BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 68, No. 2, 2020
DOI: 10.24425/bpasts.2020.133115

CONTROL, INFORMATICS AND ROBOTICS

Scheduling battery charging jobs with linearly decreasing power demands to minimize the total time

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

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

Abstract. In this work we consider a problem from the field of power- and energy-aware scheduling, in which a set of batteries have to be charged in a minimum time. The formulated problem is to schedule independent and nonpreemptable jobs to minimize the schedule length, where each job requires some amount of power and consumes a certain amount of energy during its processing. We assume that the power demand of each job linearly decreases with time, as it is the case when Li-ion batteries are being charged. For the assumed job model we prove that each next job should be started as soon as the required amount of power is available. Basing on the proven theorem we formulate a procedure generating a minimum-length schedule for an assumed order of jobs. We also analyze the case of identical jobs, and show some interesting properties of this case.

Key words: scheduling, makespan, power, energy, continuous resource.

1. Introduction

This paper deals with a scheduling problem coming, generally, from the field of green computing [1, 2], which aims at finding a good balance between computing performance and consumption of natural resources. In particular, we consider a problem belonging to the class of so-called power- and energy-aware scheduling problems [3, 4]. In the problem under consideration, there is a set of independent jobs, each of them requiring some level of power, and consuming some amount of energy while processing. The total amount of power available at a time is limited, which implies that it is not possible to execute all jobs in parallel. Thus, the goal is to find a schedule that meets the power constraint, and minimizes the makespan (schedule length).

Let us first notice that power can be treated as a doubly-constrained continuous resource (see [5]), since usually both its temporal availability and its consumption over the entire schedule (i.e. energy) are limited. In that context the considered problem is also related to the class of so-called discrete-continuous scheduling problems which have been studied in many previous papers (e.g. [6, 7]). However, in this work we assume that discrete resources (e.g. machines, terminals, etc.) are not limited; the only constrained resource is the continuous one, i.e. power. Moreover, we also assume that the energy consumption of each job is fixed and known in advance. Consequently, the energy constraint is not active in this case, since it would be only a decision constraint defining whether there is enough energy to execute the given set of jobs or not.

In this research we additionally assume that the power demands of jobs decrease with time. This is a situation when, e.g., a number of batteries are being charged from a common electrical power source. Each battery has a known capacity (energy consumption), and an initial power demand which, in general, can be different for different batteries. The power demand, however, drops along with the charging degree, as it is typical for Lithium-ion batteries used, e.g., in electric cars. The time characteristic of charging such batteries [8] shows that the power used is a nonincreasing function of the charging time, and decreases from a known initial value. Battery charging time depends on the degree of its discharge, but it can be assumed that it is known a priori. In order to model this situation, it is convenient to use a job model whose profile is described with a descending linear function with both initial and final values known. Final value corresponds to the power used at the end of charging. For the moment, we will assume that the final value is equal to 0, i.e. after a battery is fully charged, it does not require power anymore.

Although in this paper power is a continuous doubly-constrained resource, let us notice that the presented model is not restricted to that particular resource. For example, money is a classical doubly-constrained resource since, usually, both its temporal availability and the total budget of the entire project are limited [9, 10]. If, in such a case, additionally the temporary demand of each job decreases along with the job processing time, we obtain the situation considered in this paper. It is typical for many kinds of financial projects that jobs (or activities) require more money at the start of their processing, whereas towards the end of their execution their resource (money) requests drop. Similar examples concerning other practical situations adequate to the model considered in this paper can be given as well.

2. Problem formulation

We consider a problem of scheduling n independent, nonpreemptable jobs. Each job requires for its processing some

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

Manuscript submitted 2019-09-17, revised 2019-12-23, initially accepted for publication 2020-01-05, published in April 2020

amount of power, and consumes some amount of energy during its execution. Each job i, $i = 1, 2, \ldots, n$, is characterized by the amount e_i of consumed energy, which represents the size of the job, the initial power usage P_{0i}, and the power usage function $p_i(t)$. This function can be, in general, arbitrary, however, in this research we assume decreasing power usage functions of jobs, as discussed in the Introduction. Thus, the general job model can be given as follows:

$$p_i(t) = \begin{cases} 0 & \textit{for } t < s_i \\ g_i(t - s_i) & \textit{for } s_i \le t \le c_i \\ 0 & \textit{for } t > c_i \end{cases} \tag{1}$$

where $g_i(t)$ is a decreasing function, and s_i, c_i are the start and completion times of job i, respectively.

Notice that having defined the size e_i of a job, its initial power usage P_{0i}, and the power usage function $p_i(t)$, the processing time d_i of job i can be calculated using the following equations (2–4):

$$e_i = \int_{s_i}^{c_i} p_i(t)dt = \int_0^{d_i} g_i(t)dt \tag{2}$$

where

$$c_i = s_i + d_i \tag{3}$$

$$p_i(s_i) = P_{0i}. \tag{4}$$

As discussed in the Introduction, in this work we consider a special case of the presented model, in which it is assumed that:

- the power usage function $p_i(t)$ of each job is linear,
- at the completion of a job its power usage is equal to 0, i.e. $p_i(c_i) = 0$.

The above assumptions result in a special case of the described job model, presented in Fig. 1.

