ISSN 1064-2307, Journal of Computer and Systems Sciences International, 2024, Vol. 63, No. 1, pp. 132–140. © Pleiades Publishing, Ltd., 2024.

COMPUTER METHODS

Scheduling Calculations for a Multiprocessor System in Real Time

M. G. Furugyan[a,*]

[a] *Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, Moscow, 119333 Russia*

**e-mail: rtsccas@yandex.ru*

Received June 20, 2023; revised August 19, 2023; accepted October 2, 2023

Abstract—The problem of scheduling computations in a multiprocessor system is considered for the case when, at some time instants, requests for the execution of job packages with known characteristics are received. Interrupts and switching from one processor to another are allowed. In the first formulation, the composition of all complexes and the characteristics of tasks are known in advance. In the second setting, this information becomes known only at the time of each request. It is required to determine whether there is an admissible schedule for the total set of jobs and build it in the case of a positive answer. A setting is studied in which, in addition to processors, there is a nonrenewable resource. A polynomial algorithm for solving the problem is developed, based on the construction of a network flow model and the search for the maximum flow.

Keywords: scheduling computations, multiprocessor real-time system, admissible schedule, network model, maximum flow, nonrenewable resource

DOI: 10.1134/S1064230724700102

INTRODUCTION

The main distinguishing feature of real-time computing systems is that each application module must be executed in a strictly specified time interval and completed no later than a predetermined deadline. Such systems are widely used in various fields of human activity. For example, during the design, testing, and operation of complex technical objects (airplanes, rockets, power plants), during development work, in civil and military construction, when assessing mineral reserves in deposits, when processing large amounts of information, during the design and operation of transport and conveyor systems, and in many other areas. In this case, one of the main tasks is to distribute the computer system's resources between software modules and build the optimal schedule for their execution. A large number of publications study algorithms for solving such problems. Here we note fundamental works such as [1–3], in which the authors study various formulations (drawing schedules with interruptions and switching from one processor to another and without interruptions, tasks for performance and meeting deadlines, construction of uniprocessor and multiprocessor schedules). In [3], NP-hard problems of performance and minimization of the maximum time offset for one and several devices are studied. A new approach to find approximate solutions is proposed. In [4, 5], a technique for constructing optimal schedules in problems with nonfixed parameters (durations, consumed resources) is considered. The technique is based on the use of the branch-and-bound method and the construction of polyhedra of the solution's stability. In [6–8], a technique was developed for checking the fulfillment of real-time constraints, which consist in the fact that each job must be performed within the given guideline interval. Research was carried out for a multicore real-time computing system and was based on the construction of a simulation model using generalized finite state machines with a timer stop. Using this model, a timing diagram of the system's operation is constructed, allowing for direct verification of compliance with real-time constraints. In [8], a pseudo-polynomial algorithm for solving the problem was proposed, which constructs a schedule that is optimal in terms of performance for executing tasks with logical precedence conditions. In this problem, each job is given a list of its immediate predecessors and the number of completed immediate predecessors required to begin executing it. The problem is reduced to a cyclic game. In [9, 10], some work scheduling problems are reduced to minimax problems.

The publications mentioned above study the distribution of renewable resources (processors, machines, actuators, devices, workers), i.e., resources that can be used repeatedly. A number of publica-

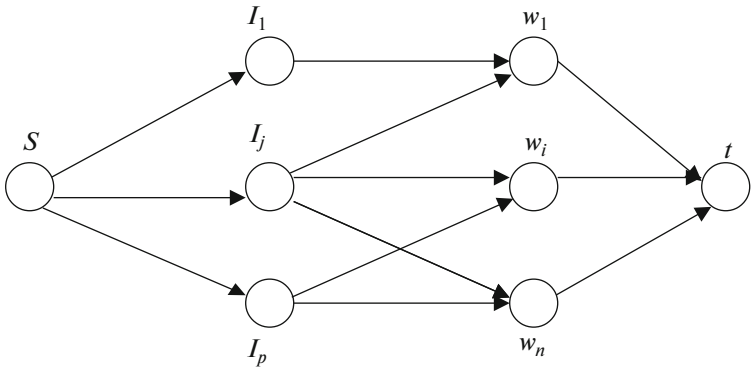

Fig. 1. Streaming network G to find an admissible schedule.

