Features of Constructing Scheduling Algorithms in Enterprise Planning Systems*

Alexander A. Chernenko [1] [0000-0002-8182-0005], Pavel Yu. Buchatskiy [1] [0000-0002-3161-6567], Andrey V. Shopin [1] [0000-0001-9504-7378], Semen V. Teploukhov [1] [0000-0003-0099-0369]

[1] Adyghe State University, Maykop, Russia

spiritfov@yandex.ru
butch_p99@mail.ru
ashop409@gmail.com
mentory@yandex.ru

Abstract. Features of the implementation of scheduling algorithms that underlie modern concepts of production planning (Advanced Planning & Scheduling, Enterprise Resource Planning, and Manufacturing Execution Systems) are considered in the article. A generalized scheduling problem is formulated and its belonging to the NP class is proved by polynomial reduction to the traveling salesman problema. Conceptual schemes of algorithms for solving this problem at different stages of the schedule's life cycle have been developed: an algorithm without decision-making procedures; an algorithm using decision-making procedures and an algorithm with optimization of an acceptable schedule. For each type of algorithm, a place in the hierarchy of enterprise planning systems is defined, the main provisions on work efficiency in a complex production system are formulated, and the mathematical apparatus of scheduling theory is considered and some recommendations for its application are given. The interrelation of the application of analytical and heuristic procedures in finding an acceptable solution is shown.

Keywords: planning algorithms, scheduling systems, greedy algorithms, heuristic algorithms.

1 Introduction

Schedule theory is used in such subject areas as production management, traffic management, project planning, resource management in computer systems. However, the variety of mathematical models and scheduling methods usually poses an inevitable problem for applied mathematicians and programmers to construct fast algorithms and their effective software implementation, taking into account the specifics of the problem being solved. The use of standard solutions for such purposes is extremely limited

and inefficient. A more promising solution is the use of an object-oriented approach, which implies the creation of a system of classes and mechanisms for their interaction [1].

To date, to solve the task of scheduling, the capabilities of information systems are actively used, taking into account the world experience of the largest enterprises in various industries. Such systems are based on the concept of Enterprise Resource Planning (ERP), covering a wide range of planning and resource management tasks. Along with ERP, systems designed to solve highly specialized tasks are widespread. Among these systems, the following can be distinguished: Advanced Planning & Scheduling (APS) – development of detailed plans with the ability to control and account for changes; Manufacturing Execution System (MES) – solving tasks of operational scheduling and scheduling; and Supply Chain Management (SCM) [2].

The task of scheduling is one of the main ones in scheduling theory. A characteristic feature of most problems of this class is their NP - complexity, that is, the impossibility of finding the exact solution in polynomial time. Complexity theory allows us to answer the question of whether a particular problem belongs to the class of polynomially solvable – P. The problem of the relation of the classes of problems P and NP has existed for many years, therefore heuristic algorithms are used to solve NP problems that find feasible solutions in polynomial or pseudopolynomial time.

From a mathematical point of view, the task of scheduling is formulated as the problem of combinatorial optimization of servicing a finite set of operations in a system containing a limited set of pieces of equipment. A lot of algorithms have been developed that give an exact solution to classical problems in a time-limited by a polynomial in the length of the input data. For example, these are various sorting algorithms; classic algorithms for solving single-instrument problems of scheduling theory; schedule with interruptions, satisfying the deadlines for an arbitrary number of machines; schedule for two machines with a minimum total service time and others [3].

However, for most of the practical problems arising in industry, transport, and other specialized technical systems, algorithms for finding optimal solutions in polynomial time are unknown. This is also because such problems are difficult to formalize (or impossible), that is, mathematical models of such problems may contain several assumptions that affect the quality of the output data, or schedules.

One of the ways to solve this problem is the transition from the search for the optimal solution to the acceptable one. The Boolean penalty function was chosen as the objective function, for which the result can be easily interpreted into the economic indicators of the schedule (cost) [4].

The development of new and modification of existing mathematical models of scheduling problems in various fields of industry, as well as algorithms for finding feasible solutions to such problems, is an actual direction in the development of scheduling theory.

