I. J. Mathematical Sciences and Computing, **2020, 5, 1-14**
Published Online October 2020 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmsc.2020.05.01

A Multi-Objective Optimization Approach for Solving AUST Classtimetable Problem Considering Hard and Soft Constraints

Md Shahriar Mahbub
Ahsanullah Institute of Science and Technology, 141-142 Love road, Dhaka 1208, Bangladesh

Shihab Shahriar Ahmed
Enosis Solutions, Banani, Dhaka 1212, Bangladesh

Kazi Irtiza Ali
Samsung Research & Development Institute Bangladesh, Sonargaon Road, Dhaka 1205, Bangladesh

Md. Taief Imam
CoKreates Limited, Kawran Bazar, Dhaka 1216, Bangladesh

Received: 11 May 2020; Accepted: 15 July 2020; Published: 08 October 2020

Abstract: Preparing a class timetable or routine is a difficult task because it requires an iterative trial and error method to handle all the constraints. Moreover, it has to be beneficial both for the students and teachers. Therefore, the problem becomes a multi-objective optimization problem with a good number of constraints. There are two types of constraints: hard and soft constraint. As the problem is an NP-hard problem, population based multi-objective optimization algorithms (multi-objective evolutionary algorithm) is a good choice for solving the problem. There are well established hard constraints handling techniques for multi-objective evolutionary algorithms, however, the technique is not enough to solve the problem efficiently. In the paper, a smart initialization technique is proposed to generate fewer constraints violated solutions in the initial phase of the algorithm so that it can find feasible solutions quickly. An experimental analysis supports the assumption. Moreover, there are no well-known techniques available for handling soft constraints. A new soft constraints handing technique is proposed. Experimental results show a significant improvement can be achieved. Finally, proposed combined approach integrates smart initialization and soft constraints handling techniques. Better results are reported when comparing with a standard algorithm.

Index Terms: Class timetable problem, Smart initialization, Multi-objective evolutionary algorithm, hard constraint, soft constraint.

1. Introduction

A class timetable problem belongs to a large class of scheduling problems that is very difficult to solve. A class timetable problem refers to the preparation of the weekly schedule to deliver lectures to students. It requires the proper scheduling of all the course lectures to available rooms and time-slots under a set of constraints.

A proper university timetable is essential to the management as it makes the efficient utilization of university resources (e.g., teachers' hour, theory and lab classroom). Efficient utilization of the resources under certain constraints make the problem is an optimization problem. In fact, the problem is an NP-hard problem [Lovelace (2010), Cooper and Kingston (1995)]. Traditionally, a timetable is prepared manually with multiple iterative processes, moreover, the preparation is one of the most tedious and labor-intensive jobs. Therefore, an automatic algorithm is required to solve the problem.

Many research work regarding preparing university and school timetable have been done [Kristiansen and Stidsen(2013), McCollum(2006)]. Most of the works have been done for preparing a school timetable where the course structures are simple. Almost all the research works mentioned in the survey papers [Kris- tiansen and Stidsen(2013)] and [McCollum(2006)] consider the problem as a single objective optimization problem such as [Lukas et al.(2009)Lukas, Aribowo, and Muchri, Ahmad and Shaari(2016), Kazarlis et al.(2005)Kazarlis, Petridis, and Fragkou, Chaudhuri and De(2010), Al-Jarrah et al.(2017)Al-Jarrah, Al-Sawalqah, and Al-Hamdan]. Formulation of the problem as a multi-objective optimization problem is the recent interest trend as mentioned in the survey article [McCollum (2006)]. However, only two papers have been found [Datta et al. (2006) Datta, Deb, and Fonseca] and [Carrasco and Pato (2000)] regarding this aspect. Carrasco et al. [Carrasco and Pato (2000)] formulated the teacher/class timetable problem by identifying a number of hard and soft constraints. The goal is to minimize two objectives: i) minimization of the constraints violation related to teachers, ii) minimization of constrains violation related to students. The authors also developed crossover and mutation to solve the problem. NSGA-II is applied to solve the problem. Datta et al. [Datta et al. (2006) Datta, Deb, and Fonseca] made an attempt to solve class timetable problem of IIT Kanpur. When formulating the problem, the authors identified six hard constraints and three soft constraints. Within three soft constraints, two soft constraints are chosen as conflicting objectives. A modified NSGA-II is applied to solve the problem.

Ahsanullah University of Science and Technology (AUST) offers courses though a close credit system [Asrar (2012)] whereas most of the university offers courses through open credit system. The following section presents the details of the class timetable problem of AUST. The problem has two distinct contradictory objectives, six hard constraints and one soft constraint.

To make a step forward in this field, a novel improved approach is proposed to solve the multi-objective timetable problem efficiently. It is already mentioned that the problem has many hard and soft constraints, therefore, it is a hugely complex problem where finding feasible solutions would take a long time. Moreover, the random initialization technique generates huge number of infeasible solutions that delays the algorithm to find feasible solutions. Therefore, a smart initialization approach has proposed to generate fewer constraints violated individuals in the early stage of the algorithm. The proposed smart initialization technique can be integrated into any meta-heuristic algorithm. The impact of the techniques is measured by integrating it into the initial phase of NSGA-II [Deb et al. (2002) Deb, Pratap, Agarwal, and Meyarivan].

There are well established techniques for handling hard constraints, however, no good soft constraint handling techniques are available in the literature. Most of the time, soft constraints are considered as an objective. In this paper, the soft constraint of a problem is considered as a condition, not as an objective. In this regard, a novel soft constraint handling technique has proposed and integrated into NSGA-II. The comparison with a standard model shows better results.

In summary, our contributions are as follows:

1. Formulation of AUST CSE timetable problem as a multi-objective optimization problem
2. Proposed smart initialization technique to efficient handling of hard constraints
3. A novel soft constraints handling approach

2. AUST Class timetable problem

Computer science and engineering (CSE) department offer a bachelor program called Bachelor of Science in computer science and engineering that is a four years program. Each year consists of two semesters (Fall and Spring). There is a total of eight semesters in the program. A student of a particular year and semester is called by his/her studied year and semester (for example, 11 is called for the students who study in 1st year and 1st semester and so on.) Each semester is divided into two or three sections. In a Spring semester, 11, 21, 31 and 41 semesters have 3 sections (named as A, B, and C) each, all other semesters have two sections each.

In CSE department, there are seven theory classrooms and eight labs. Most of the theory classrooms are similar in size; there are around 50 seats in each theory classrooms. However, a lab classroom has a seating arrangement of 25 students, there are 25 computers in a lab room. Therefore, a section has to divide into two lab groups (e.g., for section A: A1, A2). All the theory and lab courses have to be conducted within these classrooms.

