaches. Tabu search, a famous local search meta-heuristic technique, starts with an initial solution and tries to spot out an optimal solution by exploring the candidate solutions iteratively [5]. It is effective in removing hard constraints in less time and tries to exploit all the candidates to get the optimal solution in a vertical direction. Therefore, it will be expensive for the whole population. Genetic algorithm[12] , memetic algorithm [13], artificial bee colony [14], and particle swarm optimization [3] are common population-based techniques.

Hyper-heuristics approaches can optimize a solution with more performance compared with those of meta-heuristics approaches [15], and they categorize hyper-heuristics into constructive and perturbative heuristics. These approaches can optimize the timetabling problem by generating low-level heuristics both in the development of an initial population and in the refinement of the population [16]. Construction heuristics develop an initial solution to eliminate the maximum of the constraints in the beginning, whereas perturbative heuristics increase the quality of the same solution iteratively.

The use of heuristics and meta-heuristics approaches is further improved by using artificial intelligence (AI), memetic, and hybrid approaches. The hybrid approach [1][17][18], memetic approach [14], and various AI techniques such as Fuzzy theory [19] and the neural network with different optimization algorithms optimize the timetabling problem in a very robust and intelligent way.

All the optimization techniques normally minimize a single cost function, whereas multi-objective optimization techniques [20] seek to optimize a problem with multiple perspectives. These algorithms divide the cost function to calculate multiple objective functions and try to minimize their value as per the importance of those objectives.

The proposed algorithm in this paper, the multi-objective fuzzy-based adaptive memetic algorithm (MO-FAMA), is a population-based genetic algorithm with local searches that is an effective global optimization solution compared with simple optimization. It tries to find an optimal solution by using the fuzzy logic rule base to create adaptive operations with different hyper-heuristics. It implements various techniques including genetic algorithm, tabu search, hill-climbing, hyper-heuristics, and fuzzy logic. To optimize the timetabling solution with different perspectives, it also implements four objective functions in calculating the fitness of a solution. These objectives include hard fitness objectives and soft fitness objectives with the student, teacher, and management perspectives.

The objective of this study is to provide an algorithm for university course timetabling problems to create a conflict-free timetable by allocating all of the resources at their optimal level to fulfill the need of the student, teacher as well as management. The proposed algorithm could be the more realistic and dynamic approach to find an optimal solution, which resolves all of the conflicts before the actual implementation of the timetable.

The remainder of this paper is structured as follows. Details about the techniques used in the proposed algorithm to solve the university course timetabling problem in Section 2. The problem statement describes the details about the problem discussed with the proposed algorithm in section 3. The details about the proposed algorithm are available in Section 4. Section 5 presents the test plan specifications, followed by the result discussions in Section 6. The conclusion is presented in section 7.

2. Timetabling Optimization Techniques

The proposed optimization timetabling algorithm applies various optimization techniques to find the optimal solution. It tries to apply these techniques in a particular sequence and to implement different fuzzy logic controls to apply various hyper-heuristics to find the optimal solution for the timetabling problem. This section presents all the techniques that are used to implement the proposed algorithm. It includes creating an initial population, performing initial repairing, getting an optimal solution with a genetic algorithm, and improving the solution with local search [15].

The genetic algorithm (GA) is the most famous algorithm implemented by several researchers in optimizing problems and resolving timetabling problems. It was first implemented by Holland in 1969 [9], can find global solutions even in large and complex size search space. But it must not produce the best solution unless it combines with other techniques such as the hill-climbing algorithm used by Akkan & Gulcu[21] and Yusoff & Roslan [23]. A redesigned genetic algorithm, with altered crossover and mutation operators, is better at getting an optimized solution than the simple genetic algorithm [22]. As it never searches out all the solutions, so optimal solution may not be the best of all candidate solutions. Balan [23] uses a genetic algorithm with heuristics to generate an initial population to optimize the course timetabling problem. Tavakoli, Shirouyehzad, & Najafi [24] use various heuristics by defining various hard and soft constraints and implementing multiple stages with genetic algorithms and local search to solve the course timetabling problem.

In most cases, a greedy approach generates an initial population for the genetic algorithm, as implemented in [21] and [25]. Akkan & Gulcu [21] have used construction heuristics to construct an initial population. A Hybrid Immune Genetic Algorithm to Solve University Time Table Problems 2017) have performed an artificial immune system to initialize a population in the immune-genetic algorithm (IGA). Both algorithms perform well as compared with simple GA. Rjoub [26] claims the hill-climbing greedy algorithm performs better than a simple genetic algorithm and generates an optimal solution in less time.

Yongkai, Luo, & Liu [27] implement the genetic algorithm to solve the weekly course timetabling problem, and their algorithm produces better performance than those of other optimization techniques. The greedy and genetic fusion algorithm proposed by Wang, Shang, Liu, Lin, & Fu [27] that generates a high-quality initial population returns a fast convergence time as compared to a simple genetic algorithm. Soliman & Keshk [13] implement the genetic algorithm with multiple local searches by using a memetic approach, and it appears a significant improvement in getting a workable solution for the university course timetabling problem.

A hybrid algorithm with local search and GA is efficient in getting the local optimum and producing a more powerful algorithm to optimize the timetabling problem [18] describe the population-based optimization algorithms combine in different ways with different single solution-based algorithms to construct a hybrid approach to improve convergence time in solving the university course timetabling problem. Wang [28] implements a hybrid genetic algorithm (HGA) to solve the university classroom arrangement problem and shows its better performance at global search. A memetic hybrid algorithm is proposed by [4] with the use of parallel genetic algorithms by satisfying all the hard and soft constraints with local search. They test it with BenPaechter competition datasets and show better results. A memetic approach shows a fast convergence with an optimal solution in [29] that performs a global search with a genetic algorithm, which is further improved by local searches with simulated annealing and greedy random mutation with local search in GA.

Muklason, Irianti & Marom [15] have applied various hyper-heuristics with tabu search, whereas Rossi-Doria & Paechter [30] have implemented hyper-heuristics with a genetic algorithm. Both pieces of research claim a notable reduction of overall computational time with hyper-heuristics. Burke & Kendall [31]implement constructive heuristics to initialize a population with the hill-climbing by using tabu search, and then performs perturbative hyper-heuristics in GA to generate the best offspring. He identifies hyper-heuristics perform better than those of simple meta-heuristics techniques. Phased-approach with multiple objectives shows significant performance improvement with a modified genetic algorithm followed by using hyper-heuristic with a local search for the course timetabling problem [21] and [23]. Rezaeipanah, Samaneh & Ahmadi [17] propose a hybrid genetic algorithm with parallel genetic algorithms and local search with low-level heuristics to initialize an initial population in the first phase and show the initial repairing method works well than random initialization. Hyper-heuristics is implemented with tabu search and variable neighborhood search in the greedy algorithm, and it shows better performance of hyper-heuristics techniques are better than manual timetable settings [15].

June, Obit, Leau, Bolongkikit, & Alfred [32] implement AI techniques with fuzzy logic rule base to create fuzzy controlled genetic parameters in GA to improve timetabling optimization problems. Eludire and Akanbi[19] apply fuzzy logic with a genetic algorithm to schedule a timetabling problem by ranking different courses in order of difficulty level. Rjoub [26] proposes an application of the hill-climbing algorithm in various techniques of genetic algorithm for better performance of the genetic algorithm. Various intelligent techniques are used to customize mutation and crossover operators and it shows improvement in solution convergence [17]. A custom crossover technique is implemented with GA and shows a significant enhancement in performance [22].

Zuoshan and Yingbo [13] describe it would not be workable to minimize overall cost function by using multiple objectives to get an optimal solution and he applies it to minimize the number of clashes and periods in an examination timetable using an evolutionary algorithm with elitism strategy. Their approach shows better performance in getting a workable solution than single-objective approaches. Eludire & Akanbi [19] and Akkan & Gulcu [21] apply bi-objective hybrid genetic algorithms with local search to optimize the university course timetabling problem with less computation cost.

