Application of Selection Sequence Optimization Algorithm to University Timetabling Problem

Dimitriev A.P.[*] Lavina T.A. Aleksandrov A.H.

Chuvash State University, Cheboksary, 428015, Russian Federation
[*]*Corresponding author. Email: dimitrie1@yandex.ru*

ABSTRACT
The object of study is the automated scheduling and optimization of class timetables. To study the operation of various discrete optimization algorithms applied to the timetabling problem, as well as new algorithms a mathematical model of schedules is used. For this model, the so-called selection sequence optimization algorithm is characterized by the best indicators for the quality of the resulting timetable. The paper presents a modification of this algorithm based on the multiple start method. The adaptation of the selection sequence optimization algorithm directly to the timetabling problem at the university is proposed. The components of the objective function for optimization of timetable, taking into account psycho-pedagogical requirements to the educational process are considered. Such characteristic of the quality of timetable as the number of unplaced man-hours is highlighted. The results of optimization using the above algorithm and the random search method are compared in terms of time and quality of the obtained timetable. Numerical values of the parameters of the above algorithm are proposed.
Keywords: *timetabling, timetable classes, simulated annealing, discrete optimization, optimization algorithm, objective function*

1. INTRODUCTION

The university timetable has factors that can have a negative impact on the mental performance of students [1]. Therefore, it is important to make such timetables as optimal as possible. For this purpose, software packages are used that use various algorithms that optimize the quality of the timetable to one degree or another. The task of such algorithms is to find the optimal timetables at the admissible time among the vast number of possible choices. These choices may include all or part of the required events due to conflicting requirements and lack of resources. In studies of various scientific teams on this topic, metaheuristic algorithms are used, such as simulated annealing (SA) [2], where, among other things, the number of unplaced events is minimized, evolutionary algorithms, for example, [3], where it is required that each event be placed, the particle swarm method [4], etc.
An important stage of the study is modeling. So, in the process of research, a mathematical model of the class timetable was developed and a comparative assessment of the effectiveness of the above and other, including new, algorithms with this model was performed. The study showed the indisputable advantage of the so-called selection sequence optimization algorithm (SSO) [5]. The next stage of the study is the adaptation of the SSO to the real class timetable. Purpose of work: adaptation of the SSO to the task of timetabling classes.
The paper considers the task of timetabling classes for the university faculty for one semester (16 to 18 weeks). Such a timetable is characterized by repeatability. Some events are repeated every week, some once every two weeks, others once every four weeks.
Timetabling classes involves a number of problems. Many constraints and often conflicting requirements need to be considered. Most of the events, as a rule, can be scheduled using a computer without the approval procedure, and the scheduling of the remaining events requires the participation of a dispatcher or another person involved in the timetabling. So, for example, the dispatcher can contact teachers on the possibility of changing the time of their classes. He may ask the educational-methodical department of the university to allocate an additional room for a small part of the time.
A computer program cannot violate specified requirements on its own responsibility, as the dispatcher does. Requests of the program to individuals may be unsuccessful due to depersonalization. In addition, it cannot wait for several days for answers from higher authorities, etc. As a result, the timetable cannot always be fully scheduled automatically. But then students will not fulfill the curriculum and will have to graduate in violation of the state standard on the number of hours of study, which is unacceptable. In such conditions, the main priority of the program for scheduling classes is to make it possible to add the maximum number of classes to the grid of hours.
On the other hand, even a timetable that is not fully scheduled should be the best possible, since its quality helps to give learning competencies to one degree or another. Therefore, it is also necessary to optimize the

timetable using some objective optimization function, which is an additional criterion of its quality.
Next, the model and algorithm used are considered, then the transition to the real class schedule is performed.

2. METHODOLOGY

2.1. SSO and mathematical model of timetables

SSO is a discrete optimization algorithm. It is designed to work with the following mathematical model of class timetables, which is characterized by the indispensable reachability of an acceptable timetable. In terms of scheduling theory [6], events are jobs, which periodically performed by computers. Each computer performs one job for a period of time the same for all computers. The resulting schedule is repeatedly used to repeat the jobs by all computers.
Jobs have options:
- for the i-th computer, the duration of the corresponding job p_i is determined;
- each computer has its own constant significance t_i.

