# A MODIFIED GENETIC METHOD FOR AUTOMATIC SCHEDULING

I. Fedorchenko[1], A. Oliinyk[1], Jamil Abedalrahim Jamil Alsayaydeh[2], S. Shylo[3], K. Miediveidev[1], Y. Fedorchenko[1] and M. Khokhlov[4]
[1]Department of Software Tools, National University, Zaporizhzhia Polytechnic, Zhukovskoho str., Zaporizhzhya, Ukraine
[2]Department of Engineering Technology, Fakulti Teknologi and Kejuruteraan Elektronik and Komputer (FTKEK), Universiti Teknikal Malaysia Melaka (UTeM), Melaka, Malaysia
[3]Department of Electrical and Electronic Apparatus, National University, Zaporizhzhia Polytechnic, Zhukovskoho str, Zaporizhzhya, Ukraine
[4]Department of Computer Systems and Networks, National University, Zaporizhzhia Polytechnic, Zhukovskoho str, Zaporizhzhya, Ukraine
E-Mail: jamil@utem.edu.my

**ABSTRACT**

The problem of creating an optimal schedule is considered, which consists in finding the optimal distribution of educational classes for a certain period under given restrictions. Sequential and parallel scheduling methods based on genetic search have been developed. The proposed methods use adapted and modified initialization, crossover, and selection operators. Algorithms, using the objective function, minimize conflicts between classes and the time interval between classes, taking into account the recommended time and venue. The developed methods allow you to speed up the time for planning the educational process and avoid mistakes when creating a schedule. A comparative analysis was conducted between the classical and modified genetic algorithms, and it was found that the modified algorithm works faster and more efficiently than the classical one. The performance of the modified algorithm was also compared with different genetic algorithm operators and parameters to determine the best ones. The obtained results allow us to propose effective methods for improving the quality of scheduling and improving the learning process at the university.

**Keywords:** genetic algorithm, schedule, evolutionary algorithm, classes, constraints.

Manuscript Received 21 November 2023; Revised 12 February 2024; Published 29 February 2024

## INTRODUCTION

Time and resource management is a critical aspect of success in any field, particularly in university education. Drawing up a schedule of classes is a complex task that requires a lot of resources since many restrictions must be considered at the same time: compliance with the curriculum, holding classes in specialized classrooms, considering the needs of teachers and students, etc. [1]. All these factors cannot be considered when creating a schedule manually. Therefore, it is advisable to automate this process for optimal use of resources.

Due to the significant number of restrictions and the complexity of building a mathematical model, the task of compiling a schedule can be classified as a problem of the NP-complete class. For this problem, the number of possible solutions can be very large depending on the size of the input data. Algorithms that search for the best approximate solutions are usually used. One of the options for solving the task of lesson planning is the use of metaheuristic algorithms, which start with one or more initial solutions and use search strategies, with the help of which they approach the optimal solution.

The genetic algorithm is a metaheuristic algorithm, which is a powerful tool for solving optimization problems. Genetic algorithms are adapted to various tasks and conditions and can find solutions on a global scale. In this work, a modified genetic algorithm for the decision of arranging classes for the university is proposed.

## LITERATURE REVIEW AND PROBLEM STATEMENT

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

In [1], the authors use a genetic algorithm and a heuristic search to solve the problem of scheduling at a university. The method of creating a schedule is based on genetic algorithms, which are aimed at maximizing the number of successfully planned lesson units in the schedule, considering various restrictions, such as the availability of teachers, classrooms, etc.

The article [2] uses the method of genetic algorithms to solve the problem of drawing up a schedule of classes at a university or college. The authors proposed a system where complex objects and processes, including 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.

Article [3] examines the problem of scheduling classes and explores various methods for its optimization. The article provides an overview of the genetic search method for creating a schedule, but it is noted that such a method works rather slowly, which determines the feasibility of developing a parallel implementation of the corresponding method.

The task of creating a schedule is to create an optimal schedule of classes considering various restrictions and requirements. This problem 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 this problem should reflect the main criteria for optimization. The main assessment criteria for the schedule are: minimizing the number of windows between pairs: reducing the time spent by teachers and groups; even distribution of the load during the week: evenly distributing the load on teachers and students to increase their working capacity; minimization of conflicts and overlaps of classes: involves avoiding overlap between classes that have common groups, teachers, or are held in the same classroom; consideration of timetable suggestions: the timetable should be designed in such a way as to best meet the needs of students or teachers.

Therefore, the objective function for the scheduling problem is the sum of the penalty functions, which shows how much the constraint is violated during the schedule generation. Penalty functions can have different weights, depending on how important it is to comply with a certain restriction. We use the following notation: ng is the number of groups; *nt* is number of teachers; na is the number of audiences; ns is the number of classes; *D* is the number of school days; *P* is the number of classes per day; Y is the type of occupation; $G = [g_1, g_2, ..., g_{ng}]$ is the distribution relative to groups; $T = [t_1, t_2, \ldots, t_{nt}]$ is the timetable relative to teachers; $A = [a_1, a_2, \ldots, a_{na}]$ is the schedule relative to audiences; $C$ = [c1, c2, …, cnc] is the general schedule; $R = [r_{i,1}, r_{i,2}, \ldots, r_{i,nr}]$ is the recommended time of classes, where i is the class number, and nr is the amount of recommended time; $S = [s_{i,1}, s_{i,2}, \ldots, s_{i,ns}]$ is the class number for classes, where i is the class number, and ns is the number of pairs in the class; $X = [x_1, x_2, \ldots x_{nx}]$ is the schedule of a separate group, teacher or classroom, where *nx* is the number of classes in a separate group, teacher or classroom.
Let's introduce functions

