IEEJ TRANSACTIONS ON ELECTRICAL AND ELECTRONIC ENGINEERING

OFFICIAL JOURNAL OF THE INSTITUTE OF ELECTRICAL ENGINEERS OF JAPAN

VOLUME 15, NO. 3, MARCH 2020

電気学会

SCI registered

WILEY
ISSN 1931-4973

INSTITUTE OF ELECTRICAL ENGINEERS OF JAPAN

IEEJ

TRANSACTIONS ON ELECTRICAL AND ELECTRONIC ENGINEERING
OFFICIAL JOURNAL OF THE INSTITUTE OF ELECTRICAL ENGINEERS OF JAPAN

For submission instructions, subscription, and all the latest information, visit: https://onlinelibrary.wiley.com/journal/tee

IEEJ

TRANSACTIONS ON ELECTRICAL AND ELECTRONIC ENGINEERING

TABLE OF CONTENTS

VOL.15, NO.3, 2020

INVITED REVIEW PAPER

PAPERS

Continued overleaf

IEEJ TRANSACTIONS ON ELECTRICAL AND ELECTRONIC ENGINEERING
IEEJ Trans 2019
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI:10.1002/tee.23067

Paper

Solving University Course Timetabling Problem Using Localized Island Model Genetic Algorithm with Dual Dynamic Migration Policy

Alfian A. Gozali*[a], Non-member
Bobby Kurniawan*, Non-member
Wei Weng**, Non-member
Shigeru Fujimura*, Senior Member

The University Course Timetabling Problem (UCTP) is a scheduling problem of assigning a teaching event in a certain time and room by considering the constraints of university stakeholders such as students, lecturers, and departments. This problem becomes complicated for universities with a large 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 also implies the complexity of constraints, which is larger accordingly. However, current and generic solvers have failed to meet the scalability and reliability requirements for student sectioning UCTP. In this paper, we introduce the localized island model genetic algorithm with dual dynamic migration policy (DM-LIMGA) to solve student sectioning UCTP. Our research shows that DM-LIMGA can produce a feasible timetable for the student sectioning problem and get better results than previous works and the current UCTP solver. Our proposed solution also consistently yield lower violation number than other algorithms, as evidenced by UCTP benchmark experiment results.

Keywords: University course timetabling problem; island model genetic algorithm; localization strategy; migration policy

Received 3 October 2018; Accepted 8 October 2019

1. Introduction

The University Course Timetabling Problem (UCTP) is a scheduling problem of assigning a teaching event in a 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]. However, some universities, such as Telkom University [2] and Purdue University [3], can have a large number of students and classes. This condition increases the problem complexity because the search space also increases. The constraint number, which also becomes larger, makes the problem even more complicated.

Moreover, several universities such as Telkom University [2] 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 student's preference is encouraged as well.

In regular timetabling, we place student enrollment (the process by which the students choose their classes) after the class timetable becomes available. Contrarily, in student sectioning, students choose a set of preferred classes first, and then the system will create a timetable based on their preferences. Thus, student sectioning significantly increases the problem complexity. As a result, the number of search spaces grows enormously, due to the increase in the number of students, other variables, and involvement of their constraints.

For example, a university such as Telkom University can have a significant increase of its stakeholders. The number of students at Telkom University has increased from 6570 in 2011 to 23 451 in 2016. This number is a result of merging four universities: Telkom Institute of Technology, Telkom Polytechnic, Telkom Institute of Management, and Telkom School of Arts. As a result, the UCTP solver also must cover the scalability requirement. Scalability is the ability of a computing process to be used in a various ranges of capabilities.

UCTP is a minimizing optimization problem, so the objective is to minimize all the predefined constraint violations for each of the teaching events. Accordingly, there are several approaches attempting to solve this complex problem, such as the constraint satisfaction problem [5], local search [6–8], Tabu search [9,10], ant colony algorithm [11], and hybrid algorithms [12–17]. Therefore, we need a new solution that supports problem scalability and gives a feasible timetable at the same time. Hence, this paper introduces the localized island model genetic algorithm with dual dynamic migration policy (DM-LIMGA).

DM-LIMGA implements localization strategy island model GA (LIMGA), which has solved theoretical case studies [18,19] with various complexities. Furthermore, the adaptation of the dual dynamic migration policy (DDMP) is used to maintain the population diversity in LIMGA better [20]. Finally, the combination of LIMGA and DDMP in DM-LIMGA will have a significant advantage for overcoming student sectioning UCTP. Furthermore, UCTP is a real-world problem which is very complicated. So, DM-LIMGA needs modification in terms of problem formulation, encoding, and slave islands.

[a] Correspondence to: Alfian A. Gozali. E-mail: alfian@tass.telkomuniversity.ac.id

*Graduate School of Information, Production, and Systems, Waseda University, 2-7 Hibikino, Wakamatsu, Kitakyushu, Fukuoka, 808-0135, Japan
**Institute of Liberal Arts and Science, Kanazawa University, Kanazawa, Japan

Taken together, the primary motivation of this work is how to modify DM-LIMGA for student sectioning UCTP. In detail, the main goals of this research are (i) to formalize a real-world scaling student sectioning UCTP, (ii) to modify DM-LIMGA to meet the problem requirements, and (iii) to analyze DM-LIMGA's performance in handling student sectioning UCTP in the terms of violation number, scalability, and reliability.

This paper consists of six sections. We organize the remainder of this paper as follows. Section 2 talks about the student sectioning UCTP. Section 3 explains the DM-LIMGA concept in detail. Section 4 gives the DM-LIMGA design to meet UCTP problems. Section 5 shows how we conducted the experiments, results, and analysis. Section 6 includes 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 over weeks, days, hours, or even minutes. For example, a university could implement 40 min for a time, while another university could have 60 min. Rooms are UCTP resources that can vary in capacity, facility, ownership, and specialization (e.g., theory and practice classroom). The notations in Table I. give 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:

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

Following previous research [2,21], this work also uses two types of constraints: hard and soft constraint (SCs). Hard constraint (HC) is a constraint that must be satisfied. SC is better to be fulfilled to improve the quality of the timetable.

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

2.1. HC 1: No conflict of lecturers

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

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

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

2.2. HC 2: No conflict of classes

There is no packet that has been set for different events at the same time.

