Application of greedy algorithms for the formation of the educational schedule in the higher education

A A Popov, O N Lopateeva, A K Ovsyankin and M M Satsuk

Reshetnev Siberian State University of Science and Technology, Krasnoyarsk, prospect "Krasnoyarsky rabochy “31, Russia

E-mail: tolynbms@yandex.ru

Abstract. The problem of improving the quality of the organization of the educational process in a higher educational institution (university) is solved by using greedy algorithms. This task is the main one in the activity of the dispatching service of each university. A well-prepared schedule should ensure a uniform load of student groups and faculty. A greedy algorithm is an algorithm consisting in making locally optimal decisions at each stage, assuming that the final solution also turns out to be optimal. A greedy resource allocation algorithm refers to an algorithm according to which the resource allocation process can be represented as a sequence of steps. At each step, an optimal distribution of some resources occurs under certain conditions, which does not change in the future. Numerous attempts to automate the process of forming the university curriculum did not solve the problem in full. Accordingly, in order to solve this problem more effectively, as well as reduce the time and money spent on resource allocation, it is necessary to optimize existing solutions. The purpose of this work is to develop an algorithm that will improve the quality of curriculum formation based on the use of greedy algorithms.

1. Introduction

The composition of many software systems for the formation of curricula as a central functional task is developing the task of creating a calendar curriculum.

This problem belongs to the class of combinatorial optimization problems.

Note that, when solving a combinatorial optimization problem on a computer, we are actually interested not just in an approach that allows us to solve it exactly, but also in an approach that allows us to precisely solve a problem using the minimum amount of computer computing resources.

The main goal when choosing an approach is to find an algorithm that reduces the accuracy requirements for solving a problem that consumes a minimal amount of time.

Many researchers have solved this problem since the late 80s of the 20th century. Therefore, in [1], the authors study approaches and approximate methods for formalizing important in practical terms classes of discrete optimization problems and constructing methods for solving them. Examples of solving a number of problems relating to the optimization and minimization of the costs of computer resources are given.

An approximate approach to solving complex problems is to decompose a complex problem into a number of simpler ones.
Approximate approaches are sometimes called heuristic [2], because they are based on heuristics - reasonable considerations that do not necessarily lead to a goal. The most commonly used heuristics include greed [3, 4], consistent improvement [5] and random search.

The use of an approximate approach to solving the problem can be explained not only by purely computational difficulties [6], but also by the incompleteness or proximity of the available information [7], as well as the lack of stability of solutions to fluctuations of the initial data of the problem.

An approximate approach to solving complex problems consists in decomposing a complex problem into a number of simpler ones and in constructing a procedure that allows one to reduce the solution of the original complex problem to the solution of the obtained simple ones. The application of this approach to solving combinatorial optimization problems has led to the appearance of so-called greedy algorithms.

The use of greedy algorithms is often used in scheduling educational institutions. When scheduling, the problem of optimal resource management arises: the teaching staff and the classroom fund.

In [8], the author suggests using greedy algorithms for scheduling classes in educational institutions. Also proposed are the stages of curriculum formation. Among the advantages, it is possible to generate acceptable schedule options already from the first iteration. The algorithm provides for the possibility of a significant improvement in the schedule by adding additional criteria for evaluating the freedom and quality of the location of classes in the schedule. A preventive maintenance strategy is effectively and fully can be implemented in that case if staff has the knowledge, skills and time needed to carry out the relevant activities.

2. Methods
The task of creating a curriculum at a university belongs to the class of applied problems, the solution of which is to find a partition of many elements - resources, each of which has a numerical characteristic - size - into a given number of disjoint subsets, whose sizes are the least different or closest to each other. These are the so-called uniformization problems.

One of the tasks of this class is the task “Distribution of resources”, which is formulated as follows [9, 10].

The set \( R \) of resources \( r \in R \) is given. For any \( r \in R \), the size \( p(r) \in R \) is defined.

It is required to find a partition of the set \( R \) into \( m \) subsets such that, where \( S \) is the set of all partitions of the set \( R \) into \( m \) subsets;

The formulated problem is NP-hard; there is no exact polynomial algorithm for solving it. We need to find an approximate algorithm.

An approximate approach to solving complex problems consists in decomposing a complex problem into a number of simpler ones and in constructing a procedure that allows one to reduce the solution of the original complex problem to the solution of the obtained simple ones. The application of this approach to solving combinatorial optimization problems has led to the appearance of so-called greedy algorithms.

A greedy resource allocation algorithm refers to an algorithm according to which the resource allocation process can be represented as a sequence of steps. At each step, an optimal distribution of some resources occurs under certain conditions, which does not change in the future. There are two classes of greedy resource allocation algorithms: list and brute force.

