International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 13 (2021) pp. 001-009

Received: 8 July, 2020; Accepted: 4 January, 2021; Published: 28 January, 2021

Incorporating Teacher's Preferences and Student Time Management in University Course Timetabling

Seba Susan[1], and Aparna Bhutani[2]

[1] Department of Information Technology, Delhi Technological University,
Shahbad Daulatpur, Bawana Road, Delhi- 110042
seba_406@yahoo.in

[2] Department of Information Technology, Delhi Technological University,
Shahbad Daulatpur, Bawana Road, Delhi- 110042
bhutaniaparna@gmail.com

***Abstract*: University course timetabling is a challenging task today due to the mushrooming of a variety of courses with flexible timings in different streams and disciplines in global universities. The challenge is to devise non-overlapping class schedules for students enrolled in different courses with the available and limited resources of teachers and classrooms. The task of automated timetable generation is a constrained optimization problem. In any optimization problem, the hard constraints are those that need to be mandatorily satisfied. The soft constraints are the penalties that are sought to be minimized in every iteration of the optimization algorithm. In this paper, we propose a novel set of soft constraints for university course timetabling that in addition to the conventional constraints, incorporates teacher's preferences and student time management as well. The latter constraint takes the form of minimizing student mobility between classrooms and restricting time gaps between classes to ensure that student's time on-campus is fully utilized. The evolutionary algorithms- Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Simulated Annealing (SA) are used for optimizing the course schedules. The timetables so generated, based on actual university data, are found to be more humanely optimized than the previous work of the authors due to the incorporation of human factor consideration both from the perspective of teachers and students.**

***Keywords*:** University Course Timetabling, Particle swarm optimization, Genetic Algorithm, Simulated Annealing, Teacher's preferences, Student time management

I. Introduction

University course timetabling is a challenging task today due to the mushrooming of a variety of courses with flexible timings in various streams and disciplines in global universities. The challenge is to devise non-overlapping class schedules for students enrolled in different courses with the available and limited resources of teachers and classrooms [1]. The task of automated timetable generation is a constrained optimization problem. The range of optimization algorithms vary from mathematical to evolutionary to heuristic [2]. Genetic Algorithm (GA) is a highly efficient evolutionary algorithm that follows the principle of the evolution of the fittest chromosome [23]. Its application to the timetable scheduling problem can be observed in [3,4]. Highly optimized schedules are generated by the heuristic Simulated Annealing (SA) algorithm, reportedly in minimal time, as per research in [5-8]. The different types of schedules generated by SA in [5-8] range from university course timetables, exam timetables, staff job schedules etc. SA and GA are the most investigated optimization algorithms for scheduling [16, 22] and combinations of GA and SA with local search have also been explored for timetabling problems [17, 21, 19]. Most of the algorithms differ in the initialization routines or variant of SA and GA used. Several variants of GA are found in literature [12]. Particle Swarm Optimization (PSO) which is another population-based evolutionary algorithm has been explored in [18, 20] for timetable scheduling. The penalty or cost function was, in fact, found lower for PSO as compared to GA in [20]. A timetable template satisfying the constraints is initialized in [9] before application of GA to optimize the timetable. In cases where students opt for elective courses, association rule mining is applied for mining student's preferences [10, 22]. In this paper, we extend our recent work on automatic timetable generation in [22] by formulating novel additional soft constraint functions that generate human-friendly optimized schedules both from the perspective of students and teachers. GA, SA and PSO optimization algorithms are investigated for the task. The organization of the paper is as follows: The different optimization algorithms used for our experiments are reviewed in Section II, the hard and soft constraints and the additional soft constraints proposed in this paper as well as the overall methodology are detailed in Section III, the results are discussed in Section IV and the overall conclusions are drawn in Section V.

II. Optimization algorithms

This section reviews three of the most popular optimization algorithms used for scheduling problems especially university

course timetabling, namely, Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Simulated Annealing (SA) algorithm. GA and PSO are evolutionary approaches that evolve globally optimal solutions over time based on the concept of the 'survival of the fittest'. SA is based on local search heuristics, and like GA and PSO, aims to find globally optimal solutions to the course timetabling problem.

A. *Optimization by SA*

The concept of Simulated Annealing (SA) [24] is borrowed from metallurgy. The metal prone to defects is heated to a high temperature and then cooled under controlled conditions. The resulting metal has lesser defects as compared to the initial condition and hence a lower energy. Optimization problems especially in scheduling are solved quite efficiently by SA in minimal execution time [5-8]. High temperature emulates Random Walk i.e. new possibilities are randomly explored at high temperatures akin to the mutation step in Genetic Algorithm. As the temperature cools with each iteration, the solution converges as the randomness factor reduces. The method is described in more detail below.

The initial temperature is set to a pre-defined high value. In every iteration, a random neighboring node is selected for each current node for possible replacement. The difference of energy levels is computed as

$$\Delta \boldsymbol{E} = \boldsymbol{E}(\boldsymbol{new}) - \boldsymbol{E}(\boldsymbol{current}) \quad (1)$$

where $\boldsymbol{E}(\boldsymbol{new})$ = Energy level for new node

$\boldsymbol{E}(\boldsymbol{current})$ = Energy level for current node

if $\Delta \boldsymbol{E} \leq 0$ then we make the move with a probability=1.

if $\Delta \boldsymbol{E} > 0$ then we still allow that move but with a lower probability given by

$$\boldsymbol{P} = \boldsymbol{e}^{-\frac{\Delta \boldsymbol{E}}{\boldsymbol{T}}} \quad (2)$$

Here, $\boldsymbol{P}$ represents the probability with which the new node is selected and $\boldsymbol{T}$ represents the temperature. The temperature is updated in every iteration as

$$\boldsymbol{T}_{i+1} = \boldsymbol{T}_i \times \beta \quad (3)$$

where, β is the cooling rate pre-defined in the algorithm.

