Solving University Course Timetabling Problem Using Multi-Depth Genetic Algorithm

Solving UCTP Using MDGA

Alfian Akbar Gozali[1,*] and *Shigeru* Fujimura[1,]

[1]Graduate School of IPS, Waseda University, Kitakyushu, Japan

Abstract. The University Course Timetabling Problem (UCTP) is a scheduling problem of assigning teaching event in certain time and room by considering the constraints of university stakeholders such as students, lecturers, and departments. The constraints could be hard (encouraged to be satisfied) or soft (better to be fulfilled). This problem becomes complicated for universities which have an immense number of students and lecturers. Moreover, several universities are implementing student sectioning which is a problem of assigning students to classes of a subject while respecting individual student requests along with additional constraints. Such implementation enables students to choose a set of preference classes first then the system will create a timetable depend on their preferences. Subsequently, student sectioning significantly increases the problem complexity. As a result, the number of search spaces grows hugely multiplied by the expansion of students, other variables, and involvement of their constraints. However, current and generic solvers failed to meet scalability requirement for student sectioning UCTP. In this paper, we introduce the Multi-Depth Genetic Algorithm (MDGA) to solve student sectioning UCTP. MDGA uses the multiple stages of GA computation including multi-level mutation and multi-depth constraint consideration. Our research shows that MDGA could produce a feasible timetable for student sectioning problem and get better results than previous works and current UCTP solver. Furthermore, our experiment also shows that MDGA could compete with other UCTP solvers albeit not the best one for the ITC-2007 benchmark dataset.

1 Introduction

The University Course Timetabling Problem (UCTP) is a scheduling problem of assigning teaching event in certain time and room by considering the constraints of university stakeholders such as students, lecturers, and departments. The constraints could be hard (encouraged to be satisfied) or soft (better to be fulfilled). Regarding its difficulty, Garey included timetabling as an NP-Hard problem [1].

Moreover, several universities such as Telkom University [2], Purdue [3], and the University of Waterloo [4] implement student sectioning. Student sectioning is a problem of assigning students to classes of a subject while respecting individual student requests along with additional constraints [5]. Therefore, the fulfillment of each students' preference is encouraged, as well.

*e-mail: alfian@tass.telkomuniversity.ac.id

In regular timetabling, we place student enrollment (the process which the students choose their classes) after the class timetable available. Oppositely, in the student sectioning, students choose a set of preference classes first then the system will create a timetable depend on their preferences. Subsequently, student sectioning significantly increases the problem complexity. As a result, the number of search spaces grows hugely multiplied by the expansion of students, other variables, and involvement of their constraints.

UCTP is a minimizing optimization problem, so the objective is to minimize all the predefined constraint violation for each of the teaching events. One of the most recently studied for UCTP is the application of genetic algorithms (GA), which are based on the theory of evolution [6], and that have proved to be efficient for problems of moderate and realistic size such as AIMGA [2], Parallel GA [7, 8], Fast Practical GA [9], Directed GA [10], and NSGA-II [11].

In GA, the most time consuming part which is in the fitness evaluation. Thus, student sectioning with large number of variables will make GA runs impractical. To implement GA as a practical solution for this problem, we need to divide the problem to be smaller problem so we could solve it, and then increase the level gradually until we reach the goal. This paper proposes Multi-Depth GA (MDGA) as a GA variation implementing such approach.

MDGA uses three levels of objective function which is differed by its complexity (depth of the problem): shallow, medium, and deep objective function. These objective functions are applied in different step of GA. Moreover, MDGA implements advanced mutation operation by combining three different local search which specifically designed for student sectioning UCTP. Finally, the combination of multi-level objective function and advanced mutation in MDGA will be a significant advantage for overcoming student sectioning UCTP.

Taken together, the primary motivation of this work is how to design MDGA for student sectioning UCTP. In detail, the main goals of this research are (1) to formalize a real-world scaling student sectioning UCTP, (2) to design MDGA to meet the problem requirements, and (3) to analyze MDGA performance in handling student sectioning UCTP.

This paper consists of five sections. We organize the remainder of this paper as follows. Section 2 talks about the student sectioning UCTP. Section 3 shows the MDGA design to meet UCTP problems. Section 4 shows how we conducted the experiments, results, and analysis. The last but not the least, Section 5 takes place as the conclusion and discussion of this work.

2 University Course Timetabling Problem

Generally, UCTP is a problem of arranging a set of teaching events (events) into a predefined packet of time and room while satisfying all constraints within the problem. Equation 1 is the formulation of a packet (q) of time (t) and room (r).

$$q = (t, r) \tag{1}$$

Time could be different from one university to another, which could be varied in weekly, daily, hourly, or even minutely. For example, a university could implement 40 minutes for a time while a certain university could have 60 minutes. Rooms are UCTP resources which could vary in capacity, facility, ownership, and specialization (for example theory and practice classroom). The notation in Table 1 is used for UCTP formulation.

An event (e) consists of a lecturer (l) who teaches a certain class (c) with a set of students (S), which is defined by following notation:

$$e = (l, c, S) \tag{2}$$

Hence, a timetable is a mapping set of all events into several or all packets. A mapping of a packet and an event is a pair (p), which is defined by following notation:

$$p = (q, e) \tag{3}$$

As referenced from previous research [2, 8], this research also used two types of constraints: the hard and soft constraints. Hard constraint (HC) is a constraint that must be satisfied. Soft constraint (SC) is better to be fulfilled to improve the quality of the timetable.