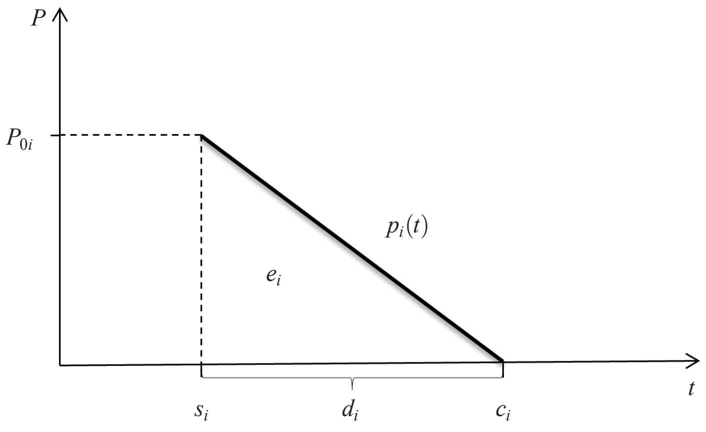

Fig. 1. Graphical presentation of the job model

Next, having two points (s_i, P_{0i}) and $(c_i, 0)$ of a linear function, we can easily derive the function formula. Consequently, the job model can be mathematically described as:

$$p_i(t) = \begin{cases} 0 & \textit{for } t < s_i \\ P_{0i} - \dfrac{P_{0i}}{d_i}(t - s_i) & \textit{for } s_i \le t \le c_i\,. \\ 0 & \textit{for } t > c_i \end{cases} \tag{5}$$

We can immediately notice that:

$$e_i = \frac{1}{2} P_{0i} d_i \tag{6}$$

which means that in this case, knowing the size e_i of job i, its processing time:

$$d_i = 2e_i / P_{0i} \tag{7}$$

can be calculated much easier than from the general formula (2).

Thus, we have a set of jobs, from among which each is graphically represented, in the system of coordinates p and t, by a rectangular triangle of height P_{0i} and length d_i.

The objective of the problem is to minimize the schedule length. However, the total amount of power available at a time is limited. We denote by P the total amount of power available at time t. Obviously, it must hold that $P \ge \max_{i=1,\ldots,n}\{P_{0i}\}$, otherwise no feasible schedule exists. Let $p(t)$ be the total power used by all jobs processed at time t, i.e.:

$$p(t) = \sum_{i \in A_t} p_i(t)$$

where A_t is the set of jobs processed at time t. Taking into account equation (5) we can write:

$$p(t) = \sum_{i \in A_t} \left(P_{0i} - \frac{P_{0i}}{d_i}(t - s_i) \right) \tag{8}$$

and consequently, the considered problem can be mathematically formulated as:

Problem T

minimize $$C_{max} = \max_{i=1,\ldots,n} \{c_i\} \tag{9}$$

subject to $$c_i = s_i + d_i, \quad i = 1, 2, \ldots, n \tag{10}$$

$$\sum_{i \in A_t} \left(P_{0i} - \frac{P_{0i}}{d_i}(t - s_i) \right) \le P \quad \text{for any } t. \tag{11}$$

Thus, the problem is to find a vector $\mathbf{s} = [s_1, s_2, \ldots, s_n]$ of starting times of jobs that minimizes the schedule length C_{max} subject to the above constraints.

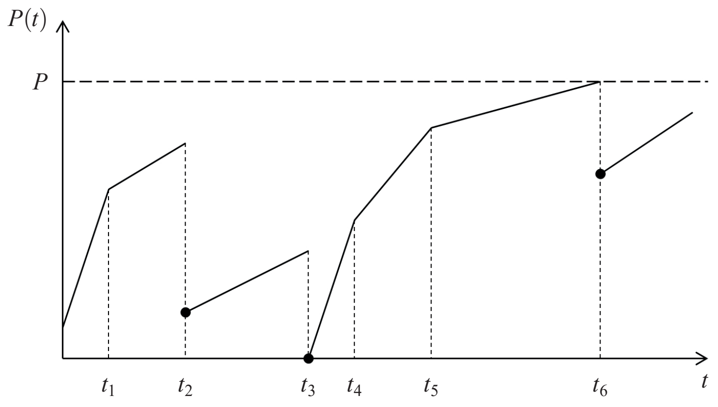

Fig. 2. Function $P(t)$ of the remaining power amount

Notice that from constraint (11) we can define the function of the remaining amount of power available at time t as:

$$P(t) = P - \sum_{i \in A_t} \left(P_{0i} - \frac{P_{0i}}{d_i}(t - s_i) \right). \tag{12}$$

The function $P(t)$ in (12) has always the shape presented in Fig. 2. In order to discuss the properties of the function, let us first introduce two definitions:

Definition 1. The function $P(t)$ is divided into intervals defined by starting or completion times of consecutive jobs. One such interval will be called interval of power function and denoted by IPF.

Definition 2. A super-interval (SPF) is a sequence of IPFs in which function $P(t)$ is continuous.

In Fig. 2 there are six IPFs, and each t_k, $k = 1, 2, \ldots, 6$, is the end of IPF$_k$ as well as the beginning of IPF$_{k+1}$. However, there are only three SPFs: $[0; t_2]$, $[t_2; t_3]$ and $[t_3; t_6]$. Notice that when a job starts, the value of function $P(t)$ rapidly drops, since the job being started requires some amount of power to begin. Such a situation always determines a discontinuity of function $P(t)$ (points t_2, t_3 and t_6 in Fig. 2). However, between such two consecutive times, the function is always increasing, concave, and piece-wise linear, which follows from completion of other jobs. Such a section, according to Definition 2, is called a super-interval (SPF). Notice that after a job ends, a next IPF starts in which function $P(t)$ grows slower; in other words, the slope of the next section to the time axis (i.e. the drop coefficient) is smaller. It follows from the fact that the completed job stops "returning" power (see formula (12)). This fact makes the function concave. Finally, function $P(t)$ may only take values from 0 to P. If a starting job (or jobs) uses the total amount of power available at a time, the value of $P(t)$ drops to 0 (point t_3 in Fig. 2). On the other hand, when successive jobs being finished "return" power, while no new job is started, the value of $P(t)$ may reach P, as it is at point t_6 in Fig. 2. In all other cases function $P(t)$ starts a new IPF with a value between 0 and P (e.g. point t_2 in Fig. 2).

