Application of Genetic Algorithm to Optimize Lecture Scheduling Based on Lecturers’ Teaching Day Willingness

Ridho Taufiq Subagio¹, Kusnadi², Tiara Eka Putri³, Petrus Sokibi⁴, Syaiful Zuhri Harahap⁵, Darmansah⁶

¹ Informatics Engineering, Catur Insan Cendekia University, Cirebon, Indonesia
²,³,⁴ Informatics Engineering, Catur Insan Cendekia University, Cirebon, Indonesia
⁵ Information System, Labuhanbatu University, Labuhanbatu, Indonesia
⁶ Information System, Telkom Institute of Technology, Purwokerto, Indonesia

Email: ¹ ridho.taufiq@cic.ac.id

Abstract. The scheduling of lectures is an important activity that is carried out every beginning of the semester. The process of making a lecture schedule in the Informatics Engineering Study Program of Universitas Catur Insan Cendekia Cirebon still uses office applications, so it takes a long time to complete. This is because certain criteria or conditions that make the scheduling of lectures complicated, such as the willingness of teaching days and courses for each lecturer to differ and the availability of parallel classes. The research methods used consist of several stages, namely data collection, analysis, and system design, as well as the implementation of software created. The method of data collection used is observation and library study. The method of analysis and design of the system used is Object-Oriented Analysis (OOA) using Unified Modelling Language (UML). Applications created using the PHP programming language with the Yii Framework and MySQL as databases, as well as applying genetic algorithms to solve scheduling problems. The result of this research is a lecture scheduling application used by BAAK Staff to create a lecture schedule in the Informatics Engineering Study Program. The application of this application can save time in the creation of lecture schedules.

1. Introduction

The scheduling problem in college is a complex one. The scheduling process that must be done every semester change becomes tiring and time-consuming work. In the case of college scheduling, any restrictions should not be violated [1]. The purpose of the scheduling problem in college is how to find a method to allocate all resources so that lectures can run well where all the constraints of these problems can be overcome [2].

The creation of a lecture schedule is a routine activity in universities and is a very important activity so that academic activities can run properly [6]. The lecture schedule at Catur Insan Cendekia University was created before the new semester began and is still done manually using office applications. The process of creating a lecture schedule that is done manually is by pairing lecturers, courses, classes, classrooms, days, and hours so that it takes a long time and still often occurs clashing schedules [3,14]. This is also the case in the Informatics Engineering study program. A common problem that often occurs is the length of the schedule creation process and the clash of schedules because there are many rules to consider. This manual process takes considerable time because it has several obstacles such as determining the lecture schedule for a lecturer who teaches a parallel class or other class for the different study program, the possibility of lecturer teaching more than one course, and how to accommodate the
schedule of a lecturer to teach according to the willingness of the desired day. Given the importance of this scheduling process, optimization steps are needed to make the scheduling process faster, and schedule clashes can be minimized [4,9].

Based on the description of the problem, this research aims to optimize the application of scheduling lectures using a genetic algorithm. The selection of the genetic algorithm is based on several references from previous research stating that the application of a genetic algorithm is capable of producing optimal scheduling and can minimize schedule clashes [3-5,7].

The scope of this research is: (1) The data sample used for testing is scheduling data from the Informatics Engineering study program, even semester 2018/2019. (2) The application that is made does not handle the combined class. (3) The lecture scheduling criteria consist of: (a) The lecturer day willingness to teach. (b) There should be no clash of time allocation for a lecture, use of the lecture room, and parallel class. (c) Certain times are not allowed for lecture activities. (4) The condition used for the genetic algorithm is: (a) Selection used is tournament selection. (b) Crosses used are position-based crosses. (c) The mutation used is position-based. (d) The parameters used were population size = 10, crossover probability = 1 and mutation probability = 0.2.

