A Custom-based Crossover Technique in Genetic Algorithm for Course Scheduling Problem

Ni Gusti Ayu Harry Saptarini, Putu Indah Ciptayani, Ida Bagus Irawan Purnama

Electrical Engineering Department, Politeknik Negeri Bali, Indonesia

***Abstract* – Genetic algorithm is widely used in scheduling, which crossover is one of the important operators. This paper aims to propose a custom crossover technique in genetic algorithm to solve the course scheduling problem. This technique proposes only one offspring on each crossover by choosing the best gene from each parent. The result shows that the proposed technique can be applied to solve the scheduling problem which is better compared to the one-point and two-point crossover with the best fitness value 0.0049. Its best fitness after the convergence state is more stable than two other techniques.**

***Keywords* – combinatoric problem, course scheduling, custom crossover, genetic algorithm, timetable**

1. Introduction

Arranging an acceptable course timetable with certain satisfaction level is challenging. This scheduling task has to consider the availability of resources with many combinatoric constraints [1]. It must organize courses, teachers, time slot and rooms in an effective way. This aims to maximize the usability of resources allocation and minimizes the violation of constraints on the other hand. There are certain constraints that cannot be violated, namely hard constraints. These types of constrains are undeniable. If they are violated, the schedule will fail. Hence, they have to be placed in the first priority of the fitness function design. Inversely, there are soft constraints that can be violated but will affect the satisfaction of the users involved in the schedule, such as students and teachers. In common practices, manual scheduling consumes much time and may result in hard constraint violations or not satisfied soft constraints.

DOI: 10.18421/TEM91-53
https://dx.doi.org/10.18421/TEM91-53

Corresponding author: Putu Indah Ciptayani,
Electrical Engineering Department, Politeknik Negeri Bali, Indonesia
Email: putuindah@pnb.ac.id

Received: 16 November 2019.
Revised: 29 January 2020.
Accepted: 05 February 2020.
Published: 28 February 2020.

Some studies have been conducted to solve the course and class scheduling problem either in high school, college, or university. Studies [2], [3], [4] have used genetic algorithm (GA) to solve the course scheduling problem. Those studies found that the genetic algorithm can produce a good and effective course schedule. Other studies [5], [6], also used GA to the solve classroom scheduling problem. The first study proposed a heuristic approach to utilize the accessible classroom space for a given time table of courses, and the second study tried to overcome the limitations of the classroom with the schedule of lectures and a large number of students.

Meanwhile, [7] in his study used an improved adaptive genetic algorithm to solve course scheduling that involved teacher, time and course. The study found that this improved genetic algorithm was better than the standard genetic algorithm. Other research conducted by Xie [8] built a mathematical model for the scheduling and implemented it based on genetic algorithm. To ensure that the resulted schedule is optimal, some studies, [9], [10], [11], combined the genetic algorithm with taboo search. In particular, the study of [3] used taboo search for local optimization, while for global optimization, this study used genetic algorithm. All of the studies result in a schedule without clashes.

The aforementioned studies showed that genetic algorithm was widely used to solve the course scheduling problem. One of the important operators in the genetic algorithm is crossover. The right crossover operator will result in better performance of the genetic algorithm [12]. The study of [13] compared two crossover techniques for the timetabling problem which are the one-point and two-point crossover. The result shows that the two-point is better than the one-point crossover. Other study conducted by [14] used the crossover technique that exchanges some genes randomly to produce offspring in each iteration.

The fact that crossover is one of the important operators on genetic algorithm leads this study to search a better crossover technique to get a better course schedule. Instead of producing two offsprings, one offspring in each crossover which has the minimum soft constraint violation from both parents could be a good solution so that a further investigation is needed. Then, to measure its performance, the comparison to the widely used one-point and two-point crossover technique is also required.