3. Basic theorem

In this section we formulate and prove a basic theorem for constructing an optimal schedule to Problem T defined in Section 2.

Let us first assume that jobs on a job list JL are ordered according to their nondecreasing starting times. i.e. for each job in position q, $q = 2, 3, \ldots, n$, on JL the following condition holds:

$$s_{JL[q]} \geq s_{JL[q-1]}, \; q = 2, \ldots, n$$

which means that job $JL[q]$ in position q on JL must not start before any of its predecessors on JL.

Theorem 1. For a defined job list JL, an optimal schedule is obtained by scheduling each successive job i from the list at the earliest possible time when the required amount P_{0i} of power becomes available.

Proof. For the purpose of the proof, let us focus on one SPF of function $P(t)$, as presented in Fig. 3.

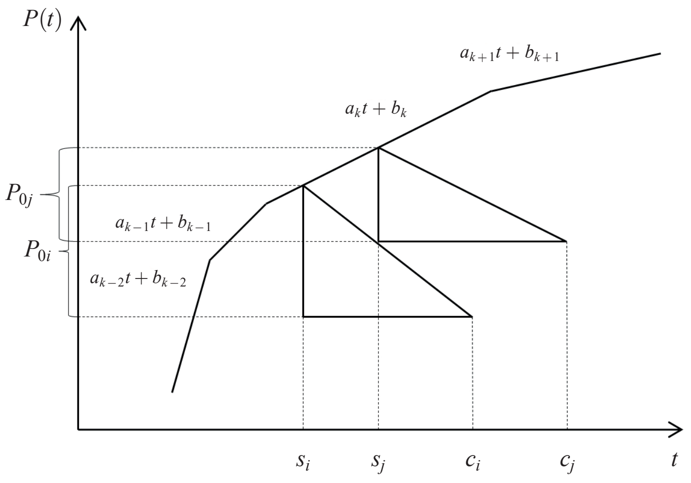

Fig. 3. An SPF

Assume that there are two jobs from JL to be scheduled: job i and job j. Since no other job is to be started, function $P(t)$ maintains its continuity, and thereby considering an SPF is justified. Then assume that job i is scheduled at time s_i within interval IPF$_k$, where the power function has the form $a_k t + b_k$. According to the assumption made at the beginning of this section, job j must not be started before s_i, i.e. $s_j \geq s_i$. As a result, one of two possible situations may happen:

– either

A) job j requiring power amount P_{0j} cannot be started within IPF$_k$ because insufficient power is available in the interval (function $a_k t + b_k$ does not reach the value of P_{0j} till the end of IPF$_k$), and – consequently – job j will be scheduled in one of the successive intervals: IPF$_{k+1}$, IPF$_{k+2}$, …

– or

B) job j may be started within IPF$_k$ since the available amount of power is high enough (function $a_k t + b_k$ reaches P_{0j} by the end of IPF$_k$).

It is obvious that the schedule obtained in case A will not be better than the schedule obtained in case B, since in both cases job i is started at the same moment s_i, whereas job j in case A is started later. Thus, it is sufficient to show that if both jobs start within the same interval IPF$_k$, a schedule obtained as a result of any delay of job i (i.e. by starting job i at a time $s_i + dt$, $dt > 0$) will not be better than a schedule obtained by starting job i as early as possible (i.e. when $a_k t + b_k = P_{0i}$).

Let us start with defining the function of the remaining power amount in IPF$_k$ when job i is started at time s_i:

$$P_k(t) = a_k t + b_k - \left(P_{0i} - \frac{P_{0i}}{d_i}(t - s_i)\right) = \\ = \left(a_k + \frac{P_{0i}}{d_i}\right)t + b_k - P_{0i} - \frac{P_{0i}}{d_i}s_i.$$

Job j may be started when function $P_k(t)$ reaches the value of P_{0j}:

$$\left(a_k + \frac{P_{0i}}{d_i}\right)t + b_k - P_{0i} - \frac{P_{0i}}{d_i}s_i = P_{0j}$$

and hence:

$$t = \left(P_{0i} + P_{0j} - b_k + \frac{P_{0i}}{d_i}s_i\right) \Big/ \left(a_k + \frac{P_{0i}}{d_i}\right) = s_j. \quad (13)$$

Finally, the part of the schedule composed of jobs i and j will be completed at time c_{ij}:

$$c_{ij} = \max\{c_i; c_j\} = \max\{s_i + d_i; s_j + d_j\}. \quad (14)$$

Now, let us assume that job i is delayed by $\Delta t > 0$, i.e. it starts at time $s_i + \Delta t$ in IPF$_k$. In such a case the function of the remaining power amount in IPF$_k$ will take the form:

$$P'_k(t) = a_k t + b_k - \left(P_{0i} - \frac{P_{0i}}{d_i}\left[t - (s_i + \Delta t)\right]\right) = \\ = \left(a_k + \frac{P_{0i}}{d_i}\right)t + b_k - P_{0i} - \frac{P_{0i}}{d_i}s_i - \frac{P_{0i}}{d_i}\Delta t.$$

