INFORMATICA, 2012, Vol. 23, No. 3, 405–425

On Multi-Start Algorithms for Optimization of High School Timetables

Jonas MOCKUS, Lina PUPEIKIENĖ
Vilnius University, Institute of Mathematics and Informatics
Akademijos 4, LT-08663 Vilnius, Lithuania
e-mail: jmockus@gmail.com

Received: May 2010; accepted: March 2011

Abstract. The paper deals with the problem of high-school time-tabling that is important in applications, but hard for solving. The algorithm is presented for timetabling based on Multi-start and Simulated Annealing with parameters adapted using the Bayes approach. The algorithm proposed is compared with other timetabling algorithms using the web-based software. A multi-start algorithm is a simple way to provide the convergence, if the number of uniformly distributed starting points is large. A disadvantage is slow convergence.

Therefore, the first aim of this paper is experimental comparisons of the efficiency of different versions of multi-start algorithms in the optimization of timetables. To obtain representative results, the algorithms should be compatible with the Lithuanian high school practice and flexible enough for adaptation to different high schools.

The second aim is a web-based implementation of these algorithms in a way convenient for high schools. The web-based software is important for evaluation and comparison of algorithms by independent experts, as well, since the efficiency of algorithms depends on subjective parameters specific to each school, so on-line calculations are needed to obtain representative data. It is useful for scientific cooperation and applications to different schools. In addition, the software for evaluating of real timetables is included to compare with the results of optimization.

Keywords: timetabling, heuristics, Bayesian, Pareto, java, multi-start.

1. Introduction

1.1. *Algorithms and Software*

A multi-start algorithm is a way to provide the convergence if the number of uniformly distributed starting points is large. The simplest version is just a greedy algorithm with fixed or random starting points without any additional local optimization.

The next improvement is two algorithms of simple local search, considering only the closest timetables in a deterministic (LD) or random (LR) way. The local search is improved using an algorithm similar to Simulated Annealing (SA) with fixed parameters. Permutations are limited to the closest timetables and are performed by closing the gaps between lectures for students and teachers.

In the final improvement, to provide independence from the human operator, to save operators' time, and to increase the efficiency of search, the initial temperature and annealing rate of SA are adapted using the Bayesian Approach (BA) which optimizes expected SA results. A useful property of BA is that it filters-out random deviations in the process (Mockus, 1989). The necessary timetabling conditions are represented as hard constraints. Desirable conditions are included as penalty points in the general framework of Pareto optimality.

The web-based software developed as a Java servlet is available on four synchronized servers (last modification in March 2010). The efficiency of recent versions of Java is close to that of the most efficient programming languages (Bull *et al.*, 2001). The graphical user interface was made close to the practice of Lithuanian schools. The software is used by schools for timetabling. The results were applied in large and small high schools and compared with the commercial software. The software is used by universities, too, but for a different purpose: as a real-life example of discrete optimization, suitable for distance graduate studies.

1.2. *New Elements*

This paper is about the investigation of efficiency of multi-start algorithms by the example of Lithuanian high schools. Preliminary results, mainly regarding the local search procedures, are in Mockus (2002), Pupeikiene and Mockus (2005, 2010). A novelty of this paper as compared with these publications is a formal definition of the timetable optimization problem and the results of systematic experimental investigation of different versions of multi-start algorithms. In addition, the new algorithm for evaluating of real school timetables is described and possibilities of parallel computing are discussed.

An important feature is adaptation of parameters of the Simulated Annealing (SA), using the Bayesian approach (Mockus, 1989) which optimizes the expected SA results as a function of initial temperature and the cooling rate. This procedure eliminates the dependence of SA results on a human operator. That is important for the objective evaluation of algorithm efficiency.

The papers describe, apparently, the first web-based platform-independent school timetabling software. Implementation of the software as a Java servlet provides possibilities of application in any school with internet connection, independent of computing environment. Any web browser works, no additional software is needed. This is important for remote schools.

The web-based software is important for evaluation and comparison of algorithms by independent experts, too, since the efficiency of algorithms depends on subjective parameters specific to each school. Thus on-line calculations are needed to obtain actual data. That is useful for scientific cooperation and for applications in different schools. The software for evaluating real timetables is included, as well, with a view to compare to the results of optimization.

1.3. *On School Timetabling*

A specific feature of the school timetabling field is a great number of research papers and wide-used commercial software. Therefore a detailed discussion of the results will be preceded by the short references to relevant publications.

An important part of an optimization model is the graphical user interface (GUI). To share the Lithuanian experience with some other countries and to illustrate similarities and differences with related commercial and open software systems, a detailed description of GUI will follow the theoretical and algorithmic parts of the paper.

The aim is to satisfy constraints. We distinguish two types of constraints: conditions that must be met (hard constraints) and desires that should be fulfilled as well as possible (soft constraints). An example of a hard constraint is that a teacher or student cannot attend more than one lesson simultaneously. Similarly, room constraints are satisfied, if each room is used for only one lesson at a time.

An example of a soft constraint is compactness of teachers' and students' schedules. We increase the compactness by eliminating gaps between lessons. Provision of free days for teachers is another example of a soft constraint. An additional set of soft constraints is defined by didactic reasons, for example, a desire to place "hard" subjects, such as mathematics or physics, into the morning hours.

The maximal number of daily hours $T_{\max}$ can be considered in both ways. For example, $T_{\max} \leqslant 24$ is obviously a hard constraint. The condition $T_{\max} \leqslant 6$ is not exactly a hard constraint since a more relaxed condition $T_{\max} \leqslant 7$ or $T_{\max} \leqslant 8$ can be accepted if that improves the school timetable significantly.

Timetabling can be generally defined as an activity of assigning, subject to constraints, the number of events to a limited number of time periods and locations so that desirable objectives were satisfied as much as possible (Wren, 1996). The key goal is to find a timetable that satisfies all the hard constraints and minimizes the violation of soft constraints. A survey on educational timetabling problems (Schaerf, 1999) gives an overview of the literature.

1.4. *Optimization in Commercial Software*

We discuss optimization possibilities of the following three commercial timetabling systems currently used in Lithuanian high schools: "Mimosa 2008", "aSc TimeTables 2008", and "Rector 2008".

"Mimosa" (2010) is a product of the Finnish company „Mimosa Software Ltd." "Mimosa" provides convenient GUI for manual timetabling and reports about violations of constraints. The form is acceptable to Lithuanian schools. Optimization is limited to closing some gaps in teachers schedules. The software is popular in the basic schools. Application in the upper classes of high schools is possible within some strict limitations by setting individual student schedules. Long and hard manual work is needed if the school is large.