The period is divided into m identical segments (discrete time instants). The performing of some job begins at one of these time instants and is not interrupted until its completion. Set of such time moments forms a schedule. The problem is that the more jobs are performed at the same time, the greater the external overload. Overload should be minimized in accordance with the following objective function:

$$C = \sum_{h=1}^{m} \sum_{i,k=1}^{n} t_{h,i} t_{h,k} \text{ , (1)}$$

where m is the number of instants in a period, n is the number of computers, $t_{h,i}$ is a function that depends on the operation of the i-th computer at the h-th instant of the period; its value is either t_i or zero, which depends on p_i and the beginning of work. Similarly, $t_{h,k}$ is defined.
The study was conducted mainly at $n = 40$ and $m = 100$, in order to get closer to the values, respectively, of the number of subgroups at the faculty and the number of calls at the beginning of the event for two weeks.
The SSO algorithm for its work uses some starting point, which is a list of computers ordered in some way. This list is called a selection sequence (SS). It was revealed that one of the quick and effective ways of ordering is sorting computers in descending order of the $t_i \cdot p_i$.
Distribution of job by time is carried out by sequentially determining the start time for each computer from the SS. The SS is then optimized by SA [7], and each time a schedule is built and the objective function is calculated for it.
The SSO has parameters:
- R is a distance in SS within which two computers can mutually rearrange;
- X is the number of such permutations before constructing a schedule once according to the resulting SS;
- J is the maximum number of iterations.

The most effective is the modification of SSO using the multiple start method [5]. The algorithm can be described as follows.

```
0. Minimum is denoted by $M$; achieved schedule is denoted by $A$; number of starting points is denoted by $I$.
1.1. Assign a large number to $M$, and an empty set to $A$.
1.2. Sort computers by any criterion, for example, by increasing significance. The sequence of numbers of the sorted list is designated as the starting point $S$ in $n$-dimensional space.
2. In the loop on $i$, execute $I$ times:
2.1. Make schedule using sequence $S$ (putting computers in the schedule sequentially at the best moments of time, which seem to be such at the time of placement, for this purpose, reviewing each of the $m$ moments for the next computer).
2.2. Calculate the objective function $C(S)$ by the formula (1).
2.3. Set number of iterations $j \leftarrow 1$.
2.4. In a neighborhood of $S$ defined by the spread of $R$, choose a random point denoted by $S_1$ by a pseudo-random permutation of $X$ elements in $S$.
2.5. Make schedule $A_1$ using sequence $S_1$ like step 2.1.
2.6. Calculate $C(S_1)$ in accordance with the schedule received.
2.7. If $C(S) > C(S_1)$, then:
2.7.1. Set $S \leftarrow S_1$;
2.7.2. Set $M \leftarrow C(S_1)$;
2.7.3. Set $A \leftarrow A_1$.
2.8. Otherwise, with probability $\exp(|C(S) - C(S_1)| \cdot (j/J))$, if $|C(S) - C(S_1)|$ less than some constant threshold, then $S \leftarrow S_1$.
2.9. Set $j \leftarrow j+1$.
2.10. If $j<J$, then go to step 2.4.
2.11. End of cycle by $i$.
3. Save $A$, display $M$. End.
```

2.2. Adapting the SSO algorithm to the class timetable

Moving from the model to the real class timetable, it should be noted that when calculating the values of the objective function, as in [8], the weighted sum of its components is used. The components of the objective function are formulated taking into account the psychological and pedagogical requirements for the organization of the educational process in Russian universities [1]. Some of these requirements are formulated by a survey of participants in the educational process at the university.
The objective function takes into account the following parameters:
- planning a lecture in the phases of active efficiency of students. This is also said in [9], where it is reported that the morning classes are more effective than evening classes;

- continuity of classes for students. During the day, for a particular subgroup, the number of gaps between classes should be minimal;
- teachers' classes continuity. Such a criterion was reported in [10];
- capacity of rooms. Obviously, the number of students in a class should not exceed the number of seats in the corresponding room. However, at the request of the teacher, additional laptops can be allocated for laboratory class in the computer class, and chairs from neighboring classrooms can be brought in at the lecture. It happens that a small proportion of students are absent for good reasons. Therefore, exceeding the capacity of the room is acceptable, but it worsens the value of the objective function;
- students' fatigue during the day (see below);
- remoteness of educational buildings. Students must have time to move from one audience, where they have a class, to another, where they have the next class, during the break, otherwise the value of the objective function worsens. The importance of this parameter is drawn in [11, 12];
- uniformity of classes on the days of the week. Such a criterion was reported in [13].

