INFORMATICA, 2001, Vol. 12, No. 1, 61–88
© 2001 *Institute of Mathematics and Informatics, Vilnius*

Sequencing with Ordered Criteria, Precedence and Group Technology Constraints

Adam JANIAK
Institute of Engineering Cybernetics, Wroclaw Technical University
Janiszewskiego 11/17, b. C3, 50–372 Wroclaw, Poland
e-mail: janiak@ict.pwr.wroc.pl

Yakov SHAFRANSKY
Alexander TUZIKOV
Institute of Engineering Cybernetics, National Academy of Sciences of Belarus
Surganov 6, 220012 Minsk, Republic of Belarus
e-mail: shafr@newman.bas-net.by

Received: November 2000

Abstract. Multicriteria sequencing problems with criteria ordered according to their importance are considered. Additional precedence and group technology constraints are imposed. We introduce a notion of a priority-generating vector function and suggest general techniques that form a base for the construction of polynomial time algorithms for numerous sequencing problems including all known polynomially solvable problems. A comprehensive survey of results for sequencing problems with ordered criteria is given as well.

Key words: sequencing, scheduling theory, multiple criteria, precedence constraints, group technology, optimization.

1. Introduction

The paper considers some problems where optimal permutation with respect to several ordered criteria is to be found.

There are number of papers devoted to multi-criteria approach in sequencing and scheduling (see [5]). In such a situation we usually try to find a Pareto-optimum set of solutions and later a decision maker should choose from this set the best solution from his point of view. This choice may be done in different ways (see [25, 27]).

Sometimes a decision maker can order importance of criteria from his point of view in advance. In this case he can number the criteria in such a way that the first criterion is the most important, the second one is the most important among the others criteria and so on. For this situation a feasible solution will be optimal if it gives an optimum to the first criterion and among all optimal solutions with respect to the first criterion it gives also an optimal value to the second criterion and so on. It is easy to notice that the optimal solution for this case is one of the Pareto-optimum solutions.

The above approach is called an ordered criteria approach (see [21]) or lexicographic approach.

In this paper we consider sequencing problems with ordered criteria. In such problems the set of feasible solutions is a set of permutations of elements (e.g., jobs, tasks) of some finite set $N = \{1, 2, \ldots, n\}$. Moreover we consider a situation that usually appears in production systems with group technology [8]. In this case a partition of set N is given: $N = N_1 \cup N_2 \cup \ldots \cup N_q$. Each subset of this partition is partially ordered and a precedence relation is given over set $N_0 = \{N_1, N_2, \ldots, N_q\}$. A permutation π is feasible if all elements belonging to the same subset are situated in π contiguously and a disposition of elements in π doesn't contradict to the mentioned above partial orders (precedence relations). The aim is to find a feasible permutation that is optimal with respect to the ordered criteria approach.

It should be mentioned that sequencing problems are considered in different branches of operations research. Particularly, a number of scheduling problems (single-machine, permutation flow-shop problems) are naturally formulated in terms of optimizing functions over a set of permutations. In such problems a schedule is uniquely represented by a permutation.

Let us consider a practical situation where such an approach to a problem formulation seems very natural.

A system consisting of q blocks, where each block contains several components, is inspected by sequentially applying tests to each component. The process of testing lasts until the first "defective" block is found or all components successfully pass their tests. There are restrictions on a possible sequence of component testing. First of all, the components of each block should be tested contiguously. Besides that there is given a precedence relation on the set of blocks and a precedence relation on the set of components of each block. These relations define possible sequences of testing. The existence of mentioned relations is caused by a mutual dependence of components inside their block as well as blocks inside the system.

For each component i of the system we have a cost w_i of its testing, a probability ρ_i of successful passing its test and a time t_i needed for testing. The tests are assumed to be statistically independent. So, for any feasible sequence $\pi = (i_1, i_2, \ldots, i_n)$ of components, value $Q(i_r) = \rho_{i_1}\rho_{i_2}\ldots\rho_{i_{r-1}}$ is the probability of the event that the component i_r will be tested (here $Q(i_1) = 1$). If components are tested according to sequence π then an expected cost of testing the component i_r is $w_{i_r}Q(i_r)$ and a total expected cost of the system testing is $F_1(\pi) = \sum_{k=1}^{n} w_{i_k}Q(i_k)$. Besides that we have an average completion time of component testing: $F_2(\pi) = \frac{1}{n}\sum_{k=1}^{n}\sum_{l=1}^{k} t_{i_l}$, where $\sum_{l=1}^{k} t_{i_l}$ is the completion time of component i_k testing.

If we know, for example, that minimizing total expected cost is the most important for us and besides that we would like to minimize the average completion time of component testing then we have a typical problem with two ordered criteria, group technology and precedence constraints.

It should be noted that a single-criterion case of the above problem without group technology and precedence constraints, where $F(\pi) = F_1(\pi)$, was discussed in [15] and [14]. In [14] an O$(nlogn)$ algorithm was proposed.

In the paper we give a review of results on ordered criteria approach in scheduling and describe new results in this field concerning to the development of polynomial time algorithms for sequencing problems with ordered criteria. Our main approach is to adapt techniques developed for single-criterion problems. Besides that we extend these techniques for solving problems with group technology constraints.

The rest of the paper is organized as follows. In Section 2 we give the main problem formulation and a survey of previous results. Section 3 is devoted to the consideration of so-called priority-generating vector functions. We introduce a notion of a priority-generating vector function as a generalization of the corresponding notion for scalar functions and give examples of such functions. Here we give also a sufficient condition for a vector function to be priority-generating. In Section 4 polynomial time algorithms for minimizing priority-generating functions on sets of permutations that are feasible with respect to group technology and precedence constraints are given. In Section 5 we consider a special case of the main problem, without group technology or precedence constraints. In Section 6 a rather restricted situation is described where most important criteria are maximal costs and less important ones are so-called non-adjacent priority-generating functions. Some concluding remarks are given in Section 7.

2. Problem Formulation

In this section we formulate a multi-criteria sequencing problem where criteria are ordered in decreasing of their importance and a set of feasible sequences (permutations) is determined by group technology constraints and precedence relations. We give also a survey of earlier obtained results for sequencing problems with ordered criteria.

Let us introduce the relation $\leqslant$ defined on the set of vectors from $\mathbb{R}^m$. For vectors $x = (x_1, x_2, \dots, x_m)$ and $y = (y_1, y_2, \dots, y_m)$ from $\mathbb{R}^m$ we shall write

$x = y$ if and only if $x_i = y_i$, $i = 1, 2, \dots, m$,

$x < y$ if and only if there exists an index $k, 1 \leqslant k \leqslant m$, such that $x_k < y_k$ and $x_l = y_l$ for all $l < k$,

$x \leqslant y$ if and only if $x < y$ or $x = y$.

This relation is usually called a lexicographic order. Further we compare vectors only with respect to the lexicographic order.

Let P_n be the set of all permutations of elements of a finite set $N = \{1, 2, \dots, n\}$. Set N is partitioned on q subsets (groups): $N = N_1 \cup N_2 \cup \dots \cup N_q$, $N_l \cap N_k = \emptyset$ for $l \neq k$. There is a precedence relation $\xrightarrow{k}$ defined for each subset N_k and represented by an acyclic directed graph $G_k = (N_k, U_k)$. There is also a precedence relation $\xrightarrow{0}$ defined over set $N_0 = \{N_1, N_2, \dots, N_q\}$ and represented by an acyclic directed graph $G_0 = (N_0, U_0)$.

Permutation $\pi = (i_1, i_2, \dots, i_n)$, $i_\nu \in N$, $\nu = 1, 2, \dots, n$, is called feasible if the following conditions hold:

(a) for any $k \in \{1, 2, \dots, q\}$ conditions $i_\nu, i_\mu \in N_k$ and $\nu < \mu$ imply $i_j \in N_k$ for every $\nu < j < \mu$ (i.e., elements from the same group should be situated contiguously in every feasible permutation);

(b) for any $k \in \{1, 2, \dots, q\}$ conditions $i_\nu, i_\mu \in N_k$ and $i_\nu \xrightarrow{k} i_\mu$ imply $\nu < \mu$ (i.e., a mutual disposition of elements of any group k should not contradict to the relation $\xrightarrow{k}$);

(c) for any $k, l \in \{1, 2, \dots, q\}$, $k \neq l$, conditions $i_\nu \in N_k$, $i_\mu \in N_l$ and $N_k \xrightarrow{0} N_l$ imply $\nu < \mu$ (i.e., a mutual disposition of elements from different groups should not contradict to the relation $\xrightarrow{0}$).

Denote a set of all feasible permutations by $P_n(G_0, G_1, \dots, G_q)$.

Let functions $F_1(\pi), F_2(\pi), \dots, F_m(\pi)$ be defined on such a set P' that $P_n(G_0, G_1, \dots, G_q) \subseteq P'$. Consider the following sequencing problem.

Problem 1. *For a vector function $\Phi(\pi) = (F_1(\pi), F_2(\pi), \dots, F_m(\pi))$ defined over a set P' it is necessary to find such a permutation π^* that*

$$\pi^* \in P_n(G_0, G_1, \dots, G_q)$$

and

$$\Phi(\pi^*) = \min\{\Phi(\pi) \colon \pi \in P_n(G_0, G_1, \dots, G_q)\}.$$

The permutation π^* is called optimal.

Since we search the minimum of the function $\Phi(\pi)$ in the lexicographic sense then Problem 1 is usually called a lexicographic sequencing problem.

Problem 1 is in fact a multi-criteria problem with criteria ordered in decreasing of their importance. The function $F_1(\pi)$ is the most important and the permutation π^* minimizes $F_1(\pi)$ on the initial set $P_n(G_0, G_1, \dots, G_q)$. The function $F_2(\pi)$ is the most important among the functions $F_2(\pi), \dots, F_m(\pi)$, i.e., π^* minimizes it on the subset of $P_n(G_0, G_1, \dots, G_q)$, which consists of all optimal with respect to $F_1(\pi)$ permutations, and so on.

Some special cases of the formulated general problem were considered in scheduling theory earlier. Now we give a survey of them.

We use common for scheduling theory notions. We call elements of N by jobs. Each job has a processing time t_i, a weight w_i or a non-decreasing penalty function φ_i or ψ_i and a due date d_i. For a given schedule that is represented by a permutation $\pi = (i_1, i_2, \dots, i_n)$ we can compute for each job $i \in N$ the following parameters:

– the completion time $C_i(\pi)$: if $i = i_k$ then $C_i(\pi) = C_{i_k} = \sum_{l=1}^{k} t_{i_l}$,

– the lateness $L_i(\pi) = C_i(\pi) - d_i$,

– the tardiness $T_i(\pi) = \max\{0, C_i(\pi) - d_i\}$,

– the unit penalty $U_i(\pi) = 0$ if $C_i(\pi) \leqslant d_i$ and $U_i(\pi) = 1$ if $C_i(\pi) > d_i$.

We use usual three-field notations $\alpha \mid \beta \mid \gamma$ for scheduling problems (see, e.g., [12]). Single-machine problems are considered in the paper. Therefore the first field α specifying the machine environment equals “1”.

The second field describes grouping constraints and precedence relations, which are defined either over set N or over its subsets. If this field is empty it means that there are no precedence relations or grouping constraints. If $\beta = G$ then there are no grouping constraints but there is a precedence relation over set N that is determined by an acyclic directed graph G. Here $G = C$ denotes that graph G is a chain. Notation $G_0, G_1, \dots, G_q$ in the second field means that we have a general problem (with q groups and precedence relations) over set $P_n(G_0, G_1, \dots, G_q)$.

The third field is used for the description of objective functions (in the decreasing order of their importance). As it was mentioned, we consider problems where all objective functions should be minimized and therefore the notation $(F_1, F_2, \dots, F_m)$ in the field γ means that a vector function

$$(F_1(\pi), F_2(\pi), \dots, F_m(\pi))$$

is to be minimized. We usually omit arguments of functions in this field. Here we consider the following types of objective functions, which are most commonly investigated in scheduling:

$L_{\max} = \max\{L_i : i \in N\}$,
$T_{\max} = \max\{T_i : i \in N\}$,
$\max \varphi_i(C_i) = \max\{\varphi_i(C_i) : i \in N\}$,
$\sum C_i = \sum_{i \in N} C_i$,
$\sum w_i C_i = \sum_{i \in N} w_i C_i$,
$\sum T_i = \sum_{i \in N} T_i$,
$\sum w_i T_i = \sum_{i \in N} w_i T_i$,
$\sum U_i = \sum_{i \in N} U_i$.

