Throughput Maximization for Speed-Scaling with Agreeable Deadlines*

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

[1] IBISC ; Université d'Évry, Évry, France
{Eric.Angel,Vincent.Chau,Dimitris.Letsios}@ibisc.univ-evry.fr
[2] LIP6 ; Université Pierre et Marie Curie; 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 which are completed on time). We show that the problem can be solved by dynamic programming when all the jobs are released at the same time in $O(n^4 \log n \log P)$, where P is the sum of the processing requirements of the jobs. For the more general case of agreeable deadlines, where the jobs can be ordered such that for every $i < j$, both $r_i \leq r_j$ and $d_i \leq d_j$, we propose a dynamic programming algorithm solving the problem optimally in $O(n^6 \log n \log P)$. In addition, we consider the weighted case where every job j is also associated with a weight w_j and we are interested in maximizing the weighted throughput. For this case, we prove that the problem becomes $\mathcal{NP}$-hard in the ordinary sense and we propose a pseudo-polynomial time algorithm.

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, Demers and Shenker [3]. In this paper, we consider the problem of maximizing the throughput for a given budget of energy. 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. (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

* This work has been supported by the ANR project TODO (09-EMER-010), by PHC CAI YUANPEI (27927VE) and by the ALGONOW project of the THALES program.

T-H.H. Chan, L.C. Lau, and L. Trevisan (Eds.): TAMC 2013, LNCS 7876, pp. 10–19, 2013.

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, we allow the preemption of jobs, i.e. 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 by Graham et al. [2], this problem can be denoted as $S1|pmtn, r_j|\sum U_j(E)$. We also consider the weighted version of the problem where every job 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 $S1|pmtn, r_j|\sum w_j U_j(E)$. In what follows, we consider the problem in the case where either all jobs have a release date equal to 0 and for an important family of instances, the agreeable instances for which the jobs can be ordered such that for every $i < j$, both $r_i \leq r_j$ and $d_i \leq d_j$.

1.1 Related Works and Our Contribution

Up to the best of our knowledge no work exists for the off-line case of our problem. On the contrary, some works exist 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. [9]. 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. In [8] Bansal et al. improved the results of [9], while in [13], Lam et al. studied the 2-processor environment. In [11], 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 [10], 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.

The paper is organized as follows: we first present an optimal algorithm for the case where all the jobs are released at time 0, and then we present another algorithm for the more general case with agreeable deadlines. The reason of presenting both these cases is that in the first case we have a complexity of $O(n^4 \log n \log P)$ which is better than the one in the second case where the complexity becomes $O(n^6 \log n \log P)$. Finally, we consider the weighted case where we are interested in maximizing the weighted throughput. For this case, we prove that the problem is $\mathcal{NP}$-hard in the ordinary sense and we propose a pseudo-polynomial time algorithm.

2 Preliminaries

Given that the processor's speed can be varied, a reasonable distinction of the scheduling problems that can be considered is the following:

- **FS** (Fixed Speed): The processor has a fixed speed which implies directly a processing time for each job. In this case, the scheduler has to decide which job must be executed at each time. This is the classical scheduling setting.
- **CS** (Constant Speed): The processor's speed is not known in advance but it can only run at a single speed during the whole time horizon. In this context, the scheduler has to define a single value of speed at which the processor will run and the job executed at each time.
- **SS** (Scalable Speed): The processor's speed can be varied over the time and, at each time, the scheduler has to determine not only which job to run, but the processor's speed as well.

3 Properties of the Optimal Schedule

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 $S1|pmtn, r_j, d_j|E$ for the jobs in J^* in order to determine a minimum energy schedule of maximum throughput for our problem. Based on this observation, we can use in our analysis some properties of an optimal schedule for $S1|pmtn, r_j, d_j|E$.

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 comes from [3].

Theorem 1. *A feasible schedule for $S1|pmtn, r_j, d_j|E$ 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. *A job J_j is executed during any interval $(t_i, t_{i+1}]$ $(1 \leq i \leq k-1)$, if it has been assigned the maximum speed among the speeds of the jobs $J_{j'}$ with $(t_i, t_{i+1}] \subseteq (r_{j'}, d_{j'}]$.*

Theorem 1 is also satisfied by the optimal schedule of $S1|pmtn, r_j|\sum U_j(E)$ for the jobs in J^*.