In each semester, fixed courses (five theory courses and three to four lab courses) have been offered. All the theory courses are 3 credit hour courses, that means, a theory course is conducted three times in a week. However, the credit hour of lab courses is either 1.5 or 0.75. A lab course with 1.5 credit hours is conducted once a week, however, a lab with 0.75 credit hours is conducted once a fortnight for a particular group. The duration of each theory class is 50 minutes, and a lab is conducted for two hours and thirty minutes. There is no gap within the classes.

The slots for theory and lab classes start at 8.00 AM and end at 3.00 PM. We call three consecutive theory classes or a lab is a session. There are mainly three sessions: morning, noon and afternoon. Morning, noon and afternoon start at 8:00 AM, 10:30 AM and 1:00 PM, respectively. The concept of a session has a significant emphasis on AUST routine. For example, a section (A) of semester 11 has 3 theory classes starts from 8.00 AM in a room called 7A03, the three theory classes end at 10.30 AM. As section (B) is divided into 2 lab groups, two groups of section B can have two lab courses in two different labs in the morning session.

3. Problem formulation

The main idea of AUST CSE routine optimization is to schedule the classes in such a way that students spend less time in the university, whereas, the teachers can take classes in a relaxed way. It ensures maxi- mum utilization of university resources, at the same time, it ensures better teaching-learning environment. Moreover, the routine optimization problem has some constraints to consider.

3.1 Objectives

The class routine has to be beneficial for both students and teacher. Therefore, we want to optimize two objectives: i) minimizing the total students hours spend in the university, ii) maximizing total teachers hours spend in the university.

$$SH(R) = \sum_{s\in S} \sum_{d\in D} endSlot_d^s - startSlot_d^s \#(1)$$

R presents the routine, SH represents total student hours spend having classes by a student group, S represents a set of students from all the semesters and the set of all the weekdays is presented by D. $\mathrm{endSlot}_d^s$ refers the end time of the last slot of student s on weekday d and the starting time of the student s on weekday d is referred by $\mathrm{startSlot}_d^s$.

$$TH(R) = \sum_{t\in T} \sum_{d\in D} endSlot_d^t - startSlot_d^t \#(2)$$

Where T presents the set representing all the teachers of the department.

3.2 Constraints

There can be two kinds of constraints in a problem: Hard and soft constraint. A solution that violates hard constraints is considered as an infeasible solution, A feasible solution of a problem has to satisfy all the hard constraints. All the solutions that violate soft constraints are still feasible solutions, however, the solutions violating fewer soft constraints are preferable.

In the routine problem, we are considering six hard constraints and one soft constraint.

3.2.1 Hard Constraints

A teacher cannot have more than one class at the same time. Mathematically,

$$HC_1(R) = \sum_{t\in T} \sum_{d\in D} \sum_{sl\in Slot} DC_{t,d,sl}(R) \#(3)$$

Where Slot is a set of all theory and lab slots. Moreover, $\mathrm{DC}_{t,d,sl}(X)$ is a function that returns 1 if there is a conflict found. Conflicts can be found if a teacher t has another class on day d in the same slot sl. When detecting conflicts, the function does not only consider theory or lab classes, it considers all combinations among theory and lab courses. For example, a teacher has a lab class on Saturday starting from 8.00 AM, the duration of the lab is 2 hours and 30 minutes, therefore, the lab ends at 10.30 AM. Please consider now that a theory class is scheduled from 9.40 AM on Saturday for the same teacher. Therefore, a conflict has been detected.

Similarly, a group of students cannot have more than one class at the same time. Mathematically,

$$HC_2(R) = \sum_{s\in S} \sum_{d\in D} \sum_{sl\in Slot} DC_{s,d,sl}(R) \#(4)$$

$\mathrm{DC}_{s,d,sl}(X)$ is a function that returns 1 if there is a conflict found.

It is not a good practice to overburden the students. Therefore, it is better to distribute the classes as much as possible throughout the weekdays. Please keep in mind that we want to minimize the student time, that means, the optimizer may try to schedule classes in one weekday and try to free another weekday. To prevent the possibility, a hard constraint, that a student group (from a particular semester and section) do not have more than three theory and one lab class in a day, is set.

$$HC_3(R) = \sum_{s\in S} \sum_{d\in D} \{TC_s^d(R) > n_c\} \text{ and } \{LC_s^d(R) > n_l\}\#(5)$$

Where $\mathrm{TC}_s^d(R)$ is a function that takes routine and finds the number of theory courses of a particular group of students (s) in any weekday (d). Similarly, $\mathrm{LC}_s^d(R)$ finds the number of lab courses for the student. Finally, n_c and n_l represents the number of theory courses and the number of lab courses, respectively, in a weekday for a particular group of students.

In the same way, it is not a good idea to have more than two lab class in a day.

$$HC_4 = \sum_{s\in S}\sum_{d\in D} \{LC_s^d(X) > n_l\}\#(6)$$

where n_l is set to 2.

Similarly, the students do not have more than four theory classes in a day.

$$HC_5(R) = \sum_{s\in S}\sum_{d\in D} \{TC_s^d(R) > n_c\}\#(7)$$

It is preferable that all the theory classes are held consecutively. For example, a group of students has three theory classes and one lab in a day. It is not a good practice that a lab is conducted after one theory class and next two theory classes are held afterward. Moreover, it is even worse if the last two theory classes are held in two different classrooms. Therefore, it is better to have three consecutive theory classes in a room in a particular session.

$$HC_6(R) = \sum_{s\in S}\sum_{d\in D}\sum_{sn\in SN} \{FR_{s,d,sn}(R) - 1\}\#(8)$$

Where SN is the set of sessions (i.e., in AUST, there are three sessions), FR(R) finds out the number of rooms used for conducting theory classes of a session for a group of students.

3.2.2 Soft Constraints

The routine optimization problem can have one soft constraint. The teachers can provide preferred session choices for each weekday. The optimizer tries to optimize the routine by respecting teachers' preferences as much as possible. Mathematically, the soft constraint can be formulated as follows:

$$SC_1(R) = \sum_{t\in T}\sum_{d\in D} \alpha_t * FCS\{R_t^d(R), PS_t^d\}\#(9)$$

Where $\mathrm{R}_\mathrm{t}^\mathrm{d}(\mathrm{R})$ is the teachers' routine of a day and $\mathrm{PS}_\mathrm{t}^\mathrm{d}$ is the preferred session choice of teacher t. FCS is a function that finds the number of courses scheduled outside of the preferred slot of a teacher. α_t is the weighting parameter for the teacher t. The idea is to prioritize the preference of a higher designated teacher over a lower one (e.g., associate over assistant professor). The weighting parameter of a higher designated teacher is larger than a lower one. The value of the weighting parameter of a teacher is decided by his/her designation.

