Optimization of University Course Scheduling Problem using Particle Swarm Optimization with Selective Search

Hossain, S. I., Akhand, M. A. H., Shuvo, M. I. R., Siddique, N., & Adeli, H. (2019). Optimization of University Course Scheduling Problem using Particle Swarm Optimization with Selective Search. *Expert Systems with Applications*, *127*, 9-24. https://doi.org/10.1016/j.eswa.2019.02.026

Published in:
Expert Systems with Applications

Publication Status:
Published (in print/issue): 01/08/2019

DOI:
10.1016/j.eswa.2019.02.026

Document Version
Author Accepted version

Optimization of University Course Scheduling Problem using Particle Swarm Optimization with Selective Search

Sk. Imran Hossain[1], M. A. H. Akhand[1], M. I. R. Shuvo[1], Nazmul Siddique[2] and Hojjat Adeli[3]

[1]Department of Computer Science and Engineering, Khulna University of Engineering & Technology, Bangladesh, e-mail: imran@cse.kuet.ac.bd; akhand@cse.kuet.ac.bd; insan_shuvo@cse.kuet.ac.bd;

[2] School of Computing, Engineering and Intelligent Systems, Ulster University, United Kingdom, e-mail: nh.siddique@ulster.ac.uk;

[3]Department of Civil, Environmental and Geodetic Engineering, The Ohio State University, USA, e-mail: adeli.1@osu.edu;

Abstract:

University Course Scheduling Problem (UCSP) is a highly constrained real-world combinatorial optimization problem. Solving UCSP means creating an optimal course schedule by assigning courses to specific rooms, instructors, students and timeslots by taking into account the given constraints. Several researches have reported solution for UCSP using Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and Harmony Search (HS). Among them a few PSO based methods with different adaptations are proved to be effective solving UCSP. In general, the existing methods consider relatively simple UCSPs where the UCSP is first transformed into numeric domain then apply PSO to solve it. In this study, Particle Swarm Optimization with Selective Search (PSOSS), a novel PSO based method has been proposed for solving highly constrained UCSP by introducing swap sequence-based velocity, selective search and forceful swap application with repair mechanism. The proposed method has been tested for optimising course schedule of Computer Science and Engineering Department of Khulna University of Engineering & Technology which is relatively complex with many hard and soft constrains. Experimental results show the superiority of the proposed method compared to other prominent methods (e.g., GA, HS) for tackling the UCSP.

Keywords: University Course Scheduling, Particle Swarm Optimisation, Selective Search, Forceful Swap Application, Swap operator, Repair Mechanism.

1. Introduction

The timetabling is a real-life optimization problem that deals with scheduling of events of fixed number of timeslots and resources satisfying the soft and hard constraints and the necessary objectives as close as possible (Chiarandini et al., 2006; Mencia et al., 2016; Tang et al., 2018). The timetabling problem is NP-hard requiring a huge volume of computation for finding solutions, which grows exponentially with increasing problem size (Yue et al., 2017). Timetabling problem has to satisfy two kinds of constraints namely hard and soft constraints where hard constraints are the conditions that must be satisfied for a working timetable whereas the soft constraints are conditions that may be violated but they affect the solution quality (Pongcharoen et al., 2008).

Timetabling problem has found many applications in different domains such as employee allotment, transport systems, educational organizations, sports activities and industrial applications. An organization may come up with different timetabling problems for different applications. In higher educational institutions, examination and course scheduling are two important common and challenging tasks for optimizing physical and human resources

(Mushi, 2006). Among the various timetabling problems, university course scheduling is the most complex task requiring a set of soft and hard constraints to be satisfied. This task is well-known as University Course Scheduling Problem (UCSP) in the literature.

The goal of UCSP is to assign all theoretical classes and laboratory sessions to instructors, rooms and time slots considering the hard and soft constraints in a way such that there is no dispute in these assignments (Feizi-Derakhshi et al., 2012). The challenges of the UCSP are the constraints, for example, instructors' dispositions, educational policies of the school, students' cohort, availability of teaching staffs and other physical resources. In UCSP, each instructor can teach one class at a time slot and students can just go to one class at any given time. Other similar kind of constraints are treated as hard constraints that must be satisfied. The common soft constraints of the UCSP are instructors' preferences for favoured days and timeslots, and preferences for the maximum length of breaks between teachings, which must be satisfied to the extent possible. In the UCSP the main issue is to handle room allocation for lectures considering maximum capacity of room, number of enrolled students in a course or class and other related facilities (Shiau, 2011; Azimi, 2005). Both the hard and soft constraints may vary from institution to intuition based on their resources and facilities. Any resource modification or update (including capacity alteration in resources) requires rescheduling of classes, which is very common at the beginning of a term. UCSP is a more complex combinatorial optimization problem than other combinatorial problems such as Travelling Salesman Problem (TSP). Although almost all metaheuristic methods have been applied to TSP, the number of studies on UCSP is much fewer than that of TSP because too many constraints must be satisfied in UCSP.

A number of meta-heuristic approaches (Yang et al., 2016) have been applied to the UCSP in the last few years. Among them are genetic algorithm (GA) (Wang, 2003; Pongcharoen, 2008; Martinez-alvarez et al., 2016; Li et al., 2017; Zhao et al., 2018), integer linear programming (Boland et al., 2008), tabu heuristic Search (Mushi, 2006), simulated annealing (Abramson et al, 1991), hybrid evolutionary algorithm (Prieto et al., 2016) with variable neighborhood search (Abdullah et al., 2007), hybrid GA with local search (Yang and Jat, 2011), hybrid evolutionary approach with nonlinear great deluge (Obit et al., 2012) and hybrid electromagnetism-like mechanism with great deluge (Turabieh et al., 2009) and Harmony Search (HS) algorithm (Al-Betar et al., 2012; Al-Betar et al., 2009). Wang et al. (2003) investigated GA for the UCSP with multiple constraints, which resulted in a timetable more acceptable to instructors. Mushi (2006) proposed a Tabu Search algorithm for UCSP with emphasis on the University of Dar-assalaam that generates schedule by heuristically minimizing penalties over infeasible solutions. Chiarandin et al. (2006) describe a metaheuristic algorithm based on the set of benchmark instances of 'International Timetabling Competition'. The method combines heuristics, simulated annealing, variable neighborhood descent and tabu search.

Recently, various swarm intelligence (SI) based optimization methods have been investigated for UCSP such as Ant Colony Optimization (ACO) (Ayob and Jaradat, 2009; Li and Zhang, 2013), honey-bee mating optimization algorithm (Sabar et al., 2012). Among different SI methods, Particle Swarm Optimization (PSO) is the most popular due to its simplicity and adaptation ability (Alexandridis et al., 2017). Various PSO based strategies have been examined for UCSP. Shiau (2011) proposed an algorithm considering a bunch of constraints and a repair mechanism for all infeasible solutions. Tassopoulos and Beligiannis (2012) proposed a hybrid PSO algorithm for generating timetable. Chen and Shih (2013) investigated two different versions of PSO, the inertia weight version and the constriction version. Osman (2015) proposed a PSO approach for UCSP of Najran

University. The algorithm consists of two steps: first, the representation of the solutions as particle (i.e. particle encoding) and second the adjustment of the fitness function. The author used the basic PSO equations to adjust each particle's position and velocity.

The main objective of this paper is to solve the highly constrained UCSP problem using a modified PSO based technique. Existing methods transform UCSP to numeric domain and then apply PSO for obtaining a viable solution (Tassopoulos and Beligiannis, 2012; Chen and Shih, 2013). In these cases, the conventional PSO method, used for function optimization, is chosen for the given UCSP. Instead of transforming the UCSP to numeric domain, a better approach would be the modification of the algorithm. Moreover, current methods considered simple instances of UCSP, which fail to provide quality solution for highly constrained scenario. In this paper, swap sequence has been adapted for velocity calculation in UCSP and selective search as well as forceful swap application with repair mechanism has been introduced for handling highly constrained nature of UCSP.

The proposed algorithm for the solution of the UCSP has been applied to the scheduling of department of Computer Science and Engineering of Khulna University of Engineering & Technology (KUET). The UCSP-KUET is considered a highly constrained realistic environment, where course scheduling is a difficult job. The rationale for working with UCSP-KUET is that: resource is much scarcer in KUET compared to western universities which makes it a more difficult task to create an optimal schedule. Experimental study shows that our proposed technique performs better compared to other traditional methods such as GA, HS, Producer-Scrounger Method (PSM) and PSO.

The rest of the paper is structured as follows: Section 2 describes the proposed method for UCSP. Section 3 presents the experimental studies with comparative analysis among algorithms. Finally, section 4 provides some concluding remarks.

2. Optimising UCSP using PSOSS

The following sections contain a brief overview of PSO and detailed description of the proposed PSOSS method for solving UCSP.

2.1 Overview of PSO

PSO developed by Kennedy and Eberhart (1995; 2001) is an optimization algorithm based on the social behaviour of swarms. In PSO, a bird of a flock or fish of a school is represented by a particle, and the swarm is a collection of particles (Chen and Shih, 2013). Every particle in the swarm indicates a candidate solution to the optimization problem. Every particle adjusts its position in the multidimensional search space based on the experience of its adjacent particles and personal experience. Particle uses its personal best position and the best position among its neighbours to move towards an optimal solution. The fitness of each particle is calculated using a fitness function which is associated with the problem at hand. PSO has been a popular technique for solving different constrained optimization problems.

Initially, PSO creates a population of particles randomly. The number of particles to be used in a population is problem dependent. Every particle representing a solution to the problem has three parameters namely velocity, position and fitness. At every iteration, a particle uses its personal best position and also the best position among its neighbours to update its position. This process continues until it reaches a stopping criterion.