According to these parameters, the analytical expression of the objective function can be written as:

$$V = \sum_{i=1}^{7} k_i v_i \,. \quad (2)$$

Here v_i is the component of the i-th type, taking into account the following parameters:

- conducting classes in basic disciplines in the phases of active efficiency of students: $v_1 = \sum_{j=1}^{n_g} w_j^1 g_j (5 - c_j)$;
- continuity of classes for students: $v_2 = \sum_{j=1}^{n_g} w_j^2 g_j$;
- teachers' requirements: $v_3 = \sum_{j=1}^{n_l} w_j^3 l_j$;
- capacity of rooms: $v_4 = \sum_{j=1}^{n_g} \sum_{t=1}^{t_{\max}} g'_j(t)$;
- students' fatigue during the day: $v_5 = \sum_{j=1}^{n_g} w_j^5 g_j$;
- remoteness of educational buildings: $v_6 = \sum_{j=1}^{n_g} \sum_{t=1}^{t_{\max}} g_j x_j^t$;
- uniformity of classes on the days of the week: $v_7 = \sum_{j=1}^{n_g} w_j^7 g_j$.

The following notation is used in these formulas:

k_i is the coefficient of significance of the term, which is set by the program user;

n_g is the number of subgroups;

g_j is the number of students in the j-th subgroup, where $j = 1,\dots, n_g$;

n_l is the number of teachers;

l_j is the weight factor of the j-th teacher, where $j = 1,\dots, n_l$. This refers to the level of professional training of the teacher (academic degree, title, work experience); it is determined by expert ball. Here, also the teacher's residence in another city is taken into account, since nonresident teachers are asked to minimize the number of days of their arrival at the university;

w^1_j, where $j = 1,\dots, n_g$, computed using a table function. Class of the j-th subgroup, conducted at the k-th hour, adds to the value w^1_j the value from the table corresponding to the k-th hour, taking into account the shift in the start of classes on the given day relative to the first hour. The table contains the following values in sequence: 4,2,0,0,1,3,5,5,5,5,5,5,5,5. Additionally, for the very last hour of subgroup classes per day, the tabular value is averaged with the number 1;

w^2_j is the total duration of the "gaps" of the j-th subgroup, $j = 1,\dots, n_g$;

w^3_j is a function of the number of days of classes of the j-th teacher, $j = 1,\dots, n_l$, the number of his "gaps" and the number of his classes during the day more than 6 hours without a break. The value w^3_j summarizes the listed factors with the given weighting coefficients;

w^5_j is calculated by the formula:

$$w_j^5 = \sum_{k=1}^{8} n_{k,j} \,,$$

where $n_{1,\ j}$ is the total number of lectures for the j-th subgroup, more than two or less than one per day;

$n_{2,j}$ is the same for practical classes;

$n_{3,\ j}$ is the total number of laboratory classes of the j-th subgroup is more than 4 hours per day;

$n_{4,\ j}$ is the total number of hours of situations, when any classes are continuing more than 6 hours without a break for the j-th subgroup;

$n_{5,j}$ is the sum of the days that the j-th subgroup has more than 3 hours of identical classes per day, if this is not in the plan;

$n_{6,\ j}$ is the total number of absences of a combination of different on profile disciplines in the schedule of one day for the j-th subgroup. For example, technical and humanitarian;

$n_{7,\ j}$ is the total number of situations “three lectures in a row” for the j-th subgroup. In contrast to $n_{1,\ j}$, it takes into account lectures that are not interspersed with classes of other types;

$n_{8,\ j}$ is the total number of lectures and (for the first two courses in basic disciplines) practical classes in the afternoon for the j-th subgroup, taking into account how much they are later than the 6th hour, in the form of a linear relationship;

w^7_j is calculated as the sum for the j-th subgroup of days in which 4 hours or less or more than 10 hours of classes;

