

Incorporating Of Constraint-Based Reasoning Into Particle Swarm Optimization For University Timetabling Problem

Ho Sheau Fen @ Irene¹, Safaai Deris¹ and Siti Zaiton Mohd Hashim¹

¹ Faculty of Computer Science and Information System, D07 Building, Level 4, Software Engineering Department, Universiti Teknologi Malaysia, 81310, Johor, Malaysia

Abstract

Timetabling problems are often difficult and time-consuming task. It involves a set of timeslots, classrooms, subjects, students and lecturers. The complexity problem is the constraints that exist within the resources. Thus, a technique that can handle constraints is needed to optimize the problem. Various approaches have been reported in the literature on solving university timetabling problem. This paper focuses on developing a hybrid algorithm consisting of a particle swarm optimization and constraint-based reasoning in solving university timetabling problem in generating a feasible and near-optimal solution. The proposed algorithm is tested using real data from the Faculty of Computer Science and Information System, Universiti Teknologi Malaysia. The result is compared against standard particle swarm optimization and hybrid particle swarm optimization-local search. It shows that the proposed method has outperformed others.

Keywords: Timetabling; Particle swarm optimization; constraint-based reasoning; University timetabling.

1. Introduction

In a university, every semester each department of the university has to produce a new timetable for subjects to be taught. The timetabling problem consists of allocating these subjects with specific number of lecturers and classrooms in certain timeslots, in a week.

Recently, a lot of attention has been paid in automating the construction of timetable planning using meta-heuristic approaches. The meta-heuristic approaches that have been tried, including graph coloring [2, 6], simulated annealing [1], tabu search [3], genetic algorithms [18] and constraint-based reasoning [20]. Most of these approaches generate feasible but not optimal solutions.

^o Corresponding Author: Ho Sheau Fen @ Irene

Email: ireneluv@hotmail.com

Telephone: +6019-8331338

Fax: +6089-673979

© 2009-2012 All rights reserved. ISSR Journals

Some studies shows that by incorporating constraint-based reasoning a near-optimal solution can be obtained. Author [14] incorporates of local search into CBR in generating university timetable shows promising results. Authors [9, 17, 21] proposed a hybrid genetic algorithms (GA)-constraints-based reasoning (CBR) in solving university timetable problem that generated feasible and near-optimal solution. The drawback of this hybrid GA-CBR approach is it consumes more computational time to generate a timetable [13]. Particle swarm optimization (PSO), a relatively new stochastic optimization method shares a lot of GA similarities [8]. According to some solution generated using PSO for university timetable; it is inadequate to control the constraints violation [4, 5, 10, 11]. Since hybrid GA-CBR successfully provides feasible and near-optimal solution, the curiosity of hybrid PSO-CBR for the university timetable is investigated. This paper dealt with solving university timetable for Faculty of Computer Science and Information System, Universiti Teknologi Malaysia, Malaysia. The objective of the paper is to find a feasible yet a near-optimal timetable that satisfies all constraints simultaneously. The paper is organized as follows. Section 2 describes the university course timetabling characteristics and desire constraints. Section 3 describes the hybrid proposed algorithm into solving UCTP. Section 4 reports and discusses the experimental and results. Finally conclusion and future work are given in Section 5.

2. University course timetabling

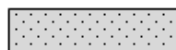
The university course timetabling problem (UCTP) consists of scheduling a set of subjects within a given number of rooms and timeslots. UCTP models must accommodate the characteristics and regulations of specific education systems. Therefore, the problem under consideration will vary from university to the other.

2.1. Characteristics of the university timetable problem

The data used in this investigation is obtained from the Faculty of Computer Science and Information System, University of Technology Malaysia in Malaysia for semester I 2008/2009. There are up to 16 different groups of students ranging from year 1 to year 3. The groups are categorized by their major (i.e. bioinformatics, software engineering, etc). The timetable consists of 43 consecutive timeslots and 11 timeslots per day (five days a week). Each timeslot consists of a 50 minutes lecture, with 10 minutes break between subjects starting at 8:00 AM till 7:00 PM for 5 days. Another 12 extra timeslots are reserved for non-academic activities and lunch hours.

A timetabling is characterized by events, such as lessons, lecturers, student groups and rooms. Each event has its own unique ID given. Each room and timeslot has its own preference depends on the subject's facilities and student's capacity. Figure 1 illustrated an example of university weekly timetable.

Time	Monday	Tuesday	Wednesday	Thursday	Friday
8:00 – 8:50	1	2	3	4	5
9:00 – 9:50	6	7	8	9	10
10:00 – 10:50	11	12	13	14	15
11:00 – 11:50	16	17	18	19	20
12:00 – 12:50	21	22	23	24	
13:00 – 13:50					
14:00 – 14:50	25	26		27	
15:00 – 15:50	28	29		30	31
16:00 – 16:50	32	33		34	35
17:00 – 17:50	36	37		38	39
18:00 – 18:50	40	41		42	43



Reserved timeslots for non-academic activities and lunch hours

Figure 1. An example of university weekly timetable

2.2. Modelling university course timetabling problem as constraint-satisfaction problem

The constraint-satisfaction problems (CSPs) are decision problems defined as set of objects whose state must satisfy a number of constraints [19]. CSP has been classified into two main groups:

- *Complete method* aims at exploring the whole search tree in order to find all the solutions or to detect none valid CSP. Backtracking search is one of the techniques under this group.
- *Incomplete method* mainly relies in the use of heuristics to provide a more efficient exploration of interesting areas of the search space in order to find some solutions. Local search (LS) technique is categorized under this group.

In order to solve the university timetabling problem using CBR, the problem has to be modelled as a CSP. The CSP consists of a set of variables $X = \{x_1, \dots, x_n\}$ which has an associated domain D_i of possible values. There is also a set of constraints restricting the values that the variables can simultaneously take. A feasible solution is an instantiation of all the variables that satisfies all the given constraints. Optimal solution can be found by instantiating all the variables that satisfy all the constraints and optimize the given fitness function.

2.3. University course timetable constraints

In order to generate a feasible and near-optimal timetable, constraints play an important role. It can be classified into hard and soft constraints. Hard constraints must be satisfied completely, while soft constraints have less priority to be satisfied. Violation of the soft constraints will not cause the timetable to lose its feasibility.

Hard Constraints that have to be met:

Lecturer time-clash constraint: A lecturer should not be assigned to teach more than one subject in the same timeslot.

- Equation (1) represents the inequality constraints for lecturer time-clash constraint.

$$T(S_i) \neq T(S_j) \quad \text{if} \quad L(S_i) = L(S_j) \quad (1)$$

where $T(S_i)$ and $T(S_j)$ are timeslots for subjects S_i and S_j respectively. $L(S_i)$ and $L(S_j)$ are the lecturers of subjects S_i and S_j respectively, $i, j = 1, 2, \dots, n$.

Student group time-clash constraint: A student group should not be assigned to attend more than one subject in the same timeslot.