tions examine issues of the distribution of nonrenewable resources (finance, fuel, electricity, various materials, computer RAM assigned to certain software modules). Unlike renewable resources, nonrenewable resources cannot be reused. In relation to this, we note the works [11, 12], in which it is assumed that the duration of tasks linearly depends on the size of the resource allocated to them. In [13], a problem with mixed types of resources—renewable and nonrenewable—was studied. We consider the problem of creating an admissible schedule with interruptions in a multiprocessor system in the case where policy intervals are specified, processors can have arbitrary performance, there are several types of nonrenewable resources, and the duration of work execution linearly depends on the amount of the resources allocated to them. Polynomial algorithms are constructed based on reducing the original problem to a flow problem in a network of a special type.

We note some interesting articles on planning in industrial production. In [14], the authors explored a methodology for the joint planning of production capacity development and scheduling taking into account market opportunities, and also detailed an integrated production capacity planning model with several discrete and continuous variants for changing short-term and medium-term capacity and developed a heuristic algorithm based on the reduction of the initial problems to a nonlinear mixed integer problem. In [15], some issues in the field of production planning and control are presented, and a hierarchical architecture for production planning and control is developed. In [16] a two-tier single-product storage system in which a regional center replenishes orders from several independent local distribution centers over a set period of time is presented. The developed model determines the values of the product's price, required replenishment time, and required delivery time that maximize the expected system-wide profit for the given period, taking into account product storage costs and fixed equipment costs.

In [17] a problem in which scheduling is carried out in two stages is studied: first, the sequence of activities is determined, then idle time is inserted into the schedule to minimize the sum of early and late costs. The sequence of activities is determined using a heuristic method, and the task of inserting idle time is solved using linear programming to set the start and end times of activities. In [18] the task of planning lead times is presented and the trade-offs that should be taken into account when setting these deadlines are also identified. A model is proposed to show how the scheduled execution time of an operation depends on the stochastic variability of the resource requirements for that operation, as well as the resource utilization related to that operation. In [19] the task of scheduling and routing two robots that deliver products to specific locations is described. The problem of minimizing the time required to complete all operations and return the robots to their original position is solved. The problem is proved to be NP-hard. The solution is based on the use of integer linear programming and genetic algorithm, as well as dynamic programming to evaluate the quality of the solutions.

This paper studies the computation scheduling problem on a multiprocessor system under the following assumptions. At the given points in time, requests are received to perform sets of work with known durations and target intervals. Interrupts and switches from one processor to another are allowed. Two formulations of the problem are considered. In each of them, the moments at which requests arrive are known in advance. However, in the first setting, the composition of all complexes and the characteristics of the work are also known in advance, and therefore, in this case, the execution of all tasks can be planned before the first request arrives. In the second formulation, the composition of task sets and their characteristics become known only at the time of receipt of each request. Then it is possible to plan the execution of the work only after receiving the corresponding request, i.e., in real time. In both settings, it is necessary

to determine whether there is an acceptable schedule for the entire set of work packages and construct it in the case of a positive answer. The problem is also considered for the case when, in addition to processors, the jobs use a nonrenewable resource. In this case, the duration of a task is a decreasing function of the amount of the nonrenewable resource allocated to it. Unlike [13], this function is not expected to be linear. The solution to the problem is based on constructing a network flow model and searching for the maximum flow.

1. PROBLEM STATEMENT

At moments in time τ_k requests to complete K packages of work (tasks) are received: $W_k = \{w_k^1, w_k^2, \ldots, w_k^{r_k}\}$, $k = \overline{1,K}$, $\tau_1 < \tau_2 < \ldots < \tau_K$. To do this, in each interval $[\tau_k, \tau_{k+1}]$, m_k processors are available (τ_{K+1} is the point in time after which these processors cannot be used). All the processors are identical. Every task w_k^i has the following characteristics: $[b_k^i, c_k^i]$ is the guideline interval (work w_k^i can only be executed in this interval), $b_k^i \geq \tau_k$, t_k^i is its duration, and $t_k^i \leq c_k^i - b_k^i$, $i = \overline{1,r_k}$. When executing the tasks, interruptions and switching from one processor to another are allowed, which, by assumption, do not require time. In addition, parallel execution of one task by several processors and simultaneous execution of several jobs by one processor are not allowed.