2. Research Methodology

2.1. Genetic Algorithm

Genetic Algorithms are heuristic search approaches that apply to a wide range of optimization problems [8]. Evolution is the basis of a genetic algorithm [18,28,30]. A genetic algorithm is a method of finding the optimal solution to a problem and will find a good solution by crossing other solutions to create new solutions [19,20]. A genetic algorithm is a computer program that simulates the process of evolution by randomly producing chromosomes from each population and allowing them to reproduce according to the laws of evolution in hopes of producing better chromosomes. This chromosome is the solution to the problem discussed, so with this better chromosome, it is expected to get a good solution to solve the problem [9,25].

The genetic algorithm begins by identifying an initial random population of individuals. Every individual in the Genetic algorithms begins with randomized populations of early individuals. Each individual in the population is a potential solution to the problem being discussed and considered. Individuals will thrive through iterations called generations. Each individual in the population will be evaluated with some measure of fitness. Next-generation populations will be created through genetic operators. This procedure will continue until the stopped conditions are met [10,11,23]. The general framework of genetic algorithms is described in Figure 1, where $P(t)$ denotes the population at generation $t$:

```
procedure: Genetic Algorithms
begin
  t := 0;
  initialize P(t);
  evaluate P(t);
  while (not termination condition) do
    begin
      t := t + 1;
      select P(t) from P(t - 1);
      alter P(t);
      evaluate P(t);
    end
end.
```

Figure 1. Genetic algorithm procedures.

2.2. Basic Structure of Genetic Algorithms

In genetic algorithms, three major genetic operators are usually used to create the next generation, namely: (1) Reproduction: According to the fitness values, increase or decrease the number of offspring for each individual in the population $P(t)$. (2) Crossover: Select two different individuals from the
population randomly and swap parts between the strings with the same probability as the crossover rate $p_c$. (3) Mutation: Alter one or more genes of a selected individual with a probability equal to the mutation rate $p_m$ [11,18].

![Flowchart of fundamental procedures of genetic algorithms](image)

**Figure 2.** A Flowchart of fundamental procedures of genetic algorithms.

2.2.1. *Initialization.* This process is carried out randomly to determine the initial population by taking into account the solution and domain boundaries used, while the subsequent population is the result of chromosome evolution through iterations called generation. The size of the population depends on the problem to be solved and the type of genetic operator that will be carried out [9]. Generate $N$ individuals at random to form the initial population $P(0)$, set the generation index $t = 0$, and determine the value of the maximal generation $T$ [11]. Each member of the population is the possible solution code of the problem discussed. After creating the initial population, each individual is evaluated and given a fitness value [29].

2.2.2. *Evaluation.* The process for evaluating each population by calculating the fitness value of each chromosome and evaluating it until the criteria are met [5]. The fitness function is an objective function of a problem that will provide value and determine the best or not the best solution [4]. After evaluation, individuals with better fitness will survive and take part in reproduction [18]. At this stage, the fitness value of each individual in the population $P(t)$ will be calculated [11].

2.2.3. *Selection.* The process of determining the individual to be selected for the crossover [5]. In this phase, the reproduction or selection operator will be applied to the $P(t)$ population [11]. This operator will select the chromosomes in the population for reproduction. The better chromosomes have the possibility to be selected for reproduction.

2.2.4. *Crossover.* The process of selecting two or more best individuals to cross to produce new individuals by exchanging their gene information [18]. Crossover is the phase in which two strings are recombined to get a better string by varying the chromosomes from one generation to the next [24]. The purpose of the crossover operation is to replicate the chromosomes [22].

2.2.5. *Mutations.* The process of changing the value of one or more genes in a chromosome so that a new individual is created by modifying one or more genes in the same individual [5]. The mutation is a gene modification to produce new individuals [15]. This process is done by replacing the gene values selected at random with a new value obtained is also randomized. In this phase, mutation operators will be applied to the population after crossover to create the new population $P(t+1)$ of the next generation $t+1$ [11].

2.3. *System development framework*

In this research, the system development method used is to adopt the System Development Life Cycle (SDLC) with the waterfall model, which consists of 4 stages as follows: [13]:

(a) Planning: making observations, collecting required lecture scheduling data, and determining the problem boundaries and conditions that must be determined in the lecture schedule.

(b) Analysis: analyze the previous lecture scheduling system using sample data on the lecture schedule, then determine the parameters to be used in the genetic algorithm.