One of the first results for a problem with ordered criteria was an $\mathrm{O}(n^2)$ algorithm by Smith [26] for the problem

$1 \mid\mid (C_i \leqslant d_i, \sum C_i)$.

Below we give a list of polynomially solvable problems.

$1 \mid\mid (T_{\max}, \sum C_i) - \mathrm{O}(n^2)$ – [9],
$1 \mid\mid (\max \varphi_i(C_i), \sum C_i) - \mathrm{O}(n^2)$ – [6],
$1 \mid\mid (\max \varphi_i(C_i), \max \psi_i(C_i)) - \mathrm{O}(n^2)$ – [33],
$1 \mid G\colon i \to j \Rightarrow t_i \leqslant t_j \mid (\max \varphi_i(C_i), \sum \psi(C_i)) - \mathrm{O}(n^2)$ – [33], where ψ is a common for all jobs non-decreasing penalty function,
$1 \mid\mid (\sum w_i C_i, T_{\max}) - \mathrm{O}(nlogn)$ – [4],
$1 \mid\mid (\sum w_i C_i, \sum U_i) - \mathrm{O}(nlogn)$ – [4].

NP-hardiness of problem

$1 \mid\mid (C_i \leqslant d_i, \sum w_i C_i)$ is established by Brucker, Lenstra and Rinnooy Kan in [2] and its NP-hardiness in the strong sense – by the same authors in [13]. The obvious extension of this result is NP-hardiness in the strong sense of problems

$1 \mid\mid (\gamma_1, \sum w_i C_i)$ where $\gamma_1 \in \{T_{\max}, \sum U_i, \sum T_i\}$.

Chen and Bulfin [4] established NP-hardiness of the following problems:

$1 \mid\mid (\sum w_i C_i, \gamma_2)$ where $\gamma_2 \in \{\sum w_i U_i, \sum T_i\}$,
$1 \mid\mid (T_{\max}, \gamma_2)$ where $\gamma_2 \in \{\sum w_i U_i, \sum T_i\}$,
$1 \mid\mid (\sum U_i, \gamma_2)$ where $\gamma_2 \in \{\sum w_i T_i, \sum C_i\}$.

They established also NP-hardiness in the strong sense of problems

$1 \mid\mid (\gamma_1, \sum w_i T_i)$ where $\gamma_1 \in \{\sum w_i C_i, T_{max}\}$.

It should be noted that two-criteria problems are evidently NP-hard (NP-hard in the strong sense) if the corresponding single-criterion problem with the first criterion is NP-hard (NP-hard in the strong sense). Problem $1 \mid\mid (\sum w_i T_i, \gamma_2)$ gives an example of such a situation. It is interesting that the replacement of the "first criterion" by the "second criterion" in the above proposition is not valid. Yuan and Zhao [32] showed, for example, that problem $1 \mid\mid (\sum w_i C_i, \sum w_i T_i)$ is pseudo-polynomially solvable though problem $1 \mid\mid \sum w_i T_i$ is strongly NP-hard. They established also pseudo-polynomial solvability of problem $1 \mid\mid (\sum w_i C_i, \sum w_i U_i)$.

There are some results on construction of branch and bound schemes and algorithms for finding a local optimum. Bansal [1] and Shantikumar [24] proposed branch and bound algorithms for problems $1 \mid\mid (C_i \leqslant d_i, \sum w_i C_i)$ and $1 \mid\mid (\sum U_i, T_{\max})$, respectively. For problem $1 \mid\mid (C_i - d_i \leqslant C, \sum w_i C_i)$, where $C \geqslant 0$ is a given number, an algorithm by Burns [3] gives a local optimum. Here permutation π is said to be a local optimum if it cannot be "improved" by a transposition of any two elements (not necessary neighboring). For problem $1 \mid\mid (T_{\max}, \sum w_i C_i)$, which is a special case of the previous one, Miyazaki [16] proposed an algorithm for finding a local optimum as well. He obtained also sufficient conditions under which his algorithm gives a global optimum.

In our paper we develop polynomial time algorithms for several sequencing problems with ordered criteria where the set of feasible permutations is $P_n(G_0, G_1, \ldots, G_q)$. We adjust techniques developed for single-criterion problems to solve problems with ordered criteria.

3. Priority-Generating Vector Functions

In this section we introduce the notion of a priority-generating vector function, describe sufficient conditions under which a vector function is priority-generating and give examples of such functions. The next section delivers algorithms for solving our main Problem (Problem 1) when the vector objective function is priority-generating on set $P_n(G_0, G_1, \ldots, G_q)$.

Let $\hat{P}$ be a set of all permutations $\pi_r = (i_1, i_2, \ldots, i_r)$, $r = 0, 1, \ldots, n$, of the elements of a set $N = \{1, 2, \ldots, n\}$. Here r is the length of a permutation π_r, $\{\pi_0\} = \emptyset$ and $\{\pi\}$ denotes the set of all elements, which the permutation π consists of. For a permutation set $P \subseteq \hat{P}$ we denote by $S[P]$ the set of all subpermutations or strings from P, i.e., $\pi' \in S[P]$ if and only if there exist $\sigma, \chi \in \hat{P}$ such that $(\sigma, \pi', \chi) \in P$.

Suppose that for a vector function $\Phi(\pi)$ defined over a set $P' \subseteq \hat{P}$ there exists a function $\omega(\pi)$ (scalar or vector), defined over set $S[P], P \subseteq P'$, and possessing the following property: for any $\alpha, \beta \in S[P]$ and any permutations $\pi' = (\sigma, \alpha, \beta, \chi)$ and $\pi'' = (\sigma, \beta, \alpha, \chi)$ such that $\pi', \pi'' \in P$, the inequality $\omega(\alpha) > \omega(\beta)$ implies $\Phi(\pi') \leqslant \Phi(\pi'')$ and equality $\omega(\alpha) = \omega(\beta)$ implies $\Phi(\pi') = \Phi(\pi'')$.

In this case we say that $\Phi(\pi)$ is a priority-generating on P vector function and $\omega(\pi)$ is its priority function.

Function $\omega(\pi)$ is called a strong priority function if in the above definition $\omega(\alpha) > \omega(\beta)$ implies $\Phi(\pi') < \Phi(\pi'')$. In this case we say that $\Phi(\pi)$ is a strong priority-generating on P vector function.

The notion of a priority-generating vector function is a natural generalization of the notion of a priority-generating function, introduced by Shafransky [22] see also [7, 28, 29]. These notions coincide when $m = 1$. Note also that the notion of a priority-generating function is close to the notions of a function possessing an adjacent sequence interchange property [18, 17] and a function admitting a string interchange relation [11]. Priority-generating functions play an important role in scheduling theory. Many spectacular results are obtained while optimizing priority-generating functions over a certain set $P \subseteq P_n$ of permutations (see, e.g., [28, 29].

Let us consider examples of single-criterion problems that can be reduced to minimizing priority-generating functions.

EXAMPLE 1. The jobs of a set N are processed on a single machine starting at time $t = 0$. Preemption is not allowed, and at most one job is processed at a time. The processing time of a job i depends on the starting time t_i^0 of its processing and equals $t_i = a_i t_i^0 + b_i, a_i > 0, b_i > 0, i = 1, 2, \ldots, n$. It is required to find a job processing sequence (i.e., a permutation π of $P \subseteq P_n$) that minimizes the makespan $F(\pi) = \sum_{i=1}^{n} t_i$.

We show that the function $F(\pi)$ is strong priority-generating over set P_n with the priority function

$$\omega(\pi) = \Psi(\pi)/F(\pi), \tag{1}$$

where $\pi = (i_1, i_2, \ldots, i_r)$ and $\Psi(\pi_0) = 0$, $\Psi(\pi) = \sum_{k=1}^{r} a_{i_k}(1 + \tau_{i_k})$, $\tau_{i_1} = 0$, $\tau_{i_k} = \sum_{l=1}^{k-1} a_{i_l}(1 + \tau_{i_l})$, $k > 1$.

Let $E(\pi, T)$ denote the makespan for the jobs of the set $\{\pi\}$ processed according to the sequence π, provided that the processing of the first job starts at time T, i.e., $t_{i_1}^0 = T$. Then $E(\pi, T) = F(\pi) + T\Psi(\pi)$. Indeed, let us use an induction over the length of the permutation π. For $|\{\pi\}| = 1$ the correctness of the equality is evident. Suppose, the equality holds for $|\{\pi\}| = k$, $k \geqslant 1$, and prove that it holds for $|\{\pi\}| = k + 1$. Let $\pi = (\pi', i_{k+1})$, where $|\{\pi'\}| = k$. We have $E(\pi, T) = E(\pi', T) + (T + E(\pi', T))a_{i_{k+1}} + b_{i_{k+1}} = E(\pi', T)(1 + a_{i_{k+1}}) + Ta_{i_{k+1}} + b_{i_{k+1}} = F(\pi') + T\Psi(\pi') + F(\pi')a_{i_{k+1}} + T\Psi(\pi')a_{i_{k+1}} + Ta_{i_{k+1}} + b_{i_{k+1}} = F(\pi) + T[\Psi(\pi')(1 + a_{i_{k+1}}) + a_{i_{k+1}}]$. On the other hand, $\Psi(\pi) = \Psi(\pi') + a_{i_{k+1}}(1 + \tau_{i_{k+1}}) = \Psi(\pi') + a_{i_{k+1}}[1 + \sum_{l=1}^{k} a_{i_l}(1 + \tau_{i_l})] = a_{i_{k+1}}[1 + \Psi(\pi')] + \Psi(\pi') = \Psi(\pi')[1 + a_{i_{k+1}}] + a_{i_{k+1}}$. Thus, $E(\pi, T) = F(\pi) + T\Psi(\pi)$.

Since we have $F(\sigma,\alpha,\beta,\chi) = F(\sigma) + E(\alpha, F(\sigma)) + E(\beta, F(\sigma,\alpha)) + E(\chi, F(\sigma, \alpha, \beta))$, the relation $F(\sigma, \alpha, \beta, \chi) \leqslant F(\sigma, \beta, \alpha, \chi)$ holds if and only if

$$\begin{aligned} &E(\alpha, F(\sigma)) + E(\beta, F(\sigma, \alpha)) + E(\chi, F(\sigma, \alpha, \beta)) \\ &\quad \leqslant E(\beta, F(\sigma)) + E(\alpha, F(\sigma, \beta)) + E(\chi, F(\sigma, \beta, \alpha)). \end{aligned}$$

The latter inequality is equivalent to $[1 + \Psi(\chi)][\Psi(\alpha)F(\beta) - \Psi(\beta)F(\alpha)] \geqslant 0$. Since $\Psi(\chi) \geqslant 0$, it follows that (1) is the desired priority function. Moreover, it is clear that this priority function is strong.

EXAMPLE 2. The jobs of a set N are processed on a single machine starting at time $t = 0$. For each job i, the processing time $t_i > 0$ and the function $\varphi_i(t)$ of the cost to be "paid" for having that job completed at time t are given. Preemption is not allowed, and at most one job is processed at a time. It is required to find a feasible (in a certain sense) job processing sequence $\pi \in P \subseteq P_n$, for which either the maximal processing cost

$$F(\pi) = \max\{\varphi_i(C_i(\pi)) \colon i \in N\}, \tag{2}$$

or the total processing cost

$$F(\pi) = \sum_{i=1}^{n} \varphi_i\left(C_i(\pi)\right) \tag{3}$$

is minimal.