Two formulations of the problem are considered. In each of them, moments τ_k are known in advance. However, in the first formulation, the composition of all complexes W_k and the characteristics of the works included in them are also known in advance. Therefore, in this case, it is possible to schedule the work of all tasks up to the point in time τ_1. In the second setting, the composition of the set of tasks W_k and their characteristics become known only at the moment τ_k. Then, the work W_k can only be planned after receiving the corresponding request, which arrives at the moment τ_k, i.e., in real time.

In both settings, it is necessary to determine whether there is an acceptable schedule for the entire set of works

$$W = \bigcup_{k=1}^{K} W_k$$

(i.e., a schedule in which each task is executed in its own guideline interval) and build it in the case of a positive answer. It is assumed that the operating time of the scheduling algorithm itself can be neglected in both cases.

The solution to the problem is based on constructing a network flow model and searching for the maximum flow. Therefore, the next section describes one of the most effective streaming algorithms, whose modification will be used to construct the schedule.

2. BRIEF DESCRIPTION OF A POLYNOMIAL ALGORITHM FOR FINDING THE MAXIMUM FLOW IN A NETWORK

Given the network $G = (V, A)$, V is the set of peaks, u is the source, v is the drain, $u, v \in V$, A is the set of oriented arcs, and $U(a,b)$ is the throughput of arc $(a,b) \in A$. The following polynomial algorithm for finding the maximum flow in the network G is proposed in [20].

Step 1. Select f as the initial zero thread , i.e., put $f(a,b) = 0$ for all $(a,b) \in A$.

Step 2. Build the residual network $G(f) = (V, A(f))$:

if $f(a,b) < U(a,b)$, then include arc (a,b) with bandwidth $U(a,b) - f(a,b)$ in $A(f)$; and if $f(a,b) > 0$, then include arc (b,a) with bandwidth $f(a,b)$ in $A(f)$.

Step 3. If in the network $G(f)$ there is no direct path from u to v, then f is the maximum flow; and the algorithm is completed. Otherwise, go to step 4.

Step 4. Build a layered network $G^*(f) = (V^*(f), A^*(f))$. It contains all the shortest oriented paths from u to v.

Step 5. Find a dead-end flow g in the network $G^*(f)$. (A dead-end flow is a flow with respect to which there is no direct increasing path.)

Step 5.1. Define node $a_0 \in V^*(f)$ with the minimal throughput. (The capacity of a node is the minimum of the maximum amount of the flow that can enter that node and the maximum amount of the flow that can leave it.)

Step 5.2. "Push" out maximum possible flow from node a_0 to the left up to source u and to the right all the way to drain v. The resulting flows along the arcs determine flow g.

Step 5.3. Remove from network $G^*(f)$ node a_0 and all other nodes with zero residual capacity, arcs incident to these nodes, all fully saturated arcs, the resulting "hanging" nodes (if any), and all arcs incident to them.

Step 5.4. If there is a path from u and v in the rest of the network $G^*(f)$, then go to step 5.1. Otherwise a dead-end flow g is built in the network $G^*(f)$.

Step 6. Correct the flow f in the network G:

if arc $(a,b) \in A$ corresponds to arc $(a,b) \in A^*(f)$, then put $f(a,b) = f(a,b) + g(a,b)$; if arc $(a,b) \in A$ corresponds to arc $(b,a) \in A^*(f)$, then put $f(a,b) = f(a,b) - g(b,a)$. Go to step 2.

The computational complexity of the described algorithm is $O\left(|V|^3\right)$.

3. BRIEF DESCRIPTION OF THE ALGORITHM BY V.S. TANAEV FOR CONSTRUCTING AN ADMISSIBLE SCHEDULE

The further study of the problem is based on the use of the algorithm of V.S. Tanaev for constructing an admissible multiprocessor schedule with interruptions and switchings [1]. Let us give a brief description of this algorithm for the task formulated in Section 1 at $K = 1$. We will assume that in this case the set of jobs $W = \{w_1, w_2, \ldots, w_n\}$, their duration is t_i, guideline intervals are $[b_i, c_i]$, and m is the number of identical processors.

