

# Development of a modified genetic method for automatic university scheduling

Ievgen Fedorchenko, Andrii Oliinyk, Tetiana Zaiko, Kyrylo Miedviediev, Yuliia Fedorchenko and Mykola Khokhlov

National University "Zaporizhzhia Polytechnic", 64 Zhukovsky Str., Zaporizhzhia, 69063, Ukraine

## Abstract

The paper investigates the problem of optimizing the university class schedule. Sequential and parallel methods for scheduling based on genetic search are developed. The proposed methods use adapted initialization, crossover, and selection operators. The algorithms minimize conflicts and the time interval between classes, taking into account recommendations for time and place. The developed methods contribute to effective planning of the educational process and avoidance of errors in scheduling. A comparative analysis of the classical and modified genetic algorithms is carried out, confirming the faster and more efficient functioning of the modified approach. The modified algorithm is also compared with different operators and parameters of the genetic algorithm to determine the optimal conditions. The obtained results indicate effective methods for improving the quality of the schedule and optimizing the educational process at the university.

## Keywords

genetic algorithm, schedule, evolutionary algorithm, classes, classes

## 1. Introduction

Time and resource management is critical to success in higher education. For students, effective time management is an important factor for successful studies, allowing them to balance studies with other activities such as work, sports or social life. On the other hand, for teachers, an optimal schedule of classes helps increase the productivity and efficiency of their work.

Scheduling is a complex task that requires a lot of resources and planning knowledge. Since there are many constraints, such as following the curriculum and taking into account the needs of students and teachers, automating this process is essential. The scheduling problem can be classified as an NP-complete problem due to the large number of possible solutions, and metaheuristic algorithms are used to solve it. This paper proposes a modified genetic algorithm for efficient university scheduling.

---


*CS&SE@SW 2023: 6th Workshop for Young Scientists in Computer Science & Software Engineering, February 2, 2024, Kryvyi Rih, Ukraine*

✉ [evg.fedorchenko@gmail.com](mailto:evg.fedorchenko@gmail.com) (I. Fedorchenko); [olejnikaa@gmail.com](mailto:olejnikaa@gmail.com) (A. Oliinyk); [nika270202@gmail.com](mailto:nika270202@gmail.com) (T. Zaiko); [kirillmedvedev279@gmail.com](mailto:kirillmedvedev279@gmail.com) (K. Miedviediev); [jul.fedorchenko@gmail.com](mailto:jul.fedorchenko@gmail.com) (Y. Fedorchenko); [khokhlov@zp.edu.ua](mailto:khokhlov@zp.edu.ua) (M. Khokhlov)

🆔 0000-0003-1605-8066 (I. Fedorchenko); 0000-0002-6740-6078 (A. Oliinyk); 0000-0003-1800-8388 (T. Zaiko); 0000-0003-3200-7306 (K. Miedviediev); 0000-0003-4436-3877 (Y. Fedorchenko); 0000-0001-8272-9847 (M. Khokhlov)



© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

## 2. Literature review and problem statement

Implementation of automatic systems for drawing up university timetables is an urgent task, that is of great importance for improving the management of the educational process. Over the last decades, many studies and developments have been carried out in order to optimize the process of the distribution of working time and resources in university educational institutions.

Lukas et al. [1] use a genetic algorithm and a heuristic search to solve the problem of scheduling at a university. The schedule creation methodology is based on genetic algorithms, which are aimed at maximizing the number of successfully planned lesson units in the schedule. This article provides a detailed description of the timetable generation process and takes into account various constraints, such as the availability of teachers, classrooms and days.

Abduljabbar and Abdullah [2] uses the method of genetic algorithms to solve the problem of a complex schedule of classes at a university or college. The authors proposed a system where complex aspects such as class schedules, lecture times, and available classrooms are encoded as binary sequences, and based on these, solutions are generated using genetic operations such as selection, crossover, and mutation.