Let us find conditions when the function (2) is priority-generating on P_n. Denote $E_i(\pi, T) = \varphi_i(C_i(\pi) + T)$ and $t(\pi) = \sum_{i \in \{\pi\}} t_i$. For $\pi' = (\sigma, \alpha, \beta, \chi)$ and $\pi'' = (\sigma, \beta, \alpha, \chi)$ consider the inequality $F(\pi') \leqslant F(\pi'')$ which holds if

$$\begin{aligned} &\max_{i\in\{\alpha\}, j\in\{\beta\}} \left\{E_i\left(\alpha, t(\sigma)\right), E_j\left(\beta, t(\sigma) + t(\alpha)\right)\right\} \\ &\quad\leqslant \max_{i\in\{\alpha\},\, j\in\{\beta\}} \left\{E_j\left(\beta, t(\sigma)\right), E_i\left(\alpha, t(\sigma) + t(\beta)\right)\right\}. \end{aligned}$$

This inequality holds if

$$\max_{j\in\{\beta\}} \left\{E_j\left(\beta, t(\sigma) + t(\alpha)\right)\right\} \leqslant \max_{i\in\{\alpha\}} \left\{E_i\left(\alpha, t(\sigma) + t(\beta)\right)\right\}$$

or, equivalently,

$$\max_{j\in\{\beta\}} \left\{E_j\left(\beta, T - t(\beta)\right)\right\} \leqslant \max_{i\in\{\alpha\}} \left\{E_i\left(\alpha, T - t(\alpha)\right)\right\}, \tag{4}$$

where $T = t(\sigma) + t(\alpha) + t(\beta)$.

a) Let $\varphi_i(t), i = 1, 2, \ldots, n$, satisfy the condition that $\varphi_i(0) \leqslant \varphi_j(0)$ implies $\varphi_i(t) \leqslant \varphi_j(t)$ for any t. Then the inequality (4) is equivalent to

$$\max_{j\in\{\beta\}} \left\{E_j\left(\beta, -t(\beta)\right)\right\} \leqslant \max_{i\in\{\alpha\}} \left\{E_i\left(\alpha, -t(\alpha)\right)\right\},$$

and we can choose

$$\omega(\pi) = \max_{i\in\{\pi\}} \Big\{\varphi_i\Big(C_i(\pi) - \sum_{i\in\{\pi\}} t_i\Big)\Big\}. \tag{5}$$

b) Let $\varphi_i(t) = \varphi(t - d_i)$, $i = 1, 2, \ldots, n$, where $\varphi(x)$ is a non-decreasing function. In this case functions $\varphi_i(t)$ satisfy the conditions from item a) and we can use the inequality

$$\max_{j\in\{\beta\}} \{E_j(\beta, -t(\beta))\} \leqslant \max_{i\in\{\alpha\}} \{E_i(\alpha, -t(\alpha))\}.$$

It is equivalent to

$$\max_{j\in\{\beta\}} \{\varphi(C_j(\beta) - t(\beta) - d_j)\} \leqslant \max_{i\in\{\alpha\}} \{\varphi(C_i(\alpha) - t(\alpha) - d_i)\}.$$

The latter inequality holds if

$$\max_{j\in\{\beta\}} \{C_j(\beta) - t(\beta) - d_j\} \leqslant \max_{i\in\{\alpha\}} \{C_i(\alpha) - t(\alpha) - d_i\}$$

or $L_{\max}(\beta) - t(\beta) \leqslant L_{\max}(\alpha) - t(\alpha)$. Therefore,

$$\omega(\pi) = L_{\max}(\pi) - \sum_{i\in\{\pi\}} t_i. \tag{6}$$

c) Let $\varphi_i(t) = at + w_i$, $i = 1, 2, \ldots, n$, where a is a real number. It is easy to check that the same argumentation gives the following formula for the priority function

$$\omega(\pi) = \max_{i\in\{\pi\}} \{aC_i(\pi) + w_i\} - a\sum_{i\in\{\pi\}} t_i. \tag{7}$$

d) Let $\varphi_i(t) = w_i \exp(\gamma t)$, $\gamma \neq 0$, $i = 1, 2, \ldots, n$. Then the function (2) is priority-generating as well and its priority function is

$$\omega(\pi) = \max_{i\in\{\pi\}} \{w_i \exp(\gamma C_i(\pi))\} / \exp\Big(\gamma \sum_{i\in\{\pi\}} t_i\Big). \tag{8}$$

All the priority functions (5)–(8) are not strong.

The function (3) is priority-generating in the following cases (we give here formulas for priority functions without proofs; they may be found in [28, 29].

a) Let $\varphi_i(t) = w_i t + b_i$, where b_i, $i = 1, 2, \ldots, n$, are real numbers. In this case a strong priority function for the function (3) is given by the formula

$$\omega(\pi) = \Big(\sum_{i\in\{\pi\}} w_i\Big) / \Big(\sum_{i\in\{\pi\}} t_i\Big). \tag{9}$$

b) Let $\varphi_i(t) = w_i \exp(\gamma t) + b_i$, $\gamma \neq 0$, $i = 1, 2, \ldots, n$. Then the function (3) is strong priority-generating and its priority function is

$$\omega(\pi) = \Big(F(\pi) - \sum_{i \in \{\pi\}} b_i\Big) / \Big(\exp\big(\gamma \sum_{i \in \{\pi\}} t_i\big) - 1\Big). \tag{10}$$

EXAMPLE 3. Let for every element $i \in N$ numbers $w_i \geqslant 0$ and $0 < \rho_i < 1$ be given. For $\pi = (i_1, i_2, \ldots, i_r) \in \hat{P}$ consider the following function (see an example with the system testing from the introduction, the first criterion)

$$F(\pi) = \sum_{k=1}^{r} w_{i_k} \Big(\prod_{l=1}^{k-1} \rho_{i_l}\Big) \quad \text{for } r > 1 \quad \text{and} \quad F(\pi) = w_{i_1} \text{ for } r = 1. \tag{11}$$

Function (11) is strong priority-generating on $\widehat{P}$ and its priority function is given by the formula

$$\omega(\pi) = F(\pi) / \Big(\prod_{i \in \{\pi\}} \rho_i - 1\Big). \tag{12}$$

Indeed, let $\pi' = (\sigma, \alpha, \beta, \chi)$, $\pi'' = (\sigma, \beta, \alpha, \chi)$ and $\rho(\pi) = \prod_{i \in \{\pi\}} \rho_i, \rho(\pi) = 1$ if $\{\pi\} = \emptyset$. Then $F(\pi') \leqslant F(\pi'')$ is equivalent to $\rho(\sigma)F(\alpha) + \rho(\sigma)\rho(\alpha)F(\beta) \leqslant \rho(\sigma)F(\beta) + \rho(\sigma)\rho(\beta)F(\alpha)$ or $(1 - \rho(\beta))F(\alpha) \leqslant (1 - \rho(\alpha))F(\beta)$. The latter relation gives the formula (12).

Some other examples of priority-generating functions may be found in [28, 29].

Now we consider sufficient conditions under which a vector function, composed of priority-generating scalar functions, is priority-generating.

Theorem 1. *A vector function $\Phi(\pi) = (F_1(\pi), \ldots, F_m(\pi))$ is priority-generating on P if each of the functions $F_1(\pi), \ldots, F_{m-1}(\pi)$ is strong priority-generating on P and $F_m(\pi)$ is priority-generating on P. If $F_m(\pi)$ is strong as well then $\Phi(\pi)$ is strong priority-generating on P.*

Proof. Let $\omega_k(\pi)$, $k \in \{1, 2, \ldots, m\}$, be a priority function for $F_k(\pi)$ (strong for $k \leqslant m - 1$). Define a function $\omega(\pi)$ over set $S[P]$ in the following way: $\omega(\pi) = (\omega_1(\pi), \omega_2(\pi), \ldots, \omega_m(\pi))$ for $\pi \in S[P]$.

Show that the function $\omega(\pi)$ is a priority function for $\Phi(\pi)$. Suppose that $\alpha, \beta \in S[P]$ and $\pi', \pi'' \in P$, where $\pi' = (\sigma, \alpha, \beta, \chi)$, $\pi'' = (\sigma, \beta, \alpha, \chi)$. Let $\omega(\alpha) > \omega(\beta)$. Then there exists k, $1 \leqslant k \leqslant m$, such that $\omega_k(\alpha) > \omega_k(\beta)$ and $\omega_l(\alpha) = \omega_l(\beta)$ for all $l < k$. If $k < m$ then $F_l(\pi') = F_l(\pi'')$ for $l = 1, 2, \ldots, k - 1$ and $F_k(\pi') < F_k(\pi'')$ since $F_j(\pi)$ are strong priority generating functions for $j < m$. So we have $\Phi(\pi') < \Phi(\pi'')$. If $k = m$, then $\omega(\alpha) > \omega(\beta)$ implies $\Phi(\pi') \leqslant \Phi(\pi'')$. In any case $\omega(\alpha) > \omega(\beta)$ implies $\Phi(\pi') \leqslant \Phi(\pi'')$. It is evident also that $\omega(\alpha) = \omega(\beta)$ implies $\Phi(\pi') = \Phi(\pi'')$.

Therefore $\omega(\pi)$ is a priority function for $\Phi(\pi)$ and $\Phi(\pi)$ is a priority-generating on P vector function. It is easy to see that $\Phi(\pi)$ is strong priority-generating on P if $F_m(\pi)$ has a strong priority function on $S[P]$.

Theorem 1 enables us to obtain priority-generating vector functions when having scalar priority-generating functions. For example, a vector function composed of functions (11) and (3) when $\phi_i(t) = w_i t + b_i$ is the objective function for the system testing problem from the introduction. This function is strong priority-generating on set P_n since both mentioned scalar functions are strong priority-generating on this set.

Let us consider two other classes of vector functions close to priority-generating functions.

Suppose that for a function $\Phi(\pi)$ there exists a function $\omega^{(1)}(i)$ defined over set N and possessing the following property: for any $i, j \in N$ and any permutations $\pi' = (\sigma, i, j, \chi)$ and $\pi'' = (\sigma, j, i, \chi)$ such that $\pi', \pi'' \in P$, the inequality $\omega^{(1)}(i) > \omega^{(1)}(j)$ implies $\Phi(\pi') \leqslant \Phi(\pi'')$ and $\omega^{(1)}(i) = \omega^{(1)}(j)$ implies $\Phi(\pi') = \Phi(\pi'')$. In this case $\omega^{(1)}(i)$ is called 1-priority function and $\Phi(\pi)$ is called 1-priority-generating on P vector function.

Let us introduce also a notion of a non-adjacent priority-generating function. A function $\Phi(\pi)$ is non-adjacent priority-generating if there exists a function $\omega^{(NA)}(i)$ defined over set N and possessing the following property. For any $i, j \in N$ and any permutations $\pi' = (\sigma, i, \delta, j, \chi)$ and $\pi'' = (\sigma, j, \delta, i, \chi)$ such that, $\pi', \pi'' \in P$, the inequality $\omega^{(NA)}(i) > \omega^{(NA)}(j)$ implies $\Phi(\pi') \leqslant \Phi(\pi'')$ and $\omega^{(NA)}(i) = \omega^{(NA)}(j)$ implies $\Phi(\pi') = \Phi(\pi'')$.

Notions of a strong 1-priority-generating function and a strong non-adjacent priority-generating function are introduced as in the case of priority-generating function.

Note 1. *A vector function $\Phi(\pi) = (F_1(\pi), \ldots, F_m(\pi))$ is 1-priority-generating on P if each of the functions $F_1(\pi), \ldots, F_{m-1}(\pi)$ is strong 1-priority-generating on P and $F_m(\pi)$ is 1-priority-generating on P. If $F_m(\pi)$ is strong 1-priority-generating as well then $\Phi(\pi)$ is strong 1-priority-generating on P.*

Validity of this proposition can be proved by the same argumentation as in the case of Theorem 1. An analogous proposition takes place for non-adjacent priority-generating functions.

Note 2. *Consider an example of a vector function composed of two scalar priority-generating functions where the first function is not strong: $\Phi(\pi) = (T_{\max}(\pi), \sum w_i C_i(\pi))$. Show that the function $\Phi(\pi)$ is not, in general, priority-generating. Set $N = \{1, 2, 3\}$, $t_1 = t_2 = 1$, $t_3 = 3$, $d_1 = 5$, $d_2 = 4$, $d_3 = 5$, $w_1 = 2$, $w_2 = 1$, $w_3 = 0$. Suppose $\alpha = (1)$, $\beta = (2)$, $\sigma = \emptyset$, $\chi = (3)$, then $\Phi(1, 2, 3) = (0, 4) < \Phi(2, 1, 3) = (0, 5)$. On the other hand, if for the same α and β we consider $\sigma = (3)$ and $\chi = \emptyset$, then $\Phi(3, 1, 2) = (1, 4) > \Phi(3, 2, 1) = (0, 5)$. Therefore, the function $\Phi(\pi)$ can be neither priority-generating nor even 1-priority-generating.*

Note 2 demonstrates that the conditions of Theorem 1 while being only sufficient are also rather close to necessary conditions for a vector function composed of scalar priority-generating functions to be priority-generating.

