Scheduling of battery charging tasks with limited common power source

Lemański T.[1], Różycki R.[1], Waligóra G.[1], and Węglarz J.[1]

[1]Poznan University of Technology, Poland
e-mail: Rafal.Rozycki@cs.put.poznan.pl

Keywords: continuous resource, e-mobility, scheduling, makespan.

1. Introduction

In this work we consider a problem of scheduling battery charging tasks, assuming that the available amount of shared power is limited, and insufficient to charge all batteries in parallel. The battery charging process is very complex, and depends mainly on the type of batteries. A very popular type of battery is a Li-ion battery, used in both portable electronic equipment and electric cars. There are four main charging periods (Manwell, McGowan(1993)), of which the longest is the saturation phase. In this phase, along with the passing of time, approximately a linear decrease in power usage is observed. Therefore, it is justified to model this process using a linear function.

In this work we consider the problem of charging a set of batteries of the same type with different capacities and degrees of discharge. For simplicity, we will assume that charging each battery is limited to the saturation phase only.

Moreover, we assume that the number of charging points (a discrete resource) is unlimited, and the only limited resource is the available power, which by its nature can be allocated to charging tasks in any amounts from a certain range, i.e. it is a continuous renewable resource (Błażewicz et al. (2007)). Once started, the charging task cannot be interrupted, as it would impair the properties of the battery being charged. As one can see, such a non-classical scheduling problem is non-trivial, because one should specify the order of charging tasks that would lead to feasible schedules with the best value of the adopted criterion. In our case, this criterion is the length of the schedule.

In the next part of the work we will present the formulation of the problem, selected properties of the problem solution, and the results of a computational experiment.

2. Problem formulation

We consider a problem of scheduling n independent, non-preemptable jobs (charging tasks). Each job requires for its processing some amount of power, and consumes some amount of energy during its execution. The number of machines (a machine represents a single charging point) is unlimited and discrete resources have no influence on the final solution. 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 linear power usage functions of jobs, as discussed in the Introduction. Moreover, at the completion of a job its power usage is equal to 0. Consequently, a job is sufficiently described by only two parameters, namely: e_i, P_{0i}, in our simplified situation. The model of a job is showed on Fig. 1, where s_i, c_i represent the start and completion times of job i, respectively.

Thus, the assumed job model can be given as follows:

$$p_i(t) = \begin{cases} 0 & for\ t < s_i \\ P_{0i} - \frac{P_{0i}}{d_i}(t - s_i) & for\ s_i \le t \le c_i \\ 0 & for\ t > c_i \end{cases} \tag{1}$$

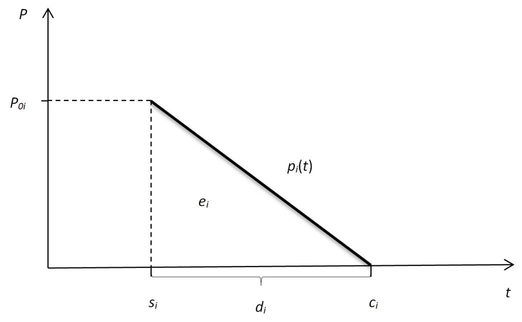

Figure 1. Graphical presentation of the job model

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 equation (2):

$$d_i = 2e_i/P_{0i} \qquad (2)$$

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 $P \geq \max_{i=1,\dots,n}\{P_{0i}\}$, 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 (1) we can write:

$$p(t) = \sum_{i \in A_t}\left(P_{0i} - \frac{P_{0i}}{d_i}(t - s_i)\right) \qquad (3)$$

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

Problem T

minimize $$C_{max} = \max_{i=1,\dots,n}\{c_i\} \qquad (4)$$

subject to $$c_i = s_i + d_i, \;\; i = 1,2,\dots,n \qquad (5)$$

$$\sum_{i \in A_t}\left(P_{0i} - \frac{P_{0i}}{d_i}(t - s_i)\right) \leq P \quad \text{for any } t \qquad (6)$$

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

3. Properties of solutions

Let us now discuss some properties of the defined problem that can be useful for the developed solution approach.

Since due to insufficient power, in general it is not possible to start all tasks in parallel, the question arises - how to construct a schedule of the minimal length. Suppose we know a certain order of job execution, i.e. there exists a list *JL* where jobs are ordered according to their non-decreasing starting times. 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*. In this situation, the following property is useful.

Property 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.

Note that the consequence of Property 1 is that in the optimal schedule, at time 0 one should run as many initial jobs from the *JL* list as possible. The moment of starting the next job from the list with the initial power consumption equal P_{0j} can be obtained by transforming (3) to the following formula:

$$s_j = \frac{P_{0j} - P + \sum_{i \in A_t} P_{0i} + \sum_{i \in A_t} \frac{P_{0i}}{d_i} s_i}{\sum_{i \in A_t} \frac{P_{0i}}{d_i}} \qquad (7)$$

In this formula, the key role is played by the information how many tasks are actually performed at the moment of starting task j, because some of the tasks may have already been finished. On the basis of Property 1, the following algorithm can be proposed, which for the known job order (represented by a particular *JL* list) determines the optimal moments of starting consecutive jobs for the considered scheduling criterion.