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

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

2.3. HC 3: Any event should be scheduled in a suitable capacity room

No event has been set in a room with a less than suitable capacity.

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

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

2.4. 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.

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

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

2.5. 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.

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

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

2.6. SC 1: Lecturer assignment spread

The teaching event for a lecturer should be set to a maximum of LC^+ events in a day.

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

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

2.7. 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 in which a class can be conducted for more than once in a week. For example, class AR002 must be taught twice a week. Thus, it is possible to have several similar events mapped into different packets. These similar events are interchangeable, which is shown by the following notation:

$$\text{Minimize } V_7 = \sum_{p \in P} \sum_{p' \in P} f_7(p, p') \tag{16}$$

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

Table I. UCTP formulation

Indices and sets			
$p \in P$	Set of pairs		
$e \in E$	Set of teaching events	e^p	Event of pair p
$q \in Q$	Set of packets	q^p	Packet of pair p
$l \in L$	Set of lecturers	l^p	Lecturer of pair p
$b \in B$	Set of subjects	b_c	Subject of class c
$G^b \subseteq L$	Set of group teaching lecturers for subject b		
$k \in K \subseteq L$	Set of special lecturers		
$r \in R$	Set of rooms	r^p	Room of pair p
$t \in T$	Set of time	t^p	Time of pair p
$c \in C$	Set of classes	c^p	Class of pair p
$d \in D$	Set of days	d^p	Day of pair p
$z \in Z$	Set of all students		
$s \in S \subseteq Z$	Set of students	S^p	Students of pair p
Functions			
$CNTTIME\ (l,d,P)$	Count event of l in a day		
$DATE(t)$	Return the date of time t		
$CNTSTIME\ (z,d,P)$	Count event of z in a day		
$TGAP\ (t,t^{'})$	Time interval of t and $t^{'}$		
Parameters			
w_i	Weighting of constraint i		
V_i	Total violation of constraint i		
CAP_r	Capacity of room r		
CAP_e^-	Minimum room capacity of event e		
X_l	Prohibited time of lecturer l		
$PREF_l$	Preference time of lecturer l		
Constant			
LC^+	Maximum lecturer event in a day		
LC^-	Minimum time interval between two events of a lecturer		
CC^-	Minimum class event interval in a week		
SC^+	Maximum student event in a day		
GC^-	Minimum group teaching event interval		

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

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

$$\text{Minimize } V_8 = \sum_{p \in P} \sum_{p' \in P} f_{11}(p,p') \tag{18}$$

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

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

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

$$\text{Minimize } V_9 = \sum_{p \in P} f_9(p) \tag{20}$$

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

2.10. SC 5: Time constraints between events for a lecturer

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

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

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

2.11. SC 6: Student assignment spread

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

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

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

2.12. SC 7: Minimize student conflict

Minimize students who have been set in different rooms or classes meeting at the same time

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

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

3. DM-LIMGA

Island's independent processing and migration policy is the main discussion in island model topics. That is because an improvement in these topics can produce a higher diversity, which leads to

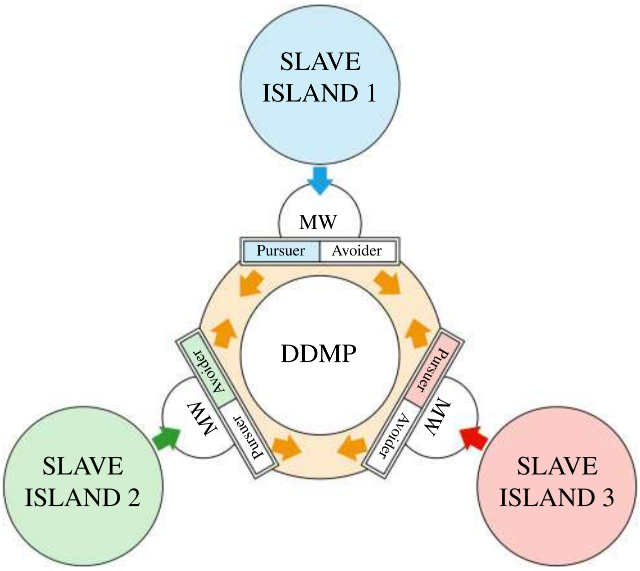


Fig. 1. DM-LIMGA mechanism

a better result. Accordingly, previous research designed DM-LIMGA as a mechanism which is the combination of improvement in these topics [22], which combines LIMGA as the island's independent processing and DDMP as the migration policy.

LIMGA is the island's independent processing approach. It sees an island as a single living environment of its population [18]. As the implication, each island configuration can be the value of its parameters, or even its core algorithm might be different. This difference could branch into separate evolution paths, which can be its speed or chromosome pattern. An island may evolve more quickly or more efficiently to produce a better individual than other islands.

Different living environments must incline toward specific goals. From previous research [18], we classified them into standard, speed-based, and performance-based GA. Speed-based GA is a GA variant that tends to patch up its computational speed. It tries to get a good result as fast as possible. On the contrary, performance-based GA tries to get a better result in every generation even though it takes more time than the others.

Figure 1 represents the combination mechanism of DDMP and localization strategy of DM-LIMGA. Every a slave island accomplishes a generation; it puts the current best individual in its migrant window (MW). MW is a buffer placed in the master island to keep the best individual from each slave island.

In DDMP, island has two states: *pursuer* and *avoider* [20]. The island has a chance to be a *pursuer* or an *avoider* depending on the current condition. If the island has a diversity level (represented by its bias value) less than the threshold, then the island state will be a *pursuer*. On the other hand, if there is an island that has diversity level more than or equal to a threshold, then the island state will be an *avoider*.