3.3 Mathematical Formulation

Finally, the problem can be formulated as follows:

$$\begin{gathered} Minimize\ SH(R), \#(10) \\ Maximize\ TH(R), \\ Subject\ to\ HC_i(R) = 0, i = 1, \dots 6. \\ SC_1(R)\ as\ minimum\ as\ possible \end{gathered}$$

Someone may argue that the SC_1 can be added as an additional objective. In the case of single-objective optimization, the soft constraints are added as a penalty function [Kerrigan and Maciejowski (2000)]. In the case of multi-objective optimization, most of time, it is added as an additional objective such as [Datta et al. (2006) Datta, Deb, and Fonseca] and [Singh et al. (2014) Singh, Asafuddoula, and Ray]. However, adding soft constraint as an additions objective has some additional problems: i) it makes the problem more complex, ii) the soft constraints may not contradictory to the objectives of the problem. Therefore, it would be a better approach to consider soft constraints as constraints and handle separately from objectives.

4. Methodology

The methodology section is divided into five main subsections. Firstly, the proposed modification of different components of a multi-objective optimization algorithm is presented. Secondly, the problem representation for the optimization algorithm is discussed. The third section presents a repair function. Finally, the fourth and fifth subsection discuss the techniques for smart initialization and handling soft constraints.

4.1 Algorithm

AUST CSE routine problem is formulated as a multi-objective optimization problem, therefore, a multi- objective optimization algorithm is required to solve the problem. Multi-objective evolutionary algorithms [Deb (2001)] are often used to solve the multi-objective optimization problems mainly because of its population-based approach. There are many multi-objective evolutionary algorithms are available in the literature, such as NSGA-II [Deb et al. (2002) Deb, Pratap, Agarwal, and Meyarivan], SPEA2 [Zitzler et al. (2001) Zitzler, Laumanns, and Thiele] and so on. As the goal of the paper is not comparing different algorithms for solving the AUST class timetable problem, any well-established algorithm in the field of multi-objective optimization could be used. Therefore, it is decided that NSGA-II is used to solve the problem as NSGA-II is used for solving many practical problems.

The original NSGA-II is not enough to solve the complex problem efficiently. The upper part of Fig. 1 shows the typical steps of NSGA-II. Two main modifications are made to adapt the algorithms to handle the specific problem. As it is mentioned earlier that the problem has many hard constraints, therefore, a smart initialization step replacing the random initialization to generate fewer constraints violated population. Moreover, the problem has a soft constraint. In the original approach, there is no technique to satisfy soft constraints. Therefore, a modified binary tournament selection is proposed instead of binary tournament selection. In addition, after the phase of crossover and mutation, an individual may get unacceptable representation. Therefore, a repair function is proposed to make the unacceptable representation back to normal representation. The lower part of the figure shows the steps of modified/proposed approach. The modifications are marked as light red color.

4.2 Problem representation

To represent the problem an N size one-dimensional array is used. N represents all total available slots for six weekdays for all the available resources that the CSE department has. Here, resources refer to theory and lab rooms. CSE has seven theory rooms and eight separate computer labs. Considering that we have six weekdays, three-session per day and three theory classes per session, the total possible number of theory slots are 378 (6*7*3*3). Similarly, the total available lab slots are 144 (6*8*3*1). Please note that only one lab class can be held in one session. Therefore, there is a total of 522 slots in a week. The figure 2 presents the different aspect of array representation of the problem. An array/vector of size 522 that contains all the course slots, each index of the vector represents a slot. The vector is divided into two parts. The first represents all theory slots starting from 8.00 AM Saturday and ends with 3.00 PM Thursday. Please note that the array/vector indexing starts with 0 for theory and the indexing of lab slots starts with 378. Moreover, the figure also shows the morning, noon and afternoon sessions for Saturday and theory room 7A03, and lab room 7B01. The $\uparrow$ shows the position of the session head (a pointer points to starting index of a session). As three theory courses can be taken place in a session, therefore, the session heads are three indexes apart from each other. Lab session heads are apart by one as a lab course holds in a session. The concept of session head will be used in the smart initialization section. Afterward, it is required to understand which course will be placed in which slot. Therefore, a list of all weekly offered courses is made. It is mentioned earlier that there are 3 credit, 1.5 credit and 0.75 credit courses. When listing the weekly courses, 3 credit courses are listed three times for a section, 1.5 credit lab courses are listed one time for each group (please note that each section is divided into two groups) and 0.75 credit courses are listed one time for two groups. Please consider an example with three courses (C_t, $C_l^{1.5}$ and $C_l^{0.75}$) and one section (A: A1 and A2). C_t is a 3.0 credit theory course, $C_l^{1.5}$ and $C_l^{0.75}$ are the lab courses with 1.5 and 0.75 credit, respectively. Therefore, C_t is listed three times, $C_l^{1.5}$ is listed two times (for A1 and A2) and $C_l^{0.75}$ is listed one time (for A1/A2, that means, if A1 group has the course this week, A2 group will have the course in next week). Please keep in mind that 300 theory courses and 97 lab courses are listed in the semester. Therefore, we must have some free slots. 78 (378 - 300) theory and 47 lab free slots are available. Finally, each course and free slots are represented by a unique numeric number starting from 0. According to the Fig. 2, T_x, T_y and T_z are unique theory course numbers assigned to 0, 1 and 2 indexes of the array.