2 Mathematical Schedule Problem

Consider the following problem. There are many devices $E = \{1, \ldots, n\}$. A certain fund

of time corresponds to each device H_k ($k \in E$). Nomenclature are available $N = \{1, \dots, m\}$, on which many operations are defined $R = \{r_{ij}, i \in N, j = 1, \dots, p_i\}$. Precedence relationships are defined over the entire set of operations $j - 1 \rightarrow j$. The processing time of the r_{ij} operation on the device is known as $E_k - t_o(r_{ijk})$. The "classic criteria" is the minimization of the total processing time of an item, that is:

$$\sum_{i=1}^{m} \sum_{j=1}^{p_i} \sum_{k=1}^{n} r_{ijk} \rightarrow min$$

This problem belongs to the class *NP*. We prove this statement using the lemma "on reducibility".

Lemma 1 ("on reducibility")

Let the tasks $U, Q \in NP$. Then the following statements are true:

1. If $Q \in P$, and the problem U polynomially reduces to the task Q, then $U \in P$.
2. If $Q \in NP$, and the problem U polynomially reduces to the task Q, then $U \in NP$.

It is known that the salesman problem $\in NP$ [5]. Let's carry out the polynomial reduction of the formulated problem (we denote it by U) to the traveling salesman problem using Lemma 1.

Let's represent the set E in the form of a directed graph G without loops with many vertices $V = \{1, 2, \dots, k\}$. The arcs between the vertices have weights corresponding to the values of $t_o(r_{ijk})$ elements of the set R. The minimum execution time for one element corresponds to the existence of a Hamiltonian graph in the task Q. Obviously, the reduction of the sets R and E to the sets G and V is polynomial.

If the Hamilton cycle exists in the traveling salesman problem, then it follows that problem U also has an optimal solution. Therefore, if the problem Q is polynomially solvable, then the problem $U \in P$. And vice versa, if there is no polynomial algorithm for problem Q, then problem U is not solvable in polynomial time.

The relationship between the classes P and NP is opened in the theory of *NP*-completeness. However, the fact that no polynomial algorithm was found for any NP-complete problem indirectly confirms the strict inclusion hypothesis $P \subset NP$, that is $P \neq NP$ [6].

3 Scheduling Algorithms Without Any Decision-Making Procedures

Let's consider the special case when it is required to find any feasible solution in task U. The optimization criteria, in this case, is not defined.

Tasks of this type are the most common in practice. The obvious solution, in this case, is the following – it is necessary to take the operation r_{ij} and assign it to the machine E_k from the set R at each iteration. In this case, two conditions must be taken into account: preservation of the relations of the precedence of operations; the sum of all operations assigned to the machine does not exceed the value H_k [7].

The algorithm for solving the problem is presented in **Fig. 1**. Here *Sorting(R)* – sorting procedure for multiple operations, *Error* – flag of an error that occurs when one of

the conditions of the loop. The error occurs in two cases: when at the next iteration it is impossible to assign an operation r_{ij} to E_k; when, as a result of the assignment, the moment of execution of the operation r_{ij+1} occurs before the completion of the processing of the operation r_{ij}.

This task can be considered as the task of assessing the possibility of performing item *N* on a variety of device *E* in the hierarchy of production planning systems [8, 9]. These are various workshop tasks: calculating the schedule for the workshop, shift, a separate machine, the entire enterprise.

The method of sorting the set of input data affects the result of solving the problem. It can find a valid or optimal (in some cases) solution on the same set of input data using sorting. For example, the sequence *EDD* (requirements are served in non-decreasing deadlines) is given an optimal solution to the problem $1 \mid\mid \sum U_j$. In terms of scheduling theory, this problem has the following formulation: minimizing the number of late requirements [10].

Heuristic procedures together with optimal sequences can be used to solve such problems.