Any priority-generating function and any non-adjacent priority-generating function is simultaneously a 1-priority-generating function. The converse proposition is not true in general.

Consider examples of scalar non-adjacent priority-generating functions.

EXAMPLE 4. Let in Example 1 the processing time of job i equal $t_i = \varphi(t_i^0) + b_i$, where $\varphi(t)$ is a non-decreasing function. Show that in this case the function $F(\pi) = \sum_{k=1}^{n} t_{i_k}$, which is the makespan for the time of the jobs processed according to the sequence $\pi = (i_1, i_2, \ldots, i_n)$, is not priority-generating on P_n.

In fact, let $\varphi(t) = t$ for $0 \leqslant t \leqslant 8$, $\varphi(t) = 2t$ for $8 < t \leqslant 15$, $\varphi(t) = 3t$ for $t > 15$, $N = \{1,2,3,4,5\}$, $b_1 = 1$, $b_2 = 1$, $b_3 = 4$, $b_4 = 2$ and $b_5 = 4$. Assume that $\alpha = (2,3)$, $\beta = (4)$. If a priority function existed, then the relation of the form $F(\sigma,\alpha,\beta,\chi) \leqslant F(\sigma,\beta,\alpha,\chi)$ being satisfied for some σ and χ would hold for any other permutations σ^σ and χ'. However we have $F(1,2,3,4,5) = 132 > F(1,4,2,3,5) = 128$ but $F(5,2,3,4,1) = 505 < F(5,4,2,3,1) = 513$.

Show that the function $F(\pi)$ is non-adjacent priority-generating on P_n. Consider permutations $\pi' = (\sigma, i, \delta, j, \chi)$ and $\pi'' = (\sigma, j, \delta, i, \chi)$, and find conditions when the inequality $F(\pi') \leqslant F(\pi'')$ holds. To satisfy this inequality it is sufficient that $F(\sigma, i, \delta, j) \leqslant F(\sigma, j, \delta, i)$. As in Example 1, let $E(\pi, T)$ denote the makespan for jobs of the set $\{\pi\}$ processed according to the sequence π, provided that the processing of the first job starts at time T, i.e., $t_{i_1}^0 = T$. The last inequality is equivalent to

$$\begin{aligned}
&F(\sigma) + \varphi\left(F(\sigma)\right) + b_i + E\left[\delta, F(\sigma) + \varphi\left(F(\sigma)\right) + b_i\right] \\
&\quad + \varphi\left[F(\sigma) + \varphi\left(F(\sigma)\right) + b_i + E\left[\delta, F(\sigma) + \varphi\left(F(\sigma)\right) + b_i\right]\right] + b_j \\
&\quad \leqslant F(\sigma) + \varphi\left(F(\sigma)\right) + b_j + E\left[\delta, F(\sigma) + \varphi\left(F(\sigma)\right) + b_j\right] \\
&\quad + \varphi\left[F(\sigma) + \varphi\left(F(\sigma)\right) + b_j + E\left[\delta, F(\sigma) + \varphi\left(F(\sigma)\right) + b_j\right]\right] + b_i,
\end{aligned}$$

or

$$\begin{aligned}
&E\left[\delta, F(\sigma) + \varphi(F(\sigma)) + b_i\right] \\
&\quad + \varphi\left[F(\sigma) + \varphi(F(\sigma)) + b_i + E\left[\delta, F(\sigma) + \varphi(F(\sigma)) + b_i\right]\right] \\
&\quad \leqslant E\left[\delta, F(\sigma) + \varphi(F(\sigma)) + b_j\right] \\
&\quad + \varphi\left[F(\sigma) + \varphi(F(\sigma)) + b_j + E\left[\delta, F(\sigma) + \varphi(F(\sigma)) + b_j\right]\right].
\end{aligned}$$

Since the function $\varphi(t)$ is non-decreasing and the function $E(\pi, T)$ is non-decreasing in the second argument, this inequality holds if $b_i \leqslant b_j$. Therefore, the function $F(\pi)$ is non-adjacent priority-generating on P_n and its priority function is

$$\omega^{(NA)}(i) = -b_i. \tag{13}$$

EXAMPLE 5. Let for $\pi = (i_1, i_2, \dots, i_n) \in P \subseteq P_n$ the function $F(\pi)$ be defined as follows

$$F(\pi) = \sum_{k=1}^{n} w_{i_k} \varphi(k), \tag{14}$$

where $w_i \geqslant 0, i = 1, 2, \dots, n$.

The function $F(\pi)$ is non-adjacent priority-generating on P_n if $\varphi(t)$ is a monotone function.

a) If $\varphi(k)$ is a non-decreasing function then its priority function is

$$\omega^{NA}(i) = w_i. \tag{15}$$

Indeed, it is easy to see that $F(\sigma, i, \delta, j, \chi) \leqslant F(\sigma, j, \delta, i, \chi)$ is equivalent to $w_i\varphi(\mid \{\sigma\} \mid +1) + w_j\varphi(\mid \{\sigma\} \mid + \mid \{\delta\} \mid +2) \leqslant w_j\varphi(\mid \{\sigma\} \mid +1) + w_i\varphi(\mid \{\sigma\} \mid +$
$\mid \{\delta\} \mid +2)$. Since $\varphi(k)$ is a non-decreasing function the last inequality holds if $w_i \geqslant w_j$.

b) If $\varphi(k)$ is a non-increasing function then its priority function is

$$\omega^{NA}(i) = -w_i. \tag{16}$$

If $\varphi(t)$ is either an increasing or a decreasing function then the priority functions (15) and (16) are strong.

EXAMPLE 6. Consider Example 2 with the objective function (3): $F(\pi) = \sum\limits_{k=1}^{n} \varphi_i(C_i(\pi))$. Let $\varphi_i(t) = \varphi(t) + b_i$, where $\varphi(t)$ is a monotone function and $b_i, i = 1, 2, \dots, n$, are real numbers. Let $E(\pi, T)$ denote the value of the function $F(\pi)$ for the jobs of the set $\{\pi\}$ processed according to the sequence π, provided that the processing of the first job starts at time T, and $t(\pi) = \sum_{i\in\{\pi\}} t_i$. Relation $F(\sigma, i, \delta, j, \chi) \leqslant F(\sigma, j, \delta, i, \chi)$ is equivalent to

$$\begin{aligned} &F(\sigma) + \varphi(t(\sigma) + t_i) + b_i + E\left[\delta, t(\sigma) + t_i\right] + \varphi\left[t(\sigma) + t_i + t(\delta) + t_j\right] + b_j \\ &\leqslant F(\sigma) + \varphi(t(\sigma) + t_j) + b_j + E\left[\delta, t(\sigma) + t_j\right] + \varphi\left[t(\sigma) + t_j + t(\delta) + t_i\right] + b_i, \end{aligned}$$

or

$$\varphi(t(\sigma) + t_i) + E\left[\delta, t(\sigma) + t_i\right] \leqslant \varphi\left(t(\sigma) + t_j\right) + E\left[\delta, t(\sigma) + t_j\right]. \tag{17}$$

a) Let the function $\varphi(t)$ be non-decreasing. Since function $E(\pi, t)$ is non-decreasing in the second argument, the inequality (17) holds if $t_i \leqslant t_j$. Therefore, in this case the function (3) is non-adjacent priority-generating on P_n and its priority function is

$$\omega^{NA}(i) = -t_i. \tag{18}$$

b) Let the function $\varphi(t)$ be non-increasing. Then the function $E(\pi, t)$ is non-increasing in the second argument and the inequality (17) holds if $t_i \geqslant t_j$. In this case the function (3) is non-adjacent priority-generating on P_n as well. Its priority function is

$$\omega^{NA}(i) = t_i. \tag{19}$$

If $\varphi(t)$ is either an increasing or a decreasing function then the priority functions (18) and (19) are strong. It may be shown that the function (3) in both cases under consideration is not priority-generating on P_n.

4. Polynomial Time Algorithms for Minimizing Priority-Generating Functions

In this section we consider Problem 1 under the assumption that the objective vector function is priority-generating on the set of feasible permutations. Besides that we make some assumptions about graphs $G_0, G_1, G_2, \dots, G_q$. Particularly, we consider situations when all these graphs are tree-like or series-parallel. A case with arbitrary precedence constraints is also discussed.

Now we introduce some notions and notations.

Let $G = (N, U)$ be a directed graph, where $N = \{1, 2, \dots, n\}$ is the set of vertices and U is the set of arcs. Further we consider directed graphs without multiple arcs and circuits.

We write $i \overset{G}{\rightarrow} j$ (or $i \rightarrow j$) if there is a path in the graph G from the vertex i to the vertex j. We write $i \overset{G}{\circ\!\!\rightarrow} j$ if $i \overset{G}{\rightarrow} j$ and there is no vertex l in G such that $i \overset{G}{\rightarrow} l$ and $l \overset{G}{\rightarrow} j$.

If $i \overset{G}{\rightarrow} j$ then the vertex i is called a predecessor of the vertex j and j is a successor of i in G. If $i \circ\!\!\rightarrow j$ then i is an immediate predecessor of j and j is a direct successor of i.

The vertices i and j are called incomparable (denoted by $i \sim j$) if neither $i \rightarrow j$ nor $j \rightarrow i$ holds.

The vertex of the graph G is called initial if it has no predecessors in G. The vertex of the graph G is called terminal if it has no successors in G.

An arc $(i, j) \in U$ is called a transitive one if there is a path in G from the vertex i to the vertex j that does not contain the arc (i, j).

Denote by $A_G(i)$ and $B_G(i)$ the sets of all successors and predecessors of the vertex i and by $E(i)$ the set of all incomparable with i vertices. Similarly, we denote by $A_G^0(i)$ and $B_G^0(i)$ the sets of all direct successors and immediate predecessors of the the vertex i, respectively. Let $G' = (N', U')$ be a subgraph of the graph G. Then for $i \in N'$ denote $A_{G'}(i) = \{j \in N' \colon (i, j) \in U'\}$, $B_{G'}(i) = \{j \in N' \colon (j, i) \in U'\}$ and $E_{G'}(i) = \{j \in N' \colon (i, j) \notin U', (j, i) \notin U'\}$. Sets $A_{G'}^0(i)$ and $B_{G'}^0(i)$ are determined in the analogous way.

Introduce the following operations over directed circuit-free graph $G = (N, U)$ without transitive arcs.

The operation of identifying vertices i and j of a graph $G = (N, U)$ such that $i \in B^0_G(j)$ involves replacing these two vertices by a single vertex followed by removing the arc (i, j). In this case, all the arcs that enter or leave either i or j are replaced by those that either enter or leave the new vertex, respectively. All transitive arcs are removed from the obtained graph.

Suppose that in graph $G = (N, U)$ vertices i and j are such that $i \in E(j)$. The operation of including an arc (i, j) consists in addition of the arc (i, j) into U with consequent removing all transitive arcs from the obtained graph.

In the following, these operations are successively and repeatedly applied to graph $G = (N, U)$. While identifying two vertices $i, j \in N$ connected by the arc $(i, j) \in U$, the permutation $(i, j) \in P$ is associated with the new vertex.

A permutation $\pi = (i_1, i_2, \ldots, i_k) \in P$, corresponding to a vertex obtained as a result of successive identifying vertices is called a composite element and is denoted by $[i_1, i_2, \ldots, i_k]$. Further we do not distinguish the vertices and the corresponding elements.

A connected graph is called an outtree if there is a single initial vertex in it (called a root), and each other vertex has exactly one immediate predecessor. A connected graph is called an intree if there is a single terminal vertex in it and each other vertex has exactly one immediate successor.

The graph is called tree-like if each of its connected components is either an outtree or intree (i.e., treelike graph may contain outtrees and intrees simultaneously).

Suppose that $q = n$ in Problem 1 (or, that is the same, $q = 1$) and $m = 1$. It means that we have a problem with a single criterion and without partitioning N into groups. Algorithms for minimizing scalar priority-generating functions on the set $P_n(G_0)$ when G_0 is a tree-like, series-parallel [31] or arbitrary directed graph were proposed by Shafransky [22], Gordon and Shafransky [7], Shafransky [23], see also [28] and [29]. Similar algorithms were developed by Monma and Sidney [18], Monma [17]. The time complexity for tree-like and series-parallel cases is $\mathrm{O}(nlogn)$.