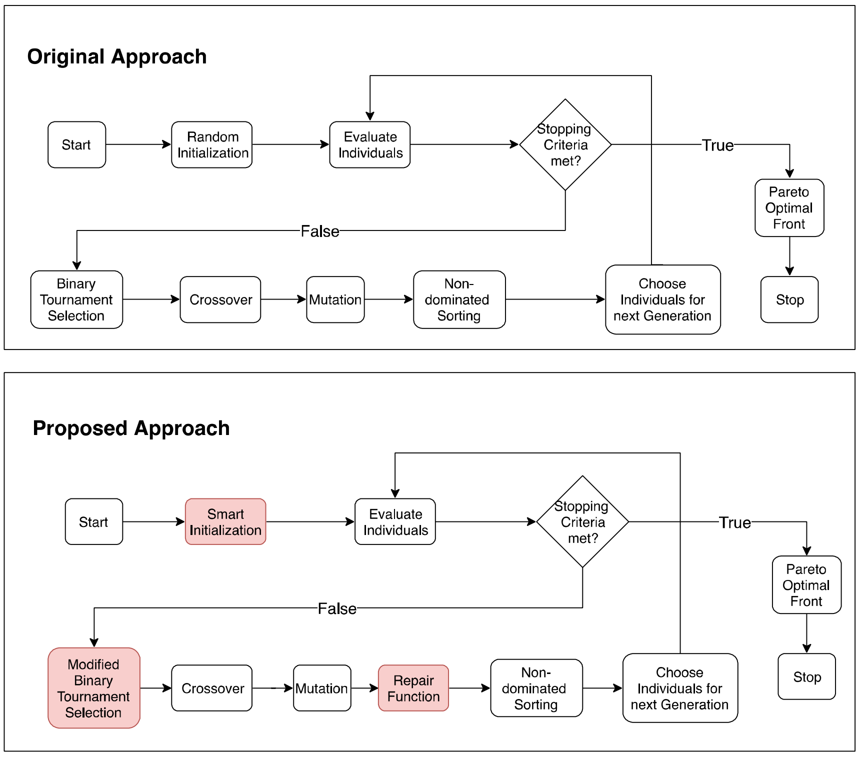
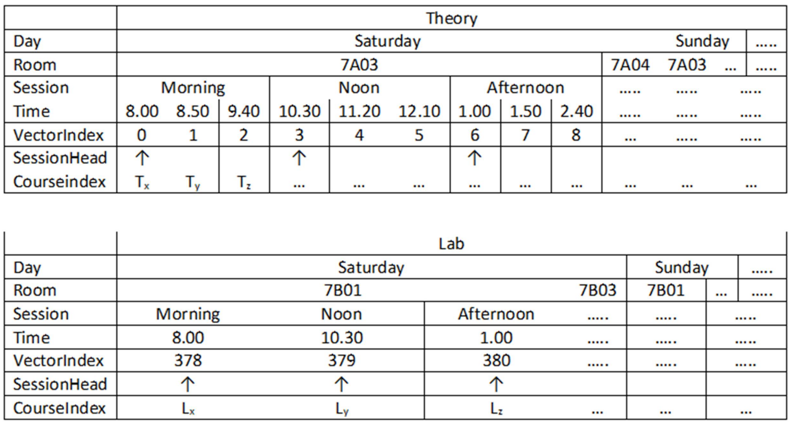


Fig. 1. Original approach and proposed approach

4.3 Repair function

There is a good possibility that a theory course is assigned in the lab slot or a lab course is placed in a theory slot after the crossover or mutation phases. This kind of occurrence may produce imperfect representation. Therefore, a repair function is proposed so that it would be possible to repair an imperfect solution.



	Theory											
Day	Saturday										Sunday	.....
Room	7A03									7A04	7A03 ...	.....
Session	Morning			Noon			Afternoon			.....	.....	.....
Time	8.00	8.50	9.40	10.30	11.20	12.10	1.00	1.50	2.40	.....	.....	.....
VectorIndex	0	1	2	3	4	5	6	7	8	...	.....	.....
SessionHead	↑			↑			↑					
Courseindex	T_x	T_y	T_z	...	...	...	...	...	...	...	...	...

	Lab					
Day	Saturday				Sunday	.....
Room	7B01			7B03	7B01	... .
Session	Morning	Noon	Afternoon	.....	.....	.....
Time	8.00	10.30	1.00	.....	.....	.....
VectorIndex	378	379	380	.....	.....	.....
SessionHead	↑	↑	↑			
CourseIndex	L_x	L_y	L_z	...	...	...

Fig. 2. Problem representation.

Algorithm 1 Algorithm for repair a solution

```
1.  for all theory courses found in a lab slot do
2.      Search for a lab course in theory slot
3.      if A misplaced lab course is found then
4.          Swap the lab course with the theory course
5.      else
6.          Search for a free slot in theory slots
7.          Swap the theory course with found free slot
8.      end if
9.  end for
```

Algorithm 1 shows the steps of repair function for misplaced theory courses. The repair function also includes the steps for misplaced lab courses that are very similar to theory courses.

4.4 Smart initialization

The idea of smart initialization comes forward to handle the hard constraints of AUST CSE routine/timetable problem. The random initialization procedure produces individuals that violate so many constraints; it would need a huge amount of time (evaluations) to get feasible individuals (please see the results) from the random population. Therefore, a new approach is required to initialize the population. The goal of smart initialization is to produce solutions that violate low number of constraints in the initial stage of the algorithm. Please note that the only goal of smart initialization is to speed up the search process.

The idea of smart initialization is based on greedy approach. Algorithm 2 shows the details of the approach. The idea is to fill up the sessions with courses while maintaining some constraints. There are four constraints implemented in four functions: f_1, f_2, f_3 and f_4 (line 5 to line 8 of the algorithm).

For all required numbers of theory sessions, a random theory course is selected that has not appeared in the vector (line 11) yet. Afterward, a random theory session is selected while maintaining consistency with four constraints (from line 12 to line 14). Subsequently, two random theory courses are selected having the same year and semester of previously selected course (line 15). Selected three courses are assigned to the array. Finally, all the free slots are assigned to the vector (line 20). The similar approach is taken to assign lab courses (from line 21 to line 28).

Please note that generated initial solutions are not feasible solutions; the generated solutions are fewer constraints violated solutions. One can argue that it is possible to produce feasible solutions with a more sophisticated greedy approach. However, the greedy approach is not enough to produce a feasible solution, it requires a recursive algorithm to produce a feasible solution. The recursive algorithm is a computationally expensive algorithm that is ignored for generating the initial population.

4.5 Soft constraints handling

There has been extensive literature available to handle hard constraints in the field of multi-objective optimization specifically multi-objective evolutionary algorithm [Fonseca and Fleming (1998), mez (2009), Woldesenbet et al. (2009) Woldesenbet, Yen, and Tessema, Michalewicz and Schoenauer (1996)]. There is a good number of literature available that develop a methodology for handling soft constraint ([Jin et al. (2009) Jin, Tsang, and Li, Singh et al. (2014) Singh, Asafuddoula, and Ray, Tsang and Jin (2006), Jin et al. (2009) Jin, Tsang, and Li, Tsang and Jin (2006)]) while dealing with single-objective optimization problem. However, Singh et al. (2014) Singh, Asafuddoula, and Ray, Datta et al. (2006) Datta, Deb, and Fonseca are the very few that consider handling soft constraints in the multi-objective evolutionary algorithms. In the mentioned papers ([Singh et al. (2014) Singh, Asafuddoula, and Ray, Datta et al. (2006) Datta, Deb, and Fonseca]) the authors consider constraints as an additional objective.

