Guided Genetic Algorithm to Solve University Course Timetabling with Dynamic Time Slot

Muhammad Fachrie
Department of Informatics
Universitas Teknologi Yogyakarta
Yogyakarta, Indonesia
muhammad.fachrie@staff.uty.ac.id

Anita Fira Waluyo
Department of Informatics
Universitas Teknologi Yogyakarta
Yogyakarta, Indonesia
anitafira@staff.uty.ac.id

***Abstract*— This paper discusses about one of the most popular technique in Evolutionary Computation, i.e., Genetic Algorithm to solve the UCTP that has dynamic teaching time slot. Duration of a lecture is determined by the course's credit which means that the structure of course timetable can be different between one day to the others. This kind of timetable structure is more complex since the flexible time slot could produce high number of schedules possibility. Hence, a flexible chromosome is developed to deal with dynamic time slot and enhanced the efficiency of the system by eliminating the crossover process which usually time consumed. A Guided Creep Mutation is proposed to act as an evolutionary operator that guides the chromosomes to find global optimum by changing certain gene value gradually in every generation. Our system successfully generated optimum schedule for 878 and 1140 courses in odd and even semester with zero collision in rooms, time, lecturers, also satisfied all soft constraints given. Based on our experiments, the system needs relatively small number of generations with few chromosomes to generate an optimum schedule.**

Keywords*— *course timetabling, genetic algorithms, combinatorial optimization, evolutionary computation, dynamic time slot

I. INTRODUCTION

University Course Timetabling Problem (UCTP) is a combinatorial optimization problem which is very popular since long time ago. It is very complex and usually contains very large searching space. Since it is very popular, researchers have been successfully developed various approaches to solve UCTP. Genetic Algorithm (GA) is the most famous algorithm that is used as in [1]–[6] since it is an easy-to-implement algorithm and it has good performance in finding the solution for combinatorial optimization. However, in some situations, GA can be trapped in a condition called premature convergence, that happens when the chromosomes cannot find better solution than the current one. To solve this condition, a strategy called Local Search is applied [7]–[11].

Another approach to enhance the searching strategy of GA in generating optimal course timetable is by applying a so called directed or guided mutation to the violating genes in the chromosome as in [6], [12], [13]. This mechanism is conducted by changing the value from violating genes only in each chromosome. In some cases, it is not necessary to use crossover operator in a system.

However, previous works deal with UCTP that uses static time slot for each lecture meeting. In this work, UCTP has dynamic time slot that depends on the duration of lectures held on that room each day. This problem produces very large possible solutions if solved using previous approaches. Therefore, a Guided Genetic Algorithm that applies guided mutation procedure is proposed in this work to help in finding the optimum timetable.

II. METHODS

A. System Architecture

The architecture of the system is quite simple with only three main process in Genetic Algorithm (GA) module as illustrated in Fig. 1. There are two main inputs to the system, i.e., list of courses and timetable policies. A list of courses could contain hundreds or thousands of data with several attributes, i.e., course name, lecturer, student group (class), course type, and course credit. Each record (row) represents a single course meeting.

Timetable policies is a set of rules that is created by university to organize the course timetable such that all course meetings can be held effectively. A good course timetable should have no conflict on lecturers, rooms, and time, also satisfies several additional policies in the university.

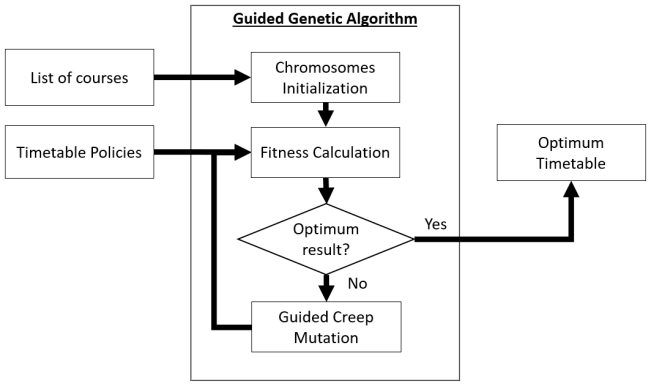

Fig. 1. The system architecture that is developed in this research.