(c) Design: designing a lecture scheduling system using the Unified Modeling Language (UML) based on the results of previous problem analysis. The next stage is designing the database requirements and user interface.

(d) Implementation: translating the results of the analysis and design of lecture scheduling into program code based on genetic algorithms and testing the results.

Figure 3 shows the framework for the system development carried out in this research with the steps taken to create a lecture scheduling application using a genetic algorithm.

![Figure 3. System development framework.](image)

### 3. Results and Discussion

#### 3.1. Specifying Input Data Variables

The data used as an example in this research is the lecture schedule data for the Informatics Engineering study program, even semester 2018/2019.

(a) Lecturer Data

The lecturer data contains the identity of the lecturer, the courses taught, and the willingness of the day to teach.

| ID      | Lecturer’s name | Teaching day readiness | Courses                                      |
|---------|-----------------|------------------------|----------------------------------------------|
| D001    | Ridho Taufiq S. | - Tuesday              | - Computer Architecture and Organization     |
|         |                 | - Thursday             | - Software Engineering 2                     |
|         |                 |                        | - Network Security                           |

Table 1. Example lecturer data and course teaching readiness.
D002 Kusnadi
- Monday - Operating System
- Tuesday - Multimedia Technology
- Thursday - Computer Security Systems

D003 Petrus Sokibi
- Monday - Data Structure
- Tuesday - Database Design
- Wednesday - Human and Computer Interaction

(b) Course Data
Course data consists of the name of the course and its type, the number of credits, and the semester.

| ID   | Code | Course name                        | Type of course     | Credit | Semester |
|------|------|------------------------------------|--------------------|--------|----------|
| M001 | 2099 | Data Structure                     | Theory             | 3      | II       |
| M002 | 2076 | Computer Architecture and Organization | Theory             | 3      | II       |
| M003 | 2029 | Software Engineering 2             | Theory             | 3      | IV       |

(c) Lecture Room Data
Lecture room data consists of room names, capacities, and room functions.

| ID   | Room | Capacity | Description |
|------|------|----------|-------------|
| R001 | 101  | 25       | Theory      |
| R002 | 102  | 25       | Theory      |
| R003 | 103  | 25       | Theory      |

(d) Class Data
Class data consists of the class name, semester, and the number of students.

| ID   | Class Name | Semester | Number of students |
|------|------------|----------|--------------------|
| K001 | TI-1/2     | 2        | 25                 |
| K002 | TI-2/2     | 2        | 25                 |
| K003 | TI-1/4     | 4        | 25                 |

(e) Group Data
Group data are a combination of class data, courses, and lecturers teaching in the classroom. The data must be determined before scheduling using genetic algorithms.

| ID   | Class ID | Course ID | LecturerID |
|------|----------|-----------|------------|
| G001 | K001     | M001      | D003       |
| G002 | K002     | M002      | D001       |
| G003 | K003     | M003      | D004       |

(f) Time Period Data
Time period data is the total number of periods that can be used for lectures [16]. After dividing the time, data is obtained as in Table 7 so that the available periods for one room are 16-time slots or 144 timeslots for nine rooms.

| Time | Time slots |
|------|------------|
| 08.00 - 09.00 | T001 | T004 | T007 | T010 | T013 | T015 |
The requirements or criteria that must be met for scheduling lectures in this research are:

1. One class cannot study in different rooms at a time.
2. One class cannot study more than one course at a time.
3. Lecturers cannot teach more than one course at a time.
4. Lecturers cannot teach more than one class at a time.
5. The class must be placed in a room according to its capacity [17, 26].
6. The lecturer must teach according to the willingness of the day specified.

