Throughput Maximization in the Speed-Scaling Setting*

Eric Angel[1], **Evripidis Bampis**[2], **and Vincent Chau**[1]

1 **IBISC, Université d'Evry Val d'Essonne, Evry, France**
`{Eric.Angel,Vincent.Chau}@ibisc.univ-evry.fr`

2 **Sorbonne Universités, UPMC Univ Paris 06, LIP6, Paris, France**
`Evripidis.Bampis@lip6.fr`

Abstract

We are given a set of n jobs and a single processor that can vary its speed dynamically. Each job J_j is characterized by its processing requirement (work) p_j, its release date r_j and its deadline d_j. We are also given a budget of energy E and we study the scheduling problem of maximizing the throughput (i.e. the number of jobs that are completed on time). While the preemptive energy minimization problem has been solved in polynomial time [Yao et al., FOCS'95], the complexity of the problem of maximizing the throughput remained open until now. We answer partially this question by providing a dynamic programming algorithm that solves the problem in pseudo-polynomial time. While our result shows that the problem is not strongly NP-hard, the question of whether the problem can be solved in polynomial time remains a challenging open question. Our algorithm can also be adapted for solving the weighted version of the problem where every job is associated with a weight w_j and the objective is the maximization of the sum of the weights of the jobs that are completed on time.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problem

Keywords and phrases energy efficiency, dynamic speed scaling, offline algorithm, throughput, dynamic programming

Digital Object Identifier 10.4230/LIPIcs.STACS.2014.53

1 Introduction

The problem of scheduling n jobs with release dates and deadlines on a single processor that can vary its speed dynamically with the objective of minimizing the energy consumption has been first studied in the seminal paper by Yao et al. [12]. In this paper, we consider the problem of maximizing the throughput for a given budget of energy. Throughput is one of the most popular objectives in scheduling literature [5, 10]. Its maximization in the context of energy-related scheduling is very natural since mobile devices, such as mobile phones or computers, have a limited energy capacity depending on the quality of their battery. The maximization of the number of jobs or of the total weight of the jobs executed on time for a given budget of energy is of great importance. Different variants of the throughput maximization problem in the online setting have been studied in the literature, but surprisingly the status of the offline problem remained open.

Formally, we are given a set of n jobs $J = \{J_1, J_2, \ldots, J_n\}$, where each job J_j is characterized by its processing requirement (work) p_j, its release date r_j and its deadline d_j.

* Supported by the French Agency for Research under the DEFIS program TODO (ANR-09-EMER-010) and by the project PHC CAI YUANPEI (27927VE)

We consider integer release dates, deadlines and processing requirements. (For simplicity, we suppose that the earliest released job is released at $t = 0$.) We assume that the jobs have to be executed by a single speed-scalable processor, i.e. a processor which can vary its speed over time (at a given time, the processor's speed can be any non-negative value). The processor can execute at most one job at each time. We measure the processor's speed in units of executed work per unit of time. If $s(t)$ denotes the speed of the processor at time t, then the total amount of work executed by the processor during an interval of time $[t, t')$ is equal to $\int_t^{t'} s(u)du$. Moreover, we assume that the processor's power consumption is a convex function of its speed. Specifically, at any time t, the power consumption of the processor is $P(t) = s(t)^\alpha$, where $\alpha > 1$ is a constant. Since the power is defined as the rate of change of the energy consumption, the total energy consumption of the processor during an interval $[t, t')$ is $\int_t^{t'} s(u)^\alpha du$. Note that if the processor runs at a constant speed s during an interval of time $[t, t')$, then it executes $(t' - t) \cdot s$ units of work and it consumes $(t' - t) \cdot s^\alpha$ units of energy.

Each job J_j can start being executed after or at its release date r_j. Moreover, the execution of a job may be suspended and continued later from the point of suspension. Given a budget of energy E, our objective is to find a schedule of maximum throughput whose energy does not exceed the budget E, where the throughput of a schedule is defined as the number of jobs which are completed on time, i.e. before their deadline. Observe that a job is completed on time if it is entirely executed during the interval $[r_j, d_j)$. By extending the well-known 3-field notation, this problem can be denoted as $1|pmtn, r_j, E|\sum U_j$. We also consider the weighted version of the problem where every job J_j is also associated with a weight w_j and the objective is no more the maximization of the cardinality of the jobs that are completed on time, but the maximization of the sum of their weights. We denote this problem as $1|pmtn, r_j, E|\sum w_j U_j$. In what follows, we consider the problem in the case where all jobs have arbitrary integer release dates, deadlines and processing requirement.