There are three main processes in our Guided Genetic Algorithm module that is started with chromosomes initialization based on list of courses input to the system. The number of gene in a single chromosome depends on the number of courses given to the system. As illustrated in Fig. 2, a chromosome has N number of genes that represents the schedule from M courses. Each course is encoded using 3 genes where each of it represents the Day (D), Time (T), and Room (R). Hence, length of a single chromosome is three times from the number of course meetings. This chromosome design is more flexible to evolve in generations since the timeslot for each course is distributed in three different genes, i.e., D, T, and R. Therefore, the algorithm can manage the gene changing based on which timetable constraints that is violated by the course. For example, if a course has lecturer conflict, then the algorithm should change the day (D) or the time (T) of related course only and it does not require to change the room (R). At the beginning of evolutionary process, the chromosomes are randomly initialized as is standard procedure in GA.

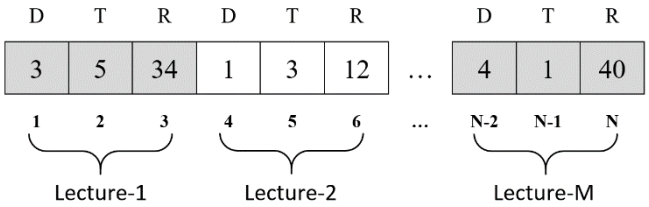

Fig. 2. Design of chromosome that has N genes that consist of M courses. Each course schedule is encoded with 3 chromosomes that represents the Day (D), Time (T), and Room (R).

Guided Creep Mutation is executed after the chromosomes has been initialized. This process is the only evolutionary operator in proposed system. It works by simply changing current gene value (allele) with the next close value.

After mutation is complete, each chromosome is measured to identify their fitness value that represents the quality of every chromosome. If a chromosome has a lot of conflicted courses, then the fitness value is small which means that this chromosome has low quality and vice versa. The next section will explain more detail about fitness function which is used in this work.

B. Timetable Policy

The system was not only designed to work effectively, but also must have feasible time in resulting the appropriate course timetable. On the first step, it is necessary to construct the timetable specification that must be satisfied by the algorithm. This process was customized based on the policies in our institution. This specification is defined as a group of constraints that guide the optimization algorithm to find the best timetable. There are two kinds of constraints that is used in this research, i.e. hard constraints (HC) and soft constraints (SC) [6], [14]–[16].

TABLE I. LIST OF HARD CONSTRAINTS (HC) AND SOFT CONSTRAINTS (SC)

Constraints	Type	Penalty
No lecturer conflict	HC1	100
No students group (class) conflict	HC2	100
No room conflict	HC3	100
Practical course must be held in laboratory	HC4	100
Every course meeting must be held in room with enough capacity	HC5	100
Lecturer with structural position may not teach before 10.30	HC6	100
A lecturer should not teach more than 6 credits in a day	SC1	10
Lecturer should not teach after 12.00 PM on Saturday	SC2	10
Lecturer should not teach in the first time slot if the day before he taught in the last time slot	SC3	10
Lecturer should not teach more than 7 hours in a day	SC4	10
The number of lectures on Saturday should be less than on other days	SC5	10
Students should not attend the class more than 6 credits in a day	SC6	5

HC is a set of constraints that must be fulfilled by the timetable system. The violation of it implies in resulting invalid course schedule. While the violation of SC should be minimized if cannot be totally avoided. Every violation of HC or SC will be penalized by certain points as shown by Table I. Every penalty is given based on the constraint significancy in resulting a good course timetable.

A chromosome must satisfy all HC and as much as possible of SC to be selected as the best solution for UCTP. If a chromosome satisfies all HC and SC, then it will have the maximum fitness value.