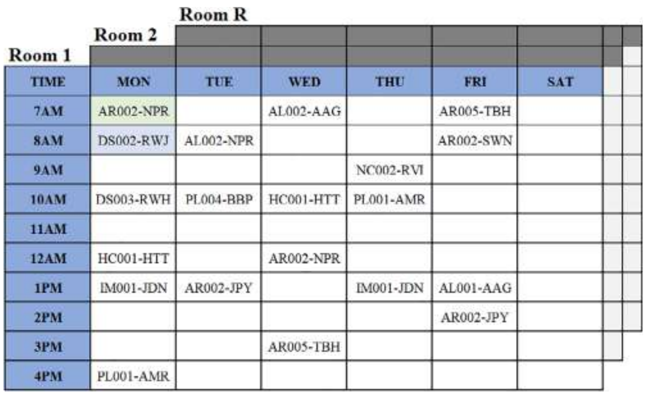
Each slave executes a different GA procedure. At the end of every generation t of slave island i, the best individual P_i^t is sent as a migrant to the master Island. Then, island i takes a migrant from another island based on the DDMP algorithm, as shown in Fig. 2. If there is any individual except this migrant in MW, the master island compares the bias value B_i^t of the original island i with the predefined threshold θ. If it is more than or equal to θ, the master fsland finds an individual in MW that has the furthest Hamming distance δ from this migrant. Finally, the master island migrates it from MW to the island i.

On the other hand, if the bias value is less than θ, the master island finds an individual in MW that has the largest attractiveness α. Finally, the master island migrates it to island i. The formulation

Algorithm 1 DDMP Algorithm

Require: Island i finished Generation t
  Island i sends the best individual P_i^t to MW_i
  if there is any individual in $MW_{i'}$ where $i' \neq i$ **then**
    if $B_i^t \geq \theta$ **then**
      $//avoider$
      Find individual in $MW_{i'}$ where $i' \neq i$ which has the furthest δ from P_i^t
      Migrate it to island i
    else
      $//pursuer$
      Find individual in $MW_{i'}$ where $i' \neq i$ which has the biggest α
      Migrate it to island i
    end if
  end if

Fig. 2. Dual dynamic migration policy algorithm



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					

Fig. 3. University timetabling representation

of the bias value, Hamming distance, and attractiveness will be discussed later.

4. DM-LIMGA for UCTP

Previous work [22] designed DM-LIMGA to solve a theoretical single-objective optimization problem. It used simple numerical encoding with general GA, pseudo-GA (PGA), and informed GA (IGA) as its slave islands. However, modification in terms of encoding and slave islands is needed.

4.1. 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. 3. This timetable has R rooms and timeslots, which consist of 6 days multiplied by 10 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. On 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. 4 as the encoding from timetable in Fig. 3.

Figure 4 shows that a gene block consists of five parts (time, room, lecturer, class, and students). We count the individual length as 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.

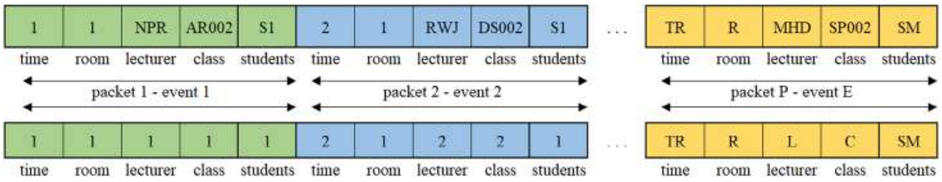


Fig. 4. Directed chromosome and its encoding

4.2. Problem definition The goal of this work is to solve student sectioning UCTP. However, this problem, especially with a large number of students, is almost impossible to solve. The large student number 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 HC satisfaction strictly, the possible search spaces will also be limited. As a result, we cannot produce any satisfactory solutions. Thus, in this work, we introduce HC satisfaction in the objective function, as shown in (28), with a large weighting.

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

The introduction of HCs in objective function means there will be no guarantee that HCs are always satisfied. Therefore, we dedicate a slave, which only focuses on HC satisfaction, while other islands will focus on satisfying class- and student-level SCs. The HC dedicated slave will generate a migrant that is satisfies the HCs and distributes it to other islands. All slaves communicate with each other via migration controlled by the DDMP algorithm in the master island.

With these considerations, our slave island localization strategy focuses on three different areas. The first slave focuses on maintaining HCs, the second slave focuses on solving class-level SCs with the objective function as shown in (29), and the third slave focuses on solving student-level SCs with the objective function as shown in (30).

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

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

4.3. Slave Island GA Previous research by Gozali *et al.* [2] succeeded in solving medium-scale student sectioning UCTP. They implemented the asynchronous island model GA (AIMGA), which is a basic island model GA with an asynchronous mechanism. However, this work has more complex student sectioning UCTP, and the previous solution cannot handle it. Therefore, this work proposes DM-LIMGA as its solution.

DM-LIMGA uses a localization strategy by implementing a different kind of GA for each slave. We modify the previous GA model used to solve Telkom UCTP [2]. We divide the slaves into shallow GA (SGA) for speed-based, medium GA (MGA) for standard, and deep GA (DGA) for performance-based. Each slave performs GA procedure, which is shown in Fig. 5.

Figure 5 shows that for each generation t of island i, 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.

Algorithm 2 GA Procedure

Require: Island i in generation t
  if $t = 1$ **then**
    Greedy Initialization population $P[PopSize]$
  end if
  $P' = \emptyset$ *//empty population*
  // elitism
  if there is a migrant found by DDMP algorithm **then**
    Put it into P'
  else
    Put the best individual of P into P'
  end if
  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
    if Island i is SGA **then**
      Evaluate using HC Evaluation
    else
      Evaluate using Class-Level Evaluation
    end if
    $[idv_1, idv_2]$ = best 2 individuals of parents&offsprings
    // stage 1 mutation
    Mutate idv_1 and idv_2 with probability P_m
    if Island i is SGA **then**
      Evaluate using HC Evaluation
    else
      Evaluate using Class-Level Evaluation
    end if
    $[idv_1, idv_2]$ = best 2 individuals of parents&offsprings
    // stage 2 mutation
    if Island i is DGA **then**
      Mutate idv_1 and idv_2 with probability P_m
      Evaluate using Student-Level Evaluation
      $[idv_1, idv_2]$ = best 2 individuals of parents&offsprings
    end if
    Put idv_1 and idv_2 into P'
    $count = count + 2$
  end while
  Replace P with P'

Fig. 5. GA procedure

We implement elitism to maintain elite individuals among the population. If there is a migrant found by DDMP algorithm, we put it into P'. Otherwise, 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. SGA island uses HC evaluation, and the others 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 divide the focus of each slave. Thus, its execution depends on the slave type. SGA and MGA execute stage 1 only, but DGA executes both stages. SGA's role is to maintain the reliability of the result by keeping the HCs. Because SGA only focuses on HC solution, SGA will be the fastest slave among all. SGA will actively correct the HC violation of the best individual from the other islands.