t is the hour ordinal in the semester, $t = 1,\ldots,t_{\max}$, $t_{\max} = 18\cdot(6\cdot14 - 4) = 1440$, where 18 is the number of weeks, 6 is the number of school days per week, 14 is the number of academic hours per day, and students do not study in the last 4 hours on Saturdays;
x^t_j is the conditional distance between the current (at t-th hour) and previous (at t – 1-th hour) classrooms where classes of the j-th subgroup are held. x^t_j is determined by a function that takes into account the time required for the transition based on the Cartesian coordinates of the classrooms;
$g'_j(t)$ is the number of students of the j-th subgroup left without a seat, at the hour with number t;
c_j is the course number. The first course takes priority, as students are not yet adapted to the university system of education.
In the class timetable, there are values similar in their role for the model with significance and duration. These are the weight coefficient of the teacher, the number of students in the set of subgroups in the same room, the percentage of availability of the room during the week, and the recommended duration of the class. With the sequential placement of classes from the SS, the larger the above values for the lesson, the more difficult it becomes to find the possibility of its placement, since many resources become occupied. Therefore, “significant” classes should be placed firstly.
The initial ordering consists in assigning priorities to classes, according to which they will be entered into the grid of hours. Automatic prioritization takes into account the above analogues of significance and duration. Initial ordering sets the starting point for optimization. As reported in [12], while providing a good starting point for optimization by SA, productivity increases significantly.
According to the classification [8], the adapted OSS is one-stage, that is, at the same time, minimization by strict constraints and minimization of the objective function are performed. If there is a shortage of the classroom fund, the timetable is not fully scheduled to the end, unplaced events remain, as in [12], where the timetable are not completely scheduled, and the percentages of event placement using different methods are given as results. The peculiarity of OSS adaptation, in particular, is that it operates not only with whole events in terms of “placed” and “not placed”, but is also aimed at minimizing man-hours related to such unplaced events, including students and the teacher. As a criterion for optimizing the timetable, first of all, a minimum of such man-hours is used, and only then (2).
The adapted SSO algorithm provides promising SS with a second, and sometimes a third chance to get better schedules, so that such SS is optimized, continuing from the point reached. As calculations show, giving these chances is often useful. Designations used in the description of the algorithm:
H is the number of unplaced man-hours achieved since the launch of the program;
E is a parameter set by the user (by default, $E = 30$), subtracted from the achieved value of the quality criterion and used when deciding on the second chance;
$N(S)$ is the number of unplaced man-hours obtained with the current SS;
$N(S_1)$ is the number of unplaced man-hours obtained with the suggested SS;
$C(S)$ is the value of the objective function obtained with the current SS;
$C(S_1)$ is the value of the objective function obtained with the suggested SS;
d_1, d_2 are the dividers set by the user of the program. By default, $d_1 = 3$, $d_2 = 1$.
The SSO algorithm can be described as follows.

```
0. Minimum is denoted by the pair ($N$, $C$), achieved timetable is denoted by $A$; number of starting points is denoted by $I$.
1.1. Assign a large numbers to $H$, $N$ and $C$, and an empty set to $A$.
1.2. Set priorities for events: the more employment of the laboratory, the teacher employment and his position, the number of students and the duration of the event, the more priority for the event. Priority list, i.e. SS, denote as the starting point $S$ in $n$-dimensional space.
2. In the loop on $i$, execute $I$ times:
2.1. Create a timetable by selecting the events according to the priority principle using $S$ (putting events in the timetable sequentially at the best moments of time, which seem to be such at the time of placement, for this purpose, reviewing each of the possible hours for a two-week cycle for a newly placed event).
2.2. Calculate $V$ using formula (1) and $N(S)$. Set $C(S) \leftarrow V$.
2.3. Set number of iterations $j \leftarrow 1$.
2.4. In a neighborhood of $S$ defined by the spread of $R$, choose a random point denoted by $S_1$ by the method of $X$-fold pseudo-random change of the values of element priorities in $S$. That is, an element is randomly selected and changed to a random number within $\pm R$ relative to the old value, and the new value should remain in the range from 1 to 255. If the new value is less than 1, the element is assigned the value 1, and if it is greater than 255, then 255, respectively.
2.5. Create timetable $A_1$ using priorities from $S_1$ like step 2.1.
2.6. Calculate $C(S_1)$ and $N(S_1)$ according to the obtained timetable. Check condition $U$ of the transition to the new SS as follows:
2.6.1. Calculate $\Delta_1 = C(S) - C(S_1)$;
2.6.2. Calculate $\Delta_2 = N(S) - N(S_1)$;
2.6.3. If $|j\Delta_1|/(Jd_1)<10$, then $L_1 \leftarrow 1000\cdot\exp(-|j\Delta_1|/(Jd_1))$, otherwise $L_1 \leftarrow 0$;
2.6.4. If $|j\Delta_2|/(Jd_2)<10$, then $L_2 \leftarrow 1000\cdot\exp(-|j\Delta_2|/(Jd_2))$, otherwise $L_2 \leftarrow 0$;
2.6.5. $U$ is the value of the expression $(N(S_1) < N(S))$ or $((N(S_1) \le N(S))$ and $((C(S_1) \le C(S))$ or(random(10000)< $L_1$))) or $((N(S_1) > N(S))$ and $((C(S_1) \le C(S))$ and (random(10000)< $L_2$))).
2.7. If condition $U$ is satisfied, then:
2.7.1. Set $S \leftarrow S_1$;
2.7.2. Set $C \leftarrow C(S_1)$, $N \leftarrow N(S_1)$.
2.7.3. Set $A \leftarrow A_1$.
2.8. Set $j \leftarrow j+1$.
```

