Operator Non-Availability Periods

Nadia Brauner, Gerd Finke, Vassilissa Lehoux-Lebacque, Christophe Rapine, Chris Potts, Vitaly Strusevich

Operator Non-Availability Periods

N. Brauner, G. Finke, V. Lehoux-Lebacque, C. Rapine
C. Potts, V. Strusevich

July 27, 2007
version-3

Abstract. In scheduling literature, the notion of machine non-availability periods is well known, for instance for maintenance. In our case of planning chemical experiments, we have special periods (the week-ends, holidays, vacations) where the chemists are not available. However, human intervention by the chemists is required to handle the starting and termination of the experiments. This gives rise to a new type of scheduling problems, namely problems of finding schedules that respect the operator non-availability periods. These problems are analyzed on a single machine with the makespan as criterion. Properties are described and performance ratios are given for list scheduling algorithms.

Keywords. One-machine scheduling, operator non-availability, complexity, list algorithms

1 Introduction

This study started with an industrial project that we carried out with the *Institut Français du Pétrole* (IFP) (see [3] for a detailed description). Lengthy and costly chemical experiments had to be conducted. The problems of scheduling those experiments had many features that could be addressed by the classical scheduling approaches, for instance, batch processing (heating of chemicals) and batch compatibility issues (in our case, equal heating temperatures). But we were also faced with a new aspect. After the heating procedure, a chemist (an operator) had to be present for the very sensitive handling of the material for the further automated analysis and for a (short) setup to prepare the next experiment. Since in practice the operator is not always available due to breaks, rest periods, week-ends, alternative scheduled activities etc., a new type of scheduling models with operator non-availability intervals has to be considered.

The main concepts of this new model can be formally defined as follows.

Definition 1 *An* operator non-availability *(*Ona*) period of length Δ corresponds to an open time interval $(s, s+\Delta)$ in which a job may neither start nor complete. A job that starts either before or at time s and completes either after or at time $s+\Delta$ is said to* cover *the* Ona *period and is a called a* crossover *job or a* straddling *job.*

The main problem that we study in this paper uses the following data:

- n jobs $J_1, J_2, \ldots, J_n$ to be processed on a single machine, where job J_j is of duration p_j;
- $\mathcal{K} \geq 1$ Ona periods ($\mathcal{K}$ may be a constant or $\mathcal{K}$ may be variable and part of the input);
- s_q is the starting time of the q-th Ona period, $q = 1, 2, \ldots, \mathcal{K}$;
- Δ_q is the duration of the q-th Ona period, $q = 1, 2, \ldots, \mathcal{K}$.

In any feasible schedule S, all ONA periods must be respected, and we are looking for a schedule S^* that minimizes the makespan, *i.e.*, the maximum completion time. For schedule S, the makespan is denoted by $C_{\max}(S)$, and extending the standard scheduling notation we denote the problem of finding S^* by $1\,|Ona(\mathcal{K})|\,C_{\max}$.

To our best knowledge, the scheduling model with ONA periods has not been studied before; however, it appears to be relevant to some previously known models, including models with *a single server* and the models with machine non-availability intervals.

A specific feature of the models with a single server is that each processing operation in the initial (setup) phase of its processing needs to be attended by the operator (server); after the setup is done the operation may run unattended. Scheduling problems with a single server have been studied in multi-machine environments and the main issue is the resolution of conflicts that arise when operations scheduled on different machines compete for the server. See [4, 2] for reviews of recent results in this area on parallel machines. Notice that in all known models of this type the server is assumed to be permanently available.

Another class of models similar to the one considered in this paper focuses on *machine non-availability* (MNA) intervals. In the most general setting, for each machine we are given a collection of fixed time intervals during which no processing activity may take place. These intervals can be attributed to scheduled machine maintenance and check-ups, lunch breaks etc. Traditionally, there are several scenarios of handling the jobs affected by an MNA interval. Under the *non-resumable* scenario, a job that cannot be completed before an MNA interval restarts from scratch, while under the *resumable* scenario, the processing of a job is interrupted by the MNA and is resumed when the machine becomes available again. See [7] for the most recent survey of the results on scheduling with the MNA intervals.

Especially relevant to our study is the problem of minimizing the makespan on a single machine with $\mathcal{K} \geq 1$ MNA intervals under the non-resumable scenario, which we denote by $1\,|Mna(\mathcal{K})|\,C_{\max}$.