Alghamdi et al. [3] examines the problem of scheduling classes in universities and explores various methods for its optimization. The article provides an overview of the genetic algorithm optimization method. Genetic algorithms are based on the principles of genetics and natural selection. The basic idea is to select the fittest individuals in a population, which are then recombined or mutated to create new groups. GAs can effectively solve optimization problems that have different parameters or characteristics that cannot be represented mathematically.

The task of creating a schedule is to create an optimal schedule of classes, taking into account various restrictions and requirements. This issue belongs to the class of NP-complete problems because it is a combinatorial optimization problem in which the number of solutions increases with the size of the input data, constraints, and parameters.

The objective function for the scheduling problem defines the main optimization criteria, which include:

- minimizing the number of windows between classes to reduce the downtime of teachers and students;
- uniform distribution of the load during the week to increase work efficiency;
- minimization of conflicts and overlaps of classes, which involves avoiding overlap between classes with joint groups, teachers or classrooms;
- taking into account proposals regarding the schedule to meet the needs of students and teachers.

The objective function calculates the sum of the penalty functions that indicate violations of the constraints when creating the schedule. Each penalty function can have a weight depending on the importance of the corresponding constraint. We use the following notations:  $n$  is the number of groups,  $m$  is the number of teachers,  $k$  is the number of audiences,  $L$  is the number of classes,  $D$  is the number of school days,  $U$  is the maximum number of classes per day,  $Y$  is a type of couple,  $G = [g_1, g_2, \dots, g_n]$  is schedule relative to groups,  $T = [t_1, t_2, \dots, t_m]$  is timetable for teachers,  $A = [a_1, a_2, \dots, a_k]$  is the timetable for the classrooms,  $S = [s_1, s_2, \dots, s_l]$  is general schedule,  $X$  is the schedule of a separate group, teacher or classrooms.

$f_1(x)$  is a function that calculates windows between classes and has a corresponding weighting factor  $k_1$ . The function calculates the difference between the next class and the previous one within one day. Depending on the type of classes, the following differences will be calculated: between two common classes, or between the common and the numerator or denominator, and between two classes in the numerator or denominator. In general, this function can be described as follows (1):

$$f_1(x) = k_1 \sum_{d=1}^D \sum_{u=1}^U \sum_{y=1}^Y x_{d,u,y} - x_{d,u-1,y}, \quad (1)$$

$f_2(x)$  is a function that calculates the overlap of classes and has the corresponding weighting coefficient  $k_2$ . This function checks whether classes in a certain group or audience do not overlap. An overlap is considered when two or more classes are scheduled for the same day of the week and class number, or different classes are held in the same auditorium. It should be noted that if two classes are held in the same time slot, the class type is the numerator and the denominator, respectively, then this is not considered overlapping. Mathematical model of function (2):

$$f_2(x) = k_2 \sum_{i=1}^L \sum_{j=i+1}^L w_2(x_i, x_j) \quad (2)$$

$w_2(x_i, x_j)$  is a Boolean function that can be described as follows (3):

$$w_2(x_i, x_j) = \begin{cases} 1, & \text{if } i=j. \\ 0, & \text{else.} \end{cases} \quad (3)$$

$f_3(x)$  is a function that calculates compliance with the recommendations regarding the time of the lesson and has a corresponding weighting factor  $k_3$ . The objective function will look like this (4):

$$F(S) = \sum_{i=1}^2 \sum_{j=1}^n k_i f_i(g_j) + \sum_{i=1}^2 \sum_{j=1}^m k_i f_i(t_j) + k_2 \sum_{j=1}^k f_2(a_j) + k_3 \sum_{j=1}^l f_3(s_j) \quad (4)$$

With the help of weighting coefficients, we can determine the importance of compliance with the corresponding restriction. If the value of the coefficient is high, the priority in the algorithm will be to minimize this limitation.