Table 1. UCTP Formulation

Indices and sets			
$\boldsymbol{p \in P}$	set of pairs		
$\boldsymbol{e \in E}$	set of events	e^p	event of pair p
$\boldsymbol{q \in Q}$	set of packets	q^p	packet of pair p
$\boldsymbol{l \in L}$	set of lecturers	l^p	lecturer of pair p
$\boldsymbol{b \in B}$	set of subjects	b_c	subject of class c
$\boldsymbol{G^b \subseteq L}$	set of group lecturers for subject b		
$\boldsymbol{k \in K \subseteq L}$	set of special lecturers		
$\boldsymbol{r \in R}$	set of rooms	r^p	room of pair p
$\boldsymbol{t \in T}$	set of time	t^p	time of pair p
$\boldsymbol{c \in C}$	set of classes	c^p	class of pair p
$\boldsymbol{d \in D}$	set of days	d^p	day of pair p
$\boldsymbol{z \in Z}$	set of all students		
$\boldsymbol{s \in S \subseteq Z}$	set of students	S^p	students of pair p
Functions			
$\boldsymbol{CNTTIME(l,d,P)}$	count event of l in a day		
$\boldsymbol{DATE(t)}$	return the date of time t		
$\boldsymbol{CNTSTIME(z,d,P)}$	count event of z in a day		
$\boldsymbol{TGAP(t,t')}$	time interval of t and t'		
Parameters			
$\boldsymbol{w_i}$	weighting of constraint i		
$\boldsymbol{V_i}$	total violation of constraint i		
$\boldsymbol{CAP_r}$	capacity of room r		
$\boldsymbol{CAP_e^-}$	minimum room capacity of event e		
$\boldsymbol{X_l}$	prohibited time of lecturer l		
$\boldsymbol{PREF_l}$	preference time of lecturer l		
Constant			
$\boldsymbol{LC^+}$	maximum lecturer event in a day		
$\boldsymbol{LC^-}$	min time interval of two lecturer event		
$\boldsymbol{CC^-}$	minimum class event interval in a week		
$\boldsymbol{SC^+}$	maximum student event in a day		
$\boldsymbol{GC^-}$	minimum group teaching event interval		

This research used five HCs and seven SCs. Where V_i is a violation count for each i constraint. Furthermore, the following equations are the mathematical models of each constraint used in this research:

HC 1. No conflict of lecturers

There is no lecturer who has been set in different room at the same time.

$$V_1 = \sum_{p \in P} \sum_{p' \in P} f_1(p, p') = 0 \tag{4}$$

$$f_1(p,p') = \begin{cases} 1, & if(l^p = l^{p'} \wedge t^p = t^{p'} \wedge p \neq p') \\ 0, & otherwise \end{cases} \tag{5}$$

HC 2. No conflict of classes
There is no packet that has been set for different event at the same time.

$$V_2 = \sum_{p \in P} \sum_{p' \in P} f_2(p,p') = 0 \tag{6}$$

$$f_2(p,p') = \begin{cases} 1, & if(q^p = q^{p'} \wedge p \neq p') \\ 0, & otherwise \end{cases} \tag{7}$$

HC 3. Any event should be scheduled in a suitable capacity room
No event that has been set in room with a less-than suitable capacity.

$$V_3 = \sum_{p \in P} f_3(p) = 0 \tag{8}$$

$$f_3(p) = \begin{cases} 1, & if(CAP^{-}_{e^p} > CAP_{r^p}) \\ 0, & otherwise \end{cases} \tag{9}$$

HC 4. Lecturers should not be scheduled within their time constraints
Lecturers such as professors, rectors, and deans should be set in their time constraints.

$$V_4 = \sum_{p \in P} f_4(p) = 0 \tag{10}$$

$$f_4(p) = \begin{cases} 1, & if(l^p \in K \wedge t^p \in X_{l^p}) \\ 0, & otherwise \end{cases} \tag{11}$$

HC 5. Some lecturers should be scheduled in their time preferences
Lecturers such as professors, rectors, and deans should be set in their preferred time.

$$V_5 = \sum_{p \in P} f_5(p) = 0 \tag{12}$$

$$f_5(p) = \begin{cases} 1, & if(l^p \in K \wedge t^p \notin PREF_{l^p}) \\ 0, & otherwise \end{cases} \tag{13}$$

SC 1. Lecturer assignment spread
The teaching event for a lecturer should be set to a maximum of LC^+ events in a day.

$$Minimize V_6 = \sum_{l \in L} \sum_{d \in D} f_6(l,d,P) \tag{14}$$

$$f_6(l,d,P) = \begin{cases} 1, & if(CNTTIME(l,d,P) > LC^+) \\ 0, & otherwise \end{cases} \tag{15}$$

SC 2. Class event spread

An event of a class should be set to a minimum of CC^{-} days of interval in a week. In a real-world, there is a special case which a class can be lectured in more than once a week. For example, class AR002 must be taught twice a week. Thus, it is possible to have several similar events mapped into different packet. These similar events are interchangeably, which is shown by following notation:

$$MinimizeV_7 = \sum_{p \in P} \sum_{p' \in P} f_7(p, p') \tag{16}$$

$$f_7(p, p') = \begin{cases} 1, & if(DATE(t^p) - DATE(t^{p'}) \geq 0 \wedge \\ & DATE(t^p) - DATE(t^{p'}) < CC^{-} \wedge \\ & c^p = c^{p'} \wedge p \neq p') \\ 0, & otherwise \end{cases} \tag{17}$$

SC 3. Time constraints between different events in the same group

Group teaching is mechanism which several classes with the same subject are taught by a group of lecturer interchangeably. The time interval between two events for a group teaching should less than its minimum time constraint.

$$MinimizeV_8 = \sum_{p \in P} \sum_{p' \in P} f_{11}(p, p') \tag{18}$$

$$f_8(p, p') = \begin{cases} 1, & if(b_{c^p} = b_{c^{p'}} \wedge TGAP(t^p, t^{p'}) < GC^{-} \wedge \\ & l^p \in G^{b_{c^p}} \wedge l^{p'} \in G^{b_{c^{p'}}}) \\ 0, & otherwise \end{cases} \tag{19}$$

SC 4. Some lecturers should be scheduled in their time preferred time

The teaching event for a lecturer should be set in their preferred time.

$$MinimizeV_9 = \sum_{p \in P} f_9(p) \tag{20}$$

$$f_9(p) = \begin{cases} 1, & if(t^p \notin PREF_{l^p}) \\ 0, & otherwise \end{cases} \tag{21}$$

SC 5. Time constraints between events for a lecturer

The time interval between two events that a lecturer should not less than the minimum time constraint interval.

$$MinimizeV_{10} = \sum_{p \in P} \sum_{p' \in P} f_{10}(p, p') \tag{22}$$

$$f_{10}(p, p') = \begin{cases} 1, & if(l^p = l^{p'} \wedge p \neq p' \\ & \wedge TGAP(t^p, t^{p'}) < LC^{-}) \\ 0, & otherwise \end{cases} \tag{23}$$