The main emphasis of this paper is on design and analysis of approximation algorithms for problem $1\,|Ona(\mathcal{K})|\,C_{\max}$. Recall that a polynomial-time algorithm that creates a schedule with makespan at most $r \geq 1$ times the optimal value is called an *$r-$approximation algorithm*. If a problem admits an $r-$approximation algorithm, it is said to be *approximable within a factor r*. We call *performance ratio* $\rho_{\mathcal{A}}$ of an algorithm $\mathcal{A}$ the *worst-case ratio bound* of $\mathcal{A}$, *i.e.* the infimum of all r such that $\mathcal{A}$ is an r-approximation. A family A_ε of $(1+\varepsilon)-$approximation algorithms is called a *fully polynomial-time approximation scheme,* or an *FPTAS,* if for any fixed $\varepsilon > 0$, the running time of A_ε is polynomial in the length of the problem input and $1/\varepsilon$.

Throughout the paper, we assume that the jobs are indexed in the SPT order, *i.e.*, in non-decreasing order of their processing times $p_1 \leq p_2 \leq \ldots \leq p_n$. For a set $Q \subseteq N = \{1, 2, \ldots, n\}$ of job indices, define $p(Q) = \sum_{j \in Q} p_j$, where for completeness $p(\varnothing) = 0$. In order to exclude a trivial case, we also assume that $p(N) > s_1$; otherwise all jobs can be scheduled before the first ONA period.

In our consideration, we use a well-known integer programming problem, the subset-sum problem. As a decision problem (denoted by SUBSETSUM), it can be formulated as follows: given n numbers $a_1, a_2, \ldots, a_n$ and a bound B, does there exist a subset $I \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{j \in I} a_j = B$? As an optimization problem (denoted by MAXSUBSETSUM), it is usually formulated as a version of the knapsack problem

$$\begin{array}{ll} \text{Maximize} & \sum_{j=1}^{n} a_j x_j \\ \text{Subject to} & \sum_{j=1}^{n} a_j x_j \leq B \\ & x_j \in \{0,1\},\ j = 1, 2, \ldots, n \end{array} \tag{1}$$

The problem MAXSUBSETSUM is NP-hard, can be solved in pseudo-polynomial time and admits an FPTAS. Notice that for a maximization problem, one has to replace $r = 1+\varepsilon$ by $r = \frac{1}{1+\varepsilon}$ in the definition of an FPTAS. See [5] for a comprehensive account on the MAXSUBSETSUM problem and methods of its exact and approximate solutions.

2 Single Ona Period

In problem $1\,|Ona(1)|\,C_{\max}$, given n jobs and a single operator non-availability period of length Δ located at $(s, s+\Delta)$, we have to find a schedule S^* that minimizes the makespan. For problem $1\,|Ona(1)|\,C_{\max}$, we may distinguish two types of instances:

- Type 1: all processing times are smaller than Δ: $p_j < \Delta$ for all j;
- Type 2: some processing times are smaller than Δ, while others are greater than Δ.

For the instances of Type 1, since all processing times are shorter than Δ, there will be no straddling jobs, *i.e.*, no processing activity takes place during the ONA period. The ONA period is effectively an MNA period and problem $1\,|Ona(1)|\,C_{\max}$ is equivalent to problem $1\,|Mna(1)|\,C_{\max}$. The latter problem is proved NP-hard in [6].

Our main emphasis is of course on Type 2 instances of problem $1\,|Ona(1)|\,C_{\max}$. We first give a dominance property.

Lemma 1 *For problem $1\,|Ona(1)|\,C_{\max}$ with $p_n \geq \Delta$, there exists an optimal schedule with the longest job J_n covering the* ONA *period.*

Proof: Consider an optimal schedule S^* with makespan $C_{\max}(S^*)$. Suppose that, in schedule S^*, job J_n is not straddling.

Let E be the set of 'early' jobs that terminate before time s. Assume that J_n is located in set E. Without loss of generality, job J_n is scheduled last among the jobs of this set. If the interval $(s, s+\Delta)$ is not covered by any job, we can delay the starting time of job J_n so that it completes at time $s+\Delta$ (covering the interval) without changing $C_{\max}(S^*)$. If some job J_x with $p_x \leq p_n$ is straddling, then job J_x immediately follows job J_n and we can interchange J_n with J_x without changing $C_{\max}(S^*)$.