The proposed algorithm is an AJAX-based dynamic and user-friendly interactive client application that is developed with JavaScript, JQuery, PHP, and MySQL.The next section will describe details related to the proposed algorithm.

3. Problem statement

A timetable is a well-structured document to schedule classes of various subjects of a specific program/student group to several fixed timeslots. These timeslots are distributed in six working days from Monday to Saturday. Each weekday is further divided into seven timeslots of 90 minutes each. The fourth slot on Friday is declared off-slot because of the Jumma Prayer. So, there are 41 timeslots in a week to be allocated for all the required subject classes. Each timeslot is assigned to a fixed number of resources such as rooms, labs, or teachers to accommodate subject classes. A class is a functional group of students in a particular subject. Each class must allocate two timeslots and other resources in a week. Every resource Is allocated by following the associated constraints on them. Every teacher has a list of subjects to be taught and is available for a range of timeslots. Every room has a seating capacity to conduct a class, so it must be allocated to a class with certain registered students. Every room is categorized into a lecture room or laboratory, depending on the particular content of the subject. So, the proposed algorithm constructs a timetable by allocating available resources to a fixed number of timeslots by satisfying all the hard and soft constraints with student, teacher, and management perspectives as given in Table 1. Hard constraints are most important than soft constraints, so a higher weight of 100 is assigned to them, software constraints are divided into management, student, and teacher constraints which are assigned 50, 25, and 25 respectively as used in [28]. The required information must be loaded manually or through comma-separated values (CSV) files related to the resources with associated constraints and subject classes. Four different variations of genetic algorithms are used with a heuristic approach based on fuzzy logic and found memetic approach is better in optimizing multiple objectives.

Table 1. List of hard and soft constraints

Constraints	***Weight***	***Description***
	Hard constraints	
Room clash	100	Allocating a room for different classes in the same timeslot.
Subject clash	100	The timeslot is assigned to subjects of the same program.
Teacher clash	100	Teacher is assigned to different classes in the same timeslot.
Teacher subject clash	100	Teacher is assigned subjects not in his domain.
	Soft management constraints	
Program off-day	50	Class is allocated in an off-day of a program.
Invalid program slot	50	Class is assigned to invalid timeslots.
Room overload	50	Room capacity is less than class size.
Room under-load	50	Room capacity is more than class size.
Room invalid	50	Class is assigned to an invalid room type.
	Soft student constraints	
Consecutive class	25	No class of same subject in consecutive days
Same-day class gap	25	All classes of the same subject must be consecutive in a day.
	Soft teacher constraints	
Teacher overload	25	Teacher is assigned more classes than his limit.
Teacher invalid slot	25	Teacher is assigned in the undesirable timeslot.
Teacher under-load	25	Teacher is assigned less classes than his limit.

3.1 Timetable constraints

The proposed algorithm solves the university course timetabling problem within the domain of various constraints and the level of their severity into hard and soft constraints, as described [12]. The proposed algorithm implements more or less the same constraints as described by them. Hard constraints are physical limitations on the execution of a timetable and must be avoided for the smooth and workable execution of a timetable, i.e., room-timeslot conflicts, or subject slot allocations. Soft constraints are options to apply resources more effectively and desirably. They provide more flexibility to concerned stakeholders, so they have no issue with the execution of a timetable, i.e., teacher preference for timeslots, or convenience in attending lectures. The proposed algorithm categorizes soft constraints further into the student, teacher, and management perspectives. The proposed algorithm tries to avoid these constraints at the maximum level to optimize these objectives.

Different weights are suggested in for these constraints with the level of their severity, as hard constraints have more weight compared with those of soft constraints. Multiple objective functions processed these constraint violation weights to identify the fitness of a solution from various perspectives. A solution with maximum overall fitness value will be the workable solution against a solution with low fitness value.

3.2 Problem formalization

The university course timetabling problem, which is discussed in this paper, is formalized as in[12]. It includes a set of *n* classes related to a set of program/student group *p*, scheduled into a set of 41 timeslots n_s. Then it allocates resources from a set of rooms *r* with an associated feature set r_f and a set of teachers *t* with an associated feature set t_f by satisfying a set of constraints *c* to maximize objective functions given in a set *obj*.

$$n = \{n_1, n_2, n_3 ...\}$$
$$n_s = \{n_{s1}, n_{s2}, n_{s3} \dots\}$$
$$p = \{p_1, p_2, p_3 ...\}$$
$$r = \{r_1, r_2, r_3 ...\}$$
$$r_f = \{r_{f1}, r_{f2}, r_{f3} \dots\}$$
$$t = \{t_1, t_2, t_3 \dots\}$$
$$t_f = \{t_{f1}, t_{f2}, t_{f3} ...\}$$
$$c = \{h_1, h_2, h_3 ..., s_1, s_2, s_3 ...\}$$
$$obj = \{h, s_t, s_s, s_m\}$$

3.3 Population structure

A population is a set of random solutions to allocate different subject classes in a certain number of timeslots. Each solution in a population also called a chromosome, is a complete timetable. It divides each chromosome into subject classes. A single chromosome is a collection of genes that is an array of assigned resources of physical space and teacher to several fixed timeslots for a particular subject class, as shown in Fig. 1. Each class is a vector of day, timeslot, room, and teacher. There are two classes of every subject in a week, so a *4x2* size array is used to accommodate the used to accommodate vector of each subject class.

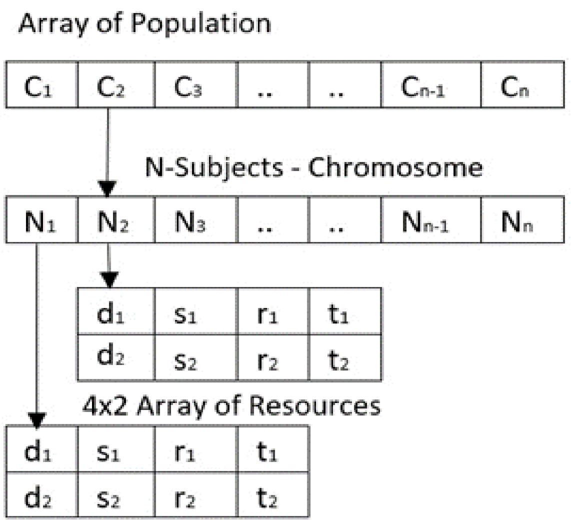


Figure 1. Population and chromosome structure

3.4 Hyper-heuristics

Hyper-heuristics are the changed form of the bin packing problem and graph coloring problem [15], as shown in Table 2. These are types of conflicts, the number of conflicts, and the size of fitness. These are applied while creating an initial population and in performing various genetic operations by using fuzzy logic. Hard constraints are the most important candidate early to be avoided. Then soft constraints are avoided in the sequence of soft student constraint, soft teacher constraint, and soft management constraints. The slot with more constraint violations must be taken over with the candidate having fewer constraint violations. The slot with a less objective function value must be replaced with the candidate carrying a large objective function value. If there are more hard constraint violations, then those must be removed by calculating the objective function value. For example, room-clash hard constraint must be selected first to resolve against consecutive-class soft constraint, or a slot having five conflicts must be resolved first instead of a slot with one conflict.

Table 2. List of hyper-heuristics

Hyper-heuristics	Priority order
Types of conflicts (Color-degree)	1. Hard 2. Soft student 3. Soft teacher 4. Soft management
Number of conflicts (Largest-degree, Next-fit-decreasing)	1. Three or more conflicts 2. Two-conflicts 3. One-conflict
Size of fitness (Largest-fit-decreasing)	1.More than 80% violations 2. 80% or less violations 3. 50% or less violations

4. Proposed system

The proposed algorithm carries out a memetic approach to find an optimal solution by using numerous AI techniques such as genetic algorithm, fuzzy logic, hyper-heuristics, tabu search, and hill-climbing, as shown in Fig 2. It implements a genetic algorithm with the initial repairing method and performs fuzzy-based adaptive genetic operators with local searches. Then the tabu search improves the solution fitness by following the hill-climbing threshold to avoid local optima. The algorithm also tries to maximize various objective functions to produce manageable solutions with different perspectives.