Assume $y_0 < y_1 < \ldots < y_p$ are all different quantities b_i, c_i, $i = \overline{1,n}$, $I_j = [y_{j-1}, y_j]$, $\delta_j = y_j - y_{j-1}$, $j = \overline{1,p}$. A streaming network $G = (V, A)$ (see figure) is being built, where $V = \{u, I_j, w_i, v\}$ is the set of peaks, u is the source, v is the drain, and $A = \{(u, I_j), (I_j, w_i), (w_i, v)\}$ is the set of oriented arcs. Arc (I_j, w_i) is entered in the network if $I_j \subseteq [b_i, c_i]$. Note that for all j and i either $I_j \subseteq [b_i, c_i]$ or $I_j \cap [b_i, c_i] = \varnothing$. Bandwidths U of the arcs are defined as follows: $U(u, I_j) = m\delta_j$, $U(I_j, w_i) = \delta_j$, and $U(w_i, v) = t_i$.

It was proved in [1] that an admissible schedule exists if and only if the maximum flow f of network G saturates all output arcs (w_i, v), i.e., when $f(w_i, v) = t_i$ for all $i = \overline{1,n}$. Magnitude $f(I_j, w_i)$ is equal to the CPU time allocated to job w_i in the interval I_j. To construct an admissible schedule, in each interval I_j, apply the packing algorithm [1], whose computational complexity is $O(n)$. To find the maximum flow in network G, we can use the polynomial algorithm described in Section 2. Since $p \le 2n$, the computational complexity of Tanaev's algorithm in this case is $O(n^3)$.

4. CONSTRUCTION OF A SCHEDULE FOR THE CASE WHEN INFORMATION ABOUT SETS W_k, $k = \overline{1,K}$, ARRIVES UNTIL τ_1

Let us consider the case when the composition of all sets W_k and characteristics of the works included in them are known in advance (i.e., before the moment of time τ_1). In this case, we can create a schedule for the entire set of tasks W even before requests for work W_k are received at moments in time τ_k (i.e., until the point in time τ_1).

Assume y_k^j are all different quantities τ_k, b_k^i, and c_k^i, $k = \overline{1,K}$, $i = \overline{1,r_k}$. Further, as in Section 3, intervals I_k^j and streaming network $G_k = (V_k, A_k)$ are determined, where $V_k = \left\{u, v, I_k^j, w_k^i,\ j = \overline{1,p_k}\right\}$ is the set of

nodes, u is the source, v is the drain, and $A_k = \{(u, I_k^j), (I_k^j, w_k^i), (w_k^i, v)\}$ is the set of oriented arcs. Arc $\left(I_k^j, w_k^i\right)$ is introduced into the network G_k if $I_k^j \subseteq [b_k^i, c_k^i]$.

Further, to solve the question of the existence and construction of an admissible schedule, an algorithm similar to the one described in Section 3 can be used. The computational complexity of the algorithm is

$$O\left(\left(\sum_{k=1}^{K} r_k\right)^3\right).$$

5. CONSTRUCTION OF A SCHEDULE FOR THE CASE WHEN INFORMATION ABOUT THE SET W_k ARRIVES AT THE MOMENT τ_k, $k = \overline{1,K}$

The case is considered when the moments τ_k of the receipt of requests for work W_k are known in advance, and the composition of each set W_k and the characteristics of the tasks included in it become known only at the moment τ_k. First, we consider the problem of constructing a schedule for W_1.

5.1. Constructing a Network Model and Making an Admissible Schedule for W_1

Assume $y_1^0 < y_1^1 < \ldots < y_1^{p_1}$ are all the different quantities among τ_1, τ_2, b_1^i, c_1^i, $i = \overline{1,r_1}$, belonging to the interval $[\tau_1, \tau_2]$. We define the intervals $I_1^j = [y_1^{j-1}, y_1^j]$, $j = \overline{1,p_1}$, and network $G_1 = (V_1, A_1)$, where $V_1 = \left\{u, v, I_1^j, w_1^i\right\}$, and the set of oriented arcs A_1 and their throughputs $U_1(i,j)$ are determined analogously to how it was done in Section 3 at $m = m_1$, $n = r_1$.

When finding the maximum flow f_1 in the network G_1, the following modification of the algorithm described in Section 2 is used. When pushing the flow from the peak I_1^j to the right (step 5.2, Section 2), the arcs (I_1^j, w_1^i) should be used first for works w_1^i for which $c_1^i \le \tau_2$. This is because such jobs can only be executed within the interval $[\tau_1, \tau_2]$, and for works with a deadline longer than τ_2, the CPU time can be allocated even after the moment τ_2.