Assume now that job J_n is contained in the set E' of jobs that start after the ONA period, *i.e.*, no earlier than time $s+\Delta$. Without loss of generality, job J_n is scheduled first among the jobs of set E'. If $(s, s+\Delta)$ is not covered by any job, we can start job J_n at time s and decrease the starting times of all other jobs in set E' accordingly, thereby decreasing $C_{\max}(S^*)$. If some job J_x with $p_x \leq p_n$ is straddling, then job J_x immediately precedes job J_n and we can interchange J_n with J_x without changing $C_{\max}(S^*)$. ∎

Theorem 1 *Problem $1\,|Ona(1)|\,C_{\max}$ is NP-hard.*

Proof: We prove the theorem by establishing a polynomial reduction of the SUBSETSUM problem to the decision version of problem $1\,|Ona(1)|\,C_{\max}$. Recall that only the Type 2 instances of problem $1\,|Ona(1)|\,C_{\max}$ have to be considered.

Given an arbitrary instance of the SUBSETSUM problem, define an instance of problem $1\,|Ona(1)|\,C_{\max}$ with n jobs of lengths $p_j = a_j$; $j = 1, 2, \ldots, n$; and add an extra job $n+1$ with $p_{n+1} = \sum_{j=1}^{n} a_j + 1$. Define $\Delta = \sum_{j=1}^{n} a_j + 1$, the ONA interval $(B, B+\Delta)$ and a bound $D = \sum_{j=1}^{n+1} p_j$.

It is easily verified that a schedule S with $C_{\max}(S) \leq D$ exists if and only if the SUBSETSUM problem has a solution. Indeed, $C_{\max}(S) = D$ if and only if in S there is no intermediate idle time and job $n+1$ is assigned to the time interval $[B, B+\Delta]$. The set of indices of the jobs that are completed by time B defines a solution to the SUBSETSUM problem. ∎

Although the problem $1\,|Ona(1)|\,C_{\max}$ is NP-hard in the general case, some instances can be solved in polynomial time.

Remark 1 *If $\sum_{j=2}^{n} p_j \leq s$, problem $1\,|Ona(1)|\,C_{\max}$ is polynomially solvable.*

Proof: Recall that $p(N) > s$. Notice that the inequality $\sum_{j=2}^{n} p_j \leq s$ implies that $\sum_{j=1}^{n-1} p_j \leq s$. If $p_n \geq \Delta$, by Lemma 1, we may assume that J_n is the straddling job in an optimal schedule, and we sequence the other jobs before s. In this case, the makespan is $\max\{p(N), s+\Delta\}$, which cannot be reduced.

Otherwise, if $p_n < \Delta$, then there is no straddling job in an optimal schedule and the smallest total processing after the ONA period is equal to p_1. Schedule jobs $J_2, J_3, \ldots, J_n$ before the ONA period and start J_1 at time $s+\Delta$. This gives the smallest possible makespan of $s+\Delta+p_1$. ∎

Theorem 2 *For problem* $1\,|Ona(1)|\,C_{\max}$ *there exists an FPTAS.*

Proof: We show how an FPTAS for MAXSUBSETSUM can be used to develop an FPTAS for $1\,|Ona(1)|\,C_{\max}$. To any instance I of $1\,|Ona(1)|\,C_{\max}$, we associate the instance $f(I)$ of MAXSUBSETSUM composed of n integers of values p_i and a bound $B = s$. Let E^* be the index set of an optimal solution of the MAXSUBSETSUM instance, while the index set found by an FPTAS applied to the same instance is denoted by E_ε. By definition, $p(E_\varepsilon) \geq p(E^*)/(1+\varepsilon)$.

We construct a schedule S for the instance I of $1\,|Ona(1)|\,C_{\max}$ from a solution E of the MAXSUBSETSUM $f(I)$ instance in an obvious way: place the jobs of E in any order before s, shifted to the left. The remaining jobs follow as early as possible, starting with J_n. Note that the schedule S has a makespan $C_{\max}(S) = s + \Delta + p(N) - p(E)$ if $p_n < \Delta$ and $C_{\max}(S) = p(N) + \max\{s + \Delta - (p(E) + p_n), 0\}$ otherwise. It implies that an optimal solution E^* for MAXSUBSETSUM gives an optimal schedule S^* for the corresponding scheduling problem. The schedule associated with E_ε is denoted by S_ε.