1.1 Related Works and our Contribution

A series of papers appeared for some online variants of throughput maximization: the first work that considered throughput maximization and speed scaling in the online setting has been presented by Chan et al. [6]. They considered the single processor case with release dates and deadlines and they assumed that there is an upper bound on the processor's speed. They are interested in maximizing the throughput, and minimizing the energy among all the schedules of maximum throughput. They presented an algorithm which is $O(1)$-competitive with respect to both objectives. Li [11] has also considered the maximum throughput when there is an upper bound in the processor's speed and he proposed a 3-approximation greedy algorithm for the throughput and a constant approximation ratio for the energy consumption. In [3], Bansal et al. improved the results of [6], while in [9], Lam et al. studied the 2-processors environment. In [8], Chan et al. defined the energy efficiency of a schedule to be the total amount of work completed in time divided by the total energy usage. Given an efficiency threshold, they considered the problem of finding a schedule of maximum throughput. They showed that no deterministic algorithm can have competitive ratio less than Δ, the ratio of the maximum to the minimum jobs' processing requirement. However, by decreasing the energy efficiency of the online algorithm the competitive ratio of the problem becomes constant. Finally, in [7], Chan et al. studied the problem of minimizing the energy plus a rejection penalty. The rejection penalty is a cost incurred for each job which is not completed on time and each job is associated with a value which is its importance. The authors proposed an $O(1)$-competitive algorithm for the case where the speed is unbounded and they showed that no $O(1)$-competitive algorithm exists for the case where the speed is bounded. In what follows, we focus on the complexity status of the offline case for general instances. Angel et al. [1] were the first to consider the throughput maximization problem in the energy setting for the offline case. They studied the problem for a particular family of instances where the jobs have agreeable deadlines, i.e. for every pair of jobs J_i and J_j, $r_i \leq r_j$ if and only if $d_i \leq d_j$. They provided a polynomial time algorithm to solve the problem for agreeable instances. However, to the best of our knowledge, the complexity of the unweighted preemptive problem for arbitrary instances remained unknown until now. In this paper, we prove that there is a pseudo-polynomial time algorithm for solving the problem optimally. For the weighted version, the problem is $\mathcal{NP}$-hard even for instances in which all the jobs have common release dates and deadlines. Angel et al. [1] showed that the problem admits a pseudo-polynomial time algorithm for agreeable instances. Our algorithm for the unweighted case can be adapted for the weighted throughput problem with arbitrary release dates and deadlines solving the problem in pseudo-polynomial time. More recently, Antoniadis et al. [2] considered a generalization of the classical knapsack problem where the objective is to maximize the total profit of the chosen items minus the cost incurred by their total weight. The case where the cost functions are convex can be translated in terms of a weighted throughput problem where the objective is to select the most profitable set of jobs taking into account the energy costs. Antoniadis et al. presented a FPTAS and a fast 2-approximation algorithm for the non-preemptive problem where the jobs have no release dates or deadlines.

We present in this paper an optimal algorithm for throughput maximization when the preemption of jobs is allowed.

2 Preliminaries

Among the schedules of maximum throughput, we try to find the one of minimum energy consumption. Therefore, if we knew by an oracle the set of jobs J^*, $J^* \subseteq J$, which are completed on time in an optimal solution, we would simply have to apply an optimal algorithm for $1|pmtn, r_j, d_j|E$ for the jobs in J^* in order to determine a minimum energy schedule of maximum throughput for our problem. Such an algorithm has been proposed in [12]. Based on this observation, we can use in our analysis some properties of an optimal schedule for $1|pmtn, r_j, d_j|E$.

Let $t_1, t_2, \ldots, t_K$ be the time points which correspond to release dates and deadlines of the jobs so that for each release date and deadline there is a t_i value that corresponds to it. We number the t_i values in increasing order, i.e. $t_1 < t_2 < \ldots < t_K$. The following theorem is a consequence of the algorithm of Yao et al. [12] and was proved in [4].

▶ **Theorem 1.** *A feasible schedule for $1|pmtn, r_j, d_j|E$ is optimal if and only if all the following hold:*
1. *Each job J_j is executed at a constant speed s_j.*
2. *The processor is not idle at any time t such that $t \in (r_j, d_j]$, for all $J_j \in J$.*
3. *The processor runs at a constant speed during any interval $(t_i, t_{i+1}]$, for $1 \leq i \leq K-1$.*
4. *If others jobs are scheduled in the span $[r_j, d_j]$ of J_j, then their speed is necessarily greater or equal to the speed of J_j.*

Theorem 1 is also satisfied by the optimal schedule of $1|pmtn, r_j, E|\sum U_j$ for the jobs in J^*. In the following, we suppose that the jobs are sorted in non-decreasing order of their deadlines (EDF order), i.e. $d_1 \leq d_2 \leq \ldots \leq d_n$. Moreover, we suppose that the release dates, the deadlines and the processing requirements are integer.