4 Agreeable Deadlines

For the special case of the problem $S1|pmtn, r_j|\sum U_j(E)$ where the deadlines of the jobs are agreeable we propose an optimal algorithm which is based on dynamic programming. As mentioned before, among the schedules of maximum throughput, our algorithm constructs a schedule of minimum energy consumption. Next, we describe our dynamic program and we elaborate on the complexity of our algorithm.

Initially, we consider the problem $1|pmtn, r_j|\sum U_j$ which is a classical scheduling problem where we are given a set of jobs $J = \{J_1, J_2, \ldots, J_n\}$ that have to be executed by a single processor. Each job J_j is associated with a processing time p_j, a release date r_j and a deadline d_j. The objective is to find a schedule of maximum throughput. We refer to the problem as FS. This problem is polynomially-time solvable and the fastest known algorithm for general instances is in $O(n^4)$ [1]. When all the release dates are equal, this problem can be solved in $O(n \log n)$ with Moore's algorithm [5]. Finally, if the jobs have agreeable deadlines, the time complexity is also in $O(n \log n)$ using Lawler's algorithm [7].

Next, we consider another problem which we denote as CS. In this problem we are given a set of jobs $J = \{J_1, J_2, \ldots, J_n\}$, where each job J_j has a processing requirement p_j, a release date r_j and a deadline d_j, that have to be executed by a single speed scalable processor. Moreover, we are given a value of throughput k. The objective is to find the minimum energy schedule which completes at least k jobs on time so that all jobs that are completed on time are assigned equal speed and the jobs not completed on time have zero speed. For notational convenience, we denote the problem $S1|pmtn, r_j|\sum U_j(E)$ as SS.

The inspiration for our dynamic programming for the special case of the SS where the deadlines are agreeable was the fact that the problem CS can be solved in polynomial time by repeatedly solving instances of the problem FS. In fact, if we are given a candidate speed s for the CS problem, we can find a schedule of maximum throughput w.r.t. to s simply by setting the processing time of each job J_j equal to $\frac{p_j}{s}$ and applying an optimal algorithm for the FS problem. So, in order to get an optimal algorithm of the CS problem, it suffices to establish a

lower and upper bound on the speed of the optimal schedule. A naive choice is $s_{min} = 0$ and $s_{max} = \infty$. Then, it suffices to binary search in $[s_{min}, s_{max}]$ and find the minimum speed s^* in which k jobs are completed on time.

Property 1. There exists an optimal solution in which all jobs are scheduled according to EDF (Earliest Deadline First) order and without preemption.

This property comes from the fact that the algorithm of [3] is optimal and that we have an agreeable instance.

In the following, we assume that the jobs $J_1, J_2, \ldots, J_n$ are sorted according to the EDF order, i.e. $d_1 \leq d_2 \leq \ldots \leq d_n$.

4.1 Special Case When $r = 0$

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

Definition 1. *Let $J(k) = \{J_j | j \leq k\}$ be the set of the first k jobs according to the EDF order. For $1 \leq u \leq |J(k)|$, we define $E(k, u)$ as the minimum energy consumption of an S-schedule such that $|S| = u$ and $S \subseteq J(k)$. If such a schedule does not exist, i.e. when $u > |J(k)|$, then $E(k, u) = +\infty$.*

Definition 2. *We define $B(t', t, \ell)$ as the minimum energy consumption of an S-schedule such that $|S| = \ell$, $S \subseteq \{J_j | t' < d_j \leq t\}$ and such that all these jobs are scheduled only within the interval $[t', t]$, and with a constant common speed. If such a schedule does not exist, then $B(t', t, \ell) = +\infty$.*

Proposition 1. *$B(d_j, d_k, \ell)$ can be computed in $O(n \log n \log P)$ time, for any j, k, ℓ, with $P = \sum_j p_j$.*