Algorithm 2 Algorithm for generating initial populations

```
1: $n_T$                ▷ Number of theory session required to cover all theory courses
2: $n_L$                ▷ Number of lab session required to cover all lab courses
3: $vector$             ▷ The array representing the problem
4: $SH$                 ▷ a pointer points to starting index of a session
5: function: $f_1(T)$   ▷ is the student group (that takes a particular course, $T$ ) free at $SH$ time slot? returns: true or false
6: function: $f_2(T,d)$ ▷ is there more than one theory session for the student group taking $T$ on a day $d$? returns: true or false
7: function: $f_3(T,d)$ ▷ are there more than two labs for the student group taking $T$ on a day $d$? returns: true or false
8: function: $f_4(T,d)$ ▷ are there more than three theory and one lab for the student group taking $T$ on a day $d$? returns: true
   or false
9: function: $d(SH)$    ▷ find the day on the index of $SH$
10: for all $n_T$ theory sessions do
11:     Randomly select a theory course ($T1$) that is not appeared in the array
12:     do
13:         Pick a random $SH$
14:     while $f_1(T1, d(SH)$ and $f_2(T1, d(SH))$ and $f_3(T1, d(SH))$ and $f_4(T1, d(SH))$
15:     Randomly selected two theory courses ($T2$ and $T3$) having same year and semester of course $T1$ that not appeared in
    $vector$
16:     vector[SH + 0] = $T1$
17:     vector[SH + 1] = $T2$
18:     vector[SH + 2] = $T3$
19: end for
20: Assign all the free theory slots where vector is not assigned yet
21: for all $n_L$ lab sessions do
22:     Randomly select a Lab course ($L$) that is not appeared in the array
23:     do
24:         Pick a random SH
25:     while $f_1(T1, d(SH)$ and $f_3(T1, d(SH)$ and $f_4(T1, d(SH)$
26:     vector[SH] = $L$
27: end for
28: Assign all free lab slots randomly where vector is not assigned
```

Any of the approaches are not suitable for us, because we plan to consider soft constraints separately from objectives. Therefore, we would like to introduce a mechanism for handling soft constraints. The basic idea is to extend the hard constraint handling technique to incorporate a mechanism of soft constraints. While using NSGA-II for optimizing a problem, hard constraints are handled using binary tournament selection [Deb (2001)]. The basic idea of binary tournament selection is very simple: two individuals are randomly chosen and the better individual is selected as a parent. In this aspect three scenarios could happen:

Case (1): Both individuals are infeasible
Case (2): One individual is feasible and other does not
Case (3): Both individuals are feasible
    Sub-case (a): Individuals are in different non-dominated fronts
    Sub-case (b): Individuals are in same non-dominated front

Here, an infeasible individual refers to the individual that violates hard constraints. The following rules are used for each of the cases.

Case (1): The solution that violets less hard constraints is chosen
Case (2): The feasible solution is chosen
Case (3): Better solution with respect to objective value is chosen
    Sub-case (a): Solution belongs to better non-dominated front is chosen
    Sub-case (b): Solution that have better crowding distance [Deb et al. (2002) Deb, Pratap, Agarwal, and Meyarivan] is chosen

We have modified the sub-case 3(b) to incorporate the concept of handling soft constraint. The modification of case (3) is shown below:

Case (3): Better solution with respect to objective values is chosen
    Sub-case (a): Solution belongs to better non-dominated front is chosen
    Sub-case (b): Solution that violates less soft constraint is chosen

It is clear from the modification that the crowding distance is sacrificed to integrate soft constraints. One could argue that sacrificing crowding distance would make an algorithm performs poorly. However, crowing distance is not only maintained in a binary tournament selection but also it is applied in the sorting of a population in NSGA-II [Deb et al. (2002) Deb, Pratap, Agarwal, and Meyarivan]. Therefore, the performance will not be worsened dramatically. Please see the results (section 5.4 and section 5.5).

5. Comprehensive Experiments and Results

The first section reports some general settings for the experiments. The second section compares the optimized solution with the manually prepared solution. The impact of smart initialization is reported in the third section. The fourth section shows the results of the modified binary tournament. The experiments are divided into three sections. The final section compares different approaches integrated with smart initialization.

5.1 General Experimental Settings

The proposed methodologies presented here are implemented in jMetal [Durillo and Nebro (2011)], a JAVA-based multi-objective meta-heuristic framework. Moreover, all the common experimental settings related to all the three experiments are presented in the section. Specific settings will be presented to the specific section, if required.Table 1 shows the general parameter settings that are used in the experiments. Additionally, two-point crossover [Magalhaes-Mendes (2013)] and swap mutation [Cicirello and Cernera (2013)] are used as crossover and mutation operators, respectively. The values of population size and the number of evaluations is relatively large compared to normal settings; please note that on average 362 (108,600) evaluations are required to get feasible solutions in the population (see Table 4. Therefore, a large population and evaluations are required to solve the particular problem. Each evaluation takes approximately 19.5 milliseconds (on average) to complete, smart initialization takes approximately 44 milliseconds to generate a single individual. Repair function takes 43 milliseconds, moreover, evaluating hard and soft constraints of a solution takes approximately 0.16 and 0.044 milliseconds, respectively. Each run takes approximately five and half hours. For each phase of the experiments, the algorithm runs independently 20 times to facilitate statistical analysis.

Table 1 General parameter settings

Parameters	value
Population size	300
Crossover probability	0.90
Mutation probability	0.20
Total number of evaluations	900,000
Number of independent runs	20

5.2 *Compare with manually generated solution*

In this section, a comparison is made between multi-objective optimization and the manually prepared routine. Each semester a committee (called routine committee) from the department prepares the class timetable. The manually generated routine takes around 1 to 2 weeks to complete. The objective values for manual routine are presented in Table 2. Please note that the students' hours need to minimize while the teachers' hours need to be maximized.