This function will reach the value of P_{0j} enabling job j to start when:

$$\left(a_k + \frac{P_{0i}}{d_i}\right)t + b_k - P_{0i} - \frac{P_{0i}}{d_i}s_i - \frac{P_{0i}}{d_i}\Delta t = P_{0j}$$

which means that:

$$t = \left(P_{0i} + P_{0j} - b_k + \frac{P_{0i}}{d_i}s_i + \frac{P_{0i}}{d_i}\Delta t\right) \Big/ \\ \Big/ \left(a_k + \frac{P_{0i}}{d_i}\right) = s'_j. \quad (15)$$

This time the part of the schedule will be completed at time c'_{ij}:

$$c'_{ij} = \max\{c'_i; c'_j\} = \max\{s_i + \Delta t + d_i; s'_j + d_j\}. \quad (16)$$

Now, comparing (14) and (16) it can be seen that:

$$c'_{ij} > c_{ij}$$

since:

$$s_i + \Delta t + d_i > s_i + d_i$$

which is obvious because $\Delta t > 0$, and:

$$s'_j > s_j$$

which, in turn, follows from the fact that $\frac{P_{0i}}{d_i}\Delta t > 0$ in (15) and thus, the numerator in (15) is greater than the numerator in (13), whereas both denominators are equal.

Thus, delaying job i will always result in an inferior partial schedule composed of job i and j. The same argument can be applied for any pair of jobs. As a result, the entire schedule may not be better that the schedule obtained by starting each job i as soon as possible. □

Summarizing, for a given list JL a minimum-length schedule is found by starting successive job from the list as soon as the required amount of power becomes available. Obviously, the first l jobs from JL for which $\sum_{q=1}^{l} P_{0,JL[q]} \le P$ start at time $t = 0$. Then, it is easy to establish the earliest start time for the next job j in position q of JL, $q = l+1, l+2, \ldots, n$, using formula (12). From equations (12) and $P(t) = P_{0j}$ we obtain:

$$P - \sum_{i \in A_t}\left(P_{0i} - \frac{P_{0i}}{d_i}(t - s_i)\right) = P_{0j}$$

hence:

$$\sum_{i \in A_t} \frac{P_{0i}}{d_i} t = P_{0j} - P + \sum_{i \in A_t} P_{0i} + \sum_{i \in A_t} \frac{P_{0i}}{d_i} s_i$$

and, by deriving t, we obtain:

$$t = \frac{P_{0j} - P + \sum_{i \in A_t} P_{0i} + \sum_{i \in A_t} \frac{P_{0i}}{d_i} s_i}{\sum_{i \in A_t} \frac{P_{0i}}{d_i}} = s_j. \quad (17)$$

The next job j from list JL should be started at time s_j.

Now, in order to find a globally optimal schedule, all lists JL have to be generated and the one leading to an optimum schedule has to be chosen. Unfortunately, the number of all such lists grows exponentially with the number of jobs, as they are permutations of n independent jobs. The number of all such permutations is, of course, $n!$. However, for a given list JL Theorem 1 can be used to construct a schedule of the minimum length. In the next section a procedure for generation such a schedule will be described.

4. Schedule generation procedure

According to Theorem 1, each successive job from list JL has to be started at the earliest possible moment. Hawing known the set A_t of jobs processed at time t, the calculation of the closest time stamp at which the next job from the list should be started is done from equation (17). Now, we can distinguish combinations (subsets) of jobs performed between two consecutive time stamps. Let Z_k denote the set of jobs executed between time stamps $k-1$ and k, $k = 1, 2, \ldots$, i.e. in the k-th time interval. t_k is the k-th time stamp, i.e. the end of the k-th interval. $\mathbf{s}_k$ and $\mathbf{c}_k$ are vectors of size h, where h is the number of jobs already scheduled, containing the starting and completion times, correspondingly, of jobs started up to time t_k. If vector Q of size n contains for each job its position q on list JL, i.e. $Q[i] = q \Leftrightarrow JL[q] = i$, then we can write:

$$\mathbf{c}_k\big[Q[i]\big] = \mathbf{s}_k\big[Q[i]\big] + d_i, \text{ for every } i \in Z_k. \tag{18}$$

Moreover, let n_k denote the number of jobs in combination Z_k, i.e. $n_k = |Z_k|$. It is easy to calculate the value of n_1 as the biggest integer for which the following condition holds:

$$\sum_{q=1}^{n_1} P_{0,\,JL[q]} \le P \tag{19}$$

i.e. in the first combination Z_1 the maximum number n_1 of the first jobs from JL is processed, for which their overall initial power demand does not exceed the power limit P. Thus, jobs $JL[1]$ to $JL[n_1]$ are processed in the first interval. If n_1 is equal to n, it means that all jobs can be started in the first interval, and the algorithm stops.