$$\mathrm{day}(x) = \text{day of the week of class x}, \tag{1}$$

$$\mathrm{pair}(x) = \text{lesson number x}, \tag{2}$$

$$\mathrm{type}(x) = \text{type of occupation x} \tag{3}$$

Let's introduce the penalty function $f_w(X,y)$, which calculates the windows between classes and has the corresponding weight factor k1. This function calculates the difference between the numbers of the next and previous classes within one day. First, you need to count the windows separately in the numerator, denominator and total. Let the value 1 correspond to the numerator, 2 to the denominator, and the value 3 to the total class. It should also be noted that before using this function, it is necessary to sort the class array: first by the day of the week, then by class number, and then by class type. We get the following formula for counting windows (4):

$$f_w(X,y) = \sum_{i=1}^{nx-1} \begin{cases} k1 \cdot (\mathrm{pair}(x_{i+1}) - \mathrm{pair}(x_i)), \text{if} \\ \mathrm{day}(x_{i+1}) = \mathrm{day}(x_i) \text{ та } \mathrm{type}(x_{i+1}) = \mathrm{type}(x_i) = y, \\ 0, \text{else}. \end{cases}$$

We will also introduce the functions $f_1(X)$, $f_2(X)$, $f_3(X)$, which consider the peculiarities of the educational process. The function $f_1(X)$ counts windows for all types of occupations (5):

$$f_1(X) = f_w(X,1) + f_w(X,2) + f_w(X,3). \tag{5}$$

The function $f_2(X)$ calculates the overlap of classes and has a corresponding weighting factor $k_2$. Overlapping classes can be considered a situation when two or more classes are scheduled at the same time. Exceptional situations arise when these two classes are located in the schedule in both the numerator and the denominator. The function that calculates the overlapping of classes (6):

$$f_2(X) = k_2 \cdot \sum_{i=1}^{nx} \sum_{j=1}^{nx} \begin{cases} 1, \text{if } \mathrm{type}(x_i) = 1 \text{ and } \mathrm{type}(x_j) = 1 \text{ or } 3, \\ 1, \text{if } \mathrm{type}(x_i) = 2 \text{ and } \mathrm{type}(x_j) = 2 \text{ or } 3, \\ 1, \text{if } \mathrm{type}(x_i) = 3 \text{ and } \mathrm{type}(x_j) = 1 \text{ or } 2 \text{ or } 3, \\ 0, \text{else}. \end{cases}, \tag{6}$$

The $f_3$ function considers compliance with the recommendations regarding the time of the lesson and has a corresponding weighting factor $k_3$. The function looks like this (7):

$$f_3 = k_3 \cdot \sum_{i=1}^{nc} \sum_{j=1}^{nr} \begin{cases} 1, \text{if } \mathrm{day}(r_{i,j}) \neq \mathrm{day}(s_{i,j}) \\ \text{or } \mathrm{pair}(r_{i,j}) \neq \mathrm{pair}(s_{i,j}), \\ 0, \quad \text{else}. \end{cases} \tag{7}$$

The general formula for calculating the penalty for the schedule using penalty functions (5) – (7) will have the form (8):

$$F = \sum_{i=1}^{ng} f_1(g_i) + f_2(g_i) + \sum_{i=1}^{nt} f_1(t_i) + f_2(t_i) + \sum_{i=1}^{na} f_2(a_i) + f_3 \tag{8}$$

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

Lesson planning is a complex combinatorial problem with a large number of constraints and a large amount of data. Traditional scheduling methods that apply to some other problems are not suitable for this particular problem. However, there are heuristic and computational methods that can be successfully used to solve it.

Usually, heuristic methods are used to solve the planning problem, which are based on heuristic algorithms and intuitive approaches for finding optimal solutions. These methods find acceptable solutions faster but do not guarantee finding the best possible option due to the lack of mathematical justification of their work. To determine how close the found schedule is to the optimal solution, it is often compared with the results of the brute force method. Despite this limitation, this approach is effective in cases where finding the absolute best option is too difficult or impossible.

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 (8). 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.

## 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: initialization of input data and parameters of the algorithm; initialization of the initial population; calculation of the adaptability of individuals; crossing of individuals; mutation of individuals in the population; calculation of the suitability of the obtained individuals; selection; continuing the steps until the stop criterion is satisfied [5].

The input data for the algorithm are: a list of classes, each class contains a teacher or teachers, one or more groups, a discipline, the type of class, the number of times the class is held; a list of the recommended schedule for classes; number of school days; the maximum number of classes per day; list of audiences.

The algorithm has the following parameters: population size; 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].

A random initialization method can be used to initialize the search. 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 considered [6].

Using random initialization can lead to a significant increase in search time. Therefore, a modified method of generating the initial population of genetic search is proposed, which uses a priori information about the features of the schedule, known from the given constraints. In the modified method of random initialization, a day of the week is generated and an audience is selected, after which a possible class time is searched. 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 method of initialization allows you to 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.