These algorithms may be adapted for lexicographic minimizing priority-generating vector functions on set $P_n(G_0, G_1, G_2, \ldots, G_q)$. Descriptions some of such algorithms are given below, but first we consider some properties of priority-generating functions. Since there is no principle difference between scalar and priority-generating vector functions, all results for scalar priority-generating functions are valid for vector functions.

Given graphs $G_i = (N_i, U_i)$, $i = 0, 1, \ldots, q$, construct a graph $G = (N, U)$ in the following way. For $i, j \in N_\nu$, $\nu \in \{1, 2, \ldots, q\}$, set $(i, j) \in U$ if $(i, j) \in U_\nu$. For $i \in N_\nu$, $j \in N_\mu$, $\nu \neq \mu$, set $(i, j) \in U$ if $(N_\nu, N_\mu) \in U_0$.

Denote by $P_n(G, q)$ the subset of all feasible with respect to G permutations, in which all elements from the same set N_ν are situated contiguously, $\nu = 1, 2, \ldots, q$. It is easy to see that $P_n(G, q) = P_n(G_0, G_1, G_2, \ldots, G_q)$. If $q = n$ (or, equivalently, $q = 1$), then $G = G_0$ and $P_n(G, n) = P_n(G)$.

Given $s, t \in N$ and graph $G = (N, U)$, denote $\bar{B}_G(s, t) = B_G(s) \backslash (B_G(t) \cup t)$ and $\bar{A}_G(s, t) = A_G(t) \backslash (A_G(s) \cup s)$

Consider first a case $q = n$, i.e., no grouping constraints are imposed.

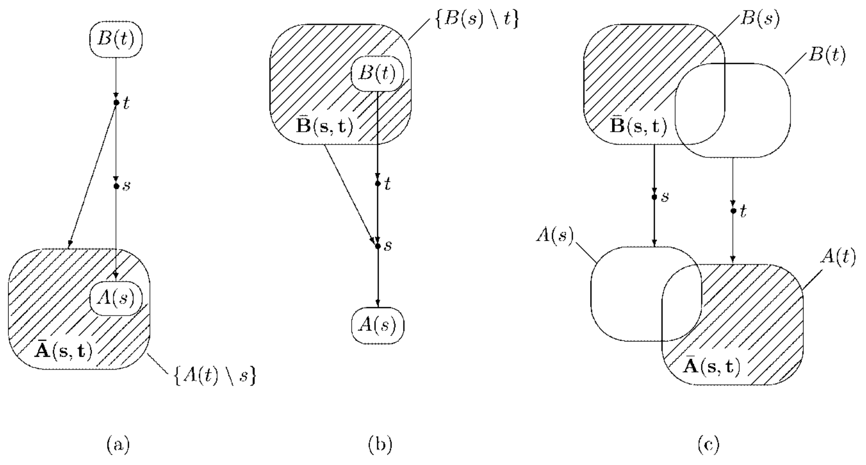
$B(t)$
t
s
$A(s)$
$\bar{\mathbf{A}}(\mathbf{s},\mathbf{t})$
$\{A(t) \setminus s\}$
$\{B(s) \setminus t\}$
$B(t)$
$\bar{\mathbf{B}}(\mathbf{s},\mathbf{t})$
t
s
$A(s)$
$B(s)$
$B(t)$
$\bar{\mathbf{B}}(\mathbf{s},\mathbf{t})$
s
t
$A(s)$
$A(t)$
$\bar{\mathbf{A}}(\mathbf{s},\mathbf{t})$
(a)
(b)
(c)

Fig. 1. Sets $\bar{A}(s,t)$ and $\bar{B}(s,t)$ ((a) $B^0(s) = t$; (b) $A^0(t) = s$; (c) $s \sim t$).

Theorem 2 [23]. *Let a function $\Phi(\pi)$ be priority-generating on $P_n(G)$, $B^0_G(s) = t$ and $\omega(s) \geqslant \omega(i)$ for all $i \in \bar{A}_G(s,t) \cup t$. Then for any permutation $\pi = (\dots,t,\sigma,s,\dots) \in P_n(G)$ there exists a permutation $\pi^0 = (\dots,t,s,\dots) \in P_n(G)$ such that $\{\pi\} = \{\pi^0\}$ and $\Phi(\pi^0) \leqslant \Phi(\pi)$.*

Theorem 3 [23]. *Let a function $\Phi(\pi)$ be priority-generating on $P_n(G)$, $s \in E_G(t)$ and $\omega(j) \geqslant \omega(i)$ for all $j \in \bar{B}_G(s,t) \cup s$ and $i \in \bar{A}_G(s,t) \cup t$. Then for any permutation $\pi = (\dots,t,\sigma,s,\dots) \in P_n(G)$ there exists a permutation $\pi^0 = (\dots,s,t,\dots) \in P_n(G)$ such that $\{\pi\} = \{\pi^0\}$ and $\Phi(\pi^0) \leqslant \Phi(\pi)$.*

As it was mentioned above, Theorems 2 and 3 are valid both for scalar and vector priority-generating functions. Now we formulate analogies of Theorems 2 and 3 for the case when $1 < q < n$.

Theorem 4. *Let a function $\Phi(\pi)$ be priority-generating on $P_n(G,q)$, $B^0_{G_\nu}(s) = t$ for some $\nu \in \{1,2,\dots,q\}$ and $\omega(s) \geqslant \omega(i)$ for all $i \in \bar{A}_{G_\nu}(s,t) \cup t$. Then for any permutation $\pi = (\dots,t,\sigma,s,\dots) \in P_n(G,q)$ there exists a permutation $\pi^0 = (\dots,t,s,\dots) \in P_n(G,q)$ such that $\{\pi\} = \{\pi^0\}$ and $\Phi(\pi^0) \leqslant \Phi(\pi)$.*

Theorem 5. *Let a function $\Phi(\pi)$ be priority-generating on $P_n(G,q)$, $s,t \in N_\nu$ and $s \in E_{G_\nu}(t)$ for some $\nu \in \{1,2,\dots,q\}$. If $\omega(j) \geqslant \omega(i)$ for all $j \in \bar{B}_{G_\nu}(s,t) \cup s$ and $i \in \bar{A}_{G_\nu}(s,t) \cup t$, then for any permutation $\pi = (\dots,t,\sigma,s,\dots) \in P_n(G,q)$ there exists a permutation $\pi^0 = (\dots,s,t,\dots) \in P_n(G,q)$ such that $\{\pi\} = \{\pi^0\}$ and $\Phi(\pi^0) \leqslant \Phi(\pi)$.*

Proof. To prove Theorems 4 and 5 it is sufficient to show that $\bar{B}_{G_\nu}(s,t) = \bar{B}_G(s,t)$ and $\bar{A}_{G_\nu}(s,t) = \bar{A}_G(s,t)$. Indeed, it follows from proofs of Theorems 2 and 3 that π^0 is of the form

$$\pi^0 = (\dots,\sigma',t,s,\sigma'',\dots) \quad \text{or} \quad \pi^0 = (\dots,\sigma',s,t,\sigma'',\dots),$$

respectively, where $\{\sigma'\} \cup \{\sigma''\} = \{\sigma\}$. In this case $\pi^0 \in P_n(G)$ implies $\pi^0 \in P_n(G,q)$. Therefore, if $\bar{B}_{G_\nu}(s,t) = \bar{B}_G(s,t)$ and $\bar{A}_{G_\nu}(s,t) = \bar{A}_G(s,t)$, then the validity of Theorems 4 and 5 follows immediately from Theorems 2 and 3.

Consider set $\bar{B}_{G_\nu}(s,t) = B_{G_\nu}(s) \backslash (B_{G_\nu}(t) \cup t)$. The procedure of the graph G construction involves that $l \in B_{G_\nu}(s)$ or $l \in B_{G_\nu}(t)$ implies $l \in B_G(s)$ or $l \in B_G(t)$, respectively. Therefore $l \in \bar{B}_{G_\nu}(s,t)$ implies $l \in \bar{B}_G(s,t)$. Let $l \in \bar{B}_G(s,t)$. If $l \in N_\nu$ then, evidently, $l \in \bar{B}_{G_\nu}(s,t)$. Otherwise, let $l \in N_\mu, \mu \neq \nu$. Then from the procedure of the graph G construction we have that $l \in B_G(s)$ and $l \in B_G(t)$ if $(N_\mu, N_\nu) \in U_0$, or $l \in A_G(s)$ and $l \in A_G(t)$ if $(N_\nu, N_\mu) \in U_0$, or $l \in E_G(s)$ and $l \in E_G(t)$ if $(N_\mu, N_\nu) \notin U_0$ and $(N_\nu, N_\mu) \notin U_0$. In any case $l \notin \bar{B}_G(s,t)$. Hence, $l \in N_\nu$ and $l \in \bar{B}_{G_\nu}(s,t)$. In the similar way we can prove the equality $\bar{A}_{G_\nu}(s,t) = \bar{A}_G(s,t)$.

Theorems 4 and 5 have a simple graph-theoretical interpretation. Satisfying the conditions of these theorems guarantees that using the operation of identifying vertices t and s (Theorem 4) or the operation of including arc (s,t) to graph G_ν (Theorem 5) enables one to obtain a new graph G'_ν with the following properties: $P_n(G_0, G_1, \ldots, G'_\nu, \ldots, G_q) \subseteq P_n(G_0, G_1, \ldots, G_\nu, \ldots, G_q)$ and for any permutation $\pi \in P_n(G_0, G_1, \ldots, G_\nu, \ldots, G_q)$ there exists a permutation $\pi^0 \in P_n(G_0, G_1, \ldots, G'_\nu, \ldots, G_q)$ such that $F(\pi^0) \leqslant F(\pi)$.

The operation of identifying vertices t and s satisfying the conditions of Theorem 2 is called a transformation I (the notation I$-[t,s]$). The operation of including arc (s,t) under the conditions of Theorem 3 is called a transformation II (the notation II$-(s,t)$). If, in graph G_ν, there exists a pair of vertices s and t satisfying either the conditions of Theorem 4 or those of Theorem 5, then we say that transformation I or transformation II, respectively, may be applied to graph G_ν or that transformation ($I - [t,s]$ or II$-(s,t)$) is feasible for graph G_ν.

COROLLARY 1. *Let a sequence $L^{(\nu)}$ of transformations I and II transforms graph G_ν into a chain G'_ν, $\nu \in \{1,2,\ldots,q\}$. Then there exists an optimal permutation $\pi^* = (\sigma_1, \sigma_2, \ldots, \sigma_q) \in P_n(G,q)$ such that $\pi'_\nu = \sigma_k$ for some $k \in \{1,2,\ldots,q\}$, where π'_ν is a feasible with respect to graph G'_ν permutation and $\{\pi'_\nu\} = N_\nu$.*

This follows from Theorems 4, 5 and the definition of set $P_n(G,q)$.

If there exists a sequence $L^{(\nu)}$ transforming graph G_ν into a chain G'_ν then by virtue of Corollary 1 we can consider permutation π'_ν as a composite element, which is associated with the corresponding vertex of graph G_0. Let in the result of applying a sequence of transformations I and II all graphs G_ν, $\nu = 1,2,\ldots,q$, be converted into chains. Denote the resulting graph by G'_0. Graph G'_0 has q vertices and a composite element π'_ν corresponds to vertex ν, $\nu = 1,2,\ldots,q$.

COROLLARY 2. *The relation $P_n(G'_0) \subseteq P_n(G,q)$ holds and there exists at least one optimal permutation $\pi^* \in P_n(G'_0)$.*

It follows directly from Corollary 1 and the procedure of graph G'_0 construction.

Theorem 6. *Let a sequence L of transformations I and II transform graph G_0' into a graph G_0''. Then $P_n(G_0'') \subseteq P_n(G, q)$ and there exists at least one optimal permutation $\pi^* \in P_n(G_0'')$.*

Proof. We have from Theorems 2 and 3 that $P_n(G_0'') \subseteq P_n(G_0')$ and for any permutation $\pi \in P_n(G_0')$ there exists a permutation $\pi^0 \in P_n(G_0'')$ such that $\Phi(\pi^0) \leqslant \Phi(\pi)$. Applying Corollary 2 we obtain $\pi^* \in P_n(G_0'') \subseteq P_n(G_0') \subseteq P_n(G, q)$.

If there exists a sequence of transformations I and II converting graph G_0' into a chain C, then the feasible with respect to C permutation is optimal.

