BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 64, No. 2, 2016
DOI: 10.1515/bpasts-2016-0043

Discrete-continuous project scheduling with preemptable activities

R. RÓŻYCKI, G. WALIGÓRA*, and J. WĘGLARZ

Institute of Computing Science, Poznan University of Technology, 2 Piotrowo St., 60-965 Poznan, Poland

Abstract. In this paper, discrete-continuous project scheduling problems with preemptable activities are considered. In these problems, activities of a project simultaneously require discrete and continuous resources for their execution. The activities are preemptable, and the processing rate of each activity is a continuous, increasing function of the amount of a single continuous resource allotted to the activity at a time. The problem is to find a precedence- and discrete resource-feasible schedule and, simultaneously, continuous resource allocation that would minimize the project duration. Convex and concave processing rate functions are considered separately. We show that for convex functions the problem is simple, whereas for concave functions a special methodology has to be developed. We discuss the methodology for three cases of the problem: no discrete resource constraints, one discrete resource being a set of parallel, identical machines, and an arbitrary number of discrete resources. In each case we analyze separately independent and precedence-related activities. Some conclusions and directions for future research are given.

Key words: project scheduling, discrete-continuous, preemptable activities, project duration.

1. Introduction

In the classical project scheduling, only discrete resources are considered. Such resources can be assigned to activities of a project in amounts from a given finite set (i.e. in discrete numbers of units). However, in many practical situations continuous resources can also appear. These are resources which can be allotted to activities in arbitrary numbers from a given interval (i.e. in real numbers). Examples of continuous resources include power, fuel, gas, space, or even money. *Discrete-continuous scheduling problems* arise when jobs or activities simultaneously require discrete and continuous resources for their executions. Machine scheduling problems of this type have been discussed in [1–6]. In these problems, a set of machines is the only discrete resource. More recently, so-called power- (or energy-) aware scheduling problems have been considered, where a set of processors is a discrete resource, and power (energy) is a continuous resource [7–10]. Discrete-continuous project scheduling problems, where activities of a project are precedence-dependent and the number of discrete resources is arbitrary, have also been considered in a few papers. In [11–13] minimization of the project duration has been taken into account, whereas in [14–16] maximization of the net present value has been considered. In all those papers, it has been assumed that one continuous, renewable (or doubly-constrained) resource is available. The *processing rate vs. resource amount* model has been considered, in which the processing rate of a job (activity) is an increasing function of the amount of a continuous resource allotted to this job at a time. For this model some important properties of optimal schedules have been proved, leading to several analytical results for some classes of the processing rate functions.

Apart from the works concerning power-aware scheduling, all the papers mentioned above have dealt with nonpreemptable jobs (or activities). In particular, discrete-continuous project scheduling problems with preemptable activities have not been considered in any paper yet. The aim of this work is to present a methodology for solving such problems. To this end, different classes of the problems will be considered, as well as different processing rate functions of activities. We will consider independent and precedence-related activities, as well as three cases of constraints imposed on discrete resources: (i) the absence of discrete resource constraints, (ii) one discrete resource with unit requests, and (iii) an arbitrary number of discrete resources with integer resource requests of activities. In this paper we assume the project duration (or the schedule length) as the scheduling criterion.

This paper is organized as follows. In Section 2 we recall the most important results concerning the problem of allocating a continuous, renewable resource among independent jobs to minimize the schedule length in the absence of discrete resources. This section reports the basic results known for the continuous resource allocation problem obtained for two classes of the processing rate functions of jobs: convex and concave functions. Section 3 contains the general formulation of the problem under consideration. Section 4 is devoted to the case of convex processing rate functions. In this section we will show that the considered problem is trivial for each case analyzed in the paper. The main section of this work is Section 5, which deals with concave processing rate functions. We divide that section into three subsections. The first one – Sect. 5.1 – concerns the case with no discrete resources. In Sect. 5.2 we consider one discrete resource and unit resource requests of activities. In other words, it is a case of scheduling on parallel, identical machines. Finally, in Sect. 5.3 we deal with the most general case, where the number of discrete resources is arbitrary, and the resource requests of activities are arbitrary

*e-mail: grzegorz.waligora@cs.put.poznan.pl

and integer. As a result, we obtain a general discrete-continuous project scheduling problem with preemptable activities. In each of the three sections, 5.1 – 5.3, we also analyze two cases of precedence constraints: independent and precedence-related activities. Some conclusions and directions for future research are given in Section 6.

2. Continuous resource allocation

In this section we very briefly recall main theoretical results concerning the case when a continuous, renewable resource is the only limited resource, and discrete resources are not present. The results relate to independent jobs with equal ready times, the processing rate vs. resource amount job processing model, and the minimization of the schedule length.

We assume that one continuous, renewable resource is available. The availability of the resource over time is constant and, without loss of generality, we assume that its total available amount is equal to 1. The resource can be allotted to jobs in (arbitrary) amounts from the interval [0,1]. The amount (unknown in advance) of the continuous resource allotted to job i at time t is denoted by $u_i(t)$, and $\sum_{i=1}^{n} u_i(t) \le 1$ for any t. The resource amount determines the processing rate of job i, which is described by the following equation:

$$\dot{x}_i(t) = \frac{dx_i(t)}{dt} = f_i[u_i(t)], \quad x_i(0) = 0, \quad x_i(C_i) = w_i, \quad (1)$$

where:

- $x_i(t)$ is the state of job i at time t;
- f_i is the processing rate function of job i, continuous, increasing, and such that $f_i(0) = 0$;
- $u_i(t)$ is the continuous resource amount allotted to job i at time t;
- C_i is the completion time (unknown in advance) of job i;
- w_i is the size (final state) of job i.

State $x_i(t)$ of job i at time t is an objective measure of work related to the processing of job i up to time t. It may denote, e.g., the number of man-hours already spent on processing job i, the volume (in cubic meters) of a constructed building, the number of standard instructions in processing computer program i, etc.

In this case, the problem is to find an allocation of the continuous resource to jobs that minimizes the schedule length. The continuous resource allocation is defined by a piecewise continuous, nonnegative vector function, whose values are (continuous) resource allocations corresponding to C^*_{max} – the minimal value of C_{max}. Completion of job i requires that:

$$x_i(C_i) = \int_0^{C_i} f_i[u_i(t)]\, dt = w_i \quad (2)$$

For simplicity, we will denote $C_{max} = \max_{i=1,\ldots,n}\{C_i\}$ by T throughout the remainder of the paper. The following result, proved by Węglarz in [17], is fundamental for the continuous resource allocation problem:

Theorem 1. The minimum schedule length T^* as a function of sizes of jobs $\mathbf{w} = (w_1, w_2, \ldots, w_n)$ can always be given by:

$$T^*(\mathbf{w}) = \min\{T > 0: \mathbf{w}/T \in \mathrm{co}V\}$$

where co$V$ is the convex hull of V, and set V is defined as:

$$V = \left\{\mathbf{v}: v_i = f_i(u_i), u_i \ge 0, i = 1, 2, \ldots, n, \text{ and } \sum_{i=1}^{n} u_i \le 1\right\}$$

$T^*(\mathbf{w})$ is a convex function.