It can be concluded that the genes used as input variables for genetic algorithms are:

- G: class, course and lecturer configuration data are called group data sets (G₀₀₁, G₀₀₂, ..., Gₙ)
- T: time slots or period data is available (T₀₀₁, T₀₀₂, ..., Tₙ)
- R: lecture room data sets (R₀₀₁, R₀₀₂, ..., Rₙ)

### 3.2. Genetic Algorithm Process

#### (a) Initial Population

![Population Formation Simulation](image)

The first stage of the genetic algorithm is to randomly create an initial population of 10 individuals. The chromosomes will be formed sequentially from genes G, T, R (group, period, lecture room) according to the number of G genes, while the values of the T and R genes will be randomly selected from each set of genes.

| Individual | Chromosome |
|------------|------------|
| 1          | G₀₀₁T₀₀₂R₀₀₂G₀₀₇G₀₀₂T₀₀₇R₀₀₂G₀₀₃T₀₀₉R₀₀₆G₀₀₄T₀₁₀R₀₀₆G₀₀₅T₀₁₁R₀₀₃G₀₀₆T₀₁₆R₀₀₉G₀₀₇T₀₁₄R₀₀₇G₀₀₈T₀₀₁R₀₀₄G₀₀₉T₀₁₀R₀₀₃G₀₁₀T₀₀₂R₀₀₂G₀₁₁T₀₀₇R₀₀₉G₀₁₂T₀₀₅R₀₀₅G₀₁₃T₀₁₆R₀₀₄G₀₁₄T₀₁₄R₀₁₀G₀₁₅T₀₀₆R₀₀₅G₀₁₆T₀₁₅R₀₁₀G₀₁₇T₀₀₇R₀₀₈G₀₁₈T₀₁₁R₀₀₉G₀₁₉T₀₁₆R₀₀₂G₀₂₀T₀₁₃R₀₁₁G₀₂₁T₀₀₆R₀₀₁G₀₂₂T₀₁₄R₀₀₇G₀₂₃T₀₀₇R₀₀₈G₀₂₄T₀₀₅R₀₀₈G₀₂₅T₀₀₆R₀₀₄G₀₂₆T₀₀₄R₀₀₃ |
| 2          | G₀₀₁T₀₀₃R₀₀₇G₀₀₂T₀₀₉R₀₀₆G₀₀₃T₀₀₄R₀₀₁G₀₀₄T₀₀₁R₀₀₃G₀₀₅T₀₁₆R₀₀₇G₀₀₆T₀₀₂R₀₀₉G₀₀₇T₀₁₃R₀₀₂G₀₀₈T₀₀₈R₀₀₇G₀₀₉T₀₀₂R₀₀₃G₀₁₀T₀₀₃R₀₁₀G₀₀₇G₀₁₁T₀₀₅R₀₀₅G₀₁₂T₀₁₆R₀₀₆G₀₁₃T₀₁₆R₀₀₆G₀₁₄T₀₀₁R₀₀₂G₀₁₅T₀₀₅R₀₀₅G₀₁₆T₀₁₃R₀₀₆ |
(b) Evaluation
Evaluation is carried out to calculate the fitness function of each individual (chromosome) in the population, where this function can be formulated as \( f = 1 / (1 + \text{penalty}) \), where \( \text{penalty} = \text{class clash} + \text{room clash} + \text{lecture clash} \) [12,21]:

**Table 8.** Example of fitness value calculation results.

| Individual | Chromosome                                      | Fitness     |
|------------|-------------------------------------------------|-------------|
| 1          | G001T002R007G002T007R002G003T009R006G004T010R006G005 T011R003G006T016R009G007T014R007G008T001R004G009T010 R003G010T002R002G011T007R009G012T005R005G013T016R004 G014T001R002G015T006R005G016T015R003G017T008R008G018 T011R009G019T016R002G020T013R001G021T006R001G022T014 R007G023T007R008G024T005R008G025T006R004G026T004R003 1 / (1 + 2 + +) = 0.33 |
| 2          | G001T003R007G002T009R006G003T004R001G004T001R003G005 T016R007G006T002R009G007T013R002G008T008R007G009T002 R003G010T003R007G011T006R008G012T001R006G013T016R006 G014T001R002G015T005R005G016T013R006G017T007R008G018 T008R008G019T015R001G020T002R001G021T009R005G022T016 R002G023T012R009G024T004R009G025T002R002G026T006R002 1 / (1 + 2 + +) = 0.33 |

(c) Selection
The selection process is the selection of individual pairs to be crossed. The process of selecting individuals for crossing is done randomly.