If $p_n < \Delta$, then one has $C_{\max}(S_\varepsilon) - C_{\max}(S^*) = p(E^*) - p(E_\varepsilon)$.

Consider now the case $p_n \geq \Delta$. If $s + \Delta \leq p(E_\varepsilon) + p_n$, then $C_{\max}(S^*) = C_{\max}(S_\varepsilon) = p(N)$ and hence $C_{\max}(S_\varepsilon) - C_{\max}(S^*) = 0$.

Otherwise, if $s + \Delta \geq p(E_\varepsilon) + p_n$, then $C_{\max}(S_\varepsilon) - C_{\max}(S^*) \leq p(E^*) - p(E_\varepsilon)$ since by definition one has $C_{\max}(S^*) \geq p(N) + s + \Delta - (p(S^*) + p_n)$.

In all cases, $C_{\max}(S_\varepsilon) - C_{\max}(S^*) \leq p(E^*) - p(E_\varepsilon) \leq p(E^*)\varepsilon/(1+\varepsilon) \leq \varepsilon s \leq \varepsilon C_{\max}(S^*)$, as required. ∎

Notice that the preceding proof implies that problem $1\,|Ona(1)|\,C_{\max}$ is solvable in pseudo-polynomial time, since the MAXSUBSETSUM problem can be solved by a pseudo-polynomial-time dynamic programming algorithm.

3 Multiple Ona Periods

In this section, we design and analyze methods to approximate problem $1\,|Ona(\mathcal{K})|\,C_{\max}$ with $\mathcal{K} > 1$ ONA periods.

Recall that the corresponding problem with machine non-availability periods is hard to approximate. As shown in [1], the general $1\,|Mna(2)|\,C_{\max}$ problem is not approximable within a constant factor, *i.e.*, for a constant r there is no $r-$approximation algorithm unless $\mathcal{P} \neq \mathcal{NP}$. This, however, holds if the length of the periods is allowed to be considerably longer than the total processing requirements. To address problems that are more practically relevant, throughout this section we assume that the length of any (either machine or operator) non-availability period does not exceed the total processing time.

Definition 2 *We say that the periods are* bounded *if the following inequality holds*

$$\Delta_q \leq p(N), \quad \text{for } 1 \leq q \leq \mathcal{K} \tag{2}$$

One of the reasons why we use $p(N)$ in (2) is that this value is a trivial lower bound for the makespan.

In the remainder of this section, for each problem $1\,|Ona(\mathcal{K})|\,C_{\max}$ and $1\,|Mna(\mathcal{K})|\,C_{\max}$ with $\mathcal{K} > 1$, we analyze the worst-case performance of approximation algorithms that employ the popular concept of algorithms based on priority lists. These algorithms scan the jobs according to a certain sequence (a list) and take decisions regarding the assignment of a job from the list on the machine; once a job is scheduled it is removed from the list.

- **Algorithm FF (First Fit):** Schedule the first job in the current list to start as early as possible. Notice that in the current partial schedule, the ONA periods may have created idle intervals large enough for this job to be placed before the end of the partial schedule.
- **Algorithm LS (List Scheduling):** As soon as the machine becomes available at some instant, check the current list and add a job to the partial schedule that can start earlier than the other jobs, breaking ties by selecting the job that is sequenced earlier in the list.

Algorithm FF does not search the list and takes the first available job; however it allows this job to be scheduled in the internal part of the current partial schedule, without increasing its makespan. On the other hand, Algorithm LS searches the current list to find the jobs that would start as early as possible, but not before the completion time of the last job in the current partial schedule.

In all schedules considered in this section the starting time of each job cannot be further reduced; schedules of this structure are called *semi-active*. Notice that the schedules found by algorithms FF and LS are semi-active.

We start with analyzing semi-active schedules for problem $1\,|Mna(\mathcal{K})|\,C_{\max}$ with $\mathcal{K}$ machine non-availability intervals.

For a given schedule S, we define the largest integer $k \leq \mathcal{K}$ such that $C_{\max}(S) \geq s_k + \Delta_k$. Let x_1 denote the idle time before the first non-availability period and x_q the idle time that occurs between the $(q-1)$-th and the q-th periods, $1 \leq q \leq k$. Since the schedule is assumed to be semi-active, there is at most one such idle period between two consecutive periods.

Clearly the inequality

$$C_{\max}(S) \leq p(N) + \sum_{q=1}^{k} x_q + \sum_{q=1}^{k} \Delta_q \tag{3}$$

holds for any schedule.