Two corollaries follow directly from Theorem 1 [17]:

Corollary 1. For convex processing rate functions of jobs, the schedule length is minimized by sequential processing of all jobs, each of them using the total available amount of the continuous resource.

Corollary 2. For concave functions f_i, $i = 1, 2, \ldots, n$, the schedule length is minimized by fully parallel processing of all jobs using the following resource amounts:

$$u_i^* = f_i^{-1}(w_i/T^*), i = 1, 2, \ldots, n\,, \quad (3)$$

where T^* is the unique positive root of equation

$$\sum_{i=1}^{n} f_i^{-1}(w_i/T) = 1. \quad (4)$$

Let us comment briefly on Corollaries 1 and 2.

Firstly, Corollary 1 holds, in fact, for all functions fulfilling the condition $f_i \le c_i u_i$, $c_i = f_i(1)$, $i = 1, 2, \ldots, n$, i.e. functions growing not faster than a linear function. In the sequel, the results presented for convex functions are true for all functions fulfilling the above condition. Secondly, Corollary 2 identifies very important cases, in which an optimal resource allocation can be found in an efficient way. Generally speaking, these are the cases when (4) can be solved analytically. From among them, the ones in which (4) is an algebraic equation of an order ≤ 4 are of special importance. This is, for example, the case of power processing rate functions of the form: $f_i(u_i) = c_i u_i^{1/\alpha_i}$, $\alpha_i \in \{1, 2, 3, 4\}$, $i = 1, 2, \ldots, n$. Using these functions we can model job processing rates in a variety of practical problems, e.g., those arising in multiprocessor scheduling with memory allocation [18].

Let us finally notice that in both Corollaries preemptability of jobs is of no importance. In Corollary 1 jobs are processed sequentially, each of them using the total available amount of the continuous resource. In Corollary 2 jobs are processed using constant resource amounts (given by Eq. (3)) from their starts to their completions. As a result, allowing job preemptions does not affect optimal schedules.

3. Problem formulation

In Section 2, properties of optimal schedules have been given, proved for the case in which a single continuous resource is the only limited resource, and independent jobs (activities) may be

performed in parallel. However, also discrete limited resources can appear, as well as precedence constraints between activities, which can restrict the execution order of the activities. Discrete-continuous project scheduling problems arise when precedence-related activities of a project simultaneously require discrete and continuous resources for their execution.

The *Preemptive Discrete-Continuous Resource-Constrained Project Scheduling Problem* (PDCRCPSP) considered in this paper is defined as follows. Given is a project consisting of n precedence-related, preemptable activities which require renewable resources of two types: discrete and continuous ones. We assume that R discrete resources are available, and r_{il}, $i = 1, 2, \ldots, n$; $l = 1, 2, \ldots, R$ is the (fixed) discrete resource request of activity A_i for resource l. The total number of units of discrete resource l available in each time period is R_l, $l = 1, 2, \ldots, R$. The activities are subject to finish-to-start precedence constraints with zero minimum time lags. The precedence constraints are represented by an Activity-On-Arc (AoA) digraph $G(Q, A)$ of q nodes, in which each of n arcs represents an activity. In digraph G, Q is the set of nodes, $|Q| = q$, and A is the set of arcs, $|A| = n$. The AoA representation and its consequences will be discussed in more detail in point 5.1.2. One continuous resource is available, and the processing rate of each activity at a time is defined by the amount of the continuous resource allotted to the activity, according to (1). Thus, each activity of the project is characterized by its processing demand, processing rate function, discrete resource requests, and precedence relations to other activities. It is assumed that all activities and resources are available from the start of the project. The problem is to find a precedence- and discrete resource-feasible schedule and, simultaneously, a continuous resource allocation that minimize the schedule length T. Following the classification given in [19], the notation of the PDCRCPSP is $m,1|pmtn, cpm, cont|C_{max}$. All the parameters of the PDCRCPSP are summarized in Table 1.

Table 1
Parameters of the PDCRCPSP

Symbol	Definition
$G(Q, A)$	AoA graph representing project
$q = \|Q\|$	number of nodes in digraph G
$n = \|A\|$	number of arcs in digraph G, i.e. number of activities
R	number of discrete resources
R_l	number of available units of discrete resource l
r_{il}	request for discrete resource l by activity A_i
f_i	processing rate function of activity A_i
w_i	size of activity A_i
S_i	starting time of activity A_i
C_i	completion time of activity A_i

It has been shown in previous works that the methodology for solving the discrete-continuous resource-constrained project scheduling problem (DCRCPSP) with nonpreemptable activities critically depends on the form of the processing rate functions. Based on Corollary 1, it was proved in [15] that for the DCRCPSP with convex processing rate functions, in an optimal schedule activities are processed sequentially, each of them using the total available amount of the continuous resource. On the other hand, following Corollary 2, it was also shown in [15] that for the DCRCPSP with concave functions the schedule length is optimized by fully parallel precedence- and discrete resource-feasible execution of all activities. Consequently, we will also distinguish between these two classes of functions for the preemptive DCRCPSP, which is described in the next two sections.

4. Convex processing rate functions

As mentioned in Section 3, it was proved that for the DCRCPSP with convex processing rate functions, the schedule length is minimized by sequential processing of all activities, where each activity uses the total available amount of the continuous resource. Obviously, in the PDCRCPSP activities may be preempted, however, in a sequential schedule it will not affect the schedule length unless there are idle times between the activities. Since preemptions cannot improve the schedule, it is reasonable not to take them into account. Thus, the same sequential schedule with full utilization of the continuous resource is optimal for the PDCRCPSP as it is for the DCRCPSP with nonpreemptable activities.

Let us now stress that since a sequential schedule is considered, the appearance of discrete resources is of no importance. The same sequential schedule leads to optimum in all the three cases considered in this paper: (i) in the absence of discrete resources, (ii) one discrete resource being a set of identical machines (in this case all activities are scheduled on one machine, whereas the other machines remain idle), and (iii) an arbitrary number of discrete resources. However, precedence relations between activities have to be taken into account in each case. Summarizing, we can state that:

- for the PDCRCPSP with convex processing rate functions and independent activities, the schedule length is minimized by sequential execution of all activities in an arbitrary order, in which each activity uses the total available amount of the continuous resource, and
- for the PDCRCPSP with convex processing rate functions and precedence-related activities, the schedule length is minimized by sequential execution of all activities in any precedence-feasible order, in which each activity uses the total available amount of the continuous resource.

Now, let us notice that if the continuous resource amount allotted to activity A_i does not change over the whole time of its execution, i.e. $u_i(t) = u_i$ for every t, we can rewrite (1) at the moment of completion of activity A_i as:

$$\frac{x_i(C_i)}{p_i} = f_i(u_i), \tag{5}$$

where p_i is the processing time of activity A_i, and, in consequence:

$$p_i = \frac{w_i}{f_i(u_i)}. \tag{6}$$

Since in this case each activity uses the total available amount of the continuous resource, i.e. $u_i = 1$ for each activity A_i, $i = 1, 2, \ldots, n$, thus (6) can be rewritten as:

$$p_i = \frac{w_i}{f_i(1)} \tag{7}$$

and the length of the optimal schedule can easily be calculated as:

$$T^* = \sum_{i=1}^{n} \frac{w_i}{f_i(1)} \tag{8}$$

Thus, for convex processing rate functions of activities, the PDCRCPSP is trivial since any schedule, in which activities are processed one after another in any precedence-feasible order, each of them using the total amount of the continuous resource, is optimal. The optimum schedule length can be easily calculated from (8). Also, as mentioned before, this result is independent of the occurrence of discrete resources in the PDCRCPSP.