SC 6. Student assignment spread

The events of a student should be set to a maximum SC^+ events in a day

$$MinimizeV_{11} = \sum_{z \in Z} \sum_{d \in D} f_8(z, d, P) \tag{24}$$

$$f_{11}(z, d, P) = \begin{cases} 1, & if(CNTSTIME(z, d, P) > SC^+) \\ 0, & otherwise \end{cases} \tag{25}$$

SC 7. Minimize students conflict
Minimize student which has been set in different room or class meeting in same time

$$MinimizeV_{12} = \sum_{z \in Z} \sum_{p \in P} \sum_{p' \in P} V_{12}(z, p, p') \tag{26}$$

$$f_{12}(z, p, p') = \begin{cases} 1, & if(z \in s^p \wedge z \in S^{p'} \wedge \\ & t^p = t^{p'} \wedge p \neq p') \\ 0, & otherwise \end{cases} \tag{27}$$

3 MDGA for UCTP

This section explains about how we encode the chromosome, formulate the problem definition in term of objective function division, and MDGA procedure.

3.1 Encoding

We use direct chromosome as the GA encoding. Direct chromosome mimics the real-world representation, which, in this case, is the university timetabling, as shown in Fig. 1. This timetable has R rooms and timeslots, which consist of 6 days multiplied by ten shifts (7 AM to 4 PM). This direct chromosome uses enumeration encoding, so the room is encoded as 1 to R for Room 1 to Room R. In the other hand, time is encoded as 1 for 7 AM Monday, 2 for 8 AM Monday, and 60 for 4 PM Saturday. As a result, the chromosome is shown in Fig. 2 as the encoding from timetable in Fig. 1.

Fig. 2 shows that a gene block consists of five parts (time, room, lecturer, class, and students). We count the individual length is equal to the number of events (gene blocks). Furthermore, because the search space is only the packet (time and room), the other parts (lecturer, class, and students) are fixed. So, programmatically, all GA operations (mutation and crossover) are only applied to a packet.

3.2 Problem Definition

The goal of this research is solving student sectioning UCTP. However, this problem, especially with a huge number of students, is almost impossible to do. The enormous student numbers will lead to extensive-time computation due to objective function evaluations.

This condition will be more problematic because we have to guarantee that HCs are always satisfied. If we limit HCs satisfaction strictly, the possible search spaces will also be limited. As a result, we cannot produce any satisfactory solutions. Thus, in this research, we introduce HCs satisfaction in the objective function, as shown in Equation 28, with a large weighting. We call this equation as shallow objective function.

Room R
Room 2
Room 1

TIME	MON	TUE	WED	THU	FRI	SAT
7AM	AR002-NPR		AL002-AAG		AR005-TBH	
8AM	DS002-RWJ	AL002-NPR			AR002-SWN	
9AM				NC002-RVI		
10AM	DS003-RWH	PL004-BBP	HC001-HTT	PL001-AMR		
11AM						
12AM	HC001-HTT		AR002-NPR			
1PM	IM001-JDN	AR002-JPY		IM001-JDN	AL001-AAG	
2PM					AR002-JPY	
3PM			AR005-TBH			
4PM	PL001-AMR					

Figure 1. University timetabling representation

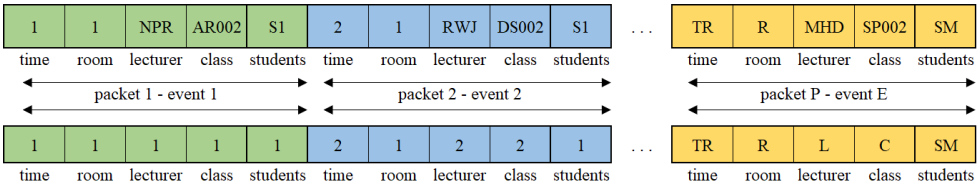

Figure 2. Directed chromosome and its encoding

$$\text{Minimize } V_{HC} = \sum_{i=1}^{5} V_i(P) \tag{28}$$

The shallow objective function focuses in maintaining HCs. The deeper objective function which are medium objective function and deep objective function focus in solving SCs and respecting the HCs as well. The medium objective function focuses in solving class-level SCs with the objective function as shown in Equation 29. The deep objective function focuses in solving student-level SCs with the objective function as shown in Equation 30.

$$\text{Minimize } V_{class} = \sum_{i=1}^{10} V_i(P) \tag{29}$$

$$\text{Minimize } V_{student} = \sum_{i=1}^{12} V_i(P) \tag{30}$$

3.3 MDGA Procedure

Basically, MDGA implements HCs satisfaction to generate initial population with greedy initialization. So, the initial population will has small number of HC violations. In other hands, the SCs satisfaction is applied in the core of GA procedure. Algorithm 1 shows that for each generation t, we perform GA steps such as elitism, selection, crossover, and mutation. Only in the first generation, we make an initial population of P by using greedy initialization with $PopSize$ as a number of individuals. For each generation, we create an empty population of P' to be the new population and save elite individuals.

We implement elitism mechanism to maintain elite individuals among the population. We put the best individual of P into P'. Furthermore, we also put the best M individuals of P into P'. So, the number of elite individuals is $1 + M$.

We use roulette wheel selection to select two individuals idv_1 and idv_2 as parents. By using a roulette wheel, we can choose the parents fairly based-on their evaluation — we crossover these selected parents with crossover probability P_c to produce offsprings then they are evaluated. In this step, we use class-level evaluation. We pick the best two individuals among parents and offsprings, to be the new parents.

We divide the mutation into two stages to solve the problem gradually. We mutate the new parents (stage 1 mutation) with mutation probability P_m to produce offsprings then they are evaluated. This stage use class-level evaluation. We pick the best two individuals among parents and offsprings, to be the new parents.

After the population size equals $PopSize$, we continue to stage 2 mutation with a mutation probability of P_m. Stage 2 mutation is the same as stage 1 mutation, albeit the evaluation. Stage 2 uses student-level evaluation for parents and offsprings. We pick the best two individuals among parents and offsprings, to be the new parents. We put the last parents into P'.