All described below algorithms work according to the following scheme. A sequence of transformations I and II is applied to each graph G_ν, $\nu = 1, 2, \ldots, q$. If all these graphs are converted into chains, then consider obtained permutations π_ν' as composite elements associated with vertices of graph G_0'. Find a sequence of transformations converting G_0' into a chain as well. If we succeeded then the resulting sequence is optimal.

4.1. *Tree-like Graphs*

Here we consider a situation where all graphs presenting precedence constraints are tree-like. We propose an algorithm for finding an optimal permutation in such a situation. The algorithm is based on the mentioned transformations I and II, and its work is aimed to converting all the graphs into special chains. The resulting chain presents an optimal permutation.

Let all graphs G_ν, $\nu \in \{0, 1, 2, \ldots, q\}$, be tree-like. The following proposition is valid due to results from [23], see also Theorem 1 from [29].

Lemma 1. *Let a function $\Phi(\pi)$ be priority-generating on $P_n(G, q)$ and G_ν, $\nu \in \{1, 2, \ldots, q\}$, be a tree-like graph. Then there exists a sequence of transformations I and II converting G_ν into a chain.*

Let us suppose first that $G_\nu = (N_\nu, U_\nu)$ is an outtree, i.e., every vertex has no more than one immediate predecessor. The algorithm of transforming G_ν into a chain consists of a sequence of transitions from one outtree to another, each time reducing the number of vertices. Transformations are to be done until a single-vertex graph is obtained. On each step, the vertices of the current outtree are associated with some chains. On the first step every such sequence consists of a single element. The sequence of elements, which corresponds to the single vertex on the last step defines the permutation π_ν^*. Consider the general step of the algorithm.

Find, in G_ν, a vertex i^0 (called further a supporting vertex), with all direct successors $i_1, i_2, \ldots, i_l$ being terminal vertices. Let $C_1 = (j_1^1, \ldots, j_{p_1}^1)$, $C_2 = (j_1^2, \ldots, j_{p_2}^2), \ldots$, $C_l = (j_1^l, \ldots, j_{p_l}^l)$, be the element sequences corresponding to these terminal vertices. Unite these sequences into the sequence $C' = (j_1', j_2', \ldots, j_p')$, $p = p_1 + p_2 + \ldots + p_l$, in the following way. Find an index k such that $\omega(j_1^k) \geqslant \omega(j_1^r)$, for every r, $1 \leqslant r \neq k \leqslant l$, set $j_1' = j_1^k$ and remove j_1^k from the sequence C_k. Then among the first elements of the sequences we find an element with the maximal priority value and assign this element to the second position in C' and so on.

Join the supporting vertex i^0 to the sequence C' from the left in the following way. If $\omega(i^0) \leqslant \omega(j'_1)$ then unite i^0 and j'_1 into the composite element $[i^0, j'_1]$ redenoted by i^0. If $\omega(i^0) \leqslant \omega(j'_2)$ then unite i^0 and j'_2 into the composite element $[i^0, j'_2]$ denoted by i^0 and so on until obtaining a sequence $C_0 = (i^0, j'_k, \dots, j'_p)$ such that $\omega(i^0) > \omega(j'_k)$.

Remove from G_ν all successors of the vertex i^0, and associate the sequence C_0 with i^0. The obtained tree $G_\nu^{(1)}$ has no more than $n - 1$ vertices.

Applying the described transformations to the graph $G_\nu^{(1)}$, we obtain an outtree $G_\nu^{(2)}$, and so on until a graph $G_\nu^{(h)}$ consisting of a single vertex is obtained. The sequence corresponding to this vertex defines the desired permutation π_ν^*.

It is easy to check that the process of constructing sequence C' from sequences $C_1, C_2, \dots, C_l$ may be represented as an implementation of some transformations II. Similarly, the procedure of obtaining new composite element i^0 it is performing suitable transformation I.

The algorithm for the construction of the permutation π_ν^* when G_ν is an intree (i.e., every vertex has no more than one direct successor) slightly differs from the one for an outtree. In this case the supporting vertex is determined as follows. All its immediate predecessors have no predecessors in the tree obtained on the previous step. The sequence C_0 is obtained by joining the vertex i^0 to the sequence C' from the right. The composite element $[j'_p, i^0]$ is formed if $\omega(j'_p) \leqslant \omega(i^0)$.

Let $G_\nu = (N_\nu, U_\nu)$ be a tree-like graph, i.e., each of l its connected components is an outtree or an intree. Applying the described algorithms for outtrees and intrees we construct sequences $C_1, C_2, \dots, C_l$ which define permutations of elements of corresponding sets. Then we unite elements of the sequences $C_1, C_2, \dots, C_l$ into one sequence in accordance with their priorities (as it was done for sequences $C_1, C_2, \dots, C_l$ in the algorithm for outtrees). The obtained sequence defines the permutation π_ν^*.

Transform all graphs G_ν, $\nu = 1, 2, \dots, q$, into single-vertex graphs with corresponding permutations π_ν^*, and consider them as vertices of G_0 instead of initial sets $N_1, N_2, \dots, N_q$. Besides, consider each π_ν^* as one composite element. Transform obtained graph G'_0 into a chain in the described above way. By virtue of Lemma 1 there exists a sequence of transformations I and II converting G'_0 into a chain irrespectively of values of priorities of its vertices. In the result we obtain an optimal permutation π^*.

Note that it is possible to implement the described algorithm so that its running time does not exceed $\mathrm{O}(mnlogn)$. Corresponding technique is described, e.g., in [29].

Note 3. *As it was mentioned, there is known an algorithm for solving Problem 1 when graph G_0 is series-parallel, $q = n$ and $m = 1$. Like in case of tree-like graphs, this algorithm may be easily adopted for solving Problem 1 for arbitrary q and m when all graphs G_ν, $\nu \in \{0, 1, 2, \dots, q\}$, are series-parallel. The running time of such an algorithm does not exceed $\mathrm{O}(mnlogn)$ if all the graphs G_ν are represented by their decomposition trees.*

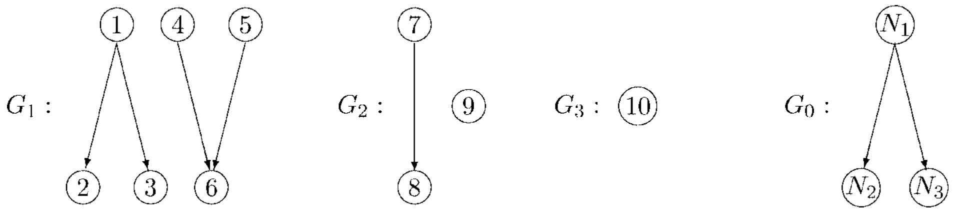

Fig. 2. Graphs of precedence constraints.

Table 1

Numerical parameters of jobs

i	1	2	3	4	5	6	7	8	9	10
t_i	5	3	2	4	5	2	5	5	2	4
w_i	3	7	1	8	5	3	2	4	3	3
d_i	6	8	10	12	14	15	18	19	20	20

Now we consider how does the above considered algorithm work in a particular example. Let number of jobs be equal to 10, number of groups be equal to 3, each of graphs G_0, G_1, G_2, G_3 be a tree (see Fig. 2) and for each job its processing time, weight and a due-date be given (see Table 1).

Objective function $\Phi(\pi) = (F_1(\pi), F_2(\pi))$ where $F_1(\pi) = \sum_{i=1}^{n} w_i C_i(\pi)$ and $F_2(\pi) = L_{max}(\pi)$. Here $L_{max}(\pi) = \max\{L_i(\pi) : i = 1, 2, \ldots, n\}$, $L_i(\pi) = C_i(\pi) - d_i$.

As it was shown in Section 3, both functions $F_1(\pi)$ and $F_2(\pi)$ are priority-generating. Besides, $F_1(\pi)$ is the strong priority-generating function. In view of Theorem 1 function $\Phi(\pi) = (F_1(\pi), F_2(\pi))$ is priority-generating vector function. Its priority function is $\omega(\pi) = (\omega_1(\pi), \omega_2(\pi))$, where $\omega_1(\pi) = \sum_{i\in\{\pi\}} w_i / \sum_{i\in\{\pi\}} t_i$ and $\omega_2(\pi) = L_{max}(\pi) - \sum_{i\in\{\pi\}} t_i$ (see formulas (9) and (6)).

Calculate values of the element priorities: $\omega(1) = (3/5, -6)$, $\omega(2) = (7/3, -8)$, $\omega(3) = (1/2, -10)$, $\omega(4) = (2, -12)$, $\omega(5) = (1, -14)$, $\omega(6) = (3/2, -15)$, $\omega(7) = (2/5, -18)$, $\omega(8) = (4/5, -19)$, $\omega(9) = (3/2, -20)$, $\omega(10) = (3/4, -20)$. Consider first connected component of graph G_1. Here vertex 1 is the supporting vertex. Since $\omega(2) > \omega(3)$ and $\omega(2) > \omega(1)$ we form the composite element $[1, 2]$, $\omega(1, 2) = (5/4, -8) > \omega(3)$. So, the first component is transformed into a single-vertex graph and sequence $([1, 2], 3)$ is associated with this vertex. Consider now second component of graph G. It is intree and vertex 6 is its supporting vertex. We have $\omega(4) > \omega(5)$ and $\omega(6) > \omega(5)$. Therefore, we form the composite element $[5, 6]$. Since $\omega(5, 6) = (8/7, -15) < \omega(4)$, the further formation of composite elements is impossible and the second component is transformed into a single-vertex graph with the associated sequence $(4, [5, 6])$. Unite the two sequences into a sequence π_1^*. We obtain $\pi_1^* = (4, [1, 2], [5, 6], 3)$ since $\omega(4) > \omega(1, 2) > \omega(5, 6) > \omega(3)$. It is easy to see that $\pi_2^* = (9, [7, 8])$, $\pi_3^* = (10)$, and $\omega(\pi_2^*) = (3/4, -19)$. Now consider graph G_0. Since $\omega(\pi_2^*) > \omega(\pi_3^*)$ we have that $\pi^* = (\pi_1^*, \pi_2^*, \pi_3^*) = (4, 1, 2, 5, 6, 3, 9, 7, 8, 10)$.

4.2. *Arbitrary Directed Graphs*

Let graphs G_ν, $\nu \in \{0,1,2,\ldots,q\}$, be arbitrary directed graphs without circuits. In this case the problem of finding π^* is NP-hard (see, e.g., [29]). Nevertheless, we can apply to the graph G_ν, $\nu \in \{1,2,\ldots,q\}$, transformations I and II that allow to obtain such a graph G'_ν that

$$P_n(G_0,G_1,\ldots,G'_\nu,\ldots,G_q) \subseteq P_n(G_0,G_1,\ldots,G_\nu,\ldots,G_q),$$

and the set $P_n(G_0,G_1,\ldots,G'_\nu,\ldots,G_q)$ contains at least one optimal permutation. In some cases there exists a sequence of these transformations, which transfers all graphs G_ν into chains. Proceeding further as in case of tree-like we may consider corresponding permutations as composite elements, which are assigned to vertices of graph G_0. Then we can use transformations I and II to try to transfer obtained graph G'_0 into a chain as well. In success we obtain an optimal permutation π^*.

Below we describe the common scheme of the algorithm.

The algorithm consists of a sequence of transformations I and II, which transforms each graph G_ν into a so-called deadlock graph (i.e., a graph for which no feasible transformations I and II exist). The time complexity of this algorithm is not greater than $\mathrm{O}(n_\nu^4)$ for $m = 1$ [23] (see also [28], [29]). Therefore, the running time of the algorithm for a graph G_ν does not exceed $\mathrm{O}(mn_\nu^4)$ and the total running time does not exceed $\mathrm{O}(mn^4)$.

For each graph G_ν, $\nu = 1,2,\ldots,q$, the algorithm consists of the following stages.

(a) Form a list of elements of the set N_ν.

(b) Scanning the list, apply all possible transformations II to the graph G_ν. If no transformation II can be applied to the current graph, return to the beginning of the list and go to the stage (c).

(c) Choosing the next element i' in the current list, check whether transformation $\mathrm{I}-[j,i']$, where $j \circ\!\!\rightarrow i'$, can be applied to the current graph. If this transformation $\mathrm{I}-[j,i']$ is feasible, then it is performed. Modify the current list (i.e., remove the elements j and i' from the list and include the composite element $[j,i']$ into it), return to its beginning and go to (b). If transformation $\mathrm{I}-[j,i']$ is non-feasible for the current graph, take the next element in the list.