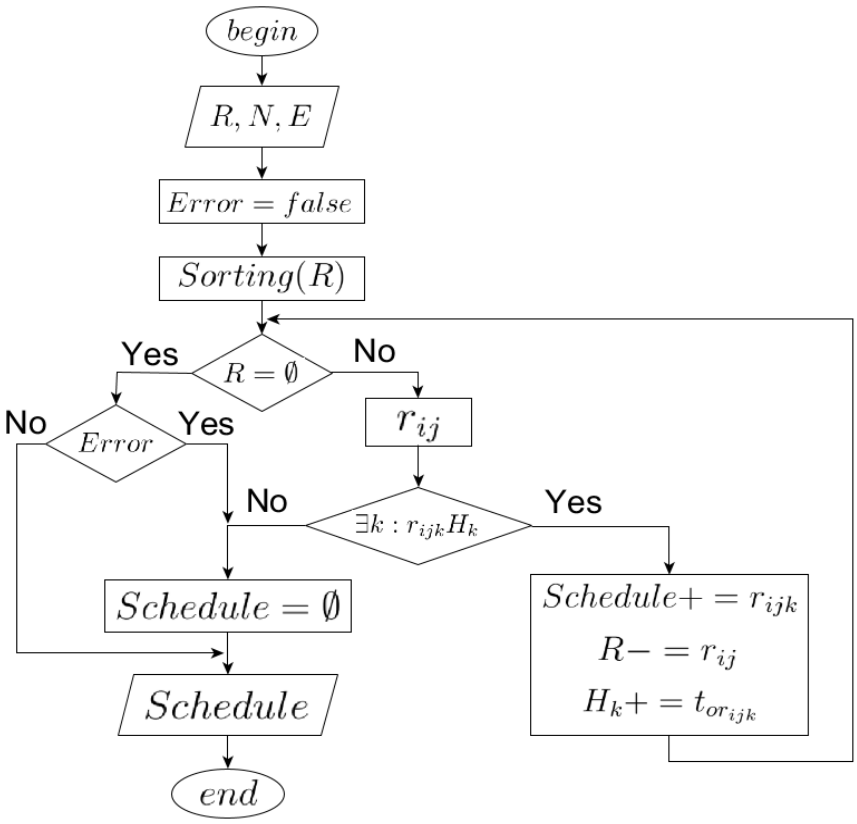

Fig. 1. Algorithm without any decision procedures

They are based on the knowledge of domain experts, calculations, and other empirical data. The heuristic method for sorting a set of input data is based on the following position: the elements of the set *R* are divided into several groups, the elements of each of which are sorted according to a certain rule. Thus:

$$R = (X_1, X_2, \ldots, X_n)$$

where X_i – is the setting in which operations are sorted following rule *i*; *n* – several groups/rules.

So, the work of [11] offers the following methodology for splitting the set of operations. The initial set is divided into three subsets: large works (their number is limited by a constant), medium (have a small total length), and small (short works). The process of building a schedule runs from large to small jobs. At the same time, medium and small operations are assigned to machines using a greedy algorithm. In terms of scheduling theory, such a technique is the most consistent with the maximum load criterion for machines:

$$\sum_{k=1}^{n} H_k - H'_k$$

where H'_k – actual device load E_k.

Indeed, most of the time fund of the device is in lengthy operations. The other operations are more flexible, that is, characterized by shorter processing times, and can be reassigned to other machines.

Despite the relative simplicity of the implementation, the assumption made in the formulation of the original problem significantly narrows the scope of the algorithm for calculating the schedule without decision-making procedures. Such algorithms are actively used in systems of the APS class, for which the main factor is the time for constructing a schedule at very large values N [12].

4 Scheduling Algorithms with Decision-Making Procedures

For several other production tasks, a simple answer to the question of the existence of an acceptable solution is not enough. Such tasks belong to the class of operational planning and are characterized by small values of the planning horizon. To find solutions to such problems, advanced algorithms are used. They can still apply the sorting procedure for the input set of operations [13, 14]. But in this case, the application of this procedure can only become one of the favorable factors for reducing the computational complexity of the combinatorial problem, but it is not a factor determining the quality of the solution. The efficiency of scheduling calculation algorithms with decision-making procedures, based on the name, is determined by the branching depth in the search tree for alternative solutions (**Fig. 2**).

The choice of a vertex is determined by a set of rules that are formed during the statement of the problem. As with sorting, rules can be analytical or heuristic.

The procedure *Pull* is responsible for choosing the operation r_{ij} from the set *R* The

difference between analytical and heuristic rules is that in the second case, the consequences of the choice are not evaluated. The heuristic rules for choosing a vertex are based on the service time of operation r_{ij} on the device E_k. The following rules are existed [15]: Shortest imminent operation (SIO), Longest remaining time (LRT), First off – first on (FOFO), First in – first out (FIFO), Last in – first on (LIFO), and Pair comparison (R). There are also combinations of the above rules, for example, LRT + SIO, LRT + R, etc. Here the "+" sign plays the role of a condition. That is, if the first rule cannot be applied, then preference is given to another, but not vice versa.