Consider a search space of D dimensions consisting of M particles. If a particle's current position is X_p, personal best position is B_p and global best position among all the particles is G then, velocity of a particle V_p is calculated using

$$V_p^{(t)} = iV_p^{(t-1)} + l_1 * r_1\left(B_p - X_p^{(t-1)}\right) + l_2 * r_2\left(G - X_p^{(t-1)}\right) \tag{1}$$

where, i is the inertia factors, $\{l_1, l_2\}$ are learning factors, and $\{r_1, r_2\}$ are random values ranging from 0 to 1. The position of the particle is updated using

$$X_p^{(t)} = X_p^{(t-1)} + V_p^{(t)} \times T \tag{2}$$

where, T represents time and is assumed to be unity. Position of a particle represents a solution and initially each particle is given a random position and a random velocity. In every iteration, updated velocity of each particle is determined using Eq. (1) and particle's position is updated according to Eq. (2). The fitness value of each particle is updated for the new position and the personal best position B_p gets updated if a better fitness value is found compared to the previous one. The global best position G is also updated in the same manner. G is considered as the final result after termination of the operation. The algorithm ends when the stopping criterion is satisfied (Montero et al., 2011).

2.2 PSOSS for solving UCSP

Proposed PSOSS method works with a population of particles in which individual particle represents a feasible solution, calculates velocity of each individual particle using swap sequence and updates each particle with the computed velocity through selective search and forceful swap application. The particle encoding, swap operator and swap sequence, velocity computation, forceful swap application with repair mechanism, selective search, fitness calculation and other operations are described in the following sections.

A. Particle Encoding

PSOSS works with a population of particles and each particle represents the complete schedule for instructors, students, classrooms and laboratories. A particle's solution (S_p) is represented by instructor-wise solutions in a

S_p	I1	I2	I3	….	Im-1	Im

(a) Instructor-wise summary view of a particle.

45 timeslots *X* *m* instructors																
45 timeslots						45 timeslots							45 timeslots				
1	2	3	…	44	45	41	42	43	…	89	90	……	45(*m*-1) +1	45(*m*-1) +2	…	45*m*-1	45*m*
Instructor 1						Instructor 2							Instructor *m*				

Course Code	Course Type	Batch Information	Group Id	Room No	Instructor Id

(b) Detailed view of particle

Figure 1: Particle representation of UCSP for KUET instance.

Time Slot		Sun	Mon	Tue	Wed	Thu	…..	Sun	Mon	Tue	Wed	Thu
	1	1	10	19	28	37	…..	45(*m*-1)+1	…	…	…	…
	2	2	11	20	29	38	…..	45(*m*-1)+2	…	…	…	…
	3	3	12	21	30	39	…..	45(*m*-1)+3	…	…	…	…
	4	4	13	22	31	40	…..	45(*m*-1)+4	…	…	…	…
	5	5	14	23	32	41	…..	…	…	…	…	…
	6	6	15	24	33	42	…..	…	…	…	…	…
	7	7	16	25	34	43	…..	…	…	…	…	
	8	8	17	26	35	44	…..	…	…	…	…	45m-1
	9	9	18	27	36	45	…..	…	…	…	…	45m
		{………..instructor1…………..}						{…………. instructor*m*……………….}				

Figure 2: Mapping of time slots of a particle in days and periods.

one-dimensional matrix as shown in Fig.1. There are 45 continuous timeslots for each instructor and each timeslot identifies one of the nine teaching periods in a day for five working days in a week. Figure 1(a) is the summary view of instructors' solution where I1, I2, and I3 denote the 1st, 2nd and 3rd instructor respectively. Figure 1(b) shows the detailed particle view of 45 time slots for each instructor. The total number of time slots is $45 \times m$ for *m* instructors; timeslots 1–45 for the first instructor, 46–90 for the second instructor, and so on. Each slot comprises of assigned course type, course code, batch information, group id, room no and instructor id as shown in Fig. 1(b). Figure 2 represents mapping of time slots of a particle in days and periods. As an example, slot 44 represents 5th working day's 8th period of the first instructor.

B. Swap Operator and Swap Sequence

A Swap Operator (SO) denotes the index of items to be swapped in a list.

Consider the following list *L*,

L	a	b	c	d	
	0	1	2	3	⟶ index

A *SO(1,3)* produces a new list *L'* as follows:

L' = L + SO(1,3)

L'	a	d	c	b
	0	1	2	3

here, '+' does not mean any arithmetic operation rather it means the swap operation SO(a,b) on *L*.

A Swap Sequence (SS) is a group of SOs defined as follows:

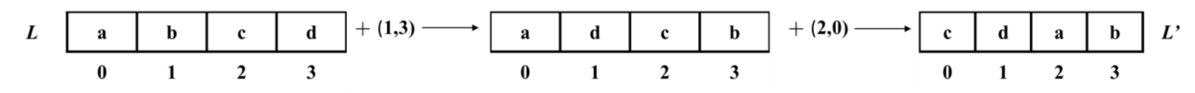
$$SS = \{ SO_1, SO_2, SO_3, \ldots SO_n\} \tag{3}$$

Employment of a SS means application of all the SOs in a SS in that very particular order. Moreover, if applying *SS* on a list A yields a list *B* (i.e., $B = A + SS$), then it can be written as

$$SS = B - A \tag{4}$$

For example, if *SS* = {(1,3), (2,0)} then,

$L' = L + SS$



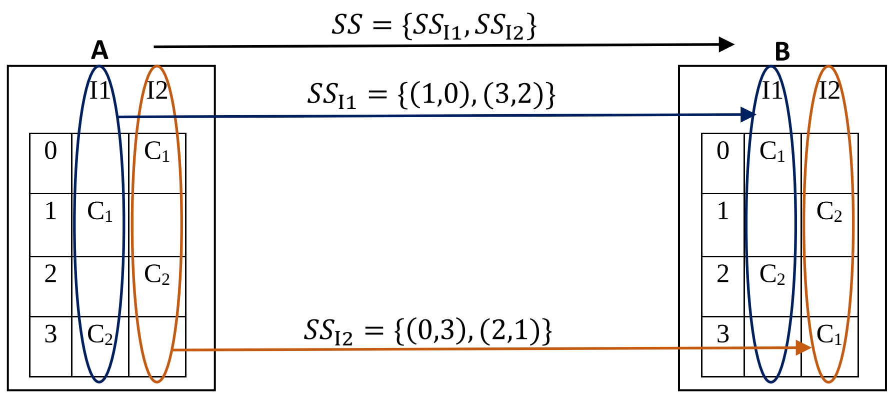


Figure 3: Instructor wise Swap Sequences (SSs) of complete SS.

C. Velocity Computation using Swap Operator and Swap Sequence

SO and SS has been used in the proposed PSOSS method for velocity calculation. The SS to convert one particle's solution to another particle's solution is a collection of swap sequences which is measured in an instructor-to-instructor basis. Consider a UCSP consisting of two instructors I1 and I2 each having two courses C1 , C2 and C3 , C4 respectively. Figure 3 shows two different solutions A and B for the UCSP in consideration. In solution A, instructor I1 has a course C1 in slot no 1 and another one C2 in slot no 3 whereas in solution B, course C_1 is at slot no 0 and C_2 is at slot no 2. So, the required SS for converting the schedule of I1 in solution A to schedule of I1 in solution B is SS_{I1} = {(1,0), (3,2)}. Similarly, SS_{I2} = {(0,3), (2,1)}. So the complete SS for converting solution A to solution B is $SS = \{SS_{I1}, SS_{I2}\}$.

In the proposed method, swap sequence SS is treated as velocity to update a particle's position at each iteration which is calculated using Eq. (5)

$$SS = \gamma SS_{PA} \otimes \alpha(S_{GB} - S_P) \otimes \beta(S_{PB} - S_P) \quad \alpha, \beta, \gamma \epsilon [0,1] \qquad (5)$$

where SS_{PA} is the previously applied velocity, S_{PB} is the previous best solution of the particle, S_{GB} is the global best solution of the swarm and $\{\alpha, \beta, \gamma\}$ are selection probabilities for selecting a bunch of SOs from the corresponding SS. The equation $SS_{GB}(=S_{GB} - S_p)$ represents instructor-wise SSs to reach S_{GB} from S_P and $SS_{PB}(S_{PB} - S_p)$ is the instructor-wise SSs to reach S_{PB} from S_P. However, such simple SS calculation and operation is not suitable for solving UCSP. In UCSP, the solution for one instructor depends on others and intended swapping in an instructor's solution may not be feasible due to unavailability of slots and resources which is held by others. To make the operation useful, Eq. (5) is represented in a different form as follows:

$$SS = \alpha(S_{GB} - S_P) + \beta(S_{PB} - S_P) \otimes \gamma SS_{PA} \qquad (6)$$

As $SS_{GB} = S_{GB} - S_p$ and $SS_{PB} = S_{PB} - S_p$, Eq. (6) can be written as:

$$SS = \alpha * SS_{GB} + \beta * SS_{PB} \otimes \gamma SS_{PA} \qquad (7)$$

After selection of SOs with $\{\alpha, \beta, \gamma\}$, SS becomes

$$SS = SS_{SGB} + SS_{SPB} \otimes SS_{SPA} = SS_{SGB} + SS_M \quad (8)$$

where, SS_{SGB}, SS_{SPB} , SS_{SPA} are the selected SS from SS_{GB}, SS_{PB} and SS_{PA} respectively and SS_M is the swap sequence resulting from merger of SS_{SPB} with SS_{SPA}.

As SS_{SGB} and SS_M can contain redundant SOs, redundant swaps are removed from them and SS_{SGB} and SS_M become SS_{MSGB} and SS_{MM}, respectively, after removing redundant SOs. Finally, SS becomes:

$$SS = SS_{MSGB} + SS_{MM} \quad (9)$$

This final velocity *SS* is then applied using forceful swap application with repair mechanism (described in section 2.2C) and selective search (described in section 2.2D).

C. Forceful Swap Application with Repair Mechanism

In the proposed method, *SS* for solving UCSP consists of swaps to global best (SS_{GB}), swaps to personal Best (SS_{PB}) and Previously Applied Swaps (SS_{PA}). UCSP is highly constrained in nature and most of the constrains are interrelated. Consequently, if a class needs to be shifted to a new time slot then all the involved members such as instructor, students and room need to be free in that time slot. As a result, most of the selected swaps can't be applied because of violation of constrains. Therefore, a portion of SS_{GB} is forcefully applied to present solution S_P to ensure that S_P moves a little towards SS_{GB}. Forcefully applying a SO can result in conflicts so, a repair mechanism is involved in forceful SO application to make sure that no invalid solution results in that process. The repair mechanism works by randomly moving conflicting courses to non-conflicting positions.