Planning training sessions is a complex combinatorial task, and traditional scheduling methods are not suitable for it. They use heuristic methods, such as genetic algorithms, which, although they do not guarantee the best solution, effectively search for optimal options.

During the analysis of the available methods of solving the problem, it was established that the most suitable algorithm for drawing up the optimal schedule of classes is the genetic algorithm. Therefore, it was decided to develop an adapted and modified genetic algorithm for solving the given problem.

The scheduling algorithm should minimize the value of the defined objective function (4). The work of the algorithm can be represented in the form of the following formula (5):

$$F(S) \rightarrow \min, \quad (5)$$

where  $S$  is the class schedule;  $F(S)$  is the objective function.

To evaluate the quality of the results of the algorithm, the objective function described above will be used, which shows the number of violations of the specified restrictions.

### 3. The purpose and objectives of the research

The *purpose of the research* is to develop a mathematical model for solving the task of creating an optimal schedule of classes for higher educational institutions.

To achieve this purpose, it was necessary to solve the following tasks:

- to develop an adapted classic genetic algorithm for drawing up an optimal class schedule;
- to develop a modified method of the classical genetic algorithm;
- to present an experimental study of the proposed genetic methods.

### 4. Development of genetic algorithm modification

An evolutionary genetic method was chosen to optimize the objective function. The genetic algorithm gradually approaches the optimal schedule by selecting, combining, and varying possible solutions.

The main steps of the genetic algorithm are [4, 5]:

- 1) initialization of input data and algorithm parameters;
- 2) initialization of the initial population;
- 3) calculation of the adaptability of individuals;
- 4) crossing of individuals;
- 5) mutation of individuals in the population;
- 6) calculation of the suitability of the obtained individuals;
- 7) selection;
- 8) continuation of steps until the stop criterion is satisfied.

The initial data for the system was obtained from the semester assignment information from the software department. This document included a list of subjects, a list of groups attending these subjects, the names of lecturers who give lectures or practical classes, and recommendations for the classrooms needed for the classes. The task was to create a schedule for 28 teachers and 40 groups, using 10 classrooms (6 lecture rooms and 4 laboratory rooms).

The input data for the algorithm were:

- a list of classes, each of which included a teacher or teachers, one or more groups, a subject, a type of class, and the number of meetings;
- recommended schedule for classes;

- the number of school days;
- the maximum number of classes per day;
- list of classrooms.

The algorithm has the following parameters;

- population size;
- the number of iterations;
- probability of mutation;
- probability of crossing;
- probability of gene mutation.

Initialization in the genetic algorithm is the process of creating an initial population of individuals that will evolve in the future [6].

Two initialization methods were developed.

The first method is random initialization. Random initialization makes it possible to include various solutions in the initial set, which increases the probability of finding an optimal schedule. For each lesson, the day of the week, session number, type and audience are randomly determined. If the class has a recommended schedule, it will be taken into account [6].

The second method is the initialization of the schedule, modification of the random one: we randomly generate a day of the week, choose an audience, then look for a possible time of the class. If this is the first lesson on the selected day, you can insert it randomly. If there is already a class, then the number is selected in such a way as to minimize violation of the restriction. As a result, we will get a schedule in which there will be no overlap of classes and a reduced number of windows. Using this initialization method will significantly speed up the work of the genetic algorithm.

Crossover is an operation in a genetic algorithm that is used to create a new population. Crossbreeding consists in the exchange of genes between two parental individuals in order to create offspring with a new combination of genes [6].

It was decided to exchange only one random activity in two random schedules. This allows other genes to change in the next generation and preserve beneficial combinations.

The  $k$ -point crossover method involves choosing  $k$  points on the parental chromosomes, dividing the chromosomes into  $k + 1$  segments, and exchanging these segments to create two offspring. An even multiple of  $k$  ensures the division of segments into equal parts [6].

Advantages of this method include diversity in offspring, preservation of useful genetic combinations from parents, and prevention of collapse into a local minimum during optimization. However, a large value of  $k$  can lead to the loss of useful information from parental chromosomes [6].