The process is repeated from selection until mutation until the number of individuals in P' equals to $PopSize$. After that, the population P' replaces P and proceed to the next generation.

3.3.1 Crossover

We use a multi-point crossover with the number of affected genes is N_c of all genes which violate constraints. The crossover follows these steps:

1. take two individuals from the selection as parents
2. select N_c of all events which violate constraint in the first parent
3. select an event of them
4. select an event randomly from the second parent which have same room capacity with selected event from first parent regardless of the violation
5. swap the selected event of first parent with second parent
6. repeat step 3-5 until all selected events from first parent are swapped

Algorithm 1 GA Procedure

Require: GA in generation t
  if $t = 1$ **then**
    Greedy Initialization population $P[PopSize]$
  end if
  $P' = \emptyset$ *//empty population*
  // elitism
  Put the best individual of P into P'
  Put the best M individuals of P into P'
  $count = 1 + M$
  while $count < PopSize$ **do**
    // selection
    Select idv_1 and idv_2 from P with roulette wheel
    // crossover
    Crossover idv_1 and idv_2 with probability P_c
    $[idv_1, idv_2]$ = best 2 individuals of parents&offsprings
    // stage 1 mutation
    Mutate idv_1 and idv_2 with probability P_m
    $[idv_1, idv_2]$ = best 2 individuals of parents&offsprings
    $count = count + 2$
  end while
  // stage 2 mutation
  Select idv_1 and idv_2 from P with roulette wheel
  Mutate idv_1 and idv_2 with probability P_m
  $[idv_1, idv_2]$ = best 2 individuals of parents&offsprings
  Put idv_1 and idv_2 into P'
  Replace P with P'

3.3.2 Mutation

We use three mutation steps to improve the probability of producing better offsprings. These three mutations are namely M1 (moving), M2 (swapping), and M3 (comparing). All mutations are always executed sequentially for each individual. The number of affected genes for mutation is N_m of all genes which violate the constraints.

- **M1 (Moving)**
  Select an event which violates the constraint. Move this event to an unused packet (see Eq. 1). The unused packet is a packet which has not been taken by an event. The target packet is selected from the list of unused packets with appropriate room capacity. M1 (Moving) is illustrated in Fig. 3.

- **M2 (Swapping)**
  Select an event which violates the constraint as the first event. Find other events which have the same subject with the first event. Select an event as a target event randomly from them. Swap the first event with the target event. If the swap decreases violations, keep the new individual, otherwise cancel the swap. M2 (Swapping) is illustrated in Fig. 4.

- **M3 (Comparing)**
  Select an event which violates the constraint. Select randomly two other events that have same room capacity regardless of the violation. Swap the violated event with the one

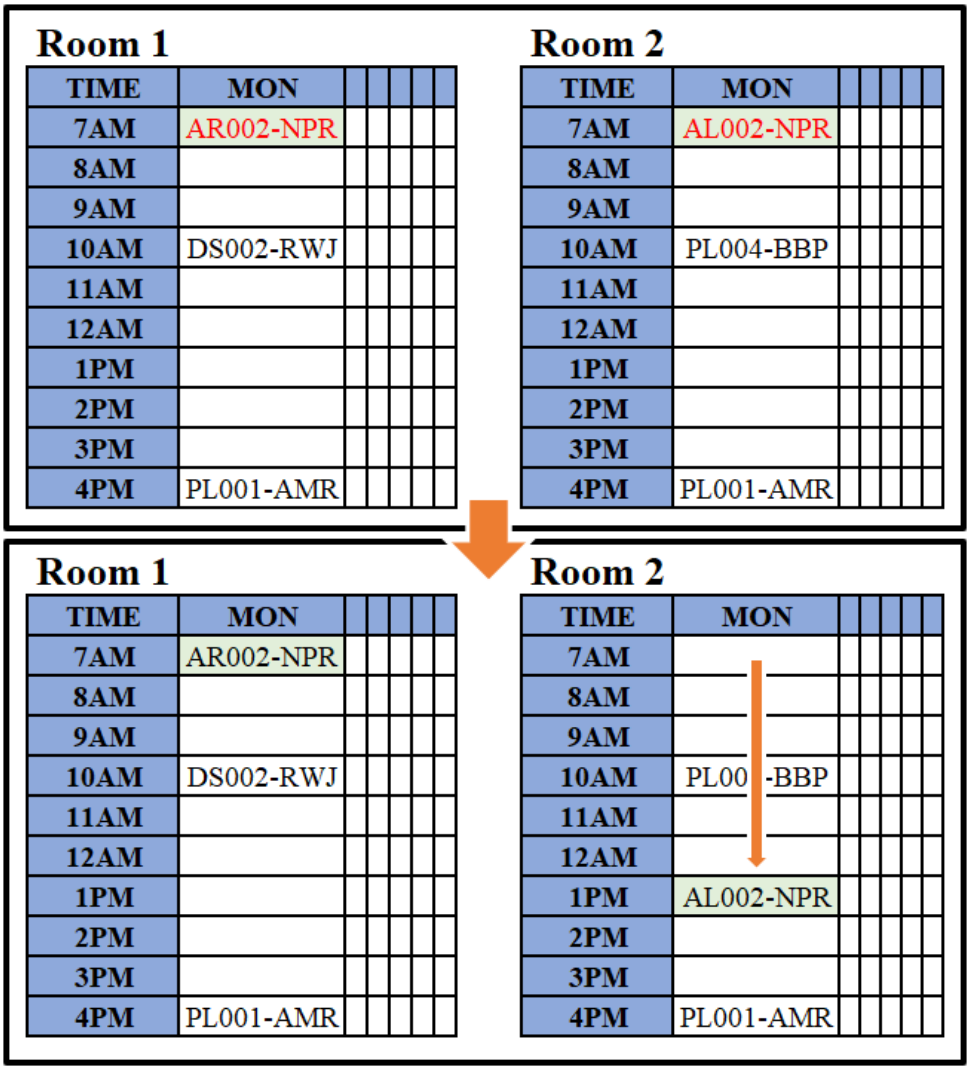

Figure 3. M1 (Moving)

which produces lower violations. If the new individual decreases violations, keep the new individual, otherwise cancel the swap. M3 (Comparing) is illustrated in Fig. 5.