D. Selective Search

In proposed method, velocity *SS* is applied using selective search mechanism. In selective search, each solution generated by applying a SO of *SS* is considered as an intermediate solution and the sequence of SOs generating the best intermediate solution is considered as the final velocity which becomes the previously applied velocity for the next iteration.

Suppose, $SS = \{SO_1, SO_2, SO_3, \ldots SO_n\}$ then the selective search can be written as

$$S_p^1 = S_p + SO_1$$
$$S_p^2 = S_p^1 + SO_2 = S_p + SO_1 + SO_2$$

………………………………………………..

$$S_p^n = S_p^{n-1} + SO_n$$

In the above cases, $S_p^1, S_p^2, \ldots\ldots, S_p^n$ are the intermediate solutions and the intermediate solution having the highest fitness becomes the final solution S_p in selective search as defined by the following equation:

$$S_p = \max\{S_p^j\}, \; j = 1,2, \ldots n \quad (10)$$

Finally, the velocity is $SS = \{SO_1, SO_2, SO_3, \ldots SO_j\}$, $1 < j \leq n$.

The ultimate solution S_p in selective search is the intermediate solution possessing the highest fitness value. Thus, the selective search technique explores the opportunity of getting better solution from the intermediate solutions.

E. Fitness Calculation

PERIODS		Sun	Mon	Tue	Wed	Thu	…..	Sun	Mon	Tue	Wed	Thu
	1	0	0	0	-1	5	…..	0	5	4	-1	5
	2	1	3	4	-1	5	…..	2	5	4	2	5
	3	1	3	4	-1	5	…..	2	5	4	2	5
	4	1	3	4	-1	5	…..	2	5	4	2	5
	5	3	5	-1	3	4	…..	1	3	-1	1	-1
	6	3	5	-1	3	4	…..	1	3	-1	1	-3
	7	3	5	-1	3	-1	…..	1	2	-1	1	-5
	8	0	0	-1	3	-1	…..	0	2	-1	1	-1
	9	0	0	-1	3	-1	…..	0	2	-1	1	-1
		{………...Instructor1………}						{……….... Instructor m…………}				

Figure 4: Sample preference values for Instructors.

Each instructor's preference for conducting a class in a particular time slot is represented by an integer value as shown in Fig. 4. A higher value corresponds to a higher preference of an instructor to conduct the class in that particular time slot. Whereas, a negative value shows the instructor's non-preference. The fitness of a particle's solution is calculated by considering fitness of each of the instructors' solution which belongs to that particle's solution using following equation:

$$F_{GS} = \sum_{i=0}^{m} F_{IS_i} \tag{11}$$

where, F_{GS} is the fitness of the particle's solution, and FIS is the fitness of the instructors' solution.

Now, fitness of each instructor's solution is calculated by considering quality and violation of the instructor's solution using the following equation:

$$F_{IS} = Q_{IS} - V_{IS} \tag{12}$$

where, Q_{IS} is the quality of the instructors' solution, and V_{IS} is the violation of the instructor's solution.

Preference values of corresponding positions where courses were assigned to an instructor are summed up to calculate the satisfaction of each instructor's solution. Violation of the instructors' solution is calculated using the following equation:

$$V_{IS} = \sum_{i=1}^{n} 2^{s_i} \tag{13}$$

where, n is the total number of blocks of consecutive classes in an instructor's solution and s is the number of classes in a block.

2.3 PSOSS Algorithm for UCSP

The proposed PSOSS algorithm for solving UCSP is shown in Algorithm 1. The notations and inputs of the proposed algorithm are listed at the beginning of the Algorithm 1.

Algorithm 1: UCSP-PSOSS
Input:
Instructors' Information, Batches' Information,
Courses' Information, Classrooms' Information, Break Times' information,
N - total number of iteration
N_P - total number of particles
α - selection probability of a swap from swap sequence to global best solution
β - selection probability of a swap from swap sequence to personal best solution

γ - selection probability of a swap from previously applied swap sequence
F_p - percentage of swaps to be forced towards global best solution

Output: An optimal solution of UCSP

Variables:

S_{PB} - personal best solution
S_{GB} - global best solution
S_P - current solution
S_R - A random solution
SS_{CA} - currently applied swap sequence
SS_{PA} - previously applied swap sequence
SS_R - random swap sequence for all Instructors
SSH - swap sequence holder for selective search
SH - solution holder for selective search
P - set of particles
T - set of Instructors
CC - set of conflicting classes

create N_P particles and append them to P
for all $p \in P$ **do**
  $S_P \leftarrow$ a random solution
  calculate fitness of S_P as described in section 2.2E
  $S_{PB} \leftarrow S_P$ *//initially current solution is assigned as personal best solution*
  $SS_{PA} \leftarrow \emptyset$
end for
$S_{GB} \leftarrow solution(\max P)$ *//select solution having highest fitness among all the particles*
for $i \leftarrow 1$ **to** N **do**
  for all $p \in P$ **do**
    $SSH \leftarrow \emptyset$
    $SH \leftarrow \emptyset$
    if $SS_{PA} = \emptyset$ **then**
      $SS_{PA} \leftarrow SS_R$
    end if
    $SS_{GB} \leftarrow S_{GB} - S_P$
    $SS_{PB} \leftarrow S_{PB} - S_P$
    $SS_{SGB} \leftarrow \alpha * SS_{GB}$
    $SS_{SPB} \leftarrow \beta * SS_{PB}$
    $SS_{SPA} \leftarrow \gamma * SS_{PA}$
    $SS_M \leftarrow SS_{PB} \oplus SS_{PA}$ *//merge* SS_{SPB} *with* SS_{SPA}
    $SS_{SMGB} \leftarrow swapMinimizer(SS_{SGB})$ *//remove redundant swaps*
    $SS_{MM} \leftarrow swapMinimizer(SS_M)$
    for all $t \in T$ **do**
      $SS_{GBt} \leftarrow SS_{SMGB}[t]$ *//select swap sequence for Instructor t*
      $NS_F \leftarrow F_p * |SS_{GBt}|$
      for $a \leftarrow 1$ **to** NS_F **do**
        $S_p \leftarrow S_P + SS_{GBt}[a]$ ***forcefully*** *//apply* $SS_{GBt}[a]$ *forcefully to* S_P
        $CC \leftarrow$ list of conflicting classes in S_P resulting from $SS_{GBt}[a]$ application
        if $CC \neq \emptyset$ **then**
          for all $cc \in CC$
            move cc to a randomly selected non-conflicting position
          end for
        $SS_{CA} \leftarrow SS_{CA} \cup \{SS_{GBt}[a]\}$
        $selectiveSearch(S_P, SS_{CA}, SH, SSH)$
      end for
      for $a \leftarrow NS_F + 1$ to $|SS_{GBt}|$ **do**
        if $SS_{GBt}[a]$ applicable **then**
          $S_p \leftarrow S_P + SS_{GBt}[a]$
          $SS_{CA} \leftarrow SS_{CA} \cup \{SS_{GBt}[a]\}$
          $selectiveSearch(S_P, SS_{CA}, SH, SSH)$
        end if

end for
end for
for all $t \in T$ **do**
$SS_{Mt} \leftarrow SS_{MM}[t]$
for $a \leftarrow 1$ to $|SS_{Mt}|$ **do**
if $SS_{Mt}[a]$ applicable **then**
$S_p \leftarrow S_P + SS_{Mt}[a]$
$SS_{CA} \leftarrow SS_{CA} \cup \{SS_{GBt}[a]\}$
selectiveSearch (S_P, SS_{CA}, SH, SSH)
end if
end for
end for
$S_P \leftarrow SH$
$SS_{PA} \leftarrow SSH$
calculate fitness of S_P as described in section 2.2 D
if *fitness*$(S_P) >$ *fitness*(S_{PB}) **then**
$S_{PB} \leftarrow S_P$
end if
end for
$S_{GBT} \leftarrow$ *solution*(max P)
if *fitness*$(S_{GBT}) >$ *fitness*(S_{GB}) **then**
$S_{GB} \leftarrow S_{GBT}$
end if
end for

Algorithm 1.1 selectiveSearch
Input: S_P, SS_{CA}, SH, SSH
if $SH = \emptyset \vee fitness(SH) < fitness(S_P)$ **then**
$SH \leftarrow S_P$
$SSH \leftarrow SS_{CA}$
end if

In the proposed algorithm, initial population of particles is generated by creating specified number of particles. Each particle's current solution S_P gets initialized by a random solution. The fitness of each particle's current solution is calculated as described in section 2.2E. S_P also becomes the personal best solution S_{PB} initially. Also, the previously applied swap sequence SS_{PA} is initially empty. Then, the solution having the highest fitness among all the particles is selected as the global best solution S_{GB}. In each iteration, for each particle SH and SSH are emptied to be used for selective search. SH is used to hold best intermediate solution and SSH holds the swap sequence that produces SSH. A random swap sequence is assigned to SS_{PA} if it is empty. Then instructor-wise swap sequences to reach S_{GB} and S_{PB} from S_P are calculated which are represented by $SS_{GB}(=S_{GB} - S_P)$ and $SS_{PB}(=S_{PB} - S_P)$ respectively. Some swaps are selected for each instructor from SS_{GB} based on the selection probability α denoted by $SS_{SGB}(=\alpha * SS_{GB})$. Similarly, $SS_{SPB}(=\beta * SS_{PB})$ and $SS_{SPA}(=\gamma * SS_{PA})$ are the selected swaps from SS_{PB} and SS_{PA} respectively. SS_{SPB} and SS_{SPA} are merged together to $SS_M(=SS_{PB} \oplus SS_{PA})$. Redundant swaps are removed from SS_{SGB} and SS_M using *swapMinimizer*() function, results of which are denoted by SS_{SMGB} and SS_{MM} respectively. After that, for each instructor a portion of SS_{SMGB} is selected using the equation $Fp * | SS_{GBt} |$, where F_P is the force percentage and SS_{GBt} is the swap sequence corresponding to an instructor t. Swaps of this selected portion are forcefully applied to S_P and resulting conflicts are resolved by randomly moving the conflicting classes to non-conflicting positions. Rest of the swaps are applied to S_P if they do not create any conflicts. Similarly, the swaps from SS_{MM} are applied only if they are applicable. Each applied swap gets added to SS_{CA} which holds currently applied swap sequence and. The selective search technique is used after applying each swap to ensure that best intermediate solution is retained. Algorithm 1.1 shows the required steps of selective search. It simply updates *SH* and *SSH* with S_P and SS_{CA} respectively only if S_P is found better than *SH*. Finally, after the application of all the swaps the best intermediate solution *SH* becomes particle's solution S_P and the swap sequence *SSH* that produces *SH* becomes SS_{PA} for next iteration. Then S_{PB} is updated if S_P is found better than S_{PB}. Finally, S_{GB} is recalculated and algorithm goes to next iteration. The algorithm uses a predefined number of iterations *N* as the termination criteria. After termination S_{GB} is considered as the final solution.