Let us now stress one important issue. At any time we know the set of jobs performed at that time, along with their starting times and completion times. It may sometimes happen (and, surely, will happen) that the next time stamp calculated from formula (17) will exceed completion time of one or more jobs. In that situation we have to reconfigure the set A_t of jobs being executed by removing from the set the job (or jobs) that is finished at the earliest, and then recalculate the next time stamp for the new reduced set A_t. More precisely, if for the current combination Z_k the calculated value of time stamp t_k is greater than any element of $\mathbf{c}_k$, we have to generate a new combination Z_{k+1} by removing from Z_k job (or jobs) with completion time $\mathrm{c}_k^{min} = \min_{i\in Z_k}\big\{\mathbf{c}_k\big[Q[i]\big]\big\}$ and then calculate a new value of t_{k+1} for the reduced combination Z_{k+1}, using formula (17) again.

On a basis of the above discussion, we can propose the following iterative procedure for generation an optimal schedule for a given list JL.

Schedule generation procedure (SGP):

```
Step 1.
  $k := 1$;
  Calculate $n_1$ as the biggest integer for which condition (19) holds;
  if $(n_1 = n)$
  then STOP and $C_{max} := \max_{i=1,\ldots,n}\{d_i\}$;
  otherwise:
    Put the first $n_1$ jobs from $JL$ to $Z_1$;
    Set 0 to the first $n_1$ positions of $\mathbf{s}_k$;
    Calculate the first $n_1$ elements of $\mathbf{c}_k$ from (18);
Step 2.
    repeat
      $A_t := Z_k$;
      $\mathrm{c}_k^{min} := \min_{i\in Z_k}\{\mathbf{c}_k[Q[i]]\}$;
      Calculate $t_k$ from (17);
        if $(t_k > \mathrm{c}_k^{min})$ then
          $t_k := \mathrm{c}_k^{min}$;
          $Z_{k+1} := Z_k \backslash \{i\colon i \in Z_k \wedge \mathbf{c}_k[Q[i]] = \mathrm{c}_k^{min}\}$
        otherwise
          $Z_{k+1} := Z_k \cup \{$next unscheduled job from $JL\}$
    Create vectors $\mathbf{s}_{k+1}$ and $\mathbf{c}_{k+1}$;
    $k++$;
    until (there is no unscheduled job on $JL$);
Step 3.
  $C_{max} := \max_{i=1,\ldots,n}\{\mathbf{c}_k[i]\}$.
```

The complexity of the procedure is $O(n^2)$ since n jobs are being successively scheduled, and for each of them at most n backsteps may be needed in Step 2 in order to reduce the current combination Z_k.

5. The case of identical jobs

In this section we will analyze a special case of the problem where initial power demands, as well as the lengths (processing times) of all jobs are identical, i.e. $P_{0i} = P_0$, $d_i = d$, $i = 1, 2, \ldots, n$. In this situation we can treat all jobs as unit-time ones, i.e. we can assume that $d_i = d = 1$, $i = 1, 2, \ldots, n$.

For unit-time jobs, formula (17) takes the form:

$$t = \frac{Y + \sum_{i\in A_t} 1 + \sum_{i\in A_t} s_i}{\sum_{i\in A_t} 1} \tag{20}$$

where $Y = \left(1 - \frac{P}{P_0}\right) < 0$ is a negative constant (since $P > P_0$).

Notice now that for $A_t = Z_k$, component $\sum_{i\in A_t} 1$ in (20) is equal to n_k, i.e. the number of jobs processed in combination Z_k. As a result, we obtain:

$$t_k = \frac{Y + \sum_{i\in Z_k} s_i}{n_k} + 1 \tag{21}$$

and, in particular:

$$t_1 = \frac{Y}{n_1} + 1. \tag{22}$$

Thus, for the case of unit-time jobs, the calculation of the next time stamp at which the next job can be started is much easier that in the general case. Moreover, it is obvious, that for unit-time jobs the choice of the next job to perform is of no importance – each job is represented by a triangle of the same height and length. Thus, the jobs can be scheduled in an arbitrary order, e.g. according to their increasing indices, and the next job is started as soon as the available amount of power is sufficient, i.e. at the consecutive time stamp t_k given by formula (21). Consequently, for identical jobs an optimal schedule is found by applying procedure SGP to an arbitrary job list JL.

Let us also add that in each IPF$_k$, the formula for the function of the remaining power amount can be easily established from formula (12) in which $P_{0i} = P_0$, $d_i = d$, $i = 1, 2, \ldots, n$, i.e.:

$$P_k(t) = P - \sum_{i \in Z_k} P_0 + \sum_{i \in Z_k} P_0 t - \sum_{i \in Z_k} P_0 s_i \tag{23}$$

where $t \in [t_{k-1}, t_k]$.

6. Numerical example

In this section we show a numerical example of applying procedure SGP to a case of identical jobs.

Let us consider the following instance of the problem. A set of 7 jobs is to be scheduled. The maximum available amount of power is equal to 5, whereas the initial power usage is equal to 2. Thus, the parameters of the problem are: $n = 7$; $P = 5$; $P_0 = 2$. We assume that jobs on list JL appear according to their increasing indices.

Now, let us analyze the steps of procedure SGP:

Step 1.

$k := 1$; $n_1 := \lfloor \nicefrac{5}{2} \rfloor = 2$;

Since $n_1 < n$, the procedure does not stop, and $n_1 = 2$ jobs (job 1 and job 2) can be started at time $t_0 = 0$. The following combination Z_1 and vectors $\mathbf{s}_1$ and $\mathbf{c}_1$ are:

$Z_1 = \begin{Bmatrix} 1 \\ 2 \end{Bmatrix}$; $\mathbf{s}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$; $\mathbf{c}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$;

We can now calculate the constant Y:

$Y := 1 - \frac{5}{2} = -\frac{3}{2}$;

and the time t_1 at which the next job will be able to start:

$$t_1 = \frac{Y}{n_1} + 1 = \frac{-\nicefrac{3}{2}}{2} + 1 = \frac{1}{4};$$

The function of remaining power amount (see formula (23)) for $t \in [0; \nicefrac{1}{4}]$ is $P_1(t) = 4t + 1$ and, as it is easy to see, $P_1(\nicefrac{1}{4}) = 2 = P_0$.