2.9. If $j<J$, then go to step 2.4.
2.10. The second chance, subject to close to the achieved and best results: if $N(S) – E \le H$, then:
2.10.1. Set $H \leftarrow N(S)$.
2.10.2. Increase i by 1.
2.10.3. Repeat steps 2.3 - 2.9.
2.11. Third chance, subject to better results: if $N(S) < H$, then:
2.11.1. Set $H \leftarrow N(S)$.
2.11.2. Increase i by 1.
2.11.3. Repeat steps 2.3 - 2.9.
2.12. End of cycle by i.
3. Save A, display C, N. End.

2.3. Random search method

For comparison, for scheduling, the random search (RS) method was also used, which consists in the following. Timetable is scheduled a specified number of times. From the results, the best one is selected according to the quality criterion, namely, the lowest N.
When using RS, a set of weeks can be randomly selected to place laboratory work. This leads to different possibilities for further filling the grid of hours and, accordingly, to different timetables.
In addition, when analyzing the placement variants of an event, sometimes the best variants are found, evaluated by the same value of the partial objective function. The value of the partial objective function is determined by formula (2) for the timetable that was not built up to the end. The choice of one or another best variant is made with some probability.

2.4. Software package

To schedule classes using the above methods, a software package was developed in the Turbo Delphi 6 environment. It is designed to work in operating systems of the Windows family on personal computers and laptops. The working version contains 18 files, the total size of 4.3 MB. The amount of memory (working set) when working the program in Windows 8.1 is a maximum of 19.6 MB. On a laptop with an AMD processor and a frequency of 1.5 GHz, the speed is 0.4 iterations per second. An iteration is a single creation of an interim timetable, its evaluation, and the accompanying actions to modify the SS. The software package retrieves the source data from the relational database. The data are placed in the database from the information system "Kafedra-2", implemented at Chuvash State University, then supplemented by the user.

3. RESEARCH RESULTS

3.1. Experimental part

The used source data set has the following characteristics:
- number of subgroups n_g =33.
- number of classrooms with laboratories is 22.
- number of teachers n_l =25.
- the number of lines of the training load, each of which can contain simultaneously lectures, practical and laboratory, which should be placed in the schedule, is 122.
To determine the optimal parameters of SSO, a number of experiments were performed like [5]. The essence of these experiments is as follows. Initially, the parameters were set arbitrarily. Then one of the parameters changed in its range of values. The value of this parameter was chosen at which SSO created the best timetable with the remaining parameters unchanged. Then another parameter was selected and similar actions were performed for it with the already new value of the first parameter. Such actions are performed for all parameters. As a result, the following set of parameters is defined: d_1 = 3, d_2 = 1, R = 10, X = 5, J = 20, E=30, I = 25. Parameters I and J can be slightly increased if it is only necessary to increase the number of iterations.
In the first set of initial data, among other auditoriums, three computer classes and two classrooms were initially free all the time. The work of SSO for this case is shown in Figure 1.