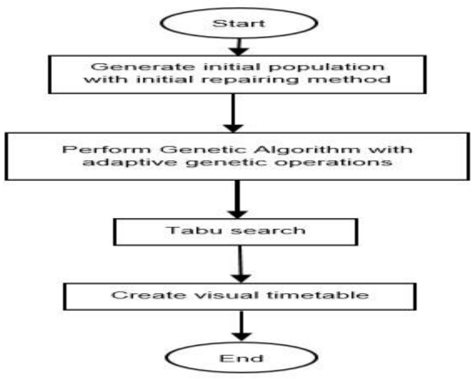
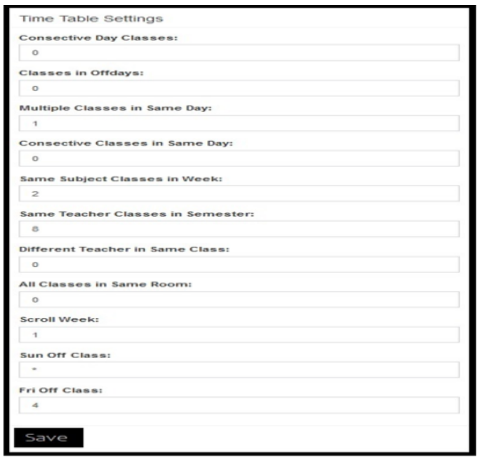

Start
Generate initial population with initial repairing method
Perform Genetic Algorithm with adaptive genetic operations
Tabu search
Create visual timetable
End

Figure 2. The proposed algorithm flowchart

4.1 Default algorithm parameters and interface settings

The proposed algorithm will be constructed initially with the number of hard and soft constraints, as indicated in Table 1. It is also set up with default parameters to generate a customized timetable, as given in Fig 3. By default, all the constraints are active, but it can be turned off to build a customized timetable for an institute. For example, if there is no issue with consecutive day allocation of various timeslots for a subject, the consecutive-day parameter can be set to 1. It creates an array of days to configure the off-days before creating a timetable. If the whole day is off-day, then the default value will be * symbol. Comma-separated slot numbers are constructed for random off-slots.


Time Table Settings

Consective Day Classes: 0

Classes in Offdays: 0

Multiple Classes in Same Day: 1

Consective Classes in Same Day: 0

Same Subject Classes in Week: 2

Same Teacher Classes in Semester: 8

Different Teacher in Same Class: 0

All Classes in Same Room: 0

Scroll Week: 1

Sun Off Class: *

Fri Off Class: 4

Figure 3. Default parameters in the proposed algorithm

4.2 Fitness functions

The fitness function is one of the important parameters to decide about the validity of resources allocated to various subject classes. It calculates overall solution fitness by Equation 5 [33], a weighted sum of all constraint violations used in multiple objective functions in Equation 1-4. The proposed algorithm tries to minimize the following objectives.

Objective 1: Avoid all the hard constraints to execute a timetable.

Objective 2: Seek to optimize all the soft constraints from a student perspective.

Objective 3: Try to optimize all the soft constraints from a teacher's perspective.

Objective 4: Seek to optimize all the soft constraints from a management perspective.

$$f_h(x) = \sum_{t=1}^{\infty} \left(\frac{1}{(1+\sum_{c=1}^{\infty} wC_h)}\right) \quad (1)$$

$$f_s(x) = \sum_{t=1}^{\infty} \left(\frac{1}{(1+\sum_{c=1}^{\infty} wC_{ss})}\right) \quad (2)$$

$$f_t(x) = \sum_{t=1}^{\infty} \left(\frac{1}{(1+\sum_{c=1}^{\infty} wC_{st})}\right) \quad (3)$$

$$f_m(x) = \sum_{t=1}^{\infty} \left(\frac{1}{(1+\sum_{c=1}^{\infty} wC_{sm})}\right) \quad (4)$$

$$f(x) = \sum_{t=1}^{\infty} \left(\frac{1}{(1+\sum_{c=1}^{\infty}(wC_h+wC_{ss}+wC_{st}+wC_{sm}))}\right) \quad (5)$$

The proposed algorithm calculates multiple objective functions in finding the fitness of a chromosome. An objective function is used to define the fitness value of a particular objective. It is a weighted sum of constraint violations for given timeslots in the same chromosome, whereas w is the constraint weight, the actual number of hard constraint violations ch, the soft student constraints css, the soft teacher constraint violations cst, and the soft management constraints csm. If there is no constraint violation, then it comes back one. In case of constraint violations of it, it returns less than one fitness value. A lower value means there are several constraint violations, whereas a higher fitness value will show a fewer number of constraint violations. Execution of timetable is not desirable in case of smaller fitness value. The overall objective function finds the overall fitness of a solution without the influence of any individual objective. If there is no hard or soft constraint violation, the objective function will return one that indicates an optimal solution for a problem.

4.3. Initial population with initial repairing method

First, the algorithm generates the initial population with the initial repairing method by allocating different resources to a fixed number of timeslots in a week to execute various classes. While creating an initial population, it randomly allocates resources to random timeslots by avoiding invalid resources or timeslots as in [15]. Initial repairing of the population ensures the allocation of exactly two unique timeslots with the same teacher for each subject class. In short, it ensures the allocation of the required quantity of valid resources to valid timeslots in each instance of the population. The selection of the initial population is the most critical part of this algorithm, as getting an optimal solution largely depends upon the original fitness of a population. If the initial population already has fewer constraint violations, then it will take less time to identify an optimal solution. Without this, it takes a longer time to find the optimal solution or may skip the best optimal solution within the defined number of iterations. For this purpose, the algorithm applies the initial repairing method to the selected population to avoid some constraints by evaluating the objective function value of the newly generated offspring. So it rejects any invalid allocation at an early stage, helps to generate the finest offspring, and improves the GA convergence time. Without this, it will be slow in convergence for the unfitted initial population. For example, every subject must have two distinct classes in a week, and those classes must be assigned to the same teacher. Timeslot number 39 must not be allocated to any class because of the Jumma Prayer.

4.4. Genetic algorithm

The genetic algorithm, which is the core part of the proposed algorithm, selects parent chromosomes with the adaptive selection process from the initial population, as shown in Fig 4. Adaptive process is a machine learning approach that automatically selects the most efficient genetic operation by evaluating the given heuristics. After the selection process, it applies the crossover operator on the selected chromosomes by applying a local search technique with the hill-climbing algorithm. If new chromosomes have a better fitness value than of its parents, then the mutation process is performed by using the hill-climbing algorithm on it. The new chromosome is compared with those of the previous population and replaced with the least fitted chromosome if it has more fitness value. Then genetic algorithm iterates for a certain hill-climbing threshold or terminates if convergence is achieved.

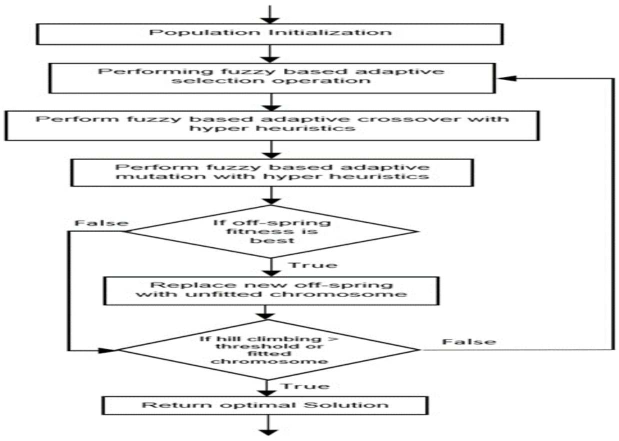

Population Initialization
Performing fuzzy based adaptive selection operation
Perform fuzzy based adaptive crossover with hyper heuristics
Perform fuzzy based adaptive mutation with hyper heuristics
False
If off-spring fitness is best
True
Replace new off-spring with unfitted chromosome
If hill climbing > threshold or fitted chromosome
False
True
Return optimal Solution

Figure 4. Genetic algorithm flowchart

4.4.1 Fuzzy-based adaptive genetic operations

The fuzzy logic The fuzzy logic rule base as implemented in [27], is used to implement adaptive genetic operations in the genetic algorithm. If the newly generated chromosome has more conflicts or its size is greater than a certain threshold, then the mutation probability will be greater. The one-point crossover will be preferable for chromosomes having a low fitness value, whereas the two-point crossover is preferable for chromosomes with a larger fitness value.