Mathematically, the crossover function for exchanging one random session between two parent schedules can be expressed as follows. Let $P_1$ and $P_2$ be the parent schedules with classes, where $G_1$ represents the selected class with $P_1$, and $G_2$ is the selected class with $P_2$. Then $C_1$and $C_2$ are the offspring formed by exchanging these occupations (10-11):

$$C_1 = (P_1 \setminus \{G_1\}) \cup \{G_2\} \tag{10}$$

$$C_2 = (P_2 \setminus \{G_2\}) \cup \{G_1\} \tag{11}$$

Also, for comparison, the k-point crossing method was implemented. In k-point crossing, k points on parental chromosomes are randomly selected. These points divide the chromosome into k+1 segment. Segments are then exchanged between the two parent chromosomes to create two new offspring chromosomes. Usually, an even multiple of k is used to ensure that each segment of the parental chromosomes will be split into two equal parts [6].

Let $P_1$and$P_2$ are two parental chromosomes, where $P_1$=[$p_1$,$p_2$,$p_3$,…,pm ] and $P_2$=[$q_1$,$q_2$,$q_3$,…,$q_n$ ]. $C_1$ and $C_2$ are two offsprings, which are formed using $k$-point crossing. $k$ is the number of points that are randomly chosen to divide chromosomes into segments.

$K$-point crossover can be described as follows. We randomly select k points on the parental chromosomes $P_1$and $P_2$, denote these points as $p_1, p_2, \dots, p_k$ for $P_1$ and $q_1, q_2, \dots, q_k$ for $P_2$. We create new offspring $C_1$and$C_2$. $C_1$ will be formed by joining segments with $P_1$ and $P_2$, where all segments are used $P_1$, except for the segments between the points $p_1$ and $p_2$, $p_2$ and $p_3$, ..., $p_k$ and $p_{k+1}$, and all segments $P_2$ between the correspondng points $q_1$ and $q_2$, $q_2$ and $q_3$, ..., $q_k$ and $q_{k+1}$. $C_2$ will be formed by the union of segments with $P_2$ and $P_1$, where all $P_2$ segments are used, except for the segments between the points $q_1$ and

$q_2$, $q_2$ and $q_3$, ..., $q_k$ and $q_{k+1}$, and all $P_1$ segments between the corresponding points $p_1$ and $p_2$, $p_2$ and $p_3$, ..., $p_k$ and $p_{k+1}$. Mathematically, it can be represented as follows (12)-(13):

$$C_1 = P_1[1:p_1] \cup P_2[q_1:q_2] \cup P_1[p_{1+1}:p_2] \cup P_2[q_{2+1}:q_3] \cup \ldots \cup P_1[p_{k+1}:p_k] \cup P_2[q_k:] \quad (12)$$

$$C_2 = P_2[1:q_1] \cup P_1[p_1:p_2] \cup P_2[q_{1+1}:q_2] \cup P_1[p_{2+1}:p_3] \cup \ldots \cup P_2[q_{k+1}:q_k] \cup P_1[p_k:] \quad (13)$$

where $P_1[\mathrm{a}:b]$ denotes the segment from a to b in the chromosome $P_1$.

The advantage of k-point crossing is that it provides diversity in the offspring, allows you to preserve useful combinations of the genetic material of the parents, and can also help avoid hitting a local minimum in the optimization of the cost function. However, its disadvantage is that it can lead to the loss of useful information from parental chromosomes, especially at large values of $k$ [6].

Mutation is used to increase population diversity and prevent stalling in local optima. It randomly changes one or more genes in order to create new possible solutions [6].

If we change many genes in an individual, it can make the individual unfit, because most of the changes will not be useful. Therefore, a mutation was implemented, which will change the schedule to one random class.

The mutation operation can be described mathematically as follows (14):

$$P_{mut} = (P \setminus \{G_{old}\}) \cup \{G_{new}\}, \quad (14)$$

where $P$ is the parental chromosome before the mutation; $P_{mut}$ is the chromosome after mutation, in which one random gene was replaced; $G_{old}$ is the gene that was deleted from father $P$ during the mutation; $G_{new}$ is a new gene that was added to father $P$ during mutation.

Mutation of all genes with a chance is a modification of the genetic algorithm, which consists in changing the value of all genes in each individual decision with a certain probability. This method is used in order to help the genetic algorithm to get out of local optima that may appear during the process of finding a solution. With the mutation of genes with a chance, random changes in genes can lead to the discovery of new, more optimal solutions [6].

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:

**- tournament:** two solutions are chosen randomly, and priority is given to the solution with a higher fitness value. Let $S_1$and $S_2$ are two randomly chosen solutions. S is a decision that gets priority based on their fitness function (15):

$$S = \begin{cases} S_1, if\ Fitness(S_1) \geq Fitness(S_2) \\ S_2, if\ Fitness(S_1) < Fitness(S_2) \end{cases} \quad (15)$$

**- ranking:** a selection method based on their suitability ranking. We calculate the rank of each solution $S_n$ in the population based on its fitness value. We calculate the sum of the ranks of all solutions in the population (16):

$$R_s = \sum_{i=1}^{n} i, \quad (16)$$

where n is the number of individuals in the population; i is the rank of the individual.