Figure 1 shows two examples of optimal schedule for different problem instances with $n = 4$ under convex processing rate functions of activities. The actual problem parameters are of no importance for the figure, it only shows the important features of the optimal schedule: sequencing and full usage of the continuous resource (equal to 1).

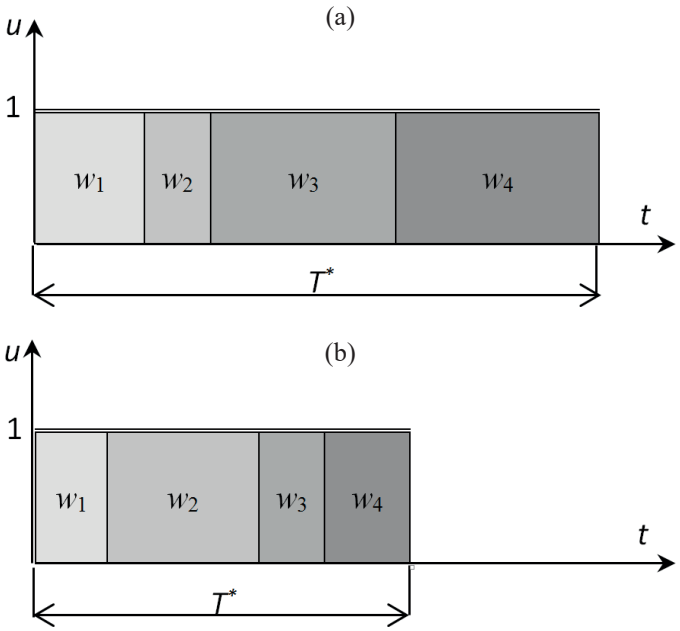


Fig. 1. Two exemplary optimal schedules for different problem instances with $n = 4$

5. Concave processing rate functions

As mentioned in Section 3, it was proved that for the DCRCPSP with concave functions, the schedule length is optimized by fully parallel precedence- and discrete resource-feasible execution of all activities. Since a parallel execution can be restricted by both discrete resource constraints and precedence relations between activities, these factors have to be analyzed separately.

In the next three sections, we will consider the following cases of the discrete resource constraints:

- no discrete resource constraints – Sect. 5.1
- one discrete resource being a set of identical machines – Sect. 5.2
- an arbitrary number of discrete resources – Sect. 5.3.

In each case we also have to distinguish between independent and precedence-related activities.

5.1 No discrete resource constraints. In this section we consider a special case of the PDCRCPSP in which there are no discrete resource constraints, i.e. the continuous resource is the only limited resource. Still, we have to analyze the cases of independent and precedence-related activities.

5.1.1 Independent activities. For the case of independent activities the problem is simple because, neither discrete resource constraints nor precedence relations between activities restrict a parallel execution of the activities. Thus, the schedule length is minimized by fully parallel execution of all activities, each of them using the continuous resource amount given by (3). The optimum schedule length can be immediately calculated from Eq. (4).

5.1.2 Precedence-related activities. For the case of dependent activities the problem becomes more complicated, as the precedence relations between activities restrict a parallel execution. At a given moment only those activities may be executed in parallel, all predecessors of which have already been completed.

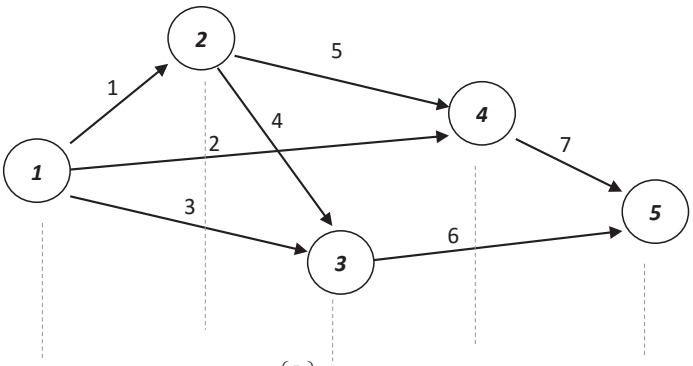
As mentioned in Section 3, in this work precedence constraints between activities are represented by an Activity-on-Arc digraph G in which q nodes represent events and n arcs represent activities. We assume that G is connected, acyclic, has one initial and one terminal node, and each pair of nodes is connected by one arc at most. Such a precedence constraints representation is always possible, although it sometime requires inserting so-called *dummy* nodes (events) and/or activities (arcs).

Usually, there can be many different activity execution orders and, in consequence, different feasible schedules, associated with a particular graph G. If there is only one activity execution order possible due to the existing precedence relations, graph G representing such relations is then called *uniconnected activity network* (*UAN*). It is a case which makes the search for optimal solution much easier, as we will show in the sequel. However, in general, we have to assume that graph G does not have the *UAN* property. In such a case, a methodology for solving the considered problem based on the assumed ordering O of nodes in graph G was discussed in [20]. In the proposed methodology the ordering of nodes is decisive for the unique activity execution order in the resulting feasible schedule. For ordering O we will assume that the only initial node has number 1, the only terminal node has number q, and each arc starts in a node with a smaller number and ends in a node with a larger number. Obviously, if graph G is not a *UAN*, there are many different orderings of nodes satisfying the above assumptions.

For a given graph G the activity execution order following from an assumed node ordering O can be represented by a sequence $S_O = [Q_1, Q_2, \ldots, Q_{q-1}]$ of so-called *main sets*. The main set Q_k, $k = 1, 2, \ldots, q-1$ contains activities that can be executed

in parallel between events represented by nodes k and k+1 in graph G, and corresponds to the k-th fragment of the feasible schedule following from ordering O.