- Equation (2) represents the inequality constraints for student group time-clash constraints.

$$T(S_i) \neq T(S_j) \quad \text{if} \quad G(S_i) = G(S_j) \quad (2)$$

where $G(S_i)$ and $G(S_j)$ are the student groups of subjects S_i and S_j respectively, $i, j = 1, 2, \dots, n$.

Classroom time-clash constraint: one room should not be assigned to more than one subject for the same timeslot.

- Equation (3) represents the inequality constraints for classroom time-clash constraints.

$$T(S_i) \neq T(S_j) \quad \text{if} \quad R(S_i) = R(S_j) \quad (3)$$

where $R(S_i)$ and $R(S_j)$ are the classrooms of subjects S_i and S_j respectively, $i, j = 1, 2, \dots, n$.

Classroom capacity constraint: the number of students of a lesson assigned to a room should be less than or equal to the capacity of the classroom.

- Equation (4) represents the inequality constraints for classroom capacity constraints.

$$N(S_i) \leq Z(R(S_i)) \quad (4)$$

where $N(S_i)$ is the number of students of subjects S_i and $Z(R)$ is the capacity of the classroom R , $i, j = 1, 2, \dots, n$.

Classroom and timeslot-domain constraint: classrooms or timeslots assigned to subjects must be within the range of domain.

Timeslot constraint: certain timeslots are reserved for non-academic activities such as co-curriculum and lunch hours; therefore, they are not available for any lectures.

Soft Constraints that have to be met:

The scheduled *timeslot* of the subject should fall within the preference sets as much as possible.

The scheduled *classroom* of the subject should fall within the preference sets as much as possible.

3. Hybrid particle swarm optimization-constraint-based reasoning: proposed algorithm

3.1. Particle swarm optimization (PSO)

The particle swarm optimization (PSO) algorithm was firstly inspired by Kennedy and Eberhart [7]. It is stochastic population-based evolutionary algorithms that use to find the optimum solution over the search space in a variety points and converge the swarm at the most promising area. Number of this variety points (called particle) is a constant in their population size and each of the particle is a candidate solution. Each particle has a velocity that allows it to “swarm” around over the search space for an optimum solution.

Assuming that the search space is D -dimensional denote by $X_i = (x_{i1}, x_{i2}, \dots, x_{iD})$ the i th particle of the swarm and by $P_i = (p_{i1}, p_{i2}, \dots, p_{iD})$ the best position from its memory ability in the search space. Let g be the index of the best particle in the swam and $V_i = (v_{i1}, v_{i2}, \dots, v_{iD})$ the velocity position of the i th particle.

The swarm is manipulated according to the following equations

$$v_{id} = \chi * (w * v_{id} + c_1 * r_1 * (p_{id} - x_{id}) + c_2 * r_2 * (p_{gd} - x_{id})) \quad (5)$$

$$x_{id} = x_{id} + v_{id} \quad (6)$$

where $d = 1, 2, \dots, D$; $i = 1, 2, \dots, N$ and N is the size of the population; w is the inertia weight; c_1 and c_2 are two positive acceleration constants; r_1 and r_2 are two random values ranging from $[0, 1]$ and χ is a constriction factor used to control magnitude of the velocity. [22] has pointed out that the purpose of constriction factor is to insure the convergence of the PSO. Equation (7) shows the find out of constriction factor.

$$\chi = 2 / |2 - \phi - \sqrt{\phi^2 - 4\phi}| \quad (7)$$

where, $\phi = c_1 + c_2$, $\phi > 4$. Typically, ϕ is set to 4.1 and χ is thus become 0.7298.

Equation (5) is used to calculate i th particle's new velocity and equation (6) will update the position of the particle. The performance of each particle is measured using a fitness function (equation (8)). Figure 2 shows the standard PSO algorithm for UCTP.

Procedure General_PSO_in_UCTP

```

1  Begin
2  For each particle
3    Initialize particle
4  End For
5  Do
6    For each particle
7      Calculate fitness value
8      If the fitness value is better than the best fitness value (Pbest) in history
9        Set current value as the new Pbest
10   End For
11  Choose the particle with the best fitness value of all the particles as the
12   Gbest
13  For each particle
14    Calculate particle velocity (classroom & timeslot) according to previous
15    equations
16    Update particle position (classroom & timeslot) according to previous
17    equations
18  End For
19  Do
20    Copy particle position (classroom & timeslot) to variables Lesson(i)
21  While total number of lesson is not reach
22  End

```

Figure 2. A standard particle swarm optimization algorithm for UCTP

3.1.1. Particle representation

A direct representation is used. Each particle contains information about the timeslot and classroom for a particular lesson. Figure 3 shows example of the particles where $Subject_i$ refers to a subject offered during the semester, N is the maximum number of subjects and $i \in \{1, \dots, N\}$. $Timeslot_j$ is a number of timeslot in a weekly timetable, M is the maximum number of timeslots while $j \in \{1, \dots, M\}$ and $Classroom_k$ is a classroom that is used for the subject, P is the maximum number of classroom and $k \in \{1, \dots, P\}$.

$Subject_1$	$Subject_2$	$Subject_3$	$Subject_N$
$Timeslot_j$	$Timeslot_j$	$Timeslot_j$		$Timeslot_j$
$Classroom_k$	$Classroom_k$	$Classroom_k$		$Classroom_k$

Figure 3. Particle representation for UCTP

3.2. Constraint-based reasoning (CBR)

CBR is a problem solving techniques that is used to solve CSP [15]. A CSP normally identified a problem based on its constraints in a finite domain. Such domain contains a set of variables whose values can only be taken from the domain, and a set of constraints, each constraint specifying the allowed values for a group of variables. A solution is therefore a way of assigning a value to each variable in such a way all constraints are satisfied by these values. To find a solution, it is usually involved with a search technique, in particular a form

of backtracking or local search. This research incorporates the backtracking search with PSO to validate the solution found. Inconsistent value found will be repaired using the backtracking search while looking for valid solution.

3.2.1. Backtracking algorithm

Backtracking is a method that repeatedly chooses a variable value which is consistent with the variable values of the current solution, if inconsistency found [15]. As soon as all the variables that are relevant to the constraint are instantiated, the constraint is checked for its validity. If parts of the variables violate any of the constraints, backtracking is performed starting from the current instantiated variables to other available variables. To facilitate the search for a solution, the UCTPs can be represented as a graph tree organization for a state space as shown in Figure 4. The levels of tree correspond to $Classroom_k$ and $Timeslot_j$. Whenever a potential solution is found by using PSO, it will validate the $Classroom_k$ and $Timeslot_j$ values for the current $Subject_i$.