(d) Crossover
The crossover used is position-based as shown in Figure 5.

**Figure 5.** Simulation of the cross-process.

The crossing process is carried out on individuals who have been determined at the selection stage. This crossing process is carried out to form new individuals in the next generation. The crossing point limit is randomly determined. [27].

**Table 9.** Example of a population resulting from a crossover.

| Individual | Chromosome                                      |
|------------|-------------------------------------------------|
| 1          | G001T002R007G002T009R006G003T004R001G004T001R003G005T016R007G006 T016R009G007T013R002G008T008R007G009T002R003G010T003R007G011T006 R008G012T001R006G013T016R006G014T001R002G015T005R005G016T013R006 G017T007R008G018T008R008G019T015R001G020T002R001G021T009R005G022 T016R002G023T012R009G024T004R009G025T002R002G026T006R002 |
| 2          | G001T003R007G002T007R002G003T009R006G004T010R006G005T011R003G006 T016R009G007T014R007G008T001R004G009T010R003G010T002R002G011T007 R009G012T005R005G013T016R004G014T001R002G015T006R005G016T015R003 G017T008R008 |
(e) Mutation
The mutation process is carried out after the crossing process is complete. In this process, a gene on a chromosome will be replaced with new genes as shown in Figure 6.

![Figure 6. Example of the mutation process.](image)

The probability of an individual being mutated is 0.2, so the mutation process will only be carried out on a few individuals or even not at all. Stop Criteria is the stage of stopping the genetic algorithm. The genetic algorithm will be terminated when the fitness value of an individual is $f = 1$. Individuals who have a fitness value of 1 are a solution to problems.

3.3. Output Variable
The output variable is the individual or chromosome, which has a fitness value = 1, which is called the solution. This output variable is still in the form of a program code that will be translated into lecture schedules.

3.4. System Design

3.4.1. Use Case Diagram. The system specifications to be made are illustrated in the use case diagram as shown in Figure 7, where the user who plays a role in this system is the Academic and Student Administration Bureau (BAAK). The system created has functioned such as login and management: academic year, lecture room, courses, classes, lecturer, distribution of lecturer teaching, and lecture schedule.

![Figure 7. Use Case Diagram.](image)

3.4.2. User Interface Design. The interface design is used as a reference to make it easier to create an application layout view. Figure 8 is the user interface for the process of creating a lecture schedule using a genetic algorithm after entering the lecturer teaching data. Figure 9 is a display of the lecture schedule of the results of the process.
Figure 8. Display for the process of making lecture schedules with GA.

Figure 9. The results of the scheduling process using a genetic algorithm.

4. Conclusion and Suggestion

Based on the results of the research, it can be concluded that the application built able to conduct the scheduling process quickly without the absence of conflicting schedules based on the test results using data on the average number of days the lecturer willingness to teach 4.375, chromosomes length 99, and the number of generation 4 required completion time of 11.23 minutes. The variety of lecturer willingness to teach and the length of chromosomes indicates that the number of lectures will affect the speed of completion time. In this research, the time slot for each lecture is static, which is 3 hours, for further development can be created time slot based on the number of SKS per course, so that each lecture has the appropriate time slot.