For each decision $S_n$, we calculate the probability of choice using its rank (17):

$$P(S_n) = R(S_n)/R_s, \quad (17)$$

where $P(S_n)$ is the probability of choosing an individual $S_n$; $R(S_n)$ is the rank of the individual $S_n$.

The selected solution S in the ranking selection is selected as follows (18):

$$S = S_n, if\ P(S_{n-1}) < rand(0{,}1) \leq P(S_n), \quad (18)$$

where $rand(0{,}1)$ is a random number from 0 to 1.

**- roulette:** selection of candidates of the next generation based on probabilities corresponding to their fitness [6].

In this method, the probability of selecting an individual is proportional to its fitness value.

We calculate the sum of the fitness values of all solutions in the population (19):

$$F_s = \sum_{i=1}^{n} Fitness(i), \quad (19)$$

where n is the number of individuals in the population; $Fitness(i)$ is the fitness of the individual i.

The probability of choosing each solution is calculated as follows (20):

$$P(S_n) = Fitness(S_n)/F_s, \quad (20)$$

where $P(S_n)$ is the probability of choosing an individual $S_n$.; $Fitness(S_n)$ is the individual fitness.

Roulette selection is calculated as follows (21):

$$S = S_i, if\ i \in [1;n] \text{ та } \sum_{i=1}^{n} P(S_i) \geq rand(0{,}1), \quad (21)$$

where n is the number of individuals in the population; $P(S_i)$ $rand(0{,}1)$is the probability of choosing an individual; is a random number from 0 to 1.

Scaling of the fitness of individuals was implemented in the selection, which allows for an increase in the speed of convergence 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 probability 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] - [27].

## EXPERIMENTS AND RESULTS

Input data for the system are: data about classrooms: classroom name, type, capacity, fixed departments; data about departments: name of the department and its abbreviated name; specialty data: 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 [27] - [37].

The initial data is the optimal class schedule compiled for the selected department, which considers 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.

The speed of the 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 (22):

$$S = \frac{|F_s - F_f|}{|t_f - t_s|}, \tag{22}$$

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; $t_s$ is final time.

The following abbreviations should also be noted:

a) $P_e$ is the value of elitism, i.e. the percentage of the total population;
b) $T_c$ is type of crossing (crossing), can take the following values: 1 is "custom_one_gene", 2 is "k_point";
c) $N_k$ is the number of k crossing points;
d) $P_c$ is the probability of crossing;
e) $T_m$ is type of mutation (mutation), can take the following values: 1 is "custom_one_gene", 2 is "all_genes";
f) $P_{mg}$ is the probability of gene mutation;
g) $P_m$ is the probability of mutation;
h) $T_s$ is the type of selection (selection), can take the following values: 1 is "roulette", 2 is "ranging", 3 is "tournament";
m) $T_i$ is the type of initialization (initialization), can take the following values: 1 is "random", 2 is "simple_algorithm";
n) $N_g$ is the maximum number of iterations of the algorithm;
o) $N_p$ is the size of the population.

The following key values were considered to create the schedule. Number of days for drawing up the schedule: 6 days. The number of classes held per day for drawing up a schedule: 6 classes. Penalty for group windows: 1, which means that attention has been paid to avoiding unfilled time between classes for student groups. Penalty for teachers' windows: 1, emphasizing the importance of avoiding free breaks between teachers' classes. Penalty for overlapping classes: 5, which shows the importance of avoiding conflicts and overlaps in the schedule. Penalty for non-compliance with the recommended class time: 5, indicating the importance of observing the set class hours. These parameters were used to support schedule optimization, ensure compliance, and avoid conflicts and overlaps during schedule creation [37]-[55].

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 2, 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 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 the step of increasing the values in the islands; $N_{it}$ is the number of iterations through which to perform migration between islands; $P_{mig}$ is how many individuals will participate in migration.

**Table-1.** Parameters for running genetic algorithm tests.

| # | $P_e$ | $T_c$ | $N_k$ | $P_c$ | $T_m$ | $P_{mg}$ | $P_m$ | $T_s$ | $T_i$ |
|---|---|---|---|---|---|---|---|---|---|
| 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 |

**Table-2.** Genetic algorithm results.

| # | $t_s$, 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 |

**Table-3.** Parameters for launching genetic algorithm modification tests - island model.

| # | $P_e$ | $P_c$ | $P_m$ | $T_i$ | $N_i$ | $N_{st}$ | $N_{it}$ | $P_{mig}$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 0.1 | 0.4 | 0.2 | 1 | 6 | 2 | 10 | 0.1 |
| 2 | 0.1 | 0.4 | 0.2 | 1 | 12 | 2 | 10 | 0.1 |
| 3 | 0.1 | 0.4 | 0.2 | 1 | 18 | 2 | 10 | 0.1 |
| 4 | 0.1 | 0.4 | 0.2 | 1 | 6 | 3 | 10 | 0.1 |
| 5 | 0.1 | 0.4 | 0.2 | 1 | 6 | 5 | 10 | 0.1 |
| 6 | 0.1 | 0.4 | 0.2 | 1 | 6 | 7 | 10 | 0.1 |
| 7 | 0.1 | 0.4 | 0.2 | 1 | 6 | 7 | 5 | 0.1 |
| 8 | 0.1 | 0.4 | 0.2 | 1 | 6 | 7 | 8 | 0.1 |
| 9 | 0.1 | 0.4 | 0.2 | 1 | 6 | 7 | 13 | 0.1 |
| 10 | 0.1 | 0.4 | 0.2 | 1 | 6 | 7 | 5 | 0.05 |
| 11 | 0.1 | 0.4 | 0.2 | 1 | 6 | 7 | 5 | 0.15 |
| 12 | 0.1 | 0.4 | 0.2 | 1 | 6 | 7 | 5 | 0.25 |
| 13 | 0.1 | 0.4 | 0.2 | 2 | 12 | 5 | 5 | 0.05 |
| 14 | 0.1 | 0.4 | 0.4 | 2 | 12 | 5 | 5 | 0.05 |
| 15 | 0.1 | 0.6 | 0.4 | 2 | 12 | 5 | 5 | 0.05 |
| 16 | 0.3 | 0.6 | 0.4 | 2 | 12 | 5 | 5 | 0.05 |