B. *Optimization by GA*

Genetic Algorithm (GA) is a highly efficient evolutionary algorithm that follows the principle of the evolution of the fittest chromosome [23]. It is a popular optimization tool in pattern recognition experiments to optimize values of unknown parameters while learning [15]. It involves the steps of initialization of chromosomes, computing the fitness function, selection, crossover and mutation. These steps are described briefly below. The steps are iterated over all the generations, with the maximum number of generations being fixed at the beginning of the algorithm. The steps in a single generation of the Genetic Algorithm are:

i. *Initialization*:

The initial population of chromosomes are randomly initialized at the beginning of the algorithm. Chromosomes indicate single solutions to the problem and are represented as binary strings.

ii. *Computing the Fitness function*:

The fitness or the cost function is that which is optimized in every iteration. Higher the fitness value, the more fit is the chromosome (for a maximization problem).

iii. Selection

This stage emulates the principle of the survival of the fittest. The fittest chromosomes indicated by high values of fitness function are retained and replicated while the ones with lower values of the fitness function are eliminated.

iv. Crossover

In Crossover, new chromosomes are created by mixing the genes of the existing chromosomes.

v. Mutation

Mutation of some genes in the chromosomes induce randomness in the population. This step is the reason why GA is claimed to converge to global optima.

C. *Optimization by PSO*

Particle Swarm Optimization (PSO) proposed by Eberhart and Kennedy [25] is an evolutionary algorithm based on swarm theory. It has been extensively used for optimizing values of unknown parameters in pattern recognition experiments [26]. It does not have the crossover, mutation steps of GA. However, the randomness induced in the position of the particles in the swarm that indicate single solutions, help to converge to global optima. The particle position and velocity are iterated over time to stable values. The steps of PSO algorithm are briefly described below.

i. *Initialization*:

Each particle in the swarm has a position vector and a velocity vector that are randomly initialized at the beginning of the algorithm. Each particle position indicates a possible solution.

ii. *Finding pBest and gBest*:

In each iteration, the local or personal best position (pBest) of each particle is updated and the global best position (gBest) amongst all particles is also updated. The velocity vector in (4) is first updated and the position vector is then updated as per (5). The position and velocity update equations are given below in Eq (4) and (5) respectively.

$$\begin{aligned} &\boldsymbol{velocity}[\boldsymbol{i}] = \boldsymbol{velocity}[\boldsymbol{i}-1] \\ &+\boldsymbol{c}1 * \boldsymbol{rand}() * (\boldsymbol{pBest} - \boldsymbol{position}[\boldsymbol{i}-1]) \\ &+\boldsymbol{c}2 * \boldsymbol{rand}() * (\boldsymbol{gBest} - \boldsymbol{position}[\boldsymbol{i}-1]) \end{aligned} \quad (4)$$

$$\boldsymbol{position}[\boldsymbol{i}] = \boldsymbol{position}[\boldsymbol{i}-1] + \boldsymbol{velocity}[\boldsymbol{i}] \quad (5)$$

Here c1 and c2 are constants and the randomness function ***rand***() in (4) introduces a certain randomness in the solutions formed.

III. Proposed Methodology for Constrained Timetable Schedule Optimization

Our previous work in [22] focusses on scheduling open elective courses by association rule mining for student's preferences [13, 14]. In this work, we schedule core subjects, labs and electives (obtained after the association rule mining in [22]) for the same enrolled students as in [22] with real

University data, available at the university website [11], using a novel set of constraints.

A. Proposed hard, soft, and additional soft constraints

A collection of hard and soft constraints is integrated from different research works (sources are cited) and additional soft constraints are added to make the timetable more human centric as shown below in boxed entries.

Hard Constraints (proposed method):

Hard constraints cannot be violated (as per the rules of the Delhi Technological University [11]). Hard constraints are as follows:

i. Electives are arranged to be taught between certain time slots: 08:00 am - 10:00 am
ii. Core Courses are taught between time slots: 10:00 am - 4:00 pm
iii. All students of B.Tech 6th semester from Information Technology department should enroll for the theory and practical courses (Core + Labs + Both Electives)

Soft Constraints (proposed method):

A 10-point penalty is assigned if soft constraints are violated. The soft constraints are as follows:

i. Classroom capacity should be greater than the size of the class. [2]
ii. No room should be assigned to a course or teacher if its already assigned to another course or teacher for that particular time slot.[2]
iii. Every teacher can teach only one class in a particular time slot. [4]
iv. Every course should have a professor to teach it.
v. Classrooms and lab for separate courses should not overlap with each other. [3]
vi. No student is assigned more than one class at a time. [2]

Additional Soft Constraints (proposed method):

A set of additional constraints are added in this paper to incorporate teacher timing preferences and restrict student mobility between classes and avoid time gaps between classes. These are a part of soft constraints only and a 10-point penalty is assigned if they are violated. These additional constraints are:

i. The students of a class are allotted one room in which they will have all the classes. This is done to avoid the time consumed for students while traveling between classrooms.
ii. There should not be empty slots in between classes in a day for the students.
iii. The teacher's preference to teach in a particular time slot is considered.