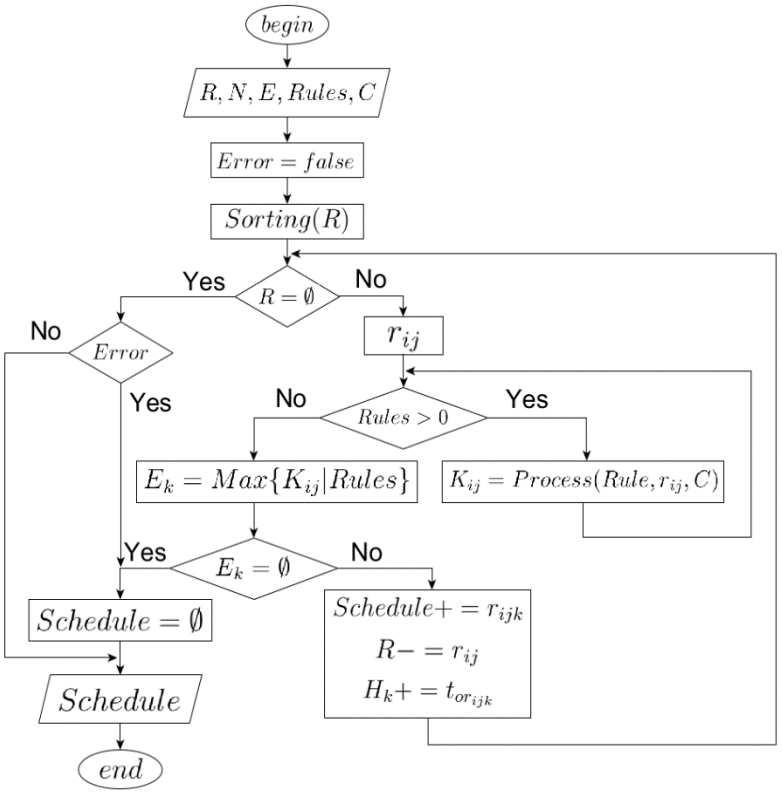

Fig. 2. Algorithm with decision procedures

A large number of rules are explained by the variety of production planning tasks. The inclusion of a rule in the Pull procedure must be justified analytically or empirically. In practice, expert judgment and weighting methods are used for these purposes. [16].

Verification of the possibility of exclusion of the considered vertex E_k is the basis of analytical rules. Consider the following example. Let it be an operation r_{ij} on an iteration of the algorithm. It is necessary to decide on the appointment of this operation on the device from the set E. As in the original statement of the problem, the duration of

processing an operation on machines from the set E and time funds H_k are known Based on these data, it is possible to determine the amount of time required to complete the processing of a nomenclature unit N_i in an iteration l:

$$\sum_{i=l}^{m}\sum_{j=l}^{p_i}\sum_{k=l}^{n} t_{or_{ijk}} \tag{1}$$

According to the condition of the problem, this value can be different when moving from one machine to another.

The criteria for assigning the operation r_{ij} to the device E_k consists of the fulfillment of the condition:

$$t_{sl} + \sum_{i=l}^{m}\sum_{j=l}^{p_i}\sum_{k=l}^{n} t_{or_{ijk}} < H_k \tag{2}$$

where t_{sl} – total processing time of a nomenclature unit N_i at an iteration l.

The more effective is the assignment, the lower the value (1). Most of the computational work is involved in calculating and checking conditions of (2). This problem is reduced to the task of finding a critical path on a graph [17], where the vertices are machines from the set E, and in the arcs are values of (2).

It should be noted that in practice, analytical rules in a "pure form" are rarely used due to the high computational complexity. The balance between the accuracy of the algorithm and the speed of its operation is regulated using the constant C, which limits the depth of the search. The computational complexity of the schedule calculation algorithm with decision-making procedures is influenced by the number of rules in the procedure *Pull*. At the same time, search operations are not typical for heuristic rules. They have constant computational complexity, which is numerically equal to the sorting time of the set R, by the selected criterion. Therefore, algorithms that use heuristic rules in the *Pull* procedure are faster than algorithms based on analytical rules [18].

5 Optimization Schedule Algorithms

Algorithms with optimization as input receive the calculated allowable schedule, and the algorithms considered earlier get a lot of operations [19]. Denote it as $Schedule_0$. The goal of the optimization algorithm is to find the best solution compared to $Schedule_0$, taking into account the selected optimization criteria F. Consider the algorithm for solving this problem in the general form (**Fig. 3**).

Since optimization is carried out according to a given criterion, it is necessary to obtain an acceptable schedule at the initial iteration taking into account F. This schedule is taken as the base. If an error occurs during the next iteration, the algorithm will return the base schedule as a result. To limit the computational complexity of the algorithm, the number of optimization operations is introduced, denoted as n. The larger is the n value, the more accurate the output schedule can be. It is impossible to guarantee a direct proportional dependence of the number of iterations n on the quality of the sched-

ule because of the convergence of heuristic algorithms that are used to solve the optimization problem.