Objectives of this paper are: (1) to propose a custom crossover technique for the course scheduling task using one best offspring approach, (2) to conduct experiments on the basis of the appropriate set of parameters such as population size, probability of crossover, probability of mutation, and selection size, (3) to investigate the performance of the proposed technique in terms of its best fitness, convergence speed, and steady-state behavior after reaching the convergence.

2. Material and methods

This section is started with data collection, then it presents all the methods and terms used in this study such as genetic algorithm, representation of chromosome, the fitness function, selection, crossover, mutation, repair, and elitism.

2.1. Data collection

The data collection techniques used in this study was documentation and interview. All of the data in this study that include hard and soft constraints are collected from SMA Surya Wisata, Jalan Wagimin, Kediri, Tabanan, Bali.

2.2. Genetic algorithm

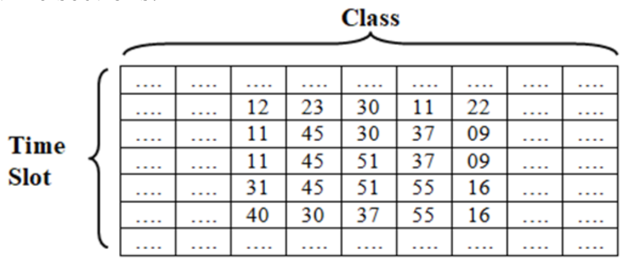
Genetic algorithm is one of the heuristic methods based on natural selection [15]. Here, the solution of the problem will be represented as a chromosome [16]. Based on [17], genetic algorithm conducts process as follows:

- Generating an initial population randomly. A population is a set of chromosomes. The possibility of higher fitness can be produced by the bigger size of the population [18].
- Calculating the chromosome fitness based on the fitness function.
- Reproducing new offspring by selection, crossover and mutation.

Selecting the best chromosome from the previous iteration and the offspring produced in the current generation, to be the population in the next iteration.

2.3. Representation of chromosome

The solution of the problem will be represented as a chromosome. Here, the chromosome is produced from the representation of the teacher and course for certain class and time slot. Each class has a fixed classroom. This study uses a 2-dimensional representation. The horizontal side (column) represents classes, while the vertical direction is for the time slot. In this case, the time slot will be the day (Monday – Saturday) and the detail will be time section for that day. One day consists of maximal 11-time sections.



....	....	....	....	....	....	....	....	....
....	....	12	23	30	11	22	....	....
....	....	11	45	30	37	09	....	....
....	....	11	45	51	37	09	....	....
....	....	31	45	51	55	16	....	....
....	....	40	30	37	55	16	....	....
....	....	....	....	....	....	....	....	....

Figure 1. The representation of chromosome

The intersection between class and time slot (a cell) will be a course and a teacher teaching the course. The genetic algorithm will randomly generate the value of the cell. Figure 1. is the chromosome representation for the problem. Each code in the cell represents the teacher and the course, for example, code 12 represents teacher Rudy and History lesson. Each teacher may have more than one code, depending on the number of courses that he teaches.

2.4. Fitness function and selection

Make Fitness function represents how good the schedule resulted in order to satisfy all of the soft constraints. The more soft constraints are violated, the lower fitness resulted. The goal of the algorithm was to minimize the soft constraint violations. Therefore, the fitness function is as shown in Equation (1).

$$fitness = \frac{1}{1+total\_pinalty} \tag{1}$$

Where total_penalty is the total score of soft constraint violations. The penalty for each soft constraint is shown in Table 1. The maximum value of fitness function was 1 if there is no violation of soft constraints.

Selection is one of the important genetic operators. Selection will choose a couple of parents to do a crossover, in order to produce offspring. The selection method used in this study was tournament selection. The tournament selection chooses k individuals from the populations and then rank all of those individuals based on their fitness. The best two individuals will be selected to do a crossover.

2.5. Crossover and mutation

Crossover is a process to recombine two parents. The result of the crossover is new offspring. The common crossover method was the one-point crossover. This technique generates a random number between 1 and chromosome length [19]. The first offspring will inherit the gene from the first parent before the cross point, and after the cross point from the second parent. Meanwhile, the second offspring will take the gene from the second parent before the cross point and the rest of the first parent [20].