An example of graph G with a fixed node ordering and the corresponding sequence of main sets is presented in Fig. 2.



$$Q_1 = \begin{Bmatrix} 1, \\ 2, \\ 3 \end{Bmatrix} \quad Q_2 = \begin{Bmatrix} 5, \\ 4, \\ 2, \\ 3 \end{Bmatrix} \quad Q_3 = \begin{Bmatrix} 5, \\ 2, \\ 6 \end{Bmatrix} \quad Q_4 = \begin{Bmatrix} 7, \\ 6 \end{Bmatrix}$$

$$S_Q = \left[\begin{Bmatrix} 1, \\ 2, \\ 3 \end{Bmatrix}, \begin{Bmatrix} 2, \\ 3, \\ 4, \\ 5 \end{Bmatrix}, \begin{Bmatrix} 2, \\ 5, \\ 6 \end{Bmatrix}, \begin{Bmatrix} 6, \\ 7 \end{Bmatrix} \right]$$

Fig. 2. Example of graph G with a fixed node ordering and the corresponding sequence of main sets S_Q

If we assume that for main set Q_k the sizes (or parts of sizes) of activities occurring in Q_k are known, then the length of the fragment of the schedule corresponding to Q_k can be calculated by using Corollary 2. Let us call such a division of activity sizes among main sets by *size division*. Thus, the methodology for solving the considered problem consists in formulating a mathematical programming (MP) problem (in general, a nonlinear one – NLP), in which such a size division is searched that – after applying Corollary 2 to each main set – minimizes the length of the entire schedule. Let us denote by w_{ik} the part of the size of activity A_i assigned to the k-th fragment of the schedule (i.e. corresponding to main set Q_k), and by T_k the length of the fragment as a function of vector $\mathbf{w}_k = \{w_{ik}\}_{A_i \in Q_k}$. Moreover, let K_i denote the set of indices of those main sets which contain activity A_i. The following MP problem finds the optimal schedule length for a given node ordering O in graph G:

Problem P1

minimize

$$T = \sum_{k=1}^{q-1} T_k^*(\mathbf{w}_k) \tag{9}$$

subject to

$$\sum_{k \in K_i} w_{ik} = w_i, \quad i = 1, 2, \ldots, n \tag{10}$$

$$w_{ik} \geq 0, i = 1, 2, \ldots, n; k \in K_i \tag{11}$$

where $T_k^*(\mathbf{w}_k)$, k = 1, 2, ..., q–1 is the unique positive root of the equation:

$$\sum_{A_i \in Q_k} f_i^{-1}(w_{ik}/T_k) = 1 \tag{12}$$

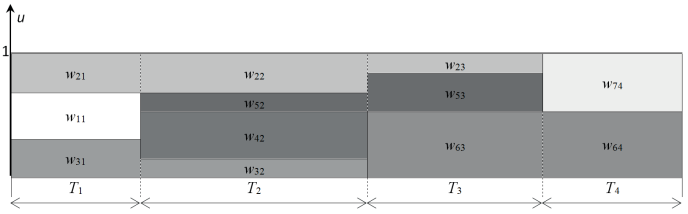
For concave processing rate functions of activities, Problem P1 is an NLP (convex) problem. The schedule length T is calculated in (9) as the sum of the lengths of all the fragments of the schedule. Constraints (10) correspond to the condition of executing each activity in its full size, whereas constraints (11) ensure that the w_{ik}'s are nonnegative. Condition (12) allows to calculate the minimal length of the k-th fragment following from an optimal continuous resource allocation. Note that the equation in (12) is an adaptation of (4) to a single main set Q_k. It can be solved analytically for some important cases, as discussed in Sect. 2. Table 2 shows the notation used for Problem P1.

Table 2
Notation for Problem P1

Symbol	Definition
Q_k	k-th main set
T_k	length of the k-th fragment of schedule (corresponding to Q_k)
w_{ik}	part of the size of activity A_i assigned to the k-th fragment of the schedule
K_i	set of indices of main sets containing activity A_i

Figure 3 presents an example of a feasible schedule, following from an assumed division of activity sizes among main sets under the fixed node ordering presented in Fig. 2.

$$S_Q = \left[\begin{Bmatrix} 1, \\ 2, \\ 3 \end{Bmatrix}, \begin{Bmatrix} 2, \\ 3, \\ 4, \\ 5 \end{Bmatrix}, \begin{Bmatrix} 2, \\ 5, \\ 6 \end{Bmatrix}, \begin{Bmatrix} 6, \\ 7 \end{Bmatrix} \right]$$



$$w_1 = w_{11}$$
$$w_2 = w_{21} + w_{22} + w_{23}$$
$$w_3 = w_{31} + w_{32}$$
$$w_4 = w_{42}$$
$$w_5 = w_{52} + w_{53}$$
$$w_6 = w_{63} + w_{64}$$
$$w_7 = w_{74}$$
$$T = T_1 + T_2 + T_3 + T_4$$

Fig. 3. Example of a feasible schedule for node ordering presented in Fig. 2

Let us stress again that the schedule generated as a result of solving Problem P1 is of the minimum length only for a given node ordering O. If this is the unique ordering for graph G (G is a *UAN*), the schedule is optimal. Otherwise, in order to find an optimal schedule it is necessary to enumerate all feasible orderings of nodes and solve Problem P1 for each of them. However, in general, the number of all feasible node orderings grows exponentially with the number of nodes, and therefore, the full enumeration approach can be justified only for small problem instances.

Thus, in practice, a reasonable approach could be to apply local search metaheuristics searching over the set of all feasible orderings. Obviously, in such a case, the set of all feasible orderings is only partially examined by a chosen metaheuristic approach. Alternatively, constructive heuristics may be designed, using some rules to generate a single suboptimal node ordering, and building a feasible schedule on its basis, as presented in [21].

5.2 One discrete resource – a set of identical machines. In this section we consider a case of the PDCRCPSP with one discrete resource being a set of parallel, identical machines. As previously, we analyze independent and precedence-related activities.

At this point it is worth noticing that the limited amount of the continuous resource never constrains the possibility of parallel execution of any activities, at least it affects the rate of their processing. In the assumed activity model (compare (1) and (2)) limited discrete resources for which the activities compete do not affect the processing times of activities; however, they can constrain the possibility of the activities' parallel execution.

5.2.1 Independent activities. For independent activities the (nonexistent) precedence constraints do not restrict a parallel activity execution. However, since a parallel assignment of machines to activities can be restricted by the number of machines, we have to distinguish two cases: $n \leq m$ and $n > m$. In the first case, the number of activities does not exceed the number of machines, and therefore, all activities can be performed in parallel. It is not possible in the second case, where only m out of n activities can be scheduled in the first step, and $n–m$ activities have to initially wait for machines. Since these two cases result in two completely different methodologies, we will discuss them in separate sections.

5.2.1.1 The case of $n \leq m$. As mentioned above, in this case all activities can be performed in parallel, and the set of machines does not constitute any restriction. In consequence, the result presented in Corollary 2 can be implicitly applied, as if there were no discrete resources at all. Obviously, analogically as in Sect. 2, preemptions cannot improve the schedule and will not be taken into account. The optimum schedule length can be calculated directly from (4). Two examples of optimal schedules for the considered case are presented in Fig. 4. As in Fig. 1, the actual problem parameters do not matter for the figure, it only shows the important features of the optimal schedule: full usage of the continuous resource (equal to 1), and parallel execution in which all activities end at the same time.

5.2.1.2 The case of $n > m$. In this case the number of machines restricts parallel assignments of machines to activities, and a special methodology has to be developed. For the sake of methodology, a structure representing assignment of machines to activities has to be used, in which combinations of activities executed in parallel in successive intervals of a schedule are defined. Since activities are preemptable, in general it is sufficient to consider a so-called *maximal sequence Smax* composed of all $s = \binom{n}{m}$ m-element combinations of activities [20]. Such a sequence exhausts all possible assignments of m (identical) machines to n activities, and therefore guarantees finding an optimal assignment. Each feasible schedule can be generated by using the maximal sequence.