"Rector 2008" (Smykalov, 2010) is a product of the Russian company "P.Yu. Smykalov". It is convenient for the basic school timetabling. There is no automatic optimization.

	CPU time	Number of samples	Number of violated constrains	Number of unscheduled subjects	Complexity of samples	Constrains
2 classes 47 students	00:07:40	141200	0	27	low	„soft“
	00:10:58	182146	20	26	average	„soft“
	01:54:05	1624528	33	24	large	„soft“
	00:35:29	1353174	0	29	low	„hard“
	05:19:07	4895895	16	26	average	„hard“
	63:52:36	68675510	26	27	large	„hard“
4 classes 106 students	28:53:21	955869	38	45	low	„soft“
	56:46:33	4525965	42	39	average	„soft“
	nedirbo	---	---	---	large	„soft“
	85:37:40	352951459	63	55	low	„hard“
	failed	---	---	---	average	„hard“
	failed	---	---	---	large	„hard“
13 classes 315 students	failed	---	---	---	low	„soft“
	failed	---	---	---	average	„soft“
	failed	---	---	---	large	„soft“
	failed	---	---	---	low	„hard“
	failed	---	---	---	average	„hard“
	failed	---	---	---	large	„hard“

Fig. 1. Testing aSc.

“aSc TimeTables 2008” (asc, 2010) is a product of the Slovak company „Applied Software Consultants s.r.o.“ The results of experimental calculations are in the Fig. 1. They show that the software works well in the basic schools, but not in large high schools.

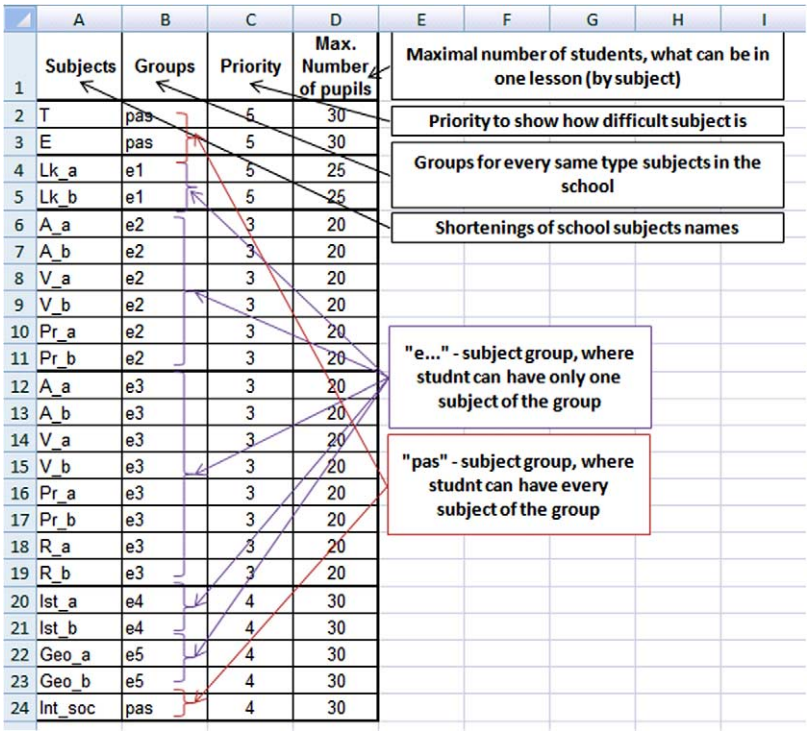
They show that the software works well in the basic schools, but not in large high schools.

The example of evaluating subjects is in the Fig. 2. A timetable that satisfies all the necessary conditions is regarded as feasible. A feasible timetable is optimal if it minimizes undesirable factors. To compare the quality of different feasible timetables we must evaluate at least the most important undesirable factors. The difficulty is that desirability is subjective and depends on the local conditions. That complicates direct comparisons of the results obtained by automatic optimization with the decisions made by human operators.

To compare the results of different methods for automatic optimization, we need some procedures for evaluating undesirable factors on some common scale. In this paper, it is done in the framework of Pareto optimality (Fudenberg and Tirole, 1983). The commercial software does not support this feature, since no direct comparison of decision quality can be made.

2. Defining the Optimization Problem

This section is meant for a formal description of objectives and constraints that are implemented in the software. The formal expressions are not easy for reading, but we think they are necessary for an exact understanding how the program works.



	A	B	C	D
1	Subjects	Groups	Priority	Max. Number of pupils
2	T	pas	5	30
3	E	pas	5	30
4	Lk_a	e1	5	25
5	Lk_b	e1	5	25
6	A_a	e2	3	20
7	A_b	e2	3	20
8	V_a	e2	3	20
9	V_b	e2	3	20
10	Pr_a	e2	3	20
11	Pr_b	e2	3	20
12	A_a	e3	3	20
13	A_b	e3	3	20
14	V_a	e3	3	20
15	V_b	e3	3	20
16	Pr_a	e3	3	20
17	Pr_b	e3	3	20
18	R_a	e3	3	20
19	R_b	e3	3	20
20	Ist_a	e4	4	30
21	Ist_b	e4	4	30
22	Geo_a	e5	4	30
23	Geo_b	e5	4	30
24	Int_soc	pas	4	30

Fig. 2. An example of evaluating subjects.

2.1. *Hard Constraints*

The school timetable d can be described formally as a binary four-dimensional array representing decisions of timetable developers

$$d = [d_{m,s,r,t}]_{M \times S \times R \times T}, \tag{1}$$

where M is a set of teachers, S is a set of students, R is a set of classrooms, and T is a set of weekly time-slots.

For example, the decision $d_{m,s,r,t} = 1$ defines a lesson of a teacher m attended by a student s in the room r at the time t. The decision $d_{m,s,r,t} = 0$ means no lesson.

Denote by A a set of timetables that satisfy the hard constraints.