The quantity of the flow $f_1(I_1^j, w_1^i)$ is equal to the amount of the CPU time allocated to job w_1^i in interval I_1^j. The schedule in interval I_1^j is built using a packing algorithm.

If for at least one job w_1^i with deadline $c_1^i \le \tau_2$, it turned out that $f_1\left(w_1^i, v\right) < t_1^i$ (i.e., arc $\left(w_1^i, v\right)$ is not completely saturated), then the admissible schedule for W_1, and, therefore, for the entire set of tasks W does not exist. In the case $f_1\left(w_1^i, v\right) = t_1^i$ for all works w_1^i with deadline $c_1^i \le \tau_2$, the construction of an admissible schedule will continue at moment τ_2 for unfinished works from W_1 (if any) and newly received works from W_2.

5.2. Constructing a Network Model and Making Admissible Schedules for $W_1 \cup \ldots \cup W_k, k = \overline{2,K}$

We assume that we have built a streaming network $G_{k-1}, k = \overline{2,K}$, and the maximum flow f_{k-1} is found in it. If for at least one job $w_r^i, r = \overline{2,k-1}$, for which $c_r^i \le \tau_k$, after finding the maximum flow f_{k-1} in the network G_{k-1}, it turned out that $f_{k-1}\left(w_r^i, v\right) < t_r^i$, i.e., arc $\left(w_r^i, v\right)$ is not completely saturated, then the admissible schedule for $W_1 \cup \ldots \cup W_{k-1}$, and, therefore, for the entire complex W does not exist. If $f_{k-1}\left(w_r^i, v\right) = t_r^i$ for all arcs $\left(w_r^i, v\right)$, $r = \overline{1,k-1}$, for which $c_r^i \le \tau_{k-1}$, then constructing an admissible sched-

ule for W continues at the moment τ_k for unfinished works from the set $W_1 \cup \ldots \cup W_{k-1}$ (if any) and newly received tasks from W_k.

Assume $y_k^0 < y_k^1 < \ldots < y_k^{p_k}$ are all different quantities τ_k, τ_{k+1}, b_k^i, c_k^i, $i = \overline{1,r_k}$. We define the intervals $I_k^j = \left[y_k^{j-1}, y_k^j\right]$ and assume $\delta_k^j = y_k^j - y_k^{j-1}$. From the network G_{k-1}, we remove the following elements:

—nodes I_{k-1}^j, $j = \overline{1, p_{k-1}}$, and all the arcs incident to them $\left(u, I_{k-1}^j\right)$, $\left(I_{k-1}^j, w_{k-1}^i\right)$, $j = \overline{1, p_k}$, $i = \overline{1, r_{k-1}}$;

—nodes w_z^i for which

$$f_{k-1}\left(w_z^i, v\right) = t_z^i \tag{5.1}$$

and the corresponding arcs $\left(w_z^i, v\right)$, $z = \overline{1, k-1}$, $i = \overline{1, r_z}$ (since the fulfillment of condition (5.1) means the completion of work w_z^i).

Next, the duration of each remaining job w_z^i decreases by the amount of the flow $f_{k-1}\left(w_z^i, v\right)$ along the arc $\left(w_z^i, v\right)$. Network G_k is built from the remaining nonremote part of the network G_{k-1} by adding to it nodes I_k^j, w_k^i and arcs $\left(I_k^j, w_z^i\right)$, $j = \overline{1, p_k}$, $i = \overline{1, r_z}$, $z = \overline{1, k}$, $\left(w_k^i, v\right)$, $i = \overline{1, r_k}$, analogously to how it is described in Section 3 at $m = m_k$, $n = n_k +$ (the number of remaining vertices G_{k-1}). The maximum flow f_k is found in the network G_k. By analogy with Section 5.1 when pushing the flow from the vertex I_k^j to the right (step 5.2, Section 2) arcs $\left(I_k^j, w_k^i\right)$ should be used first for works w_k^i for which $c_k^j \le \tau_{k+1}$. The quantity of the flow $f_k\left(I_k^j, w_k^i\right)$ is equal to the amount of CPU time allocated to the job w_k^i in the interval I_k^j. Schedule of work execution in the interval I_k^j is found using the packing algorithm [1].