Mutation randomly changes one or more genes in a certain solution to increase diversity and avoid local optima [6]. It is important not to change too many genes in one solution, as this may render it unusable. Therefore, the mutation is often limited to the change of only one random activity in the schedule [6].

Mutation of all genes with a certain probability is used to get out of local optima and find more optimal solutions [6].

In the context of a scheduling problem, a population represents different schedule options. A crossover involves changes in class numbers, days, or classrooms between two different schedule variants for a particular group or instructor. Mutation makes similar random changes in an individual schedule.

In the genetic algorithm, selection is the process of selecting the best solutions for forming the next population. The essence of selection is to keep the best individuals and get rid of the worst ones, which leads to a gradual improvement of decisions in each subsequent iteration. The following selection methods were implemented [6]:

- *tournament*: two decisions are randomly chosen, priority is given to the decision with a higher fitness value;
- *ranking*: a selection method based on their suitability ranking;
- *roulette*: selection of candidates of the next generation based on probabilities corresponding to their fitness.

An individual's fitness determines his or her ability to effectively solve a task or meet certain optimality criteria. It is a numerical value that reflects how well a particular individual meets the requirements.

In the case of scheduling, an individual's fitness can be measured by the number of conflicts in the schedule, the efficiency of resource use, and the satisfaction of students' and teachers' needs. In its algorithm, the function takes into account such strict constraints as overlapping class schedules, inconsistencies with the recommended schedule, and the number of breaks between classes.

Scaling of the adaptability of individuals was implemented in selection. Scaling of fitness values is one of the optimization methods of genetic algorithms, which allows to increase the speed of convergence, unit/s of the algorithm. The basic idea is to rescale the fitness values so that they lie in the range from  $a$  to  $b$ . This can be done using a scaling function that transforms the original range of fitness values into a new range corresponding to the range from  $a$  to  $b$  [7].

Scaling of the fitness of individuals helps to reduce the influence of different scales of fitness functions on the results of the genetic algorithm and allows to more efficiently find the optimal solution [7].

Also, to increase the chances of finding an optimal solution, you can use elitism, which preserves a part of the best individuals from one generation to another without changes. The preservation of elite individuals from the previous population allows to preserve diversity and prevent the loss of useful information [8].

Criteria for stopping the algorithm:

- if the maximum number of iterations is reached;
- if a schedule is found, the fitness value of which is equal to 0.

One of the methods for improving the classical genetic algorithm is its island model. The GA island model is a model in which the population is divided into several groups (islands). Each island contains its own subpopulation that evolves independently of other islands [9].

With the help of the migration mechanism, individuals can move from one island to another to speed up the convergence of the algorithm. This will help reduce the risk of falling into local minima [9].

In the island model, subpopulations can use different algorithm parameters, different crossing, mutation and selection operators [9].

Also, the genetic algorithm, like its island model, can be speeded up by parallel implementation. Parallel implementation of GA will speed up the convergence time of the algorithm using several processors [10].

## 5. Software implementation of the developed modification

The main idea of the genetic algorithm is to gradually create and improve schedules using operators such as mutation, crossover, and selection. The basic steps of a genetic algorithm include initialization, main cycle, crossover, mutation, calculation of fitness values, selection, termination, and saving the results.

The island model of the genetic algorithm is similar to the classical genetic algorithm, but involves the creation of separate groups of populations that function in parallel and can interact by exchanging individuals.

The island model of the genetic algorithm includes such operations as the initialization of individual islands with different parameters and the possibility of migration of individuals between these islands.

The main steps of the GA island model include initialization, initialization of schedules, main loop, island traversal, crossover, mutation, calculation of fitness values, selection, migration (if necessary), selection of best individuals, search of best schedules, replacement of individuals, termination, search of best individual, completing the algorithm and saving the results.