The hard constraints are mandatory therefore the set A of feasible timetables is well defined. Here is a formal definition of hard constraints:

$$h_1(d) = \sum_{s,r} d_{m,s,r,t} \leqslant 1 \quad \text{for all } m, t, \tag{2}$$

$$h_2(d) = \sum_{m,r} d_{m,s,r,t} \leqslant 1 \quad \text{for all } s, t, \tag{3}$$

$$h_3(d) = \sum_{m,s,r\in R_j} d_{m,s,r,t} \leqslant R_{j,\max} \quad \text{for all } t,\ j \in J, \tag{4}$$

$$h_4(d) = \sum_{s} d_{m,s,r,t} \geqslant S_{\min} \quad \text{for all } m,\ r,\ t, \tag{5}$$

$$h_5(d) = \sum_{s} d_{m,s,r,t} \leqslant S_{\max} \quad \text{for all } m,\ r,\ t, \tag{6}$$

$$h_6(d) = \sum_{m,s,t\in T_i} d_{m,s,r,t} \leqslant T_{\max} \quad \text{for all } r,\ i \in I, \tag{7}$$

where the symbol T_i denotes a set of time-slots of the ith week-day, I is a set of week-days, J is a set of different classrooms, $T_{\max}$ limits daily time-slots, R_j is a set of rooms of type j, and $R_{j,\max}$ is the number of available j-rooms.

Condition (2) means no simultaneous lessons for teachers. Condition (3) denotes no simultaneous lessons for students. Condition (4) limits the number of classrooms of type j. Conditions (5) and (6) set the lower and upper class-size limits, and condition (7) limits the maximal number of daily time-slots. These inequalities are written assuming "one-to-one" teacher-subject relation: *teacher* $\Leftrightarrow$ *subject*.

The limit $T_{\max}$ of daily time-slots is an important hard constraint. However, it is convenient to regard this as a soft constraint with a large violation penalty. The reason is that schools may accept an extra hour if that improves the general timetable.

The decision array $d_{m,s,r,t}$ is for timetable developers. Students select subjects $v = v(m)$, they do not select teachers $m(v)$ of these subjects. Under the *teacher* $\Leftrightarrow$ *subject* assumption, the decision array (1) can be directly transformed into the following decision array:

$$d^v = d^v_{v,s,r,t}, \quad v \in V,\ s \in S,\ r \in R,\ t \in T, \tag{8}$$

where $v = v(m)$ denotes a subject of the teacher m and V is a set of subjects.

Now we define some hard constraints specific to the upper classes of high schools.

Denote by $c = c(v)$ students' reward for selecting an optional subject v. Denote by $V_1 \subset V$ a set of optional subjects, by $V_2 \subset V$ a set of mutually exclusive optional subjects, and by $V_3 \subset V$ a set of mandatory subjects. Denote a decision array of the student s as $d^s_{v,s}$, where $d^s_{v,s} = 1$, if the student s selects the subject v, otherwise $d^s_{v,s} = 0$.

Then the condition that a timetable should be based on students' decisions can be defined by the following hard constraint:

$$d^s_{v,s} = \sum_{r,t} d_{v,s,r,t} \quad \text{for all } v, s \tag{9}$$

or

$$h_7(d) = d^s_{v,s} - \sum_{r,t} d_{v,s,r,t} = 0 \quad \text{for all } v, s. \tag{10}$$

A set of hard constraints is defined by the demand for students of upper classes to earn specific rewards:

$$h_8(d) = \sum_{v \in V_1, r, t} d_{v,s} c(v) = C_1 \quad \text{for all } s, \tag{11}$$

$$h_9(d) = \sum_{v \in V_2, r, t} d_{v,s} c(v) = C_2 \quad \text{for all } s, \tag{12}$$

$$h_{10}(d) = \sum_{v \in V_3, r, t} d_{v,s} c(v) = C_3 \quad \text{for all } s. \tag{13}$$

Here C_1 is the sum of rewards a student needs to obtain by selecting a subset of optional subjects, C_2 is the reward for selecting one of the optional mutually-exclusive subjects, and C_3 is the sum of rewards for mandatory subjects. Expressions (11)–(13) define hard constraints for all students to obtain the prescribed rewards. The attribution of rewards to subjects is presented in the form of a table, an example of which is in Fig. 2.

The numbers of weekly working days and teaching hours are defined by initial data and remain unchanged during optimization. We considered only individual teachers and students so far.

We must assign teachers to subjects and students to subject-groups for school applications. Assigning teachers to subjects is straightforward, if a subject is assigned to no more than one teacher. We assume that at present.

The mapping $g = g(v)$ defines a subject-group (a group of students) that select an optional subject v. Using the subject-groups g and subjects v instead of teachers m and students s, we can transform the original decision array (1) into the group decision array as follows:

$$d^g = d^g_{v,g,r,t}, \quad v \in V,\ g \in G,\ r \in R,\ t \in T. \tag{14}$$

Here G is a set of groups.

2.2. *Soft Constraints*

We use the theory of the Pareto optimum (Fudenberg and Tirole, 1983) to balance conflicting desires represented by soft constraints.

Compactness of the timetable is a desirable property. The compactness can be improved by reducing the number of gaps between lessons.

$$\sum_{i \in I} \sum_{g, r, t \in T_i} \left| d^g_{v,g,r,t+1} - d^g_{v,g,r,t} \right| = B(v, d) \quad \text{for a subject } v, \tag{15}$$

$$\sum_{i \in I} \sum_{v, r, t \in T_i} \left| d^g_{v,g,r,t+1} - d^g_{v,g,r,t} \right| = C(g, d) \quad \text{for a group } g, \tag{16}$$

where I is a set of week-days.

The numbers of gaps $B(v,d)$ and $C(g,d)$ in timetables d are defined for subject-groups g and subjects v assuming that the names of teachers and students are irrelevant. Then the total number of gaps in the timetable d can be expressed in such a way:

$$B(d) = \sum_{v} B(v,d), \tag{17}$$

$$C(d) = \sum_{g} C(g,d). \tag{18}$$

Another indicator of timetable d compactness is the number of free week-days for teachers:

$$W(v,i,d) = \sum_{g,r,t \in T_i} d^{g}_{v,g,r,t}, \quad i \in I. \tag{19}$$

The week-day $W(v,i,d)$ is a free week-day for the subject v if the indicator $W(v,i,d) = 0$, otherwise, it is a working week-day. Denote as $I^{v}_{w(v,d)}$ a set of working week-days for the subject v in the timetable d. A set of working days in the timetable d for the teacher m is a union of working week-days for a set of subjects $V(m)$ delivered by the teacher m.

$$I_{w(m,d)} = \bigcup_{v \in V(m)} I^{v}_{w(v,d)}. \tag{20}$$

Compactness is improved by reducing the number of working week-days in the timetable d for all the teachers I_w.

$$I_{w(d)} = \sum_{m \in M} I_{w(m,d)}. \tag{21}$$

In the upper classes of high schools subject-groups are different from the traditional classes. The traditional classes can be regarded as social groups. A subject-group involves students from different classes united by a set of selected subjects. The simplest subject-group is an individual student. In Lithuanian schools the smallest subject-group is five students, otherwise, the subject is closed. The subject-groups are split, if the number of students exceeds the class-room limits, 30 students, as usual.