We will represent sequences of combinations in a form of vectors since, in a general case, the position of a combination in a sequence is important. For instance, the maximal sequence *Smax* for an exemplary problem with $n = 4$ and $m = 3$ can be represented as:

$$Smax = [\{1, 2, 3\}, \{1, 2, 4\}, \{1, 3, 4\}, \{2, 3, 4\}]$$

However, in the considered problem, since preemptable activities are scheduled on identical machines, the order of com-

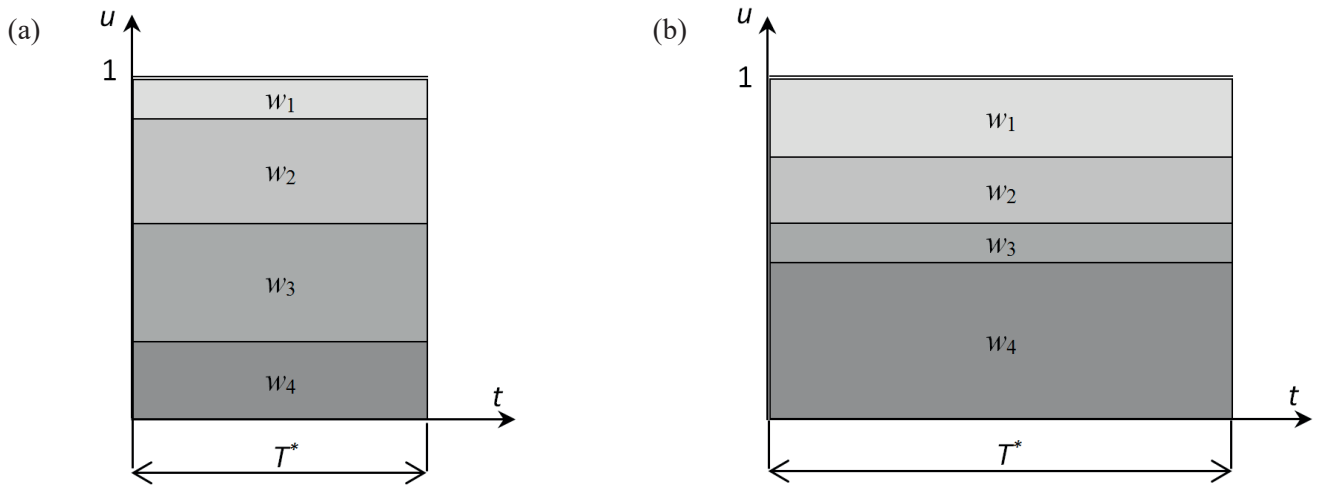


Fig. 4. Two exemplary optimal schedules for different problem instances with $n = 4$ and $m \geq 4$

binations in the sequence, as well as the order of activities in a particular combination, do not matter.

Now, for the maximal sequence we look for a division of the sizes of activities (i.e. a size division) among combinations of the sequence that leads to optimum. More precisely, size w_i of each activity A_i, $i = 1, 2, \ldots, n$ has to be divided into parts $w_{ik} \geq 0$ (unknown in advance) corresponding to particular time intervals (combinations), i.e. w_{ik} is a part of size of activity A_i processed in the interval associated with combination Z_k, $k = 1, 2, \ldots, s$. Note that approaches based on searching for an optimal division of sizes (or processing times) of jobs in a schedule are often used to solve classical problems of scheduling preemptable jobs [21].

Next, an NLP problem can be formulated finding an optimal size division for the maximal sequence *Smax*, i.e. a division that leads to a schedule of the minimal length from among all schedules generated by *Smax*. In the problem, the sum of the minimum-length intervals generated by consecutive combinations in *Smax*, as functions of the vector $\mathbf{w}_k = \{w_{ik}\}_{i \in z_k}$, is minimized subject to the constraints that each activity has to be completed. Let $T_k^*(\mathbf{w}_k)$ be the minimal length of the fragment of the schedule generated by $Z_k \in Smax$, and let K_i be the set of all indices of Z_k's such that $A_i \in Z_k$. The following NLP problem finds an optimal size division (and, in consequence, an optimal continuous resource allocation) for the maximal sequence *Smax*:

Problem P2

minimize

$$T = \sum_{k=1}^{s} T_k^*(\mathbf{w}_k) \tag{13}$$

subject to

$$\sum_{k \in K_i} w_{ik} = w_i\,, \quad i = 1, 2, \ldots, n \tag{14}$$

$$w_{ik} \geq 0\,, i = 1, 2, \ldots, n; k \in K_i \tag{15}$$

where $T_k^*(\mathbf{w}_k)$ is the unique positive root of the equation:

$$\sum_{A_i \in Z_k} f_i^{-1}(w_{ik}/T_k) = 1 \tag{16}$$

The schedule length T is calculated in (13) as the sum of the lengths of all the intervals of the schedule. The intervals correspond to combinations Z_k, $k = 1, 2\ldots, s$, and in this case $s = \binom{n}{m}$. Constraints (14) and (15) are identical to (10) and (11). Condition (16) allows to calculate the minimal length of the k-th interval following from an optimal continuous resource allocation. The equation in (16) is an adaptation of (4) to a single combination Z_k.

Figure 5 shows maximal sequence *Smax* for $n = 4$ and $m = 3$ and two exemplary optimal schedules for different problem instances.

5.2.2 Precedence-related activities. In this case, in order to find optimal schedules, an approach being an extension of the method proposed in point 5.1.2 can be used. Obviously, this time not only precedence- but also resource constraints have to be taken into consideration, following from the limited number of m machines.

As shown in 5.1.2, for a given node ordering O in digraph G an activity execution order feasible with respect to precedence constraints can be represented by a sequence S_Q of main sets. However, in general, main sets can now contain activities which cannot be executed in parallel (which is desirable for concave processing rate functions) because of the existence of additional resource constraints. Consequently, for each main set a vector of combinations of activities can be constructed. A single activity combination is a subset of the considered main set containing these activities that can be executed in parallel, with respect to the resource constraints (in this case – a limited number of