## 6. Experiments and results

Input data for the system are:

- *data on classrooms*: classroom name, type, capacity, established departments;
- *data on departments*: name of the department and its abbreviated name;
- *data on specialties*: specialty name, code, department;
- *data on disciplines*: name, specialty, semester in which the discipline is taught;
- *group data*: group name, number of students, current semester, specialty;
- *data on teachers*: full name of the teacher and the department where he works;
- *data about classes*: discipline, class type, number of classes per week, teachers, groups, recommended audiences, recommended time.

The initial data is the optimal schedule of classes compiled for the selected department, which takes into account the recommendations for conducting and minimizes the following parameters:

- the number of overlapping classes for groups, teachers and classrooms;
- the number of windows between classes for groups and teachers.

A computer with the following characteristics was used for the tests: processor AMD Ryzen 5 2600 with a frequency of 3.9 GHz, 6 cores and 6 threads, the amount of RAM is 8.00 GB with a speed of 3200 MHz, the type of hard disk is SSD, and the operating system is the Virtual Machine Windows Server 2022.

The speed of convergence parameter was used to compare the performance of the algorithm. The speed of convergence shows how quickly or efficiently the algorithm approaches its optimal solution. This parameter can be calculated using the following formula (6):

$$S = \frac{|F_s - F_f|}{|t_f - t_s|} \quad (6)$$

where  $S$  is the speed of convergence;  $F_s$  is the initial fitness value;  $F(f)$  is the final value of fitness;  $t_s$  is the initial time of the algorithm;  $t_f$  is the final time.

The following abbreviations should also be noted:  $P_e$  is the value of elitism, i.e. the percentage of the total population;  $T_c$  is type of crossing: 1 – “custom one gene”, 2 – “ $k$ -point”;  $N_k$  is the number of  $k$  crossing points;  $P_c$  is the probability of crossing;  $T_m$  is mutation type: 1 – “custom one gene”, 2 – “all genes”;  $P_{mg}$  is probability of gene mutation;  $P_m$  is probability of mutation;  $T_s$  is selection type: 1 – “roulette”, 2 – “ranging”, 3 – “tournament”;  $T_i$  is initialization type: 1 – “random”, 2 – “simple algorithm”;  $N_g$  is the maximum number of iterations of the algorithm;  $N_p$  is the size of the population.

The following parameters were used for the tests: number of days for making a schedule – 6, number of classes for making a schedule – 6, penalty for windows in groups – 1, penalty for windows in teachers – 1, penalty for overlapping classes – 5, and penalty for inconsistency recommended class time – 5.

Table 1 lists the test run parameters for the genetic algorithm, including the population size (500) and the maximum number of iterations (2000). According to table 1, the results of these tests are given. It is important to note that each entry in table 2 represents the average value of the results of three conducted experiments with the same parameters.

Genetic algorithm parameters, such as population size and maximum number of iterations, were kept constant for all tests to ensure comparability of results between different experiments.

Table 3 presents the parameters used during tests of the genetic algorithm modification – the island model. Table 4 contains the results of these tests, and each test was performed three times, after which the average value of the results was calculated. All tests used the same parameters: a population size of 500 and a maximum number of iterations of 2000.

The following designations should be entered:  $N_i$  is the number of islands;  $N_{st}$  is a step of increasing the values in the islands;  $N_{it}$  is the number of iterations through which to migrate between islands;  $P_{mig}$  is the number individuals participated in migration.

## 7. Discussion of research results

Analyzing table 2, we can draw the following conclusions about the highest convergence rate for the GA parameters: mutation probability – 0.2; mutation type – single gene mutation; crossover probability – 0.6; elitism value – 0.2; sampling type – roulette; crossover type – single gene crossover.



**Table 1**

Parameters for running genetic algorithm tests.