The soft constraints include teacher's preferences of timings of lectures that were compiled on an individual basis, and also the consideration for student time management in between classes to reduce the mobility of students in between classrooms and to avoid free time gaps between classes. The novel set of constraints- hard constraints (to be mandatorily satisfied) and soft constraints (to be minimized to the best extent possible) are outlined in the text box. For each soft constraint, we have stated, where appropriate, the reference works that have used these constraints before. We have also included three additional soft constraints that consider the human factors of teacher preferences and student time management as a separate box entitled 'Additional soft constraints'. Timetable scheduling is done for 6th semester undergraduate B.Tech (Information Technology) students (strength of 100 in numbers) of Delhi Technological University. The data for students enrolled for various courses is available at [11]. Students of sixth semester are divided into two batches or sections of 6-A and 6-B. The number of classrooms is limited to two- Room numbers: TW2GF2 and TW1TF3, as opposed to three classrooms in our previous work in [22]. The Professors involved are the current staff of Department of Information Technology, Delhi Technological University (names withheld on request).

There are 5 theory courses (Elective + Core subjects) and two laboratory courses (or labs). The two Electives were derived by association rule mining of student's subject preferences as per our recent work in [22].

a. Electives (Selected by students)
   i. Cyber Forensics (CFCC)
   ii. Machine Learning (ML)
b. Core Courses:
   i. Compiler Design (CD)
   ii. Software engineering (SE)
   iii. Artificial Intelligence (AI)
c. Laboratory:
   i. Compiler Design (CD) Lab
   ii. Artificial Intelligence (AI) Lab

B. Step wise procedure for automated timetable generation using GA

- Fitness function = number of penalties (to be minimized)
- Maximum number of generations=1000

Initial Parameters:

- Mutation Rate = 0.01
- Population size = 100
- Crossover Rate= 0.9

Procedure: Applying GA to create optimized timetable

1. Initialize a timetable with given number of professors, time slots, classes and electives
2. Define population size, mutation rate, crossover rate
3. For each chromosome, calculate number of penalties as per constraints not satisfied
4. Calculate fitness of the chromosomes using fitness function=number of penalties
5. Conduct Selection, Crossover and Mutation for the current population of chromosomes
6. Repeat steps 3, 4, 5 for all generations
7. Analyze the performance with respect to following factors:
   a. No of penalties vs generations.
   b. Time taken for execution of each generation with respect to generation number
8. Compare these factors with that of SA and PSO

C. Step wise procedure for automated timetable generation using SA

- Fitness function = number of penalties

- Maximum number of iterations=1000

Initial Parameters:
- Initial Temperature = 1000
- Cooling Rate = 0.05

Procedure: Applying SA to create optimized timetable

1. Initialize the timetable with the number of professors, time slots, classes and subjects.
2. Initialize temperature T to be very high (T=1000).
3. Initialize the cooling_rate.
4. Repeat while (T > 1)
   a. Define current node C and randomly select a neighbor N.
   b. Evaluate ΔE = fitness(N)- fitness(C).

fitness(N)= number of penalties calculated for soft constraint violation in new node

fitness(C)= number of penalties calculated for soft constraint violation in current node

   a. If fitness(N) < fitness(C) i.e. ΔE≤0, choose the neighbor N with a probability = 1.
   b. If fitness(N) > fitness(C) i.e. ΔE >0, choose the neighbor N with a probability P=-ΔE/T.
   [Fitness Function (or cost function) = -ΔE/T]
   c. T =T * cooling_rate; cooling_rate ϵ [0,1]
   d. Stop when T < 1
5. Analyze the performance with respect to following factors:
   a. No of penalties vs iterations.
   b. Time taken for execution of each iteration with respect to iteration number
6. Compare these factors with that of GA and PSO

D. Step wise procedure for automated timetable generation using PSO
- Fitness function = number of penalties
- Maximum number of iterations=1000

Initial Parameters:
Maximum velocity = 100
Learning factors c1=2, c2=2

Procedure: Applying PSO to create optimized timetable

1. Initialize solutions (particle positions in the swarm)
2. Repeat till maximum iterations reached
   a. Calculate fitness value of each particle.
   b. Fitness value= number of penalties
   c. If Current fitness value <pBest , do
      pBest = current fitness value
   d. else
      retain previous pBest
   e. Assign best particle's pBest to gBest (gBest= best {pBest})
   f. Calculate velocity of each particle using (4)
   g. Update Eq (5) using values from Eq (4)
3. Analyze the performance with respect to following factors:
   a. No of penalties vs iterations.
   b. Time taken for execution of each iteration with respect to iteration number
4. Compare these factors with that of GA and SA

IV. Results and discussions

The experiments were conducted in JAVA eclipse and Python 3.7 software on a 1.6 GHz PC. As stated before, the student data is available at the university website at [11]. The timetables are generated by GA, SA and PSO algorithms by ensuring that no hard constraints are violated and the soft constraints are minimized to the maximum extent.

A. Performance Analysis:

Performance of GA:
The graph in Fig. 1 shows that the number of penalties decrease with the increase in generations in GA. Initially there is a huge decrease in penalties and then it becomes almost constant. The optimized fitness function is when the penalties = 40 at 1000 iterations.