This study used a custom crossover technique which will be described in a separate section. This study will also compare this proposed crossover technique with the one-point as well as the two-point crossover.

The mutation occurs based on the probability of mutation. Mutation generally happens in a very small chance [21]. The algorithm will generate a random number, if the number is smaller than the probability, then the mutation will happen. This study uses change technique to do mutation. Some genes will be selected and change by random value. The mutation aims to maintain the various chromosomes from one generation to the next generation. This is expected in order to avoid the algorithm being trapped in the local optima.

2.6. Repair and elitism

Repair is the extra step in this study. The crossover and mutation may result in hard constraint violations. This step will repair the chromosome, so the hard constraint violation can be avoided. The repair will be done using this pseudo code:

```
Repair_psudo_code
{
    For each gene in chromosome do
        If the gene violates the hard constraint
            Change the gene with an appropriate value
}
```

Elitism is the last step in each iteration. The algorithm will select the old population member and the offspring based on the fitness. Only the chromosome with higher fitness will be selected to do a crossover in the next iteration. The number of chromosomes selected will be the same amount of population number.

2.7. The proposed crossover technique

The basic crossover technique usually results in two offsprings in the crossover process. Meanwhile, the crossover technique proposed in this study customizes only one offspring in each crossover process. Each gene in the offspring will be taken from the first and the second parent based on the penalty for soft constraint 13 – 19 in Table 1. The lower penalty will be selected to form the part of the offspring.

The illustration of the proposed technique is shown in Figure 2. Other soft constraints will be ignored in this step, because for soft constraints 5 – 12, it needs to check the global schedule, while the crossover only takes part from the first and the second parent, and not the whole schedule from one parent.

Parent 1

….	….	….	….	….	….	….	….	….
….	….	15	11	17	20	13	….	….
….	….	15	11	23	20	12	….	….
….	….	11	15	23	31	12	….	….
….	….	23	15	11	45	36	….	….
….	….	23	20	11	45	37	….	….
….	….	….	….	….	….	….	….	….
Penalty 13-19	**….**	**17**	**20**	**5**	**0**	**6**	….	….

Parent 2

….	….	….	….	….	….	….	….	….
….	….	12	23	30	11	22	….	….
….	….	11	45	30	37	09	….	….
….	….	11	45	51	37	09	….	….
….	….	31	45	51	55	16	….	….
….	….	40	30	37	55	16	….	….
….	….	….	….	….	….	….	….	….
Penalty 13-19	**….**	**6**	**15**	**7**	**18**	**5**	….	….

Offspring

….	….	….	….	….	….	….	….	….
….	….	12	23	17	20	22	….	….
….	….	11	45	23	20	09	….	….
….	….	11	45	23	31	09	….	….
….	….	31	45	11	45	16	….	….
….	….	40	30	11	45	16	….	….
….	….	….	….	….	….	….	….	….
Penalty 13-19	**….**	**6**	**15**	**5**	**0**	**5**	….	….

Figure 2. The proposed crossover technique

Based on Figure 2., here the penalty value of soft constraint violations (13 – 19) was summed to be the total penalty in each column (class). The next step is to compare the value total penalty of parent 1 and parent 2 in each column (schedule from one class). The smaller penalty will be taken to build the part of the offspring. For example, in parent 1, the total penalty of column 3 is 17, while the second parent is 6, so the third column of the offspring will be taken from the third column of parent 2.

Using this technique, it is expected that the total penalty will decrease quickly from generation to generation. As shown in Figure 2., the total penalty of parent 1 is 48, parent 2 is 51, and the offspring is 31. This method guarantees that the violation of soft constraint 13– 19 in the offspring will be less than the parents, compared to the one-point crossover or the randomly selected crossover.