**Table-4.** Results of tests of genetic algorithm modification - 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 |

## DISCUSSION OF RESEARCH RESULTS

After analyzing Table-2, the following conclusions can be drawn regarding the highest rate of convergence in GA parameters: mutation probability is 0.2; type of mutation is mutation of one gene; crossing probability is 0.6; the value of elitism is 0.2; sampling type is roulette; crossing type is crossing of one gene.

To analyze the influence of the parameter *Ti*=2 (using a modified initialization parameter), let's calculate how much the algorithm's running time has decreased on average compared to the values of $T_i = 1$. To do this, we'll calculate the average value of the algorithm's execution time for both options. The average value of the working time can be calculated according to the following formula (23):

$$t_{avr} = \frac{\sum_{i=1}^{n}(t_{fi} - t_{si})}{n}, \quad (23)$$

where $t_{fi}$ is the final time of work in the start-up; $t_{si}$ is the initial time of the algorithm at startup; n is the number of starts.

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

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

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

After analyzing Table-4, the following conclusions can be drawn regarding the highest rate of convergence in the parameters of the island model: the number of islands is 6 (the number of logical processor cores); the parameter discrepancy step is 7; after how many iterations to carry out migration is 10; the number of individuals to participate in migration is 0.1.

The average value of the working time of the island model at $T_i$ = 1 is 2335.027 s. The average value of the working time of the island model at $T_i$ = 2 is 1313.531 s.

When using the modified initialization method, the GA island model works 1.8 times faster on average. This can improve the performance of the algorithm, which allows you to save time and resources when solving the problem.

We test algorithms with parameters that give the best results (Table-5).

**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 |

Based on the results, it can be concluded that the island model of the genetic algorithm is better in terms of speed and quality of finding a solution compared to the classical genetic algorithm. On average, the GA island model works much faster - the running time is reduced by 41.3% on average, it has a better fitness function result, the average fitness value is 0, which means that the algorithm has found an ideal solution and makes it a more efficient algorithm for of this task.

## CONCLUSIONS

A modified genetic method has been developed that uses an initialization opera-tor based on a priori information about the learning process, which is available from given constraints. The use of the developed approach to the initialization of the genetic method makes it possible to significantly (many times) reduce the search time.

A modified island model of the developed genetic method was also developed to solve the problem of creating an optimal schedule of educational classes. The fundamental difference between the proposed method and the existing analogues is the use of a modified initialization operator, which tries to reduce the initial value of fitness through a simple selection of possible options. Using the modified initialization opera-tor, the running time of the classical genetic algorithm was reduced by 40.9%. It is also worth noting that the operating 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 made it possible to significantly save the algorithm execution time. The modified initialization method helps to improve the performance and execution speed of the genetic algorithm for drawing up a schedule of training sessions, which is important for effective problem solving.

An experimental study of the proposed genetic methods was carried out. The island model of the genetic algorithm turned out to be more effective both in terms of speed and in terms of the quality of the obtained solutions. On average, the GA island model works much faster - the running time is reduced by 41.3% on average, it has a better fitness function result, and the average fitness value is 0, which means that the algorithm has found an ideal solution and makes it a more efficient algorithm for of this task.

**ACKNOWLEDGEMENTS**

The authors extend their appreciation to Universiti Teknikal Malaysia Melaka (UTeM) and to the Ministry of Higher Education of Malaysia (MOHE) for their support in this research.

## REFERENCES

[1] Solving Timetable Problem by GeneticAlgorithm and Heuristic Search Case Study: Universitas Pe-lita Harapan Timetable [Electronic resource]. – Access mode: https://www.researchgate.net/publication/221927228_Solving_Timetable_Problem_by_Genetic_Algorithm_and_Heuristic_Search_Case_Study_Universitas_Pelita_Harapan_Timetable.

[2] An Evolutionary Algorithm for Solving Academic Courses Timetable Scheduling Problem [Electronic resource]. - Access mode: https://www.researchgate.net/publication/356044560_An_Evolutionary_Algorithm_for_Solving_Academic_Courses_Timetable_Scheduling_Problem.

[3] A Review of Optimization Algorithms for University Timetable Scheduling. - Access mode: https://www.researchgate.net/publication/347802207_A_Review_of_Optimization_Algorithms_for_University_Timetable_Scheduling

[4] Some Methods of Solving the NP-difficult Problem of Optimal Schedule for the University. – Access mode: https://www.sciencedirect.com/science/article/pii/S187705091930417X#:~:text=However%2C%20there%20are%20a%20number%20of%20heuristic%20and,method%2C%20graph %20coloring%20method%2C%20intelligent%20method%2C%20genetic%20algorithm.