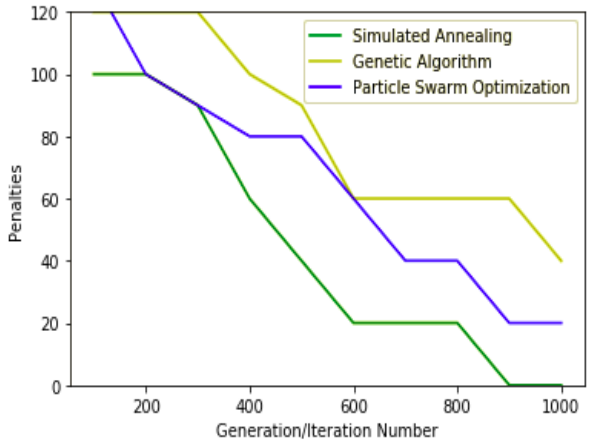

Figure 1. The number of penalties minimized over generation/iteration number (upto 1000) for GA, SA, PSO

Performance of SA:
The graph in Fig. 1 shows that the number of penalties decrease rapidly with the increase in generations in SA. The optimized solution is obtained for iterations close to 900 where number of penalties=0 and an optimized timetable is obtained.

Performance of PSO:
The graph in Fig. 1 shows that the number of penalties decrease with the increase in generations in PSO. The convergence speed is lesser than SA but more than GA. The optimized fitness function is when the penalties =20 at around 900 iterations when an optimized timetable is obtained.

B. Time taken for execution of each generation/iteration
The three algorithms are also compared with respect to their execution time (in ms) for every generation/iteration.

Performance of GA:
Here we study the execution time of each generation with respect to the generation number which is plotted in Fig. 2 for GA. We can observe that for GA, the execution time for consecutive generations decreases consistently with increase in

the number of generations. Initial execution time for GA =50 ms.

Performance of PSO:
Next we study the execution time of each iteration with respect to the iteration number in Fig. 3 for PSO. We can observe that for PSO, the execution time for consecutive iterations decreases (though inconsistently) with increase in the number of iterations. We can also observe that execution time for PSO is less than GA and SA and decreases slowly as compared to GA, SA.

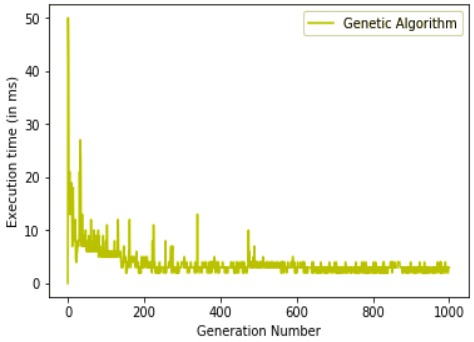

Figure 2. The variation of execution time (ms) over generation/iteration number for GA

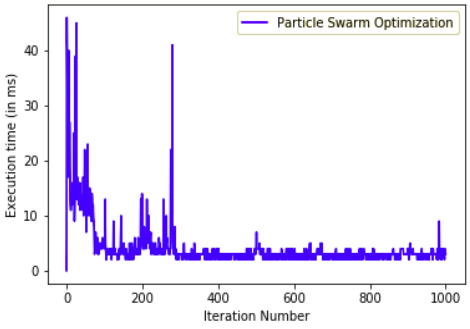

Figure 3. The variation of execution time (ms) over generation/iteration number for PSO

Performance of SA:
Fig. 4 shows the execution time of each iteration with respect to the iteration number for SA. We can observe that for SA, the execution time for consecutive iterations decreases with increase in the number of iterations with last few iterations taking up negligible time. SA has maximum execution time for the initial iterations than either GA and PSO i.e. 250 ms. However, it reduces steeply to less than 50 ms in less than 100 iterations.

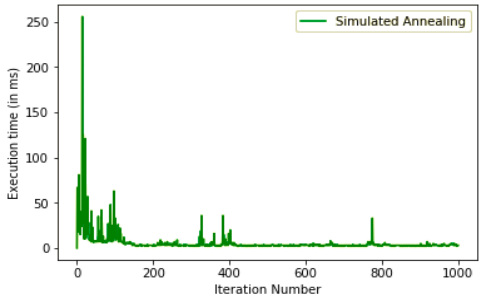

Figure 4. The variation of execution time (ms) over generation/iteration number for SA.

Thus, in summary, SA converges much faster in fewer iterations as compared to PSO and GA as observed from Fig. 1. GA takes more time to converge as compared to the other two algorithms. A comparison of the timing graphs in Figs. 2-4 indicates that initially, PSO followed by GA has lower execution time. However, overall SA has faster execution as observed from the tail end of the graph in Fig. 4. We compare the above graphs with our results in [22] that are shown in Table I for reference. Here also, GA is found to take up more iteration time than SA. The performance analysis for timetable optimization thus establishes the computational efficiency of SA involving local search heuristics. This indicates potential future research directions and would form the basis for incorporating some local search strategies, as in SA, within evolutionary algorithms like GA and PSO for quicker convergence.

C. Timetables generated using GA, PSO and SA

An instance of teacher preference input is shown in Fig. 5. Out of all the professors, two professors of the university gave their choice of slots as in Fig. 5, while the other professors indicated that they were comfortable with all slots. The timetables generated by our method are shown in Tables II and III for GA, Tables IV and V for SA, and Tables VI and VII for PSO, for the two sections 6-A and 6-B. An observation from the timetables is that SA gives a more uniformly distributed schedule than GA and PSO. It satisfies the constraints to a maximum extent such that there are no free gaps in between. PSO, however, creates more free time gaps for students than SA as observed from Tables VI and VII.