The flowchart of the proposed method is shown in Figure 3. The algorithm starts by calculating the constraint violation for constraint 13 – 19 in each class and then save all the violations to the penalty array for each parent. Each array element represents the penalty of each class. The next step is to compare the penalty of certain class from parent 1 to parent 2, and then the class with lower penalty will be selected to form the new offspring.

3. Results and discussion

This section starts with data collection, then it presents all the methods and terms used in this study such as genetic algorithm, representation of chromosome, the fitness function, selection, crossover, mutation, repair, and elitism.

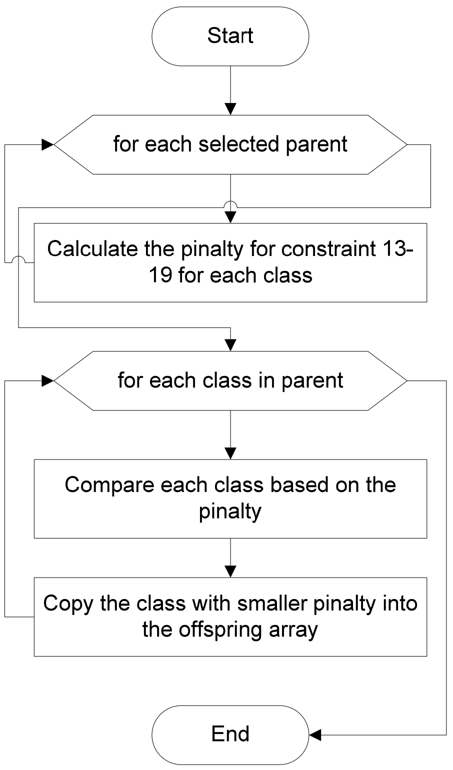

Start
for each selected parent
Calculate the pinalty for constraint 13-19 for each class
for each class in parent
Compare each class based on the pinalty
Copy the class with smaller pinalty into the offspring array
End

Figure 3. Flowchart for the proposed crossover technique

3.1. Constraints

All of the constraints were collected from SMA Surya Wisata, Jalan Wagimin, Kediri, Tabanan, Bali. Table 1. lists all of the hard and soft constraints with the penalty for each violation of soft constraints. In this case, the penalty uses a priority scheme defined by the school. On the other hand, there is no penalty for hard constraint, because the hard constraint cannot be violated. They are immutable and must be executed so that they have to be placed in the first priority.

There are four hard constraints which are (1) the course must be taught by the teacher in the associated field, (2) there must be no teacher clash in the same time slot which means each teacher can only teach one class at the same time, (3) the first section on Monday morning will be assigned for national ceremony, and (4) one course must be held in less than 4 time slots in a day for one class. All these hard constraints are listed with the number 1 – 4 in the table, while the rest, 5 – 19, are soft constraints which are fulfilled as much as possible if circumstances permit. Every school can formulate their own constraints according to their plans and objectives.

Table 1. Hard and Soft Constraints List

No	Constraint	Type	Penalty
1	The course must be taught by the teacher in the associated field	Hard	-
2	There must be no teacher clash in the same time slot	Hard	-
3	First section on Monday morning will be assign for national ceremony	Hard	-
4	One course must be held in less than 4 time slots in a day for one class	Hard	-
5	A teacher teaches 50 time slots or more	Soft	12
6	A teacher teaches 45 time slots or more	Soft	11
7	A teacher teaches 40 time slots or more	Soft	10
8	A teacher teaches 35 time slots or more	Soft	9
9	A teacher teaches 30 time slots or more	Soft	8
10	Uneven total time slot for all teachers in the same field	Soft	8
11	A teacher teaches 25 time slots or more	Soft	7
12	A teacher teaches 20 time slots or more	Soft	6
13	The morning sessions are completed over the 9 time slots	Soft	5
14	The morning sessions are completed over the 8 time slots	Soft	3
15	The afternoon sessions are completed over 9 time slots	Soft	5
16	The afternoon sessions are completed over 8 time slots	Soft	3
17	A class attends the same course for 3 time slots	Soft	6

18	Lesson hours with 2 section/week load broken down on different days	Soft	2
19	There are interludes of lessons in a class	Soft	1

The numbers of classes (a group of students) managed in this study were 27 classes which are 9 classes of first-year students, 9 classes of second-year students and 9 classes of third-year students. Some classes were held in the morning session while the rest were in the afternoon session. Each class has their fix classroom. The number of teachers to be managed was 61 teachers.