The computational complexity of the proposed algorithm is

$$O\left(K\left(\sum_{k=1}^{K} r_k\right)^3\right).$$

6. DISTRIBUTION OF A HETEROGENEOUS SET OF RESOURCES

It is assumed that when performing work, in addition to processors, a nonrenewable resource is used, the amount of which in the interval $[\tau_k, \tau_{k+1}]$ amounts to S_k, $k = \overline{1, K}$. The work w_k^i of a nonrenewable resource can only be allocated at the point in time τ_k. If its volume is s_k^i, then the duration of the task w_k^i is

$$t_k^i = \varphi_k^i\left(s_k^i\right), \tag{6.1}$$

where

$$s_k^i \in \left[0, \overline{s}_k^i\right], \tag{6.2}$$

$$\sum_{i=1}^{r_k} s_k^i \le S_k. \tag{6.3}$$

Here, φ_k^i is a strictly decreasing function in the interval $\left[0, \overline{s}_k^i\right]$ that takes positive values and $\overline{s}_k^i$ are the specified values, $i = \overline{1, r_k}$.

Under these assumptions, to solve the problem posed in Section 1, we will need a generalization of Tanaev's algorithm.

6.1. Generalization of Tanaev's Algorithm in the Case a Nonrenewable Resource is Available

We will use the notation introduced in Section 3. In addition, the characters t_k^i, s_k^i, $\overline{s}_k^i$, S_k, φ_k^i, W_k, and w_k^i are replace by t_i, s_i, $\overline{s}_i$, S, φ_i, W, and w_i, respectively. In this case, restrictions (6.1)–(6.3) are written as follows:

$$t_i = \varphi_i(s_i), \tag{6.4}$$

$$s_i \in \left[0, \overline{s}_i\right], \tag{6.5}$$

$$\sum_{i=1}^{n} s_i \le S. \tag{6.6}$$

Let us prove the following statements.

Lemma 1. *An acceptable schedule for executing a set of jobs W in a problem without a nonrenewable resource (without restrictions* (6.4)–(6.6)*), in which the task w_i is given CPU time amounting to f_i, exists if and only if in the network G there is a flow $f(a,b)$, $(a,b) \in A$ such that the equalities*

$$f(w_i, v) = f_i \tag{6.7}$$

are satisfied in front of all $i = \overline{1,n}$.

The proof follows from [1]. Assume $\varphi_i^{-1}(s_i)$ is a function inverse to $\varphi_i(s_i)$.

Lemma 2. *An acceptable schedule for executing a set of jobs W in a problem with a nonrenewable resource (taking into account restrictions* (6.4)–(6.6)*), in which the task w_i is given CPU time amounting to f_i, exists if and only if in the network G there is a flow $f(a,b)$, $(a,b) \in A$ such that equalities* (6.7) *and the following inequalities are valid:*

$$\sum_{i=1}^{n} \varphi_i^{-1}(f_i) \le S, \tag{6.8}$$

$$\varphi_i^{-1}(f_i) \le \overline{s}_i, \quad i = \overline{1,n}. \tag{6.9}$$

Proof. 1. Assume that in the network G there is a flow f for which relations (6.7)–(6.9) are valid. In this case, from (6.8) it follows that each work w_i can be allocated a nonrenewable resource in the amount of $s_i = \varphi_i^{-1}(f_i)$. In this case, inequality (6.6) will be true, and due to (6.4), the duration of work w_i (i.e., required CPU time) will be $\varphi_i(\varphi_i^{-1}(f_i)) = f_i$. Since $\varphi_i^{-1}(f_i) = s_i$, (6.5) follows from (6.9). Then Lemma 1 implies the existence of an admissible schedule for W taking into account restrictions (6.4)–(6.6).

2. Now assume there is an admissible work schedule W in a problem with a nonrenewable resource (taking into account restrictions (6.4)–(6.6)), in which the task w_i is allocated CPU time in the amount of f_i. Assume in this case work w_i is allocated a nonrenewable resource in the amount s_i. Then, by (6.4) $f_i = \varphi_i(s_i)$ or $s_i = \varphi_i^{-1}(f_i)$. In this case, inequalities (6.8) and (6.9) follow from (6.6) and (6.5), respectively. The proof now follows from Lemma 1. The lemma is proved.