Table 2 Objective values for manual routine

Objective	value
Total students' hour	880
Total teachers' hour	576

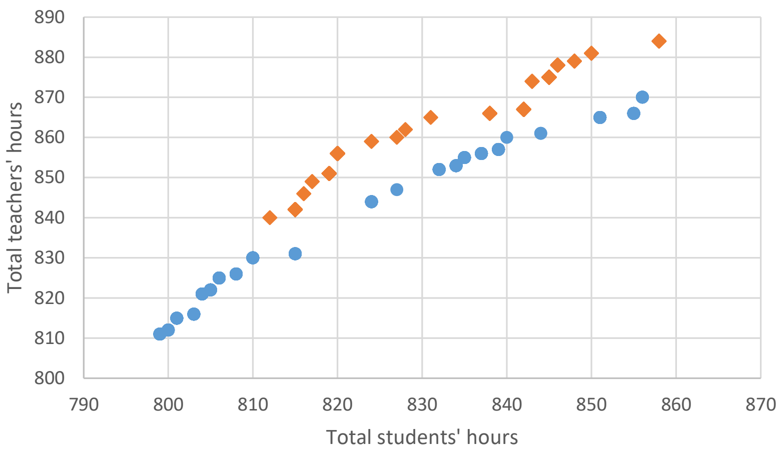


Fig. 3. Two instances of found approximation of Pareto-optimal fronts

Fig. 3 presents two instances of found approximation of Pareto-optimal fronts. It is not possible to show all the Pareto fronts because of the space constraint. The X-axis represents total students' hours and the Y-axis represents total teachers' hours. When comparing with the manual routine, remarkable improvement can be observed. All the Pareto front solutions of two different runs are better on both objectives. It is worth to mention that all the found solutions are better than the manual solution. Moreover, a Pareto front provides management many options to choose from.

5.3 *Impact of Smart Initialization*

Please recall that the main purpose of smart initialization is to generate individuals that violate less hard constraints. Therefore, the algorithm has more time to optimizing the solutions rather than finding feasible solutions.

To investigate the effect of smart initialization over random initialization (RI), two evaluation metrics are defined. The first metric is the generation number when a first feasible solution is produced in the population. The second metric is the generation number when no infeasible individuals exist in the population. When comparing different techniques, lower metrics values indicate better results. As the simulation runs multiple times (20 times), the average generation numbers are reported.

In total two approaches are compared. One of the approaches is random initialization. In addition, two approaches of smart initialization are investigated: i) all the individuals of a population are generated using smart initialization (SIA), ii) half of the total population is generated using smart initialization and the rest of the population is generated using random approach (SIH). The latter approach is investigated to test the idea of mixing random and smart individuals.Table 3 reports the number of average generation numbers. It shows that when SIA is used, the algorithm finds a first feasible solution faster (in terms of generation number) than any other technique. Similarly, SIA approach reaches faster producing a population with all feasible solutions. There is not much difference between SIA and SIH. Please note that RI is not able to produce any feasible solution within 3000 generations (900,000/300). We also further investigated by increasing stopping generation at 6000 (1,800,000 evaluations), however, no feasible solution is produced. Therefore, it can be concluded that without the smart initialization approach it would be very difficult to optimize the routine.

Table 3 Average generations numbers for different approaches

Approaches	Average generation no, when first feasible individual appears	Average generation no. when no infeasible individuals in a population
RI	0	0
SIA	373	386
SIH	387	399

5.4 *Effect of modified binary tournament selection*

The modified binary tournament (MBT) selection is introduced to minimize the soft constraints violation as much as possible. To assess the impact of modified binary tournament, binary tournament (BT) technique (found in literature) is compared with MBT. For the comparison, the average soft constraints violations (ASCV) for two approaches for each run are compared. The ASCV is calculated by $\frac{\sum_{i=1}^{N} SC_1(R)}{N}$ where N represents the number of individuals in a population that violate soft constraints.

Table 4 reports the starting and ending ASCV for two approaches for 10 independent runs. Please note that starting ASCV is not reported from 1st generation; it is because the compared approach is not even activated until all hard constraints are satisfied. Therefore, the reporting generation number is the generation when all hard constraints are satisfied for all the individuals. It is worth to mention that the reporting generation numbers are different for BT and MBT approaches. It is completely predictable as the initial populations for two different runs are different, moreover, other operations (i.e., crossover, mutation and selection) are based on random act. Therefore, two approaches are activated in two different times of evolution.

The results show that MBT can reduce ASCV for each run, whereas the ASCV stays almost constant for BT technique. It also shows that the ASCV does not reach 0 (that is ideal). Reaching 0 ASCV may not be possible as it is not a primary goal of the algorithm. We do not report the other 10 runs because of space constraint, moreover, very similar results are achieved for the other 10 runs.

Table 4 Compare average soft constraint violations (SCV) for BT and MBT

	BT			MBT		
	Reported Generation	Starting ASCV	Ending ASCV	Reported Generation	Starting ASCV	Ending ASCV
Run # 1	613	1131.76	1182.30	309	1014.17	726.47
Run # 2	458	1110.23	1159.77	283	1115.10	769.91
Run # 3	361	1225.44	1357.84	926	1077.38	776.22
Run # 4	286	1340.44	1322.67	471	1087.99	790.50
Run # 5	197	947.02	1073.60	295	1250.42	925.99
Run # 6	276	951.66	1050.97	253	996.12	640.87
Run # 7	246	1399.29	1425.03	292	974.39	728.32
Run # 8	442	966.82	1073.42	355	1137.91	696.12
Run # 9	311	911.99	1098.73	366	1114.39	909.93
Run # 10	315	956.03	975.51	312	1275.88	938.88

To understand the effect of applying MBT more clearly, a generation vs ASCV plot is presented in Fig. 4. The x-axis represents the generations and the y-axis represents the ASCV for a particular run. It is clearly seen that when applying MBT, the ASCV decreases over time, however, in the case of BT, the ASCV stays almost constant

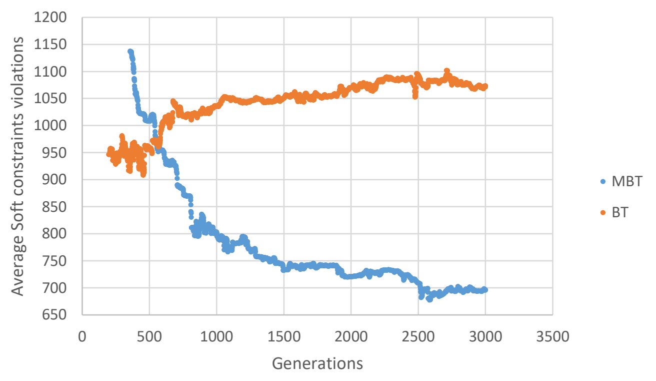


Fig. 4. Generations vs SCV plot

One can argue that the modification of binary tournament selection may decrease the performance of the algorithm. Therefore, the algorithmic performance of the different selection techniques with smart initialization has been compared in the next section.

5.5 Compare smart initialization with binary tournament and modified binary tournament

In the section, a comparison is made among three different approaches with smart initialization i.e., all of the individuals generated using smart initialization and binary tournament (SIABT), half of the individuals generated using smart initialization and modified binary tournament (SIHMBT), and all of the individuals generated using smart initialization and modified binary tournament (SIAMBT).