$$Smax = [\{1, 2, 3\}, \{1, 2, 4\}, \{1, 3, 4\}, \{2, 3, 4\}]$$

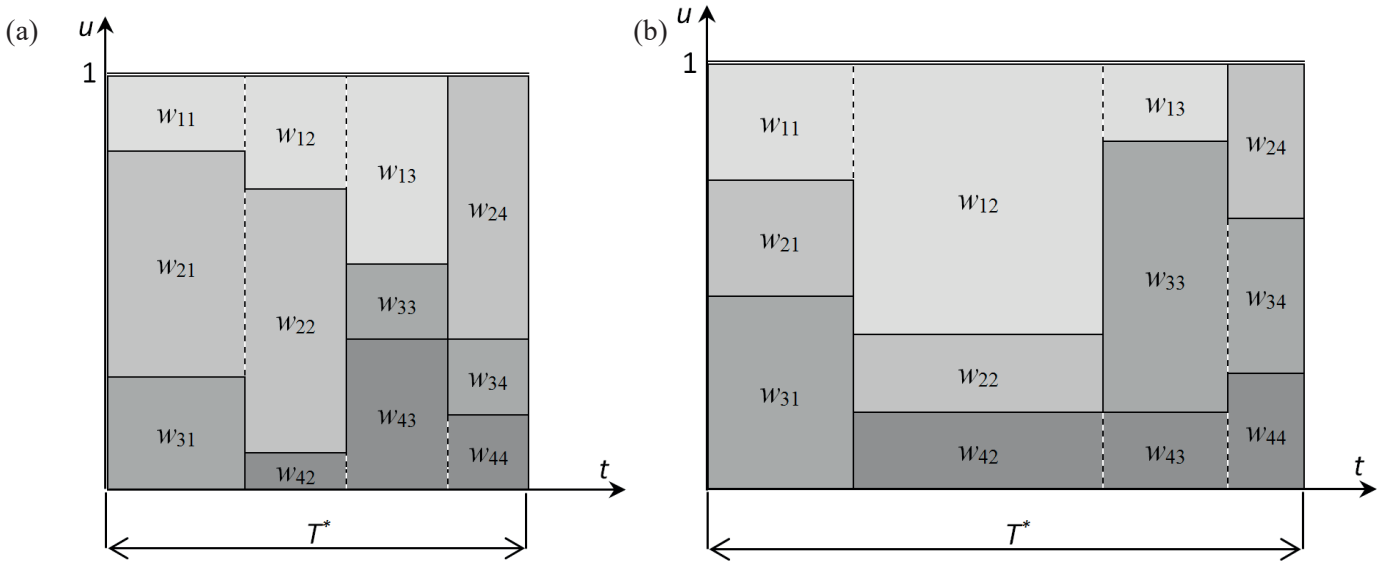


Fig. 5. Maximal sequence *Smax* for $n = 4$ and $m = 3$ and two exemplary optimal schedules for different problem instances

identical machines). Notice, that since such a combination is constructed as a subset of the main set, precedence constraints are already satisfied. As discussed in [20], in the considered case of identical machines and concave processing rate functions, a vector of combinations for a given main set Q_k consists of z_k elements, where:

$$z_k = \begin{cases} 1 & if \ |Q_k| \le m \\ \binom{|Q_k|}{m} & if \ |Q_k| > m \end{cases} \tag{17}$$

In the first case, the main set contains no more that m elements (activities), thus, it is resource-feasible alone. In the second case, the vector of combinations contains all possible m-element combinations of activities from main set Q_k. Let Z_{kl} denote the l-th activity combination generated from main set Q_k.

In consequence, in order to represent a schedule feasible with respect to both precedence and resource constraints, we may use a sequence S_Z of vectors of combinations of the form:

$$S_Z = \Big[[Z_{11}, Z_{12}, \dots, Z_{1z_1}], [Z_{21}, Z_{22}, \dots, Z_{2z_2}], \dots, \\ [Z_{q-1,1}, Z_{q-1,2}, \dots, Z_{q-1,z_{q-1}}]\Big]$$

In this case, a feasible schedule will be considered as consisting of fragments corresponding to successive combinations in S_Z. In order to find a schedule of the minimal length for a given node ordering O, it is sufficient to formulate and solve an MP problem in which we look for an optimal division of sizes of activities among the combinations in which they occur. For defined sizes of activities (or their parts) in a combination we can find, using Corollary 2, such an allocation of the continuous resource that minimizes the length of the fragment of the schedule corresponding to this combination. To formulate the appropriate MP problem, we have to slightly modify the notation used in Problem P1. Let T_{kl} denote the length of the fragment of the schedule corresponding to combination Z_{kl}, as a function of vector $\mathbf{w}_k = \{w_{ik}\}_{A_i \in Z_{kl}}$, whereas K_{ik} is the set of indices of combinations in the vector of combinations generated from main set Q_k and containing activity A_i. If w_{ikl} denotes the part of the size of activity A_i, which is to be executed in combination Z_{kl}, then the MP problem can be formulated as:

Problem P3

minimize

$$T = \sum_{k=1}^{q-1} \sum_{l=1}^{z_k} T_{kl}^*(\mathbf{w}_{kl}) \tag{18}$$

subject to

$$\sum_{k=1}^{q-1} \sum_{l \in K_{ik}} w_{ikl} = w_i, \quad i = 1, 2, \dots, n \tag{19}$$

$$w_{ikl} \ge 0, i = 1, 2, \dots, n; k = 1, 2, \dots, q-1; l \in K_{ik} \tag{20}$$

where $T_{kl}^*(\mathbf{w}_{kl})$, $k = 1, 2, \dots, q-1$; $l = 1, 2, \dots, z_k$ is the unique positive root of the equation:

$$\sum_{A_i \in Z_{kl}} f_i^{-1}\left(\frac{w_{ikl}}{T_{kl}}\right) = 1 \tag{21}$$

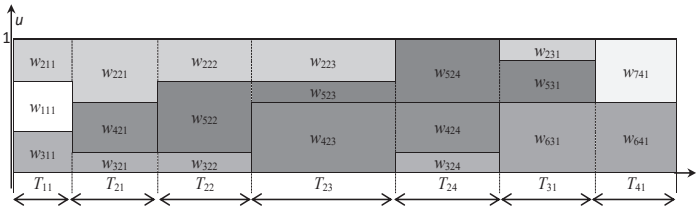
As previously, for concave processing rate functions of activities, Problem P3 is an NLP (convex) problem. The schedule length T is calculated in (18) as the sum of the lengths of all the fragments of the schedule. Constraints (19) and (20) are analogical as (10) and (11) in Problem P1. Condition (21) allows to calculate the minimal length of the k-th fragment corresponding to combination Z_{kl}. The equation in (21) is an adaptation of (4) to a single combination Z_{kl}. Table 3 shows the notation used for Problem P3.

Table 3
Notation for Problem P3

Symbol	Definition
Z_{kl}	l-th activity combination generated from main set Q_k
z_k	number of combinations constructed out of Q_k
T_{kl}	length of the fragment of the schedule corresponding to Z_{kl}
w_{ikl}	part of size of activity A_i assigned to Z_{kl}
K_{ik}	set of indices of combinations generated from Q_k and containing A_i

Figure 6 presents an exemplary sequence of vectors of combinations and the corresponding precedence- and resource-feasible schedule.

$$S_Z = \left[\left[\begin{Bmatrix}1,\\2,\\3\end{Bmatrix}\right], \left[\begin{Bmatrix}2,\\3,\\4\end{Bmatrix}, \begin{Bmatrix}2,\\3,\\5\end{Bmatrix}, \begin{Bmatrix}2,\\4,\\5\end{Bmatrix}, \begin{Bmatrix}3,\\4,\\5\end{Bmatrix}\right], \left[\begin{Bmatrix}2,\\5,\\6\end{Bmatrix}\right], \left[\begin{Bmatrix}6,\\7\end{Bmatrix}\right]\right]$$



$$w_1 = w_{111}$$
$$w_2 = w_{211} + w_{221} + w_{222} + w_{223} + w_{231}$$
$$w_3 = w_{311} + w_{321} + w_{322} + w_{324}$$
$$w_4 = w_{421} + w_{423} + w_{424}$$
$$w_5 = w_{522} + w_{523} + w_{524} + w_{531}$$
$$w_6 = w_{631} + w_{641}$$
$$w_7 = w_{741}$$
$$T = T_{11} + T_{21} + T_{22} + T_{23} + T_{24} + T_{31} + T_{41}$$

Fig. 6. Example of a feasible schedule for $m = 3$ and node ordering presented in Fig. 2

It is easy to notice that for some problem instances the same combinations of activities can occur in several vectors of sequence S_Z. It can be proved that a schedule constructed by solving Problem P3 will be feasible and not inferior with respect to the considered criterion, if only one such combination is left in sequence S_Z (in any vector of combinations). This is, obviously, justified from the computational point of view, since it allows to eliminate a number of redundant variables in Problem P3. To this end, for instance, a modified version of the algorithm presented in [9] can be used. The algorithm generates a sequence of unique combinations of activities for the case of scheduling dependent jobs (activities) on parallel machines, as considered in this point.