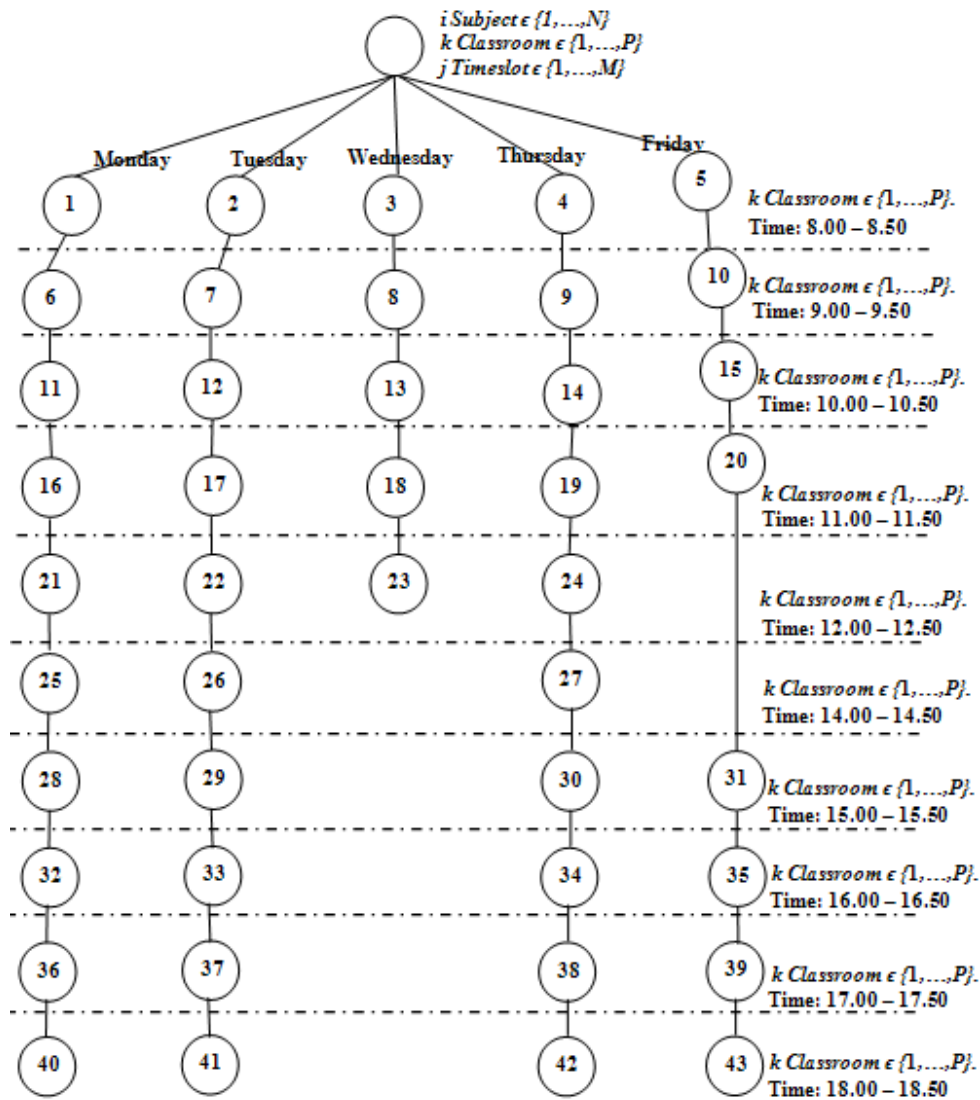


Figure 4. A search tree for a constraint satisfaction problem

3.3. Incorporation of constraint-based reasoning into particle swarm optimization

PSO is known to have the ability to find a near-optimal solution by updating its particle flying position and velocity vector with a suitable fitness function. Authors [11, 12] have reported that by adapting PSO alone in solving UCTP is not enough to get near optimal solution in complex problems such as optimization of the constraint satisfaction problems (CSPs). This is mainly because the nature of the PSO was designed to find potential solutions by updating its particle position and velocity based on a specific fitness function but not constraint handling. Therefore, a technique that handles constraints is needed to find a near-optimal solution for the UCTP. The CBR was specially designed to handle CSPs [16]. Thus, by incorporating CBR into PSO, it can efficiently optimize the UCTP constraints.

The incorporation of CBR into PSO has been divided into two phases. The first is PSO phase. PSO particles are used to find potential solution for the classroom and timeslot allocation for a subject. Every single subject has the information of which student group is taking it and who will teach the subject. Each particle will carry two values: the coordinate of classroom and timeslot value. A search space range that equalize to a total number of classroom that represented by axis-x and timeslot that represented by axis-y are defined. For example, if the number of classrooms and timeslots is 10 and 10, then the search space range is a 10x10. By declaring the search space size, the particles will not swarm out of the search space range. Each particle will update its classroom and timeslot position (coordinate value) by sharing information in between its neighbourhood and a fitness function (equation (8)) is used to maximize the classroom and timeslot preferences. Each best position is the summation preference value of classroom and timeslot. A preferred value is a value that carries each classroom and timeslot based on its facilities and accessibility. The higher the preferred value the more facilities it provide for classroom and more accessibility time for a timeslot. The highest value among the particle, the position of that particle will be chosen as a potential solution for a subject. When a potential solution is found, validity of the potential solution will be done at phase two.

The second phase is the CBR phase. The CBR is used to validate the solution validity. For example, to check the student group and lecturer availability for that subject, it is big enough for the classroom to accommodate the number of student for the subject and to check the availability of classroom and timeslot for the subject. If the potential solution is valid, the subject will be allocated. When there occur clashes or invalid potential solution, the CBR will search for the higher preferred value for classroom and timeslot while avoiding the hard constraints violation. If it found an available classroom and timeslot, the subject will be allocated into it or else it will backtrack to other available classroom and timeslot for the subject allocation while satisfying all the constraints. The incorporation of CBR into PSO algorithm is shown in Figure 5 and the detail flowchart of this incorporation is shown in Figure 6, adapted from [13]. Figure 7 shows the CBR-backtracking search space tree strategies. According to Figure 7, a backtracking search will be perform when the validity test meet with a symbol "X" until a tick symbol is found, while the potential solution will be accept when it meet up with a tick symbol. The symbol "X" means invalid or clashes occurred while the tick symbol means the potential solution found is valid with all the constraints satisfied. Figure 8 shows the checking of lecturer time-clash algorithm. Figure 9 shows the checking of student group time-clash algorithm. Figure 10 shows the checking of classroom time-clash algorithm. Finally, Figure 11 shows the checking of classroom capacity constraint algorithm.