Step 2a. (* beginning of „repeat until" loop *)

A new job 3 is added to the set of jobs being processed. It starts at time $t_1 = \nicefrac{1}{4}$.

$k = 2$; $n_2 = 3$;

$Z_2 = \begin{Bmatrix} 1 \\ 2 \\ 3 \end{Bmatrix}$; $\mathbf{s}_2 = \begin{bmatrix} 0 \\ 0 \\ \nicefrac{1}{4} \end{bmatrix}$; $\mathbf{c}_1 = \begin{bmatrix} 1 \\ 1 \\ \nicefrac{5}{4} \end{bmatrix}$;

Now we calculate the starting time of job 4:

$$t_2 = \frac{-\nicefrac{3}{2} + (0 + 0 + \nicefrac{1}{4})}{3} + 1 = \frac{7}{12}.$$

None of the jobs from combination Z_2 is completed before time t_2.

Remaining power amount function for $t \in [\nicefrac{1}{4}, \nicefrac{7}{12}]$ is $P_2(t) = 6t - \nicefrac{3}{2}$, and $P_2(\nicefrac{1}{4}) = 0$; $P_2(\nicefrac{7}{12}) = 2$.

Step 2b.

This iteration goes analogically to Step 2a.

$k = 3$; $n_3 = 4$;

$Z_3 = \begin{Bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{Bmatrix}$; $\mathbf{s}_3 = \begin{bmatrix} 0 \\ 0 \\ \nicefrac{1}{4} \\ \nicefrac{7}{12} \end{bmatrix}$; $\mathbf{c}_1 = \begin{bmatrix} 1 \\ 1 \\ \nicefrac{5}{4} \\ \nicefrac{19}{12} \end{bmatrix}$;

$$t_3 = \frac{-\nicefrac{3}{2} + (\nicefrac{1}{4} + \nicefrac{7}{12})}{4} + 1 = \frac{5}{6}.$$

Again, no job from combination Z_3 is completed before time t_3.

Remaining power amount function for $t \in [\nicefrac{7}{12}, \nicefrac{5}{6}]$ is $P_3(t) = 8t - 4\nicefrac{2}{3}$, and $P_3(\nicefrac{7}{12}) = 0$; $P_3(\nicefrac{5}{6}) = 2$.

Step 2c.

$k = 4$; $n_4 = 5$;

$Z_4 = \begin{Bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{Bmatrix}$; $\mathbf{s}_4 = \begin{bmatrix} 0 \\ 0 \\ \nicefrac{1}{4} \\ \nicefrac{7}{12} \\ \nicefrac{5}{6} \end{bmatrix}$; $\mathbf{c}_4 = \begin{bmatrix} 1 \\ 1 \\ \nicefrac{5}{4} \\ \nicefrac{19}{12} \\ \nicefrac{11}{6} \end{bmatrix}$;

$$t_4 = \frac{-\nicefrac{3}{2} + (\nicefrac{1}{4} + \nicefrac{7}{12} + \nicefrac{5}{6})}{5} + 1 = \frac{31}{30}.$$

Notice that now $\mathbf{c}_4[1] = \mathbf{c}_4[2] = 1 < t_4$, and thus job 1 and job 2 from combination Z_4 would be completed before time t_4. We have to remove them from the combination and calculate time t_5 for the reduced combination Z_5 containing $n_5 = 3$ jobs. Consequently:

$Z_5 = \begin{Bmatrix} 3 \\ 4 \\ 5 \end{Bmatrix}$; $\mathbf{s}_5 = \begin{bmatrix} \cancel{0} \\ \cancel{0} \\ \nicefrac{1}{4} \\ \nicefrac{7}{12} \\ \nicefrac{5}{6} \end{bmatrix}$; $\mathbf{c}_5 = \begin{bmatrix} \cancel{1} \\ \cancel{1} \\ \nicefrac{5}{4} \\ \nicefrac{19}{12} \\ \nicefrac{11}{6} \end{bmatrix}$;

$n_5 = 3$;

$$t_5 = \frac{-\nicefrac{3}{2} + (\nicefrac{1}{4} + \nicefrac{7}{12} + \nicefrac{5}{6})}{3} + 1 = \frac{19}{18}.$$

Now, in order to define the formula for the remaining power amount function, we have to divide the 4th SPF into two IPFs: $\left[{}^5/_6, 1\right]$ and $\left[1, {}^{19}/_{18}\right]$. For $t \in \left[{}^5/_6, 1\right]$ the function is $P_4(t) = 10t - 8\,{}^1/_3$, where $P_4\left({}^5/_6\right) = 0$; $P_4(1) = 1\,{}^2/_3$, whereas for $t \in \left[1, {}^{19}/_{18}\right]$ the function is $P_5(t) = 6t - 4\,{}^1/_3$, where $P_5(1) = 1\,{}^2/_3$; $P_5\left({}^{19}/_{18}\right) = 2$.

Step 2d.