References

[1] Ashari I A, Muslim M A and Alamsyah 2016 Comparison Performance of Genetic Algorithm and Ant Colony Optimization in Course Scheduling Optimizing Sci. Journal of Informatics 3 149
[2] Hamed B, Jaber K and Amin H 2015 A survey of Approaches for University Course Timetabling Problem Computers & Industrial Engineering 86 43
[3] Parera S, Sukmana H T and Wardhani L K 2016 Proc. Int. Conf. on Cyber and IT Service Management (Bandung) 36957 (USA: IEEE) pp 1–5
[4] Izah R. Ahmad et al 2018 J. Phys.: Conf. Ser. 995 012050
[5] Suwirmayanti N L G P, Sudarsana I M and Darmayasa S 2016 Implementation of Genetic Algorithm for Course Scheduling Journal of Applied Intelligent System 1 225–227
[6] Kristiadi D and Hartono R 2019 Genetic Algorithm for Lecturing Schedule Optimization (Case Study: University of Boyolali) IJCCS 13 84
[7] Wen-jing W 2018 Improved Adaptive Genetic Algorithm for Course Scheduling in Colleges and Universities iJET 13 29–31
[8] Kramer O 2017 Genetic Algorithm Essentials Genetic Algorithm Essentials vol 679, ed Polish Academy of Sciences (New York: Springer International Publishing) pp 11–19
[9] Gellysa Urva and Ari Sellyana 2019 J. Phys.: Conf. Ser. 1175 012042
[10] Chambers I 2001 The Practical Handbook of GA (Washington: Chapman & Hall/CRC)
[11] Sakawa M 2002 Genetic Algorithm and Fuzzy Multiobjective Optimization (US: Kluwer Academic Publishers, Springer) pp 11–27
[12] Zukhti Z 2014 Algoritma Genetika: Metode Komputasi Evolusien untuk Menyelesaikan Masalah Optimasi (Yogyakarta: Andi Publishers)
[13] Dennis A, Wixom B H, Tegarden D 2015 System Analysis and Design An Object-Oriented Approach with UML (USA: John Willey & Sons, Inc.) pp 2–5
[14] R Ansari and N Saubari 2020 IOP Conf. Ser.: Mater. Sci. Eng. 821 012043
[15] Mauluddin S, Iskandar I and Nursikuw A 2018 Optimasi Aplikasi Penjadwalan Kuliah Menggunakan Algoritma Genetik J RESTI 2 793
[16] Al-Majmar N A and Al-Shfaq T H 2016 Solving of Lectures Timetabling Problem and Automatic Timetable Generation using Genetic Algorithm IJARCE 5 505
[17] Jacobson L and Kanber B 2015 GA in Java Basics (New York: Apress) chapter 2 pp 105–106
[18] Timilsina S, Rohit N, Khurana Y and Jyotsna S 2015 Genetically Evolved Solution to Timetable Scheduling Problem IJCA 114 12-13
[19] Budhi G S, Gunadi K and Wibowo D A 2015 Proc. Int. Conf. on Soft Computing, Intelligence Systems, and Information Technology (Bali) vol 516 (Berlin, Heidelberg: Springer) pp 51–63
[20] Eka Mala Sari Rochman et al 2020 J. Phys.: Conf. Ser. 1569 020274
[21] M F Syahputra et al 2018 IOP Conf. Ser.: Mater. Sci. Eng. 308 012046
[22] A Syarif et al 2020 IOP Conf. Ser.: Mater. Sci. Eng. 857 012009
[23] Limkar S, Khalwadekar A, Tekale A, Mantri M and Chaudhari Y 2015 Proc. Int. Conf. on Frontiers of Intelligent Computing vol 327 (Cham: Springer) pp 774–775
[24] Zheng F Li J 2018 Proc. Int. Conf. on Advanced Hybrid Information Processing (Harbin) vol 219 (Cham: Springer) pp 498–499
[25] Pandey H M 2016 Proc. Int. Conf. Cloud System and Big Data Eng. (Noida) (USA: IEEE) pp 45
[26] Soyemi J, Akinode J and Oloruntoba S 2017 IEEE International Symposium on Dependable, Autonomic and Secure Computing (Orlando) (USA: IEEE) pp 711–712
[27] Chauhan P M, Parmar K B and Mendapara M B 2015 Proc. Int. Conf. on Circuits, Power and Computing Technologies (Nagercoil) (USA: IEEE) p 4
[28] Yadav S L and Sohal A 2017 Comparative Study of Different Selection Techniques in Genetic Algorithm Int. J. Sci. Math. Educ. 6 174
[29] Mittal D, Doshi H, Sunasra M and Nagpure 2015 Automatic Timetable Generation Using Genetic Algorithm IJARCE 4 245–246
[30] Bhosale V, Shastri S S and Khandare M A 2017 A Review of Genetic Algorithm Used for Optimizing Scheduling of Resource Constraint Construction Projects IRJET 4 2869