The simplest greedy algorithm for solving the “Resource Allocation” task is shown in figure 1.

The condition under which the resource is distributed optimally at some step in the sense of the functional \( F \) is that not all resources allocated at the previous steps can be redistributed, and any distribution of all remaining resources in the set \( R \) will not have any effect on the functional \( F \). The latter, naturally, happens when, for example, the hosted resource is the last or all remaining resources are equal to zero.

To solve the “Resource Allocation” problem, there is another, wide class of greedy algorithms. In a generalized form, the algorithm of this class is presented in figure 2.
Figure 1. The simplest greedy algorithm for solving the problem “Resource allocation.”

Figure 2. A generalized view of the greedy algorithm for solving the problem “Resource allocation.”

A specific greedy algorithm of this class is obtained by clarifying the concepts of “some set” and “some condition” and specific values of variables.

Clarification of the concept of “some set” is carried out in two ways. In the first method, at the j-th step of the algorithm, the maximum unallocated resources are selected from the set R from the set R. In the second method, at the j-th step, any unallocated resources are selected. The concept of “some condition” is specified in three ways. Common to all three methods is that resources allocated at any step are not redistributed at all subsequent steps of the algorithm. In the first method, the search for the optimal distribution is carried out under the assumption that the sizes $p(r) = 0$ of all resources, i.e. all resources except those already distributed and distributed at the j-th step. In the second method, a distribution search is carried out under the assumption that one continuous resource is formed from all resources. A continuous resource can, in contrast to discrete resources, be divided into m parts of arbitrary size with a total size equal to the size of a continuous resource. In the third method, the distribution is searched under the assumption that all the resources of the set will be distributed according to the simplest greedy algorithm.
The use of the concept of greed in a minimum size leads to the simplest type of greedy algorithms - list algorithms.

The general view of the list algorithm is shown in figure 3.

A distinctive feature of the list algorithm is its low time complexity, which does not exceed $O(n \log n + n \log m)$. From an intuitive point of view, list algorithms look very convincing: any list algorithm allocates $n$ resources in $n$ steps, and at each step, the placed resource is optimally added to the resources allocated in the previous steps of the algorithm.

It is easy to determine the subset $R_i$ into which the resource $r$ is distributed if there is a sequence of subsets $R_{i-1}, R_{i-2}, ..., R_0$ ordered in order of non-increasing size. Then if $p(r) > E$, then the resource is distributed in $R_0$; if $p(r) < E$, then the resource is allocated to $R_i$. After the subset $R_i$ is defined and the resource $r$ is allocated to it, the resulting subsets should be reordered by not increasing the size.

As a result of the analysis of the requirements for the schedule of training sessions, a decision was made on the need to develop an algorithm that would include the possibility of expanding the list of requirements for the schedule of classes, as well as the possibility of regulating priorities for fulfilling individual requirements in scheduling. Figure 4 shows a block diagram of the implementation stages of the proposed list greedy algorithm.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure3.png}
\caption{General view of the list algorithm.}
\end{figure}
The main stages of the implementation of the proposed list greedy algorithm for the formation of the university curriculum are as follows.

- For each of the four lists – sets $\mathcal{R}_{11}, \mathcal{R}_{12}, \mathcal{R}_{21}$, sorted by attributes: $kol_p$ (number of hours (pairs)), $kol_g$ (number of groups), $kol_lex$ (number of lectures), $kol_st$ (number of students), $av_rp$ (average grade of teachers) we form queues for inclusion in the schedule. For all the “works” of all four lists, we determine the “size of work” and “complexity” of mastering the work. By “size of work”, we mean the total number of tapes for all types of classes in the discipline of the basis of this work.

- For all four sets we set the maximum allowable “sizes” of the work. Given the maximum “sizes” of blocks, we are reviewing the lists of work. If the “job size” is larger than the maximum allowable “size” for a given list $\mathcal{R}_{ij}$ ($i = 1,2; j = 1,2$), then this work is divided into two smaller jobs, otherwise we leave the job unchanged. We divide the work into two smaller jobs so that the size of the first of them is equal to the maximum allowable “size” for a given one $\mathcal{R}_{ij}$ ($i = 1,2; j = 1,2$).

The size of the second work may turn out to be larger than the maximum allowable “size”, in this case we will again divide it into two smaller works, and so on, until the sizes of all the works become smaller, or equal to the maximum allowable “size” for the given one $\mathcal{R}_{ij}$ ($i = 1,2; j=1,2$). The process
of dividing the work into two smaller works is carried out taking into account the requirements of SanPiN [11] and the rules of the organization of the educational process in this university.

The result of this process is 4 new lists consisting of works with sizes smaller or equal to the maximum allowable sizes for the data $\mathbf{R}_{ij}$ ($i = 1, 2; j = 1, 2$).