Several evaluation metrics can be found in the literature for comparing multi-objective optimization algorithms. In the paper, two well-known metrics, Hyper-volume (HV) [Zitzler and Thiele (1999)] and additive epsilon approximation [Zitzler et al. (2003) Zitzler, Thiele, Laumanns, Fonseca, and Da Fonseca], are used to compare the performance. HV calculates the total volume covered by the set of solutions in a Pareto front form a particular reference point. An algorithm with a higher value of HV than other algorithms indicates better algorithm. On the other hand, epsilon determines the minimum translation distance among all the distances for the solutions in a Pareto front. The translated distance of a solution is calculated by translating the solution in objective space to dominate the true Pareto front (tPF). a tPF is required to calculate values of epsilon. HV requires a reference point that set manually or generated from tPF. As the tPF is not known for AUST CSE routine optimization problem, it is approximated by merging all the Pareto fronts (total 60 runs) and find out tPF from the merged front.

The means and standard deviations of two metrics for three different approaches are presented in Table 5. The dark gray shade indicates better results. It is clear from the table that SIAMBT outperforms SIHMBT and SIHBT on all the metrics.

Table 5 Mean and standard deviation (in subscript) for different quality indicators.

Evaluation metrics	SIABT	SIHMBT	SIAMBT
HV	$7.75e-03_{2.4e-02}$	$1.26e-01_{1.2e-01}$	$1.38e-01_{1.4e-01}$
Epsilon	$8.51e+01_{1.6e+01}$	$6.26e+01_{1.8e+01}$	$6.15e+01_{1.5e+01}$

In addition to previously reported results in Table 5, it is also required to perform statistical analyses to understand the performance of the algorithms. A Mann-Whitney U-tests [Mann and Whitney (1947)] is performed on all the evaluation metric values for all twenty different runs. The function of Mann-Whitney U-tests to determine whether two samples (i.e., metrics values) come from the same population or not by comparing the null hypothesis against an alternative hypothesis. The null hypothesis will be rejected if the corresponding p-value is less than 0.05.

Table 6 Mann-Whitney U-tests: p-values for different metrics when comparing smart initialization with different techniques.

	P-values		
Evaluation metrics	Compare SIAMBT and SIHMBT	Compare SIAMBT and SIABT	Compare SIHMBT and SIABT
HV	0.839	2.274^{-06}	1.088^{-05}
Epsilon	0.715	5.518^{-05}	3.192^{-04}

Table 6 presents p-values for two metrics. When comparing SIAMBT and SIHMBT, the p-values are higher than 0.05 for both the metrics, as a result, it can be concluded that there is not a difference when all the individuals or half of the individuals are generated by smart initialization process. Therefore, any approach can be used. However, there is a significant difference among the metrics values when comparing SIAMBT with SIABT and SIHMBT with SIABT, as all p-values are less than 0.05.

5.6 Discussion

Finally, which combinations of techniques should be used? According to section 5.3 it is obvious to use smart initialization than random initialization. Section 5.4 concludes that modified binary selection is far superior to binary selection to minimize the soft constraint violations as much as possible. Section 5.5 investigates that which combination of proposed techniques performs best. It turns out that smart initialization (all individuals generated) with modified binary selection perform best for handing multi-objective routine problem.

6. Conclusion

Formulation of class timetable problem as a multi-objective optimization problem is rare, however, it is very practical to prepare a routine in such a way that it is connivance for both student and teacher. A timetable problem becomes complex for having many hard and soft constraints. Therefore, a multi-objective optimization algorithm solely cannot solve the problem efficiently. Two new components of the algorithm are developed to help the algorithm. The test results show significant improvements in terms of time and algorithmic performance when compared with a standard algorithm for solving the problem.

Two different future directions can be taken. The first direction is related to the problem. When optimizing the problem using a population-based algorithm, the algorithm generates many duplicate solutions. The duplicate solutions are different in decision space but the same in objective space. In other words, different routines can have the same objective values. Therefore, the search process is slowed down. Hence, it is required to develop techniques to handle this kind of duplicate solution.

The second direction is related to handling soft constraints. There is not a generalized approach exists for handling soft constraint in population-based multi-objective optimization field. Besides, many other real-world problems can have soft constraints. Hence, a theoretically sound and well tested soft constraint handling techniques need to be developed.

References

[1] mez (2009). Constraint-handling in evolutionary optimization. Number volume 198 in Studies in computational intelligence. Springer, Berlin, 2009. ISBN 978-3-642-00618-0.

[2] Ahmad and Shaari (2016). A. Ahmad and F. Shaari. Solving university/polytechnics exam timetable problem using particle swarm optimization. In Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication, page 46. ACM, 2016.

[3] Al-Jarrah et al. (2017) Al-Jarrah, Al-Sawalqah, and Al-Hamdan. M. A. Al-Jarrah, A. A. Al-Sawalqah, and S. F. Al-Hamdan. Developing a course timetable system for academic departments using genetic algorithm. Jordanian Journal of Computers and Information Technology (JJCIT), 3(1), 2017.

[4] Asrar (2012). C. Asrar. Closed and open credits in undergraduate programmes in Bangladesh. The Daily Star, 2012. URL https://www.thedailystar.net/campus/2012/11/04/post.htm.

[5] Carrasco and Pato (2000). M. P. Carrasco and M. V. Pato. A multiobjective genetic algorithm for the class/teacher timetabling problem. In International Conference on the Practice and Theory of Automated Timetabling, pages 3–17. Springer, 2000.

[6] Chaudhuri and De (2010). A. Chaudhuri and K. De. Fuzzy genetic heuristic for university course timetable problem. Int. J. Advance. Soft Comput. Appl, 2(1):100–123, 2010.

[7] Cicirello and Cernera (2013). V. A. Cicirello and R. Cernera. Profiling the distance characteristics of mutation operators for permutation-based genetic algorithms. In the Twenty-Sixth International FLAIRS Conference, 2013.

[8] Cooper and Kingston (1995). T. B. Cooper and J. H. Kingston. The complexity of timetable construction problems. Technical report, University of Sydney, 1995.

[9] Datta et al. (2006) Datta, Deb, and Fonseca. D. Datta, K. Deb, and C. M. Fonseca. Solving class timetabling problem of iit kanpur using multi-objective evolutionary algorithm. KanGAL, Report, 2006006:1–10, 2006.

[10] Deb (2001). K. Deb. Multi-objective optimization using evolutionary algorithms, volume 16. John Wiley & Sons, 2001.

[11] Deb et al. (2002) Deb, Pratap, Agarwal, and Meyarivan. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6(2):182–197, 2002.

[12] Durillo and Nebro (2011). J. J. Durillo and A. J. Nebro. jMetal: A Java framework for multi-objective optimization. Advances in Engineering Software, 42:760–771, 2011. ISSN 0965-9978.