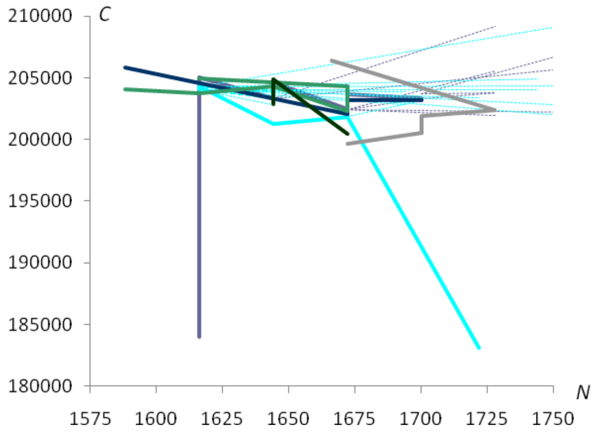

Figure 1 Trajectories of transitions to the values of the objective function and the numbers of man-hours not included in the schedule, starting from various points

In figure 1 solid lines show the transition paths between the achieved values in the coordinates (N, C). Several of these lines of different colors illustrate the use of several starting points. The lines are broken, since the transitions are carried out spasmodically, from one break point to the next.
Intermittent thin segments show the study of the neighborhood of the break points. The first end of the segment is the break point of the solid line, and the second

end is the point in its neighborhood, for which a schedule was drawn up, leading to unacceptably worse results.

In the second set of initial data, three computer classes and two classrooms were initially occupied for 6.3% of the time. For this set, the RS was performed five times by 1000 iterations. An average value of *N* = 2314 with a standard deviation of 8.2 was achieved with an average of *C* = 189411 with a standard deviation of 5160. The results of SSO with the second data set for the same machine time as for the RS are as follows: *N* = 1764.8 with a standard deviation of 30.8, *C* = 192583 with a standard deviation of 9199.

Figure 2 illustrates the achievement of *N* depending on the iteration number by both algorithms. Here, the iteration number is plotted along the abscissa, and *N* is plotted along the ordinate. The thick line shows the averaged value for five experiments, and the thin lines show the actual values of specific experiments.

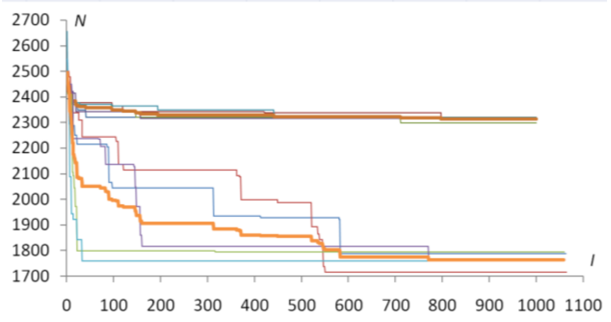

Figure 2 Reducing the number of unplaced man-hours depending on the iteration number when using RS and SSO

The upper group of lines shows the results of the RS method, and the lower, respectively, the OSS algorithm.

3.2. Discussion of results

As can be seen from the graphs, OSS allows to get significantly better results compared to RS.

During a series of experiments using OSS, the minimum values are reached no further than at the 770th iteration. On average, this is the 480th iteration with a standard deviation of 303. Since 1000/480>2, on average there is more than a double margin of the number of iterations performed by OSS; therefore, the number of iterations was chosen sufficient for analysis. Accordingly, for the RS this is the 565th iteration with a standard deviation of 264 and a maximum at the 798th iteration. This means that OSS more quickly finds minimum values for itself than RS.

On average, for the entire duration of the process that implements OSS, *N* = 2545 with a standard deviation of 295. Accordingly, for RS these values are 2486 and 64. Therefore, the RS method finds values in a close vicinity of the starting point, while OSS considers a wider area. This allows OSS to find better schedules than the RS finds. When performing OSS, significantly worse decisions are considered than when performing a RS, since the maximum *N* with a RS is on average 2771, and with OSS 3797. This is due to the fact that with RS events are always placed in the same sequence, whereas OSS, for example, can sometimes try to place a stream lecture closer to the end of the scheduling process, which is difficult.

With OSS, the value of *C* is on average greater than with RS (respectively, 199015> 191274). This is due to the fact that if less events are included in the timetable, as when using a RS, then the *C* components become smaller: students become less tired and less transfer from building to building, teachers are more free. Some components may increase, for example, technical and humanitarian disciplines may cease to be present on the same day. But nevertheless, the decrease of *C* prevails provided that the automatically generated SS is used. If the SS is created randomly, an increase in *C* prevails. For example, for the above initial data with random SS and three iterations of OSS, *N* = 8350 and *C* = 3091948 were obtained. This is many times higher than the values shown in figure 2 and is caused primarily by the influence of such factors:

- events for subgroups appear in the amount of zero per day;
- an increase in the percentage of events that take place only in the first two hours during the day (this is an unfavorable time);
- the absence of a combination of disciplines of different profiles in the timetable of one day.