By assumption, each Δ_q is no larger than the total processing time $p(N)$. Notice that for any semi-active schedule, each x_q must be smaller than the largest processing time p_n, otherwise we would have found a job that fits into the corresponding idle interval. Therefore, we have

$$x_q \leq p_n \leq C^*_{\max}, \qquad \text{for} \quad 1 \leq q \leq k \tag{4}$$

Proposition 1 *For problem $1\,|Mna(\mathcal{K})|\,C_{\max}$ with bounded machine non-availability periods, the inequality $C_{\max}(S)/C_{\max}(S^*) \leq 2\mathcal{K}$ holds for semi-active schedules S, and there are lists for which this bound is tight asymptotically for $\mathcal{K} \geq 2$, even if the schedule is found either by Algorithm FF or Algorithm LS.*

Proof: Consider an optimal schedule S^*. We have

$$C_{\max}(S^*) \geq p(N) + \Delta_1$$

It follows from (3) that

$$C_{\max}(S) - C_{\max}(S^*) \leq \sum_{q=2}^{k} \Delta_q + \sum_{q=1}^{k} x_q$$

Therefore with (4) we get

$$C_{\max}(S) - C_{\max}(S^*) \leq (k-1)\,p(N) + kp_n \leq (2\mathcal{K}-1)\,C_{\max}(S^*)$$

and the required bound follows.

To verify that the bound of $2\mathcal{K}$ is tight, take a small positive ε such that $\mathcal{K}\,\varepsilon < 1$ and define the following instance of problem $1\,|Mna(\mathcal{K})|\,C_{\max}$ with two jobs:

- two jobs J_1 and J_2 with $p_1 = \varepsilon$ and $p_2 = 1$;
- $\Delta_1 = (\mathcal{K}-1)\varepsilon$ and $\Delta_q = 1$,s for $2 \le q \le \mathcal{K}$;
- $s_1 = 1$ and $s_q = s_{q-1} + \Delta_{q-1} + 1 - \varepsilon = (2q-2) + (\mathcal{K}-q)\varepsilon$, for $2 \le q \le \mathcal{K}$.

In an optimal schedule, job J_2 is scheduled before the first MNA period, and job J_1 is scheduled immediately after that period, so that $C_{\max}(S^*) = 1 + \mathcal{K}\varepsilon$. Consider the list (J_1, J_2). Based on this list, all three schedules, semi-active, algorithm FF, algorithm LS yield the same schedule S. Since $s_q - (s_{q-1} + \Delta_{q-1}) = 1 - \varepsilon$ for each q, $2 \le q \le \mathcal{K}$, it follows that in schedule S, job J_2 may only start after the last non-availability interval, so that $C_{\max}(S) = (2\mathcal{K}-1)+1 = 2\mathcal{K}$. As $\varepsilon \to 0$, the ratio $C_{\max}(S)/C_{\max}(S^*)$ approaches $2\mathcal{K}$. ∎

We now turn to problem $1\,|Ona(\mathcal{K})|\,C_{\max}$ with operator non-availability periods. Since scheduling with ONA periods includes more possibilities (an ONA period may be covered by a job), it is reasonable to expect an improvement of the worst-case performance of list scheduling algorithms, compared to a ratio $2\mathcal{K}$ valid for the problem with machine non-availability periods.

Recall that we have, for an optimal schedule S^*, the inequalities $C_{\max}(S^*) \ge p(N) > s_1$.

Proposition 2 *For problem $1\,|Ona(\mathcal{K})|\,C_{\max}$ with bounded operator non-availability periods, the bound $C_{\max}(S)/C_{\max}(S^*) \le 2\mathcal{K}$ holds for all semi-active schedules S, and there are lists for which this bound is tight asymptotically for $\mathcal{K} \ge 2$, even if the schedule is found by Algorithm FF.*

Proof: We know that

$$\Delta_q \le p(N) \le C_{\max}(S^*),\ 1 \le q \le \mathcal{K}$$
$$x_1 + \Delta_1 \le s_1 + \Delta_1 \le C_{\max}(S^*)$$

Semi-active schedules S satisfy (3) and (4). Hence,

$$C_{\max}(S) \le C_{\max}(S^*) + C_{\max}(S^*) + 2(\mathcal{K}-1)C_{\max}(S^*) \le 2\mathcal{K}C_{\max}(S^*)$$