4.4.2 Selection process

The selection process is a way to get chromosomes with high fitness value from a population by its objective function value. If it has some constraint violations, then selected chromosomes can generate the best offspring. The adaptive selection strategy is applied to select fitted chromosomes with different selection strategies by using the fuzzy rule base, as defined in Table 3.

Table 3. Fuzzy rule base for the selection process
Time slots Number of conflicts Selection method

Time slots	Number of conflicts	Selection method
>= 50	>= 24 x subjects x 5% of the population	Tournament
>= 50	< 24 x subjects x 5% of the population	Wheel
< 50	>= 24 x subjects x 5% of the population	Rank
< 50	< 24 x subjects x 5% of the population	Best

Tournament selection, one of the efficient methods in finding the best parents, is used to select parent chromosomes by selecting a set of random chromosomes from a population. After that, selected chromosomes are entered in n size tournament where chromosomes with the highest fitness value are selected as parents. The fuzzy logic rule base determines the tournament-size n, as defined in table 4. It repeats the above process three times to get parents with the highest fitness value.

Table 4. Fuzzy rule base for tournament size

Population size	Fitness level	Tournament size
<=20	<= 50%	4
<= 50	<= 50%	10
> 50	<= 50%	16
<=20	> 50%	6
<= 50	> 50%	16
> 50	> 50%	24

The best selection is a simple technique to find parent chromosomes. All the chromosomes in a population are sorted by their fitness value to pick the best chromosomes. It influences most fitted chromosomes and may ignore hidden fitness in some chromosomes.

Roulette wheel selection is a technique with equal opportunity to take parent chromosomes from a population. Chromosomes are mapped to their fitness percentage against the whole population and represented as space in the roulette wheel. If it has more space on the wheel, it will have more chance of selection. For this purpose, a random number is generated and incremented to a certain value that is used to get selected space on the roulette wheel to decide about parents.

Rank selection calculates the rank of each chromosome by calculating its fitness value in a specific range. Rank is selected randomly, and the most fitted chromosome is chosen from it. It gives an equal chance of selection for every chromosome and maintains diversity in the next generation, but its performance is slow for the same fitness value in most cases.

After the selection process, GA performs a crossover operation to create new offspring with having the best features of their parents.

4.4.3 Crossover

The crossover is just like a natural genetic operation to get inherited qualities of the parents in offspring by performing a cross between two or more parents. The adaptive crossover approach is used to select a crossover technique by using the fuzzy rule base, as shown in Table 5. In the proposed algorithm, any crossover method will exchange a gene as a class instead of bits in the gene [17]. In this way, it will create no new constraint violations in the new offspring. It applies the hill-climbing threshold to regenerate offspring with better features than their parents. It also generates a mirror offspring with the reverse process to avoid skipping the best offspring for mutation operation.

Table 5. Fuzzy rule base for crossover

Fitness level	Constraint violations	Crossover technique
>= 50	>= 5	One-point
>= 50	< 5	Two-point
< 50	>= 5	Uniform

In a one-point crossover, a crossover point n is generated randomly between 2 to subjects-1. Then it is used to generate a new off-spring by copying all subject classes from 1 to n from the first parent and n+1 to n from the second parent, as shown in Fig 5. One-point is normally good for both parents to have nearly equal fitness value.

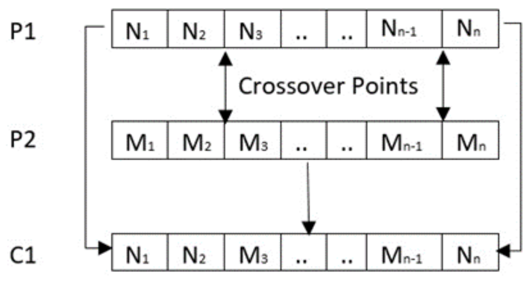


Figure 5. One-point crossover

In a two-point crossover, two random locations, n1 and n2, are created between 2 to subjects-1. Then these are used to create a new off-spring by copying classes from 1 to 1-n1 and from n2+1 to n from the first parent. Then copy classes from n1+1 to n2 from the second parent, as shown in Fig 6. It performs well if constraint violation is less in the parents.

The uniform crossover technique is a preferable choice to get balanced features in offspring. It interchanges a group of random genes between two parents. It will be a better technique when both parents have different fitness values, so one offspring must have the finest properties of both parents.

After the crossover process, GA performs mutation operation to create new best features in the new offspring for optimal results.

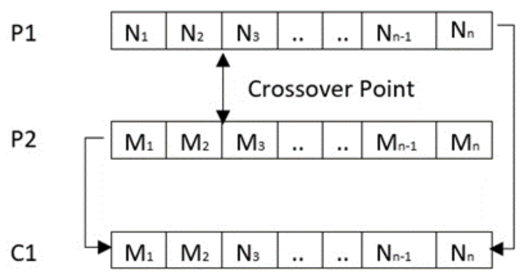


Figure 6. Two-point crossover

4.4.4 Mutation

The mutation is the most important genetic operator as it produces random changes in the new offspring and adjusts its fitness level as in the guided mutation as discussed in [27]. It applies the hill-climbing threshold to get offspring with more fitness than its parents. Two different mutation procedures are applied in the proposed algorithm, such as the mutation type and mutation process. The selection of these depends upon the fitness level of offspring. The mutation process will be conducted in a gene/bit of the new offspring. It depends upon the size of constraint violations, mutation rate, and the number of genes/bits as defined by hyper-heuristics, as shown in Table 2. The Adaptive mutation technique performs to adjust the mutation process in the new offspring by using the fuzzy rule base, as shown in Table 6.

The algorithm first selects the mutation type that may be resource/bit or class/gene based on the previous fitness of the gene. The resource/bit mutation is performed to change the single resource, whereas the class/gene mutation is performed to change all the resources of that class. It adjusts the mutation probability between 5% to 20% for the resource/bit mutation and 2% - 10% for the class/gene mutation [34]. The critical conflict with the class/gene mutation is a teacher-subject clash that must be avoided before performing this mutation. Only classes having the same teacher are swapped to avoid the teacher-subject clash. The resource/bit mutation is preferable with a chromosome having a few constraint violations and can be used without violating any other constraint.

Table 6. Fuzzy rule base for mutation
Fitness level Constraint Violations Used slots

Fitness level	Constraint violations	Used slots	Mutation type
>= 50%	>= 5	>=80 %	Class swap
>= 50%	< 5	>=80 %	Resource swap
< 50%	>= 5	>=80 %	Resource swap
< 50%	< 5	>=80%	Resource swap
>= 50%	>= 5	<80%	Class flip
>= 50%	< 5	<80%	Resource flip
< 50%	**>= 5**	**<80%**	**Resource flip**
< 50%	**< 5**	**<80%**	**Resource flip**

The mutation process, as described in [4] with multiple heuristics, is selected to improve the fitness of the new offspring. If the fitness of the new offspring is less than that of its parents, thus it will reject new changes. In that case, the mutation process repeats to generate offspring by using the hill-climbing threshold. Flip mutation performs by flipping the random gene/bit in the new offspring, as shown in Fig 7. If several free timeslots or resources are available, then it will be a preferred technique, as claimed by [35]. It quickly improves the fitness level of a chromosome. With a fully congested chromosome, it will perform minor or no improvement. Swap mutation performs by interchanging two random genes/bits with each other, as shown in Fig 8 if the new fitness value of the gene/bit is more than its previous value. It is slow in convergence, but it will be the preferred technique if some free resources are available. The only related resource may swap with another resource in the case of the resource/bit swapping.