CalculateSchedule procedure solves the optimization problem taking into account the criterion *F*, passed as an argument. As a result, the order of r_{ijk} assignments in the *Schedule* is changed. In addition to solving the optimization problem, it solves the problem of choosing the initial operation (a return point).

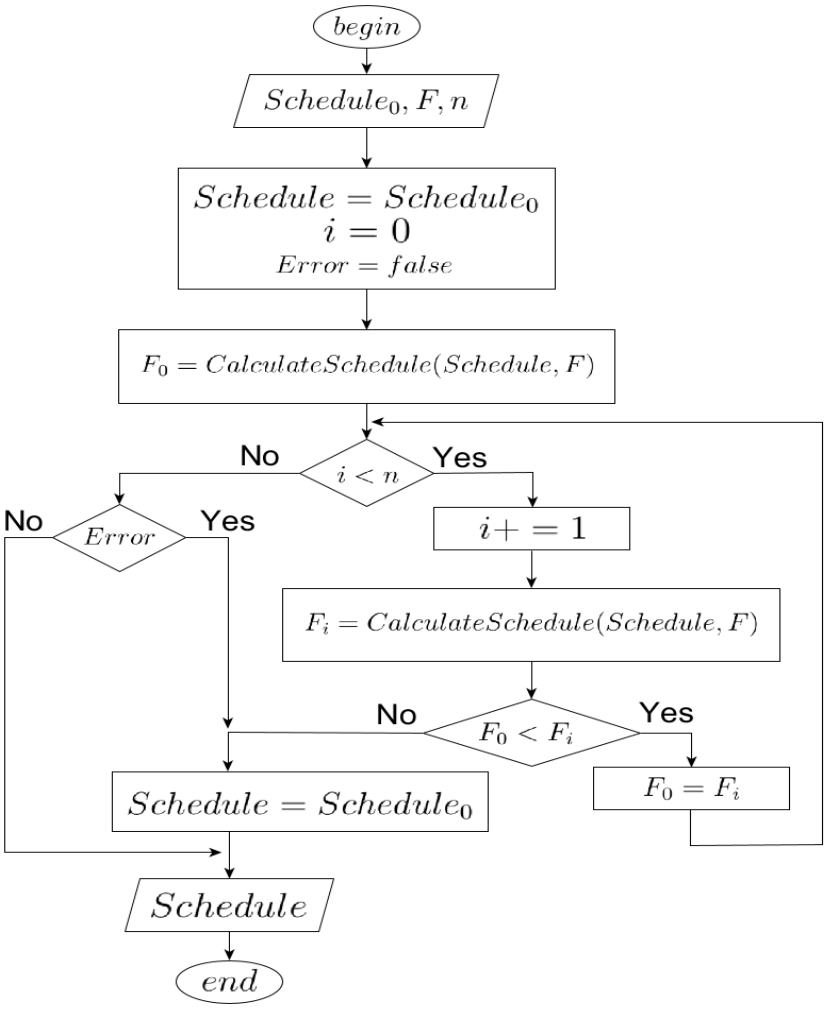

Fig. 3. Scheduling algorithm with optimization

The return point is the vertex on the graph from which the search for the best path in terms of the problem is started. For large *R*, the analytic problem of finding a return point by exhaustive search has complexity *n!* which does not apply to practical problems. In particular, the iterative transition method to the previous vertex (branch and

bound method) and the method of returning to the initial vertex also do not provide an acceptable speed. The following heuristic approaches to determining the return point are proposed in the literature [20]: consider the return point as the part of the valid *Schedule* solution in which only large works are assigned; taking into account that the speed of the algorithm for calculating the schedule and the number of search vertices are known, it should be taken as the return point for which the total time of the algorithm will not exceed the specified value *T*.

At each iteration, the schedule is calculated for the current set of *Schedule* assignments, taking into account criteria *F*. The graphic meaning of the solution is the following. Many assignments are represented as a directed graph with vertices r_{ijk}. Starting at some vertex marked with a return point, the algorithm tries to find a different sequence of assignments. If such a sequence exists, and the value of the criterion F_i is greater than the similar value of *F* for the previous iteration, then the solution is considered improved. This defines the task of finding the critical path.