Proof. In order to compute $B(d_j, d_k, \ell)$, we consider the set of jobs $\{J_{j'} | d_j < d_{j'} \leq d_k\}$. For each job in this set, we modify its release date to d_j. Since we want the minimum energy consumption and there is only one speed, we search the minimum speed such that there are exactly ℓ jobs scheduled. This minimum speed can be found by performing a binary search in the interval $[0, s_{max}]$, with $s_{max} = P/(d_k - d_j)$. For every speed s, the processing time of a job J_j is $t_j = p_j/s$, and we compute the maximum number m of jobs which can be scheduled using Moore's algorithm [5] in $O(n \log n)$. If $m < \ell$ (resp. $m > \ell$) the speed s must be increased (resp. decreased). □

Proposition 2. *One has*

$$E(k, u) = \min\{E(k-1, u), B(0, d_k, u), \min_{\substack{1 \leq j < k \\ 1 \leq \ell < u}} \{E(j, \ell) + B(d_j, d_k, u - \ell)\}\}.$$

Proof. Let S be an optimal schedule associated with $E(k, u)$. We can assume that this schedule satisfies the properties of Theorem 1 and Property 1.

If $J_k \notin S$, then $E(k, u) = E(k-1, u)$. If $J_k \in S$, then there are two cases to consider. The first case is when all the jobs in S are scheduled at the same speed.

This case is equivalent to the CS problem, and one has $E(k,u) = B(0,d_k,u)$. The second case is when the schedule S has at least two different speeds. Let C_j be the completion time of job J_j in the schedule S. Let $t = \min_j\{C_j|$ all the jobs scheduled after J_j (at least one job) are executed with the same speed $\} = C_{j^*}$. Necessarily, job J_{j^*} is executed with a different speed. This means that at time C_{j^*} the processor is changing its speed, and using Property 3. of Theorem 1 we can deduce that $C_{j^*} = d_{j^*}$. Now we consider the subschedule S_1 obtained from S by considering only the tasks executed during the interval $[0, d_{j^*})$. Let us assume that there are ℓ^* tasks in this subschedule. Then, necessarily the energy consumption of S_1 is equal to $E(j^*, \ell^*)$, otherwise by replacing S_1 with a better subschedule with energy consumption $E(j^*, \ell^*)$ we could obtain a better schedule than S. Now we consider the subschedule S_2 obtained from S by considering only the tasks executed from time d_{j^*} until the end of the schedule. In a similar way, the energy consumption of S_2 is equal to $B(d_{j^*}, d_k, u - \ell^*)$.

Notice that since the jobs involved in $E(j,\ell)$ have a deadline smaller or equal to d_j, whereas the jobs involved in $B(d_j, d_k, u-l)$ have a deadline greater than d_j, those sets of jobs are always distinct, and therefore the schedule associated with $E(j,\ell) + B(d_j, d_k, u-\ell)$ is always feasible. □

Theorem 2. *The problem* $S1|pmtn, r_j = 0|\sum U_j(E)$ *can be solved in* $O(n^4 \log n \log P)$ *time.*

Proof. We use a dynamic program based on Proposition 2, with $E(0,u) = +\infty$, $\forall u > 0$. The maximum throughput is equal to $\max\{u|E(n,u) \leq E\}$.

The number of values $B(d_j, d_k, \ell)$ is $O(n^3)$. They can be precomputed with a total processing time $O(n^4 \log n \log P)$, using Proposition 1. The number of values $E(k,u)$ is $O(n^2)$, and the complexity to calculate each $E(k,u)$ value is $O(n^2)$ (we have to look for $O(n^2)$ values for j, ℓ and we assume that the previous $E(.,.)$ values have already been computed). Thus the overall complexity is $O(n^4 \log n \log P)$. □

4.2 Agreeable Deadlines

Definition 3. *We define* $E_k(t,u)$ *as the minimum energy consumption of an* S*-schedule, such that* $|S| = u$, $S \subseteq J(k,t) = \{J_j | j \leq k, r_j < t\}$ *and such that all these jobs are executed within the interval* $[r_{min}, t]$*. If such a schedule does not exist, then* $E_k(t,u) = +\infty$*.*