Note that splitting may change the composition of groups while closing the gaps between the lessons of students. For example, if there are two groups for the same subject, then a gap of some first group student can be closed by transferring this student into the second group. That is not convenient for teachers.

However, stabilization of subject-groups requires additional resources. Thus, the subject-group stability is a soft constraint that depends on the policies of a local school.

Formally the desire to stabilize the group sizes can be expressed as the total number of group-size changes $\delta(d,)$ in the timetable d. Denote by $L = L1 \cup L2$ a set of parallel

subject-groups, where $L1$ is the first group and $L2$ is the second group. Introduce a group indicator

$$\begin{aligned} l(s,t) &= 0, \quad \text{if } s \in L1 \text{ at a time } t, \\ &= 1, \quad \text{if } s \in L2 \text{ at a time } t. \end{aligned} \tag{22}$$

Then the group-change indicator

$$\delta(d,s,t) = |l(t,s) - l(t+1,s)|. \tag{23}$$

The total number of group-changes in the timetable d:

$$\delta(d) = \sum_{s \in L, t} \delta(d,s,t). \tag{24}$$

We represent different soft constraints as different objectives $f_j(d),\ j = 1,\ldots,6$, for example:

$$f_1(d) = B(d), \tag{25}$$
$$f_2(d) = C(d), \tag{26}$$
$$f_3(d) = I_{w(d)}, \tag{27}$$
$$f_4(d) = \delta(d), \tag{28}$$
$$f_5(d) = T_{\max}. \tag{29}$$

Formally $T_{\max}$ represents hard constraint (7). However, we include it to a set of soft constraints because by exceeding this constraint we can improve the general timetable.

Didactic constraints, such as a desirability of higher proportion of difficult subjects early and higher proportions of easy subjects later, are not mandatory in Lithuanian schools. Thus they create an additional set of soft constraints. The difficulty of subjects is defined by the 'Priority' column (bottom table in Fig. 1). Denote by $\rho(d)$ the number of reverse priorities where the easy subjects are scheduled before the difficult ones. Then we can write the soft constraint as follows:

$$f_6(d) = \rho(d). \tag{30}$$

2.3. *Pareto Optimum*

Timetables are defined by the decisions d. Feasible timetables (a set A) are defined by a set of hard constraints (2)–(7), (10), and (11)–(13). A set of desirable timetables (a subset of A) is not well-defined, since desires to minimize different objectives (25)–(30) are contradicting. A theoretical framework for balancing contradicting objectives (in our case, the soft constraints) is presented by the Pareto optimum (Fudenberg and Tirole,

1983). The feasible timetable $d^* \in A$ belongs to the Pareto-optimal set D^* if there are no other timetables $d' \in A$ such that:

$$f_j(d) \leqslant f_i(d^*) \quad \text{for all } j = 1, \ldots, 6, \tag{31}$$

$$f_j(d) < f_i(d^*) \quad \text{for at least one } j. \tag{32}$$

The simplest way to obtain the Pareto-optimal timetable is by minimizing the weighted sum

$$d(c) = \arg\min_{d \in A} F(c, d), \tag{33}$$

$$F(c, d) = \sum_j c_j f_j(d), \tag{34}$$

$$c = (c_1, \ldots, c_6), \quad c_j > 0.$$

The set A can be defined by conditions (2)–(7), (10), and (11)–(13):

$$h_1(d) \leqslant 1, \tag{35}$$

$$h_2(d) \leqslant 1, \tag{36}$$

$$h_3(d) \leqslant R_{j,\max}, \quad j \in J, \tag{37}$$

$$h_4(d) \leqslant S_{\max}, \tag{38}$$

$$h_5(d) \geqslant S_{\min}, \tag{39}$$

$$h_6(d) \leqslant T_{\max}. \tag{40}$$

$$h_7(d) = 0, \tag{41}$$

$$h_8(d) = C_1, \tag{42}$$

$$h_9(d) = C_2, \tag{43}$$

$$h_{10}(d) = C_3. \tag{44}$$

The proof in Fudenberg and Tirole (1983) shows that $d(c)$ is Pareto-optimal $d(c) \in D^*$. The multipliers c_j show the importance of various objectives reflecting the subjective local interests. Thus, the weights c_j should be defined by the end user himself. The task of timetabling software is to present a user-friendly way to do that.

3. Optimization Method

3.1. *Simple Multi-Start Algorithm (MC)*

The initial timetable is defined using greedy heuristics: the algorithm starts by fixing the most favorable timetable for the first teacher in the list and by attaching the corresponding subject groups. Then the time-slots are defined according to teachers' preferences while keeping the hard constraints. The timetables of other teachers are defined in a similar way

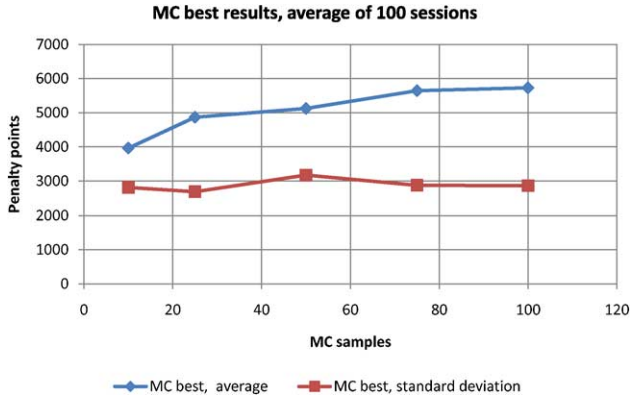


Fig. 3. The average and standard deviation of the best results in 100 sessions of MC.

assuming that the timetables of the previous teachers are fixed. Thus, the best timetable is provided for the first teacher. The timetable of the last teacher is just what remains after timetabling all the others. Therefore, an important part of the greedy algorithm is to define the sequence of teachers.

In this paper, we apply greedy heuristics by a multi-start algorithm when the sequence of teachers is defined randomly with equal probabilities, as usual. Therefore, we refer to the simplest version of multi-start as the Monte-Carlo algorithm (MC).

Figure 3 shows how the average and standard deviation of the best results depend on the number of MC samples (randomly generated initial timetables). The results were obtained using 100 MC sessions (repeating 100 samples of MC 100 times). Experiments of 100 sessions was performed to estimate the standard deviation.