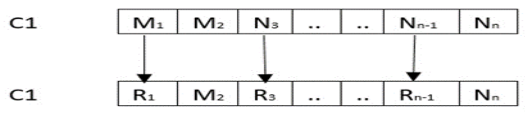


Figure 7. Class flip mutation

After performing crossover and mutation operations, it will replace new offspring with the least fitted chromosome in the population by elitist selection if it has more fitness value than its parents. Elitist selection is a way to retain original parents in the later generation to avoid information loss while performing genetic operations. In this way, the genetic algorithm will never produce the population with the unfitted offspring in the later generation and will generate an optimal solution in less time. The hyper-heuristics, as defined in Table 2, are used to select the unfitted chromosome to take over. All the above genetic operations generate offspring with higher fitness by using the hill-climbing threshold until it achieves convergence. If the genetic algorithm cannot bring about the best global optimal solution, then it will be further improved by tabu search [36].

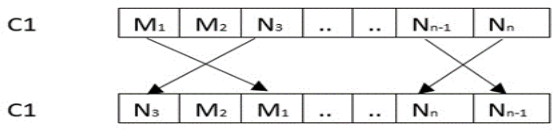


Figure 8. Class swap mutation

The proposed algorithm applies the tabu search to improve the quality of the solution generated by GA, as shown in Fig. 9. It iterates using the tabu_number to apply to a candidate with a higher objective function value or terminates if there is no fitted candidate.

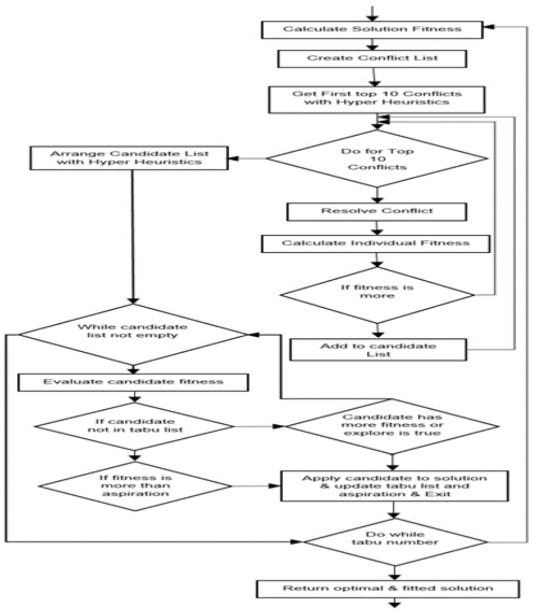

Calculate Solution Fitness
Create Conflict List
Get First top 10 Conflicts with Hyper Heuristics
Do for Top 10 Conflicts
Arrange Candidate List with Hyper Heuristics
Resolve Conflict
Calculate Individual Fitness
If fitness is more
Add to candidate List
While candidate list not empty
Evaluate candidate fitness
If candidate not in tabu list
Candidate has more fitness or explore is true
If fitness is more than aspiration
Apply candidate to solution & update tabu list and aspiration & Exit
Do while tabu number
Return optimal & fitted solution

Figure 9. Tabu search flowchart

4.5 Candidate List

The candidate list contains many solutions with more fitness value than its parent chromosome and is used to replace less fitted genes in it. Constraint violation of each gene is calculated by using multiple fitness functions, as shown in Equation 1-4. The hyper-heuristics with fuzzy logic are applied to perform changes in resources allocated to the class. If the new fitness value of the candidate is more than its previous fitness, then it will be added to the candidate list. It performs the process for some random genes instead of traversing all of them. It is also organized in the sequence of hyper-heuristics mentioned in table 2 [37].

4.5.1 Aspiration condition

The aspiration condition is the height fitness level that has been achieved so far in all the previous iterations of the tabu search. The aspiration condition will compare with the new candidate's fitness. If the candidate fitness value is more than the aspiration condition, then it may apply to the current solution and the tabu list. If it has more candidate solutions, then the same process will apply to the next candidate.

4.5.2 Tabu list

The tabu search traverses the candidate list to select a candidate with the highest fitness value. The move or swap operation performs to replace the selected candidate in a solution. The move operation shifts a timeslot having constraints with the chosen candidate slot, whereas the swap operation interchanges a timeslot with the selected candidate slot. If the candidate solution is not in the tabu list already, then it will be added to the tabu list. Without this, the candidate solution is rejected, and it selects the next candidate to repeat the same procedure. No timeslot in the tabu list may overwrite unless it is shifted to another timeslot already. If the candidate has more fitness value than candidates in tabu_list, thus it will be offered to be overwritten but must wait for tabu_unfit_number turns unless the candidate fitness is more than the aspiration condition. After each tabu iteration, it applies the selected candidate to the current solution and generates a new candidate list, if it has no better candidate. The process terminates on convergence, or at the end of tabu number iterations. The tabu_unfit_number explores the hidden fitness of a solution by applying a less fitted candidate to a solution. The adaptive tabu number, which is implemented by the fuzzy rule base, maintains a balance in conflict resolutions and computation time in the proposed algorithm.

4.6 Visualize and review the timetable

The web-based dynamic timetable manager visualizes the solution returned by the proposed algorithm. It displays all the allocated resources for all subject classes to the fixed timeslots in a week. It also represents various hard and soft constraints with different color tags for each timeslot block in case of any conflict. So it becomes easy to identify the constraint violations. It also provides a mechanism to adjust the timeslots to remove clashes or to improve resource utilization. The move-slot event transfers allocated subject class resources to another empty slot. The swap-slot event exchanges associated resources between two allocated timeslots. The delete-slot event removes all the resources from the timeslot. With missing resources, the add-resource event allocates all the required resources to the timeslot.

4.7 Print or generating CSV file

It provides an option to download the CSV format timetable for further processing in Microsoft Excel. It is also possible to display the customized timetable in HTML format with the web browser. The print-option is too available to print the hard copy of the weekly university timetable.

4.8 Complexity of the proposed algorithm

The proposed algorithm proposed algorithm is a combination of various optimization techniques with genetic algorithm and tabu search. Therefore, its complexity largely depends upon the size of generations, population size, number of conflicts, threshold size, and optimization techniques used [31].

$$O(\text{MO-FAMA}) = O(\text{GA}) + O(\text{tabu}) \qquad (6)$$

$$O(\text{GA})=\text{threshold}*\text{generations}*\text{population size}*(O(\text{selection}) + \text{crossover probability}*O(\text{crossover}) + \text{mutation probability}* O(\text{mutation})) \qquad (7)$$

O(Selection) = population size*conflicts*O(selection technique) (8)
O(Crossover) = chromosomes*genes*conflicts*O(fitness) (9)
O(Mutation) = genes*bits*conflicts*O(fitness) (10)
O(tabu)=threshold*population_size*conflicts*(O(conflicts* chromosomes*candidate_list*tabu_list*O(fitness))) (11)

If all variables used in the above equations are assumed as constant values, as executed in a fixed time, then we can obtain O(GA) = O(fitness) and O(tabu) = O(fitness), so the complexity of the proposed algorithm is equal to O(fitness). So fitness that is used to check the fitness of the population, will decide about the overall complexity of the proposed algorithm lies between O(n) to O(n2) depending upon the implementation of fitness function. The complexity of the fitness function will normally depend upon the actual application of the proposed algorithm in solving the university course timetabling problem.

5. System testing

The proposed algorithm is a web-based application configured on the Linux Shared Server with a graphical user interface. A test is conducted, with the default parameters given in Fig 3, to confirm its accuracy with each dataset individually. Two sets of tests for each sample dataset, with variable population size, are performed with 25 iterations each. A set of 50, 100, 150, 200, and 250 population sizes, is applied to analyze the proposed algorithm. The test performs a comparison of the proposed algorithm for its performance and efficiency with the other three algorithms, simple GA, IGA [17], and simple HGA [22]. All the objective functions are calculated to get the fitness of the overall solution from multiple perspectives. All the genetic operations were also tested independently with the simple GA to explore their impact on overall performance. In the same way, all the individual techniques implemented in the proposed algorithm are also tested, individually. The average result of each dataset identifies its impact at the convergence time and overall fitness of the solution. All the results are further summarized to get the overall accuracy of the proposed algorithm.