Definition 4. *We define* $A(t', t, \ell, j, k)$ *as the minimum energy consumption of a* S*-schedule such that* $|S| = \ell$, $S \subseteq \{J_j, \ldots, J_k\}$*, and such that all these jobs are scheduled within the interval* $[t', t]$*, and with a constant common speed.*

Proposition 3. $A(t', t, \ell, j, k)$ *can be computed in* $O(n \log n \log P)$ *time, for any* t', t, ℓ, j, k*.*

Proof. In order to compute $A(t', t, \ell, j, k)$, we change the release date of job J_j to t' if $r_j < t'$, and the deadline of job J_j to t if $d_j > t$. The set $\{J_j, \ldots J_k\}$ still

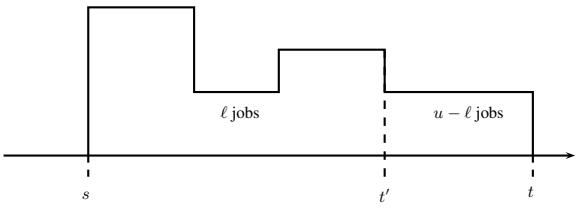
has agreeable deadlines. Then we proceed as in the proof of Proposition 1 using a binary search over the interval $[0, s_{max}]$, with $s_{max} = P/(t - t')$. Note that in this case, we use Lawler's algorithm in [7]. □

Proposition 4. *One has*

$$E_k(t,u) = \min_{\substack{r_{min} \le t' \le t, 0 \le j < k \\ 0 \le \ell \le u}} \Big\{ E_j(t',\ell) + A(t',t,u-\ell,j+1,k) \Big\}.$$

Proof. Let S be an optimal schedule associated with $E_k(t,u)$. We can assume that this schedule satisfies the properties of Theorem 1 and Property 1.

If $J_k \notin S$, then $E_k(t,u) = E_{k-1}(t,u)$. In that case, $t' = t$, $j = k-1$ and $\ell = u$ in the above expression. If $J_k \in S$, then there are two cases to consider. The first case is when the optimal schedule S has one speed. In that case $t' = r_{min}$, $\ell = 0$, $j = 0$ in the above expression. This case is equivalent to the CS problem. The second case is when the optimal schedule S has at least two speeds. In that case we proceed as in the Proposition 2, we split the schedule S into two subschedules S_1 and S_2 (see the figure below).



There exists t' with $r_{min} < t' < t$, such that all the jobs scheduled after t' are scheduled with a common speed, and this is the subschedule S_2. The subschedule S_1 (resp. S_2) has an energy consumption equal to $E_j(t',\ell)$ (resp. $A(t',t,u-\ell,j+1,k)$). Notice that we have to guess the value of j and ℓ in the first subschedule, and the sets of jobs in the second subschedule depend on the first one. □

Theorem 3. *The problem* $S1|pmtn, agreeable|\sum U_j(E)$ *can be solved in* $O(n^6 \log n \log P)$ *time.*

Proof. We use a dynamic program based on Proposition 4. Notice that the important dates are included in the set $\Theta = \{r_j | 1 \le j \le n\} \cup \{d_j | 1 \le j \le n\}$. This comes from the Property 1 and Theorem 1, i.e. the changes of speed of the processor occur only at some release date or some deadline. Therefore we can always assume that $t', t \in \Theta$. Notice also that $|\Theta| = O(n)$.

We define $E_0(t,0) = 0 \ \forall t \in \Theta$, and $E_0(t,u) = +\infty \ \forall u > 0, t \in \Theta$. The maximum throughput is equal to $\max\{u | E_n(d_{max}, u) \le E\}$.

The number of values $A(t',t,\ell,j,k)$ is $O(n^5)$. They can be precomputed with a total processing time $O(n^6 \log n \log P)$, using Proposition 3. The number of

values $E_k(t,u)$ is $O(n^3)$. To compute each value, we have to look for the $O(n^3)$ cases (for each value of t', j, ℓ). In each case, we pick up two values which are already computed. Thus the $E_k(t,u)$ values are computed in $O(n^6)$ time. The overall complexity is $O(n^6 \log n \log P)$. □

4.3 Weighted Version

Next we consider the weighted version of our problem, i.e.