In one chromosome, there may be more than one violation for each soft constraint. For example, there may be ten teachers who teach more than 40 hours a week, or there may be four classes in the morning session completed over eight-time slots. The higher the number of violations will make the lower satisfaction.

The experiment conducted by using population size 50, the maximal generation 1000, the probability of crossover (pc) 0.8, the probability of mutation (pm) 0.1 and the k value for tournament selection was 10. The experiment was conducted 30 times for one-point crossover, two-point crossover, and the proposed crossover technique. Then, the results of the best fitness, convergence speed, and consistency after the convergence state are discussed in the next subsection.

3.2. *Best fitness*

The result of the experiment is shown in Figure 3. The experiment conducted by averaging the value of best fitness resulted by all 30 experiments for each crossover technique. The value will be averaged every multiple of 100 generations, then the iteration will be stopped when it has converged, or it has reached 1000 as the maximal generation. Only the offspring with the best fitness will be selected in accordance to define the best fitness average. This approach is different with [7] who tested a certain number of iterations for the experiment. Using the one-point crossover, he emphasized the use of adaptive genetic parameters, pc and pm, which adjust themselves with the fitness states at certain generations.

The experiment results demonstrate that the proposed crossover technique was more superior from the one-point and two-point crossover. On the 100th generations, the best fitness of the proposed method is already higher than the one-point and two-point crossover. The convergence state in the proposed crossover technique was faster than the one-point crossover but slightly slower than the two-point crossover. The proposed technique reaches the convergence state at 400th generation with the best fitness 0.0049, while the one-point at 700th with the best fitness value 0.0036 and the two-point at 300th with the best fitness value 0.004. In this case, the best fitness indicates the goodness of the solution where the higher fitness value will result in a more acceptable course schedule.

From the aforementioned results, the differences in the best fitness values among the three methods seem very small. For example, it is just 0.0009 between custom (0.0049) and two-point (0.004) crossover. However, since the fitness function acts on the basis of the total penalty, as formulated in formula (1), then the total penalty can be gained from the conversion of the best fitness value. Quantitatively, they will be equal to 203, 249, and 276 for custom, two-point, and one-point crossover respectively. This can be interpreted that the best fitness 0.0049 of the custom crossover is coming from 203 penalties of soft constraints. Similarly, the best fitness 0.004 of the two-point crossover is coming from 249 penalties of soft constraints. Thus, using the proposed crossover, there is a significant reduction of total penalties which are 73 penalties from the one-point and 46 penalties from the two-point crossover. This lessening clearly shows that choosing only the best gene from each parent for producing an offspring is a reasonable approach to reduce the total penalties.