6. System Results

The test datasets are collected from four different schools of UMT Lahore for session 2021. These are in Microsoft Excel format and must be converted into the CSV format before uploading it to the system. Each dataset contains information about subjects, teachers, rooms, and classes to be executed, as shown in Table 7. These resources are allocated to create an executable and optimal timetable. Classes are divided into a number of sections to accommodate class strength and also categorized into various programs in that school. There is no issue in executing classes of different programs and sections at the same time. Some students from different programs can share the same class, which is an exceptional case, so that will be excluded while executing the proposed algorithm.

Table 7. Sample test datasets

Test Datasets	***Subjects***	***Teachers***	***Rooms***	***Classes***
School of Business & Economics (SBE)	123	79	35	242
School of Systems & Technology (SST)	119	92	42	232
School of Professional Advancement (SPA)	62	31	18	106
School of Textile Design (STD)	115	83	42	224

5.1 Results and discussion

Table 8. Test results of the performance of the multi-objective fuzzy-based adaptive memetic algorithm

Algorithm	**Population Size**	**Conflicts**	**Time (ms)**	**Accuracy %**	**Hard fitness**	**Soft student fitness**	**Soft teacher fitness**	**Soft management fitness**	**Overall fitness**
GA	50	19235	9012	45.18	0.000106	0.002169	0.015152	0.001695	0.00010
GA	100	19853	13217	45.88	0.000102	0.002294	0.022727	0.001992	0.00010

GA	150	20047	14124	47.72	0.000105	0.002488	0.016129	0.002049	0.00010
GA	200	20185	16721	45.98	0.000101	0.002151	0.021739	0.002016	0.00010
GA	250	19959	33705	48.79	0.000109	0.002103	0.027027	0.002000	0.00010
IGA	50	19235	14489	62.40	0.000155	0.002307	1.00000	0.002933	0.00010
IGA	100	19853	18078	51.71	0.000116	0.002079	1.00000	0.001919	0.00010
IGA	150	20047	17528	52.74	0.000116	0.002208	1.00000	0.002488	0.00010
IGA	200	20185	18111	54.28	0.000120	0.001990	1.00000	0.002110	0.00011
IGA	250	19959	31022	54.62	0.000120	0.002310	1.00000	0.002400	0.00011
HGA	50	19235	12712	58.39	0.000140	0.002180	1.00000	0.003690	0.00012
HGA	100	19853	18334	57.72	0.000130	0.002080	1.00000	0.002530	0.00012
HGA	150	20047	19201	55.33	0.000120	0.002120	1.00000	0.002210	0.00011
HGA	200	20185	22297	60.32	0.000140	0.002120	1.00000	0.002160	0.00012
HGA	250	19959	37082	59.16	0.000140	0.002120	1.00000	0.001990	0.00012
MO-FAMA	50	19235	10619	95.64	0.003021	0.003120	1.00000	0.005236	0.00120
MO-FAMA	100	19853	15790	96.21	0.003436	0.003053	1.00000	0.007353	0.00130
MO-FAMA	150	20047	20610	96.16	0.004405	0.002890	1.00000	0.005025	0.00130
MO-FAMA	200	20185	16281	96.18	0.004255	0.003021	1.00000	0.004808	0.00130
MO-FAMA	250	19959	23190	96.29	0.004329	0.003140	1.00000	0.005181	0.00140

The comprehensive results of allocating classes of all of the schools are given in Table 8 which shows the objective function values are greater in the proposed algorithm compared with those of other algorithms. It shows a higher convergence time in a large size population than in a small size population, but there is no significant progress in solution fitness. Due to some unavoidable soft constraints, any optimization algorithm can never achieve 100% fitness in most cases. The proposed algorithm returns with satisfactory results overall allocation in schools as well as individual allocation of classes in schools, as shown in Fig 10. It shows 96.29% accuracy with 250 size population in resolving overall constraints compare with 48.79% accuracy in resolving conflicts with the simple GA, 54.62% accuracy with the IGA in (Boonyopakorn and Meesad, A Hybrid Immune Genetic Algorithm to Solve University Time Table Problems 2017), and 59.16% with the simple HGA [38]in with the same number of iterations.

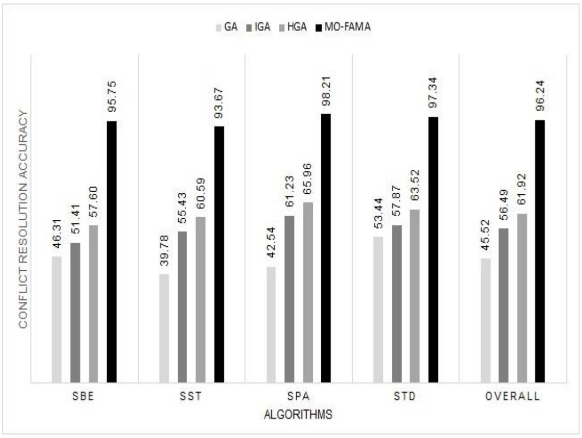


Figure 10. The proposed algorithm performance

The fitness value of new chromosomes improves quickly at the initial stage of the algorithm, but there is no substantial improvement in new chromosomes in the later stage, as shown in Fig 11.

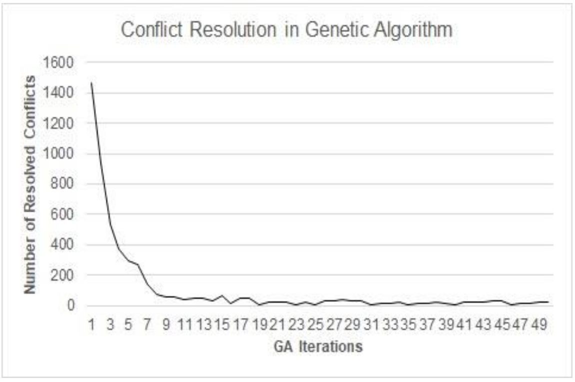


Figure 11. Conflict resolution in the genetic algorithm

The drastic shift in the early stage is because of several conflicts in a population and to replace the low-fitted chromosomes with high fitness value chromosomes by using hyper-heuristics. The optimal solution will be a point where the rate of change of improvement zeros. A variation of mutation probability impacts convergence time, as shown in Fig 12 it takes more time to converge with a high rate of mutation probability instead of resolving more conflicts.

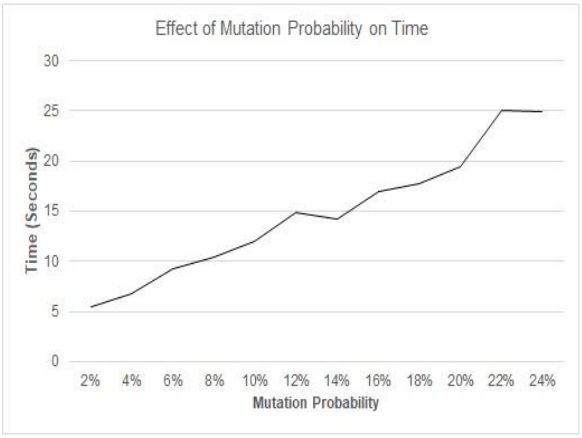


Figure 12. Effect of mutation probability on the time

The performance of the genetic algorithm in resolving with hard and soft constraints highly depends upon the techniques used in it.[39] The efficiency of these ultimately improves or degrades overall algorithm performance [40].The adaptive approach, to perform various genetic operations in the genetic algorithm, is also tested to observe the impact of these on the solution fitness and convergence time. These greatly affect the performance of the proposed algorithm, as shown in Table 9. The adaptive crossover and mutation techniques in the genetic algorithm work well compared with those of others. The adaptive selection strategy is just behind the tournament selection, which is the best selection strategy in the GA, but it takes more time than the adaptive selection technique.