In this research, beside HC and SC policies, there is another thing that makes this research more challenging, i.e., flexibility of time slot between one day and the others. This happens because the duration of every course meeting depends on its credit, where 1 credit equals to 50 minutes of course meeting. In a single day with maximum 8 hours of lecture meetings for a single room, a dynamic time slot can produce many time slot possibilities, e.g., 2-2-2-2, 2-2-4, 2-4-2, 4-2-2, 2-3-3, 3-2-3, 3-3-2, 4-4, 3-4, etc. Hence, there are a lot of time slot possibilities that can be produced in dynamic time slot. This is quite different with regular course timetable which has static time slot where each course meeting has equal duration every single day. Fig. 4 illustrates the difference between timetable with static time slot and dynamic time slot.

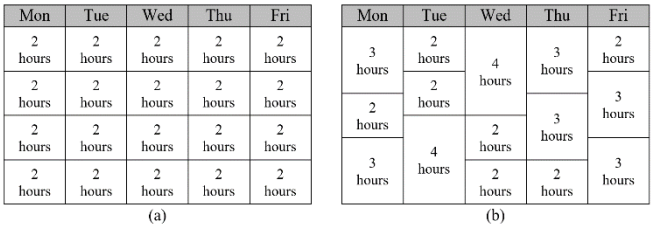

Fig. 3. The difference between timetable with static time slot (a) and dynamic time slot (b).

The chromosome that is mentioned in Fig. 1 is designed to deal with a lot of time slot possibilities that appears in dynamic timetable. The proposed system did not set the time slot as fixed group of all possible combinations of day, time, and room, instead the day, time, and room are set independently to get a flexible chromosome. As shown in Table II, there are total of 6 days available for lecture meeting, 53 classrooms for theoretical courses and 13 computer laboratories for practical courses.

TABLE II. LIST OF DAYS AND ROOMS SPECIFICATION

Items	Specification
Days	6 days (Monday – Saturday)
Small Classroom	1 room (max. 40 students)
Medium Classrooms	15 rooms (max. 60 students)
Large Classrooms	32 rooms (max. 70 students)
Extra Large Classrooms	5 rooms (max. 90 students)
Small Laboratories	11 rooms (max. 45 students)
Large Laboratories	2 rooms (max. 70 students)

Since the duration of a lecture depends on credit of the course, then the lecture time slot is set by defining 8 possibilities of lecture starting time from 07.00 until 16.30 as described in Table III. The end time for each time slot is not set since it is determined based on the credit of the course.

TABLE III. TIME SLOT SPECIFICATION

Starting Time	Constraint
07.00	Suitable for all courses
08.50	Suitable for all courses
09.40	Suitable for all courses
10.40	Suitable for all courses
12.50	Suitable for all courses
14.40	Suitable for all courses
15.30	Not suitable for courses with credit > 3
16.30	Not suitable for courses with credit > 3

C. Guided Creep Mutation

To enhance the searching mechanism in Genetic Algorithm, the proposed system does not utilize too much random number generation, especially in mutation schema. Hence, the system uses the Creep Mutation procedure that changes the value of gene in a small step size to make the searching process is directed to find the optimum solution. Different to random-based mutation that, this strategy observes the solution in a structured way so that the system can reach the global optimum in feasible time.

Beside changing the gene value in a small step size, the mutation process is also guided by some rules to reduce the searching space. Gene from each lecture that violates HC and SC is mutated based on which constraint that is violated. Violation to SC1 or SC2 or SC5 or SC6 will change the 'day' gene because those constraints are related to 'day' utilization. Violation to SC3 will change the 'time' gene. Since HC1, HC2, and SC4 is related to 'day' and 'time' utilization, hence the 'day' or 'time' gene can be changed when there is violation to HC1 or HC2. In this case, the system set a random selection to choose whether 'day' or 'time' gene that will be changed. Violation to HC3 which is related to 'day', 'time', and 'room' utilization will change one of 'day', 'time', or 'room' gene depending on random value generated by the computer. Meanwhile, HC4 and HC5 will change only the 'room' gene when violated. The algorithm from combination between Creep Mutation and these customized rules is described in Algorithm 1.