[5] Gaitan O. M. 2020. Automation of the generation of the schedule of the educational process of the university / O. M. Gaitan // Informatics, computer technology and automation. 31(70) Part 1. - No. 2. pp. 58-66.

[6] V. Ye. Snityuk. 2022. Technology of evolutionary formation of timetables in institutions of high-er education: monograph / V. Ye. Snityuk, O. M. Sipko. - Kyiv: Yu.V. Picha FOP Publisher. p. 136.

[7] I. Ya. Mulyava. 2018. Solving the problem of automated formation of the schedule of an educa-tional institution using genetic algorithms / I. Ya. Mulyava. International scientific journal Inter-nauka. 9(1): 77-83.

[8] V. V. Kysil. 2019. The model of the task of drawing up and optimizing the class schedule under the condition of meeting the objective and subjective requirements of the educational institution / V. V. Kysil, I. V. Drach, T. M. Kysil. Informatics, computing technology and automation. 30(69) Part 1. - No. 6. pp. 65-70.

[9] Tomashevskyi V. M. 2010. Compiling class schedules in distance learning systems / V. M. To-mashevskyi, Yu. L. Novikov, P. A. Kaminska // Bulletin of the National Technical University of Ukraine KPI. Informatics, management and computer technology. (52): 118-130.

[10] Genetic Algorithm in Machine Learning [Electronic resource]. – Access mode: https://www.javatpoint.com/genetic-algorithm-in-machine-learning.

[11] Genetic algorithms. Key concepts and implementation methods [Electron. resource]. - Access mode: http://www.znannya.org/?view=ga_general.

[12] An Illustrated Guide to Genetic Algorithm [Electronic resource]. - Access mode: https://towardsdatascience.com/an-illustrated-guide-to-genetic-algorithm-ec5615c9ebe.

[13] Genetic Algorithm - Advantages & Disadvantages [Electronic resource]. - Access mode: https://electricalvoice.com/genetic-algorithm-advantages-disadvantages/.

[14] Semenchuk V. M. 2020. Peculiarities of using the island model of genetic algorithms / V. M. Semenchuk. - Ternopil National Technical University named after Ivan Pulyuy. - UDC 004.021. p. 114.

[15] D. Verma, C. Bose, N. Tufchi, K. Pant, V. Tripathi, A. Thapliyal. 2020. An efficient framework for identification of Tuberculosis and Pneumonia in chest X-ray images using Neural Network, Procedia Computer Science 171: 217-224. doi: 10.1016/j.procs.2020.04.023.

[16] Oliinyk A., Fedorchenko I., Stepanenko A., Katschan, A., Fedorchenko Y., Kharchenko A., Goncharenko D. 2019. Development of genetic methods for predicting the incidence of volumes of emissions of pollutants in air. 2019 2nd International Workshop on Informatics and Data-Driven Medicine, IDDM, CEUR Workshop Proceedings. 2488: 340-353.

[17] Fedorchenko I., Oliinyk A., Stepanenko, A., Svyrydenko, A, Goncharenko, D. 2019. Genetic method of image processing for motor vehicle recognition. 2019 2nd International Workshop on Computer Modeling and Intelligent Systems, CMIS, Zaporizhzhia, April 15-19, CEUR Workshop Proceedings. 2353: 211-226.

[18] Fedorchenko I., Oliinyk A., Stepanenko A., Zaiko T., Korniienko S., Burtsev N. 2019. Development of a genetic algorithm for placing power supply sources in a distributed electric network. European Journal of Enterprise Technologies, 5(101): 6-16, doi: 10.15587/1729-4061.2019.180897

[19] Oliinyk, A., Fedorchenko, I., Stepanenko, O. Rud M., Goncharenko, D. 2021. Implementation of evolutionary methods of solving the travelling salesman problem in a robotic warehouse. Lecture Notes on Data Engineering and Communications Technologies. 48, pp. 263-292.

[20] Fedorchenko I., Oliinyk A., Stepanenko Zaiko T., Korniienko S., Kharchenko A. 2020. Construction of a genetic method to forecast the population health indicators based on neural network models. Eastern-European Journal of Enterprise Technologies, 1(4-103): 52-63. DOI: 10.15587/1729-4061.2020.197319

[21] Phang F. A., Pusppanathan J, Nawi N. D., Zulkifli N. A., Zulkapri I., Harun F. K. C., Wong A. Y. K., Alsayaydeh J. A. J., Sek T. K. 2021. Integrating Drone Technology in Service Learning for Engineering Students. International Journal of Emerging Technologies in Learning. 16(15): 78-90.

[22] Zakir Hossain A. K. M., Hassim N. B., Alsayaydeh J. A. J., Hasan M. K. & Islam M. R. 2021. A tree-profile shape ultra wide band antenna for chipless RFID tags. International Journal of Advanced Computer Science and Applications, 12(4): 546-550. doi:10.14569/IJACSA.2021.0120469.