N <sup>o</sup>	$P_e$	$T_c$	$N_k$	$P_c$	$T_m$	$P_{mg}$	$P_m$	$T_s$	$T_i$
1	0.1	1	-	0.7	1	-	0.2	1	1
2	0.1	1	-	0.7	1	-	0.4	1	1
3	0.1	1	-	0.7	1	-	0.6	1	1
4	0.1	1	-	0.7	2	0.05	0.2	1	1
5	0.1	1	-	0.7	2	0.1	0.2	1	1
6	0.1	1	-	0.7	2	0.2	0.2	1	1
7	0.1	1	-	0.4	1	-	0.2	1	1
8	0.1	1	-	0.6	1	-	0.2	1	1
9	0.1	1	-	0.8	1	-	0.2	1	1
10	0.2	1	-	0.6	1	-	0.2	1	1
11	0.3	1	-	0.6	1	-	0.2	1	1
12	0.1	1	-	0.6	1	-	0.2	2	1
13	0.1	1	-	0.6	1	-	0.2	3	1
14	0.1	2	2	0.6	1	-	0.2	3	1
15	0.1	2	4	0.6	1	-	0.2	3	1
16	0.1	2	6	0.6	1	-	0.2	3	1
17	0.1	1	-	0.6	1	-	0.2	3	2
18	0.1	1	-	0.6	1	-	0.4	3	2
19	0.1	1	-	0.8	1	-	0.2	3	2
20	0.1	1	-	0.6	2	0.1	0.2	3	2
21	0.3	1	-	0.6	1	-	0.2	3	2
22	0.1	2	4	0.6	1	-	0.2	3	2

To analyze the effect of the parameter  $T_i = 2$  (using a modified initialization parameter), we calculate how much the algorithm's running time has decreased on average compared to the values of  $T_i = 1$ . To do this, let's calculate the average value of the algorithm execution time for both variants.

The average value of the GA running time at  $T_i = 1$  is 1888.429 s.

The average value of the GA running time at  $T_i = 2$  is 1115.938 s.

The reduction in the running time of the GA algorithm with  $T_i = 2$  compared to  $T_i = 1$  is approximately 40.9%. A 40.9% reduction in running time indicates an improvement in algorithm performance by using  $T_i = 2$ .

After analyzing table 4, we can draw the following conclusions about the fastest convergence rate for the island model parameters: number of islands – 6 (number of logical processor cores); parameter divergence step – 7; number of iterations to perform migration – 10; number of individuals to participate in migration – 0.1.

The average running time of the island model at  $T_i = 1$  is 2335.027 s.

The average running time of the island model at  $T_i = 2$  is 1313.531 s.

When using the modified initialization method, the island GA model is on average 1.8 times faster. This can improve the performance of the algorithm, which saves time and resources in solving the problem.

**Table 2**  
Genetic algorithm results.

N <sup>o</sup>	$t_s, s$	$t_f, s$	$F_s$	$F_f$	$S$
1	0.983	1937.365	1128.667	22.000	0.577
2	0.968	1852.033	1157.333	24.667	0.613
3	0.981	1884.862	1148.000	119.000	0.546
4	0.991	1840.709	1178.667	75.000	0.600
5	0.964	1865.484	1164.333	80.333	0.581
6	1.008	1881.805	1156.667	54.333	0.586
7	0.953	1851.585	1183.667	22.000	0.628
8	0.952	1796.640	1165.000	17.667	0.639
9	0.966	1808.654	1164.000	15.000	0.636
10	0.960	1794.841	1158.333	14.000	0.638
11	0.979	1840.432	1148.333	18.000	0.615
12	0.983	1878.421	1165.333	10.000	0.615
13	1.003	1839.763	1155.333	14.667	0.620
14	1.023	1980.765	1159.000	15.667	0.578
15	1.135	2196.617	1156.667	10.000	0.522
16	1.023	1980.765	1159.000	15.667	0.578
17	1.116	947.481	8.333	1.333	0.014
18	0.922	1334.211	9.667	1.333	0.008
19	0.914	1320.637	9.667	2.000	0.009
20	0.929	680.211	10.333	2.667	0.076
21	1.021	859.483	14.667	1.000	0.019
22	1.075	1559.583	6.333	1.333	0.016

Let's test the algorithms with the parameters that give the best results (table 5).