$k = 6$; $n_6 = 4$;

$$Z_6 = \begin{Bmatrix} 3 \\ 4 \\ 5 \\ 6 \end{Bmatrix}; \ \mathbf{s}_6 = \begin{bmatrix} \cancel{0} \\ \cancel{0} \\ {}^1/_4 \\ {}^7/_{12} \\ {}^5/_6 \\ {}^{19}/_{18} \end{bmatrix}; \ \mathbf{c}_6 = \begin{bmatrix} \cancel{1} \\ \cancel{1} \\ {}^5/_4 \\ {}^{19}/_{12} \\ {}^{11}/_6 \\ {}^{37}/_{18} \end{bmatrix};$$

$$t_6 = \frac{-{}^3/_2 + \left({}^1/_4 + {}^7/_{12} + {}^5/_6 + {}^{19}/_{18}\right)}{3} + 1 = \frac{47}{36}.$$

Job 3 is completed before time t_6, and it will be removed from combination Z_6.

$$Z_7 = \begin{Bmatrix} 4 \\ 5 \\ 6 \end{Bmatrix}; \ \mathbf{s}_7 = \begin{bmatrix} \cancel{0} \\ \cancel{0} \\ \cancel{{}^1/_4} \\ {}^7/_{12} \\ {}^5/_6 \\ {}^{19}/_{18} \end{bmatrix}; \ \mathbf{c}_7 = \begin{bmatrix} \cancel{1} \\ \cancel{1} \\ \cancel{{}^5/_4} \\ {}^{19}/_{12} \\ {}^{11}/_6 \\ {}^{37}/_{18} \end{bmatrix};$$

$n_7 = 3$;

$$t_7 = \frac{-{}^3/_2 + \left({}^7/_{12} + {}^5/_6 + {}^{19}/_{18}\right)}{3} + 1 = \frac{143}{108}.$$

As in Step 4, the current SPF has to be divided into two IPFs: $\left[{}^{19}/_{18}, {}^5/_4\right]$ and $\left[{}^5/_4, {}^{143}/_{108}\right]$. For $t \in \left[{}^{19}/_{18}, {}^5/_4\right]$ the remaining power amount function is $P_6(t) = 8t - 8\,{}^4/_9$, where $P_6\left({}^{19}/_{18}\right) = 0$; $P_6\left({}^5/_4\right) = 1\,{}^5/_9$, whereas for $t \in \left[{}^5/_4, {}^{143}/_{108}\right]$ the function is $P_7(t) = 6t - 5\,{}^{17}/_{18}$, where $P_7\left({}^5/_4\right) = 1\,{}^5/_9$; $P_7\left({}^{143}/_{108}\right) = 2$.

Step 2e.

$k = 8$; $n_8 = 4$;

$$Z_8 = \begin{Bmatrix} 4 \\ 5 \\ 6 \\ 7 \end{Bmatrix}; \ \mathbf{s}_8 = \begin{bmatrix} \cancel{0} \\ \cancel{0} \\ \cancel{{}^1/_4} \\ {}^7/_{12} \\ {}^5/_6 \\ {}^{19}/_{18} \\ {}^{143}/_{108} \end{bmatrix}; \ \mathbf{c}_8 = \begin{bmatrix} \cancel{1} \\ \cancel{1} \\ \cancel{{}^5/_4} \\ {}^{19}/_{12} \\ {}^{11}/_6 \\ {}^{37}/_{18} \\ {}^{251}/_{108} \end{bmatrix};$$

Since all the jobs have been scheduled then STOP.

Step 3.

$$C_{max} = \mathbf{c}_6[7] = {}^{251}/_{108} \approx 2{,}3.$$

The final schedule is shown in Fig. 4.

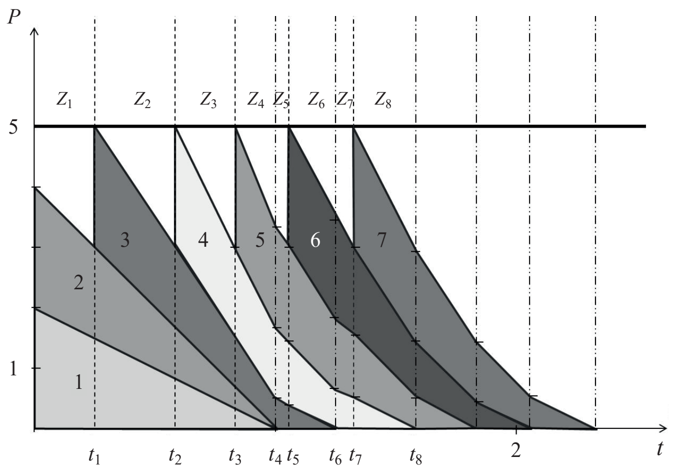

Fig. 4. Gantt chart for the numerical example

Notice that for the case of identical jobs vector $\mathbf{c}_k$ is not necessary, since its elements are always greater by 1 than the corresponding elements of vector $\mathbf{s}_k$ (i.e. $\mathbf{c}_k[i] = \mathbf{s}_k[i] + 1$). It is, obviously, not the case for the general problem.

7. Property of identical jobs case

In this section we show an interesting property of the case of identical jobs, defining the maximum number of jobs performed in parallel at a time.

Property 1. The maximum number of jobs performed in parallel at a given moment does not exceed the number $n_1 + 1 + x$, where x is the maximum integer for which the following inequality is met:

$$t_1 + \sum_{i=1}^{x} \frac{1}{n_1 + i} < 1. \tag{24}$$

As you can see, the maximum number of jobs performed in parallel in a given schedule obtained using procedure SGP occurs just before the end of the earliest completed job (or jobs). It is enough to specify how many jobs will be started up to this point. At the moment $t = 0$, n_1 jobs are started. It is known that the next job will be launched at the moment t_1 given by formula (22).