2.4 Illustration of the Mechanism of PSOSS with a Sample Problem

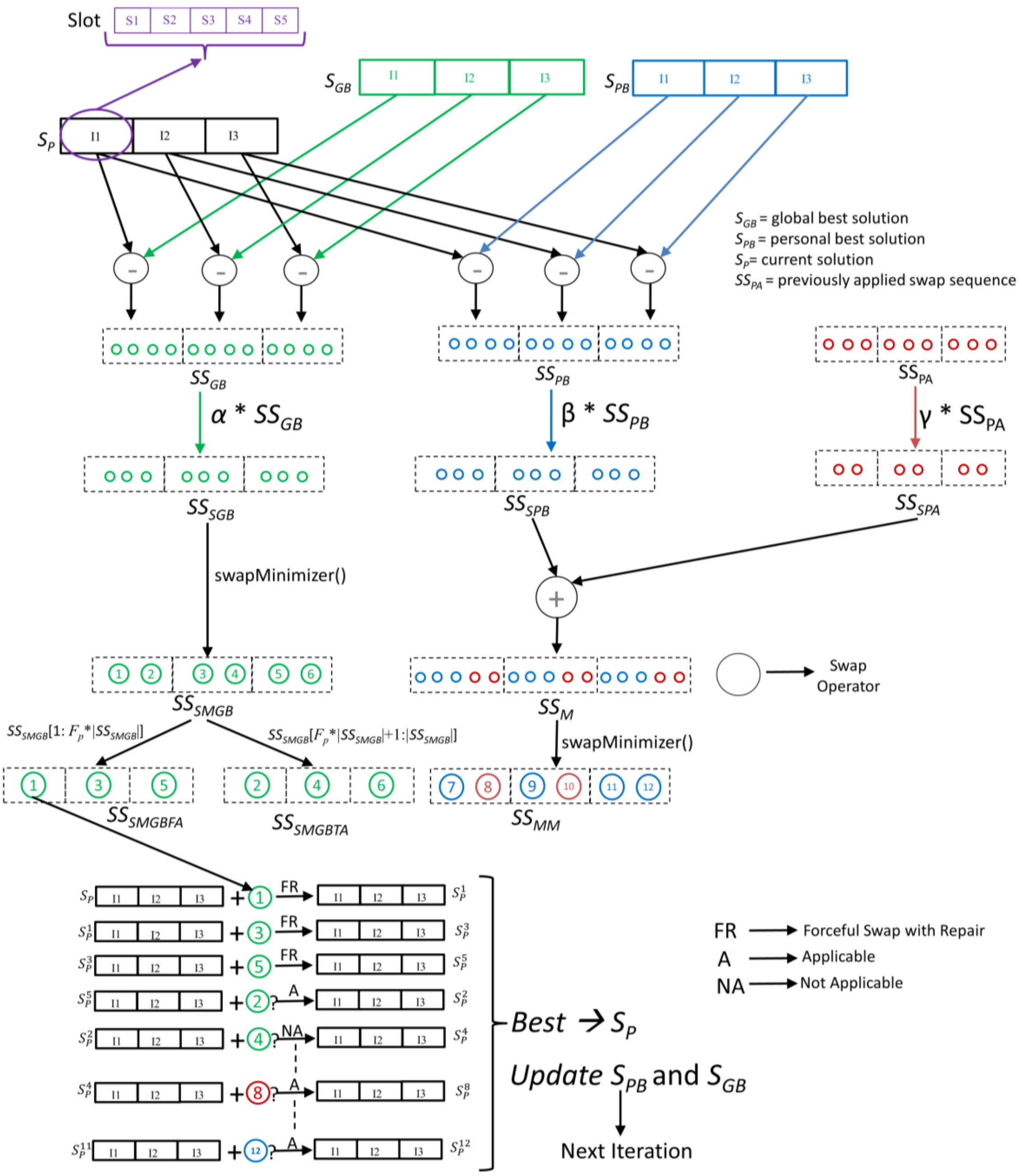


Figure 5: Illustration of the mechanism of PSOSS with a sample problem.

A schematic representation of the proposed method in shown in Fig. 5. Suppose, a system consisting of three instructors I1, I2 and I3, each having five weekly slots that need to be scheduled. In Fig. 5, S_P is a particle's present solution which consists of individual solution of all three instructors, S_{GB} is the global best solution and S_{PB} is the particle's personal best solution. $SS_{GB}(=S_{GB} - S_p)$ represents instructor-wise swap sequences to reach S_{GB} from S_P , and $SS_{PB}(S_{PB}$ - $S_p)$ is the instructor-wise swap sequences to reach S_{PB} from $S_{P.}$ SS_{PA} is the previously applied instructor-wise swap sequences. The circle (○) symbol inside the swap sequences represents a swap operator.

In the first step, some swaps are selected for each instructor from SS_{GB} based on the selection probability α denoted by $SS_{SGB}(=\alpha * SS_{GB})$. Similarly, $SS_{SPB}(=\beta * SS_{PB})$ and $SS_{SPA}(=\gamma * SS_{PA})$ are the selected swaps from SS_{PB} and SS_{PA} respectively. The redundant swaps are removed from SS_{SGB} using *swapMinimizer*() function, result of which is denoted by SS_{SMGB} (swaps numbered 1, 2, 3, 4, 5 and 6 in Fig. 5) . Then, SS_{SPB} and SS_{SPA} are merged together to $SS_M(=SS_{PB} \oplus SS_{PA})$ before removing the redundant swaps from them. The redundant swaps are removed from SS_M using *swapMinimizer*() function, result of which is denoted by SS_{MM} (swaps numbered 7, 8, 9, 10, 11 and 12 in Figure 5). After that, a portion of SS_{SMGB} is selected using the equation $Fp * |SS_{SMGB}|$, where F_P is the force percentage. This selected portion of SS_{SMGB} is denoted by SS_{SMGBFA} (swaps numbered 1, 3 and 5 in Fig. 5) and the rest of the swaps are denoted by SS_{SMGBTA} (swaps numbered 2, 4 and 6 in Fig. 5). Swaps of SS_{SMGBFA} are forcefully applied to S_P and then the swaps of SS_{SMGBTA} are applied to S_P if they do not create any conflicts. Any conflict resulting from forceful swap application is handled by repair mechanism as described in section 2.2C. Similarly, the swaps from SS_{MM} are applied only if they are applicable. In Fig. 5, a solution resulting from application of a swap is represented by assigning that swap number above the solution. For example, if a swap say 1 is applied on solution S_P then it becomes S_P^1 and similarly applying swap 3 on S_P^1 makes it S_P^3. In the example shown in Fig. 5, swaps numbered 1, 3 and 5 are forcefully applied on initial solution S_P making it S_P^5 . Then swap numbered 2 gets applied on S_P^5 resulting in S_P^2 as there is no conflicts. Solution stays at S_P^2 because swaps numbered 4, 6 and 7 are not applied because of conflicts. Then rest of the swaps 8, 9, 10, 11 and 12 are applied because then do not give any conflicts. The best one among these solutions is then picked as particle's solution. Accordingly, S_{PB} and S_{GB} are updated for the next iteration.

3. Experimental Studies

This section investigates the effectiveness and performance of the PSOSS algorithm on UCSP-KUET instance for obtaining a viable timetable. The performance of the proposed algorithm has been compared with the performances of GA, HS, PSO and PSM for the same UCSP-KUET instance with the same experimental and parameter settings. This section also contains an experimental analysis for better understanding of the performance of the proposed method.

GA is a search and optimization algorithm mimicking natural selection and genetic mechanisms which includes crossover and mutation (Pillon et al., 2016; Kyriklidis and Dounias 2016; Padillo et al., 2018). The main notion of GA is the survival of the fittest (Rostami Neri, 2016). GA obtains a solution with the highest fitness after several iterations, which is considered the optimal solution (Adeli and Hung, 1995; Siddique and Adeli, 2013; Siddique, 2014). For ease of implementation, single point crossover is used in this paper with a crossover probability of 0.70**.** Mutation is performed by randomly changing the time slot of a course for a randomly selected instructor with mutation rate of 0.20. Elitism is also considered for implementation with an elite list of size 2.

HS algorithm is based on the notion of harmonic phenomena in musical performance. It is a population based algorithm inspired by improvisation process of musicians [Siddique and Adeli, 2017; 2015a; 2015b; 2015c]. There are two unique operators in HS: Harmony Memory Consideration Rate (HMCR) and Pitch Adjustment Rate (PAR) which are used to produce and modify a solution, respectively [Wang et al., 2015]. HS is implemented with a HMCR of 0.95 and a PAR of 0.1 respectively.

In PSM, solution having the best fitness becomes the producer, some solutions having the worst fitness become dispersed members and the rest of the solutions are considered as scroungers. In each iteration, producer tries to find a better solution, scroungers move toward the producer with the hope of finding better solution and dispersed members move randomly for finding new solutions [Akhand et al., 2015]. PSM is implemented with a swap selection probability of 0.3 in this paper.

Standard PSO is also investigated for comparison in this paper. Picked values of the tuning parameters alpha, beta and gamma for implementation of PSO and PSOSS are 0.3, 0.5 and 0.2 respectively. Also a force rate of 100% has been used for PSOSS.

The hard constraints of UCSP-KUET instance are:

- A student can only go to a single class in a timeslot.
- An instructor cannot conduct multiple classes in a timeslot.
- Courses cannot be assigned to break periods.
- Courses requiring multiple slots such as laboratory courses cannot include break periods.
- Courses can be assigned to allowed rooms only.