▶ **Definition 2.** Let $J(k,s,t) = \{J_j \mid j \leq k \text{ and } s \leq r_j < t\}$ be the set of jobs, among the k first ones w.r.t. the EDF order, whose release dates are within s and t.

▶ **Lemma 3.** *The total period in which the processor runs at a same speed in an optimal solution for $1|pmtn, r_j, d_j|E$ has an integer length.*

Proof. The total period is defined by a set of intervals $(t_i, t_{i+1}]$ for $1 \leq i \leq K-1$ thanks to the property 3) in Theorem 1. Since each t_i corresponds to some release date or some deadline, then $t_i \in \mathbb{N},\ 1 \leq i \leq K$. Thus every such period has necessarily an integer length. ◀

▶ **Definition 4.** Let $L = d_{max} - r_{min}$ be the span of the whole schedule. To simplify the notation, we assume that $r_{min} = 0$.

▶ **Definition 5.** Let $P = \sum_j p_j$ be the total processing requirement of all the jobs.

▶ **Definition 6.** We call an EDF schedule, a schedule in which at any time, the processor schedules the job that has the smallest deadline among the set of available jobs at this time.

In the sequel, all the considered schedules are EDF schedules.

3 The Dynamic Program and its Correctness

In this part, we propose an optimal algorithm which is based on dynamic programming depending on the span length L and the total processing requirement P. As mentioned previously, among the schedules of maximum throughput, our algorithm constructs a schedule with the minimum energy consumption.

For a subset of jobs $S \subseteq J$, a schedule which involves only the jobs in S will be called a S-schedule.

▶ **Definition 7.** Let $G_k(s,t,u)$ be the minimum energy consumption of a S-schedule with $S \subseteq J(k,s,t)$ such that $|S| = u$ and such that the jobs in S are entirely scheduled in $[s,t]$.

Given a budget of energy E that we cannot exceed, the objective function is
$\max\{u \mid G_n(0, d_{max}, u) \leq E;\ 0 \leq u \leq n\}$.

▶ **Definition 8.** Let $F_{k-1}(x,y,u,\ell,i,a,h)$ be the minimum energy consumption of a S-schedule with $S \subseteq J(k-1,x,y)$ such that $|S| = u$ and such that the jobs in S are entirely scheduled in $[x,y]$ during at most $a + h \times \frac{\ell}{i}$ unit times. Moreover, we assume that each maximal block of consecutive jobs of S starts at a release date and has a length equal to $a' + h' \times \frac{\ell}{i}$ with $a', h' \in \mathbb{N}$.

Next, we define the set of all important dates of an optimal schedule in which every job can start and end, and we show that the size of this set is pseudo-polynomial.

▶ **Definition 9.** Let $\Omega = \{r_j \mid j = 1, \ldots, n\} \cup \{d_j \mid j = 1, \ldots, n\}$.

▶ **Definition 10.** Let $\Phi = \{s + h \times \frac{\ell}{i} \leq L \mid i = 1, \ldots, P;\ h = 0, \ldots, i;\ s = 0, \ldots, L;\ \ell = 1, \ldots, L\}$

▶ **Proposition 11.** *There exists an optimal schedule $\mathcal{O}$ in which for each job, its starting times and finish times belong to the set Φ, and such that each job is entirely executed with a speed $\frac{i}{\ell}$ for some $i = 1, \ldots, P$ and $\ell = 1, \ldots, L$.*

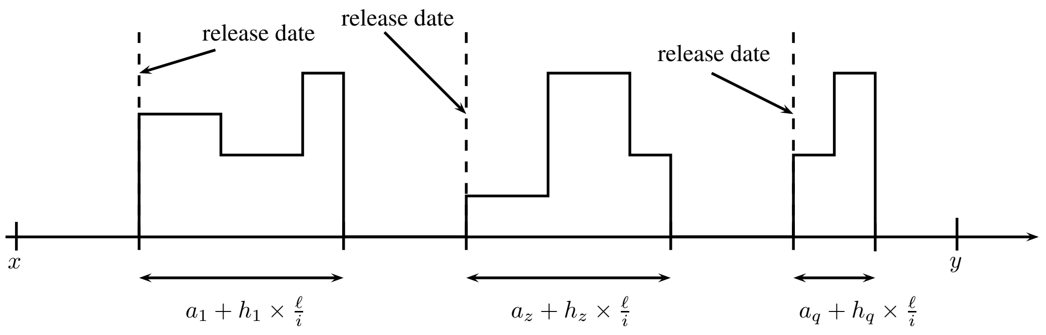
release date
release date
release date
x
y
$a_1 + h_1 \times \frac{\ell}{i}$
$a_z + h_z \times \frac{\ell}{i}$
$a_q + h_q \times \frac{\ell}{i}$