We also show for comparison purpose the timetables generated by our previous work in [22]. The timetables generated in [22] are shown for reference in Tables VIII, IX, X for GA, and Tables XI, XII, XIII for SA. Only the open elective courses were scheduled in [22] as seen. The entire core, lab, elective courses are scheduled. As observed from Tables VIII to XIII, the timetables generated for the elective courses, for the same students as in [22], did not incorporate teacher's preferences of slots, and the two classes A batch and B batch were allotted classrooms randomly causing the students to move constantly from one class to the other. In the present set of timetables in Tables II to VII, due to the additional soft constraints introduced, the two professors who requested their respective slots (Fig. 5) had their requests granted. There are not much of free gaps between classes and the student's time is utilized to a great extent. The movement of students from one classroom to the other was also minimized. The A batch was concentrated in classroom no. TW1TF3 while the B batch classes were found scheduled in TW2GF2. We achieved this through the additional soft constraints that are not mandatory, yet their satisfaction in our case proved to be crucial to achieve high efficiency.

Best Case Makespan		Average Case Makespan		Worst Case Makespan	
Generation No.	Time (in ms)	Generation No.	Time (in ms)	Generation No.	Time (in ms)
1	170	7	265	13	413
Simulated Annealing					
Best Case Makespan		Average Case Makespan		Worst Case Makespan	
Iteration No.	Time (in ms)	Iteration No.	Time (in ms)	Iteration No.	Time (in ms)
1	143	5	220	10	311

Table I. Performance analysis for SA and GA for timetables generated in [22]

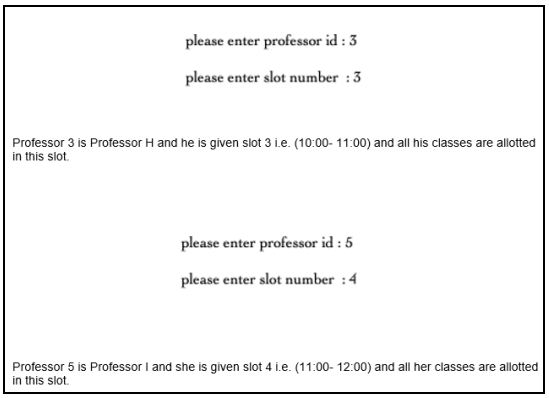

Figure 5. The Graphical User Interface (GUI) for accepting teacher preference against Teacher ID

Room: TW1TF3	6-A							
Time	08:00- 09:00	09:00- 10:00	10:00- 11:00	11:00-12:00	12:00- 13:00	13:00- 14:00	14:00- 15:00	15:00- 16:00
MONDAY	CFCC- A Theory: Prof. B	CFCC- A Theory: Prof. B	SE-A Theory: Prof: E	SE-A Theory: Prof: E			AI Lab - A- G1: Prof: G	AI Lab - A- G1: Prof: G
TUESDAY	ML- A Theory: Prof. A	CFCC- A Theory: Prof. B		SE-A Theory: Prof: E	SE - A - T - G2: Prof: E		AI Lab - A- G2: Prof: A	AI Lab - A- G2: Prof: A
WEDNESDAY	CFCC- T- A- G1: Prof. B	ML- T- A- G1: Prof. A	AI-A Theory: Prof: F	SE - A - T - G1: Prof: E			CD Lab - A- G2: Prof: D	CD Lab - A- G2: Prof: D
THURSDAY	CFCC- T- A- G2: Prof. B	ML- A Theory: Prof. A	AI-A Theory: Prof: F	CD-A Theory: Prof: D		CD Lab - A- G1: Prof: D	CD Lab - A- G1: Prof: D	
FRIDAY	ML- A Theory: Prof. A	ML- T- A- G2: Prof. A	AI-A Theory: Prof: F		CD-A Theory: Prof: D	CD-A Theory: Prof: D		

Table II. Timetable generated by GA for the students of batch 6-A

Room: TW2GF2	6-B							
Time	08:00- 09:00	09:00- 10:00	10:00- 11:00	11:00-12:00	12:00- 13:00	13:00- 14:00	14:00- 15:00	15:00- 16:00
MONDAY	CFCC- B Theory: Prof. C	CFCC- T- B- G2: Prof. B	SE-B Theory: Prof: H		AI Lab - B- G2: Prof: A	AI Lab - B- G2: Prof: A	CD Lab - B- G1: Prof: D	CD Lab - B- G1: Prof: D
TUESDAY	ML- B Theory: Prof. A	CFCC- T- B- G1: Prof. B	SE - B - T - G2: Prof: H	AI-B Theory: Prof: I		CD Lab - B- G2: Prof: D	CD Lab - B- G2: Prof: D	
WEDNESDAY	ML- B Theory: Prof. A	ML- B Theory: Prof. A	SE - B - T - G1: Prof: H	AI-B Theory: Prof: I	CD-B Theory: Prof: J			
THURSDAY	ML- T- B- G1: Prof. A	CFCC- B Theory: Prof. C	SE-B Theory: Prof: H		AI Lab - B- G1: Prof: A	AI Lab - B- G1: Prof: A		
FRIDAY	ML- T- B- G2: Prof. A	CFCC- B Theory: Prof. C	SE-B Theory: Prof: H	AI-B Theory: Prof: I		CD-B Theory: Prof: J	CD-B Theory: Prof: J	