Finally, let us stress that as discussed in point 5.1.2, in order to find an optimal schedule, an ordering of nodes in graph G leading to optimum has to be found first.

5.3 Arbitrary discrete resources. In this section we consider a case of the PDCRCPSP with an arbitrary number of discrete resources.

As previously, we separately analyze independent and precedence-related activities.

5.3.1 Independent activities. In this case activities are not precedence-related, however the existence of limited discrete resources restricts a potential parallel execution of the activities. Additionally, the important difference to the case considered in point 5.2.1 is that now the number of activities processed in parallel (equal to m for identical machines) is not known in advance. That number will depend on the combination of activities processed simultaneously, since the number of discrete resources is arbitrary and activities have different resource requests. In consequence, all resource-feasible combinations of activities have to be generated, analogically as it was discussed in point 5.2.1 for the set of identical machines. Since parallel execution is desired for concave processing rate functions, only *maximal combinations* have to be taken into account. A maximal combination has such a feature, that each its extension obtained by adding any other activity becomes resource-infeasible.

Efficient generation of such maximal resource-feasible combinations is not a trivial task. To this end, for instance, a procedure called GEN, proposed in [22] and originally developed for a more general case (see point 5.3.2), can be applied. As the procedure allows the existence of precedence constraints between activities, successive maximal combinations are built on a basis of the sequence of main sets following from an assumed node ordering in the precedence-relation graph. The absence of precedence constraints can be interpreted in this context, as a case of a single main set containing all activities.

Procedure GEN efficiently finds a sequence of all maximal combinations due to preliminary ordering of activities in the main set, according to their decreasing resource requests with respect to the given discrete resource type. The order of maximal combinations in the constructed sequence is of no importance.

Next, Problem P2 (see point 5.2.1.2) can be formulated, in which s is the number of all maximal combinations. The solution of Problem P2 defines an optimal size division among the maximal combinations and, consequently, an optimal schedule length for the considered case.

5.3.2 Precedence-related activities. The case discussed in this point is the most general one of all considered in this work. A parallel execution of activities may be now restricted by both the precedence and resource constrains with respect to any number and types of discrete resources. In order to find an optimal schedule, similarly as in the special cases of the problem described earlier, an MP problem can be formulated and solved. In this case, the MP problem will be analogical to Problem P3 (see point 5.2.2), where the only difference concerns the way of generating the sequence of vectors of combinations S_Z.

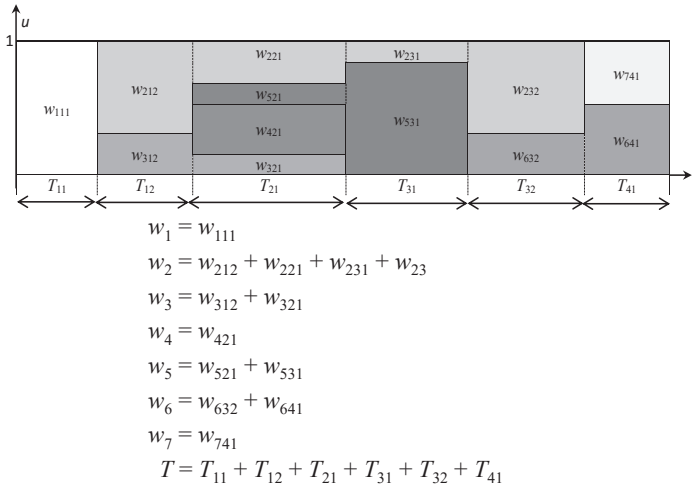
As previously, the initial point for constructing sequence S_Z is the sequence S_Q of main sets. Let us remind that it depends on the assumed node ordering O in the precedence-relation graph G. It is natural to assume that to each main set from S_Q, there is one corresponding vector of maximal combinations in S_Z. As mentioned in point 5.2.2, it is desirable from the computational point of view that unique maximal combinations should occur in successive vectors of sequence S_Z. In order to efficiently generate sequences S_Z of that property, procedure GEN [22] can be adopted again. The procedure generates resource-feasible subsets corresponding to maximal combinations for successive main sets in S_Q. With respect to the desired property, the key feature is dividing each main set into "new" and "old" activities. In a main set Q_k, $k = 1, 2, \ldots, q–1$, "new" are those activities that do not belong to Q_l, $l < k$. All the other activities in Q_k are "old". For a given main set, resource-feasible subsets are generated on a basis of so-called *primary subsets,* in which the division into "new" and "old" activities is maintained. In consequence, to each main set Q_k, there is a corresponding vector of z_k such maximal combinations that do not occur in any vector corresponding to main sets Q_l, $l < k$.

For more details concerning the generation of the maximal resource-feasible subsets, see [22].

Figure 7 shows an exemplary sequence of vectors of combinations for the considered case of the problem, as well as the corresponding precedence- and resource-feasible schedule. It is assumed that there are two discrete resources ($R = 2$), available in 5 ($R_1 = 5$) and 10 ($R_2 = 10$) units, respectively, and that the activity resource requests are given in the table below:

i	r_{i1}	r_{i2}
1	4	3
2	2	2
3	2	4
4	0	1
5	1	3
6	1	8
7	3	2

$$S_Z = \left[\left[\{1\}, \begin{Bmatrix} 2, \\ 3 \end{Bmatrix} \right], \left[\begin{Bmatrix} 2, \\ 3, \\ 4, \\ 5 \end{Bmatrix} \right], \left[\begin{Bmatrix} 2, \\ 5 \end{Bmatrix}, \begin{Bmatrix} 2, \\ 6 \end{Bmatrix} \right], \left[\begin{Bmatrix} 6, \\ 7 \end{Bmatrix} \right] \right]$$



$$w_1 = w_{111}$$
$$w_2 = w_{212} + w_{221} + w_{231} + w_{23}$$
$$w_3 = w_{312} + w_{321}$$
$$w_4 = w_{421}$$
$$w_5 = w_{521} + w_{531}$$
$$w_6 = w_{632} + w_{641}$$
$$w_7 = w_{741}$$
$$T = T_{11} + T_{12} + T_{21} + T_{31} + T_{32} + T_{41}$$

Fig. 7. Example of a feasible schedule for $R = 2$, $R_1 = 5$, $R_2 = 10$, and node ordering presented in Fig. 2

Let us finally discuss in short the complexity of the presented approaches.