In the sequel, the results of only single multi-start sessions of 100 samples will be shown. The single-session experiments limits the computing time. The calculation results are just for illustration, since the aim of this paper is to provide a web-based software for on-line optimization of school timetables under specific local conditions. This is another reason justifying the short experiments.

3.2. *Adapting to Local Conditions*

The basic way to adapt to local conditions is to set importance parameters.

An additional way is implementation of a single-session algorithm with a fixed sequence of teachers. This algorithm is for the preferential treatment of teachers according to their importance. The first teacher is getting the best timetable, the last one gets what remains.

3.3. *Defining Neighborhood*

In the following versions of the multi-start algorithm, some local optimization is performed in the neighborhood of initial timetables. In continuous optimization problems,

distances are well defined. So are the neighborhood and local optimum. However, when determining the neighborhood in the discrete set A of feasible timetables d, many different definitions can be used. Here the intuitive understanding of a distance is not very helpful. We define as neighbors those timetables that can be reached using some sequence of operations. This definition differs from the traditional one which describes neighbors in terms of distances. In contrast, we define neighbors as a set of achievable timetables. This definition is not so clear, but seems natural in school timetabling and, possibly, in some other problems of combinatorial optimization.

The definition is important, since the local search is performed in the neighborhood of the given point. In this paper, we search for better timetables by subsequent closing of gaps for students and teachers. We keep all the hard constraints during the search. In this case, the neighbors of a timetable d' are all timetables d'' that can be reached from d' by a sequence of closing gap operations. We close only the nearest gaps, because it is time-consuming to test all the hard constraints while closing gaps in the distant time-slots. That simplifies the calculations and limits the neighborhood.

Therefore, we obtain just a locally optimal timetable $d^*(d')$ that depends on the initial point d'. This is the Local Deterministic (LD) algorithm.

The local search can be randomized by selecting a current candidate (a student or a teacher) for gap closing with some probability x_0. By closing gaps for randomly selected students and teachers we modify the search sequences. This is the Local Randomized (LR) algorithm.

LR improves the search, but does not provide a convergence to the global optimum, since the neighborhood remains limited to the "achievable" timetables.

3.4. *Simulated Annealing*

A way to provide the global convergence is to apply Simulated Annealing (Abramson, 1991; Pedroso *et al.*, 2004; Pupeikiene and Mockus, 2005; Mockus, 2002; Aarts and Laarhoven, 1987; Zhang *et al.*, 2010) with a proper selection of SA parameters such as the "initial temperature" x_1 and "cooling rate" x_2. It is well-known that unbounded neighborhood is needed to provide the global convergence (Aarts and Laarhoven, 1987). However, we deliberately restrict the neighborhood to speed-up the calculations. It means that by applying SA we obtain an additional improvement, but not the global convergence. In the case of limited neighborhood, the global convergence is provided by the multi-start algorithm, because then any timetable can be reached with some probability. We achieve a further improvement of the algorithm by automatic optimization of the SA parameters $x = (x_1, x_2)$, using the Bayesian approach (Mockus, 1989).

Now we shall describe in detail a version of SA which is used in this paper. Denote

$$\delta_n = F\big(c, d^{n+1}\big) - F\big(c, d^n\big). \tag{45}$$

Here. d^n is a current timetable, d^{n+1} is a new timetable generated by the closing gap operation. Define the probability

$$p_n = e^{\frac{-\delta_n}{x_1/\ln(1+x_2 n)}}, \quad \text{if } \delta_n > 0, \tag{46}$$

$$p_n = 1, \quad \text{if } \delta_n < 0, \tag{47}$$

where the parameter x_1 is the initial temperature, and the parameter x_2 defines the cooling rate. The SA algorithm means:

$$\text{go to new timetable } d^{n+1} \text{ with probability } p_n. \tag{48}$$

To apply SA to a specific problem, one must specify the parameters x_1 and x_2. The choice can have a significant impact on the efficiency of the algorithm. For example, Fig. 5 shows that the efficiency of SA is a multi-modal function of the cooling rate x_2. This function is stochastic because the results depend on random samples. Thus, the methods of stochastic global optimization are needed to maximize the efficiency of SA.

3.5. *Optimization of Parameters*

Using this timetabling model we need to adapt three heuristic parameters $x = (x_0, x_1, x_2)$. The LR parameter x_0 was fixed for all the examples of this paper. The traditional way to adapt SA parameters x_1, x_2 to a given problem is by experimental calculations. This is a long time manual job. Figure 5 illustrates the manual adaptation of SA parameters. The improvement of the initial timetable of 11,000 penalty points was obtained after 1000 SA iterations at the parameters $x_2 = 9$ and $10 \leqslant X_1 \leqslant 100$. The automatic adaptation using BA is illustrated in Fig. 6. Here almost twice as good improvement of 20,000 penalty points was reached after 500 BA iterations. Figure 4 illustrates how BA manages SA parameters during the first 7 iterations. The complete data of BA optimization are in the file 'Analysis_BA.xml' included in the archive of final results 'tvark.zip' which can be downloaded ending the optimization.

Therefore, in this paper, we apply regular optimization techniques to adapt SA parameters. That is not an easy task, since we have to optimize a multi-modal function with a considerable noise (Mockus, 2000; Dzemyda and Sakalauskas, 2011). The standard methods of stochastic approximation (Kushner and Yin, 2003) are for local optimization. The Bayesian approach (Mockus, 2000) is designed for global optimization of functions with noise. It works by generating a sequence of pairs x_1^n, x_2^n. The next duple x_1^{n+1}, x_2^{n+1} is generated by minimizing the expected deviation from the global minimum. The expected deviation from the global minimum is calculated using some multi-dimensional extension of the Wiener process. The algorithm stops after N iterations. The result x_1^N, x_2^N is considered as the optimal parameter for similar timetabling

	A	B	C	D
1	Iter	Init-temp	Cool-rate	PenaltyPoints
2	1	10.0	1.0	50420
3	2	10.0	10.0	50485
4	3	10000.0	1.0	50430
5	4	10000.0	10.0	50430
6	5	4995.0	4.0	50485
7	6	23.0	3.0	50420
8	7	480.0	3.0	49810

Fig. 4. Illustration of several iterations of BA in search of the best SA parameters.