The described process is completed when neither transformation I nor transformation II can be applied to the current graph, i.e., in performing (c) no transformation has been done.

Let all graphs G_ν be converted into chains. Considering corresponding permutations as composite elements and substituting them into graph G_0, we obtain graph G'_0. Implement possible transformations I and II for G'_0 according to the above algorithm.

5. Minimization on the Set of All Permutations

Consider a case of Problem 1 when $q = n$ and $G_0 = (N,\emptyset)$, i.e., there is neither group partition nor precedence constraints. We discuss a situation where $\Phi(\pi) =$

$(\Phi_1(\pi), \Phi_2(\pi))$ and vector function $\Phi_1(\pi)$ is 1-priority-generating. Besides, the problem of minimizing vector function $\Phi_2(\pi)$ on a set $P_n(G_0, G_1, \ldots, G_q)$ is supposed to be polynomially solvable if G_0 is a chain and $G_k = (N_k, \emptyset)$, $k = 1, 2, \ldots, q$.

Theorem 7. *Let the problem* $1 \mid\mid \Phi(\pi)$*, where* $\Phi(\pi) = (\Phi_1(\pi), \Phi_2(\pi))$*, satisfy the following conditions. Vector function* $\Phi_1(\pi)$ *has a strong 1-priority function* $\omega'(i)$ *and there exists a polynomial time algorithm for the problem* $1 \mid G_0 = C, G_k = (N_k, \emptyset)$, $k = 1, 2, \ldots, q \mid \Phi_2(\pi)$*. Then the problem* $1 \mid\mid \Phi(\pi)$ *is polynomially solvable if the time complexity of calculating values of function* $\omega'(i)$ *is polynomial.*

Proof. Let us consider the problem $1 \mid\mid \Phi_1(\pi)$. We construct graph $G = (N, U)$ as follows. For $i, j \in N$ we set $(i, j) \in U$ if $\omega'(i) > \omega'(j)$. Here $\Phi_1(\pi) = (F'_1(\pi), \ldots, F'_\tau(\pi))$ and $\omega'(i)$ is a strong 1-priority function for $\Phi_1(\pi)$. We claim that the set of all optimal permutations for $1 \mid\mid \Phi_1(\pi)$ and a set of all feasible with respect to G permutations coincide. In fact, if $\pi^* = (i_1, i_2, \ldots, i_n)$ is an optimal permutation for $1 \mid\mid \Phi_1(\pi)$ then $\omega'(i_\eta) \geqslant \omega'(i_\mu)$ for any $\eta < \mu$. From the procedure of the graph G construction we have $i_\eta \to i_\mu$ if $\omega'(i_\eta) > \omega'(i_\mu)$ and $i_\eta \sim i_\mu$ if $\omega'(i_\eta) = \omega'(i_\mu)$. Therefore π^* is feasible with respect to G. The reverse proposition may be easily checked in the same manner.

Now we consider a structure of graph G. Let $N_k = \{i \in N \setminus \cup_{l=1}^{k-1} N_l : \omega(i) = \max\{\omega(j) : j \in N \setminus \cup_{l=1}^{k-1} N_l\}\}$, where $N_0 = \emptyset$. It follows from the definition of graph G that $i \to j$ for any $i \in N_\eta$, $j \in N_\mu$ if $\eta < \mu$. That is $G = (\cup_{l=1}^{q} N_l, U)$, where $\cup_{l=1}^{q} N_l = N$, $U = \{(i, j) : i \in N_\eta, j \in N_\mu, 1 \leqslant \eta < \mu \leqslant q\}$ and $i' \sim j'$ for any i', j' from the same subset N_k. Therefore, $P_n(G) = P_n(C_0, G_1, \ldots, G_q)$, where $C_0 = (\{N_1, \ldots, N_q\}, U_c)$, $G_k = (N_k, \emptyset)$, $k = 1, \ldots, q$, $U_c = \{(N_\eta, N_\mu) : 1 \leqslant \eta < \mu \leqslant q\}$.

The time complexity of obtaining sets N_k, $k = 1, 2, \ldots, q$, does not exceed $O(\rho n log n)$ where ρ is the time complexity of calculating values of $\omega'(i)$. The problem of finding such a permutation $\pi' \in P_n(G) = P_n(C_0, G_1, \ldots, G_q)$ that $\Phi_2(\pi') = \min\{\Phi_2(\pi) : \pi \in P_n(G)\}$ is polynomially solvable. So we conclude that the problem $1 \mid\mid (\Phi_1(\pi), \Phi_2(\pi))$ is polynomially solvable as well.

Consider now situations where Theorem 7 can be applied. Suppose that $\Phi_1(\pi) = F'_1(\pi), \ldots, F'_\tau(\pi))$ and $\Phi_2(\pi) = F_1(\pi), \ldots, F_\lambda(\pi)$, $\lambda = m - \tau$.

5.1. *1-priority-generating Vector Functions*

In the case when vector function $\Phi_2(\pi)$ is 1-priority-generating with 1-priority function $\omega(i)$, the problem $1 \mid G_0 = C, G_k = (N_k, \emptyset), k = 1, 2, \ldots, q \mid \Phi_2(\pi)$ is $O(\rho n log n)$ solvable. Here ρ is the time complexity of calculating values of function $\omega(i)$. Indeed, to solve the mentioned problem it is enough to order elements of each set N_k in non-increasing of their priorities $\omega(i)$.

5.2. *Maximal Cost*

Let for each job i there are given $\lambda = m - \tau$ non-decreasing penalty functions $\varphi_i^{(\mu)}(t)$, $\mu = 1, 2, \ldots, \lambda$, $F_\mu(\pi) = \max\{\varphi_i^{(\mu)}(C_i(\pi)) : i \in N\}$ and $\Phi_2(\pi) =$

$(F_1(\pi), F_2(\pi), \ldots, F_\lambda(\pi))$. As usually, here $C_i(\pi)$ is the completion time of the job i in a schedule determined by the permutation π.

Consider problem

$$1 \mid r_i = T,\ G \mid \Phi_2(\pi). \tag{20}$$

Here r_i is the release time of job i and for all jobs their release times are the same and equal to T, $G = (N, U)$ is an arbitrary directed circuit-free graph. To solve Problem (20) modify known algorithm by Lawler [10] in the following way.

Algorithm 1. Denote $\varphi_i(t) = (\varphi_i^{(1)}(t), \varphi_i^{(2)}(t), \ldots, \varphi_i^{(\lambda)}(t)), i \in N$. Let N^- be the set of all terminal vertices of the graph G. Considering inequality $\varphi_i(t) \leqslant \varphi_j(t)$ in the lexicographic sense, we find such an element $i_n \in N^-$ that

$$\varphi_{i_n}(\theta) = \min\{\varphi_i(\theta) : i \in N^-\},$$

where $\theta = T + \sum_{i\in N} t_i$. Set $N = N \backslash i_n$ and delete the vertex i_n from G, calculate new value of θ. Find the next element $i_{n-1} \in N^-$ such that $\varphi_{i_{n-1}}(\theta) = \min\{\varphi_i(\theta) : i \in N^-\}$ and so on. Repeating this process, we eventually find elements $i_{n-2}, \ldots, i_2, i_1$. Set $\pi^* = (i_1, i_2, \ldots, i_n)$.

The proof of the optimality of the permutation π^* is almost the same as in case of Lawler's algorithm. Running time of Algorithm 1 is $\mathrm{O}(\rho n^2)$, where ρ is the time complexity of calculating values of function $\varphi_i(t)$.

For solving problem $1 \mid G_0 = C, G_k = (N_k, \emptyset), k = 1, 2, \ldots, q \mid \Phi_2(\pi)$ it is enough to apply Algorithm 1 to each problem $1 \mid r_i = T_k, G_k \mid \Phi_2(\pi)$, $k = 1, 2, \ldots, q$, where $T_k = \sum_{\nu=1}^{k-1} t(N_\nu)$, $t(N_\nu) = \sum_{i\in N_\nu} t_i$, $T_1 = 0$. These problems are even simpler than the one considered above since $G_k = (N_k, \emptyset)$. Here we have $N^- = N$. Having obtained optimal permutations $\pi_1^*, \pi_2^*, \ldots, \pi_q^*$ for these problems, we construct resulting permutation $\pi^* = (\pi_1^*, \pi_2^*, \ldots, \pi_q^*)$. Optimality of π^* follows from the fact that processing jobs of set N_k cannot start before the moment T_k, $k = 1, 2, \ldots, q$. Total running time of the algorithm will not exceed $\mathrm{O}(\rho n^2)$.

5.3. *Number of Late Jobs*

Let for each job $i \in N$ we are given a due-date d_i. Define $\Phi_2(\pi) = F(\pi) = \sum_{i\in N} U_i(\pi)$, where $U_i(\pi) = 0$ if $C_i(\pi) \leqslant d_i$ and $U_i(\pi) = 1$ if $C_i(\pi) > d_i$. Thus we have problem $1 \mid G_0 = C, G_k = (N_k, \emptyset), k = 1, 2, \ldots, q \mid \sum U_i$.

Consider an auxiliary problem $1 \mid r_i = T \mid \sum U_i$. This one is equivalent to the problem $1 \mid\mid \sum U_i'$, where U_i' are determined for modified due-dates $d_i' = d_i - T, i \in N$. To solve the last problem we use the known $\mathrm{O}(nlogn)$ algorithm by Moore [20].

To solve problem $1 \mid G_0 = C, G_k = (N_k, \emptyset), k = 1, 2, \ldots, q \mid \sum U_i$, calculate values $T_k = \sum_{\nu=1}^{k-1} t(N_\nu)$, $t(N_\nu) = \sum_{i\in N_\nu} t_i$, $T_1 = 0$, and solve q auxiliary problems $1 \mid r_i = T_k, G_k = (N_k, \emptyset) \mid \sum U_i, k = 1, 2, \ldots, q$, using Moor's algorithm in the described above way. Having obtained optimal permutations $\pi_1^*, \pi_2^*, \dots, \pi_q^*$ for these problems, we construct resulting permutation $\pi^* = (\pi_1^*, \pi_2^*, \dots, \pi_q^*)$. Optimality of π^* follows from the fact that processing jobs of set N_k cannot start before the moment T_k, $k = 1, 2, \dots, q$.

It is clear, that the time complexity of the resulting algorithm does not exceed $\mathrm{O}(nlogn)$.

6. Maximal Cost and Non-Adjacent Priority-Generating Functions

In this section we consider a case of Problem 1 when $q = n$, i.e., there are no group partition constraints. For simplicity, denote G_0 by $G = (N, U)$. We discuss a situation where $\Phi(\pi) = (\Phi_1(\pi), \Phi_2(\pi))$, function $\Phi_1(\pi) = (F_1(\pi), F_2(\pi), \dots, F_{m_1}(\pi))$ is a vector maximal cost, i.e., all functions $F_k(\pi)$, $k = 1, 2, \dots, m_1$, are maximal costs (see Subsection 5.2). The vector function $\Phi_2(\pi)$ is supposed to be a non-adjacent priority-generating on $P_n(G)$.

If G is an arbitrary directed graph then the problem under consideration is NP-hard. Indeed, let $m_1 = 1$ and $\varphi_i(t) = \mathrm{const}$ for all $i \in N$. It implies that any permutation from $P_n(G)$ is optimal with respect to $\Phi_1(\pi)$. So we should minimize $\Phi_2(\pi)$ over $P_n(G)$, but this problem is NP-hard in the strong sense because of strongly NP-hardiness of minimizing the sum of completion times $\sum_{i=1}^n C_i$ (this function is non-adjacent priority-generating), see, e.g., [29].

Consider a polynomial solvable case of the problem. Suppose that the following conditions hold

$$\omega^{NA}(i) \geqslant \omega^{NA}(j) \quad \text{implies} \quad t_i \leqslant t_j \quad \text{for all} \;\; i, j \in N \tag{21}$$

and

$$i \rightarrow j \quad \text{implies} \quad \omega^{NA}(i) \geqslant \omega^{NA}(j) \quad \text{for all} \;\; i, j \in N. \tag{22}$$

For finding the permutation π^* which minimizes $\Phi(\pi)$ on $P_n(G)$ we make two steps. On the first step we find the permutation π' such that $\Phi_1(\pi') = \min\{\Phi_1(\pi) : \pi \in P_n(G)\}$. It may be done by Algorithm 1 in $\mathrm{O}(\rho n^2)$ operations (see Subsection 5.2). Denote $F_k = F_k(\pi')$, $k = 1, 2, \dots, m_1$.