4 Experimental Result

We perform the experiments to analyze MDGA performance in handling student sectioning UCTP. We also compare our proposed solution with other solvers to solve class-level UCTP benchmarks.

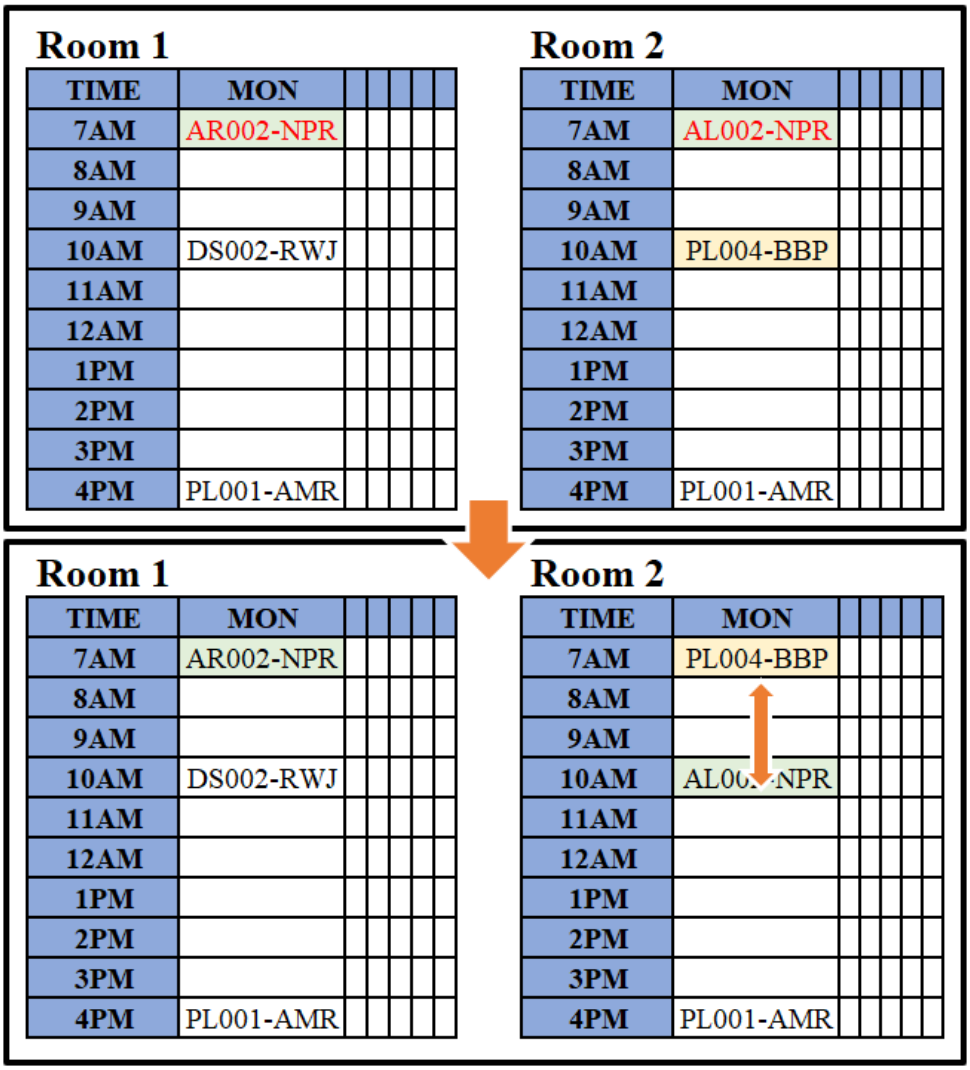

Figure 4. M2 (Swapping)

4.1 Parameter Settings

We set the weight of HCs far larger than SCs. We set the HC weight with the big number M, which $M = 1000$ programmatically. As a result, MGA will prioritize poor fitness caused by HCs. The SCs become the focus after all HCs have been satisfied. We set the penalty value of SCs as proportional to its influence. Regarding this consideration, the SCs penalty value configuration is presented in Table 2.

We set the GA parameters by considering previous research [2]. The GA parameter configurations are mutation probability $P_m = 0.1$ with number of mutated genes $N_m = 10\%$, crossover probability $P_c = 0.8$ with number of crossovered genes $N_c = 10\%$, maximum generation $MaxGen = 200$, and population size $PopSize = 30$.