MGA focuses on yielding result by satisfying the class-level SCs. MGA is slower than SGA but faster than DGA, which focuses on yielding result by satisfying the student-level SCs. DGA is the slowest among all slaves because of the large number of students.

We mutate the new parents (stage 1 mutation) with mutation probability P_m to produce offsprings, and then they are evaluated. SGA island uses HC evaluation, and the others use class-level evaluation. We pick the best two individuals among parents and offsprings to be the new parents.

If we implement DGA, we continue to stage 2 mutation with a mutation probability P_m. Stage 2 mutation is the same as stage 1 mutation, but not 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 we proceed to the next generation.

These are the specific configurations of each slave:

- SGA

The main goal of SGA is to focus on solving HCs. To achieve this goal, SGA runs GA operations (mutation, crossover, and selection) by considering only the HCs. Thus, SGA uses (28) for the evaluation and only runs stage 1 mutation.

- MGA

The main goal of medium GA (MGA) is to yield class-level timetable. MGA runs GA operation by considering class-level SCs. As a consequence, MGA uses (29) for the evaluation and only runs stage 1 mutation.

- DGA

The main goal of deep GA (DGA) is to yield student-level timetabling. MGA runs GA operation by considering all constraints, including student-level SCs. As a consequence, DGA runs not only stage 1 mutation with (29) but also stage 2 with (30) for the evaluation.

4.3.1. Crossover We use a multi-point crossover where the number of affected genes is N_c of all genes that violate constraints. The crossover follows these steps:

1. Take two individuals from the selection as parents.
2. Select N_c of all events that violate constraint in the first parent.
3. Select an event out of them.
4. Select an event randomly from the second parent which has same room capacity with the selected event from first parent regardless of the violation.
5. Swap the selected event of first parent with second parent.
6. Repeat steps 3–5 until all selected events from first parent are swapped.

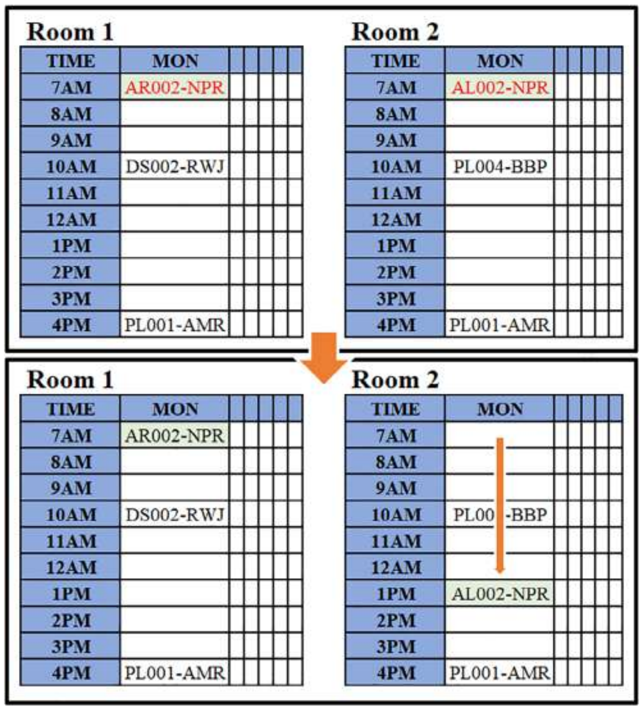


Fig. 6. M1 (moving)

4.3.2. Mutation We use three mutation steps to improve the probability of producing better offsprings. These three mutations are 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 that violate the constraints.

- M1 (Moving)

Select an event that violates the constraint. Move this event to an unused packet (see (1)). The unused packet is a packet that 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. 6.

- M2 (Swapping)

Select an event that violates the constraint as the first event. Find other events that 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. 7.

- M3 (Comparing)

Select an event that violates the constraint. Select randomly two other events that have same room capacity regardless of the violation. Swap the violated event with the one that produces lower violations. If the new individual decreases violations, keep the new individual; otherwise cancel the swap. M3 (Comparing) is illustrated in Fig. 8.

4.4. Bias value We adapted *Bias* value from forking genetic algorithm [23] to check diversity of current island i''s population at generation t. The bias value B_i^t is defined as the diversity degree of island i, which is $0.5 \leq B_i^t \leq 1.0$.

The previous bias formulation was binary type [23]. However, because this work uses enumeration type, we modified the bias formulation to normalize its value. The bias modification is shown

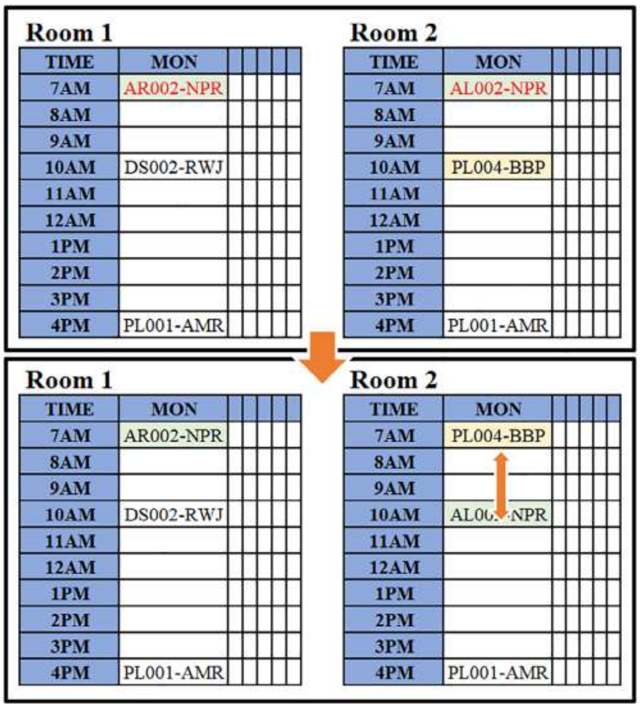


Fig. 7. M2 (swapping)