[23] Jamil Abedalrahim Jamil Alsayaydeh, Azwan Aziz, A. I. A. Rahman, Syed Najib Syed Salim, Maslan Zainon, Zikri Abadi Baharudin, Muhammad Inam Abbasi and Adam Wong Yoon Khang. 2021. Development of Programmable Home Security Using Gsm System for Early Prevention, ARPN Journal of Engineering and Applied Sciences. 16(1): 88-97.

[24] S. Mishra, R. Sachan, and D. Rajpal. 2020. Deep convolutional neural network based detection system for real-time corn plant disease recognition. Procedia Comput. Sci., 167: 2003-2010, Accessed: Nov. 8, 2023. [Online]. Available: https://doi.org/10.1016/j.procs.2020.03.236

[25] Indra W.A., Zamzam N.S., Saptari A., Alsayaydeh J.A.J, Hassim N.B. 2020. Development of Security System Using Motion Sensor Powered by RF Energy Harvesting. 2020 IEEE Student Conference on Research and Development, SCOReD 2020 9250984, pp. 254-258.

[26] M. Jung *et al*. 2023. Construction of deep learning-based disease detection model in plants. Scientific Rep. 13(1). Accessed: Nov. 8, 2023. [Online]. Available: https://doi.org/10.1038/s41598-023-34549-2

[27] Adam Wong Yoon Khang, Shamsul J. Elias, Nadiatulhuda Zulkifli, Win Adiyansyah Indra, Jamil Abedalrahim Jamil Alsayaydeh, Zahariah Manap, Johar Akbar Mohamat Gani. 2020. Qualitative Based QoS Performance Study Using Hybrid ACO and PSO Algorithm Routing in MANET. Journal of Physics, Conference Series 1502, 012004, doi:10.1088/1742-6596/1502/1/012004.

[28] S.-H. Lee, C.-C. Wu and S.-F. Chen. 2018. Development of image recognition and classification algorithm for tea leaf diseases using convolutional neural network. in 2018 Detroit, Mich. July 29 - August 1, St. Joseph, MI: Amer. Soc. Agricultural Biol. Engineers, 2018. Accessed: Nov. 8, 2023.

[Online]. Available: https://doi.org/10.13031/aim.201801254

[29] I. Fedorchenko, A. Oliinyk, A. Stepanenko, T. Fedoronchak, Y. Fedorchenko, A. Kharchenko and D. Goncharenko. 2020. Development of a genetic method for image recognition in the form of radiographs. 3rd International Conference on Informatics and Data-Driven Medicine, pp. 94-107, ISSN: 16130073.

[30] D. Goncharenko, A. Oliinyk, I. Fedorchenko, S. Korniienko, O.O. Stepanenko, A.S. Kharchenko, Y. Fedorchenko. 2020. Genetic Algorithm for Solution of the Problem of Optimal Location of the Distributed Electrical Networks. 10th International Conference on Advanced Computer Information Technologies (ACIT), Deggendorf, Germany, pp. 380-385, Doi: 10.1109/ACIT49673.2020.9208888.

[31] I. Grobelna, “Model checking of reconfigurable FPGA modules specified by Petri nets,” J. Syst. Arch., vol. 89, pp. 1–9, 2018.

[32] A. A. Khan et al., "A drone-based data management and optimization using metaheuristic algorithms and blockchain smart contracts in a secure fog environment," Computers and Electrical Engineering, vol. 102, p. 108234, 2022.

[33] I. Grobelna, “Formal verification of control modules in cyber-physical systems,” Sensors (Basel), vol. 20, no. 18, p. 5154, 2020.

[34] A. Ayub Khan, A. A. Wagan, A. A. Laghari, A. R. Gilal, I. A. Aziz and B. A. Talpur, "BIoMT: A State-of-the-Art Consortium Serverless Network Architecture for Healthcare System Using Blockchain Smart Contracts," in IEEE Access, vol. 10, pp. 78887-78898, 2022, doi: 10.1109/ACCESS.2022.3194195.

[35] R. Rani, S. Kumar, R. R. Kadam, and S. K. Pippal. 2023. A machine learning model for predicting innovation effort of firms. Int. J. Elect. Comput. Eng. (IJECE), 13(4): 4633, Accessed: Nov. 10, 2023. [Online]. Available: https://doi.org/10.11591/ijece.v13i4.pp4633-4639

[36] A. Lateef Haroon P.S and U. Eranna. 2019. A simplified machine learning approach for recognizing human activity. Int. J. Elect. Comput. Eng. (IJECE), 9(5): 3465, Accessed: Nov. 10, 2023. [Online]. Available: https://doi.org/10.11591/ijece.v9i5.pp3465-3473

[37] L. Elhaloui, S. El Filali, E. H. Benlahmer, M. Tabaa, Y. Tace and N. Rida. 2023. Machine learning for internet of things classification using network traffic parameters. Int. J. Elect. Comput. Eng. (IJECE), 13(3): 3449, Accessed: Nov. 10, 2023. [Online]. Available: https://doi.org/10.11591/ijece.v13i3.pp3449-3463

[38] S. A. Ebiaredoh-Mienye, E. Esenogho and T. G. Swart. 2021. Artificial neural network technique for improving prediction of credit card default: A stacked sparse autoencoder approach. Int. J. Elect. Comput. Eng. (IJECE), 11(5): 4392, Accessed: Nov. 10, 2023. [Online]. Available: https://doi.org/10.11591/ijece.v11i5.pp4392-4402

[39] S. Krishnan, P. Magalingam, and R. Ibrahim. 2021. Hybrid deep learning model using recurrent neural network and gated recurrent unit for heart disease prediction. Int. J. Elect. Comput. Eng. (IJECE), 11(6): 5467, Accessed: Nov. 10, 2023. [Online]. Available: https://doi.org/10.11591/ijece.v11i6.pp5467-5476

[40] R. Saifan and F. Jubair. 2022. Six skin diseases classification using deep convolutional neural network. Int. J. Elect. Comput. Eng. (IJECE), 12(3): 3072, Accessed: Nov. 10, 2023. [Online]. Available: https://doi.org/10.11591/ijece.v12i3.pp3072-3082.