Algorithm A

```
Step 1. At time 0 run the maximum number of initial jobs from the JL list enabled by the
        available amount of power
Step 2. If any job remains on the JL list, repeat:
Step 2a. Find the moment to start the next job from the JL list from (7) based on information
        about jobs being currently performed.
Step 2b. If the calculated starting time is later than the fastest-ending job being completed,
        Then: remove the fastest-ending job from the set of jobs being currently performed;
        Otherwise: remove the consecutive job from the JL list, put it in the schedule at the
                calculated start time, and add to the set of jobs being currently performed.
        Return to the beginning of Step 2
Step 3. Take the finish time of the last job as the schedule length.
```

Algorithm A has the complexity of $O(n^2)$ which results from Step 2. The importance of Property 1 follows from the fact that optimal schedule can be found by using Algorithm A for each possible job permutation on the *JL* list. Of course, a full enumeration technique has an exponential complexity and is computationally inefficient. However, the solutions found in that way can be a useful reference point for assessing solutions obtained using heuristic algorithms.

Another immediate consequence of the above property is the following natural observation for the identical jobs case (i.e. $P_{0i} = P_0$ and $e_i = e$ for $i = 1,2,\ldots,n$).

Property 2. For identical jobs, Algorithm A finds an optimal schedule.

It is obvious that for identical jobs the choice of the next job to perform is of no importance – each job is represented by the same profile of power usage. As a result, the jobs can be scheduled in an arbitrary order, e.g. according to their increasing indices.

Let us denote by n_1 the number of jobs started at the moment 0 and by $s_{JL[n_1+1]}$ the moment when the next job from *JL* will be launched (calculated from (7)). The following property may also be relevant for the situation where, additionally, the number of charging connections is limited.

Property 3. The maximum number of jobs performed at a given moment does not exceed the number $n_1 + 1 + x$, where x is the maximum integer for which the inequality is met:

$$s_{JL[n_1+1]} + \sum_{i=1}^{x} \frac{1}{n_1 + i} < 1$$

4. Computational experiment

Simple priority rules can be used to set a suboptimal order of jobs in the *JL* list. The parameters that can be taken into account are the following: P_{0i}, d_i and the ratio P_{0i}/d_i. Of course, one can sort the jobs on the list according to non-decreasing or non-increasing values of these parameters.
In order to examine the suitability of individual priority rules, preliminary computational experiments were carried out. The assumptions of the experiments were as follows:

- number of jobs, $n = 12$; available power amount $P = 12$;
- values of P_{0i}, $i = 1, 2,\dots, n$, were chosen randomly according to discrete uniform distribution from the set $\{1,.., P_0^{\max}\}$, and $P_0^{\max}$ took the following two values in particular groups of experiments: 3 (a large number of jobs run in parallel in the resulting schedule), 8 (a small number of jobs executed in parallel in the resulting schedule)
- values d_i, $i = 1, 2,\dots, n$, were chosen randomly according to discrete uniform distribution from the set $\{1,.., d^{\max}\}$, and $d^{\max}$ took the following two values in particular groups of experiments: 12 (short jobs) and 50 (long jobs).

Ten test instances were generated for each case. Both~~:~~ non-decreasing and non-increasing values of the chosen parameters were tested. For each sequence of jobs in *JL*, the final schedule was generated by using Algorithm A. The results obtained in this way were compared to optimal solutions obtained by the full enumeration technique (all possible permutations of *n* jobs on a *JL* list) and with random sequence of jobs on *JL*. The obtained results of the experiment show that under the adopted assumptions for the considered scheduling problem, the best rule for ordering jobs on *JL* is according to non-increasing order of d_i. The representative results for the experiment with $P_0^{\max} = 8$ and $d^{\max} = 12$ are shown in Table 1, where average (Δ_{ave}) and maximum ($\Delta_{\max}$) deviations from the optimal solutions are presented for each tested rule.

Table 1. Exemplary results of the computational experiment

Rule:	random	$\uparrow P_{0i}$	$\downarrow P_{0i}$	$\uparrow d_i$	$\downarrow d_i$	$\uparrow P_{0i}/d_i$	$\downarrow P_{0i}/d_i$
Δ_{ave}	0.29	0.23	0.34	0.32	**0.06**	0.11	0.3
$\Delta_{\max}$	0.58	0.43	0.54	0.5	**0.14**	0.22	0.5

5. 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 shown 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, some priority rules can be applied to look for an optimal job permutation. We have performed computational tests to examine a few simple priority rules. They have shown that ordering the jobs according to their non-increasing processing times leads to the best suboptimal solutions.

References

Manwell J. F., McGowan J. G.,1993, "Lead acid battery storage model for hybrid energy systems", Solar Energy, vol. 50, pp 399 -405, 1993.

Błażewicz J., Ecker K., Pesch E., Schmidt G., Sterna M., Węglarz J., "Handbook on Scheduling: from Theory to Applications", Springer, Heidelberg, 2019.