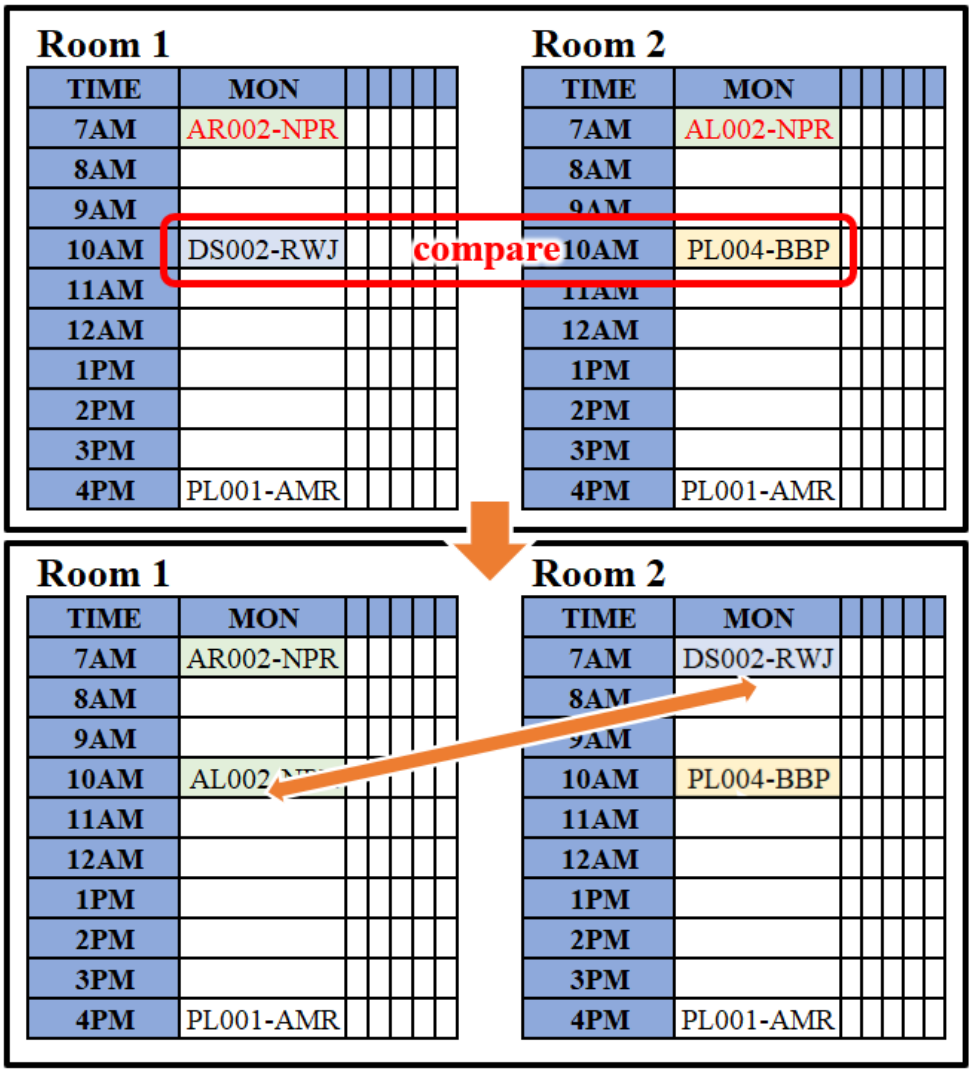

Figure 5. M3 (Comparing)

Table 2. Soft Constraint Penalty Configuration

Soft Constraint	Value
$SC1, SC2$ - high lecturer and class SCs	50
$SC3$ - group teaching SC	5
$SC4, SC5$ - low lecturer SCs	20
$SC6, SC7$ - student SCs	1

4.2 Dataset

Dataset used in this research was Telkom University odd/even semester for 2011/12 (before merging) and 2016/17 (after merging) enrollment years. To be specific, the student body at

Telkom University has increased from 6,570 students in 2011 to 23,451 in 2016. This number is a result of the merging of four universities. The detail dataset characteristics comparison is shown in Table 3.

Table 3. Telkom UCTP Characteristics

No	Attributes	2011/12	2016/17
1	Rooms	80	562
2	Classes (avg. per semester)	813	5309.25
3	Average number of meetings per class	2.75	2.62
4	Lecturers	316	1470
5	Average classes per lecturers	2.58	4.78
6	Students (avg. per semester)	6570	23451
7	Average number of classes per students	6.48	5.03

4.3 Experiment 1 - Proof of Concept

The first experiment implemented MDGA for Telkom UCTP 2011/12 as well as 2016/17 enrollment years. We observed its performance based on the best and average fitness values in 5 runs. For additional insight, we include the HC violation percentages. Table 4 shows that MDGA could yield an acceptable fitness value for 2011/12 as well as 2016/17 enrollment years. MDGA could achieve a small violation percentage timetabling, which means we can accept these results as a feasible timetable.

Table 4. MDGA Result for Telkom UCTP

Problem	Fitness		Violation %	
	Best	Average	Best	Average
2011/12	4540	4887	0.16%	0.18%
2016/17	96562	97023.43	7.36%	7.43%

4.4 Experiment 2 - Comparison Analysis

The third experiment put MDGA together with Standard Genetic Algorithm (GA) [6], Asynchronous Island Model Genetic Algorithm (AIMGA) as a previous solver for Telkom UCTP [2], and UniTime [12] as a current generic UCTP solver. The parameter configuration and chromosome structure of GA and AIMGA were the same with MDGA. The comparison details are shown in the following Table 5.

According to [12], UniTime has different constraint configuration format with Telkom UCTP. Therefore, we conducted a constraints mapping from Telkom UCTP into UniTime format (v2.3), which is explained in Table 6. Where **SAME_ROOM** means given classes must be taught in the same room, **SPREAD** means given classes have to be spread in time (overlapping of the classes in time needs to be minimized), **NHB_GTE** means given classes must have 1 hour or more in between, **NHB_LT** means given classes must have less than 6 hours from the end of first-class to the beginning of the next, and **NHB** means given classes must have exactly x hours in between the end of one and the beginning of another.

Table 5. MDGA, GA, and AIMGA Configuration Comparison

Differences	Algorithm		
	MDGA	**GA**	**AIMGA**
Chromosome encoding	direct encoding	direct encoding	direct encoding
Mutation	directed mutation	random mutation	directed mutation
Crossover	directed multi-point	random multi-point	random multi-point
Elitism	implemented	implemented	implemented
Island model	no	no	yes
GA core	three level GA	standard GA	informed GA

Table 6. Constraints Mapping from Telkom UCTP into UniTime

Telkom UCTP	UniTime
HC1	embedded in solver
HC2	embedded in solver
HC3	strictly supported in data format, SAME_ROOM
HC4	strictly supported in data format
HC5	implicitly supported in related class constraints
SC1	share same constraint with HC5
SC2	SPREAD
SC3	NHB_GTE, NHB_LT, NHB
SC4	share same constraint with HC5
SC5	NHB_GTE, NHB_LT, NHB
SC6	implicitly supported in related class constraints
SC7	embedded in solver

Table 7 shows the average of violation percentage comparison of MDGA, GA, AIMGA, and UniTime in 5 runs. The unfeasible value in UniTime cell for Telkom UCTP 2016/17 enrollment year means it could not get a result in a reasonable time (6 hours runtime limit exceeded). Besides, this table points out that MDGA could lead other algorithm results for both problems.

Table 7. MDGA Violation Percentage Comparison

Problem	MDGA	GA	AIMGA	UniTime
2011/12	1.87%	2.72%	2.39%	13.85%
2016/17	18.34%	57.34%	25.54%	unfeasible

4.5 Experiment 4 - Benchmark Analysis

This last experiment compares the MDGA performance with several UCTP solutions by using the International Timetabling Competition (ITC) 2007 benchmark datasets[13]. Table 8

shows the problem specification of this dataset. There are 24 test cases with various combination of events, rooms, features, and students. We only used the Track 3 ITC curriculum-based course timetabling. This problem is only general UCTP without student sectioning, so we must modify our algorithm to meet this requirement by the process only the Stage 1 of Directed GA (Fig. 1, Stage 1).