Figure 1 Illustration of $F_{k-1}(x, y, u, \ell, i, a, h)$ in Definition 8 with respect to $\sum_{z=1}^{q} a_z + h_z \times \frac{\ell}{i} = a + h \times \frac{\ell}{i}$.

Proof. W.l.o.g. we can consider that each job has a unit processing requirement. If it is not the case, we can split a job J_j into p_j jobs, each one with a unit processing requirement.

We briefly explain the algorithm proposed in [12] which gives an optimal schedule. At each step, it selects the (critical) interval $I = [s, t]$ with s and $t > s$ in $\Omega = \{r_j \mid j = 1, \dots, n\} \cup \{d_j \mid j = 1, \dots, n\}$, such that $s_I = \frac{|\{J_j \mid s \le r_j \le d_j \le t\}|}{t-s}$ is maximum. All the jobs inside this interval are executed at the speed s_I, which is of the form $\frac{i}{\ell}$ for some $i = 1, \dots, P$ and $\ell = 1, \dots, L$, and according to the EDF order. This interval cannot be used any more, and we recompute a new critical interval without considering the jobs and the previous critical intervals, until all the jobs have been scheduled.

We can remark that the length of each critical interval (at each step) $I = [s, t]$ is an integer. This follows from the fact that $s = r_z \in \mathbb{N}$ for some job J_z, and $t = d_j \in \mathbb{N}$ for some job J_j, moreover we remove integer lengths at each step (the length of previous critical intervals which intersect the current one), so the new considered critical interval has always an integer length.

Then we can define every potential starting time or completion time of each job in this interval. We first prove that the completion time of a job in a continuous critical interval, i.e. a critical interval which has an empty intersection with all other critical intervals, belongs to Φ. Let J_k be any job in a continuous critical interval and let x and y be respectively its starting and completion times. Then there is no idle time between $s = r_f$ (for some J_f) and y since it is a critical interval. Let $v = \frac{i}{\ell}$ be the processor speed in this interval and $p = \frac{\ell}{i}$ be the processing time of a job (Recall that each job has the same processing requirement). The jobs that are executed (even partially) between x and y are not executed neither before x nor after y since we consider an EDF schedule. Thus $y - x$ is a multiple of p. Two cases may occur:

- Either J_k causes a preemption and hence $x = r_k$,
- or J_k does not cause any preemption and hence the jobs that are executed between s and x, are fully scheduled in this interval. Consequently, $x - s$ is a multiple of p.

In both cases, there is a release date r_g (either r_k or r_f) such that between r_g and y, the processor is never idle and such that y is equal to r_g modulo p. On top of that, the distance between r_g and t is not greater than $n \times p$. Hence, $y \in \Phi$. Now consider the starting time of any job. This time point is either the release date of the job or is equal to the completion time of the "previous" one. Thus, starting times also belong to Φ.

Now we consider the starting and the completion times of a job in a critical interval I in which there is at least another critical interval (with greater speeds) included in I or intersecting I. Let A be the union of those critical intervals. Since the jobs of I cannot be scheduled during the intervals A, the starting time and completion time of these jobs have to be (right)-shifted by an integer value (since each previously critical interval has an integer length). Thus the starting time and completion time of all the jobs still belong to Φ. ◀

▶ **Proposition 12.** *One has*

$$G_k(s,t,u) = \min \begin{cases} G_{k-1}(s,t,u) \\ \min\limits_{\substack{x\in\Phi \\ 0\le u_1\le u \\ 0\le u_2\le u \\ 0\le u_1+u_2\le u-1 \\ 0\le a\le L;\ 1\le \ell\le L \\ 1\le i\le P;\ 0\le h\le P \\ y-x=a+(p_k+h)\frac{\ell}{i} \\ r_k\le x\le y\le d_k}} \left\{ \begin{array}{l} G_{k-1}(s,x,u_1) + F_{k-1}(x,y,u_2,\ell,i,a,h) \\ +\left(\frac{i}{\ell}\right)^{\alpha-1} p_k + G_{k-1}(y,t,u-u_1-u_2-1) \end{array} \right\} \end{cases}$$