The soft constraints of UCSP-KUET instance are:

- Maintain preference of instructor as much as possible.
- Keep the amount of consecutive classes as few as possible for instructors.

The algorithm has been implemented in Visual C++ of Microsoft's Visual Studio 2013 on Windows 10 platform on Intel$^{(R)}$ Core$^{(TM)}$ i7-7700 CPU @ 3.60 GHz processor, and 8 GB RAM.

3.1 Experimental Environment

In the experimental environment, both instructors' flexibility, and students' flexibility are considered. The weekly time slots for instructors and their preferences are given in Fig. 2 and Fig. 4 respectively. The preferences varied from -1 to 5, where -1 means the lowest preference and 5 means the highest preference. Experiments with real-world input data taken from the department of Computer Science and Engineering (CSE) of Khulna University of Engineering & Technology (KUET) have been conducted. In KUET, there are 5 days for teaching in a week and each teaching day is divided into 9 teaching time slots of 45 minutes duration. The theory and laboratory classes are conducted by a single instructor and the duration of each laboratory session as well as an M.Sc. class is three consecutive time slots.

3.2 Input Data Preparation

Table 1 lists the used preference values of all the instructors. There are five batches of students in the CSE Department at KUET: four batches in the undergraduate level and one batch at the postgraduate (MSc) level. In total 38 courses are taught by 27 instructors. Odd-numbered courses represent theory courses and even-numbered courses represent laboratory courses. Table 2 shows which courses belong to which batch, the required credit hours for each course, number of weekly classes required for a course, time duration of a class, course type and the number of registered students of a course.

Table 3 shows the number of courses assigned to an instructor, the courses allocated to each instructor and the weekly workload of each instructor. Table 4 shows the class room id, room type, maximum seating capacity and allowable courses that can be taught in the class room. There are two types of class rooms: lecture room and laboratory room. As the laboratory rooms support a maximum of 30 students, a batch of 60 students needs to be divided into two subgroups of 30 students.

Table 1: Input preference values for instructors.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
I1	-1	-1	0	2	2	1	4	4	4	1	1	2	5	5	5	4	4	4	-1	1	1	5	5	5	4	4	2	2	2	2	5	5	5	4	4	4	4	4	3	2	1	0	-1	-1	-1
I2	1	1	1	2	2	1	4	4	4	2	2	2	5	5	5	4	4	4	0	1	1	5	5	5	4	4	2	2	2	2	5	5	5	4	4	4	4	4	3	2	1	0	-1	-1	-1
I3	-1	-1	0	2	2	1	4	4	4	1	1	2	5	5	5	4	4	4	-1	1	1	5	5	5	4	4	2	2	2	2	4	5	5	4	4	4	4	4	3	2	1	0	1	1	1
I4	1	1	1	2	2	3	4	4	4	2	2	2	5	5	5	4	4	4	0	1	1	5	5	5	4	4	2	2	2	2	4	5	5	4	4	4	4	4	3	2	1	0	-1	-1	-1
I5	-1	-1	0	-1	-1	3	1	1	1	-1	-1	0	1	1	2	2	2	1	-1	2	2	1	2	2	4	2	-1	2	2	2	4	2	2	2	2	2	1	2	3	5	1	0	-1	-1	-1
I6	-1	-1	0	2	2	3	2	2	2	0	1	2	2	1	2	2	2	1	-1	2	2	2	2	2	4	2	-1	2	2	2	4	2	2	2	2	2	1	2	4	5	1	0	-1	-1	-1
I7	1	1	1	2	2	3	2	2	2	0	2	2	2	2	2	2	2	1	0	2	2	2	2	2	4	2	1	2	1	2	4	2	2	2	2	2	1	2	4	5	1	0	1	1	1
I8	-1	-1	0	2	2	3	2	2	2	0	2	2	2	2	2	2	2	1	-1	2	2	2	2	2	4	2	1	2	1	2	4	2	2	2	2	2	1	2	4	5	1	0	-1	-1	-1
I9	1	1	1	2	2	3	2	2	2	0	2	2	2	2	2	2	2	2	0	2	2	2	2	2	4	2	-1	2	1	2	4	2	2	2	2	2	2	4	4	2	1	0	-1	-1	-1
I10	-1	-1	0	2	2	3	2	2	2	0	2	2	2	2	2	2	2	2	-1	2	2	2	2	2	4	2	-1	-1	1	2	4	2	2	2	2	2	2	4	4	2	1	0	-1	-1	-1
I11	-1	-1	0	2	2	3	2	2	2	0	2	2	2	2	2	2	2	2	-1	2	2	2	2	2	4	2	-1	-1	4	2	4	2	2	2	2	2	2	4	4	2	1	0	1	1	1
I12	1	1	1	2	2	3	2	4	2	0	2	2	2	2	2	2	2	2	0	2	2	2	2	2	4	2	-1	-1	4	2	4	2	2	2	2	2	2	4	4	2	1	0	-1	-1	-1
I13	-1	-1	0	2	2	3	2	4	2	0	2	2	2	4	2	2	2	2	-1	2	2	2	4	2	2	4	-1	-1	4	2	4	2	2	2	2	2	-1	2	4	2	1	0	-1	-1	-1
I14	1	1	1	2	2	3	2	4	2	0	4	2	1	4	2	2	2	2	1	2	2	1	4	2	2	4	1	-1	4	2	4	2	2	2	2	2	-1	2	3	2	1	0	-1	-1	-1
I15	-1	-1	0	2	2	2	1	4	2	0	4	2	2	4	2	2	2	2	-1	2	2	2	4	2	2	4	1	1	1	2	4	2	2	2	2	2	-1	2	3	2	1	0	1	1	1
I16	-1	-1	0	2	2	2	1	4	2	0	4	2	2	4	2	2	2	1	-1	2	2	2	4	2	2	4	1	1	1	2	4	2	2	2	2	2	-1	2	3	2	1	0	-1	-1	-1
I17	1	1	0	2	2	2	1	4	2	0	4	2	2	4	2	2	2	1	1	2	2	2	4	2	2	4	1	1	1	2	4	2	2	2	2	2	0	4	3	2	1	0	-1	-1	-1
I18	-1	-1	0	2	2	2	3	4	2	0	4	2	2	2	4	2	2	1	-1	2	2	2	4	2	2	4	1	2	1	2	4	2	2	2	2	2	0	4	3	2	1	0	-1	-1	-1
I19	1	1	1	2	2	2	3	4	2	0	4	2	2	2	4	2	2	1	1	2	2	2	4	2	2	4	1	2	4	2	4	2	2	2	2	2	0	4	3	2	1	0	1	1	1
I20	-1	-1	0	2	2	2	3	2	2	0	4	2	2	2	4	2	2	2	-1	2	2	2	4	2	2	4	1	2	4	2	4	2	2	2	2	2	0	4	2	2	1	0	-1	-1	-1
I21	-1	-1	0	2	2	2	3	2	2	0	4	2	4	2	4	2	2	2	-1	2	2	4	4	2	2	2	1	2	4	2	4	2	2	2	2	2	-1	2	2	2	1	0	-1	-1	-1
I22	1	1	1	2	2	3	2	2	2	0	2	2	4	2	4	2	2	2	1	2	2	4	4	2	2	2	1	2	1	2	4	2	2	2	2	2	-1	2	2	2	1	0	-1	-1	-1
I23	-1	-1	0	2	2	3	2	2	2	-1	0	2	4	2	4	2	2	2	-1	2	2	4	4	2	2	2	1	2	1	2	4	2	2	2	2	2	-1	2	2	2	1	0	1	1	1
I24	1	1	1	2	2	3	2	2	2	-1	0	2	4	2	4	2	2	2	0	2	2	4	4	2	2	2	1	2	1	2	4	2	2	2	2	2	-1	2	2	2	1	0	-1	-1	-1
I25	-1	-1	0	2	2	3	2	2	2	-1	0	2	4	2	4	2	2	2	-1	2	2	4	2	2	2	2	1	2	4	2	4	2	2	2	2	2	-1	2	2	2	1	0	-1	-1	-1
I26	1	1	1	2	2	3	2	2	2	-1	0	2	2	2	4	2	2	2	0	2	2	2	2	2	2	2	1	2	4	2	4	2	2	2	2	2	-1	2	2	2	1	0	1	1	1
I27	-1	-1	0	2	2	3	2	2	2	-1	0	2	2	2	4	2	2	2	-1	2	2	2	2	2	2	2	1	2	4	2	4	2	2	2	2	2	1	2	3	2	1	0	-1	-1	-1

Table 2: Batch and course information.

Batch Name	Course No.	Credit	Nos / Week	Hrs / Class	Type of Course	No of Students
B1 / 1st Year	CSE 1201	3.0	3	1	Theory	60
	CSE 1202	1.5	2	3	Laboratory	30
	CSE 1203	3.0	3	1	Theory	60
	CSE 1204	1.5	2	3	Laboratory	30
	EEE 1217	3.0	3	1	Theory	60
	EEE 1218	0.75	1	3	Laboratory	30
	CHEM 1207	3.0	3	1	Theory	60
	CHEM 1208	0.75	1	3	Laboratory	30
	MATH 1207	3.0	3	1	Theory	60
	ME 1270	0.75	1	3	Laboratory	30
B2 / 2nd Year	CSE 2200	1.5	2	3	Laboratory	30
	CSE 2201	3.0	3	1	Theory	60
	CSE 2202	1.5	2	3	Laboratory	30
	CSE 2207	3.0	3	1	Theory	60
	CSE 2208	0.75	1	3	Laboratory	30
	CSE 2213	3.0	3	1	Theory	60
	EEE 2217	3.0	3	1	Theory	60
	EEE 2218	1.5	2	3	Laboratory	30
	MATH 2207	3.0	3	1	Theory	60
B3 / 3rd Year	CSE 3200	1.5	2	3	Laboratory	30
	CSE 3201	3.0	3	1	Theory	60
	CSE 3202	1.5	2	3	Laboratory	30
	CSE 3203	3.0	3	1	Theory	60
	CSE 3204	0.75	1	3	Laboratory	30
	CSE 3207	3.0	3	1	Theory	60
	CSE 3211	3.0	3	1	Theory	60
	CSE 3212	0.75	1	3	Laboratory	30
	ECE 3215	3.0	3	1	Theory	30
B4/ 4th Year	CSE 4207	3.0	3	1	Theory	60
	CSE 4208	0.75	1	3	Laboratory	30
	CSE 4211	3.0	3	1	Theory	60
	CSE 4212	0.75	1	3	Laboratory	30
	CSE 4239	3.0	3	1	Theory	60
	IEM 4227	3.0	3	1	Theory	60
	HUM 4207	3.0	3	1	Theory	60
B5/ M.Sc.	CSE 6225	3.0	1	3	Theory	10
	CSE 6465	3.0	1	3	Theory	10
	CSE 6471	3.0	1	3	Theory	10

3.3 Experimental Results and Analysis

The size of the initial population of all the algorithms investigated in this study (i.e. GA, PSO, PSM, HS and PSOSS) is kept equal. The population size is one of the important parameters. Its impact is evaluated as the computation cost increases with growing population size. A second parameter is the maximum number of iterations for convergence to an optimal solution. On the other hand, individual algorithms have their own parameters. In this section, first the effect of the number of iterations and population size on the algorithms is investigated. Next, algorithms are compared based on instructors' satisfaction followed by the sample timetables generated for an instructor by the algorithms.