## 8. Conclusion

A modified genetic method has been developed that uses an initialization operator based on a priori information about the learning process that is available from the given constraints. The use of the developed approach to initializing the genetic method can significantly (several times) reduce the search time.

Also, a modified island model of the developed genetic method was developed to solve the problem of drawing up an optimal schedule of classes. The fundamental difference between the proposed method and existing analogues is the use of a modified initialization operator that tries to reduce the initial fitness value by simply searching through possible options. Using the modified initialization operator, the running time of the classical genetic algorithm was reduced by 40.9%. It is also worth noting that the running time of the island model of the genetic algorithm with the modified initialization operator was reduced by 1.8 times compared to the use of the classical initialization method. This means that the application of this method has significantly saved the algorithm execution time. The modified initialization method improves the performance and speed of the genetic algorithm for scheduling classes, which is important

**Table 3**

Parameters for launching genetic algorithm modification tests – island model.

№	$P_e$	$T_c$	$P_c$	$T_m$	$P_m$	$T_s$	$T_i$	$N_i$	$N_{st}$	$N_{it}$	$P_{mig}$
1	0.1	1	0.4	1	0.2	1	1	6	2	10	0.1
2	0.1	1	0.4	1	0.2	1	1	12	2	10	0.1
3	0.1	1	0.4	1	0.2	1	1	18	2	10	0.1
4	0.1	1	0.4	1	0.2	1	1	6	3	10	0.1
5	0.1	1	0.4	1	0.2	1	1	6	5	10	0.1
6	0.1	1	0.4	1	0.2	1	1	6	7	10	0.1
7	0.1	1	0.4	1	0.2	1	1	6	7	5	0.1
8	0.1	1	0.4	1	0.2	1	1	6	7	8	0.1
9	0.1	1	0.4	1	0.2	1	1	6	7	13	0.1
10	0.1	1	0.4	1	0.2	1	1	6	7	5	0.05
11	0.1	1	0.4	1	0.2	1	1	6	7	5	0.15
12	0.1	1	0.4	1	0.2	1	1	6	7	5	0.25
13	0.1	1	0.4	1	0.2	1	2	12	5	5	0.05
14	0.1	1	0.4	1	0.4	1	2	12	5	5	0.05
15	0.1	1	0.6	1	0.4	1	2	12	5	5	0.05
16	0.3	1	0.6	1	0.4	1	2	12	5	5	0.05

**Table 4**

Results of tests of modification of the genetic algorithm – island model.

№	$t_s, s$	$t_f, s$	$F_s$	$F_f$	$S$
1	1.144	1924.096	1123.667	14.667	0.577
2	2.211	3663.413	1101.000	6.333	0.299
3	3.068	5399.514	1118.667	4.000	0.207
4	1.123	1835.984	1119.333	9.333	0.605
5	1.121	1892.263	1112.667	6.000	0.585
6	1.100	1893.736	1094.333	5.333	0.576
7	1.160	1898.343	1131.000	5.000	0.593
8	1.060	1929.846	1124.000	8.000	0.579
9	1.120	1880.434	1152.333	10.667	0.608
10	1.070	1899.741	1143.000	5.667	0.599
11	1.080	1913.294	1109.000	7.000	0.576
12	1.161	1906.083	1146.667	11.000	0.596
13	1.497	1314.078	383.000	3.333	0.217
14	1.640	1819.898	10.667	0.667	0.012
15	1.606	1485.881	7.000	1.000	0.011
16	1.971	640.981	9.667	0.000	0.015

for effective problem solving.