Table 8. Complete specification of ITC-2007 dataset

Problem	#Events	#Rooms	#Features	#Students	Max. students per event	Max. events per students
ITC-1	400	10	10	500	33	25
ITC-2	400	10	10	500	32	24
ITC-3	200	20	10	1000	98	15
ITC-4	200	20	10	1000	82	15
ITC-5	400	20	20	300	19	23
ITC-6	400	20	20	300	20	24
ITC-7	200	20	20	500	43	15
ITC-8	200	20	20	500	39	15
ITC-9	400	10	20	500	34	24
ITC-10	400	10	20	500	32	23
ITC-11	200	10	10	1000	88	15
ITC-12	200	10	10	1000	81	15
ITC-13	400	20	10	300	20	24
ITC-14	400	20	10	300	20	24
ITC-15	200	10	20	500	41	15
ITC-16	200	10	20	500	40	15
ITC-17	100	10	10	500	195	23
ITC-18	200	10	10	500	65	23
ITC-19	300	10	10	1000	55	14
ITC-20	400	10	10	1000	40	15
ITC-21	500	20	20	300	16	23
ITC-22	600	20	20	500	22	25
ITC-23	400	20	30	1000	69	24
ITC-24	400	20	30	1000	41	15

Table 9 displays the benchmark experimental result of ITC-2007 dataset. The Table values show the fewest violation result of each solutions. We compare MDGA with several current UCTP solutions, such as **CBS:** Constraint Based Solver by Muller[5], **TSA:** Tabu Search Approach by Lu&Hao[14], **CSP:** Constraint Satisfaction Problem by Atsuta[15], **TAM:** Threshold Acceptance Metaheuristic by Geiger[16], **RBT:** Repair Based TimeTable Solver by Clark[17], **ATS:** Adaptive Tabu Search by Lu&Hao[18], **HMA:** A Hybrid Metaheuristic Approach by Salwani Abdullah[19], **ITS-LS:** Incorporating Tabu Search and Local Search by Atsuta et al.[15], **GDA:** Great Deluge Algorithm with Kempe Chain by McCollum et al.[20], **ILS:** Iterative Local Search by Soria-Alcaraz et al.[21], **HGATS:** The Hybrid Approach Hybrid Genetic Algorithm and Tabu search by Jat&Yang[22], **MMA:** Mixed Metaheuristic Approach by Cambazard et al.[23], **CTI:** Combination of a General Purpose Constraint Satisfaction Solver, Tabu Search and Iterative Local Search Techniques by Atsuta et al.[24], **HA:** A Hybrid Algorithm by Chiarandini et al.[13], and **ACO:** Ant Colony Optimization algorithm in Conjunction with A Iterative Local search by Nothegger et al.[25].

These results are extracted from each paper or a review paper in UCTP by Babaei et al. [26]. Similar to the review paper, we only compare the violation numbers because, in general practice of university timetabling, the computational time is usually not the primary

consideration. That is because a university is usually required to make a timetable once in a semester. So the time limit might be around a few days in the end or beginning of a semester.

Table 9 shows that MDGA could get the fewest violations for 6 of 24 test cases. This result proves MDGA could compete with other UCTP solvers albeit not the best one for the ITC-2007 benchmark dataset.

Table 9. Violation Numbers of All Solvers for ITC-2007 dataset

Problem	CBS	TSA	CSP	TAM	RBT	ATS	HMA	ITS-LS	GDA	ILS	HGATS	MMA
ITC-1	**5**	**5**	**5**	**5**	10	**5**	**5**	**5**	**5**	**5**	523	571
ITC-2	51	55	50	111	111	**34**	39	50	60	48	342	993
ITC-3	84	71	82	128	119	**70**	76	82	81	76	379	164
ITC-4	37	43	**35**	72	72	38	**35**	**35**	39	41	234	310
ITC-5	330	309	312	410	426	298	315	312	31	303	**0**	5
ITC-6	48	53	69	100	130	47	50	69	45	54	**0**	**0**
ITC-7	20	28	42	57	110	19	12	42	21	25	**0**	6
ITC-8	41	49	40	77	83	43	37	40	41	47	**0**	**0**
ITC-9	109	105	110	150	139	99	104	110	102	106	1102	1560
ITC-10	16	4	27	71	85	16	10	9	17	23	515	2163
ITC-11	**0**	**0**	**0**	**0**	3	**0**	**0**	**0**	**0**	**0**	246	178
ITC-12	333	343	351	442	**4.8**	320	337	351	349	324	241	146
ITC-13	66	73	68	622	113	65	61	68	43	68	**0**	**0**
ITC-14	59	57	59	90	84	52	53	59	59	53	**0**	1
ITC-15	84	71	82	128	119	69	73	82	82	74	**0**	**0**
ITC-16	34	39	40	81	84	38	32	40	49	42	**0**	2
ITC-17	83	91	102	124	152	80	72	102	81	81	**0**	**0**
ITC-18	83	96	68	116	110	67	77	68	79	69	**0**	**0**
ITC-19	62	65	75	107	111	59	60	75	67	65	121	1824
ITC-20	27	47	61	88	144	35	**22**	61	30	35	304	445
ITC-21	103	106	123	174	169	105	95	123	110	106	36	**0**
ITC-22	–	–	–	–	–	–	–	–	–	–	1154	29
ITC-23	–	–	–	–	–	–	–	–	–	–	963	**238**
ITC-24	–	–	–	–	–	–	–	–	–	–	274	**21**

5 Conclusions

This research shows that the MDGA could overcome not only Telkom UCTP 2011/12 (before merging) but also 2016/17 (after merging) enrollment year with acceptable accuracy represented by the fitness function. For both problems, this proposed approach yielded small violation percentages for all constraints. This result shows that MDGA could handle scaling UCTP well and produce a feasible timetable.