[41] Fedorchenko, A. Oliinyk, A. Stepanenko, *et al.* 2021. Development of a genetic method for x-ray images analysis based on a neural network model. Open Bioinformatics Journal. 14(1): 51-62. DOI: 10.2174/1875036202114010051.

[42] Mochurad L., Hladun Y., Zasoba Y., Gregus M. 2023. An Approach for Opening Doors with a Mobile Robot Using Machine Learning Methods. Big Data Cogn. Comput. 7, 69. https://doi.org/10.3390/bdcc7020069.

[43] S. Ljaskovska, Y. Martyn, I. Malets, and O. Velyka, "Optimization of Parameters of Technological Processes Using the FlexSim Simulation Program," in Proceedings of the 2020 IEEE 3rd International Conference on Data Stream Mining and Processing (DSMP), 2020, pp. 391–397, 9204029.

[44] Fedorchenko, A. Oliinyk, Jamil Abedalrahim Jamil Alsayaydeh, A. Kharchenko, A. Stepanenko and V. Shkarupylo, 2020. Modified Genetic Algorithm to Determine the Location of the Distribution Power Supply Networks in the City. ARPN Journal of

Engineering and Applied Sciences. 15(23): 2850-2867.

[45] Mochurad L, Hladun Y, Tkachenko R. 2023. An Obstacle-Finding Approach for Autonomous Mobile Robots Using 2D LiDAR Data. Big Data and Cognitive Computing. 7(1): 43. https://doi.org/10.3390/bdcc7010043.

[46] M. S. Hamid, N. A. Manap, R. A. Hamzah, A. F. Kadmin, S. F. A. Gani, A. I. Herman. 2022. A New Function of Stereo Matching Algorithm Based on Hybrid Convolutional Neural Network. Indonesian Journal of Electrical Engineering and Computer Science, 25(1): 223-231, doi: 10.11591/ijeecs.v25.i1.pp223-231.

[47] Z. A. Shaikh et al., "A New Trend in Cryptographic Information Security for Industry 5.0: A Systematic Review," in IEEE Access, vol. 12, pp. 7156-7169, 2024, doi: 10.1109/ACCESS.2024.3351485.

[48] Lesia Mochurad, Andrii Dereviannyi, Uliana Antoniv. Classification of X-Ray Images of the Chest Using Convolutional Neural Networks. IDDM 2021 Informatics & Data-Driven Medicine. Proceedings of the 4th International Conference on Informatics & Data-Driven Medicine. Valencia, Spain, November 19 - 21, 2021. 269-282.

[49] A. F. Kadmin, R. A. Hamzah, M. N. Abd Manap, M. S. Hamid, S. F. Abd Gani. 2021. Improved Stereo Matching Algorithm Based on Census Transform and Dynamic Histogram Cost Computation. International Journal of Emerging Technology and Advanced Engineering, 11(8): 48-57, doi: 10.46338/IJETAE0821_07.

[50] Lesia Mochurad, Glib Shchur. 2021. Parallelization of Cryptographic Algorithm Based on Different Parallel Computing Technologies. Proceedings of the Symposium on Information Technologies & Applied Sciences (IT&AS 2021). Bratislava, Slovak Republic, March 5, Vol. 2824, ISSN 1613-0073, pp. 20-29.

[51] S. Liaskovska and Y. Martyn, "Investigation of Anomalous Situations in the Machine-Building Industry Using the Phase Trajectories Method," in Lecture Notes in Networks and Systems, vol. 463, pp. 49–59, 2022.

[52] Z. A. Shaikh and S. A. Khoja, "Higher education in Pakistan: An ICT integration viewpoint," Int. J. Comput. Theory Eng., vol. 5, no. 3, p. 410, 2013.

[53] Y. Martyn, S. Liaskovska, M. Gregus, and O. Velyka, "Optimization of Technological Processes in Industry 4.0 for Details Manufacturing via Stamping: Rules of Queuing Systems," Procedia Computer Science, vol. 191, pp. 290–295, 2021.

[54] Z. A. Shaikh and S. A. Khoja, "Towards Guided Personal Learning Environments: Concept, Theory, and Practice," 2014 IEEE 14th International Conference on Advanced Learning Technologies, Athens, Greece, 2014, pp. 782-784, doi: 10.1109/ICALT.2014.230.

[55] Y. Martyn, S. Liaskovska, M. Gregus, and O. Velyka, "Optimization of Technological Processes in Industry 4.0 for Details Manufacturing via Stamping: Rules of Queuing Systems," Procedia Computer Science, vol. 191, pp. 290–295, 2021