An experimental study of the proposed genetic methods was performed. The island model of the genetic algorithm proved to be more efficient both in terms of speed and quality of the solutions obtained. On average, the island model of the GA works much faster – the execution time was reduced by 41.3% on average, it has a better fitness function result, the average fitness

**Table 5**

Test results of algorithms with the best parameters.

Algorithm	№	$t_s, s$	$t_f, s$	$F_s$	$F_f$
Genetic algorithm	1	1.077	847.888	8.000	0.000
	2	1.020	1320.838	11.000	0.000
	3	1.178	1946.044	12.000	2.000
	Average	1.092	1371.590	10.333	0.667
Island model of GA	4	1.857	2016.956	3.000	0.000
	5	1.009	349.973	12.000	0.000
	6	1.048	50.780	4.000	0.000
	Average	1.305	805.903	6.333	0.000

value is 0, which means that the algorithm has found the ideal solution and makes it a more efficient algorithm for this problem.

## Acknowledgments

The work was carried out with the support of the state budget research project of the state budget of the National University “Zaporozhzhia Polytechnic” “Intelligent methods and tools for diagnosing and predicting the state of complex objects” (state registration number 0122U000972).

## References

- [1] S. Lukas, A. Aribowo, M. Muchri, Genetic algorithm and heuristic search for solving timetable problem case study: Universitas Pelita Harapan timetable, in: 2009 Second International Conference on the Applications of Digital Information and Web Technologies, 2009, pp. 629–633. doi:10.1109/ICADIWT.2009.5273979.
- [2] I. A. Abduljabbar, S. M. Abdullah, An Evolutionary Algorithm for Solving Academic Courses Timetable Scheduling Problem, Baghdad Science Journal 19 (2021) 399–408. doi:10.21123/bsj.2022.19.2.0399.
- [3] H. Alghamdi, T. Alsubait, H. Alhakami, A. Baz, A Review of Optimization Algorithms for University Timetable Scheduling, Engineering, Technology & Applied Science Research 10 (2020) 6410–6417. doi:10.48084/etasr.3832.
- [4] O. M. Haitan, The university course timetabling automation, Scientific notes of Taurida National V.I. Vernadsky University. Series: Technical Sciences 1 (2020) 58–66. doi:10.32838/2663-5941/2020.2-1/09.
- [5] A. R. Petrosian, R. V. Petrosyan, I. A. Pilkevych, M. S. Graf, Efficient model of PID controller of unmanned aerial vehicle, Journal of Edge Computing 2 (2023) 104–124. doi:10.55056/jec.593.
- [6] V. Y. Snityuk, O. M. Sipko, Technology of Evolutionary Formation of Timetables in Institutions of Higher Education, Yu.V. Picha FOP Publisher, 2022.

- [7] I. Mulyava, The solution of the problem of automated formation of schedule of educational institutions with genetic algorithms, *Mizhnarodnyi naukovi zhurnal "Internauka"* (2018) 77–83. URL: [http://nbuv.gov.ua/UJRN/mnj\\_2018\\_9%281%29\\_\\_20](http://nbuv.gov.ua/UJRN/mnj_2018_9%281%29__20).
- [8] V. V. Kysil, I. V. Drach, T. M. Kysil, Task of creation and optimization of learning schedule that supplements strict and volatile requirements model, *Scientific notes of Taurida National V.I. Vernadsky University. Series: Technical Sciences* 6 (2019) 65–70. doi:10.32838/2663-5941/2019.6-1/12.
- [9] W. A. Indha, N. S. Zamzam, A. Saptari, J. A. Alsayaydeh, N. b. Hassim, Development of Security System Using Motion Sensor Powered by RF Energy Harvesting, in: *2020 IEEE Student Conference on Research and Development (SCORED)*, 2020, pp. 254–258. doi:10.1109/SCORED50371.2020.9250984.
- [10] JavaTpoint, *Genetic Algorithm in Machine Learning*, 2021. URL: <https://www.javatpoint.com/genetic-algorithm-in-machine-learning>.