The standard deviation *C* when using SSO is also greater (13943> 4393). This is due to the same reasons that cause a large difference in the maxima of *N*.

In terms of events, 1800 person-hours correspond to about 126 events, which depends on the size of subgroups and their union in the events. A value of *N* = 1800 indicates that about 21% of man-hours or about 20% of classes are not scheduled.

4. CONCLUSION

An experimental comparison of the results of optimization by random search and SSO shows that the proposed SSO shows results that are better by 22-26%. Thus, SSO can be successfully adapted to the task of timetabling classes at a university.

REFERENCES

[1] MU 2515-81. Metodicheskiye ukazaniya po organizatsii obucheniya studentov vysshikh uchebnykh zavedeniy (Gigiyenicheskiye i meditsinskiye voprosy) (Guidelines for the organization of training of students of higher educational institutions (Hygienic and medical issues)). Designed by: NII gigiyeny im. F.F. Erismana, Glavnoye sanitarno-epidemiologicheskoye upravleniye Minzdrava SSSR, TSNIL. Approved by: Ministerstvo zdravookhraneniya SSSR (USSR Ministry of Health Care 10-11/9) (1982).

[2] T. Song, S. Liu, X. Tang, X. Peng, M. Chen, An iterated local search algorithm for the University Course Timetabling Problem, Applied Soft Computing 68 (2018) 597–608. DOI: https://doi.org/10.1016/j.asoc.2018.04.034

[3] M-X. Zhang, B. Zhang, N. Qian, University course timetabling using a new ecogeography-based optimization algorithm, Nat. Comput. 16 (2017) 61–74. DOI: https://doi.org/10.1007/s11047-016-9543-8

[4] D. Qarouni-Fard, A. Najafi-Ardabifi, M-H. Moeinzadeh, et al., Finding feasible timetables with particle swarm optimization, in: 4th IEEE international conference on innovations in information technology, 2007, pp. 387–391. DOI: https://doi.org/10.1109/IIT.2007.4430422

[5] A. P. Dimitriev, Determination optimal parameters for modifications algorithm of selection sequence optimization, // Modern high technologies 10(1) (2019) 213–216. DOI: https://doi.org/10.17513/snt.37726

[6] R. W. Conway, W. L. Maxwell, L. W. Miller, Theory of Scheduling, Addison-Wesley Publishing company (1967).

[7] S. Kirkpatrick, C. D. Gelatt Jr., M. P. Vecchi, Optimization by Simulated Annealing, Science 220(4598) (1983) 671–680. DOI: https://doi.org/10.1126/science.220.4598.671

[8] R. Lewis, A survey of metaheuristic-based techniques for University Timetabling problems, OR Spectrum 30 (2008) 167–190. DOI: https://doi.org/10.1007/s00291-007-0097-0

[9] N. Pope, Replication data for: How the Time of Day Affects Productivity: Evidence from School Schedules, Review of economics and statistics, 98(1) (2015) 1–11. DOI: https://doi.org/10.7910/DVN/28663

[10] J. H. Kingston, Resource assignment in high school timetabling, in: Proceedings of the Seventh International Conference on the Practice and Theory of Automated Timetabling, 2008.

[11] A. Constantino, W. Filho, D. Landa-Silva, Iterated Heuristic Algorithms for the Classroom Assignment Problem, in: Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2010), 2010, pp. 152–166.

[12] S. Elmohamed, G. Fox, P. Coddington, A comparison of annealing techniques for academic course scheduling, in: E. Burke, M. Carter (Eds.), Practice and theory of automated timetabling II (PATAT) 1408, Springer, Berlin, 1998, pp. 146–166. DOI: https://doi.org/10.1007/BFb0055883

[13] I. B. Ushakov, E. P. Melikhova, I. I. Libina, O. I. Gubina, Hygienic and psychophysiological peculiarities of forming health of students of the medical university, Hygiene and sanitation, 97(8) (2018) 756–761. DOI: https://doi.org/10.18821/0016-9900-2018-97-8-756-761