D. Fitness Function

Fitness function is used to measure the effectiveness of solution given by every chromosome. In UCTP, fitness function is calculated based on the number of constraint violations occurs in the chromosome. More violations on hard or soft constraints resulting on lower fitness value. This research uses fitness function which is given in (1), where M is the number of lectures and P is total of penalties for each lecture.

$$fitness = 100 \times M - \left(\sum_{i=1}^{M} P_i\right) \quad (1)$$

Equation (1) aims to maximize the fitness value by lowering the number of penalties. If a chromosome has zero penalty, then its fitness value equals 100 times the number of total lectures, e.g. if there are 1000 lectures, hence the fitness value is 100000. Whereas, when the penalty is maximum, then the fitness value can be lower than 0.

Algorithm 1. Guided Creep Mutation Algorithm

```
input: set of chromosomes, list of violating lectures
for every chromosome do:
    for every lecture in current chromosome do:
        day_changed, time_changed, room_changed = false

        if current lecture violates HC1 or HC2 or SC4:
            generate random number in range [0, 1]
            if random number < 0.5:
                current day += 1
                day_changed = true
            else:
                current time += 1
                time_changed = true
            end if
        end if

        if current lecture violates HC3:
            generate random number in range [0, 1]
            if random number < 0.2 and not day_changed:
                current day += 1
                day_changed = true
            else if random number < 0.4 and not time_changed:
                current time += 1
                time_changed = true
            else if not room_changed:
                current room += 1 + random[0, 3]
                room_changed = true
            end if
        end if

        if current lecture violates HC4 or HC5:
            if not room_changed:
                current room += 1 + random[0, 3]
            end if
        end if

        if current lecture violates SC1 or SC2 or SC5 or SC6:
            if not day_changed:
                current day += 1
            end if
        end if

        if current lecture violates SC3:
            if not time_changed:
                current time += 1
            end if
        end if
    end for
end for
```

E. Dataset

Two different datasets containing the courses, lecturers, credits, types of lecture, and number of participating students were prepared to test and evaluate the proposed system. Both datasets which are from odd and even semester, were gathered from Operational Division of Universitas Teknologi Yogyakarta. Each of it consists of 1140 and 878 lecture meetings. Table IV describes the detail of both datasets.

TABLE IV. UCTP DATASET SPECIFICATIONS

Odd Semester	Even Semester
1140 lecture meetings	878 lecture meetings
431 different courses	296 different courses
188 lecturers	157 lecturers
912 theoretical courses	660 theoretical courses
228 practical courses	218 practical courses

III. EXPERIMENTS

The experiment was conducted eight times for every dataset to evaluate the system performance comprehensively since Genetic Algorithm is started with random value. For each testing scenario, the system used only 10 chromosomes and 1000 generations at most. Since there is no crossover, then there is no crossover probability. All the violating gene are mutated using Guided Creep Mutation mechanism as explained in the previous section. In this work, the probability of mutation is set as 1 which means that all violating genes in every chromosome is mutated.

The experiment was run on personal computer with Intel core i5 8th gen with 16 GB or RAM to run the experiment. This hardware specification does not affect the experimental result, but only to give clarity related to computational time.

IV. RESULT AND DISUCUSSIONS

Using the dataset from odd semester that contains 1140 lecture meetings, the system successfully obtained optimum solution in less than 500 generations based on eight different observations which is about 40 to 60 minutes of computation. Based on our observation, one generation is executed about 11 to 13 seconds.

TABLE V. EXPERIMENTAL RESULTS FOR ODD SEMESTER

Obs.	Best Fitness	Average Fitness	Generations
1	114000	111307	352
2	114000	112208	450
3	114000	111809	415
4	114000	112873	480
5	114000	112592	374
6	114000	112609	397
7	114000	112200	310
8	114000	112892	498
Average		**112311.3**	**409.5**

TABLE VI. EXPERIMENTAL RESULTS FOR EVEN SEMESTER

Obs.	Best Fitness	Average Fitness	Generations
1	87800	86536	209
2	87800	87259	163
3	87800	86593	124
4	87800	86853	117
5	87800	87102	130
6	87800	86149	115
7	87800	87023	106
8	87800	87136	128
Average		**86831.4**	**136.5**

Meanwhile, using the dataset from even semester with 878 lecture meetings, the system works faster with only less than 200 generations in average to find the optimum timetable. This equals with 7 up to 15 minutes of computer execution. Table V and VI give the detail results from odd and even semester, respectively.