$S1|pmtn, r_j|\sum_j w_j U_j(E)$. In this version a job J_j is defined by its release date r_j, its deadline d_j, its amount of work p_j and its weight w_j. We want to maximize the total weight of the jobs scheduled. We first show that the problem is $\mathcal{NP}$-hard even in the case where all the jobs are released at the same time and have equal deadlines. Then, we present a pseudo-polynomial algorithm for the case where the deadlines are agreeable.

Theorem 4. *The problem $S1||\sum_j w_j U_j(E)$ is $\mathcal{NP}$ hard.*

Proof. In order to establish the $\mathcal{NP}$-hardness of $S1||\sum_j w_j U_j(E)$, we present a reduction from the Knapsack problem which is known to be $\mathcal{NP}$-hard. In an instance of the Knapsack problem we are given a set I of n items. Each item $i \in I$ has a value v_i and a capacity c_i. Moreover, we are given a capacity C, which is the capacity of the knapsack, and a value V. In the decision version of the problem we ask whether there exists a subset $I' \subseteq I$ of the items of total value not less than V, i.e. $\sum_{i \in I'} v_i \geq V$, whose capacity does not exceed the capacity of a knapsack, i.e. $\sum_{i \in I'} c_i \leq C$.

Given an instance of the Knapsack problem, we construct an instance of $S1||\sum_j w_j U_j(E)$ as follows. For each item i, $1 \leq i \leq n$, we introduce a job J_i with $r_i = 0$, $d_i = 1$, $w_i = v_i$ and $p_i = c_i$. Moreover, we set the budget of energy equal to $E = C^\alpha$.

We claim that the instance of the Knapsack problem is feasible iff there is a feasible schedule for $S1||\sum_j w_j U_j(E)$ of total weighted throughput not less than V.

Assume that the instance of the Knapsack is feasible. Therefore, there exists a subset of items I' such that $\sum_{i \in I'} v_i \geq V$ and $\sum_{i \in I'} c_i \leq C$. Then we can schedule the jobs in I' with constant speed $\sum_{i \in I'} c_i$ during $[0,1]$. Their total energy consumption of this schedule is no more that C^α since the instance of the Knapsack is feasible. Moreover, their total weight is no less than V.

For the opposite direction of our claim, assume there is a feasible schedule for $S1||\sum_j w_j U_j(E)$ of total weighted throughput not less than V. Let J' be the jobs which are completed on time in this schedule. Clearly, due to the convexity of the speed-to-power function, the schedule that executes the jobs in J' with constant speed during the whole interval $[0,1]$ is also feasible. Since the latter schedule is feasible, we have that $\sum_{j \in J'} p_j \leq C$. Moreover, $\sum_{j \in J'} w_j \geq V$. Therefore, the items which correspond to the jobs in J' form a feasible solution for the Knapsack. □

In this part, we propose a pseudo-polynomial time algorithm based on a dynamic programming algorithm for the KNAPSACK problem.

Definition 5. *We redefine $E_k(t,w)$ to be the minimum energy consumption of a S-schedule, with $S \subseteq J(k,t) = \{J_j | j \leq k, r_j < t\}$, such that all the jobs in S are scheduled within the interval $[r_{min}, t]$ and such that the sum of their weight is at least w. If such a schedule does not exist, then $E_k(t,w) = +\infty$.*

We redefine $A(t', t, w, j, k)$ to be the minimum energy consumption of a S-schedule such that $S \subseteq \{J_j, \ldots, J_k\}$, $w(S) \geq w$ and such that these jobs are scheduled within the interval $[t', t]$, and with a constant common speed.