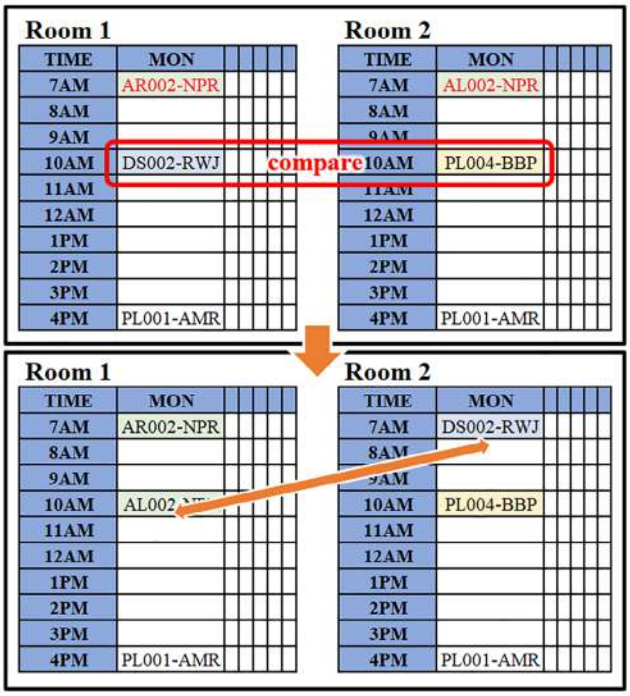


Fig. 8. M3 (comparing)

in (31), where p_i^t is the individual of island i ($p_i^t \in P_i^t$) so that $p_{i,e}^t$ is its gene value at event e. UB_i is an upper bound (maximum) value of all individual genes in island i (Max($p_{i,e}^t$) : $\forall p_{i,e}^t \in P_i^t \wedge e \in E$). $|P_i^t|$ is the population size and $|E|$ is the number of events.

$$B_i^t = \frac{1}{|P_i^t| \times |E|} \sum_{p_i^t \in P_i^t} \left(\left| \sum_{e \in E} \left[\frac{p_{i,e}^t + UB_i}{2 \times UB_i} \right] - \frac{|E|}{2} \right| + \frac{|E|}{2} \right) \tag{31}$$

4.5. Hamming distance Similar to bias formulation, the binary δ from previous research [18] needs to be modified for enumeration type chromosome representation. Instead of using all packet and event elements (1) and (2), we used three main elements (time, room, and lecturer) for the sake of simplicity and performance.

Equation (32) shows the modification of δ used in this work. Here, p_i and $p_{i'}$ are the compared individuals of island i and i'. Moreover, $p_{i,f(e,x)}$ is the gene value of individual from island i with $f(e,x)$ as its index ($f(e,x) = 5e + x$). e is an event ($e \in E$) and x: $x \in$ [1–3] is representation of time ($x = 1$), room ($x = 2$), and lecturer ($x = 3$), as can be seen in Fig. 4. Accordingly, MAX_1,MAX_2, and MAX_3 are the maximum index values of time $|T|$, room $|R|$, and lecturer $|L|$, respectively (see Table I).

$$\delta(i, i') = \sum_{e \in E} \sum_{x=1}^{3} \frac{p_{i,f(e,x)} - p_{i',f(e,x)}}{MAX_x} \tag{32}$$

4.6. Attractiveness We use *attractiveness* from previous research [24] to find the most potential island that produces better fitness in its last generation. Attractiveness α_i of an island i is given by

$$\alpha_i = \alpha_i^{prev} + (\eta_i^{pop} + \eta_i^{mig}), \qquad i = 1, 2, \ldots, I \tag{33}$$

where I is the total number of islands in the model, and α_i^{prev} is the attractiveness of the island i accumulated until the previous migration. Equations (34) and (35) explain the formulation of η_i^{pop} and η_i^{mig}, respectively.

$$\eta_i^{pop} = \left| \frac{\sum_{k=1}^{S_i^p} (f_k{}^{P_i^{prev}} - f_k{}^{P_i})}{S_i^p} \right| \tag{34}$$

where S_i^p is the size of native (original) population of the island i, $f_k{}^{P_i}$ is the fitness value of the kth solution, and $f_k{}^{P_i^{prev}}$ is the previous fitness value of kth solution before migration.

$$\eta_i^{mig} = \left| \frac{\sum_{k=1}^{S_i^m} (f_k{}^{M_i^{prev}} - f_k{}^{M_i})}{S_i^m} \right| \tag{35}$$

Similar to η_i^{pop}, S_i^m is the size of migrant population of the island i, $f_k{}^{M_i}$ is the fitness value of the kth solution, and $f_k{}^{M_i^{prev}}$ is the previous fitness value of kth solution before migration.

5. Experimental Result

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

5.1. Parameter settings We set the weight of HCs much larger than SCs. We set the HC weight with a large number M, i.e., $M = 1000$, 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 values of SCs as proportional to their influence. From this consideration, the SCs penalty value configuration is presented in Table II.

We set the GA parameters from previous work [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 $PopSize = 30$.

Table II. 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

Table III. Telkom UCTP characteristics

No	Attributes	2011/2012	2016/2017
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	23 451
7	Average number of classes per students	6.48	5.03

5.2. Dataset Dataset used in this work was Telkom University odd/even semester for 2011/2012 (before merging) and 2016/2017 (after merging) enrollment years. To be specific, the student body at Telkom University has increased from 6570 students in 2011 to 23 451 in 2016. This increase is a result of the merging of four universities. The detailed dataset characteristics comparison is shown in Table III.

5.3. Experiment 1—proof of concept The first experiment implemented DM-LIMGA for Telkom UCTP 2011/2012 as well as 2016/2017 enrollment years. We observed its performance based on the best and average fitness values in five runs. For additional insight, we include the HC violation percentages. Table IV shows that DM-LIMGA could yield an acceptable fitness value for 2011/2012 as well as 2016/2017 enrollment years. DM-LIMGA could achieve a small violation percentage in timetabling, which means that we can accept these results as a feasible timetable.

5.4. Experiment 2—diversity analysis The reason behind the implementation of DM-LIMGA is to further maintain population diversity while pursuing a better result. The second experiment aims to analyze DM-LIMGA's performance for Telkom UCTP problems and monitor its bias value trend-line. Figures 9 and 10 represent the DM-LIMGA experimental results for Telkom UCTP 2011/2012 and 2016/2017, respectively. We took the fitness and bias values from the best island in every generation.