$G_0(s,t,0) = 0\ \forall s,t \in \Phi$
$G_0(s,t,u) = +\infty\ \forall s,t \in \Phi$ *and* $u > 0$

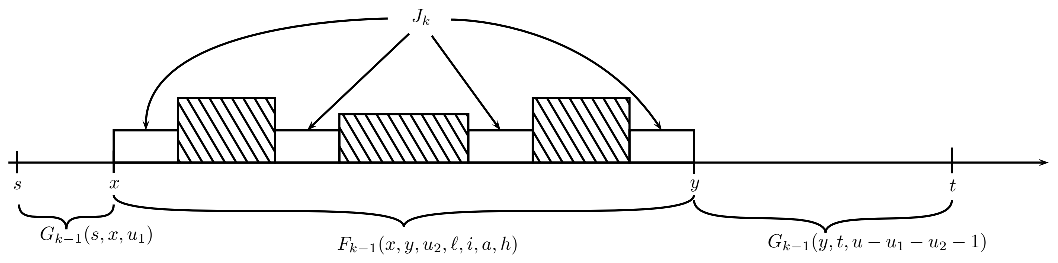

Figure 2 Illustration of Proposition 12 where x is the first starting time of J_k and y is the last completion time of J_k.

Proof. Let G' be the right hand side of the formula, G'_1 be the first line of G' and G'_2 be the second line of G'.

We first prove that $G_k(s,t,u) \le G'$.

Since $J(k-1,s,t) \subseteq J(k,s,t)$, then $G_k(s,t,u) \le G_{k-1}(s,t,u) = G'_1$.

Now consider a schedule $\mathcal{S}_1$ that realizes $G_{k-1}(s,x,u_1)$, a schedule $\mathcal{S}_2$ that realizes $F_{k-1}(x,y,u_2,\ell,i,a,h)$ such that $y-x = a + (p_k+h) \times \frac{\ell}{i}$ and a schedule $\mathcal{S}_3$ that realizes $G_{k-1}(y,t,u-u_1-u_2-1)$. We build a schedule with $\mathcal{S}_1$ from s to x, with $\mathcal{S}_2$ from x to y and with $\mathcal{S}_3$ from y to t.

Since $F_{k-1}(x,y,u_2,\ell,i,a,h)$ is a schedule where the processor executes the jobs during at most $a + h \times \frac{l}{i}$ unit times and we have $y-x = a+(p_k+h) \times \frac{\ell}{i}$, then there is at least $p_k \times \frac{\ell}{i}$ units time for J_k. Thus J_k can be scheduled with speed $\frac{i}{\ell}$ during $[x,y]$.

Obviously, the subsets $J(k-1,s,x), J(k-1,x,y)$ and $J(k-1,y,t)$ do not intersect, so this is a feasible schedule, and its cost is G'_2. Hence $G_k(s,t,u) \le G'_2$.

We now prove that $G' \le G_k(s,t,u)$.

If $J_k \notin \mathcal{O}$ such that $\mathcal{O}$ realizes $G_k(s,t,u)$, then $G'_1 = G_k(s,t,u)$.

Now, let us consider the case $J_k \in \mathcal{O}$.

We denote by $\mathcal{X}$ the schedule that realizes $G_k(s,t,u)$ in which the first starting time x of J_k is maximal, and in which y is the last completion time of J_k is also maximal. According to Proposition 11, we assume that $x, y \in \Phi$. We split $\mathcal{X}$ (which is an EDF schedule) into three sub-schedules $\mathcal{S}_1 \subseteq J(k-1,s,x)$, $\mathcal{S}_2 \subseteq J(k-1,x,y) \cup \{J_k\}$ and $\mathcal{S}_3 \subseteq J(k-1,y,t)$.

We claim that we have the following properties:

P1) all the jobs of $\mathcal{S}_1$ are released in $[s,x]$ and are completed before x,
P2) all the jobs of $\mathcal{S}_2$ are released in $[x,y]$ and are completed before y,
P3) all the jobs of $\mathcal{S}_3$ are released in $[y,t]$ and are completed before t.

We prove P1

Suppose that there is a job $J_j \in \mathcal{S}_1$ which is not completed before x. Then we can swap some part of J_j of length ε which is scheduled after x with some part of J_k of length ε at time x. This can be done since we have $d_j \leq d_k$. Thus we have a contradiction with the fact that x was maximal.

We prove P2

Similarly, suppose that there is a job $J_j \in \mathcal{S}_2$ which is not completed before y. Then we can swap some part of J_j of length ℓ which is scheduled after y with some part of J_k of length ℓ in $[x,y]$. This can be done since we have $d_j \leq d_k$. Thus we have a contradiction with the fact that y was maximal.