Proposition 5. *$A(t', t, w, j, k)$ can be computed in $O(nW \log P)$ time, where W is the sum of weights of the jobs.*

Proof. The proof is similar to Proposition 3. In this case, we use Lawler's algorithm in [6]. □

Lemma 1. *One has*

$$E_k(t,w) = \min_{\substack{r_{min} \leq t' \leq t, 0 \leq j < k \\ 0 \leq \ell \leq w}} \Big\{ E_j(t', \ell) + A(t', t, w - \ell, j+1, k) \Big\}.$$

Proof. The proof is similar to the Proposition 4. □

Theorem 5. *The problem $S1|pmtn, agreeable| \sum_j w_j U_j(E)$ can be solved in $O(n^5 W^2 \log P)$ time.*

Proof. We use a dynamic program based on Proposition 5, with $E_0(t,0) = 0$ $\forall t \in \Theta$ and $E_0(t,w) = +\infty$ $\forall w > 0, t \in \Theta$. The maximum weighted throughput is obtained with $\max\{w | E_n(d_{max}, w) \leq E\}$. The number of values $A(t', t, \ell, j, k)$ is $O(n^4 W)$. They can be precomputed and finally it takes $O(n^5 W^2 \log P)$ time. The number of values $E_k(t,u)$ is $O(n^2 W)$. To compute each value, we have to look for the $O(n^2 W)$ cases (for each value of t', j, ℓ). In each case, we pick up two values which are already computed. Thus the $E_k(t,u)$ values are computed in $O(n^4 W^2)$ time. Thus the overall complexity is $O(n^5 W^2 \log P)$. □

5 Future Work

While the throughput maximization problem is polynomially-time solvable for agreeable deadlines its complexity remains open for general instances. This is a challenging open question for future research.

References

1. Baptiste, P.: An $O(n^4)$ algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. Operations Research Letters 24(4), 175–180 (1999)

2. Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. Annals of Operations Research 5, 287–326 (1979)
3. Yao, F.F., Demers, A.J., Shenker, S.: A Scheduling Model for Reduced CPU Energy. In: Symposium on Foundations of Computer Science (FOCS), pp. 374–382 (1995)
4. Kise, H., Ibaraki, T., Mine, H.: A Solvable Case of the One-Machine Scheduling Problem with Ready and Due Times. Operations Research Letters 26(4), 121–126 (1978)
5. Moore, J.M.: An n job, one machine sequencing algorithm for minimizing the number of late jobs. Management Science 15, 102–109 (1968)
6. Lawler, E.L.: 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)
7. Lawler, E.L.: Knapsack-like scheduling problems, the Moore-Hodgson algorithm and the 'tower of sets' property. Mathematical and Computer Modeling, 91–106 (1994)
8. Bansal, N., Chan, H.-L., Lam, T.-W., Lee, L.-K.: Scheduling for Speed Bounded Processors. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 409–420. Springer, Heidelberg (2008)
9. Chan, H.-L., Chan, W.-T., Lam, T.W., Lee, L.-K., Mak, K.-S., Wong, P.W.H.: Energy Efficient Online Deadline Scheduling. In: SODA 2007, pp. 795–804 (2007); ACM Transactions on Algorithms
10. Chan, H.-L., Lam, T.-W., Li, R.: Tradeoff between Energy and Throughput for Online Deadline Scheduling. In: Jansen, K., Solis-Oba, R. (eds.) WAOA 2010. LNCS, vol. 6534, pp. 59–70. Springer, Heidelberg (2011)
11. Chan, J.W.-T., Lam, T.-W., Mak, K.-S., Wong, P.W.H.: Online Deadline Scheduling with Bounded Energy Efficiency. In: Cai, J.-Y., Cooper, S.B., Zhu, H. (eds.) TAMC 2007. LNCS, vol. 4484, pp. 416–427. Springer, Heidelberg (2007)
12. Li, M.: Approximation Algorithms for Variable Voltage Processors: Min Energy, Max Throughput and Online Heuristics. In: Dong, Y., Du, D.-Z., Ibarra, O. (eds.) ISAAC 2009. LNCS, vol. 5878, pp. 372–382. Springer, Heidelberg (2009)
13. Lam, T.-W., Lee, L.-K., To, I.K.K., Wong, P.W.H.: Energy Efficient Deadline Scheduling in Two Processor Systems. In: Tokuyama, T. (ed.) ISAAC 2007. LNCS, vol. 4835, pp. 476–487. Springer, Heidelberg (2007)