Procedure Hybrid_PSO_CBR

```
1  Begin
2  Read constraints - Subject(i)
3  Post constraints - variables of Subject(i)

4  For each particle
5  Initialize particle
6  End For

7  Do
8  For each particle
9  Calculate fitness value
10  If the fitness value is better than the best fitness value (Pbest) in history
    Set current value as the new Pbest
11  End For

12  Choose the particle with the best fitness value of all the particles as the Gbest

13  For each particle
14  Calculate particle velocity (Room & timeslot) according to previous equations
15  Update particle position (Room & timeslot) according to previous equations
16  End For

17  Copy particle position (Room & timeslot) to variables Subject(i)
18  Do
19  Check if particle position is legal or valid while satisfying the constraints
20  While particle position is not legal or valid

21  Return legal or valid particle position
22  Evaluate solution

23  While total number of subject is not reach
24  End
```

Figure 5. The proposed hybrid PSO-CBR algorithm

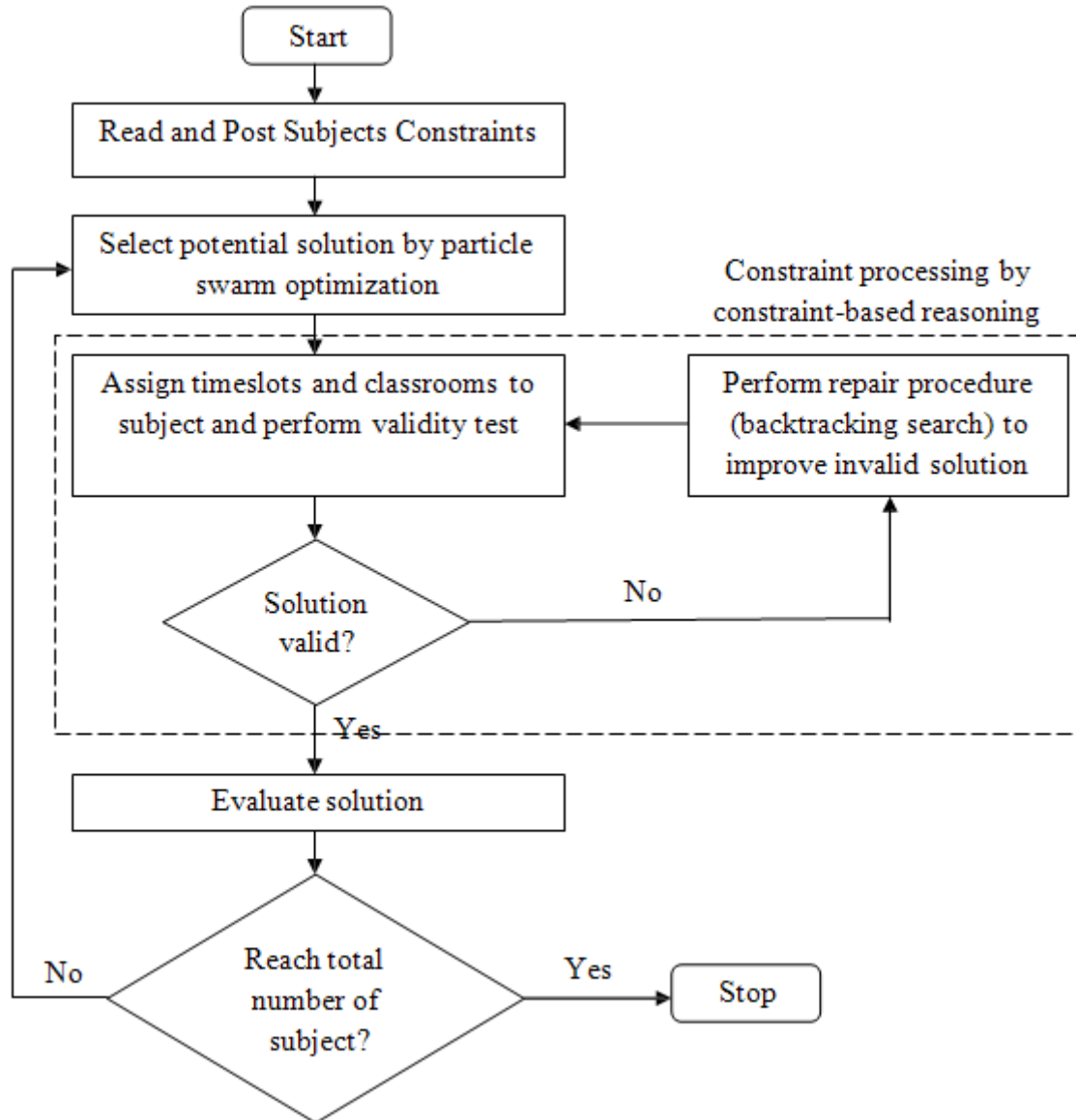


Figure 6. Flowchart of the proposed hybrid particle swarm optimization- constraint-based reasoning

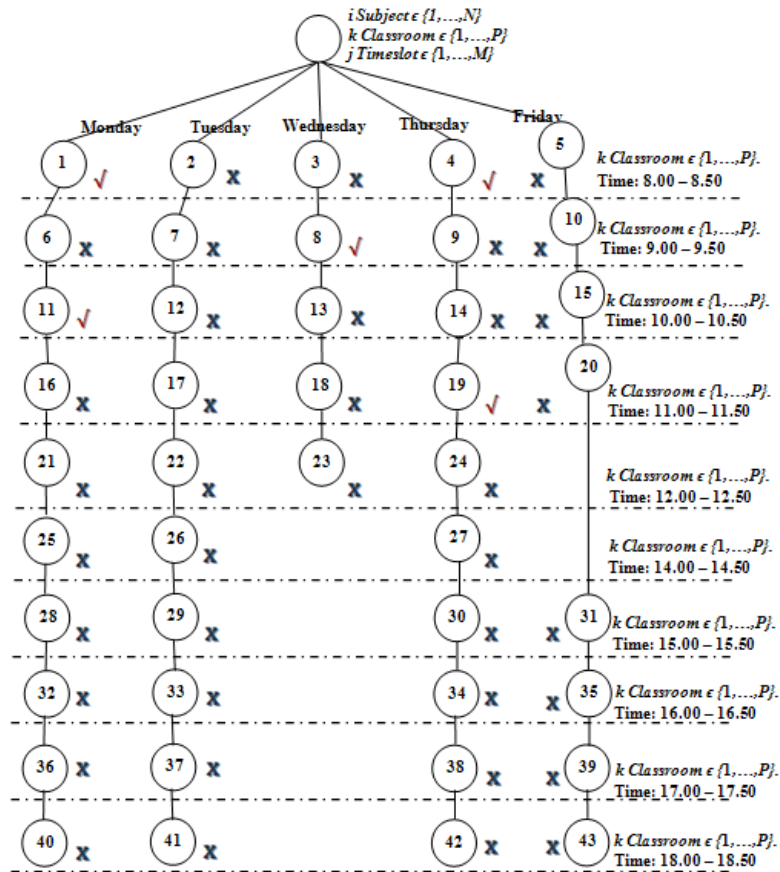


Figure 7. The CBR-backtracking search space tree strategies

```

Algorithm: is_lecturer_free

Input: Potential solution from PSO → Timesloti value and Classroomk value with the current Subjecti

no_of_subject: integer
subject[n]: array of total number of lesson in that subject
timeslot: integer
classroom: integer
lecturer_id: integer

solution[classroom][timeslot]: array for subject allocation
chLecAvai[timeslot][lecturer_id];

Output: lecturer_free : 0

Begin
  For i := 1 to no_of_subject do
    For i := 1 to subject[n] do
      If solution[classroom][timeslot]=0 && chLecAvai[timeslot][lecturer_id]=0 Then
        solution[classroom][timeslot]=no_of_subject
        chLecAvai[timeslot][lecturer_id]=no_of_subject
      EndIf
    EndFor
  EndFor
End