Table IV. DM-LIMGA result for Telkom UCTP

	Fitness		Violation %	
Problem	Best	Average	Best	Average
2011/2012	4540	4887	0.16%	0.18%
2016/2017	96 562	97 023.43	7.36%	7.43%

Those figures show that DM-LIMGA could preserve island diversity as well as get a better result by generations. The bias value lies between 0.82 and 1, which means good diversity preservation. Flat trend-line shows that DM-LIMGA still could tend to fall in convergence, though the population diversity is already well preserved.

5.5. Experiment 3—comparison analysis The third experiment compares DM-LIMGA together with the standard (GA) [25], and asynchronous island model genetic algorithm (AIMGA) as a previous solver for Telkom UCTP [2], and UniTime [26] as a current generic UCTP solver. The parameter configuration and chromosome structure of GA and AIMGA were the same as with DM-LIMGA. The comparison details are shown in Table V.

According to [26], UniTime has a different constraint configuration format from Telkom UCTP. Therefore, we conducted a constraint mapping from Telkom UCTP into UniTime format (v2.3), which is explained in Table VI, 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 h or more in between, **NHB_LT** means given classes must have less than 6 h 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 VII shows the average of violation percentage comparison of DM-LIMGA, GA, AIMGA, and UniTime in five runs. The unfeasible value in UniTime cell for Telkom UCTP 2016/2017 enrollment year means that it could not get a result in a reasonable time (6 h runtime limit exceeded). Besides, this table points out that DM-LIMGA could surpass other algorithm results for both problems.

5.6. Experiment 4—benchmark analysis This last experiment compares the DM-LIMGA's performance with several UCTP solutions by using the International Timetabling Competition (ITC) 2007 benchmark datasets [12]. Table VIII shows the problem specification of this dataset. There are 24 test cases with

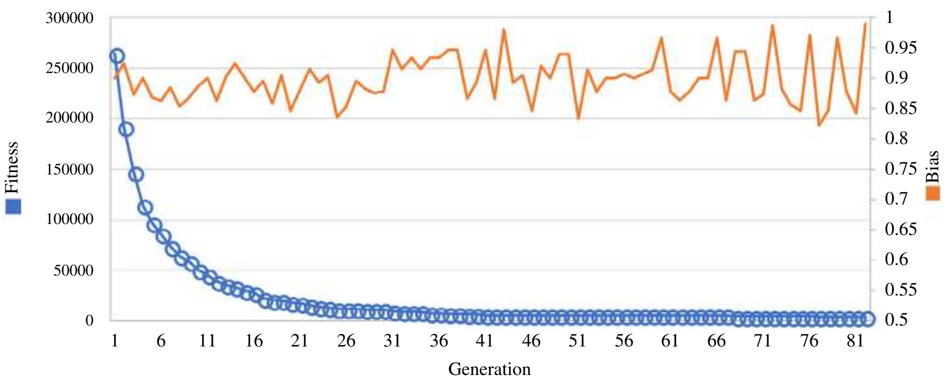


Fig. 9. Fitness-bias trend-line for problem 2011/2012

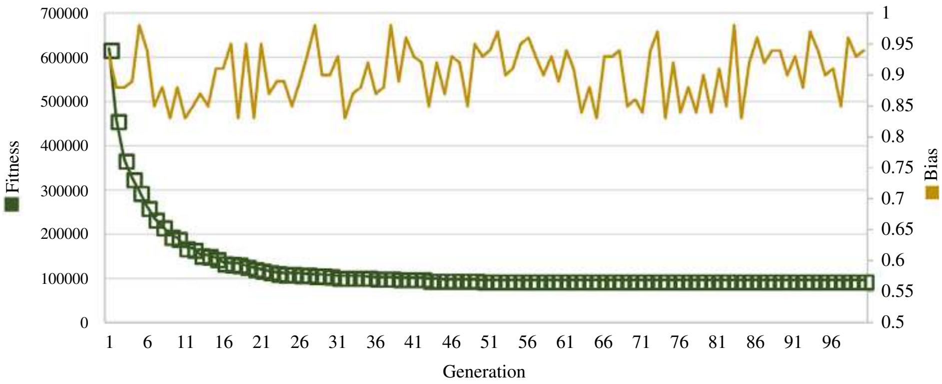


Fig. 10. Fitness-bias trend-line for problem 2016/2017

Table V. M-LIMGA, GA, and AIMGA configuration comparison

	Algorithm		
Differences	DM-LIMGA	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	Yes	No	Yes
GA core	Three level directed GA	Standard GA	All informed GA
Migration policy	DDMP	None	Migration protocol

Table VI. 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 VII. DM-LIMGA violation percentage comparison

Problem	DM-LIMGA	GA	AIMGA	UniTime
2011/2012	0.18%	2.72%	2.39%	13.85%
2016/2017	7.43%	57.34%	25.54%	Unfeasible

various combinations 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. 5, stage 1).

Table IX displays the benchmark experimental result of ITC-2007 dataset. The values show the fewest violation result of each solution. We compare DM-LIMGA with several current UCTP solutions, such as CBS: Constraint Based Solver by Muller [5], TSA:Tabu Search Approach by Lu and Hao [9], **CSP:** Constraint Satisfaction Problem by Atsuta [13], TAM: Threshold Acceptance Metaheuristic by Geiger [7], RBT: Repair Based TimeTable Solver by Clark [8], ATS: Adaptive Tabu Search by Lu and Hao [10], HMA: A Hybrid Metaheuristic Approach by Salwani Abdullah [14], ITS-LS: Incorporating Tabu Search and Local Search by Atsuta *et al.* [13], GDA: Great Deluge Algorithm with Kempe Chain by McCollum *et al.* [17], ILS: Iterative Local Search by Soria-Alcaraz *et al.* [6], HGATS: The Hybrid Approach Hybrid Genetic Algorithm and Tabu search by Jat and Yang [15], MMA: Mixed Metaheuristic Approach by Cambazard *et al.* [16], CTI: Combination of a General Purpose Constraint Satisfaction Solver, Tabu Search and Iterative Local Search Techniques by Atsuta *et al.* [27], HA: A Hybrid Algorithm by Chiarandini *et al.* [12], and ACO: Ant Colony Optimization algorithm in Conjunction with A Iterative Local search by Nothegger *et al.* [11].