We prove P3

If there exists a job in $\mathcal{S}_3$ which is not entirely executed at time t, then the removal of this job would lead to a lower energy consumption schedule for $\mathcal{S}_3$ with the same throughput value. This contradicts the definition of $G_{k-1}(y,t,|\mathcal{S}_3|)$.

Let us now consider the schedule $\mathcal{S}'_2 = \mathcal{S}_2 \setminus J_k$ in $[x,y]$. Since $[x,y] \subseteq [r_k, d_k]$, thanks to property 4) of Theorem 1, the speeds of jobs in $\mathcal{S}'_2$ are necessarily greater than or equal to the speed of J_k. Let us consider any maximal block b of consecutive jobs in $\mathcal{S}'_2$. This block can be partitioned into two sub-blocks b_1 and b_2 such that b_1 (resp. b_2) contains all the jobs of b which are scheduled with a speed equal to (resp. strictly greater than) the speed of J_k. All the jobs scheduled in block b are also totally completed in b (this comes from the EDF property and because J_k has the biggest deadline). Notice that the speed of J_k is equal to $\frac{i}{\ell}$ for some value $i = 1, \dots, P$ and $\ell = 1, \dots, L$ thanks to Proposition 11. Thus the total processing time of b_1 is necessarily $h' \times \frac{\ell}{i}$. Moreover since from property 3 of Theorem 1, all the speed changes occur at time $t_i \in \mathbb{N}$, the block b_2 has an integer length. Therefore, every block b has a length equal to $a' + h' \times \frac{\ell}{i}$ and the total processing time of $\mathcal{S}'_2$ is $a + h \times \frac{\ell}{i}$. Furthermore, every block b in $\mathcal{S}'_2$ starts at a release date (this comes from the EDF property). On top of that, we have $y - x = a + (h + p_k) \times \frac{\ell}{i}$ with $a = 0, 1, \dots, L$ and $h = 0, \dots, i$. Moreover, every block b in $\mathcal{S}'_2$ starts at a release date (this comes from the EDF property). Hence the cost of the schedule $\mathcal{S}'_2$ is greater than $F_{k-1}(x, y, |\mathcal{S}'_2|, \ell, i, a, h)$. The energy consumption of J_k is exactly $p_k \times (\frac{i}{\ell})^{\alpha-1}$.

Similarly, the cost of the schedule $\mathcal{S}_1$ is greater than $G_{k-1}(s,x,|\mathcal{S}_1|)$ and the cost of $\mathcal{S}_3$ is greater than $G_{k-1}(y,t,|\mathcal{S}_3|)$.

Therefore, $G_k(s,t,u) \geq G_{k-1}(s,x,|\mathcal{S}_1|) + F_{k-1}(x,y,|\mathcal{S}_2|,\ell,i,a,h) + G_{k-1}(y,t,|\mathcal{S}_3|) + p_k\left(\frac{i}{\ell}\right)^{\alpha-1} = G'_2$ and $G_k(s,t,u) \geq G'$. ◀

▶ **Proposition 13.** *One has*

$$F_{k-1}(x,y,u,\ell,i,a,h) = \min_{\substack{0\le a'\le a;\ 0\le h'\le h\\ x\le x'=r_j\le y;\ j\le k\\ 1\le\beta\le u\\ y'=x'+a'+h'\times\frac{\ell}{i}\le y}} \{G_{k-1}(x',y',\beta)+F_{k-1}(y',y,u-\beta,\ell,i,a-a',h-h')\}$$

$$F_{k-1}(x,y,0,\ell,i,a,h) = 0$$
$$F_{k-1}(x,y,u,\ell,i,0,0) = +\infty$$

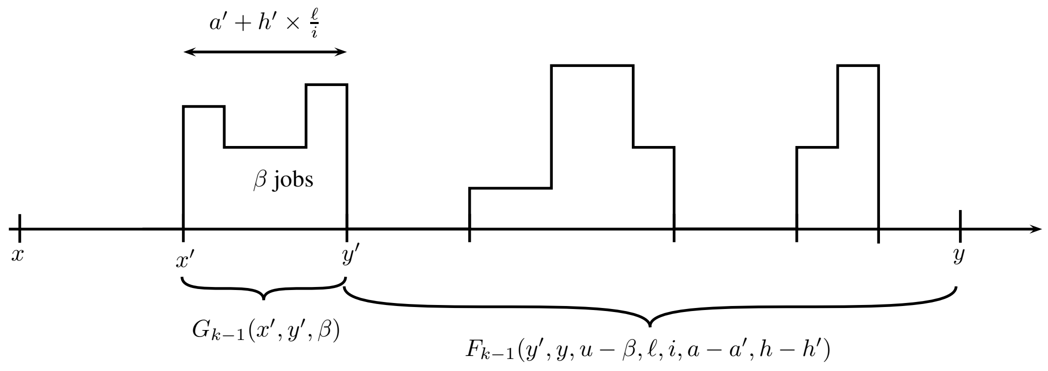
$a' + h' \times \frac{\ell}{i}$
β jobs
x
x'
y'
y
$G_{k-1}(x',y',\beta)$
$F_{k-1}(y',y,u-\beta,\ell,i,a-a',h-h')$