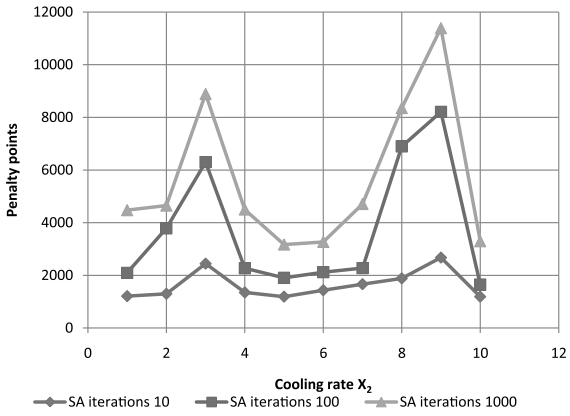


Fig. 5. The best results of Simulated Annealing by 100 samples.

problems. For different timetabling problems, optimization of the parameters is repeated. In the experimental setup, designed for comparison of manual and automatic adaptation of these parameters, we fixed the value of $x_0 = 0.5$, for simplicity. Optimized values of $x^* = (x_1^*, x_2^*)$ are obtained using the data of some specific school. However, to limit the computing time, the values of $x^* = (x_1^*, x_2^*)$ can be used in similar schools as the initial approximation.

The Bayesian Approach (BA) is defined by fixing a prior distribution P on a set of functions $f(x)$ and by minimizing the Bayesian risk function $R(x)$ (DeGroot, 1970; Mockus, 1989). The risk function describes the average deviation from the global minimum. The distribution P is regarded as a stochastic model of $f(x),\ x \in R^m$ where $f(x)$ might be a deterministic or stochastic function BA is a convenient tool to optimize continuous parameters of various heuristic techniques. This procedure is called the Bayesian Heuristic Approach (BHA, Mockus, 1989) and is used in this paper to optimize SA parameters.

Figure 6 illustrates the efficiency of automatic adaptation of parameters using BA. In this example improvement almost doubled: from 11,000 penalty points, using the manual adaptation of SA parameters (Fig. 5), to 20,000, using automatic optimization by BA (Fig. 6). Figure 7 shows how the average BA optimization time depends on BA and SA iterations for a small school. For large schools with a complete set of options the time is longer. Therefore, a multi-processor version of the software is under development. Each processor generates a random initial timetable and performs a local optimization. The results of different processors are compared and the best timetable selected.

Another way to increase the efficiency of search is the improvement of single-session optimization using the BA algorithm with unrestricted neighborhood. Both ways are subjects of future investigations. By the existing software, we can limit the optimization time by applying the best SA parameters defined using BA in similar cases as the first approximation. Figure 8 shows the difference of the average BA results between the large and small schools. In the small school (the lower line denoted as BA-2), the average improvement was almost two times less. This is natural, because timetabling in large schools is

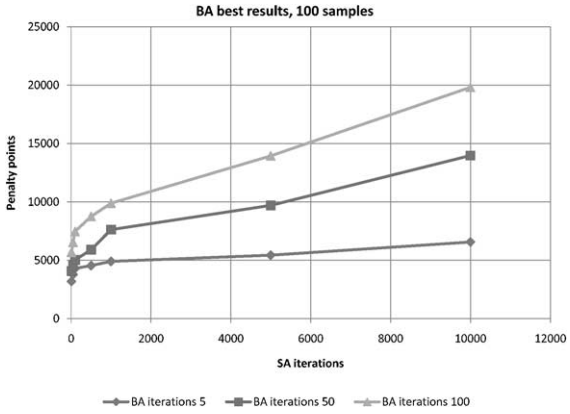


Fig. 6. The best results of Bayesian approach by 100 samples.

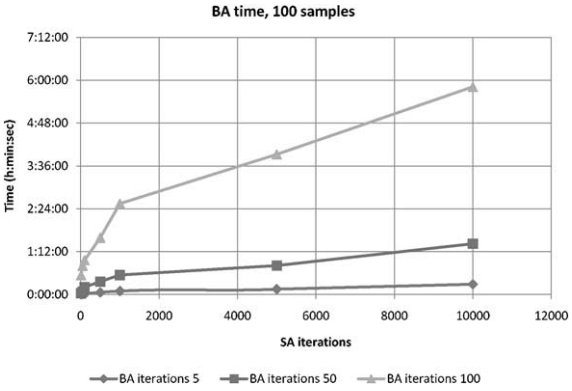


Fig. 7. CPU time of Bayesian approach.

more difficult. Comparison of Fig. 6 with Fig. 8 illustrates the difference between the best and average results obtained through 100 samples of a single multi-start session.

3.6. *Final Multi-Start Algorithm*

SA and BA improve the search significantly but do not provide a convergence to the global optimum, since the neighborhood remains limited to the "achievable" timetables. The global convergence can be provided by a multi-start algorithm because then any timetable can be achieved with some probability. The results of a single multi-start session of 100 samples (starting timetables) are in Figs. 5, 6.

Note that all these parameters are defined by 100 samples of a single multi-start session. The results of other multi-start session did not differ significantly. The statistical analysis of many multi-start sessions requires too much time using the present implementation. Therefore, the multi-processor implementation of the multi-start algorithm is under development where the number of processors will match the number of multi-start

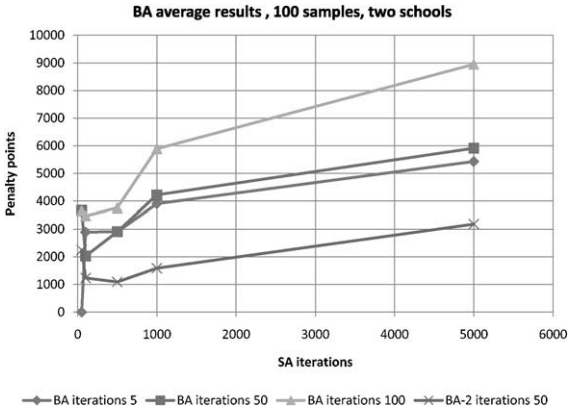


Fig. 8. The average results of Bayesian approach, comparing two schools.

iterations. In addition, a version with unbounded neighborhood is regarded as an important future task that presents an alternative way to achieve the global convergence.

Figure 5 shows that the best result 11,000 of 100 samples, using manually adapted SA parameters, was obtained at the cooling rate $x_2 = 9$ and the number of iterations being 1,000. Using the automatic adaptation by BA, a similar result has been achieved after 100 BA iterations. However, the most important feature of BA is that it saves time of a human operator. For example, a human operator, using 100 BA iterations and 10,000 SA iterations (Fig. 6), was able to obtain the improvement of 20,000 in about five times less time as compared with the improvement of 11,000 obtained by 1000 SA iterations by means of manual adaptation of the parameter x_2 (Fig. 5).