These results are extracted from each paper or a review paper in UCTP by Babaei *et al.* [28]. 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 10 shows that DM-LIMGA could get the fewest violations for 13 of 24 test cases which are shown by bold value in the table. This result proves the consistency and reliability of DM-LIMGA in handling UCTP. It supports our finding in previous experiments. Moreover, DM-LIMGA could yield better results among the current UCTP solvers not only for Telkom University datasets but also general UCTP benchmarks.

6. Conclusions

This paper showed that the DM-LIMGA could overcome not only Telkom UCTP 2011/2012 (before merging) but also 2016/2017 (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 DM-LIMGA could handle scaling UCTP well and produce a feasible timetable.

Furthermore, from the second experiment, we could conclude that the reason behind DM-LIMGA's performance is its ability

Table VIII. Complete specification of ITC-2007 dataset

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

Table IX. Violation numbers of all solvers for ITC-2007 dataset

Problem	CBS	TSA	CSP	TAM	RBT	ATS	HMA	ITS-LS	GDA	ILS	HGATS	MMA	CTI	HA	ACO	DM-LIMGA
ITC-1	**5**	**5**	**5**	**5**	10	**5**	**5**	**5**	**5**	**5**	523	571	61	1482	15	**5**
ITC-2	51	55	50	111	111	**34**	39	50	60	48	342	993	547	1635	**0**	382
ITC-3	84	71	82	128	119	**70**	76	82	81	76	379	164	382	288	391	82
ITC-4	37	43	**35**	72	72	38	**35**	**35**	39	41	234	310	529	385	239	38
ITC-5	330	309	312	410	426	298	315	312	31	303	**0**	5	5	559	34	5
ITC-6	48	53	69	100	130	47	50	69	45	54	**0**	**0**	**0**	851	87	**0**
ITC-7	20	28	42	57	110	19	12	42	21	25	**0**	6	**0**	10	**0**	**0**
ITC-8	41	49	40	77	83	43	37	40	41	47	**0**	**0**	**0**	**0**	4	**0**
ITC-9	109	105	110	150	139	99	104	110	102	106	1102	1560	**0**	1947	**0**	**0**
ITC-10	16	4	27	71	85	16	10	9	17	23	515	2163	**0**	1741	**0**	**0**
ITC-11	**0**	**0**	**0**	**0**	3	**0**	**0**	**0**	**0**	**0**	246	178	548	240	547	**0**
ITC-12	333	343	351	442	**4.8**	320	337	351	349	324	241	146	869	475	32	242
ITC-13	66	73	68	622	113	65	61	68	43	68	**0**	**0**	**0**	675	166	**0**
ITC-14	59	57	59	90	84	52	53	59	59	53	**0**	1	**0**	804	**0**	**0**
ITC-15	84	71	82	128	119	69	73	82	82	74	**0**	**0**	379	**0**	**0**	**0**
ITC-16	34	39	40	81	84	38	32	40	49	42	**0**	2	91	1	41	32
ITC-17	83	91	102	124	152	80	72	102	81	81	**0**	**0**	1	5	68	81
ITC-18	83	96	68	116	110	67	77	68	79	69	**0**	**0**	**0**	3	26	**0**
ITC-19	62	65	75	107	111	59	60	75	67	65	121	1824	1862	1868	**22**	75
ITC-20	27	47	61	88	144	35	**22**	61	30	35	304	445	1215	396	2735	46
ITC-21	103	106	123	174	169	105	95	123	110	106	36	**0**	**0**	602	33	**0**
ITC-22	—	—	—	—	—	—	—	—	—	—	1154	29	**0**	1364	**0**	**0**
ITC-23	—	—	—	—	—	—	—	—	—	—	963	**238**	430	688	1275	378
ITC-24	—	—	—	—	—	—	—	—	—	—	274	**21**	720	822	30	25

to preserve the population diversity for each of the slave islands. However, because of the problem complexity, DM-LIMGA still could tend to fall in convergence, though it gives an acceptable result. Moreover, the final experiment shows that DM-LIMGA's performance is better than those of other solvers not only for Telkom University datasets but also general UCTP benchmark datasets.

Finally, this study confirms that DM-LIMGA can solve the student sectioning Telkom UCTP with an acceptable result. This proposed approach also proves its scalability by overcoming scaling Telkom UCTP. This study also gives additional evidence that encourages the implementation of DDMP in LIMGA, which could maintain its population diversity. Also, further studies still need to be conducted for applying DM-LIMGA to the other UCTP benchmarks. A more in-depth investigation into the convergence in the last half of generations is also needed. Further studies on its network cost are necessary, too, to investigate the real computational time and cost.

Acknowledgments

We thank the Indonesia Endowment Fund for Education (LPDP), a scholarship from the Ministry of Finance, Republic of Indonesia, for supporting this work. This work was conducted at the Graduate School of Information, Production, and Systems, Waseda University.

References

(1) Garey M, Johnson DS. *Computer and Intractability*. W.H. Freeman and Company: New York; 1979.

(2) Gozali AA, Tirtawangsa J, Basuki TA. *Asynchronous Island Model Genetic Algorithm for University Course Timetabling*. *Proceedings of the 10th International Conference on the Practice and Theory of Automated Timetabling* PATAT, New York, 2014; 179–187.

(3) Murray K, Muller T, Rudova H. Modeling and solution of a complex university course timetabling problem. In Practice and Theory of Automated Timetabling VI, *Number 3867 in Lecture Notes in Computer Science*. Burke EK, Rudová H (eds). Springer: Berlin, Heidelberg; 2006; 189–209. https://doi.org/10.1007/978-3-540-77345-0_13.

(4) Carter MW. 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*. Burke E, Erben W (eds). Springer: Berlin, Heidelberg; 2001; 64–82.

(5) Muller T, Murray K. Comprehensive approach to student sectioning. *Annals of Operations Research* 2010; **181**:249–269.

(6) Soria-Alcaraz JA, Özcan E, Swan J, Kendall G, Valadez JMC. Iterated local search using an add and delete hyper-heuristic for university course timetabling. *Applied Soft Computing* 2016; **40**:581–593.