Table III. Timetable generated by GA for the students of batch 6-B

Room: TW1TF3	6-A							
Time	08:00- 09:00	09:00- 10:00	10:00- 11:00	11:00-12:00	12:00- 13:00	13:00- 14:00	14:00- 15:00	15:00- 16:00
MONDAY	ML- A Theory: Prof. A	CFCC- A Theory: Prof. B	CD-A Theory: Prof: D		AI-A Theory: Prof: F	AI Lab - A- G2: Prof: A	AI Lab - A- G2: Prof: A	
TUESDAY	CFCC- A Theory: Prof. B	ML- A Theory: Prof. A	CD-A Theory: Prof: D	CD-A Theory: Prof: D		CD Lab - A- G2: Prof: D	CD Lab - A- G2: Prof: D	
WEDNESDAY	CFCC- T- A- G1: Prof. B	ML- A Theory: Prof. A		AI-A Theory: Prof: F	AI-A Theory: Prof: F			
THURSDAY	CFCC- T- A- G2: Prof. B	ML- T- A- G2: Prof. A	SE - A - T - G1: Prof: E	SE-A Theory: Prof: E	SE-A Theory: Prof: E	CD Lab - A- G1: Prof: D	CD Lab - A- G1: Prof: D	
FRIDAY	ML- T- A- G1: Prof. A	CFCC- A Theory: Prof. B	AI Lab - A- G1: Prof: G	AI Lab - A- G1: Prof: G	SE-A Theory: Prof: E			SE - A - T - G2: Prof: E

Table IV. Timetable generated by SA for the students of batch 6-A

Room: TW2GF2	6-B							
Time	**08:00- 09:00**	**09:00- 10:00**	**10:00- 11:00**	**11:00-12:00**	**12:00- 13:00**	**13:00- 14:00**	**14:00- 15:00**	**15:00- 16:00**
MONDAY	CFCC- B Theory: Prof. C	CFCC- T- B- G2: Prof. B	SE - B - T - G2: Prof: H		CD Lab - B- G1: Prof: D	CD Lab - B- G1: Prof: D	CD Lab - B- G2: Prof: D	CD Lab - B- G2: Prof: D
TUESDAY	CFCC- B Theory: Prof. C	ML- T- B- G2: Prof. A	SE-B Theory: Prof: H	AI-B Theory: Prof: I	CD-B Theory: Prof: J		CD-B Theory: Prof: J	
WEDNESDAY	CFCC- T- B- G1: Prof. B	ML- T- B- G1: Prof. A	SE - B - T - G1: Prof: H	AI-B Theory: Prof: I		CD-B Theory: Prof: J		
THURSDAY	CFCC- B Theory: Prof. C	ML- B Theory: Prof. A	SE-B Theory: Prof: H	AI-B Theory: Prof: I				
FRIDAY	ML- B Theory: Prof. A	ML- B Theory: Prof. A	SE-B Theory: Prof: H		AI Lab - B- G1: Prof: A	AI Lab - B- G1: Prof: A	AI Lab - B- G2: Prof: A	AI Lab - B- G2: Prof: A

Table V. Timetable generated by SA for the students of batch 6-B

Room: TW1TF3	6-A							
Time	**08:00- 09:00**	**09:00- 10:00**	**10:00- 11:00**	**11:00-12:00**	**12:00- 13:00**	**13:00- 14:00**	**14:00- 15:00**	**15:00- 16:00**
MONDAY	ML- T- A- G2: Prof. A	ML- A Theory: Prof. A		AI Lab - A- G2: Prof: A	AI Lab - A- G2: Prof: A		CD-A Theory: Prof: D	
TUESDAY	CFCC- A Theory: Prof. B	CFCC- A Theory: Prof. B		SE-A Theory: Prof: E		CD-A Theory: Prof: D	CD-A Theory: Prof: D	
WEDNESDAY	CFCC- A Theory: Prof. B	ML- A Theory: Prof. A	CD Lab - A- G2: Prof: D	CD Lab - A- G2: Prof: D		AI Lab - A- G1: Prof: G	AI Lab - A- G1: Prof: G	
THURSDAY	CFCC- T- A- G2: Prof. B	ML- A Theory: Prof. A	SE - A - T - G2: Prof: E	AI-A Theory: Prof: F	AI-A Theory: Prof: F		CD Lab - A- G1: Prof: D	CD Lab - A- G1: Prof: D
FRIDAY	CFCC- T- A- G1: Prof. B	ML- T- A- G1: Prof. A	SE-A Theory: Prof: E	SE-A Theory: Prof: E		AI-A Theory: Prof: F	SE - A - T - G1: Prof: E	

Table VI. Timetable generated by PSO for the students of batch 6-A

Room: TW2GF2	6-B							
Time	**08:00- 09:00**	**09:00- 10:00**	**10:00- 11:00**	**11:00-12:00**	**12:00- 13:00**	**13:00- 14:00**	**14:00- 15:00**	**15:00- 16:00**
MONDAY	CFCC- B Theory: Prof. C	ML- T- B- G2: Prof. A	SE-B Theory: Prof: H	AI-B Theory: Prof: I	CD Lab - B- G2: Prof: D	CD Lab - B- G2: Prof: D		
TUESDAY	ML- B Theory: Prof. A	ML- T- B- G1: Prof. A	SE-B Theory: Prof: H		AI Lab - B- G1: Prof: A	AI Lab - B- G1: Prof: A		
WEDNESDAY	CFCC- T- B- G1: Prof. B	CFCC- T- B- G2: Prof. B	SE - B - T - G1: Prof: H	AI-B Theory: Prof: I		AI Lab - B- G2: Prof: A	AI Lab - B- G2: Prof: A	
THURSDAY	ML- B Theory: Prof. A	ML- B Theory: Prof. A	SE-B Theory: Prof: H	AI-B Theory: Prof: I	CD-B Theory: Prof: J			
FRIDAY	CFCC- B Theory: Prof. C	CFCC- B Theory: Prof. C	SE - B - T - G2: Prof: H		CD-B Theory: Prof: J	CD-B Theory: Prof: J	CD Lab - B- G1: Prof: D	CD Lab - B- G1: Prof: D