Figure 3 Illustration of Proposition 13.

Proof. Let F' be the right hand side of the equation.
We first prove that $F_{k-1}(x,y,u,\ell,i,a,h) \le F'$.

Let us consider a schedule $\mathcal{S}_1$ that realizes $G_{k-1}(x',y',\beta)$ and a schedule $\mathcal{S}_2$ that realizes $F_{k-1}(y',y,u-\beta,\ell,i,a',h')$. We suppose that the processor is idle during $[x,x']$. We build a schedule with an empty set from x to x', with $\mathcal{S}_1$ from x' to y' and with $\mathcal{S}_2$ from y' to y.

Obviously, the subsets $J(k-1,x,z)$ and $J(k-1,z,y)$ do not intersect, so this is a feasible schedule, and its cost is F', thus $F_{k-1}(x,y,u,f,\ell,i) \le F'$.

We now prove that $F' \le F_{k-1}(x,y,u,\ell,i,a,h)$.

Let $\mathcal{O}$ be an optimal schedule that realizes $F_{k-1}(x,y,u,\ell,i,a,h)$ such that x' is the first starting time of the schedule and y' is the completion time of the first block of jobs in $\mathcal{O}$. We split it into two sub-schedules $\mathcal{S}_1 \subseteq J(k-1,x',y')$ and $\mathcal{S}_2 \subseteq J(k-1,y',y)$ such that the value of x' is maximal and the value of y' is also maximal.

Then $y'-x' = a'+h'\times\frac{\ell}{i}$ for some value $a' = 0,\dots,a$ and $h' = 0,\dots,h$ by definition. Thus we can assume that the jobs in $\mathcal{S}_2$ have to be scheduled during at most $(a-a')+(h-h')\times\frac{\ell}{i}$ units of time in $[y',y]$. We claim that x' is a release date by definition.

Moreover, we claim that all the jobs of $\mathcal{S}_2$ are released in $[y',y]$ and are completed before y. If there exists a job in $\mathcal{S}_2$ which is not completed at time t, then the removal of this job would lead to a lower energy consumption schedule for $\mathcal{S}_2$ which contradicts the definition of $F_{k-1}(y',y,|\mathcal{S}_2|,\ell,i,a-a',h-h')$.

Then the restriction $\mathcal{S}_1$ of $\mathcal{O}$ in $[x',y']$ is a schedule that meets all constraints related to $G_{k-1}(x',y',|\mathcal{S}_1|)$. Hence its cost is greater than $G_{k-1}(x',y',|\mathcal{S}_1|)$. Similarly, the restriction $\mathcal{S}_2$ of $\mathcal{O}$ to $[y',y]$ is a schedule that meets all constraints related to $F_{k-1}(y',y,|\mathcal{S}_2|,\ell,i,a-a',h-h')$.

Thus $F' \le F_{k-1}(x,y,u,\ell,i,a,h)$. ◀

▶ **Theorem 14.** *The preemptive throughput maximization problem can be solved in* $O(n^6 L^9 P^9)$ *time and in* $O(nL^6 P^6)$ *space.*

Proof. The values of $G_k(s,t,u)$ are stored in a multi-dimensional array of size $O(|\Phi|^2 n^2)$. Each value need $O(|\Phi| n^2 L^2 P^2 r(F))$ time to be computed where $r(F)$ is the running time for computing $F_{k-1}(x,y,u,\ell,i,a,h)$. Since we fix every value of x,y,u,ℓ,i,a,h in the minimization step, the table F does not need to be pre-computed. Then the running time is $O(n^2 LP)$ for each value of F. Therefore, the total running time of the dynamic programming is $O(n^6 L^9 P^9)$. Moreover, the values of $F_{k-1}(x,y,u,\ell,i,a,h)$ are stored in a multi-dimensional array (since we don't need to remember the F_i values for $i < k-1$) of size $O(n|\Phi|^2 L^2 P^2) = O(nL^6 P^6)$. ◀