For convex processing rate functions, i.e. for the case described in Sect. 4, the problem is trivial, since only a precedence-feasible sequence of activities has to be found. The optimal project duration can be then calculated analytically.

For concave processing rate functions, i.e. for the cases described in Sect. 5, the problem becomes more complex. From among all the cases, only problems discussed in points 5.1.1 and 5.2.1.1 are computationally easy. In each of the other cases an NLP problem has to be solved in order to find an optimal continuous resource allocation, i.e. to find a solution of the continuous part of the problem. Note that because of the existence of the continuous part, the PDCRCPSP problem, as a whole, cannot be classified by the computational complexity theory, i.e. in terms of the P and NP classes, since it is not a combinatorial optimization problem. However, it is worth stressing that for dependent activities the complexity of the discrete part of the problem is already exponential, since it requires finding an optimal ordering of nodes in the precedence-relation graph (see point 5.1.2).

6. Conclusions

In this work discrete-continuous project scheduling problems with preemptable activities have been considered. The Preemptive Discrete-Continuous Resource-Constrained Project Scheduling Problem (PDCRCPSP) has been defined in the most general form to minimize the project duration. Convex and concave processing rate functions of activities have been analyzed for the cases of no discrete resource constraints, one discrete resource being a set of parallel, identical machines, and an arbitrary number of discrete resources. Each time independent and precedence-related activities have been separately considered.

We have shown that for convex functions all cases of the problem are trivial, since discrete resources do not affect the optimal schedule. In order to minimize the schedule length, the activities have to be processed sequentially, each of them using the total available amount of the continuous resource. Obviously, for dependent activities precedence constraints have to be satisfied in a sequential schedule. For concave functions the defined problem is simple only in two cases with independent activities: (i) with no discrete resource constraints, and (ii) with the number of machines not less than the number of activities. In all the other cases, an NLP problem has to be solved to find a minimal-length schedule. Moreover, for precedence-related activities a special methodology based on main sets has to be applied.

Future research should be mainly conducted towards heuristic approaches to the computationally hard problems. It can be done at two levels: the discrete and the continuous part of the problem. For the discrete part, heuristic (and metaheuristic) approaches can be developed to construct (or search for) near-optimal sequences of vectors of combinations. For the continuous part, heuristic approaches to allocate the continuous resource may be designed in order to avoid the necessity of solving the nonlinear mathematical programming problems.

Acknowledgments. This research has been supported by the Polish National Science Centre under projects 2013/08/A/ST6/00296 (R.R., J.W.) and 2013/11/B/ST6/00970 (G.W.).

References

[1] J. Józefowska, M. Mika, R. Różycki, G. Waligóra, and J. Węglarz, "Local search metaheuristics for discrete-continuous scheduling problems", *European Journal of Operational Research* 107 (2), 354–370 (1998).

[2] J. Józefowska, M. Mika, R. Różycki, G. Waligóra, and J. Węglarz, "Discrete-continuous scheduling to minimize the makespan with power processing rates of jobs", *Discrete Applied Mathematics* 94 (1–3), 263–285 (1999).

[3] J. Józefowska, M. Mika, R. Różycki, G. Waligóra, and J. Węglarz, "A heuristic approach to allocating the continuous resource in discrete-continuous scheduling problems to minimize the makespan", *Journal of Scheduling* 5 (6), 487–499 (2002).

[4] J. Józefowska, G. Waligóra, and J. Węglarz, "Tabu list management methods for a discrete-continuous scheduling problem", *European Journal of Operational Research* 137 (2), 288–302 (2002).

[5] G. Waligóra, "Tabu search for discrete-continuous scheduling problems with heuristic continuous resource allocation", *European Journal of Operational Research* 193 (3), 849–856 (2009).

[6] M.S. Barketau, M.Y. Kovalyov, J. Węglarz, and M. Machowiak, "Scheduling arbitrary number of malleable tasks on multiprocessor systems", *Bull. Pol. Ac.: Tech.* 62 (2), 255–261 (2014).

[7] R. Różycki and J. Węglarz, "On job models in power management problems", *Bull. Pol. Ac.: Tech.* 57 (2), 147–151 (2009).

[8] R. Różycki and J. Węglarz, "Power-aware scheduling of preemptable jobs on identical parallel processors to meet deadlines", *European Journal of Operational Research* 218 (1), 68–75 (2012).

[9] R. Różycki and J. Węglarz, „Power-aware scheduling of preemptable jobs on identical parallel processors to minimize makespan", *Annals of Operations Research* 213 (1), 235–252 (2014).

[10] R. Różycki and J. Węglarz, "Solving a power-aware scheduling problem by grouping jobs with the same processing characteristic", *Discrete Applied Mathematics* 182, 150–161 (2015).

[11] J. Józefowska, M. Mika, R. Różycki, G. Waligóra, and J. Węglarz, "Solving the discrete-continuous project scheduling problem via its discretization", *Mathematical Methods of Operations Research* 52 (3), 489–499 (2000).

[12] M. Mika, G. Waligóra, and J. Węglarz, "Modelling and solving grid resource allocation problem with network resources for workflow applications", *Journal of Scheduling* 14 (3), 291–306 (2011).

[13] G. Waligóra, "Heuristic approaches to discrete-continuous project scheduling problems to minimize the makespan", *Computational Optimization and Applications* 48 (2), 399–421 (2011).

[14] G. Waligóra, "Discrete-continuous project scheduling with discounted cash flows – a tabu search approach", *Computers & Operations Research* 35 (7), 2141–2153 (2008).

[15] G. Waligóra, "Discrete-continuous project scheduling with discounted cash inflows and various payment models – a review of recent results", *Annals of Operations Research* 213 (1), 319–340 (2014).

[16] G. Waligóra, "Simulated annealing and tabu search for discrete-continuous project scheduling with discounted cash flows", *RAIRO – Operations Research* 48 (1), 1–24 (2014).

[17] J. Węglarz, "Time-optimal control of resource allocation in a complex of operations framework", *IEEE Transactions on Systems, Man and Cybernetics* 6 (11), 783–788 (1976).

[18] J. Węglarz, "Multiprocessor scheduling with memory allocation – a deterministic approach", *IEEE Transactions on Computers* 29 (8), 703–709 (1980).

[19] E.L. Demeulemeester and W.S. Herroelen, *Project Scheduling – A Research Handbook*, Kluwer, Boston, 2002.

[20] J. Węglarz, "Project scheduling with discrete and continuous resources", *IEEE Transactions on Systems, Man and Cybernetics* 9 (10), 644–651 (1979).

[21] J. Józefowska, M. Mika, R. Różycki, G. Waligóra, and J.Węglarz, "An almost optimal heuristic for preemptive Cmax scheduling of dependent tasks on identical parallel processors", *Annals of Operations Research* 129 (1–4), 205–216 (2004).

[22] J. Węglarz, J. Błażewicz, W. Cellary, and R. Słowiński, "An automatic revised simplex method for constrained resource network scheduling", *ACM Transactions on Mathematical Software* 3 (3), 295–300 (1977).