Table 1. Characteristics of the scheduling algorithms

Algorithm / Criteria	**Algorithm without any decision procedures**	**Algorithm with decision procedures**	**Optimization algorithm**
Sync frequency	low	high	high
Planning horizon	long	long/short	short
The degree of interpretation of the result	low	high	high
The ability to evaluate the result	absent	present	present
The dependence of time complexity on the quality of the solution	absent	direct proportionality	Depending on the conditions of the task
Optimization criterion	absent	single-criteria optimization	single/multi-criteria optimization
Application area	APS	MES/APS	MES/ERP

Algorithms with decision-making procedures, exact algorithms for finding solutions for canonical problems of scheduling theory, “greedy” algorithms can be used to schedule in the *CalculateSchedule* procedure.

Given the previously discussed features of the implementation of scheduling algorithms, it seems possible to determine the number of criteria that characterize planning tasks.

Conclusion

Features of scheduling algorithms used in modern production planning systems were identified in the article. Knowing the features of the functioning of the corresponding

algorithms at different stages of the life cycle of the schedule makes it possible to balanced the use of limited computing power and time resources.

The described methodology for decomposing a complex problem into many simple ones can be applied by developers of production planning systems. As a further direction of research development, it is planned to develop a prototype of an information system for scheduling and synchronization based on an open architecture, which in terms of MES is a function of Operations – Details Scheduling (ODS).

References

1. Anichkin A. S., Semenov V. A. Actual sheduling theory models and methods. Works of ISP RSA. Vol.26. No.3. pp. 5 – 6, 2014.
2. Chernenko A. A. Features of increasing the efficiency of business processes in enterprises of various types by using the capabilities of industrial information systems. New technologies of MSTU. Vol.4. pp. 64 – 66, 2014.
3. Lazarev A. A., Gafarov E. R. Scheduling theory. Tasks and algorithms. M.: MSU of Lomonosov. pp. 40 – 46, 2011.
4. Lazarev A. A. Scheduling theory. Estimates of the absolute error and the scheme for the approximate solution scheduling theory problems. M.: MPTI. pp. 93 – 96, 2008.
5. Shkurba V. V. Calendar planning. Constructive optimization. Automation & Telemechanics. Vol. 10. pp. 122 – 125, 2010.
6. Lavalle Steven M. Planning algorithms. Cambridge University Press. pp. 236 – 237, 2006.
7. Chernenko A. A. Production schedule management in the information system of operative-calendar planning. Bulletin of the Adyghe State University. Ser. Natural-mathematical and technical sciences. Vol.3. No.206. pp. 122–128, 2017.
8. ISA – 95.00.01 – CDV3 Enterprise – Control System Integration, Part 1: Models and Terminology. Research Triangle Park, North Carolina, USA: International Society of Automation, 2008.
9. Williams Theodore J. The Purdue enterprise reference architecture. Computers in industry. Vol.24. No.2. pp. 141 – 158, 1994.
10. Moore J. M. An n job one machine sequencing algorithm for minimizing the number of late jobs. Manag. Sci. Vol.15. No.1. pp. 102 – 109, 1968.
11. Sevastyanov S. V. Geometry methodes and effective algorithms in scheduling theory: dissertation of the doctor of physical and mathematical sciences. Novosibirsk: Sobolev RSA Institute, 2000, pp. 37 – 41.
12. Liqing Li, Hai Lu, Sichuan Mianyang. Integrated Production Planning and Scheduling System Design. IEEE. pp. 732 – 734, 2017.
13. Dasgupta S., Papadimitriu H., Vazirni U. Algorithms. M.: MCIMO. pp. 106 – 108, 2014.
14. Pinedo Michael L. Scheduling: Theory, Algorithms, and Systems. Springer Science & Business Media. pp. 209 – 212, 2008.
15. Smolyar L. I. Discrete production operative scheduling models. M.: Science. pp. 192 – 193, 1978.
16. Velychko O., Gordiyenko T. Methodologies of expert's competence evaluation and group expert evaluation. Article in Metallurgical and Mining Industry. pp. 262 – 264, 2015.
17. Cormen Thomas H. Algorithms unlocked. The MIT Press, Cambridge, Massachusetts. pp. 237 – 237, 2013.
18. Levitin A. A. Algorithms. Introduction into development and analyse. M.: Villiams. pp. 283 – 284, 2006.

19. Zagidullin R. R. Engineering Planning. Stariy Oskol: TNT. Pp. 315 – 319, 2017.
20. Farkas G. A., Martinek P. Production plan scheduling on SMT manufacturing lines. IEEE 23rd International Symposium for Design and Technology in Electronic Packaging (SIITME). pp. 103 – 104, 2017.