```

Figure 8. Checking of lecturer time-clash algorithm

```
Algorithm: is_student_group_free

Input: Potential solution from PSO → Timeslotj value and Classroomk value with the current
Subjecti

no_of_subject: integer
subject[n] : array of total number of lesson in that subject
timeslot : integer
classroom: integer
student_group_id : integer
subjectSec[m]: array of subject section

solution[classroom][ timeslot] : array for subject allocation
chStudentGroupAvai[timeslot][ student_group_id][ subjectSec[m]];

Output: student_group_free : 0

Begin
  For i := 1 to no_of_subject do
    For i := 1 to subject[n] do
      If solution[classroom][ timeslot]=0
      && chStudentGroupAvai[timeslot][ student_group_id][ subjectSec[m]]= 0 Then
        solution[classroom][ timeslot]= no_of_subject
        chStudentGroupAvai[timeslot][ student_group_id][ subjectSec[m]]=
          no_of_subject
      EndIf
    EndFor
  EndFor
End
```

Figure 9. Checking of student group time-clash algorithm

```
Algorithm: is_room_time_free

Input: Potential solution from PSO → Timeslotj value and Classroomk value with the current
Subjecti

no_of_subject: integer
timeslot : integer
classroom: integer

solution[classroom][timeslot] : array for subject allocation

Output: room_time_free : 0

Begin
  For i := 1 to no_of_subject do
    If solution[classroom][timeslot]=0Then
      solution[classroom][timeslot]= no_of_subject
    EndIf
  EndFor
End
```

Figure 10. Checking of classroom time-clash algorithm

```

Algorithm: check_classroom_capacity_constraint

Input: Potential solution from PSO → Timeslotj value and Classroomk value with the current
        Subjecti

no_of_subject: integer
classroom_capacity[Classroomk]: array for classroom_capacity allocation
no_of_stu[no_of_subject]: array for number of student taking the subject
timeslot : integer
classroom: integer

solution[classroom][timeslot] : array for subject allocation

Output: classroom_capacity_const : Boolean

Begin
  For i := 1 to no_of_subject do
    If classroom_capacity[Classroomk] >= no_of_stu[no_of_subject] Then
      classroom_capacity_const = true
    Else
      classroom_capacity_const = false
    EndIf
  EndFor
End

```

Figure 11. Checking of classroom capacity constraint algorithm

3.4. Fitness Function

The fitness function chosen is used to optimize the value ordering criteria of the preference for the best classrooms and the best timeslots [17]. By ordering the values (classroom and timeslot), the preferences for the values can be determined. The best classroom and timeslot are given the highest preference score. By using this fitness function, subject will be assigned with the best classroom and timeslot while no constraints is violated. This solution is called the feasible and near-optimal solution for the CSP. The fitness function $f(t)$ of a timetable t is given as follow:

$$f(t) = \sum_{i=1}^n (P(T(S_i)) + P(R(S_i))) \quad (8)$$

where $P(T(S_i))$ are timeslot preferences for subjects S_i , $i=1,2,\dots,n$ and $P(R(S_i))$ are room preferences for subjects S_i , $i=1,2,\dots,n$. The fitness function $f(t)$ for each particle (solution) is computed by the total summation of preference value of timeslot $P(T(S_i))$ and preference value of room $P(R(S_i))$ for subject S_i , $i=1,2,\dots,n$.

4. Experimental and results

The proposed algorithms have been tested using course timetable data from Faculty of Computer Science and Information System, University Teknologi Malaysia, for Semester I 2008/2009. Table 1 summarizes the information for this university timetable. A lesson (sometimes referred as a lecture) is an event where lecturers meet students for the purpose of teaching and learning in a classroom (resource). Each subject has two to four lessons per week, depending on the categories of the subject course curriculum. There are 145 subjects supporting various course major programs, and 17 rooms of various sizes. As shown in Table 1, there are 55 timeslots (five days a week with eleven sessions per day of 50 minutes per each lesson), 12 were reserved and 43 timeslots is available for lectures. The parameter settings for the PSO are reported in Table 2 while Table 3 and 4 summarize the timeslot and room preference scores.

Table 1. Summary of the university timetable information's

Items	Data
Number of subjects	145
Number of subjects with four lessons per week	80
Number of subjects with three lessons per week	60
Number of subjects with two lessons per week	5
Number of lessons for all the subjects	510
Number of lecturers	82
Number of student groups	16
Number of students	916
Number of classrooms	17
Number of classrooms with 50 persons capacity	10
Number of classrooms with 120 persons capacity	6
Number of classrooms with 160 persons capacity	1
Total timeslots	55
Total timeslots reserved	12
Total timeslots available	43

Table 2. Parameters of PSO

Parameter	Value
No. of Generation	300
No. of Particles	10
c1	2.8
c2	1.3
w	$1 / (2 * \log(2))$

Table 3. Summary of the university timeslots preference score

Timeslot No.	Duration	Day	Preference Score
1, 2, 3, 4, 5	8:00 – 8:50 am	Mon, Tues, Wed, Thurs, Fri	4
6, 7, 8, 9, 10	9:00 – 9:50 am	Mon, Tues, Wed, Thurs, Fri	4
11, 12, 13, 14, 15	10:00 – 10:50 am	Mon, Tues, Wed, Thurs, Fri	4
16, 17, 18, 19, 20	11:00 – 11:50 am	Mon, Tues, Wed, Thurs, Fri	2
21, 22, 23, 24	12:00 – 12:50 pm	Mon, Tues, Wed, Thurs	2
25, 26, 27	14:00 – 14:50 pm	Mon, Tues, Thurs	2
28, 29, 30, 31	15:00 – 15:50 pm	Mon, Tues, Thurs, Fri	2
32, 33, 34, 35	16:00 – 16:50 pm	Mon, Tues, Thurs, Fri	1
36, 37, 38, 39	17:00 – 17:50 pm	Mon, Tues, Thurs, Fri	1
40, 41, 42, 43	18:00 – 18:50 pm	Mon, Tues, Thurs, Fri	1

Table 4. Summary of the university rooms preference score

Room No.	Room Name (N28 Building)	Room Capacity	Preference Score
1, 10	KPU 1, KPU 10	50	4
2, 5, 6, 7, 8, 9	KPU 2, KPU 5, KPU 6, KPU 7, KPU 8, KPU 9	50	1
3, 4	KPU 3, KPU 4	50	2
11	BK 1	160	4
12	BK 2	120	4
13, 14, 16, 17	BK 3, BK 4, BK 6, BK 7	120	2
15	BK 5	120	1

The results generated by standard PSO algorithm is shown in Table 5 while Table 6 and 7 show the results generated by hybrid PSO-LS and incorporation of CBR into PSO algorithm (hybrid PSO-CBR). To test the stability of the algorithm, results of 5 separated runs for each algorithm are compared.