Table VII. Timetable generated by PSO for the students of batch 6-B

Room: TW2GF2				
Time	**08:00- 09:00**	**09:00- 10:00**	**10:00- 11:00**	**11:00-12:00**
MONDAY	CFCC- B- Theory: Prof. C			
TUESDAY		CFCC- B- Theory: Prof. C	CFCC- B- Theory: Prof. C	
WEDNESDAY			CFCC- A- Theory: Prof. B	CFCC- A- Theory: Prof. B
THURSDAY	CFCC- A- Theory: Prof. B			
FRIDAY				

Table VIII. Timetable generated by GA for the elective subject CFCC in [22] for classroom TW2GF2

Room: TW1TF3				
Time	**08:00- 09:00**	**09:00- 10:00**	**10:00- 11:00**	**11:00-12:00**
MONDAY	CFCC- T- A- G2: Prof. B	CFCC- T- A- G1: Prof. B		
TUESDAY				
WEDNESDAY				
THURSDAY				
FRIDAY			CFCC- T- B- G1: Prof. B	CFCC- T- B- G2: Prof. B

Table IX. Timetable generated by GA for the elective subject CFCC in [22] for classroom TW1TF3

Room: TW3TF3				
Time	08:00- 09:00	09:00- 10:00	10:00- 11:00	11:00-12:00
MONDAY				
TUESDAY		ML- A Theory: Prof. A	ML- A Theory: Prof. A	
WEDNESDAY	ML- T- G3: Prof. A			
THURSDAY			ML- A Theory: Prof. A	ML- T- G4: Prof. A
FRIDAY	ML- T- G1: Prof. A	ML- T- G2: Prof. A		

Table X. Timetable generated by GA for the elective subject ML in [22] for classroom TW3TF3

Room: TW2GF2				
Time	08:00- 09:00	09:00- 10:00	10:00- 11:00	11:00-12:00
MONDAY				
TUESDAY				
WEDNESDAY			CFCC- B- Theory: Prof. C	
THURSDAY		CFCC- T- A- G2: Prof. B		
FRIDAY	CFCC- B- Theory: Prof. C	CFCC- B- Theory: Prof. C		

Table XI. Timetable generated by SA for the elective subject CFCC in [22] for classroom TW2GF2

Room: TW1TF3				
Time	08:00- 09:00	09:00- 10:00	10:00- 11:00	11:00-12:00
MONDAY	CFCC- A- Theory: Prof. B			
TUESDAY	CFCC- T- B- G1: Prof. B	CFCC- A- Theory: Prof. B	CFCC- A- Theory: Prof. B	
WEDNESDAY	CFCC- T- B- G2: Prof. B			
THURSDAY				
FRIDAY			CFCC- T- A- G1: Prof. B	

Table XII. Timetable generated by SA for the elective subject CFCC in [22] for classroom TW1TF3

Room: TW3TF3				
Time	08:00- 09:00	09:00- 10:00	10:00- 11:00	11:00-12:00
MONDAY		ML- T- G2: Prof. A	ML- T- G4: Prof. A	
TUESDAY	ML- A Theory: Prof. A			
WEDNESDAY	ML- A Theory: Prof. A	ML- T- G3: Prof. A		
THURSDAY	ML- A Theory: Prof. A		ML- T- G1: Prof. A	
FRIDAY				

Table XIII. Timetable generated by SA for the elective subject ML in [22] for classroom TW3TF3

We propose to further explore more complex teacher and student preferences for testing the efficiency of our constraint model. This forms the future scope of our work.

V. Conclusions

In this paper, we have investigated genetic algorithm, simulated annealing and particle swarm optimization algorithms for the task of automatically creating an entire timetable (electives + core courses + laboratory sessions) for two separate batches of undergraduate students in a recognized university. We have proposed a novel set of hard constraints and soft constraints suited to the task that incorporate teacher's preferences for slots, and minimizes student movement between classes and the gaps between classes. The three optimization algorithms are compared on the basis of execution time of each generation/iteration with respect to generation/iteration number which decreases most in Simulated Annealing. We also compared the algorithms on the basis of number of penalties vs generations/iterations. With respect to the minimized penalties, the algorithms can be ranked as SA < PSO < GA. Simulated annealing minimizes penalties to the greatest extent and obtains an optimal solution in the least number of iterations.

References

[1] Gür, Şeyda, and Tamer Eren. "Scheduling and Planning in Service Systems with Goal Programming: Literature Review." *Mathematics* 6, no. 11 (2018): 265.

[2] Yazdani, Mehdi, Bahman Naderi, and Esmaeil Zeinali. "Algorithms for university course scheduling problems." *Tehnicki Vjesnik-Technical Gazette* 24 (2017): 241-247.

[3] Akkan, Can, and Ayla Gülcü. "A bi-criteria hybrid Genetic Algorithm with robustness objective for the course timetabling problem." *Computers & Operations Research* 90 (2018): 22-32.

[4] Rozaimee, Azilawati, Adibah Nabihah Shafee, Nurul Anissa Abdul Hadi, and Mohamad Afendee Mohamed. "A Framework for University's Final Exam Timetable Allocation Using Genetic Algorithm." *World Applied Sciences Journal* 35, no. 7 (2017): 1210-1215.

[5] AlHadid, Issam, Khalid Kaabneh, Hassan Tarawneh, and Aysh Alhroob. "Investigation of simulated annealing

components to solve the university course timetabling problem." *Italian Journal of Pure and Applied Mathematics* (2020): 291.