Furthermore, from the second experiment, we could conclude that MDGA shows better performance compare with previous Telkom UCTP solver and generic UCTP solver. The final experiment shows that MDGA could compete with other UCTP solvers albeit not the best one for the ITC-2007 benchmark dataset.

Finally, this research confirms that MDGA can solve the student sectioning Telkom UCTP with an acceptable result. However, further studies still need to be conducted for MDGA to the other UCTP benchmark. A more in-depth investigation for diversity preservation is also needed. Moreover, MDGA implementation in island model GA such as DM-LIMGA [27] is also encouraged.

Acknowledgments

Indonesia Endowment Fund for Education (LPDP), a scholarship from the Ministry of Finance, Republic of Indonesia, supports this work. We conduct this research while at the Graduate School of Information, Production, and Systems, Waseda University, Japan.

References

[1] M. Garey, D.S. Johnson, *Computer and Intractability* (W.H. Freeman and Company, New York, 1979)

[2] A.A. Gozali, J. Tirtawangsa, T.A. Basuki, *Asynchronous Island Model Genetic Algorithm for University Course Timetabling*, in *Proceedings of the 10th International Conference on the Practice and Theory of Automated Timetabling* (PATAT, 2014), pp. 179–187, ISBN 978-0-9929984-0-0

[3] K. Murray, T. Muller, H. Rudova, in *Practice and Theory of Automated Timetabling VI*, edited by E.K. Burke, H. Rudová (Springer Berlin Heidelberg, 2006), Number 3867 in Lecture Notes in Computer Science, pp. 189–209, ISBN 978-3-540-77344-3 978-3-540-77345-0, dOI: 10.1007/978-3-540-77345-0_13, http://link.springer.com/chapter/10.1007/978-3-540-77345-0_13

[4] M.W. Carter, *A Comprehensive Course Timetabling and Student Scheduling System at the University of Waterloo*, in *Practice and Theory of Automated Timetabling III: Third International Conference, PATAT 2000 Konstanz, Germany, August 16–18, 2000 Selected Papers*, edited by E. Burke, W. Erben (Springer Berlin Heidelberg, Berlin, Heidelberg, 2001), pp. 64–82, ISBN 978-3-540-44629-3, http://dx.doi.org/10.1007/3-540-44629-X_5

[5] T. Muller, K. Murray, Annals of Operations Research **181**, 249 (2010)

[6] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st edn. (Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989), ISBN 0201157675

[7] J. Abela, *A Parallel Genetic Algorithm for Solving the School Timetabling Problem*, in *Division of Information Technology, CSIRO* (Citeseer, 1991)

[8] K. Banczyk, T. Boinski, H. Krawczyk, *Parallelisation of Genetic Algorithms for Solving University Timetabling Problems*, in *International Symposium on Parallel Computing in Electrical Engineering (PARELEC'06)* (2006), pp. 325–330

[9] D. Corne, P. Ross, H.L. Fang, *Fast practical evolutionary timetabling*, in *AISB Workshop on Evolutionary Computing* (Springer, 1994), pp. 250–263

[10] Suyanto, Artificial Intelligence Soft Computing Lecture Notes of Computer Science, Springer Berlin Heidelberg **6114**, 229 (2010)

[11] F. Titel, K. Belarbi, International Journal of Electrical and Computer Engineering **7**, 2614 2626 (2017)

[12] T. Muller, *University Course Timetabling: Solver Evolution*, in *Proceedings of the 11th international conference on the Practice And Theory of Automated Timetabling, 2016* (2016)

[13] L.D. Gaspero, B.M. Mccollum, A.S. Schaerf, *The Second International Timetabling Competition (ITC-2007): Curriculum-based Course Timetabling (Track 3)*, in *Proceedings of the 1st International Workshop on Scheduling a Scheduling Competition (SSC 2007)* (2007)

[14] G. Lu, S. Areibi, *An Island-Based GA Implementation for VLSI Standard-Cell Placement*, in *Genetic and Evolutionary Computation – GECCO 2004: Genetic and Evo-*

lutionary Computation Conference, Seattle, WA, USA, June 26-30, 2004. Proceedings, Part II, edited by K. Deb (Springer Berlin Heidelberg, Berlin, Heidelberg, 2004), pp. 1138–1150, ISBN 978-3-540-24855-2, http://dx.doi.org/10.1007/978-3-540-24855-2_123

[15] M. Atsuta, K. Nonobe, T. Ibaraki, *ITC2007 Track 2: An Approach using general CSP solver*, in *Proceedings of the Practice and Theory of Automated Timetabling (PATAT 2008)* (2008)

[16] M.J. Geiger, Annals of Operations Research **194**, 189 (2010)

[17] M. Clark, M. Henz, B. Love, *QuikFix A Repair-based Timetable Solver*, in *Proceedings of the Practice and Theory of Automated Timetabling (PATAT 2008)* (2008)

[18] Z. Lü, J.K. Hao, European Journal of Operational Research **200**, 235 (2010)

[19] S. Abdullah, H. Turabieh, B. McCollum, P. McMullan, Journal of Heuristics **18**, 1 (2010)

[20] B. McCollum, P. McMullan, A.J. Parkes, E. Burke, S. Abdullah, *An Extended Great Deluge Approach to the Examination Timetabling Problem*, in *Proceeding of the Multidisciplinary International Scheduling Conference (MISTA) 2009* (2009)

[21] J.A. Soria-Alcaraz, E. Özcan, J. Swan, G. Kendall, J.M.C. Valadez, Appl. Soft Comput. **40**, 581 (2016)

[22] S.N. Jat, S. Yang, Journal of Scheduling **14**, 617 (2010)

[23] H. Cambazard, E. Hebrard, B. O'Sullivan, A. Papadopoulos, Annals of Operations Research **194**, 111 (2010)

[24] B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A.J. Parkes, L.D. Gaspero, R. Qu, E.K. Burke, INFORMS Journal on Computing **22**, 120 (2010)

[25] C. Nothegger, A. Mayer, A. Chwatal, G.R. Raidl, Annals of Operations Research **194**, 325 (2012)

[26] H. Babaei, J. Karimpour, A. Hadidi, Computers & Industrial Engineering **86**, 43 (2015)

[27] A.A. Gozali, S. Fujimura, Evolutionary Intelligence (2019)