Table 5. Results generated by standard PSO after 5 runs

Seed	No. of subjects unallocated	Time (Sec)	Max. value of objective function
1	203	36.00	1375
2	205	36.00	1384
3	201	36.00	1416
4	195	36.00	1452
5	204	36.00	1428
Average	201.6	36.00	1411

Table 6. Results generated by hybrid PSO-LS after 5 runs

Seed	No. of subjects unallocated	Time (Sec)	Max. value of objective function
1	0	36.00	2335
2	0	36.00	2331
3	0	36.00	2336
4	0	36.00	2336
5	0	37.00	2338
Average	0	36.20	2335.2

Table 7. Results generated by hybrid PSO-CBR after 5 runs

Seed	No. of subjects unallocated	Time (Sec)	Max. value of objective function
1	0	37.00	2494
2	0	37.00	2490
3	0	36.00	2496
4	0	36.00	2496
5	0	36.00	2496
Average	0	36.40	2494.4

According to the results generated at Table 7, the timetable generated using proposed algorithm (hybrid PSO-CBR) is attractive because it produces a feasible and near-optimal solution (maximizing the classroom and timeslot preferences) while satisfied all the constraints. By looking at the maximum value of objective function at Table 7, it is obvious that the preferences are maximized if compare to objective function at Table 5 (standard PSO algorithm) and Table 6 (hybrid PSO-LS algorithm).

The computational time used to generate a solution by proposed algorithm is slightly longer compared to hybrid PSO-LS and standard PSO algorithm due to the potential solution is validated and backtrack the search when invalid solution found. Hybrid PSO-LS have produce solution faster than hybrid PSO-CBR because it doesn't provide any backtrack procedure but just accept any solution as long as the constraints is not violated (doesn't maximize the preferences). As mentioned earlier, PSO alone is inadequate to solve CSPs and it is proven that there are some unallocated subjects if only PSO algorithm is used due to its disability to handle constraints. Hence, the time usage for a solution generated using PSO alone also reduce compare to proposed algorithm and hybrid PSO-LS. The performance of the proposed algorithm is shown by the resources (classroom and timeslot) utilization as shown in Figure 12 and 13. Appendix A shows the example of a timetable generated by proposed algorithm for classroom 1 and 16.

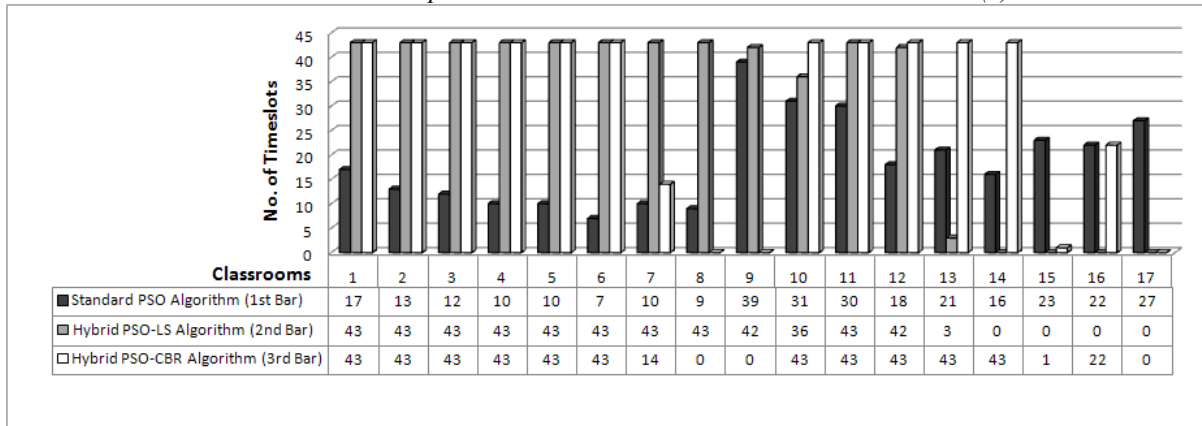


Figure 12. Timeslot utilization

Based on Figure 12, it can be seen that, by using the proposed algorithm all high preference score classrooms are chosen for the subject allocation first before the less preferred one is chosen. Thus, the number of timeslots usage is higher compared to the less preferable classrooms. As can be seen from the figure, the maximum timeslot usage is 43 because the total available timeslot in a week is 43. Since the hybrid PSO-LS does not have backtracking ability, the subjects are allocated if all the constraints are not violated. Thus, it does not maximize the preferences. Some of the subjects are not successfully allocated by standard PSO algorithm because it has no ability to handle constraints so when there are clashes occur, the algorithm just count the unallocated subjects and continue allocating the rest of the subjects. Hence, the number of timeslot utilization for each classroom is lesser. Thus, the total number of timeslot being utilized by proposed algorithm is 315 times compared to 510 times using the standard PSO algorithms.

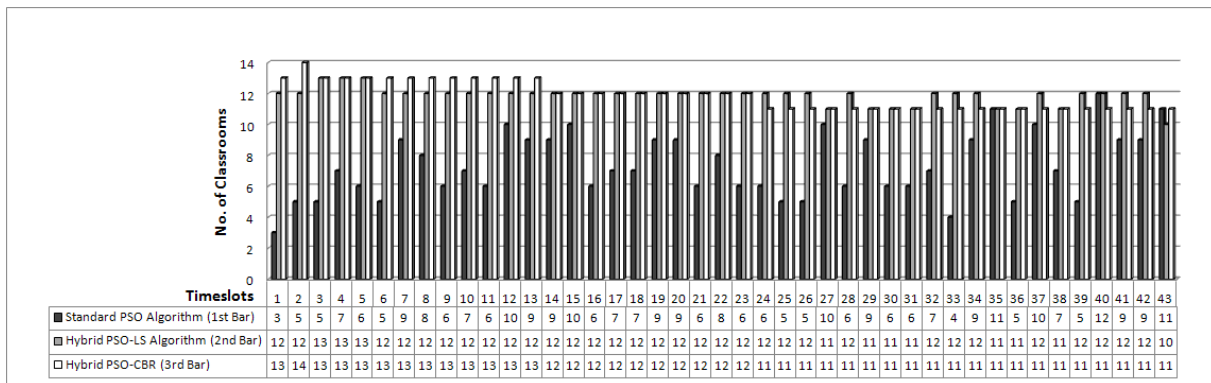


Figure 13. Classroom utilization

Figure 13 shows that the proposed algorithm attempted to maximize the classrooms utilization. All good classrooms and timeslots are used for the subject allocation. Thus, it can be seen that all the less preferable timeslots were less utilized compared to the higher preferred ones. By using hybrid PSO-LS algorithm, even the less preferable timeslots are fully used for subject allocation. Standard PSO algorithm has even worsened the case since it had no ability to handle constraints and also cannot maximize the classrooms and timeslots utilization.