For identical jobs, it can be shown that until the first job (or jobs) is completed, the intervals between subsequent jobs' start times do not depend on P or P_0, and are equal to the inverse of the number of currently running jobs. Therefore, the maximum number of jobs running simultaneously can be determined by formula (24).

Let us consider the following instance of the problem. The maximum available amount of power is equal to 10, whereas the initial power usage is equal to 3. Thus, the parameters of the problem are: $P = 10$; $P_0 = 3$.

The schedule where consecutive jobs are started till the completion of the first job is presented in Fig. 5.

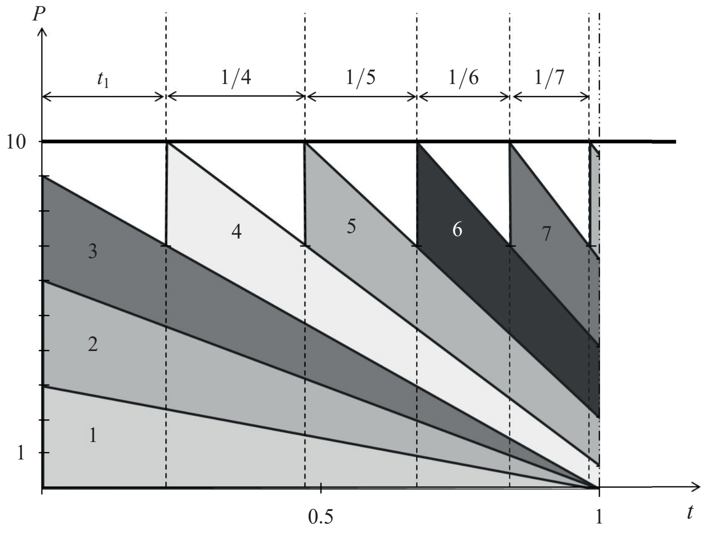

Fig. 5. Jobs started till the completion of the first job

For the considered instance, the maximum number of jobs executed in parallel equals 8 since:

$$t_1 = \frac{-10/3}{3} + 1 = \frac{1}{9}$$

$$t_1 + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} = \frac{1237}{1260} < 1 < \frac{1237}{1260} + \frac{1}{8}.$$

8. Conclusions

In this work, we have considered a problem of scheduling non-preemptable and independent jobs with power demands linearly decreasing with time, in order to minimize the schedule length. We have proved that in an optimal schedule, each job should be started as soon as the required power amount becomes available. As a result, in order to find a globally optimal schedule, all sequences of jobs have to be examined, in general. Thus, various heuristics, metaheuristics, or priority rules can be applied to look for an optimal job permutation. We have presented a schedule generation procedure (SGP), building a minimum-length schedule for a given sequence of jobs.

We have also analyzed a special case of the problem with identical jobs, for which an arbitrary job sequence leads to an optimal schedule. For this case we have illustrated the SGP procedure by an in-depth numerical example. We have also shown an interesting property of this case, related to a maximum number of jobs performed in parallel.

In the future research we plan to further analyze the general problem in order to identify properties of optimal schedules. On the other hand, various heuristics for finding sequences of jobs leading to high-quality schedules can be computationally tested. Practically justified extensions and special cases of the problem should be analytically and numerically studied in future works.

Acknowledgement. This research has been partially supported by the statutory funds of Poznan University of Technology.

References

[1] B.E. Smith, *Green Computing: Tools and Techniques for Saving Energy, Money, and Resources*, CRC Press, 2013.
[2] S. Asadi, A.R.C. Hussin, and H.B.M. Dahlan, "Organizational research in the field of Green IT: A systematic literature review from 2007 to 2016", *Telemat Inform* 34(7), 1191–1249 (2017).
[3] R. Różycki and J. Węglarz, "On job models in power management problems", *Bull. Pol. Ac.: Tech.* 57(2), 147–151 (2009).
[4] R. Różycki and G. Waligóra, "On a certain class of power- and energy-related scheduling problems", *Discrete Appl. Math.* 264, 167–187 (2019).
[5] J. Węglarz, "Project scheduling with continuously-divisible doubly constrained resources", *Manage. Sci.* 27(9), 1040–1053 (1981).
[6] J. Józefowska, M. Mika, R. Różycki, G. Waligóra, and J. Węglarz "Discrete-continuous scheduling to minimize the makespan for power processing rates of jobs", *Discrete Appl. Math.* 94 (1–3), 263–285 (1999).
[7] R. Różycki and J. Węglarz, "Solving a power-aware scheduling problem by grouping jobs with the same processing characteristic", *Discrete Appl. Math.* 182, 150–161 (2015).
[8] I. Buchmann, Batteries in a Portable World – A Handbook on Rechargeable Batteries for Non-Engineers, Fourth Edition, Cadex Electronics Inc., 2016.
[9] G. Waligóra, "Discrete-continuous project scheduling with discounted cash inflows and various payment models – a review of recent results", *Ann. Oper. Res.* 213(1), 319–340 (2014).
[10] G. Waligóra, "Simulated annealing and tabu search for discrete-continuous project scheduling with discounted cash flows", *RAIRO Oper. Res.* 48(1), 1–24 (2014).