Table 3: Course information for each instructor.

Instructor ID	No. of Courses	Course Code	Weekly Workload (Hrs/Week)
I1	4	CSE 1203, CSE 1204, CSE 2201, CSE 2202	18
I2	4	CSE 3211, CSE 3212, CSE 4239, CSE 6225	12
I3	2	CSE 3201, CSE 3202	9
I4	1	CSE 3200	6
I5	3	CSE 4211, CSE 4212, CSE 6471	9
I6	2	CSE 4207, CSE 6465	6
I7	1	CSE 2207	3
I8	1	CSE 1201	3
I9	1	CSE 2208	3
I10	2	CSE 2200, CSE3207	9
I11	1	CSE 3203	3
I12	1	CSE 1202	6
I13	1	CSE 4208	3
I14	1	CSE 2213	3
I15	1	CSE 3204	6
I16	1	EEE 1217	3
I17	1	EEE 1218	3
I18	1	EEE 2217	3
I19	1	EEE 2218	6
I20	1	MATH 1207	3
I21	1	MATH 2207	3
I22	1	ECE 3215	3
I23	1	ME 1270	3
I24	1	IEM 4227	3
I25	1	CHEM 1207	3
I26	1	CHEM 1208	3
I27	1	HUM 4207	3

For better understanding of the effect of varying population size on the algorithms, fitness values are calculated by varying the population size from 5 to 300 while keeping the iteration number fixed at 100. The results of varying population sizes are shown in Fig. 6. It is clear from the figure that PSOSS outperforms all the other algorithms for varying population sizes because of the use of the force in PSOSS that assures all the particles

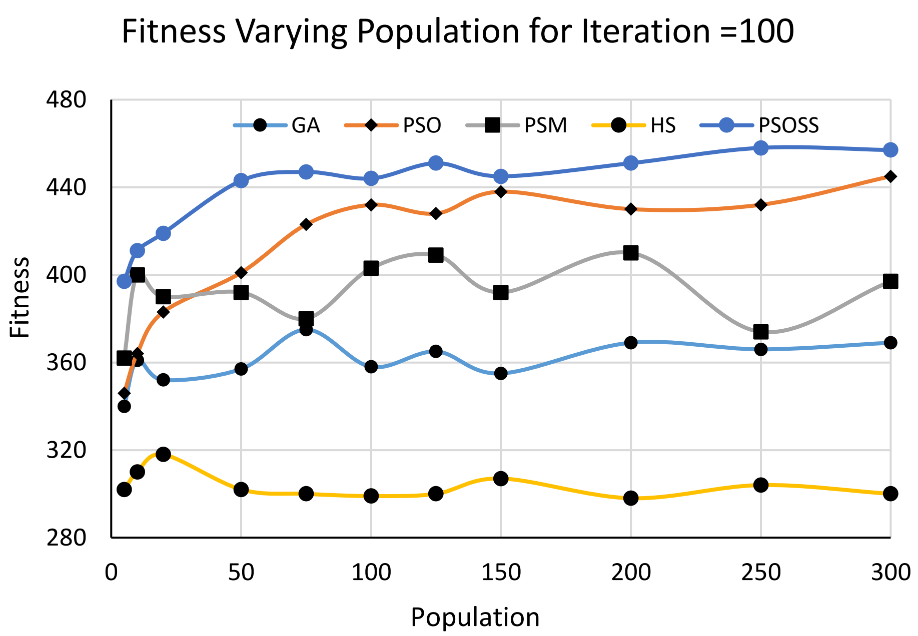


Figure 6: Variation effect of Population Size.

Table 4: Information for classrooms and laboratories.

Room ID	Room Type	Room capacity	Allowable Courses
CR1	Lecture	60	Any Theoretical Subjects
CR2	Lecture	60	
CR3	Lecture	60	
CR4	Lecture	60	
CR5	Lecture	60	
LB1	Laboratory	30	CSE 2200, CSE 3202, CSE 4208, CSE 4212
LB2	Laboratory	30	CSE2200, CSE4212, CSE3212, CSE2208, CSE3202
LB3	Laboratory	30	CSE 1202, CSE 2202
LB4	Laboratory	30	CSE1204
LB5	Laboratory	30	CSE3204
LB6	Laboratory	30	ME1270
LB7	Laboratory	30	EEE1218, EEE2218
LB8	Laboratory	30	CHEM1208

move a little towards the global best and the personal best solutions as described in section 2.2C. Among the other algorithms PSM works well for low population size because the producer improves its fitness in each iteration.

The effect of varying iterations on the algorithms is shown in Fig. 7. The maximum number of iterations is varied from 5 to 300 while keeping the population size fixed at 50. It is clear from the figure that PSOSS performs better than other algorithms for all max number of iterations. Standard PSO is the second best performer.

One of the objectives of optimizing UCSP is to satisfy the demands of instructors' having high workload as much as possible. Therefore, algorithms are compared based on the percentage of instructors' satisfaction. The percentage of satisfaction for an instructor is computed using the formula:

$$Satisfaction_I = \frac{F_{IS}}{MF_{IS}} * 100 \tag{13}$$

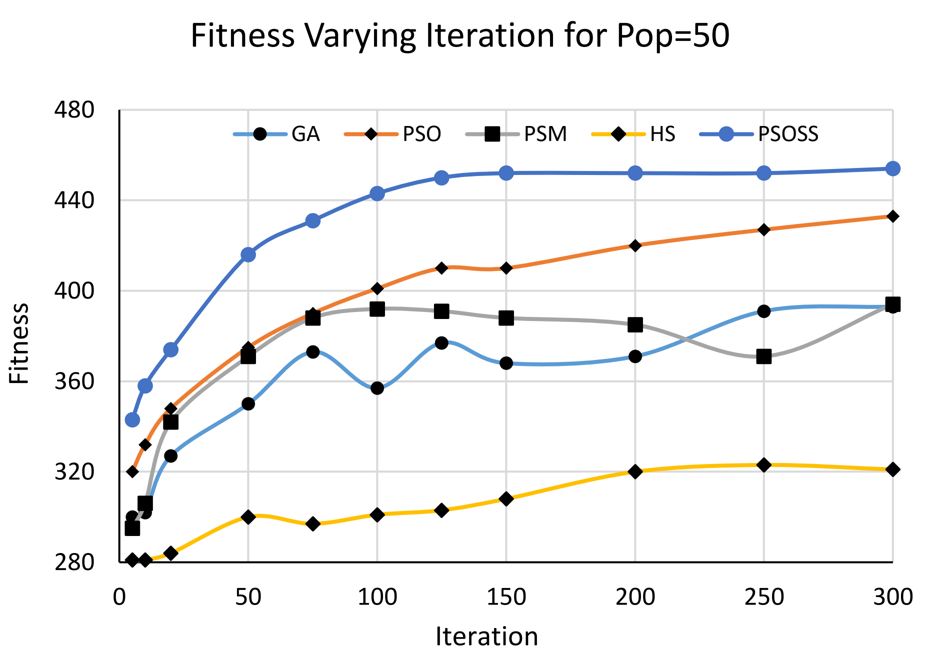


Figure 7: Variation effect of Iterations.

Table 5: Instructors' satisfaction values achieved by implemented algorithms

Instructor	Course Load	Achieved satisfaction values (in %)				
		GA	**PSO**	**HS**	**PSM**	**PSOSS**
I1	18	71.6	64.2	76.54	83.95	95.06
I2	12	87.72	71.93	56.14	82.46	89.47
I3	9	70.45	75	65.91	72.73	93.18
I4	6	80	53.33	80	60	80
I5	9	74.07	70.37	33.33	85.19	85.19
I6	6	54.55	54.55	59.09	68.18	81.82
I7	3	46.15	46.15	46.15	69.23	69.23
I8	3	46.15	30.77	46.15	76.92	100
I9	3	41.67	25	41.67	41.67	66.67
I10	9	77.78	77.78	51.85	70.37	81.48
I11	3	25	66.67	66.67	100	75
I12	6	62.5	66.67	37.5	75	75
I13	3	83.33	50	83.33	50	66.67
I14	3	75	50	66.67	58.33	66.67
I15	6	50	29.17	45.83	62.5	62.5
I16	3	41.67	25	66.67	83.33	66.67
I17	3	58.33	25	66.67	50	66.67
I18	3	66.67	50	50	66.67	50
I19	6	62.5	79.17	37.5	70.83	75
I20	3	83.33	58.33	41.67	66.67	66.67
I21	3	100	66.67	41.67	66.67	100
I22	3	41.67	75	50	83.33	75
I23	3	83.33	50	58.33	66.67	83.33
I24	3	66.67	66.67	75	66.67	83.33
I25	3	50	66.67	33.33	66.67	75
I26	3	41.67	50	50	50	66.67
I27	3	58.33	58.33	25	75	100
Avg. Satisfaction		**67.14**	**60.07**	**56.89**	**72.44**	**81.63**

where, MF_{IS} is the maximum possible fitness of an instructor's solution, F_{IS} is the achieved fitness of an instructor's solution as described by Eq. (12).