We now prove that this bound is tight. Take an arbitrary Δ and a sufficiently small ε, $4\varepsilon < \Delta$, and define the following instance of problem $1\,|Ona(\mathcal{K})|\,C_{\max}$ with two jobs (see Figure 1):

- $p_1 = \varepsilon, \quad p_2 = \Delta - \varepsilon/2$;
- $\Delta_1 = \Delta - \varepsilon, \quad \Delta_q = \Delta, \quad$ for $2 \le q \le \mathcal{K}$;
- $s_1 = 0, \quad s_q = s_{q-1} + \Delta_{q-1} + \Delta - 2\epsilon, \quad$ for $2 \le q \le \mathcal{K}$.

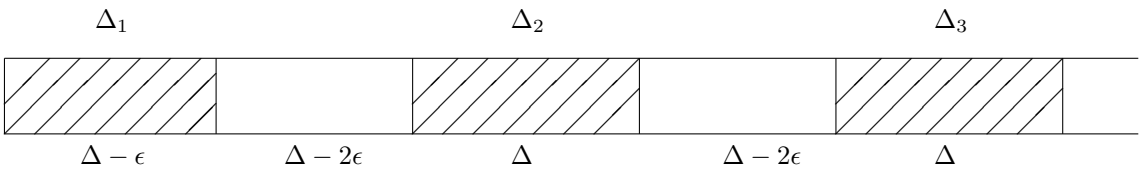

Figure 1: ONA periods for the instance that achieves the ratio of $2\mathcal{K}$

An optimal schedule S^* is associated with the sequence (J_2, J_1), provided that J_2 starts at time zero, so that $C_{\max}(S^*) = \Delta + \frac{\varepsilon}{2} = p(N)$.

Using list (J_1, J_2), Algorithm FF results in a schedule S with $C_{\max}(S) = s_{\mathcal{K}} + \Delta + p_2$, since job J_2 is too small to overlap any of the ONA periods starting from the second one, and is too big to be placed between any two of those ONA periods. Thus, $C_{\max}(S) = \Delta - \varepsilon + (\mathcal{K}-1)(2\Delta - 2\epsilon) + \Delta - \varepsilon/2 = 2\mathcal{K}(\Delta - \varepsilon) + \varepsilon/2$.

Therefore, we have

$$\frac{C_{\max}(S)}{C_{\max}(S^*)} = \frac{2\mathcal{K}(\Delta - \varepsilon) + \varepsilon/2}{\Delta + \varepsilon/2} \longrightarrow_{\varepsilon \to 0} 2\mathcal{K}$$

as required. ∎

We now show that schedules found by Algorithm LS are closer to the optimum than an arbitrary semi-active schedule. Let us call ρ_L the performance ratio for algorithm LS associated with list L.

Theorem 3 *Consider problem* $1\,|Ona(\mathcal{K})|\,C_{\max}$ *with bounded operator non-availability periods. Then,* $\rho_L = 2(\mathcal{K}-1)$ *holds for any* $\mathcal{K} \geq 4$.

We prove this theorem by means of several lemmas.

Lemma 2 *For all lists* L *and any* $\mathcal{K} \geq 2$, *the performance ratio* ρ_L *satisfies* $\rho_L \geq 2(\mathcal{K}-1)$.

Proof: Take an arbitrary Δ and a sufficiently small ε, and define the following instance of problem $1\,|Ona(\mathcal{K})|\,C_{\max}$ with two jobs (see Figure 2):

- $p_1 = \varepsilon$, $p_2 = \Delta + \varepsilon/2$;
- $\Delta_1 = \Delta$; $\Delta_2 = \varepsilon$; $\Delta_q = \Delta + \varepsilon$, for $2 \leq q \leq \mathcal{K}$;
- $s_1 = \varepsilon$; $s_2 = \Delta + \varepsilon$; $s_3 = 2\Delta + \varepsilon$; $s_q = s_{q-1} + 2\Delta + \varepsilon$, for $4 \leq q \leq \mathcal{K}$.