These and the following experimental results serve as an illustration. We consider the web-based software as a tool for future collaboration including the comparison with other timetable optimization models. The web-based software is developed in the form of a Java servlet which provides the most convenient means for repeating all calculations independently by any interested person. Any browser can be used for that.

4. Input Data

We express the soft constraints by conditions (25)–(30). The hard constraints can be represented by conditions (2)–(7), (10), and (11)–(13).

4.1. *Initial and Optimized Timetables*

Initial timetables are generated automatically by generating timetables for teachers and students in some prearranged order trying to favor their requests and keep the hard constraints. The usual result is a feasible timetable with a number of penalty points, mainly for exceeding the time limit $T_{\max}$.

5. Evaluation of Algorithms

The efficiency of different versions of the multi-start algorithm can be estimated as the difference of penalty points between the optimized and initial timetables. The problem is that this difference depends on the evaluation of importance of different factors that reflects the opinion of a particular school. Therefore, we use the same values of importance parameters for most of the examples. These values were defined after consulting administrations of a large and a small school.

Most of these parameters are intended for soft constraints. Two parameters play a double role. The parameter "Exceeding the hour limit $= 2000$" allows exceeding 7 hours, if that improves the timetable significantly. The parameter "mixing students in the group $= 5$" prevents mixing students of different grades.

By comparing the average improvements of timetables shown in Figs. 3, 5, and 6 we estimate contributions of the three multi-start versions as follows:

5,800 penalty points for MC (standard deviation 2,900), 11,000 points for SA with fixed parameters, and 20,000 points for BA with SA parameters optimized using the Bayesian approach. Note that to achieve the improvement of 20,000 points using BA took about five times less human operators' time, at the expense of longer computing time. All that indicates two ways of improvement:

(i) To increase the efficiency of MC by multi-processor implementation, preferably using different processors for different starting points. An advantage is a very high efficiency. A disadvantage is that hundreds of processors should be used to achieve this result.

(ii) Providing the global convergence of BA with unrestricted neighborhoods.

We meet an additional difficulty while comparing the optimization results with the real timetables, since real timetables are for the whole school, and we optimize just for two upper classes. The connecting element is teachers that work in both the upper and lower classes. Their "gaps" in the upper classes can be closed by the corresponding lectures in the lower classes.

Thus, only the bounds of real timetable evaluation can be calculated. We obtain the upper bound equal to 178,830 penalty points for the real timetable by setting the teacher gap penalty to 300 for both the real and optimized timetables. To obtain the lower bound for the real timetable, we set this penalty to zero, and the result is 9,330. The average results of MC algorithms of 23,035 penalty points are between the lower and upper bounds.

6. Web-Based Software for Optimization and Evaluation of timetables

Four servers (last modification in March 2010) are running simultaneously, for reliability:

`http://soften.ktu.lt:8080/`,
`http://optimum2.mii.lt:8080/`,
`http://pilis.if.ktu.lt:8090/`,
`http://kopustas.elen.ktu.lt:8080/`.

Following the local practice, input/output is performed using Excel xml files.

A sample of input data for optimization is on the web-site `http://soften.ktu.lt/ mockus/`, part "Software Systems", folder "Servlet" file 'optimal-1.xml'. A sample for evaluation of the real timetable is 'real-1.xml'. They are samples of a large school with a complete set of options. Smaller schools with limited options are represented by another sample: 'optimal-2.xml'.

To apply the software, the school administrator uploads the initial data and starts the optimization according to the specifications of a particular school.

The software is implemented as the Java servlet. It means that all the calculations are performed by servers, using one of the following five versions.

1. 'TvarkaLT/EN-rnd'
   This version starts a session of a multi-start procedure from a randomly generated initial timetable and ignores hard constraint (6) and soft constraints (28), (30). Permutations are made by closing teachers' gaps. Three possibilities to finish the session can be selected by the menu: Local Random (LR), Simulated Annealing (SA), Global Bayesian (BA). LR optimizes by closing the gaps for teachers selected with some probability x_0. SA is Simulated Annealing with the fixed parameters x_1 and x_2. BA is Simulated Annealing with the parameters x_1 and x_2 optimized using the Bayesian approach.
2. 'TvarkaLT/EN-fix'
   The only difference from the previous version is that it starts from the fixed initial timetable. This is a single-session algorithm to provide a possibility of preferential treatment of important teachers by fixing their sequence.
3. 'TvarkaLT/EN-rnd-up'
   This version starts from a random timetable and includes an additional possibility to finish the optimization by "Local" (LD). The timetable is optimized in three stages: in the first stage, the optimization is performed by closing student gaps, in the second stage, the optimization is achieved by closing teacher gaps, and it is completed by repeating the first stage.
   The hard (2)–(7) and soft (25)–(30) constraints are involved. The group-stability is evaluated in the form of the soft constraint (28). Mixing the subject-groups is limited to different classes of the same grade by the penalty parameter "mixing students", didactic preferences are controlled by the penalty parameter "didactic".
4. 'TvarkaMC'
   The algorithm selects the best randomly generated timetables without additional optimization. It includes hard constraints (2)–(7) and soft constraints (25)–(30). This is the simplest version of the multi-start algorithm.
   A sample 'optimal-1.xml' of the input data for these four versions is on the web, for example
   `http://soften.ktu.lt/ mockus/servlet/contservlet.html`.
   Frequently the evaluation (in terms of penalty points) of the existing school schedules is desirable. It can be performed by means of the evaluation software.

5. 'real'
   This algorithm is to evaluate the real schedules. A sample 'real-1.xml' of the real school schedule is on the web, too. Special interests of a particular school can be represented by selecting penalty points for various soft constraint violations.
   For a preliminary inspection of the results, the optimized timetables are presented on the screen in a simplified form. The help files are provided on-line for different stages of optimization and evaluation. For example, help 1 is about making an initial data file, help 2 is on initial settings of school parameters, help 3 illustrates choosing the optimization algorithm, help 4 shows how to set the optimization parameters, and help 5 illustrates the results of optimization.

Complete information on the initial and optimized timetables for all the school and for individual teachers and students can be downloaded as a 'zip' archive. In addition, the archive provides some additional information about the optimization process, see Fig. 4.

The authors think that the web-based implementation is an important part of timetabling investigations because no statistical data collected while considering some set of the selected examples, can replace on-line calculations. The main reason is that the function to be optimized reflects the subjective opinions of potential users which are not willing to disclose these opinions, as usual.