Table 9. Results of genetic techniques in GA

Genetic technique	Initial conflicts	Time (ms)	Conflicts	Overall fitness
Selection techniques				
Adaptive	19254	5899	10675	0.000094
Best	19254	5724	10890	0.000092
Rank	19254	5967	10741	0.000093
Tournament	19254	6728	10587	0.000094
Wheel	19254	5880	10982	0.000091
Crossover techniques				
Adaptive	19520	4731	10172	0.000098
One-Point	19520	5309	10388	0.000096
Two-Point	19520	4681	10829	0.000092
Uniform	19520	4922	10823	0.000092
Mutation types				
Class/Gene	18912	5402	10552	0.000095
Resource/ Bit	18912	4711	11512	0.000087
Adaptive	18912	4692	10344	0.000097
Mutation process				
Adaptive	19194	4901	10279	0.000097
Flip	19194	5681	10619	0.000094
Swap	19194	5499	10550	0.000095

Using a genetic algorithm alone is not well for solving the timetabling problem, so implementing a memetic algorithm with local search improves the solution fitness, as indicated in Fig 10. However, its excessive use negatively affects computation time [41].It increases computation time from 7.6 seconds to 8.3 seconds as the tabu_number change from 50 to 250, as illustrated in Fig 13. A deep search takes more processing time with an optimal solution as compared to a narrow search takes less time without a workable solution[42].

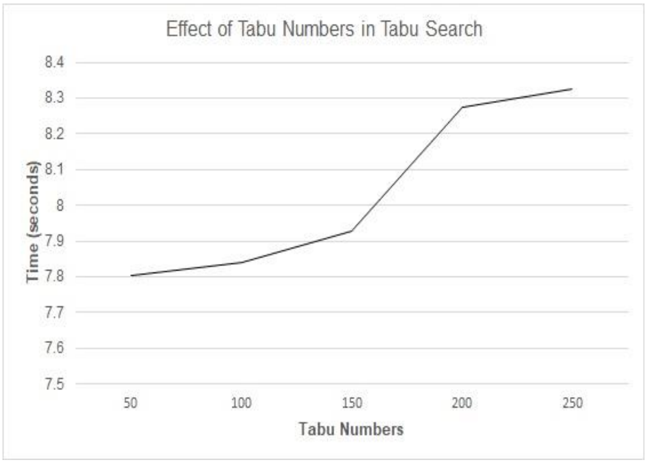


Figure 14. Effect of tabu numbers on the tabu search

Using hyper-heuristics also corrects the performance of the genetic algorithm and tabu search. A significant improvement in the convergence time of these algorithms is because of hyper-heuristics, as shown in Fig 14.

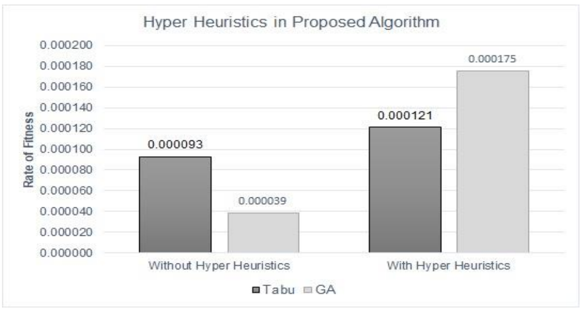


Figure 13. Effect of tabu numbers on the tabu search

A large size population returns a low rate of convergence in the proposed algorithm. It takes 23 seconds in the 250-size population compared with 10 seconds in the 50-size population, as shown in Table 8.[41] The accuracy of the proposed algorithm is 95.91% in the 50-size population as compared with 96.42% in the 250-size population. It also has more fitness for a large size population than for a small size population, but it increases processing time.

7. Conclusion

The results obtained through the proposed memetic algorithm are satisfactory and adequate in optimizing the university course timetabling problem. The proposed algorithm minimizes several conflicts and optimizes a manageable timetable from the student, teacher, and management perspectives. It returns an efficient and workable timetabling solution by implementing a genetic algorithm with adaptive genetic operations with hyper-heuristics and tabu search. The use of initial repairing, local search, and hill-climbing algorithms affect the solution convergence time. A simple genetic algorithm is an effective optimization technique, but it seldom generates an optimal and workable solution, depending upon the input data. Therefore, it must be combined with local search optimization techniques to form a population-based hybrid algorithm to generate a more accurate solution as the proposed algorithm in this paper. It is also impossible to find an optimal solution by satisfying all the hard and soft constraints due to the physical limits of resources. So achieving a single objective is not a desirable approach and multiple objective functions implemented in the proposed algorithm effectively optimize the solution from different perspectives. The proposed multi-objective fuzzy-based adaptive memetic algorithm is a well-organized and structured solution to generate an optimal and applicable solution for the university course timetabling problem. Future work includes fuzzy-based controls to select a suitable optimization algorithm from a set of global and local search algorithms.

Acknowledgements.
The authors thank reviewers for their valuable feedback and anonymous comments.

References

[1] H. Babaei, J. Karimpour, and A. Hadidi, “A survey of approaches for university course timetabling problem,” *Comput. Ind. Eng.*, vol. 86, pp. 43–59, 2015, doi: 10.1016/j.cie.2014.11.010.

[2] M. C. Chen, S. N. Sze, S. L. Goh, N. R. Sabar, and G. Kendall, “A Survey of University Course Timetabling Problem: Perspectives, Trends and Opportunities,” *IEEE Access*, vol. 9, pp. 106515–106529, 2021, doi: 10.1109/ACCESS.2021.3100613.

[3] P. Boonyopakorn and P. Meesad, “A hybrid immune genetic algorithm to solve university time table problems,” *Walailak J. Sci. Technol.*, vol. 14, no. 10Special Issue, pp. 825–835, 2017, doi: 10.14456/vol14iss9pp%p.

[4] A. A. Gozali and S. Fujimura, “Solving University Course Timetabling Problem Using Multi-Depth Genetic Algorithm,” *SHS Web Conf.*, vol. 77, no. October 2019, p. 01001, 2020, doi: 10.1051/shsconf/20207701001.

[5] “Using Reinforcement Learning in Solving Exam Timetabling Problems Queen ’ s University Belfast October 2018 COPYRIGHT © 2018 BY KEHAN HAN,” 2018.

[6] G. A. Neufeld and J. Tartar, “Graph Coloring Conditions for the Existence of Solutions to the Timetable Problem,” *Commun. ACM*, vol. 17, no. 8, pp. 450–453, 1974, doi: 10.1145/361082.361092.

[7] G. H. G. Fonseca, H. G. Santos, E. G. Carrano, and T. J. R. Stidsen, “Integer programming techniques for educational timetabling,” *Eur. J. Oper. Res.*, vol. 262, no. 1, pp. 28–39, 2017, doi: 10.1016/j.ejor.2017.03.020.

[8] Z. Lixi and L. SimKim, “Constructing university timetable using constraint satisfaction programming approach,” *Proc. - Int. Conf. Comput. Intell. Model. Control Autom. CIMCA 2005 Int. Conf. Intell. Agents, Web Technol. Internet*, vol. 2, no. November, pp. 55–60, 2005, doi: 10.1109/cimca.2005.1631445.

[9] W. Alomoush and W. Banzhaf, “A Comprehensive Review of Uncapacitated University.”

[10] F. Glover, “Artificial intelligence, heuristic frameworks

and tabu search," *Manag. Decis. Econ.*, vol. 11, no. 5, pp. 365–375, 1990, doi: 10.1002/mde.4090110512.

[11] P. Bangert, "Optimization: Simulated Annealing," *Optim. Ind. Probl.*, vol. 220, no. 4598, pp. 165–200, 2012, doi: 10.1007/978-3-642-24974-7_7.

[12] A. Colorni, M. Dorigo, and V. Maniezzo, "A genetic algorithm to solve the timetable problem," *Politec. di Milano, Milan, …*, pp. 1–24, 1992, [Online]. Available: http://www.researchgate.net/publication/2253354_A_Genetic_Algorithm_To_Solve_The_Timetable_Problem/file/9fcfd50ff95b8c862d.pdf.

[13] M. Joudaki, M. Imani, and N. Mazhari, "Using improved memetic algorithm and local search to solve university Course Timetabling problem (UCTP)," *Proc. 2011 Int. Conf. Artif. Intell. ICAI 2011*, vol. 2, pp. 501–506, 2011.