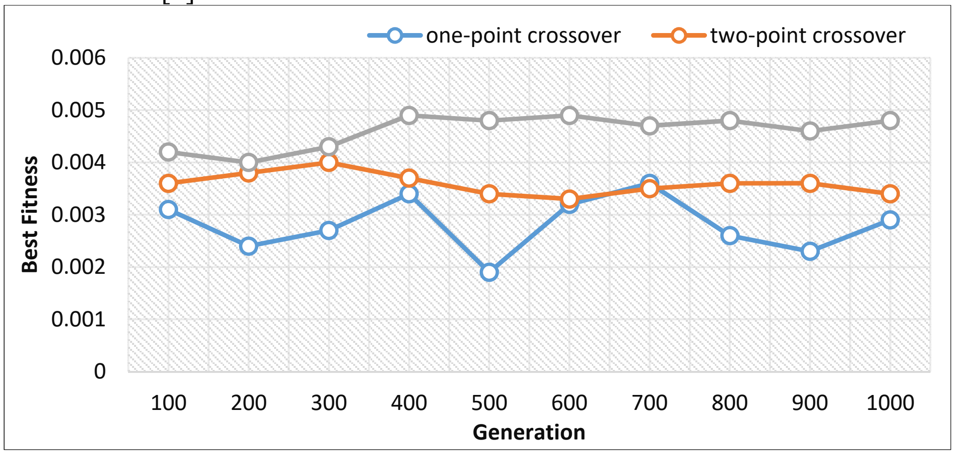


Figure 4. The experiment result of one-point and two-point crossover compared to the proposed crossover

3.3. Steady state behaviour

The superiority of the proposed method was as expected because the technique only chooses the genes with less soft constraint violations. This also explains how the behaviour of the best fitness is reached in the next generation after the convergence state. As shown in Figure 3., the proposed technique reaches almost stable fitness after the convergence state. This steady state is shown by no significant changes after the 400th generation. Furthermore, the exact similar best fitness value reoccurs at the 600th generations. On the other hand, the one-point crossover results in various values of the best fitness after the convergence state, because it just took the gene based on the randomly cutting-point. Similarly, for the two-point crossover, the consistency of its best fitness after the convergence state is not as constant as the proposed method.

4. Conclusion

This study has investigated the genetic algorithm with custom proposed crossover technique to solve the scheduling problem in high school. Instead of producing two offsprings in each crossover process, the proposed method produces only one offspring that takes the best value of each gene or the lowest number of penalty from both parents. This approach has well-designed consideration of soft constraints towards scheduling. The experiment shows that the proposed crossover technique can be used to solve the scheduling problem and was superior compared to the widely used one-point and two-point crossover technique. The convergence state of the proposed technique was reached at the 400th generation with the best fitness value 0.0049. This proposed technique gave a better result and faster convergence state compared to the one-point crossover, but it was a bit slower than the two-point crossover. In addition, it also had more stable fitness values after the convergence state. It is expected that this proposed solution can generate the course timetable which meets the satisfaction level of the users involved in the schedule. Hence, a further evaluation process of the satisfaction level is needed. In order to enhance the solution, the future study will combine the improved adaptive genetic algorithm [7] and the proposed crossover technique in this study, to find out whether this combination will result in better fitness.

Acknowledgements

Authors wish to acknowledge that this research is funded by Directorate General of Indonesia Higher Education, Ministry of Higher Education and Research and Technology, Republic of Indonesia. This research also supported by SMA Surya Wisata, Tabanan, Bali, Indonesia to provide data and conducting the testing phase.

References

[1]. Kohshori, M. S., & Abadeh, M. S. (2012). Hybrid genetic algorithms for university course timetabling. *International Journal of Computer Science Issues (IJCSI)*, *9*(2), 446.

[2]. Budhi, G. S., Gunadi, K., & Wibowo, D. A., (2015), Genetic Algorithm for Scheduling Courses, *In: Intan R., Chi CH., Palit H., Santoso L. (eds) Intelligence in the Era of Big Data. ICSIIT 2015. Communications in Computer and Information Science, vol 516.* Springer, Berlin, Heidelberg (pp. 51-63)

[3]. Ni, J., & Yang, N. N. (2013). Genetic algorithm and its application in scheduling system. *Telkomnika Indonesian Journal of Electrical Engineering*, *11*(4), 1934-1939.

[4]. Ming, H., & Qi, C. (2010, June). Course scheduling system design and implementation based on genetic algorithm. In *2010 International Conference On Computer Design and Applications* (Vol. 3, pp. V3-611). IEEE.