Table 5 shows the achieved satisfaction value (in %) for all the instructors for all the algorithms. It also includes weekly work load for each instructor. It is seen from the table that the instructor I1 has the highest workload among all others and PSOSS achieved the highest satisfaction value (in %) for instructor I1 which is also the case for the second highest work load for instructor I2. Though for some instructors with some less work load, the satisfaction value achieved by PSOSS is not higher than other algorithms but the average satisfaction value achieved by PSOSS is much higher than other algorithms. This shows a significant performance indicator.

Table 6 shows the timetable for instructor I1 generated by the algorithms GA, PSO, HS, PSM and PSOSS. In the generated timetable G0 denotes a batch and G1 and G2 represent subgroups. From the generated timetable by PSOSS shown in Table 6 (e), it can be seen that I1 has a Laboratory class CSE 1204 in timeslots 7, 8 and 9 on Sunday which is desirable because I1 has maximum preference of 4 in these periods for Sunday as stated in Table 1. Similarly, timetable for other days also adheres to instructor I1's preference in most of the cases. Overall, the timetable generated by PSOSS for instructor I1 is satisfactory compared to timetables generated by other algorithms. It is to be noted that KUET has working days from Sunday to Thursday.

Table 6: Sample Timetable for Instructor 1 generated by GA, PSO, HS, PSM and PSOSS.

(a) GA

<table>
<tr><th rowspan="2">Day</th><th colspan="10">Time Slot</th></tr>
<tr><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th></th><th>7</th><th>8</th><th>9</th></tr>
<tr><td>Sun</td><td></td><td></td><td></td><td colspan="3">CSE1204 (LB4|B1|G1|I1)</td><td rowspan="5">LUNCH BREAK</td><td></td><td>CSE1203 (CR1|B1|G0|I1)</td><td></td></tr>
<tr><td>Mon</td><td></td><td></td><td></td><td></td><td></td><td></td><td>CSE2201 (CR4|B2|G0|I1)</td><td></td><td>CSE2201 (CR5|B2|G0|I1)</td></tr>
<tr><td>Tue</td><td></td><td></td><td>CSE1203 (CR5|B1|G0|I1)</td><td></td><td></td><td></td><td colspan="3">CSE2202 (LB3|B2|G0|I1)</td></tr>
<tr><td>Wed</td><td></td><td></td><td></td><td colspan="3">CSE1204 (LB4|B1|G2|I1)</td><td></td><td>CSE1203 (CR2|B1|G0|I1)</td><td></td></tr>
<tr><td>Thu</td><td colspan="3">CSE2202 (LB3|B2|G1|I1)</td><td></td><td></td><td>CSE2201 (CR1|B2|G0|I1)</td><td></td><td></td><td></td></tr>
</table>

(b) PSO

<table>
<tr><th rowspan="2">Day</th><th colspan="10">Time Slot</th></tr>
<tr><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th></th><th>7</th><th>8</th><th>9</th></tr>
<tr><td>Sun</td><td colspan="3">CSE2202 (LB3|B2|G0|I1)</td><td></td><td></td><td></td><td rowspan="5">LUNCH BREAK</td><td></td><td></td><td></td></tr>
<tr><td>Mon</td><td></td><td></td><td></td><td></td><td></td><td>CSE1203 (CR5|B1|G0|I1)</td><td></td><td></td><td>CSE2201 (CR3|B2|G0|I1)</td></tr>
<tr><td>Tue</td><td></td><td></td><td></td><td>CSE2201 (CR1|B2|G0|I1)</td><td>CSE1203 (CR4|B1|G0|I1)</td><td></td><td colspan="3">CSE2202 (LB3|B2|G1|I1)</td></tr>
<tr><td>Wed</td><td></td><td colspan="3">CSE1204 (LB4|B1|G1|I1)</td><td></td><td></td><td colspan="3">CSE1204 (LB4|B1|G2|I1)</td></tr>
<tr><td>Thu</td><td></td><td>CSE2201 (CR4|B2|G0|I1)</td><td></td><td></td><td></td><td>CSE1203 (CR4|B1|G0|I1)</td><td></td><td></td><td></td></tr>
</table>

(c) HS

<table>
<tr><th rowspan="2">Day</th><th colspan="10">Time Slot</th></tr>
<tr><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th></th><th>7</th><th>8</th><th>9</th></tr>
<tr><td>Sun</td><td colspan="3"></td><td></td><td>CSE2201 (CR4|B2|G0|I1)</td><td></td><td rowspan="5">LUNCH BREAK</td><td colspan="3">CSE1204 (LB4|B1|G1|I1)</td></tr>
<tr><td>Mon</td><td></td><td></td><td></td><td></td><td>CSE2201 (CR4|B2|G0|I1)</td><td></td><td></td><td></td><td>CSE2201 (CR3|B2|G0|I1)</td></tr>
<tr><td>Tue</td><td></td><td></td><td></td><td></td><td></td><td></td><td colspan="3">CSE2202 (LB3|B2|G0|I1)</td></tr>
<tr><td>Wed</td><td>CSE1203 (CR4|B1|G0|I1)</td><td></td><td></td><td colspan="3">CSE2202 (LB3|B2|G1|I1)</td><td colspan="3">CSE1204 (LB4|B1|G2|I1)</td></tr>
<tr><td>Thu</td><td></td><td></td><td></td><td></td><td>CSE1203 (CR3|B1|G0|I1)</td><td></td><td></td><td>CSE1203 (CR1|B1|G0|I1)</td><td></td></tr>
</table>

(d) PSM

<table>
<tr><th rowspan="2">Day</th><th colspan="10">Time Slot</th></tr>
<tr><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th></th><th>7</th><th>8</th><th>9</th></tr>
<tr><td>Sun</td><td></td><td></td><td></td><td></td><td></td><td></td><td rowspan="5">LUNCH BREAK</td><td>CSE1203 (CR2|B1|G0|I1)</td><td></td><td>CSE1203 (CR2|B1|G0|I1)</td></tr>
<tr><td>Mon</td><td></td><td colspan="3">CSE1204 (LB4|B1|G2|I1)</td><td></td><td>CSE2201 (CR3|B2|G0|I1)</td><td colspan="3">CSE2202 (LB3|B2|G1|I1)</td></tr>
<tr><td>Tue</td><td></td><td></td><td></td><td></td><td></td><td></td><td colspan="3">CSE1204 (LB4|B1|G1|I1)</td></tr>
<tr><td>Wed</td><td></td><td></td><td></td><td colspan="3">CSE2202 (LB3|B2|G0|I1)</td><td>CSE2201 (CR4|B2|G0|I1)</td><td></td><td>CSE1203 (CR5|B1|G0|I1)</td></tr>
<tr><td>Thu</td><td></td><td>CSE2201 (CR5|B2|G0|I1)</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

(e) PSOSS

<table>
<tr><th rowspan="2">Day</th><th colspan="10">Time Slot</th></tr>
<tr><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th></th><th>7</th><th>8</th><th>9</th></tr>
<tr><td>Sun</td><td></td><td></td><td></td><td></td><td></td><td></td><td rowspan="5">LUNCH BREAK</td><td colspan="3">CSE 1204 (LB4|B1|G1|I1)</td></tr>
<tr><td>Mon</td><td></td><td></td><td></td><td colspan="3">CSE2202 (LB3|B2|G2|I1)</td><td></td><td>CSE1203 (CR3|B1|G0|I1)</td><td></td></tr>
<tr><td>Tue</td><td></td><td></td><td></td><td>CSE2201 (CR5|B1|G0|I1)</td><td></td><td>CSE2201 (CR3|B2|G0|I1)</td><td></td><td></td><td>CSE1203 (CR5|B1|G0|I1)</td></tr>
<tr><td>Wed</td><td></td><td></td><td></td><td colspan="3">CSE2202 (LB3|B2|G1|I1)</td><td>CSE2201 (CR1|B2|G0|I1)</td><td></td><td>CSE1203 (CR5|B1|G0|I1)</td></tr>
<tr><td>Thu</td><td colspan="3">CSE 1204 (LB4|B2|G2|I1)</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

4. Conclusions

In this paper, a PSO based innovative technique PSOSS has been proposed to solve UCSP. The proposed method differs from existing methods, including many variants of PSO-based approaches, where UCSP is transformed into an equivalent numerical domain. The proposed PSOSS approach uses a swap sequence based discrete PSO with a number of modifications. The velocity swap sequence is managed in two different parts: sequences for global best; and sequences combining personal best and previous velocity. A portion of global sequence portion is considered to be applied forcefully with repair mechanism to change other dependent schedule. After applying SOs one by one, the best intermediate solution is considered as the final solution based on selective search. The results obtained by our proposed method show significant improvement in respect of quality of solutions compared to other traditional methods.

References

[Chiarandini et al., 2006] M. Chiarandini, M. Birattari, K. Socha and O. Rossi-Doria, Olivia (2006), "An effective hybrid algorithm for university course timetabling", *Journal of Scheduling*, vol. 9, no. 5, pp. 403- 432.

[Mencia et al., 2016] R. Mencıa, M.R. Sierra, C. Mencıa, and R. Varela (2016), "Genetic algorithms for the scheduling problem with arbitrary precedence relations and skilled operators," *Integrated Computer-Aided Engineering*, vol. 23, no. 3, pp. 269-285.

[Tang et al., 2018] Tang, Y., Liu, R., Wang, F., Sun, Q., and Kandil, A.A. (2018), "Scheduling Optimization of Linear Schedule with Constraint Programming," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 2, pp. 124-151.

[Yue et al., 2017] Yue, Y., Han, J., Wang, S., and Liu, X. (2017), "Integrated Train Timetabling and Rolling Stock Scheduling Model Based on Time-Dependent Demand for Urban Rail Transit," *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 10, pp. 856-873.

[Pongcharoen et al., 2008] P. Pongcharoen, W. Promtet, P. Yenradee and C. Hicks (2008), "Stochastic Optimisation Timetabling Tool for university course scheduling", *International Journal of Production Economics*, vol. 112, no. 2, pp.903-918.