[6] Leite, Nuno, Fernando Melício, and Agostinho C. Rosa. "A fast simulated annealing algorithm for the examination timetabling problem." *Expert Systems with Applications* 122 (2019): 137-151.

[7] Zheng, Shuang, Long Wang, Yueyue Liu, and Rui Zhang. "A simulated annealing algorithm for university course timetabling considering travelling distances." *International Journal of Computing Science and Mathematics* 6, no. 2 (2015): 139-151.

[8] Goh, Say Leng, Graham Kendall, and Nasser R. Sabar. "Simulated annealing with improved reheating and learning for the post enrolment course timetabling problem." *Journal of the Operational Research Society* 70, no. 6 (2019): 873-888.

[9] Wang, Yao-Te, Yu-Hsin Cheng, Ting-Cheng Chang, and S. M. Jen. "On the application of data mining technique and genetic algorithm to an automatic course scheduling system." In *Cybernetics and Intelligent Systems, 2008 IEEE Conference on*, pp. 400-405. IEEE, 2008.

[10] Agrawal, Rakesh, and Ramakrishnan Srikant. "Fast algorithms for mining association rules." In *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215, pp. 487-499. 1994.

[11] [online] reg.exam.dtu.ac.in/register_all.php [*website: Delhi Technological University, New Delhi, India*]

[12] Sastry, Kumara, David E. Goldberg, and Graham Kendall. "Genetic algorithms." In *Search methodologies*, pp. 93-117. Springer, Boston, MA, 2014

[13] Savasere, Ashok, Edward Robert Omiecinski, and Shamkant B. Navathe. An efficient algorithm for mining association rules in large databases. Georgia Institute of Technology, 1995.

[14] Viktoratos, Iosif, Athanasios Tsadiras, and Nick Bassiliades. "Combining community-based knowledge with association rule mining to alleviate the cold start problem in context-aware recommender systems." *Expert systems with applications* 101 (2018): 78-90.

[15] Susan, Seba, Puneet Sharawat, Sandeep Singh, Ramkesh Meena, Amit Verma, and Mukesh Kumar. "Fuzzy C-means with non-extensive entropy regularization." In *Signal Processing, Informatics, Communication and Energy Systems (SPICES), 2015 IEEE International Conference on*, pp. 1-5. IEEE, 2015.

[16] Shen, Jia-ji, Hui-zhu Dong, Yi-ran Su, and Zhi-gang Zhang. "Application of Genetic Algorithm and Simulated Annealing Algorithm for Course Scheduling Problem." In *2019 International Conference on Modeling, Analysis, Simulation Technologies and Applications (MASTA 2019)*. Atlantis Press, 2019.

[17] Abdullah, Salwani, and Hamza Turabieh. "Generating university course timetable using genetic algorithms and local search." In *Convergence and Hybrid Information Technology, 2008. ICCIT'08. Third International Conference on*, vol. 1, pp. 254-260. IEEE, 2008.

[18] Hossain, Sk Imran, M. A. H. Akhand, M. I. R. Shuvo, Nazmul Siddique, and Hojjat Adeli. "Optimization of university course scheduling problem using particle swarm optimization with selective search." *Expert Systems with Applications* 127 (2019): 9-24.

[19] Susan, Seba, and Aparna Bhutani. "A Novel Memetic Algorithm Incorporating Greedy Stochastic Local Search Mutation for Course Scheduling." In *2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, pp. 254-259. IEEE, 2019.

[20] Adrianto, Dennise. "Comparison Using Particle Swarm Optimization and Genetic Algorithm for Timetable Scheduling." *Journal of Computer Science* 10, no. 2 (2014): 341

[21] Zhang, Defu, Yongkai Liu, Rym M'Hallah, and Stephen CH Leung. "A simulated annealing with a new neighborhood structure based algorithm for high school timetabling problems." *European Journal of Operational Research* 203, no. 3 (2010): 550-558.

[22] Susan, Seba, and Aparna Bhutani. "Data Mining with Association Rules for Scheduling Open Elective Courses Using Optimization Algorithms." In *International Conference on Intelligent Systems Design and Applications*, pp. 770-778. Springer, Cham, 2018.

[23] Holland, John Henry. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press, 1992.

[24] Henderson, Darrall, Sheldon H. Jacobson, and Alan W. Johnson. "The theory and practice of simulated annealing." In *Handbook of metaheuristics*, pp. 287-319. Springer, Boston, MA, 2003.

[25] Eberhart, Russell, and James Kennedy. "Particle swarm optimization." In *Proceedings of the IEEE international conference on neural networks*, vol. 4, pp. 1942-1948. 1995.

[26] Susan, Seba, Rohit Ranjan, Udyant Taluja, Shivang Rai, and Pranav Agarwal. "Neural Net Optimization by Weight-Entropy Monitoring." In *Computational Intelligence: Theories, Applications and Future Directions-Volume II*, pp. 201-213. Springer, Singapore, 2019.

.

Author Biographies

Seba Susan received her Ph.D in Image Processing from the Electrical Engineering Department of Indian Institute of Technology (IIT), Delhi in 2014. She is currently Associate Professor in the Department of Information Technology, Delhi Technological University (DTU), India. Her research areas are Machine Learning, Computer Vision, Speech & Natural Language Processing.

Aparna Bhutani completed her B.Tech in Information Technology from Pune University in 2013. She is currently a postgraduate student in Information Systems in the Department of Information Technology, Delhi Technological University (DTU), Delhi, India. Her research area is exploring the use of evolutionary algorithms for scheduling problems.