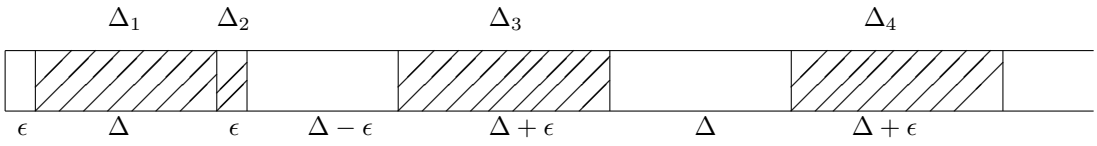

Figure 2: ONA periods for the instance that achieves the ratio of $2\mathcal{K} - 1$

An optimal schedule S^* starts with J_2 at time $\varepsilon/2$ and places J_1 without idle time, so that $C_{\max}(S^*) = \Delta + 2\varepsilon$.

Independent of the two possible lists, (J_1, J_2) and (J_2, J_1), algorithm LS places job J_1 at time 0. Then the second job J_2 can only start after the last ONA period so that

$$C_{\max}(S_L) = (\mathcal{K}-1)(2\Delta + \varepsilon) + \varepsilon/2 \quad \text{for each } \mathcal{K} \geq 2$$

Therefore,

$$\frac{C_{\max}(S_L)}{C_{\max}(S^*)} \longrightarrow_{\varepsilon \to 0} 2(\mathcal{K}-1)$$

for each $\mathcal{K} \geq 2$. ∎

Lemma 3 *For all instances such that the optimal schedule* S^* *finishes after the second* ONA *period, every semi-active schedule* S *verifies* $\frac{C_{\max}(S)}{C_{\max}(S^*)} \leq 2(\mathcal{K}-1)$ *for each* $\mathcal{K} \geq 2$.

Proof: Consider a semi-active schedule S with idle times x_q, $q = 1, 2, \ldots, k$. We know that

$$\Delta_q \leq p(N) \leq C_{\max}(S^*),\ 1 \leq q \leq \mathcal{K}$$
$$x_1 + \Delta_1 + x_2 + \Delta_2 \leq C_{\max}(S^*)$$

Using (3) and (4), we get

$$C_{\max}(S) \leq p(N) + C_{\max}(S^*) + \sum_{q=3}^{\mathcal{K}}(x_q + \Delta_q) \leq (2 + 2(\mathcal{K} - 2))C_{\max}(S^*)$$

and the lemma follows. ∎

It remains to consider the case where an optimal schedule completes between the first two ONA periods. For completeness, we also include here the case $\mathcal{K} = 1$ with $s_2 = +\infty$.

Lemma 4 *For the set $\mathcal{I}$ of instances for which an optimal schedule S^* completes between the first two* ONA *periods (*i.e. *$s_1 + \Delta_1 \leq C_{\max}(S^*) \leq s_2$), every list schedule, restricted to $\mathcal{I}$, verifies $\rho_L \leq 3\mathcal{K}/2$ for $\mathcal{K} \geq 1$.*

Proof: Let J_l be the last job scheduled in a list schedule S_L. We know that the processing time of job J_l must be larger than the length of any machine idle interval x_q.

We first show that if schedule S_L is not optimal, then $p_l \leq \frac{1}{2}C_{\max}(S^*)$. This means that the duration of the job scheduled last in S_L is at most a half of the optimal makespan. We split our consideration into two cases.

Case 1. Assume that the first ONA period is covered in S_L. Notice that if no idle time occurs before s_1 (*i.e.*, $x_1 = 0$), clearly S_L is optimal since $p(N) \leq s_2$. Otherwise, in S_L there exists a machine idle interval $[t, t + x_1]$, where $t + x_1 < s_1$. The processing time of any job that remains to be scheduled by Algorithm LS at time t is larger than $s_1 - t$ and smaller than $s_1 + \Delta_1 - t$. Hence, any job J_j with the processing time larger than Δ_1 has an earliest date $s_1 + \Delta_1 - p_j$ when it can be scheduled to overlap the first ONA period. Algorithm LS will select the job that may start earlier than the other candidates, *i.e.*, it will select a job J_β with the largest processing time among the remaining jobs to become the crossover job for the first ONA period and to be completed at time $s_1 + \Delta_1$. Note that the jobs J_β and J_l are different, since otherwise $C_{\max}(S) = s_1 + \Delta_1$ and this schedule is optimal. Thus, $C_{\max}(S^*) \geq p_\beta + p_l \geq 2p_l$.