[Mushi, 2006] A. R. Mushi (2006), "Tabu Search Heuristic for University Course Timetabling Problem," *African Journal of Science and Technology (AJST), Science and Engineering Series*, vol. 7, no. 1, pp. 34 – 40.

[Feizi-Derakhshi et al., 2012] M. Feizi-Derakhshi, H. Babaei and J. Heidarzadeh (2012), "A Survey of Approaches for University Course Timetabling Problem," *Proceedings of 8th International Symposium on Intelligent and Manufacturing Systems* (*IMS*) , Adrasan, Antalya, Turkey, pp. 307-321.

[Shiau, 2011] D. Shiau (2011), "A hybrid particle swarm optimization for a university course scheduling problem with flexible preferences," *Expert Systems with Applications*, vol. 38, pp. 235–248.

[Azimi, 2005] Z. N. Azimi (2005), "Hybrid heuristics for Examination Timetabling problem", *Applied Mathematics and Computation*, vol. 163, pp. 705–733.

[Yang et al., 2016] Z. Yang, M. Emmerich, T. Baeck, and J. Kok (2016), "Multiobjective Inventory Routing with Uncertain Demand Using Population-based Metaheuristics," *Integrated Computer-Aided Engineering*, vol. 23, no. 3, pp. 205-220.

[Wang, 2003] Y. Wang (2003), "Using genetic algorithm methods to solve course scheduling problems," *Expert Systems with Applications*, vol. 25, pp. 39-50.

[Martinez-alvarez et al., 2016]A. Martinez-alvarez, R. Crespo Cano, A. Diaz-Tahoces, S. Cuenca-Asensi, J.M. Ferrandez Vicente, and E. Fernandez Jover (2016), " Automatic Tuning of a Retina Model for a Cortico Visual Neuroprosthesis Using a Multi-objective Optimization Genetic Algorithm," *International Journal of Neural Systems*, vol. 26, no. 7.

[Li et al., 2017] Li, W., Pu, H., Schonfeld, P., Yang, J., Zhang, H., Wang, L., and Xiong, J. (2017), "Mountain railway alignment optimization with bidirectional distance transform and genetic algorithm," *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 8, pp. 691-709.

[Zhao et al., 2018]Zhao, W., Guo, S., Zhou, Y., and Zhang, J. (2018), "A Quantum-Inspired Genetic Algorithm-Based Optimization Method for Mobile Impact Test Data Integration," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 5, pp. 411-422.

[Boland et al., 2008] N. Boland, B. D. Hughes, L. T .G. Merlot, P. J. Stuckey (2008), "New integer linear programming approaches for course timetabling", *Computers & Operations Research*, vol. 35, no. 7, pp. 2209-2233.

[Abramson et al, 1991] Abramson, D. (1991), “Constructing School Timetables Using Simulated Annealing: Sequential and Parallel Algorithms.” *Management Science*, vol. 37, no. 1, pp. 98–113.

[Prieto et al., 2016] A. Prieto, F. Bellas, P. Trueba, and R.J. Duro (2016), “Real-Time Optimization of Collective Non-separable Problems through Distributed Embodied Evolution,” *Integrated Computer-Aided Engineering*, vol. 3, no. 3, pp. 237-253.

[Abdullah et al., 2007] S. Abdullah, E. K. Burke and B. McCollum (2007), "A hybrid evolutionary approach to the university course timetabling problem," *IEEE Congress on Evolutionary Computation*, Singapore, pp. 1764-1768.

[Yang and Jat, 2011] Shengxiang Yang, and Sadaf Naseem Jat (2011), "Genetic Algorithms With Guided and Local Search Strategies for University Course Timetabling," *IEEE Transaction on systems, Man, and Cybernetics-part C: applications and reviews*, vol. 41, no. 1, pp.93-106.

[Obit et al., 2012] Joe Henry Obit, Djamila Ouelhadj, Dario Landa-Silva and Rayner Alfred (2012), “An Evolutionary Non-Linear Great Deluge Approach for Solving Course Timetabling Problems,” *IJCSI International Journal of Computer Science Issues*, vol. 9, no 2.

[Turabieh et al., 2009] Turabieh H., Abdullah S., McCollum B. (2009), “Electromagnetism-like Mechanism with Force Decay Rate Great Deluge for the Course Timetabling Problem,” *In: Wen P., Li Y., Polkowski L., Yao Y., Tsumoto S., Wang G. (eds) Rough Sets and Knowledge Technology. RSKT 2009. Lecture Notes in Computer Science*, vol 5589, Springer, Berlin, Heidelberg.

[Al-Betar et al., 2012] Al-Betar, Mohammed Azmi and Ahamad Tajudin Abdul Khader (2012), “A harmony search algorithm for university course timetabling.” *Annals of Operations Research*, vol. 194, pp. 3-31.

[Al-Betar et al., 2009] Al-Betar, Mohammed & Khader, Ahamad Tajudin (2009), “A hybrid harmony search for university course timetabling,” *Proceedings of the 4nd Multidisciplinary Conference on Scheduling: Theory and Applications (MISTA)*.

[Ayob and Jaradat, 2009] M. Ayob and G. Jaradat (2009), "Hybrid Ant Colony systems for course timetabling problems," *2nd Conference on Data Mining and Optimization*, Kajand, pp. 120-126.

[Li and Zhang, 2013] H. Li and H. Zhang (2013), “Ant colony optimization-based multi-mode scheduling under renewable and nonrenewable resource constraints,” *Automation in Construction*, vol. 35, pp. 431-438.

[Sabar et al., 2012] N. R. Sabar, M. Ayob, G. Kendall, R. Qu (2012), “A honey-bee mating optimization algorithm for educational timetabling problems,” *European Journal of Operational Research*, vol. 216, pp. 533–543.

[Alexandridis et al., 2017] A. Alexandridis, E. Paizis, E. Chondrodima and E. Aliaj, (2017) “A particle swarm optimization approach in printed circuit board thermal design,” *Integrated Computer-Aided Engineering*, vol. 24, no. 2, pp. 143-155.

[Tassopoulos and Beligiannis, 2012] I. X. Tassopoulos and G. N. Beligiannis (2012), “A hybrid particle swarm optimization based algorithm for high school,” *Applied Soft Computing*, vol. 12, pp. 3472–3489.

[Chen and Shih, 2013] R. Chen and H. Shih, “Solving University Course Timetabling Problems Using Constriction Particle Swarm Optimization with Local Search (2013),” *Algorithms*, vol. 6, pp. 227-244.

[Osman, 2015] A. Osman, “Particle Swarm Optimization Based Najran University Course Timetable Scheduling (2015),” *NNGT Int. J. on Artificial Intelligence*, vol. 2.

[Kennedy and Eberhart, 1995] J. Kennedy and R. C. Eberhart, “Particle swarm optimization (1995),” *Proceedings of IEEE international conference on neural networks*, pp. 1942–1948.

[Montero et al., 2011] E. Montero, M. C. Riff and L. Altamirano (2011), "A PSO algorithm to solve a Real Course+Exam Timetabling Problem," *International conference on swarm intelligence*, Cergy, France, pp. 14-15.

[Pillon et al., 2016] P. E. Pillon, E. C. Pedrino, V. O. Roda, M.C. Nicoletti (2016), “A hardware oriented ad-hoc computer-based method for binary structuring element decomposition based on genetic algorithm,” Integrated Computer-Aided Engineering, vol. 23, no.4, pp. 369-383.

[Kyriklidis and Dounias 2016] C. Kyriklidis and G. Dounias (2016), “Evolutionary Computation for Resource Leveling Optimization in Project Management,” *Integrated Computer-Aided Engineering*, vol. 23, no.2, pp. 173-184.

[Padillo et al., 2018] Padillo, F., Luna, J.M., Herrera, F., and Ventura, S (2018), “Mining Association Rules on Big Data through MapReduce Genetic Programming,” *Integrated Computer-Aided Engineering*, vol. 25, no.1, pp. 31-48.

[Rostami Neri, 2016] S. Rostami and F. Neri (2016), “Covariance Matrix Adaptation Pareto Archived Evolution Strategy with Hypervolume-sorted Adaptive Grid Algorithm (2016),” *Integrated Computer-Aided Engineering*, vol. 23, no.4, pp. 313-329.

[Adeli and Hung, 1995] Adeli, H. and Hung, S.L. (1995), *Machine Learning - Neural Networks, Genetic Algorithms, and Fuzzy Systems*, John Wiley and Sons, New York.

[Siddique, N. and Adeli, H., 2013] Siddique, N. and Adeli, H. (2013) *Computational Intelligence: Synergies of Fuzzy Logic, Neural Networks and Evolutionary Computing*, John Wiley and Sons, New York.

[Siddique, N., 2014] Siddique, N. (2014), "Intelligent Control: A Hybrid Approach based on Fuzzy Logic, Neural Networks and Genetic Algorithms", *Studies in Computational Intelligence*, vol. 517, Springer Verlag.

[Siddique, N. and Adeli, H., 2017] Siddique, N. and Adeli, H. (2017) "Nature Inspired Computing: Physics and Chemistry based Algorithms*", CRC Press/Taylor & Francis Group*.

[Siddique, N. and Adeli, H., 2015a] Siddique, N. and Adeli, H. (2015), "Harmony Search Algorithm and Its Variants," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 29, no.8.

[Siddique, N. and Adeli, H., 2015b] Siddique, N. and Adeli, H. (2015), "Hybrid Harmony Search Algorithms," *International Journal on Artificial Intelligence Tools*, vol. 24, no.6.

[Siddique, N. and Adeli, H., 2015c] Siddique, N. and Adeli, H. (2015), "Applications of Harmony Search Algorithms in Engineering," *International Journal on Artificial Intelligence Tools*, vol. 24, no.6.

[Wang et al., 2015] Wang, Xiaolei & Gao, Xiao-Zhi & Zenger, Kai. (2015). An Introduction to Harmony Search Optimization Method.

[Akhand et al., 2015] M. A. H. Akhand, Pintu Chnadra Shill, Md. Forhad Hossain, A.B.M Junaed (2015), "Producer-Scrounger Method to Solve Traveling Salesman Problem," *I.J. Intelligent Systems and Applications*, vol. 3, pp. 29-36.