5. Conclusion and future work

This paper proposed an incorporation of CBR into PSO algorithm to solve UCTP. To validate the proposed algorithm, comparison between hybrid PSO-LS algorithm and standard PSO algorithm has been done. From the experiments, it indicates that all techniques provide feasible solution but CBR-PSO gives more optimal and promising result. Using hybrid PSO-CBR algorithm in solving UCTP, the classrooms and timeslots preferences are maximized with the help of the chosen fitness function. The CBR in the hybrid algorithm maintained valid values of the PSO flying position through constraint posting and backtrack to search for valid solution if there's any invalidity. This study also shows that the convergence of optimizing UCTP is efficient due to CBR ability to significantly reduce the search space. Hence, the proposed algorithm is capable in finding feasible and near-optimal solution compared to hybrid PSO-LS and standard PSO algorithm. Future research work will be focusing into experimenting different case of CSPs to further validate the performance of the proposed algorithm.

References


- [1] Adora, E. C., Augusta, Y. H., Bobby O. C. JR., 2002. Parallel Hybrid Adventures with Simulated Annealing and Genetic Algorithms. In: *Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks*.
- [2] Burke, E. K., Elliman, D.G., Weare, R., 1993. Automated Scheduling of University Exams. In: *Proceedings of the IEEE Colloquium on Resource Scheduling for Large Scale Planning System*.
- [3] Chu, S. C., Fang, H. L., 1999. Genetic Algorithms vs. Tabu Search in Timetable Scheduling. In: *Knowledge-Based Intelligent Information Engineering Systems, Third International Conference*, Adelaide, SA, Australia, pp. 492-495.
- [4] Chu, S. C., Chen, Y. T., Ho, J. H., 2006. Timetable Scheduling Using Particle Swarm Optimization. In: *Proceedings of the First International Conference on Innovative Computing, Information and Control*.
- [5] Danial, Q. F., Amir, N. A., M-Hosseini, M., Sarah, S.R., Ehsan, A., Javad, M., 2007. Finding Feasible Timetables with Particle Swarm Optimization. In: *The 4th International Conference of Innovations in Information Technology*, pp. 387-391.
- [6] de Werra, D., 1985. An introduction to timetabling. *European Journal of Operations Research* 19, pp. 151—162.
- [7] Engelbrecht, A. P. *Fundamentals of Computational Swarm Intelligence*. The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England; 2005.
- [8] Fealko, R. D., 2005. Evaluating Particle Swarm Intelligence Techniques for Solving University Examination Timetabling Problems. Graduate School of computer and Information Sciences, Nova Southeastern University: A Dissertation for the degree of Doctor of Philosophy.
- [9] Hany, T. A. Alashwal., 2003. The Development Of Reactive Constraint Agents For The Dynamic Timetabling Problem. Faculty of Computer Science & Information System,

- [10] Hendtlass, T., 2006. A Particle Swarm Algorithm for Complex Quantized Problem Spaces. In: *IEEE Congress on Evolutionary Computation Sheraton Vancouver Wall Centre Hotel*, Valcouver, BC, Canada, July 16-21, pp. 1015-1019.
- [11] Irene, S. F. Ho., Safaai, D., Siti Zaiton, M. H., 2009. A Study on PSO-based University Course Timetabling Problem. In: *IEEE 2009 International Conference on Advanced Computer Control (ICACC 2009)*, Singapore, pp. 648-651.
- [12] Irene, S. F. Ho., Safaai, D., Siti Zaiton, M. H., 2009. A Combination of PSO and Local Search in University Course Timetabling Problem. In: *IEEE 2009 International Conference on Computer Engineering and Technology (ICCET 2009)*, Singapore, pp. 492-495.
- [13] Irene, S. F. Ho., Safaai, D., Siti Zaiton, M. H., 2009. University Course Timetable Planning using Hybrid Particle Swarm Optimization. In: *ACM 2009 World Summit on Genetic and Evolutionary Computation*, 12-14 June 2009, Shanghai, China, pp. 239-246.
- [14] Legierski, W., 2002. Constraint-based reasoning for timetabling. In: *Artificial Intelligence Method: AI-METH*, Gliwice, Poland.
- [15] Roman, B., 1995. Constraint Propagation and Backtracking-Based Search - A brief introduction to mainstream techniques of constraint satisfaction: Dick Pountain, BYTE.
- [16] Safaai, D., Sigeru, O., Hiroshi, O., Puteh, S., 1999. Incorporating Constraint Propagation in Genetic Algorithm for University Timetable Planning. *Engineering Applications of Artificial Intelligence* 12, pp. 231-253.
- [17] Safaai, D., Sigeru, O., Hiroshi, O., 2000. Timetable Planning using the Constraint-based Reasoning. *Computer & Operations Research*, 27, 2000, pp. 819-840.
- [18] Sigl, B., Golub, M., Mornar, V., 2003. Solving Timetable Scheduling Problem Using Genetic Algorithms. In: *The 25th International Conference Information Technology Interfaces ITI 2003*, Cavtat, Croatia.
- [19] Tony, L., Eric, M., Frederic, S., 2004. Hybridization Strategies for Local Search and Constraint Propagation. In: *Proceeding of the 4th Workshop on Cooperative Solvers in Constraint Programming*.
- [20] Tuncay, Y., 2007. Constraint-based School Timetabling Using Hybrid Genetic Algorithms. In: *AI*IA 2007: Artificial Intelligence and Human-Oriented Computing, LNCS*, vol. 4733, Springer, Heidelberg, pp. 848--855.
- [21] Zalmiyah, Z., Case-Based Reasoning Approach for Reactive Timetabling. 2001. Faculty of Computer Science & Information System, University of Technology Malaysia, Master Thesis.
- [22] Clerc, M., Kennedy, J., 2002. The particle swarm explosion, stability, and convergence in a multidimensional complex space. In: *IEEE Transaction on evolutionary Computation* 6, PP.

Appendix A

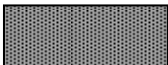
An example of timetable for classroom no. 1 generated by proposed algorithm