7. Conclusions

1. The experimental calculations, using the examples of large and small high schools, show that the optimized timetables are better as compared with the existing timetables formed with the help of popular commercial software.
2. The contributions to the improvement of the initial timetable, achieved by different versions of a multi-start algorithm are as follows: MC: 5,800, SA (manual adaptation): 11,000, BA (automatic adaptation): 20,000.
3. Application of BA saves the human operators' time at the expense of longer computing time. 4. These results indicate different ways to a further improvement of the optimization efficiency:
   (a) multi-processor implementation of multi-start algorithms,
   (b) unbinding the neighborhood for global convergence of the single start BA.
5. To provide entirely automatic timetabling, the optimization should be able to include all the classes, not only the upper ones; this possibility is limited now by an increased computing time.
6. The automatic optimization of SA parameters using BA, proposed in this paper, significantly improves the efficiency of SA.
7. The web-based implementation is an important part of school timetabling for two reasons:
   (a) implementation of software as the Java servlet establishes conditions for its application in any school with the internet connection, independently of the computing environment,

(b) the function to be optimized reflects the subjective opinions of potential users which are not willing to disclose these opinions, as usual. Thus, the statistical data collected while considering some set of the selected examples cannot replace on-line calculations.

8. The paper describes apparently the first web-based platform-independent implementation of the software as a Java servlet.
9. The existing commercial and open software facilitates manual timetabling by providing realistic school models by limited or no regular optimization.
10. Different timetabling optimization algorithms have been proposed and investigated in many publications using the simplified high school models. In this work, an attempt is made to develop and apply regular optimization techniques in timetabling models which are entirely compatible with Lithuanian high schools.

Acknowledgments. The work was supported by the Lithuanian State Science and Studies Foundation.

References

Aarts, E, Laarhoven. P.V. (1987). *Simulated Annealing: Theory and Applications*. Reidel, Dordrecht.

Abramson, D. (1991). Constructing school timetables using simulated annealing: Sequential and parallel algorithms. *Management Science*, 37, 98–113.

ASC (2010). *Timetables ASC*. `http://www.asctimetables.com/`.

Bull, J.M., Smith, L.A., Pottage, L., Freeman, R. (2001). Benchmarking Java against C and Fortran for scientific applications. In: *Proceedings of the 2001 Joint ACM-ISCOPE Conference on Java Grande*, Palo Alto, California, pp. 97–105.

DeGroot, M. (1970). *Optimal Statistical Decisions*. McGraw-Hill, New York.

Dzemyda, G., Sakalauskas, L. (2011). Large-scale data analysis using heuristic methods. *Informatica*, 22, 1–10.

Fudenberg, D., Tirole, J. (1983). *Game Theory*. MIT Press, Boston.

Kushner, H.J., Yin, G.G. (2003). *Stochastic Approximation Algorithms and Applications*, 2nd edn. Springer, New York.

Mimosa (2010). *MIMOSA Scheduling Software*. `http://www.mimosasoftware.com/`.

Mockus, J. (1989). *Bayesian Approach to Global Optimization*. Kluwer Academic, Dordrecht.

Mockus, J. (2000). *A Set of Examples of Global and Discrete Optimization: Application of Bayesian Heuristic Approach*. Kluwer Academic, Dordrecht.

Mockus, J. (2002). Bayesian heuristic approach to global optimization and examples. *Journal of Global Optimization*, 22, 191–203.

Pedroso, J.P., Moreira, N., Reis, R. (2004). A web-based system for multi-agent interactive timetabling. In: *ICKEDS'04: International Conference on Knowledge Engineering and Decision Support*, Porto, Portugal, pp. 1–6.

Pupeikiene, L., Mockus, J. (2005). School schedule optimization program. *Information Technology and Control*, 34, 161–170.

Pupeikiene, L., Mockus, J. (2010). *School Scheduling Optimization, Investigation and Applications*. Lambert Academic, Saarbrcken.

Schaerf, A. (1999). A survey of automated timetabling. *Journal of Operational Research Society*, 13, 87–127.

Smykalov, P. (2010). *School Timetabling: Rector*. `http://www.rector.spb.ru/uk/index.html`.

Wren, A. (1996). Scheduling, timetabling and rostering – a special relationship? In: *Practice and Theory of Automated Timetabling*, *Lecture Notes in Computer Science*, Vol. 1153. Berlin, Springer, pp. 46–75.

Zhang, D., Liu, Y., M'Hallah, R., Leung, S.C. (2010). A simulated annealing with a new neighborhood structure based algorithm for high school timetabling problems. *European Journal of Operational Research*, 203, 550–558.

J. Mockus graduated Kaunas University of Technology, Lithuania, in 1952. He got his doctor habilitus degree at the Institute of Computers and Automation, Latvia, in 1967. He is a principal researcher of System Analysis Department, Vilnius University, Institute of Mathematics and Informatics and professor of Kaunas University of Technology. His research interests include global and discrete optimization.

L. Pupeikienė graduated Kaunas University of Technology, Lithuania, in 2004. She got her PhD degree at the Institute of Mathematics and Informatics, in 2009. She is an associated professor of Department of Information Technologies at Vilnius Gedimino Technical University. Her research interests include discrete optimization and scheduling.

Apie „multi-start“ algoritmus vidurinių mokyklų tvarkaraščiams optimizuoti

Jonas MOCKUS, Lina PUPEIKIENĖ

Straipsnis nagrinėja svarbų tačiau sudėtingą vidurinių mokyklų tvarkaraščių optimizavimo uždavinį. Pateikiamas „multi-start“ (daugelio pradinių taškų) algoritmas naudojant „modeliuojamą atkaitinimą“ (SA), kurio parametrai adaptuojami naudojant Bayes'o požiūrį. Šis algoritmas lyginamas su kitais tvarkaraščių optimizavimo algoritmais. Siūlomas algoritmas užtikrina konvergavimą didinant pradinių taškų skaičių, tačiau konverguoja lėtai. Todėl pirmasis straipsnio tikslas yra eksperimentiškai palyginti įvairius šio algoritmo variantus. Antrasis tikslas yra jo įgyvendinimas interneto aplinkoje taip, kad būtų patogu naudoti mokyklose. Tai duoda galimybę nepriklausomiems ekspertams atlikti algoritmo bandymus, patikrinant jo efektyvimą konkrečioje mokykloje. Ši galimybė svarbi todėl, kad algoritmo efektyvumas priklauso nuo mokyklos sąlygų ir jas apibrėžiančių parametrų. Optimizavimo programos papildytos programine įranga esamų tvarkaraščių įvertinimui.