Both results show that the proposed system is running effectively in feasible computational time. The system successfully finds the optimum timetable with zero violation to all HC and SC. However, the system needs more than twice of generations in even semester to solve the timetable in odd semester. Using only 10 chromosomes, the system only needs to explore as many as 4095 average possible solutions from total of 7.8541e+3990 (6^{1140} x 66^{1140} x 8^{1140}) searching space for odd semester dataset and 1365 average possible solutions from total of 4.8908e+3073 (6^{878} x 66^{878} x 8^{878}) possible solutions for even semester dataset.

Beside the efficient searching space, the proposes system always produced high quality chromosome in every testing scenario. Based on Fig. 4, it shows that in every generation, every single chromosome always has fitness value that is close to the best fitness. At the beginning of evolution process, the fitness values are significantly increase. However, the evolution was trapped in local maxima for many generations since the fitness starts to be close to the maximum fitness, but finally it can reach the optimum global at the end due to the help of Guided Creep Mutation mechanism. The result shows that the chromosome design and mutation mechanism which is proposed enable to bring GA out of local maxima.

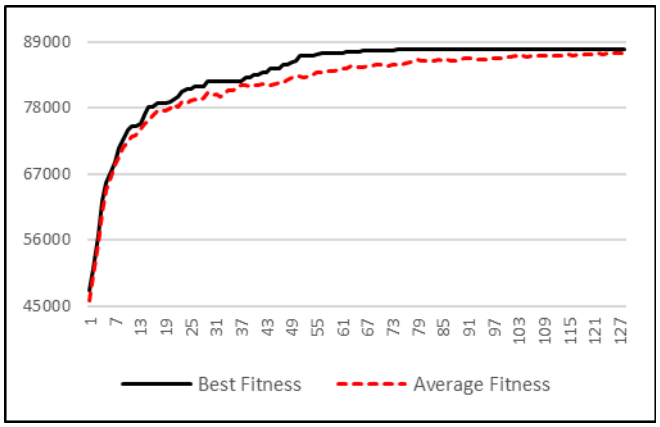

Fig. 4. The evolution process shows that every generation has high quality chromosome that has fitness value close to the best one.

Compared to other works in [13], [17], [18] that used much less lecture meetings, our proposed system works more efficiently with smaller number of chromosome and generation as shown in Table VII. To be noticed, there is no violation to all HC and SC in this work, while other mentioned works still have violation on some SC.

TABLE VII. COMPARISON BETWEEN THIS WORK AND PREVIOUS WORKS

References	Lectures	Chromosomes	Generations
[13]	100	100	200
[17]	300	20	1000
[18]	166	-	900
This paper	1140	10	409.5
This paper	878	10	136.5

V. CONCLUSIONS

A University Course Timetable System that can generate the timetable with dynamic time slot has been successfully developed. In this research, we designed a chromosome where a lecture meeting is encoded in three different genes to represent 'day', 'time', and 'room' for each lecture meeting. We also proposed the Guided Creep Mutation which works by changing the gene value in a small step size based on certain rules. This mutation mechanism successfully helps GA to avoid local maxima and finally finding the best solution in feasible time. All in all, our system enables to find the optimum timetable that satisfies all hard and soft constraints specified by Universitas Teknologi Yogyakarta.

ACKNOWLEDGMENT

We would like to thank to Direktorat Riset dan Pengabdian Masyarakat, Deputi Bidang Penguatan Riset dan Pengembangan, Kementerian Riset dan Teknologi/ Badan Riset dan Inovasi Nasional Republik Indonesia for the financial support, so that the research can be completed successfully.

REFERENCES

[1] D. Qaurooni and M. R. Akbarzadeh-T, "Course timetabling using evolutionary operators," *Appl. Soft Comput. J.*, vol. 13, no. 5, pp. 2504–2514, 2013.

[2] R. Ilyas and Z. Iqbal, "Study of hybrid approaches used for university course timetable problem (UCTP)," *Proc. 2015 10th IEEE Conf. Ind. Electron. Appl. ICIEA 2015*, pp. 696–701, 2015.