The dynamic program can be adapted for the weighted version of the problem and has a running time of $O(n^2 W^4 L^9 P^9)$ where W is the sum of the weight of all jobs. This can be done by considering the total weight of completed jobs of a schedule instead of considering the number of completed jobs. More formally, this can be done by modifying the definitions of G_k and F_{k-1} in the following way:

- $G_k(s,t,w)$ is the minimum energy consumption of a S-schedule with $S \subseteq J(k,s,t)$ such that $\sum_{J_j \in S} w_j \geq w$ and such that the jobs in S are entirely scheduled in $[s,t]$, and
- $F_{k-1}(x,y,w,\ell,i,a,h)$ is the minimum energy consumption of a S-schedule with $S \subseteq J(k-1,x,y)$ such that $\sum_{J_j \in S} w_j \geq w$ and such that the jobs in S are entirely scheduled in $[x,y]$ during at most $a + h \times \frac{\ell}{i}$ unit times. As for the cardinality case, we assume that each maximal block of consecutive jobs of S starts at a release date and has a length equal to $a' + h' \times \frac{\ell}{i}$ with $a', h' \in \mathbb{N}$.

4 Conclusion

In this paper, we proved that there is a pseudo-polynomial time algorithm for solving the problem optimally. This result is a first (partial) answer to the complexity status of the throughput maximization problem in the offline setting. Our result shows that the problem is not strongly NP-hard, but the question of whether there is a polynomial time algorithm for it remains a challenging open question.

References

1 Eric Angel, Evripidis Bampis, Vincent Chau, and Dimitrios Letsios. Throughput maximization for speed-scaling with agreeable deadlines. In T.-H. Hubert Chan, Lap Chi Lau, and Luca Trevisan, editors, *TAMC*, volume 7876 of *Lecture Notes in Computer Science*, pages 10–19. Springer, 2013.

2 Antonios Antoniadis, Chien-Chung Huang, Sebastian Ott, and José Verschae. How to pack your items when you have to buy your knapsack. In Krishnendu Chatterjee and Jiri Sgall, editors, *MFCS*, volume 8087 of *Lecture Notes in Computer Science*, pages 62–73. Springer, 2013.

3 Nikhil Bansal, Ho-Leung Chan, Tak Wah Lam, and Lap-Kei Lee. Scheduling for speed bounded processors. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP (1)*, volume 5125 of *Lecture Notes in Computer Science*, pages 409–420. Springer, 2008.

4 Nikhil Bansal, Tracy Kimbrel, and Kirk Pruhs. Speed scaling to manage energy and temperature. *J. ACM*, 54(1), 2007.

5 Peter Brucker. *Scheduling Algorithms*. Springer Publishing Company, Incorporated, 5th edition, 2010.

6 Ho-Leung Chan, Wun-Tat Chan, Tak Wah Lam, Lap-Kei Lee, Kin-Sum Mak, and Prudence W. H. Wong. Energy efficient online deadline scheduling. In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *SODA*, pages 795–804. SIAM, 2007.

7 Ho-Leung Chan, Tak Wah Lam, and Rongbin Li. Tradeoff between energy and throughput for online deadline scheduling. In Klaus Jansen and Roberto Solis-Oba, editors, *WAOA*, volume 6534 of *Lecture Notes in Computer Science*, pages 59–70. Springer, 2010.

8 Joseph Wun-Tat Chan, Tak Wah Lam, Kin-Sum Mak, and Prudence W. H. Wong. Online deadline scheduling with bounded energy efficiency. In Jin yi Cai, S. Barry Cooper, and Hong Zhu, editors, *TAMC*, volume 4484 of *Lecture Notes in Computer Science*, pages 416–427. Springer, 2007.

9 Tak Wah Lam, Lap-Kei Lee, Isaac Kar-Keung To, and Prudence W. H. Wong. Energy efficient deadline scheduling in two processor systems. In Takeshi Tokuyama, editor, *ISAAC*, volume 4835 of *Lecture Notes in Computer Science*, pages 476–487. Springer, 2007.

10 E. L. Lawler. A dynamic programming algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. *Annals of Operations Research*, 26:125–133, 1990.

11 Minming Li. Approximation algorithms for variable voltage processors: Min energy, max throughput and online heuristics. *Theor. Comput. Sci.*, 412(32):4074–4080, 2011.

12 F. Frances Yao, Alan J. Demers, and Scott Shenker. A scheduling model for reduced CPU energy. In *FOCS*, pages 374–382. IEEE Computer Society, 1995.