From (6.7), (6.9), it follows that when constructing an admissible schedule in a problem with a nonrenewable resource, we must look for the flow f in the network G for which $f(w_i, v) = f_i \le \varphi(\overline{s}_i)$ in front of all $i = \overline{1,n}$. Considering that the functions $\varphi_i^{-1}(f_i)$ are strictly decreasing, quantities f_i need to be maximized. Therefore, the following algorithm is proposed to solve this problem.

In the network G, first the vertex w_{i_1} is determined, for which the maximum value of flow g from u to w_i is the largest among all vertices w_i, $i = \overline{1,n}$. Next, number i_1 is included in the set N and the throughput capacities of all arcs $(a,b) \in A$ of network G decrease by the quantity $g(a,b)$. This procedure is then repeated for vertices w_i, $i = \overline{1,n}$, $i \notin N$. Next, condition (6.8) is checked.

Algorithm 1.

Step 1. In the network G put $U(w_i, v) = \varphi_i\left(\overline{s}_i\right)$, $N = \varnothing$.

Step 2. For $i_0 = \overline{1,\ n},\ \ i_0 \notin N$, follow steps 3–5.

Step 3. Remove from network G all arcs (w_i, v), $i = \overline{1,n},\ \ i \neq i_0$.

Step 4. Find the maximum flow g in the network G and assume g_{i_0} is its magnitude.

Step 5. Connect to network G the arcs removed in step 3.

Step 6. Let $\underline{\min}_{i_0=1,n,\ i_0 \notin N} \varphi_{i_0}^{-1}(g_{i_0}) = g_{i_1}$. Include i_1 in N. Put $f_{i_1} = g(w_{i_1}, v) = g_{i1}$. Reduce the throughput capacities $U(a,b)$ of all arcs $(a,b) \in A$ of network G by quantity $g(a,b)$.

Step 7. If $|N| < n$, then go to step 2. If $|N| = n$, then go to step 8.

Step 8. If inequality (6.8) is satisfied, then an admissible schedule exists. In this case, work w_i is allocated a nonrenewable resource amounting to $\varphi_i^{-1}(f_i)$, $i = \overline{1,n}$. The duration of work w_i amounts to f_i. The schedule is constructed as described in Section 3. If inequality (6.8) is not satisfied, then an admissible schedule does not exist. The algorithm is completed.

The computational complexity of Algorithm 1 is $O(n^5)$.

6.2. Generalization of the Original Problem in the Case of the Presence of a Nonrenewable Resource

Let us now turn to the problem formulated in Section 1, with additional restrictions (6.1)–(6.3) related to the distribution of the nonrenewable resource. We return to the notations t_k^i, s_k^i, $\overline{s_k^i}$, S_k, φ_k^i, W_k, and w_k^i, which were previously replaced by t_i, s_i, $\overline{s_i}$, S, φ_i, W, and w_i, respectively.

Taking into account the research carried out in Sections 5.1, 5.2, and 6.1, the following algorithm is proposed for solving the original problem formulated in Section 1, for the case of the presence of a nonrenewable resource.

Algorithm 2.

Step 1. Put $k = 1$.

Step 2. Build network G_k (see Sections 5.1, 5.2).

Step 3. From network G_k remove nodes w_m^i, corresponding to works with deadline $c_m^i > \tau_{k+1}$, and the arcs incident to them.

Step 4. Apply Algorithm 1 to the resulting network. If at step 8 of Algorithm 1 it turns out that an admissible schedule does not exist, then Algorithm 2 is completed and there is no solution. Otherwise, go to step 5.

Step 5. Connect to network G_k the nodes and arcs removed in step 3. Put $k = k + 1$. If $k \leq K$, then go to step 2. If $k > K$, then the solution is constructed. The algorithm is completed.

The computational complexity of Algorithm 2 is

$$O\left(K\left(\sum_{k=1}^{K} r_k\right)^5\right).$$

CONCLUSIONS

The problem of creating a multiprocessor admissible schedule for a set of work complexes, for which execution requests arrive at the given points in time, has been studied. The composition of each complex and the characteristics of the works included in it become known at the time the request is received. When executing the tasks, interruptions and switching from one processor to another are allowed. Setups with and without a nonrenewable resource were studied. A polynomial algorithm for solving the problem, based on constructing a network flow model and searching for the maximum flow, has been developed.