- Each of $\mathbf{R}_{ij}$ ($i = 1, 2; j = 1, 2$) work lists is divided into $d(\mathbf{R}_{ij})$ ($i = 1, 2; j = 1, 2$) sub lists, where are $d(\mathbf{R}_{ij})$ the maximum allowable work sizes for the data $\mathbf{R}_{ij}$. We sort each of the sub-lists by $kol_g$ (number of groups), $kol_lex$ (number of lectures), $kol_st$ (number of students), $av_rp$ (average grade of teachers).

- In the list $\mathbf{R}_{11}$ (the work of the first week of the first shift), we select the sublist (turn) of the most "fat" jobs - that is, those jobs whose dimensions are equal to the maximum allowable size for $\mathbf{R}_{11}$. We extract the next "work" from the queue and try to include it in the schedule.

When "work" is included in the schedule, the corresponding "resources" are consumed. The algorithms for allocating "works" with "resources" and the inclusion of "works" in the schedule differ significantly from each other depending on the composition of the data included in the "work" (the number of lectures, practices and laboratory works included in the work).

The sequence of application of the basic algorithms is determined by the complexity of the work, depending on this complexity, attempts are being made to include this work in the schedule entirely on different days and different pairs.

The total complexity of the training load of the $g$-th group on the $d$-th day is denoted $\frac{\psi}{g}(x_{gdl})$. According to the recommendations of the SES, the complexity of the workload should be minimal on Monday, increase by Wednesday, and decrease by the end of the school week. The distribution of the workload by the days of the week can be set algorithmically. You can distribute the workload by the days of the week by entering the share of the weekly workload of the student group on the $d$-th day of the week: $\eta_d, \sum D_d \eta_d$. In this case, the function $\frac{\beta}{pd}(x_{gdl})$ for the criterion for accounting for the load balance in the schedule of student groups is as follows

$$\frac{\beta}{pd}(x_{gdl})=\sum_{g=1}^{G} \sum_{d=1}^{D} \frac{\psi}{g}(x_{gdl})-\eta_d \sum_{d=1}^{D} \psi(x_{gdl}) \Rightarrow \min \quad (1)$$

For the algorithmic support of this balance, the following sequence of application of basic algorithms is laid. For disciplines with a high complexity index, the selected local algorithms are applied in the sequence: Wednesday, Thursday, Tuesday, Friday, Monday, and Saturday. For disciplines with a low complexity index, the sequence of applying local algorithms is different: Saturday, Monday, Friday, Tuesday, Thursday, and Wednesday. For disciplines with an average complexity index, the sequence of applying local algorithms is as follows: Monday, Tuesday, Wednesday, Thursday, Friday, and Saturday.

For disciplines with a low complexity index, the sequence of applying local algorithms is as follows: Monday, Tuesday, Wednesday, Thursday, Friday, and Saturday. For each day, the sequence of applying local algorithms is direct, i.e., if, for example, the size of the work is 1 and the work consists of one lecture, then the sequence of applying local algorithms will be as follows: $alg\_lex (1)$, $alg\_lex (2)$, $alg\_lex (3)$, $alg\_lex (4)$, i.e. The algorithm tries to include this lecture in the schedule, first with the first pair, then the second, then the third and, finally, the fourth.
• We proceed to the next work of the sublist under consideration and carry out a series of attempts to include this work according to the scheme described in paragraph 4. Therefore, we look through all the work of the sublist.

• If, after viewing all the “works” of the sublist under consideration, some “works” are not included in the schedule due to the congestion of individual “resources-audiences”, an algorithmic assignment to “works” of other audiences is carried out. This possibility is feasible because several alternative audiences are assigned to the audience directory for each audience. This possibility is feasible because several alternative audiences are assigned to the audience directory for each audience. Alternative audiences are algorithmically assigned to the work, and the queue of non-scheduled “works” is worked out again. This process continues until either the “work” from the sublist (queue) in question is included in the schedule, or it is concluded that it is impossible to include some work in the schedule.

• In the event that after repeated application of the procedures for including work in the schedule (items 4, 5, 6), works not included in the schedule remain in the sublist, and the following actions are performed. Each work not included in the schedule is divided into two: the size of the first of them is d-1, and the second is 1 (where d is the size of the work not included). The first works are added to the sub-list of works of size d-1, the second to the sub-list of works of size 1.

• Supplemented by non-scheduled work sub-list of works of size d-1 is sorted by kol_g (number of groups), kol_lex (number of lectures), kol_st (number of students), av_rp (average grade of teachers). This forms a new lineup for inclusion in the schedule.