[14] A. L. A. Bolaji, A. T. Khader, M. A. Al-Betar, and M. A. Awadallah, "A hybrid nature-inspired artificial bee colony algorithm for uncapacitated examination timetabling problems," *J. Intell. Syst.*, vol. 24, no. 1, pp. 37–54, 2015, doi: 10.1515/jisys-2014-0002.

[15] A. Muklason, R. G. Irianti, and A. Marom, "Automated course timetabling optimization using tabu-variable neighborhood search based hyper-heuristic algorithm," *Procedia Comput. Sci.*, vol. 161, pp. 656–664, 2019, doi: 10.1016/j.procs.2019.11.169.

[16] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, "A Classification of Hyper-Heuristic Approaches : Revisited."

[17] A. Rezaeipanah, S. S. Matoori, and G. Ahmadi, "A hybrid algorithm for the university course timetabling problem using the improved parallel genetic algorithm and local search," *Appl. Intell.*, vol. 51, no. 1, pp. 467–492, 2021, doi: 10.1007/s10489-020-01833-x.

[18] C. Ding, L. Chen, and B. Zhong, "Exploration of intelligent computing based on improved hybrid genetic algorithm," *Cluster Comput.*, vol. 22, no. s4, pp. 9037–9045, 2019, doi: 10.1007/s10586-018-2049-7.

[19] A. Eludire and C. Akanbi, "A Conceptual Approach to Resources Allocation Scheduling," *J. Adv. Math. Comput. Sci.*, vol. 25, no. 6, pp. 1–12, 2017, doi: 10.9734/jamcs/2017/32569.

[20] Z. Li and Y. Li, "Application of multi-objective optimization problem based on genetic algorithm," *J. Phys. Conf. Ser.*, vol. 2037, no. 1, 2021, doi: 10.1088/1742-6596/2037/1/012021.

[21] C. Akkan and A. Gülcü, "A bi-criteria hybrid genetic algorithm with robustness objective for the course timetabling problem," *PATAT 2016 - Proc. 11th Int. Conf. Pract. Theory Autom. Timetabling*, pp. 451–456, 2016.

[22] N. G. A. H. Saptarini, P. I. Ciptayani, and I. B. I. Purnama, "A custom-based crossover technique in genetic algorithm for course scheduling problem," *TEM J.*, vol. 9, no. 1, pp. 386–392, 2020, doi: 10.18421/TEM91-53.

[23] I. BALAN, "A New Genetic Approach for Course Timetabling Problem," *J. Appl. Comput. Sci. Math.*, vol. 15, no. 1, pp. 9–14, 2021, doi: 10.4316/jacsm.202101001.

[24] M. M. Tavakoli, H. Shirouyehzad, F. H. Lotfi, and S. E. Najafi, "Proposing a novel heuristic algorithm for university course timetabling problem with the quality of courses rendered approach; a case study," *Alexandria Eng. J.*, vol. 59, no. 5, pp. 3355–3367, 2020, doi: 10.1016/j.aej.2020.05.004.

[25] K. Wang, W. Shang, M. Liu, W. Lin, and H. Fu, "A Greedy and Genetic Fusion Algorithm for Solving Course Timetabling Problem," *Proc. - 17th IEEE/ACIS Int. Conf. Comput. Inf. Sci. ICIS 2018*, pp. 344–349, 2018, doi: 10.1109/ICIS.2018.8466405.

[26] A. Rjoub, "Courses timetabling based on hill climbing algorithm," *Int. J. Electr. Comput. Eng.*, vol. 10, no. 6, pp. 6558–6573, 2020, doi: 10.11591/IJECE.V10I6.PP6558-6573.

[27] Y. Sun, X. Luo, and X. Liu, "Optimization of a university timetable considering building energy efficiency: An approach based on the building controls virtual test bed platform using a genetic algorithm," *J. Build. Eng.*, vol. 35, p. 102095, 2021, doi: 10.1016/j.jobe.2020.102095.

[28] H. Zhang, B. Xiao, J. Li, and M. Hou, "An improved genetic algorithm and neural network-based evaluation model of classroom teaching quality in colleges and universities," *Wirel. Commun. Mob. Comput.*, vol. 2021, 2021, doi: 10.1155/2021/2602385.

[29] S. Susan and A. Bhutani, "A novel memetic algorithm incorporating greedy stochastic local search mutation for course scheduling," *Proc. - 22nd IEEE Int. Conf. Comput. Sci. Eng. 17th IEEE Int. Conf. Embed. Ubiquitous Comput. CSE/EUC 2019*, no. August 2019, pp. 254–259, 2019, doi: 10.1109/CSE/EUC.2019.00056.

[30] O. Rossi-doria and B. Paechter, "An hyperheuristic

approach to course timetabling problem using an evolutionary algorithm," *Evol. Comput.*, no. January 2014, 2003, [Online]. Available: https://www.researchgate.net/publication/228724044.

[31] E. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg, "Hyper-Heuristics: An Emerging Direction in Modern Search Technology," *Handb. Metaheuristics*, pp. 457–474, 2006, doi: 10.1007/0-306-48056-5_16.

[32] T. L. June, J. H. Obit, Y. B. Leau, J. Bolongkikit, and R. Alfred, "Sequential Constructive Algorithm incorporate with Fuzzy Logic for Solving Real World Course Timetabling Problem," *Lect. Notes Electr. Eng.*, vol. 603, no. February, pp. 257–267, 2020, doi: 10.1007/978-981-15-0058-9_25.

[33] F. G. Lobo, D. E. Goldberg, and M. Pelikan, "Time complexity of genetic algorithms on exponentially scaled problems," *Proc. Genet. Evol. Comput. Conf.*, no. June, pp. 151-- 158, 2000.

[34] M. Imran, M. U. Sattar, H. Wazirkhan, and A. Ghaffar, "Selecting a Better Classifier Using Machine Learning for."

[35] A. Ghafar, N. Shah, and M. M. Iqbal, "Gender Recognition for Urdu language Speakers Using Composite and Multi-Layer Feature Approaches with Fuzzy Logic," *Tech. J.*, vol. 24, no. 02, pp. 61–75, 2019.

[36] B. Ahmad *et al.*, "Intelligent Digital Twin to make Robot Learn the Assembly process through Deep Learning," 2021.

[37] M. U. Sattar, S. Palaniappan, A. Lokman, N. Shah, Z. Riaz, and U. Khalid, "User experience design in virtual reality medical training application," *J. Pak. Med. Assoc.*, vol. 71, no. 7, pp. 1730–1735, 2021, doi: 10.5455/JPMA.22992.

[38] B. Wang, Y. Geng, and Z. Zhang, "Applying genetic algorithm to university classroom arrangement problem," *J. Phys. Conf. Ser.*, vol. 1325, no. 1, 2019, doi: 10.1088/1742-6596/1325/1/012157.

[39] Y. F. Ge, J. Cao, H. Wang, Y. Zhang, and Z. Chen, *Distributed Differential Evolution for Anonymity-Driven Vertical Fragmentation in Outsourced Data Storage*, vol. 12343 LNCS, no. October. Springer International Publishing, 2020.

[40] Y. F. Ge *et al.*, "Distributed Memetic Algorithm for Outsourced Database Fragmentation," *IEEE Trans. Cybern.*, vol. 51, no. 10, pp. 4808–4821, 2021, doi: 10.1109/TCYB.2020.3027962.

[41] Y. F. Ge, M. Orlowska, J. Cao, H. Wang, and Y. Zhang, "Knowledge transfer-based distributed differential evolution for dynamic database fragmentation," *Knowledge-Based Syst.*, vol. 229, p. 107325, 2021, doi: 10.1016/j.knosys.2021.107325.

[42] Y. F. Ge, J. Cao, H. Wang, Z. Chen, and Y. Zhang, "Set-Based Adaptive Distributed Differential Evolution for Anonymity-Driven Database Fragmentation," *Data Sci. Eng.*, vol. 6, no. 4, pp. 380–391, 2021, doi: 10.1007/s41019-021-00170-4.