[5]. Kumar, S. V., Karthik, J., & Rao, D. S. (2018). A Computational Heuristics Approach For Classroom Scheduling Using Genetic Algorithm Technique. *International Journal of Pure and Applied Mathematics*, *119*(14), 167-172.

[6]. Parera, S., Sukmana, H. T., & Wardhani, L. K. (2016, April). Application of genetic algorithm for class scheduling (Case study: Faculty of science and technology UIN Jakarta). In *2016 4th International Conference on Cyber and IT Service Management* (pp. 1-5). IEEE.

[7]. Wen-jing, W. (2018). Improved Adaptive Genetic Algorithm for Course Scheduling in Colleges and Universities. *International Journal of Emerging Technologies in Learning (iJET)*, *13*(06), 29-42.

[8]. Xie, F. (2011, August). The Mathematical Model of Course Scheduling and Its Realization Based on Genetic Algorithm. In *2011 International Conference on Intelligence Science and Information Engineering* (pp. 374-377). IEEE.

[9]. Sonawane, M. P. A., & Ragha, L. (2014). Hybrid genetic algorithm and TABU search algorithm to solve class time table scheduling problem. *International Journal of Research Studies in Computer Science and Engineering*, *1*(4), 19-26.

[10]. OuYang, Y., & Chen, Y. (2011, August). Design of automated Course Scheduling system based on hybrid genetic algorithm. In *2011 6th International Conference on Computer Science & Education (ICCSE)* (pp. 256-259). IEEE.

[11]. Sutar, S. R., & Bichkar, R. S. (2017). High school timetabling using tabu search and partial feasibility preserving genetic algorithm. *International Journal of Advances in Engineering & Technology*, *10*(3), 421.

[12]. Umbarkar, A. J., & Sheth, P. D. (2015). Crossover operators in genetic algorithms: a review. *ICTACT journal on soft computing*, *6*(1).

[13]. Obaid, O. I., Ahmad, M., Mostafa, S. A., & Mohammed, M. A. (2012). Comparing performance of genetic algorithm with varying crossover in solving examination timetabling problem. *J. Emerg. Trends Comput. Inf. Sci*, *3*(10), 1427-1434.

[14]. Hariyadi, H. P., Widiyaningtyas, T., Arifin, M. Z., & Sendari, S. (2016, November). Implementation of Genetic Algorithm to academic scheduling system. In *2016 IEEE Region 10 Conference (TENCON)* (pp. 2013-2016). IEEE.

[15]. Ambole, R. H., & Hanchate, D. B., (2013), Class Timetable Scheduling with Genetic Algorithm, *International Journal of Computer Science and Technology (IJCST). 2013 Oct;4(4)* (pp. 371-375).

[16]. Sivasankar, S., Nair, S., & Judy, M. V. (2015). Feature reduction in clinical data classification using augmented genetic algorithm. *International Journal of Electrical and Computer Engineering*, *5*(6), 1516-1524.

[17]. Yohanes, B. W., Handoko, H., & Wardana, H. K. (2013). Focused crawler optimization using genetic algorithm. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, *9*(3), 403-410.

[18]. Kotimah, Q., Mahmudy, W. F., & Wijayaningrum, V. N. (2017). Optimization of Fuzzy Tsukamoto Membership Function using Genetic Algorithm to Determine the River Water. *International Journal of Electrical & Computer Engineering (2088-8708)*, *7*(5).

[19]. Sapru, V., Reddy, K., & Sivaselvan, B. (2010, December). Time table scheduling using genetic algorithms employing guided mutation. In *2010 IEEE International Conference on Computational Intelligence and Computing Research* (pp. 1-4). IEEE.

[20]. Abdullah, S., & Turabieh, H., (2008). Generating University Course Timetable Using Genetic Algorithms and Local Search, *in Third 2008 International Conference on Convergence and Hybrid Information Technology, 2008,* (pp.254-260).

[21]. Mitchell, M. (1999). *An Introduction To Genetic Algorithm*. A Bradford Book The MIT Press, Fifth printing.