• Repeat all actions for inclusion in the work schedule with a size equal to d-1, that is, those actions described in clauses 4, 5,6,7,8 of the algorithm, then all actions for inclusion in the work schedule with size equal to d-2 and, finally, all actions for inclusion in the schedule of work with a size equal to 1.

• As a result of repeated application of items 4, 5, 6,7,8,9, all the works of the list either will be included in the schedule or will be included in the list of non-included works. Note that according to the developed algorithm of work, those included in the list of not included jobs have a size of 1.

• We carry out all the steps to include in the schedule of work, first a list  $\mathbb{R}_1^{12}$, then a list $\mathbb{R}_2^{12}$, and finally a list $\mathbb{R}_2^{22}$. To do this, you need to perform the actions described in paragraphs 4, 5,6,7,8,9,10 three more times.

The scheduling of classes begins with the entry of data on the workload. In addition, for the algorithm to work, data on audiences is required, if they were not entered earlier, they are entered. Depending on the options for implementing the user interface, lists of teachers and groups can be generated automatically, based on data on the workload, and then supplemented with missing data, or can be entered manually.

As a result of processing the data on the training load of the flows, a list of classes is formed (Table 1).

| Group  | Discipline name | Occupation Type | Teacher       | Additional requirements |
|--------|-----------------|-----------------|---------------|------------------------|
| BPA 17-01 | Programming    | Lecture         | Ivanov I.I.   | Computers              |
| BPA 17-01 | Programming    | Practice        | Ivanov I.I.   | Computers              |
| BIN 17-03 | Programming    | Lecture         | Ivanov I.I.   | Computers              |
| MIV 19-01 | English language | Practice     | Ivanova I.I.  | No                     |
| MPA 19-01 | English language | Lecture       | Ivanova I.I.  | No                     |
As you can see from the example, repetitions can occur in the list of classes. This is because each lesson is a separate object, which must be placed in the schedule. Duplication occurs due to the fact that within one week two classes of the same type are held.

3. Discussion
The tasks of preparing the curriculum are NP-hard task; therefore, various approximate approaches and methods are used to solve them.

To solve the task of forming the curriculum formulated in the work, which is NP-hard task, there is no exact polynomial algorithm to solve it. We need to construct an approximate algorithm. To construct such an algorithm, it is proposed to use greedy algorithms.

The essence of greedy algorithms, their similarities and differences with dynamic programming algorithms, is revealed. It is noted that for problems solved by greedy algorithms, two features are characteristic: the greedy choice principle and the optimality property for subtasks.

A characteristic feature of greedy algorithms is that they often use heuristics. Heuristics do not give any guarantees that a solution will be found, nor that it will be optimal.

The general descriptions of the list greedy resource allocation algorithm are presented.

An algorithm has been developed for solving the problem of forming a university curriculum.

4. Conclusion
To solve the task of forming the curriculum formulated in the work, which is NP-hard task, there is no exact polynomial algorithm to solve it. We need to construct an approximate algorithm. To construct such an algorithm, it is proposed to use greedy algorithms.

Schematic descriptions of two greedy algorithms for solving the “Resource Allocation” problem are given. It is shown that a specific greedy algorithm of this class is obtained by clarifying the concepts of “some set” and “some condition” for specific values of variables.

An algorithm has been developed to improve the quality of the organization of the educational process in a higher educational institution (university) using a greedy list algorithm, with which it is possible to ensure uniform loading of student groups and faculty.

In the future, it is planned to create an information system using the developed algorithm.

References
[1] Glushkov V M, Kasphitskaya M F and Sergienko I V 1980 Formalization and solution of one class of discrete optimization problems J. Comput. mate. and mate. Fiz. 20
[2] Kaushik K 2019 Optimizing-engineering problems through heuristic techniques (London)
[3] Kononov A V and Kononova P A 2014 Approximate algorithms for NP-hard task (Novosibirsk)
[4] Fedoryaeva T I 2011 Combinatorial Algorithms (Novosibirsk)
[5] Okulov S M 2019 Discrete programming
[6] Korbut A A and Finkelstein Yu Yu 1983 Approximate methods of discrete programming Academy of Sciences Tech Cybernetics B 1 165-76
[7] Traub J, Vasilkovsky G and Vozhnyakovskyy H 1988 Information: uncertainty, complexity
[8] Beregovykh Yu V, Vasiliev B A and Volodin N A 2009 Class Schedule Algorithm Artificial Intelligence 2
[9] Raffgarden Tim 2020 The Perfect Algorithm Greedy algorithms and dynamic programming (St. Petersburg)
[10] Raffgarden Tim 2019 The Perfect Algorithm The basics (St. Petersburg)
[11] SanPiN 2.4.2.2821-10 Sanitary and epidemiological requirements for conditions and organization of training in general educational organizations