(7) Geiger MJ. Applying the threshold accepting metaheuristic to curriculum based course timetabling. *Annals of Operations Research* 2010; **194**(1):189–202.

(8) M. Clark, M. Henz, and B. Love. *Quikfix a repair-based timetable solver*. *Proceedings of the Practice and Theory of Automated Timetabling (PATAT 2008)*, Montreal, 2008.

(9) G. Lu and S. Areibi. An Island-based ga implementation for vlsi standard-cell placement. In K. Deb, ed, *Genetic and Evolutionary Computation—GECCO* 2004*: Genetic and Evolutionary Computation Conference*, *Seattle*, *WA*, *USA*, *June 26–30* ,2004. *Proceedings*, *Part II*, pages 1138–1150, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

(10) Lü Z, Hao J-K. Adaptive tabu search for course timetabling. *European Journal of Operational Research* 2010; **200**(1):235–244.

(11) Nothegger C, Mayer A, Chwatal A, Günther. R. Raidl. Solving the post enrolment course timetabling problem by ant colony optimization. *Annals of Operations Research* 2012; **194**(1):325–339.

(12) L. D. Gaspero, B. M. Mccollum, and A. S. A. 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.

(13) M. Atsuta, K. Nonobe, and T. Ibaraki. Itc2007 track 2: An approach using general csp solver. In *Proceedings of the Practice and Theory of Automated Timetabling (PATAT 2008)*, Montreal, 2008.

(14) Abdullah S, Turabieh H, McCollum B, McMullan P. A hybrid metaheuristic approach to the university course timetabling problem. *Journal of Heuristics* 2010; **18**(1):1–23.

(15) Jat SN, Yang S. A hybrid genetic algorithm and tabu search approach for post enrolment course timetabling. *Journal of Scheduling* 2010; **14**(6):617–637.

(16) Cambazard H, Hebrard E, O'Sullivan B, Papadopoulos A. Local search and constraint programming for the post enrolment-based course timetabling problem. *Annals of Operations Research* 2010; **194**(1):111–135.

(17) B. McCollum, P. J. McMullan, A. J. Parkes, E. K. Burke, and S. Abdullah. *An extended great deluge approach to the examination timetabling problem*. *Proceeding of the Multidisciplinary International Scheduling Conference (MISTA)* 2009, Dublin, 2009.

(18) Gozali AA, Fujimura S. Localization strategy for Island model genetic algorithm to preserve population diversity. In *Computer and Information Science, Chapter Studies in Computational Intelligence*. Lee R (ed). Springer International Publishing: New York, 2018; 149–161.

(19) Gozali AA, Fujimura S. *Performance analysis of localization strategy for island model genetic algorithm*. *18th IEEE/ACIS International Conference on Software Engineering*, *Artificial Intelligence*, *Networking and Parallel/Distributed Computing (SNPD 2017)*, Kanazawa, 2017; 3.

(20) A. A. Gozali and S. Fujimura. A dual dynamic migration policy for island model genetic algorithm. In *2017 International Conference on Sustainable Information Engineering and Technology (SIET)*. IEEE2017.

(21) Banczyk K, Boinski T, Krawczyk H. *Parallelisation of genetic algorithms for solving university timetabling problems*. In International Symposium on Parallel Computing in Electrical Engineering (PAR-ELEC'06), Bialystok, 2006; 325–330.

(22) Gozali AA, Fujimura S. Dual migration localized Island model genetic algorithm, a better diversity preserver Island model. *Evolutionary Intelligence* 2019; **12**(4):527–539.

(23) Tsutsui S, Fujimoto Y, Ghosh A. Forking genetic algorithms: Gas with search space division schemes. *Evolutionary Computation* 1997; **5**(1):61–80.

(24) Duarte G, Lemonge A, Goliatt L. *A dynamic migration policy to the island model*. 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, 2017; 1135–1142.

(25) Goldberg DE. Genetic Algorithms in Search,Optimization and Machine Learning. 1st ed. Addison-Wesley Longman Publishing Co., Inc.: Boston, MA; 1989.

(26) Muller T. University course timetabling: Solver evolution. In *Proceedings of the 11th international conference on the Practice And Theory of Automated Timetabling*, *2016*, 082016.

(27) B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A. J. Parkes, L. DiGaspero, R. Qu, and E. K. Burke. Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal on Computing*, **22**(1):120–130, feb 2010.

(28) Babaei H, Karimpour J, Hadidi A. A survey of approaches for university course timetabling problem. *Computers & Industrial Engineering* 2015; **86**:43–59.

Alfian A. Gozali (Non-member) was born in Yogyakarta, Indonesia, on October 26, 1988. He received the Master's degree in informatics engineering from Telkom University, Indonesia, in 2014. Currently, he is pursuing the Ph.D. degree at the Graduate School of Information, Production, and Systems, Waseda University. His research interests include optimization algorithms, especially a multi-population evolutionary algorithm for scheduling problem.

Bobby Kurniawan (Non-member) received the Bachelor's and Master's degrees from Institute Teknologi Bandung, Indonesia, in 2002 and 2012, respectively. He is currently pursuing the Ph.D. degree at the Graduate School of Information, Production and Systems, Waseda University. His research interests include scheduling, production planning and control, metaheuristics, and mathematical modeling.

Wei Weng (Non-member) received the B.E. degree from Fudan University, China, and the M.E. and Ph.D. degrees from Waseda University in 2008 and 2011, respectively. After being a postdoctoral researcher with the Research Center of Information, Production and Systems, Waseda University, she worked as an Assistant Professor at the Graduate School of Information, Production and Systems, Waseda University, from 2013 to 2018. She is now an Associate Professor with the Institute of Liberal Arts and Science, Kanazawa University. Her research interests include manufacturing control, logistics, scheduling, multi-agent systems, and just-in-time systems.

Shigeru Fujimura (Senior Member) received the B.E., M.E., and Dr. Eng. degrees from Waseda University in 1983, 1985, and1995, respectively. From 1985 to 2003, he was with the Yokogawa Electric Corporation. Currently, is a Professor with the Graduate School of Information, Production, and Systems, WU. His research interests include production management, production scheduling, intelligent interface agent, object-oriented modeling, and software engineering.