FUNDING

This work was supported by ongoing institutional funding. No additional grants to carry out or direct this particular research were obtained.

CONFLICT OF INTEREST

The author of this work declares that he has no conflicts of interest.

REFERENCES

1. V. S. Tanaev, V. S. Gordon, and Ya. M. Shafranskii, *Scheduling Theory. Single-Stage Systems* (Nauka, Moscow, 1984) [in Russian].
2. P. Brucker, *Scheduling Algorithms* (Springer, Heidelberg, 2007).
3. A. A. Lazarev, *Scheduling Theory. Absolute Error Estimation and Scheme for Approximate Solution of Scheduling Theory Problems* (MFTI, Moscow, 2008) [in Russian].
4. M. A. Gorskii, A. V. Mishchenko, L. G. Nesterovich, and M. A. Khalikov, "Some modifications of integer optimization problems with uncertainty and risk," J. Comput. Syst. Sci. Int. **61** (5), 813–823 (2022).
5. P. S. Koshelev and A. V. Mishchenko, "Optimizing management of jobs in a logistic project under conditions of uncertainty," J. Comput. Syst. Sci. Int. **60** (4), 595–609 (2021).
6. A. B. Glonina and V. V. Balashov, "On the correctness of the simulation of modular real-time computing systems using networks of timed automata," Model. Anal. Inf. Sist. **25** (2), 174–192 (2018).
7. A. B. Glonina, "A generalized model of modular real-time computing systems for checking the admissibility of configurations of these systems," Vestn. Yuzho-Ural. Gos. Univ., Ser.: Vychisl. Mat. Inf. **6** (4), 43–59 (2017).
8. A. B. Glonina, "Tool system for testing real-time constraints for modular computational system configurations," Moscow Univ. Comput. Math. Cybern. **44** (3), 120–132 (2020).
9. D. V. Alifanov, V. N. Lebedev, and V. I. Tsurkov, "Optimization of schedules with precedence logical conditions," J. Comput. Syst. Sci. Int. **48** (6), 932–936 (2009).
10. A. A. Mironov and V. I. Tsurkov, "Minimax in transportation models with integral constraints: I," J. Comput. Syst. Sci. Int. **42** (4), 562–574 (2003).
11. A. A. Mironov and V. I. Tsurkov, "Minimax under nonlinear transport constraints," Dokl. Akad. Nauk **381** (3), 305–308 (2001).
12. D. Phillips and A. Garcia-Diaz, *Fundamentals of Network Analysis* (Prentice Hall, Englewood Cliffs, N.J., 1981; Mir, Moscow, 1984).
13. E. G. Davydov, *Operations Research* (Vysshaya shkola, Moscow, 1990) [in Russian].
14. M. G. Furugyan, "Computation scheduling in multiprocessor systems with several types of additional resources and arbitrary processors," Moscow Univ. Comput. Math. Cybern. **41** (3), 145–151 (2017).
15. X. Yao, N. Almatooq, R. G. Askin, and G. Gruber, "Capacity planning and production scheduling integration: Improving operational efficiency via detailed modelling," Int. J. Prod. Res. **60** (1), 7239–7261 (2022).
16. H. Missbauer and R. Uzsoy, "Order release in production planning and control systems: Challenges and opportunities," Int. J. Prod. Res. **60** (1), 256–276 (2022).
17. Y. Wang, J. Geunes, and X. Nie, "Optimising inventory placement in a two-echelon distribution system with fulfillment-time-dependent demand," Int. J. Prod. Res. **60** (1)), 48–72 (2022).
18. M. F. Gorman and D. G. Conway, "A tutorial of integrating duality and branch and bound in earliness–tardiness scheduling with idle insertion time problems," Int. J. Prod. Res. **56** (1–2) (2018).
19. S. C. Graves, "How to think about planned lead times," Int. J. Prod. Res. **60** (1) (2022).
20. O. Thomasson, M. Battarra, G. Erdogan, and G. Laporte, "Scheduling twin robots in a palletising problem," Int. J. Prod. Res. **56** (1–2) (2018).
21. T. Cormen, Ch. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms* (MIT Press, Cambridge, 2001; Vil'yams, Moscow, 2005).