Time	Monday	Tuesday	Wednesday	Thursday	Friday
8:00 – 8:50	SCB ₁ SCB ₁₀₃₂ S ₁ , L ₂ , Lec ₁	SCB ₁ SCB ₁₀₃₂ S ₁ , L ₃ , Lec ₁	SCB ₁ SCB ₁₀₃₂ S ₁ , L ₄ , Lec ₁	SCB ₁ SCK ₁₂₁₃ S ₁ , L ₁ , Lec ₂	SCB ₁ SCK ₁₂₁₃ S ₁ , L ₂ , Lec ₂
9:00 – 9:50	SCB ₁ SCK ₁₂₁₃ S ₁ , L ₃ , Lec ₂	SCJ ₁ SCJ ₁₀₁₃ S ₁ , L ₁ , Lec ₁₄	SCJ ₁ SCJ ₁₀₁₃ S ₁ , L ₃ , Lec ₁₄	SCJ ₁ SCJ ₁₀₁₃ S ₂ , L ₁ , Lec ₁₄	SCJ ₁ SCJ ₁₀₁₃ S ₂ , L ₃ , Lec ₁₄
10:00 – 10:50	SCJ ₁ SCJ ₁₀₁₃ S ₂ , L ₄ , Lec ₁₄	SCJ ₁ SCJ ₁₀₁₃ S ₃ , L ₂ , Lec ₁₅	SCJ ₁ SCJ ₁₀₁₃ S ₃ , L ₃ , Lec ₁₅	SCJ ₁ SCJ ₁₀₁₃ S ₄ , L ₁ , Lec ₁₅	SCJ ₁ SCJ ₁₀₁₃ S ₄ , L ₂ , Lec ₁₅
11:00 – 11:50	SCJ ₁ SCJ ₁₀₁₃ S ₄ , L ₃ , Lec ₁₅	SCJ ₁ SCJ ₁₀₁₃ S ₄ , L ₄ , Lec ₁₅	SCJ ₁ SCJ ₁₀₁₃ S ₅ , L ₂ , Lec ₁₆	SCJ ₁ SCJ ₁₀₁₃ S ₅ , L ₃ , Lec ₁₆	SCJ ₁ SCJ ₁₀₁₃ S ₅ , L ₄ , Lec ₁₆
12:00 – 12:50	SCJ ₁ SCJ ₁₀₁₃ S ₆ , L ₂ , Lec ₁₇	SCJ ₁ SCJ ₁₀₁₃ S ₆ , L ₃ , Lec ₁₇	SCJ ₁ SCJ ₁₀₁₃ S ₇ , L ₁ , Lec ₁₈	SCJ ₁ SCJ ₁₀₁₃ S ₇ , L ₂ , Lec ₁₈	
13:00 – 13:50					
14:00 – 14:50	SCJ ₁ SCJ ₁₀₁₃ S ₇ , L ₄ , Lec ₁₈	SCJ ₁ SCJ ₁₀₁₃ S ₈ , L ₂ , Lec ₁₈		SCJ ₁ SCJ ₁₀₁₃ S ₉ , L ₁ , Lec ₁₉	
15:00 – 15:50	SCJ ₁ SCJ ₁₀₁₃ S ₉ , L ₂ , Lec ₁₉	SCJ ₁ SCJ ₁₀₁₃ S ₈ , L ₄ , Lec ₁₈		SCJ ₁ SCJ ₁₀₁₃ S ₉ , L ₃ , Lec ₁₉	SCJ ₁ SCJ ₁₀₁₃ S ₁₀ , L ₁ , Lec ₉
16:00 – 16:50	SCJ ₁ SCJ ₁₀₁₃ S ₁₀ , L ₂ , Lec ₉	SCJ ₁ SCJ ₁₀₁₃ S ₁₀ , L ₃ , Lec ₉		SCJ ₁ SCJ ₁₀₁₃ S ₁₀ , L ₄ , Lec ₉	SCB ₂ SCK ₂₂₄₃ S ₁ , L ₁ , Lec ₂
17:00 – 17.50	SCB ₂ SCK ₂₂₄₃ S ₁ , L ₃ , Lec ₂	SCB ₂ SCK ₂₂₄₃ S ₁ , L ₄ , Lec ₂		SCS ₂ SCK ₂₂₄₃ S ₂ , L ₁ , Lec ₅	SCS ₂ SCK ₂₂₄₃ S ₂ , L ₃ , Lec ₅
18.00 – 18.50	SCB ₂ SCK ₁₈₁₃ S ₁ , L ₁ , Lec ₃	SCB ₂ SCK ₁₈₁₃ S ₁ , L ₂ , Lec ₃		SCB ₂ SCK ₁₈₁₃ S ₁ , L ₄ , Lec ₃	SCS ₁ SCK ₁₈₁₃ S ₂ , L ₁ , Lec ₃

- Notes:
- SCS₂ - Student course major code
 - SCK₂₂₄₃ - Subject code
 - S₂ - Subject section
 - L₁ - Subject lesson
 - Lec₅ - Lecturer ID
 -  - Reserved timeslots

An example of timetable for classroom no. 16 generated by proposed algorithm

Time	Monday	Tuesday	Wednesday	Thursday	Friday
8:00 – 8:50	SCS _{3KI} SCK ₃₁₄₃ S ₁ , L ₁ , Lec ₇₅	SCS _{3KP} SCK ₂₂₉₃ S ₁ , L ₃ , Lec ₈	SCS _{3KI} SCK ₃₁₄₃ S ₁ , L ₁ , Lec ₇₅	SCS _{3KP} SCK ₃₃₂₃ S ₁ , L ₂ , Lec ₇₉	SCS _{3KI} SCK ₃₁₅₃ S ₁ , L ₂ , Lec ₇₆
9:00 – 9:50	SCS _{3KI} SCK ₃₁₅₃ S ₁ , L ₃ , Lec ₇₆	SCS _{3KI} SCK ₃₁₉₃ S ₁ , L ₁ , Lec ₇₆	SCS _{3KI} SCK ₃₁₉₃ S ₁ , L ₂ , Lec ₇₆	SCS _{3KI} SCK ₃₁₉₃ S ₁ , L ₃ , Lec ₇₆	SCS _{3KP} SCK ₃₃₂₃ S ₂ , L ₁ , Lec ₇₉
10:00 – 10:50	SCS _{3KP} SCK ₃₂₁₃ S ₁ , L ₁ , Lec ₅₅	SCS _{3KP} SCK ₃₃₂₃ S ₂ , L ₂ , Lec ₇₉	SCS _{3KP} SCK ₃₂₉₃ S ₁ , L ₂ , Lec ₁₆	SCS _{3GMM} SCK ₃₆₀₃ S ₁ , L ₃ , Lec ₈₀	SCS _{3GMM} SCK ₃₆₁₃ S ₁ , L ₂ , Lec ₈₁
11:00 – 11:50	SCS _{3GMM} SCK ₃₆₂₃ S ₁ , L ₁ , Lec ₈₂	SCS _{3GMM} SCK ₃₆₂₃ S ₁ , L ₂ , Lec ₈₂	SCS _{3GMM} SCK ₃₆₉₃ S ₁ , L ₁ , Lec ₃₄	SCS _{3GMM} SCK ₃₆₉₃ S ₁ , L ₂ , Lec ₃₄	SCS _{3GMM} SCK ₃₆₉₃ S ₁ , L ₃ , Lec ₃₄
12:00 – 12:50		SCS _{3GMM} SCK ₃₇₂₃ S ₁ , L ₂ , Lec ₁₉	SCS _{3GMM} SCK ₃₇₂₃ S ₁ , L ₃ , Lec ₁₉		
13:00 – 13:50					
14:00 – 14:50					
15:00 – 15:50					
16:00 – 16:50					
17:00 – 17:50					
18:00 – 18.50					

- Notes:
- SCS_{3KP} - Student course major code
 - SCK₃₃₂₃ - Subject code
 - S₂ - Subject section
 - L₁ - Subject lesson
 - Lec₇₉ - Lecturer ID
 -  - Reserved timeslots