On the second step we use the following algorithm.

Algorithm 2. Set $M = N$, $l = n$. Let M_1 be the set of all elements from M which are associated with terminal vertices of G. Find such a set M_1' that

$$M_1' = \Big\{ j \in M_1 : \varphi_j^k\Big(\sum_{p \in M} t_p \Big) \leqslant F_k, k = 1, 2, \dots, m_1 \Big\}. \tag{23}$$

On the l-th position in the permutation π^* we allocate such an element $i \in M_1'$, that $\omega^{NA}(i) \leqslant \omega^{NA}(j)$ for every $j \in M_1'$.

Then we set $l = l - 1$, $M = M \backslash i$ and allocate the next element on l-th position.

Theorem 8. *Algorithm 2 gives an optimal permutation for the function* $\Phi(\pi)$ *under the constrains* (21) *and* (22).

Proof. Show that the permutation $\pi^* = (i_1, i_2, \dots, i_n)$ minimizes the function $\Phi(\pi)$ on $P_n(G)$. Let $\pi' = (s_1, s_2, \dots, s_n)$ be an optimal permutation and $i_n = s_\nu$, $\nu < n$. Find the least r, $\nu < r \leqslant n$ such that $\omega^{NA}(s_r) \geqslant \omega^{NA}(s_\nu)$. The index r exists, since due to Algorithm 2 an inequality $\omega^{NA}(s_n) \geqslant \omega^{NA}(s_\nu)$ holds. Then for every r_1, $\nu < r_1 < r$ we have $\omega^{NA}(s_{r_1}) < \omega^{NA}(s_\nu)$. Due to condition (22), s_{r_1} is not a predecessor of s_r for every r_1, $\nu < r_1 < r$. So swapping s_ν and s_r we get a feasible permutation π'' and $\Phi_2(\pi'') \leqslant \Phi_2(\pi')$. Now we show that the permutation π'' is optimal with respect to $\Phi_1(\pi)$. Due to condition (21) the inequality $t_{s_\nu} \geqslant t_{s_r}$ holds. So $\varphi_i^k(C_i(\pi'')) \leqslant \varphi_i^k(C_i(\pi'))$, $k = 1, 2, \dots, m_1$, for every $i \neq s_\nu$ and $\varphi_{s_\nu}^k(C_{s_\nu}(\pi'')) \leqslant \varphi_{s_\nu}^k(C_{s_\nu}(\pi^*)) \leqslant F_k$, $k = 1, 2, \dots, m_1$. Thus we have $F_k(\pi'') \leqslant F_k$, $j = 1, 2, \dots, m_1$, and the permutation π'' is optimal with respect to $\Phi_1(\pi)$. We may repeat such transformations and find an optimal permutation which has the element i_n on the last position. Similarly, in at most $n-1$ steps we find an optimal permutation that coincides with π^*.

The time complexity of Algorithm 2 is $\mathrm{O}(\rho n^2)$, so time complexity of finding π^* is also $\mathrm{O}(\rho n^2)$. Here ρ is the time complexity of calculating values of functions φ_j^k. We do not take into account the complexity of calculating values of functions $\omega^{NA}(i)$.

7. Conclusion

Usually optimization problems with ordered criteria are solved in the following way. First we are looking for the set of all optimal solutions with respect to the first, most important, criterion. Then we are searching for the subset of this set that consists of all optimal solutions with respect to the second criterion and so on. This manner of solving is very difficult and complicated. In this paper, due to constructing and applying an extension of the notion of priority-generating vector function we managed to solve several cases of the general problem in very effective polynomial way. The similar approach we use in case of vector objective function composed of maximal penalties and in some other situations.

It easy to check that all polynomially solvable problems listed in Section 2 are special cases of problems considered in Sections 5–6. So in the paper we give general techniques for solving all known and numerous new problems with ordered criteria.

The investigation of other sequencing and scheduling problems with ordered criteria is of a certain interest for future research.

The problem with ordered criteria may be reduced to sequential solving of single-criterion problems. Sometimes it is the only possible way for a problem solving. So the single-criterion problem of finding all optimal solutions is of a special interest for solving problems with ordered criteria. Some results on this topic for sequencing problems were obtained by Monma and Sidney [19] and Tuzikov [30].

8. Acknowlegments

The first author is partially supported by the KBN Grant No P406 003 05, the second author is supported by ISTC Project B-104-98.

References

[1] Bansal, S.P. (1980). Single machine scheduling to minimize weighted sum of completion times with secondary criterion. *Europ. J. Oper. Res.*, **5**, 177–181.

[2] Brucker, P., J.K. Lenstra, A.H.G. Rinnooy Kan (1975). Complexity of machine scheduling problems. Technical Report BW 43, Mathematische Centrum, Amsterdam.

[3] Burns, R.N. (1976). Scheduling to minimize the weighted sum of completion times with secondary criteria. *Naval. Res. Logist. Quart.*, **23**, 125–129.

[4] Chen, C.-L., R.L. Bulfin (1993). Complexity of single machine multi-criteria scheduling problems. *Eur. J. Oper. Res.*, **70**, 115–125.

[5] Dileepan, P., P. Sen (1988). Bicriterion static scheduling research for a single machine. *Omega*, **16**, 53–59.

[6] Emmons, H. (1975). A note on a scheduling problem with dual criteria. *Naval. Res. Logist. Quart.*, **22**, 615–616.

[7] Gordon, V.S., Y.M. Shafransky (1978). Optimal ordering with series-parallel precedence constraints. *Doklady Akademii Nauk BSSR*, **22**(3), 244–247 (in Russian).

[8] Ham, I., K. Hitomi, Y. Yoshida (1985). *Group Technology*. Kluwer-Nijhoff, Dordrecht.

[9] Heck, H., S.A. Roberts (1972). A note on the extension of a result on scheduling with secondary criteria. *Naval. Res. Logist. Quart.*, **19**, 403–405.

[10] Lawler, E.L. (1973). Optimal sequencing of a single machine subject to precedence constraints. *Management Science*, **19**(5), 544–546.

[11] Lawler, E.L. (1983). Recent results in the theory of machine scheduling. In A. Bachem, M. Grötschel and B. Korte (Ed.), *Mathematical Programming: the State of Art*. Springer, Berlin, pp. 202–234.

[12] Lawler, E.L., J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys (1989). Sequencing and scheduling: Algorithms and complexity. *Technical Report BS-R8909*, Center for Mathematics and Computer Science, Amsterdam.

[13] Lenstra, J.K., A.H.G. Rinnooy Kan, P. Brucker (1977). Complexity of machine scheduling problems. *Ann. Discr. Math.*, **1**, 343–362.

[14] Livshitz, E.M. (1968). The sequence of operations for complex detail production. *Avtomatika i Telemekhanika*, **11**, 94–95 (in Russian).

[15] Mitten, L.G. (1960). An analytic solution to the least cost testing sequence problem. *J. Industrial Engineering*, **11**, 17–23.

[16] Miyazaki, S. (1981). One machine scheduling problem with dual criteria. *J. Oper. Res. Soc. Japan*, **24**, 37–50.

[17] Monma, C.L. (1981). Sequencing with general precedence constraints. *Discrete Appl. Math.*, **3**, 137–150.

[18] Monma, C.L., J.B. Sidney (1979). Sequencing with series-parallel precedence constraints. *Math. Oper. Res.*, **4**, 215–224.

[19] Monma, C.L., J.B. Sidney (1987). Optimal sequencing via modular decomposition: Characterization of sequencing functions. *Math. Oper. Res.*, **12**, 22–31.

[20] Moore, J.M. (1968). An n job, one machine sequencing algorithm for minimizing the number of late jobs. *Management Science*, **15**(1), 102–109.

[21] Podinovsky, V.V., V.M. Gavrilov (1975). *Optimization on Sequentially Applied Criteria*. Sovetskoe Radio, Moscow (in Russian).

[22] Shafransky, Y.M. (1978). On optimal sequencing for deterministic systems with tree-like partial order. *Vestsi Akademii Navuk BSSR. Ser. Fizika-Matem. Navuk* **2**, 119–120 (in Russian).

[23] Shafransky, Y.M. (1980). On a problem of minimizing functions over a set of permutations of partially ordered elements. *Vestsi Akademii Navuk BSSR. Ser. Fizika-Matem. Navuk* **5**, 132 (in Russian).

[24] Shanthikumar, J.G. (1983). Scheduling n jobs on one machine to minimize the maximal tardiness with minimum number tardy. *Comput. and Oper. Res.*, **10**, 255–266.

[25] Slowinski, R. (1989). Multiobjective project scheduling under multiple-category resource constraints. In R. Slowinski and J. Weglon (Eds.), *Advances in Project Scheduling*. Elsevier Science Publishers B.U., Amsterdam.

[26] Smith, W.E. (1956). Various optimizers for single stage production. *Naval. Res. Logist. Quart.*, **3**, 59–66.

[27] Tabucanon, M.T., V. Chankong (1989). *Multiple Criteria Decision Making: Application in Industry and Service*. Asian Institute of Technology.

[28] Tanaev, V.S., V.S. Gordon, Y.M. Shafransky (1984). *Scheduling Theory. Single-Stage Systems*. Nauka, Moscow (in Russian).

[29] Tanaev, V.S., V.S. Gordon, Y.M. Shafransky (1994). *Scheduling Theory. Single-Stage Systems*. Kluwer Academic Publishers, Dordrecht.

[30] Tuzikov, A.V. (1985). *Methods of Solving Some Classes of Multicriterion Scheduling Problems*. PhD thesis, Institute of Engineering Cybernetics, Minsk (in Russian).

[31] Valdes, J., R.E. Tarjan, E.L. Lawler (1982). The recognition of series-parallel digraphs. *SIAM J. Comput.*, **11**, 298–313.

[32] Yuan, J., Y. Zhao (1994). Strongly NP-hardness results in the single machine secondary criterion scheduling problems with weigted flowtime or tardiness as one of the criteria. *ZOR*.

[33] Zinder, Y.A., T.P. Podchasova (1976). Minimizing ordered criteria in single machine systems. In *Automated Control and Data Processing Systems*. Kiev, pp. 20–25 (in Russian).

A. Janiak is a full professor in Computer Science, Operations Research and Industrial Engineering Areas in Institute of Engineering Cybernetics at Wroclaw University of Technology. He is the author of 2 books and more than 100 papers. His research interests are: sequencing and scheduling problems with classical and generalized models of operations in computer and manufacturing systems, resource allocation problems, complexity theory, theory of algorithms. He is a member of Computer Science Committee of Polish Academy of Science and a member of Computer Science Methods Section of State Committee for Scientific Research. He was invited as a visiting professor to some universities in Australia, Canada, Germany, Hong Kong, Israel, New Zeland, Thailand and France.

Y. Shafransky graduated from the Belarus State University (Minsk) in 1972. Now he is a leading researcher at the Institute of Engineering Cybernetics of National Academy of Sciences of Belarus. He has more than 80 scientific publications, including three books. Present scientific interests are concentrated in development of methods and algorithms for solving discrete optimization problems, complexity analysis. He is a winner of Belarus State Prize in the Field of Natural Sciences.

A. Tuzikov graduated from the Belarus State University in 1980. He received candidate and doctor of physics-mathematical sciences degree in 1985 and 2000, respectively. He joined the Institute of Engineering Cybernetics in 1980, where he is currently a principal researcher. His research interests are mathematical morphology, image analysis, and discrete applied mathematics.

Nuoseklus išdėstymas su sutvarkymo kriterijais, pirmenybės bei grupinių technologijų ribojimais

Adam JANIAK, Yakov SHAFRANSKY, Alexander TUZIKOV

Nagrinėjamos daugiakriterinio nuoseklaus išdėstymo problemos, kai kriterijai yra sutvarkomi pagal reikšmingumą, atsižvelgiant į pirmenybės bei grupinių technologijų ribojimus. Įvedama pranašumą generuojančio vektoriaus sąvoka ir nagrinėjama bendroji metodologija polinominiams nuoseklaus skaitmeninio išdėstymo algoritmams pagrįsti, apimanti visas žinomas polinomiškai sprendžiamas problemas. Pateikiama išsami žinomų nuoseklaus išdėstymo su sutvarkytais kriterijais rezultatų apžvalga.