[3] R. Çolak and T. Yiğit, "Ders Çizelgeleme Probleminin Çözümüne Genetik Algoritma Parametrelerinin Etkisi," *2nd Int. Conf. Comput. Sci. Eng. UBMK 2017*, pp. 1090–1094, 2017.

[4] R. Ferdiana, N. Ridwan, and N. F. Hidayat, "A pragmatic algorithm approach to develop course timetable web application based on cloud technology," *Proc. - 2017 7th Int. Annu. Eng. Semin. Ina. 2017*, 2017.

[5] S. A. F. Oliveira and A. R. R. Neto, "A Novel Educational Timetabling Solution through Recursive Genetic Algorithms," *2015 Latin-America Congr. Comput. Intell. LA-CCI 2015*, 2016.

[6] Suyanto, "An Informed Genetic Algorithm for University," *Lect. Notes Comput. Sci.*, vol. vol 6114, no. 1, pp. 229–236, 2010.

[7] A. A. Mahiba and C. A. D. Durai, "Genetic algorithm with search bank strategies for university course timetabling problem," *Procedia Eng.*, vol. 38, pp. 253–263, 2012.

[8] J. Pandey and A. K. Sharma, "Survey on University timetabling problem," *Proc. 10th INDIACom; 2016 3rd Int. Conf. Comput. Sustain. Glob. Dev. INDIACom 2016*, pp. 160–164, 2016.

[9] T. Mauritsius, A. N. Fajar, Harisno, and P. John, "Novel Local Searches for Finding Feasible Solutions in Educational Timetabling Problem," *Proc. 2017 5th Int. Conf. Instrumentation, Commun. Inf. Technol. Biomed. Eng. ICICI-BME 2017*, no. November, pp. 270–275, 2018.

[10] T. Song, S. Liu, X. Tang, X. Peng, and M. Chen, "An iterated local search algorithm for the University Course Timetabling Problem," *Appl. Soft Comput. J.*, vol. 68, pp. 597–608, 2018.

[11] D. Lohpetch and S. Jaengchuea, "A hybrid multi-objective genetic algorithm with a new local search approach for solving the post enrolment based course timetabling problem," *Adv. Intell. Syst. Comput.*, vol. 463, pp. 195–206, 2016.

[12] A. A. Gozali and S. Fujimura, "Reinforced island model genetic algorithm to solve university course timetabling," *Telkomnika (Telecommunication Comput. Electron. Control.*, vol. 16, no. 6, pp. 2747–2755, 2018.

[13] J. B. Matias, A. C. Fajardo, and R. M. Medina, "A fair course timetabling using genetic algorithm with guided search technique," *Proc. 2018 5th Int. Conf. Bus. Ind. Res. Smart Technol. Next Gener. Information, Eng. Bus. Soc. Sci. ICBIR 2018*, pp. 77–82, 2018.

[14] H. Babaei, J. Karimpour, and A. Hadidi, "A survey of approaches for university course timetabling problem," *Comput. Ind. Eng.*, vol. 86, pp. 43–59, 2015.

[15] S. Yang and S. N. Jat, "Genetic algorithms with guided and local search strategies for university course timetabling," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 41, no. 1, pp. 93–106, 2011.

[16] S. Imran Hossain, M. A. H. Akhand, M. I. R. Shuvo, N. Siddique, and H. Adeli, "Optimization of University Course Scheduling Problem using Particle Swarm Optimization with Selective Search," *Expert Syst. Appl.*, vol. 127, pp. 9–24, 2019.

[17] K. Wang, W. Shang, M. Liu, W. Lin, and H. Fu, "A Greedy and Genetic Fusion Algorithm for Solving Course Timetabling Problem," *Proc. - 17th IEEE/ACIS Int. Conf. Comput. Inf. Sci. ICIS 2018*, pp. 344–349, 2018.

[18] D. Kristiadi and R. Hartanto, "Genetic Algorithm for lecturing schedule optimization," *IJCCS (Indonesian J. Comput. Cybern. Syst.*, vol. 13, no. 1, p. 83, 2019.