[13] Fonseca and Fleming (1998). C. Fonseca and P. Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms. II. Application example. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, 28(1):38 –47, Jan. 1998. ISSN 1083-4427. doi: 10.1109/3468.650320.

[14] Jin et al. (2009) Jin, Tsang, and Li. N. Jin, E. Tsang, and J. Li. A constraint-guided method with evolutionary algorithms for economic problems. Applied Soft Computing, 9(3):924 – 935, 2009. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2008.11.006. URL http://www.sciencedirect.com/science/article/pii/S1568494608001695.

[15] Kazarlis et al. (2005) Kazarlis, Petridis, and Fragkou. S. Kazarlis, V. Petridis, and P. Fragkou. Solving university timetabling problems using advanced genetic algorithms. GAs, 2(7):8–12, 2005.

[16] Kerrigan and Maciejowski (2000). E. C. Kerrigan and J. M. Maciejowski. Soft constraints and exact penalty functions in model predictive control. 2000.

[17] Kristiansen and Stidsen (2013). S. Kristiansen and T. R. Stidsen. A comprehensive study of educational timetabling-a survey. Technical report, Department of Management Engineering, Technical University of Denmark, 2013.

[18] Lovelace (2010). A. L. Lovelace. On the complexity of scheduling university courses. Master's thesis, California Polytechnic State University, San Luis Obispo, 2010.

[19] Lukas et al. (2009) Lukas, Aribowo, and Muchri. S. Lukas, A. Aribowo, and M. Muchri. Genetic algorithm and heuristic search for solving timetable problem case study: Universitas pelita harapan timetable. In 2009 Second International Conference on the Applications of Digital Information and Web Technologies, pages 629–633, Aug 2009. doi: 10.1109/ICADIWT.2009.5273979.

[20] Magalhaes-Mendes (2013). J. Magalhaes-Mendes. A comparative study of crossover operators for genetic algorithms to solve the job shop scheduling problem. WSEAS transactions on computers, 12(4):164–173, 2013.

[21] Mann and Whitney (1947). H. B. Mann and D. R. Whitney. On a test of whether one of two random variables are stochastically larger than the other. The annals of mathematical statistics, pages 50–60, 1947.

[22] McCollum (2006). B. McCollum. University timetabling: Bridging the gap between research and practice. In in Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling, pages 15–35. Springer, 2006.

[23] Michalewicz and Schoenauer (1996). Z. Michalewicz and M. Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. Evol. Comput., 4(1):1–32, Mar. 1996. ISSN 1063-6560. doi: 10.1162/evco.1996.4.1.1. URL http://dx.doi.org/10.1162/evco.1996.4.1.1.

[24] Singh et al. (2014) Singh, Asafuddoula, and Ray. H. K. Singh, M. Asafuddoula, and T. Ray. Solving problems with a mix of hard and soft constraints using modified infeasibility driven evolutionary algorithm (idea-m). In2014 IEEE Congress on Evolutionary Computation (CEC), pages 983–990, July 2014. doi: 10.1109/CEC.2014.6900239.

[25] Tsang and Jin (2006). E. Tsang and N. Jin. Incentive method to handle constraints in evolutionary algorithms with a case study. In Proceedings of the 9th European Conference on Genetic Programming, EuroGP'06, pages 133–144, Berlin, Heidelberg,2006. Springer-Verlag. ISBN 3-540-33143-3,978-3-540-33143-8.doi:10.1007/1172997612. URL http://dx.doi.org/10.1007/11729976_12.

[26] Woldesenbet et al. (2009) Woldesenbet, Yen, and Tessema. Y. Woldesenbet, G. Yen, and B. Tessema. Constraint Handling in Multiobjective Evolutionary Optimization. IEEE Transactions on Evolutionary Computation, 13(3):514–525, 2009. ISSN1089-778X. doi: 10.1109/TEVC.2008.2009032.

[27] Zitzler and Thiele (1999). E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Transactions on Evolutionary Computation, 3(4):257–271, 1999.

[28] Zitzler et al. (2001) Zitzler, Laumanns, and Thiele. E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. Technical Report TIK-Report No. 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, 2001.

[29] Zitzler et al. (2003) Zitzler, Thiele, Laumanns, Fonseca, and Da Fonseca. E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. IEEE Transactions on Evolutionary Computation, 7(2):117–132, 2003.

Authors' Profiles

Md. Shahriar Mahbub obtained his Ph.D. from the University of Trento, Italy. His Ph.D. research work focused on the development of tools for optimizations of energy systems. In his research, he developed a framework for optimizing an energy system for minimizing more than one objective. Currently, he is working as an associate professor in the department of Computer Science and Engineering at Ahsanullah University of Science and Technology, Bangladesh. His current research focuses on applications of multi-objective optimization algorithms and the development of different techniques for the improvements of multi-objective optimization algorithms.

Shihab Shahriar has been working as a software developer for almost three years. He has completed his graduation in CSE from Ahsanullah University of Science and Technology in late 2018. He has worked as an adjunct lecturer in the same university. He wants to see himself as a successful entrepreneur and so he is working accordingly. https://www.bangladesh-made.com/ is one of the developing websites that he initiates. His favourite pastime includes playing football, gaming etc.

Kazi Irtiza Ali has received his B.Sc. degree in Computer Science and Engineering from Ahsanullah University of Science and Technology, Bangladesh, in 2019. He's currently working as a User Experience Designer at Samsung Research and Development Institute Bangladesh for two years. He's mainly involved with creating digital products for Samsung which have a huge user base. Currently his research interest is mainly focused on human-centered designs.

Md Taief Imam has received his B.Sc. degree in Computer Science and Engineering from Ahsanullah University of Science and Technology, Bangladesh, in 2019. He's currently working as a Software Quality Assurance Engineer at Cokreates Ltd. He's involved with Bangladesh e-Government ERP Project (GRP). This is a project of Planning division, ICT Division, High-Tech Park, BCC, a2i, BUET. He had received an appreciation letter as an SQA Engineer from Bangladesh Computer Council last December to work actively in the QA team. His current goal is to represent himself as a QA professional at an ever growing and challenging corporate environment to excel through work, dedication and quick learning.

How to cite this paper: Md Shahriar Mahbub, Shihab Shahriar Ahmed, Kazi Irtiza Ali, Md. Taief Imam. " A Multi-Objective Optimization Approach for Solving AUST Classtimetable Problem Considering Hard and Soft Constraints ", International Journal of Mathematical Sciences and Computing (IJMSC), Vol.6, No.5, pp.1-14, 2020. DOI: 10.5815/IJMSC.2020.05.01