Case 2. Assume now that the first ONA period is not covered in S_L. Since it was not possible to schedule J_l to overlap Δ_1, we have $p_l < \Delta_1$. If in an optimal schedule S^* the first ONA period is covered by some job J_α, then due to $p_\alpha \geq \Delta_1 > p_l$ we have that $C_{\max}(S^*) \geq p_\alpha + p_l \geq 2p_l$. Otherwise, if the first ONA period is not covered in an optimal schedule S^*, then $C_{\max}(S^*) \geq p(N) + \Delta_1 \geq 2p_l$.

Now let us denote by A the total length of the machine idle intervals in schedule S_L in interval $[0,\ s_1 + \Delta_1]$. It is clear that $A \leq x_1 + \Delta_1 \leq C_{\max}(S^*)$. Since job J_l cannot start earlier, we get

$$\begin{aligned} C_{\max}(S) &\leq p(N) + A + \sum_{q=2}^{k}(x_q + \Delta_q) \leq p(N) + A + (\mathcal{K} - 1)(p_l + p(N)) \\ &\leq 2C_{\max}(S^*) + (\mathcal{K} - 1)C_{\max}(S^*)/2 + (\mathcal{K} - 1)C_{\max}(S^*) \leq \left(\frac{3}{2}\mathcal{K} + \frac{1}{2}\right) C_{\max}(S^*) \end{aligned}$$

The latter estimate can be further improved by noticing that in fact

$$p(N) + A \leq 2C_{\max}(S^*) - p_l \tag{5}$$

provided that S_L is not optimal. To see this, consider again the following cases:

- If the first ONA period is covered in S_L, then we simply have $A \leq x_1 \leq p_l$. Using the fact that $p_l \leq C_{\max}(S^*)/2$, we get $p(N) + A \leq C_{\max}(S^*) + p_l \leq 2C_{\max}(S^*) - p_l$.
- If the first ONA period is not covered in an optimal schedule S^*, then $C_{\max}(S^*) \geq p(N) + \Delta_1$. Hence, $p(N) + A \leq C_{\max}(S^*) + x_1 \leq C_{\max}(S^*) + p_l \leq 2C_{\max}(S^*) - p_l$.
- The only remaining case occurs if the first ONA period is covered in an optimal schedule, and not in schedule S_L. As above, let J_α be the crossover job for the first ONA period in schedule S^*, and let $[t, t + x_1]$ be the first machine idle interval in schedule S_L. Since $C_{\max}(S^*) \leq s_2$ and $p_\alpha \geq \Delta_1$, the only reason why the first ONA period is not covered in S_L is that every job with the processing time larger than Δ_1, including job J_α, has been scheduled before time t. This implies that $p_l < \Delta_1 \leq p_\alpha \leq t \leq s_1 - x_1$ and we obtain that $C_{\max}(S^*) \geq s_1 + \Delta_1 \geq p_\alpha + x_1 + \Delta_1 \geq p_l + A$, so that (5) follows immediately.

Using (5), we derive

$$C_{\max}(S) \leq (2C_{\max}(S^*) - p_l) + (\mathcal{K} - 1)(p_l + C_{\max}(S^*)) \leq \frac{3}{2}\mathcal{K}C_{\max}(S^*)$$

■

Since $3\mathcal{K}/2 \leq 2(\mathcal{K}-1)$ for $\mathcal{K} \geq 4$, Lemma 4 completes the proof of Theorem 3. The obtained results can be summarized as follows.

Theorem 4 *For problem $1\,|Ona(\mathcal{K})|\,C_{\max}$ with bounded operator non-availability periods, the following bounds hold for all lists L*

- $\rho_L \leq 3/2$ *for* $\mathcal{K} = 1$*;*
- $2 \leq \rho_L \leq 3$ *for* $\mathcal{K} = 2$*;*
- $4 \leq \rho_L \leq 4.5$ *for* $\mathcal{K} = 3$*;*
- $\rho_L = 2(\mathcal{K} - 1)$ *for* $\mathcal{K} \geq 4$.

We now show that the bound 2 on the performance ratio found for list scheduling algorithms for $\mathcal{K} = 2$ also holds for any polynomial time algorithm.

Theorem 5 *Problem $1\,|Ona(2)|\,C_{\max}$ with bounded operator non-availability periods is not approximable within a factor smaller than or equal to 2, unless $\mathcal{P} = \mathcal{NP}$.*

Proof: We prove that a polynomial-time $r-$approximation algorithm A for problem $1\,|Ona(2)|\,C_{\max}$ with $r \leq 2$, if it exists, would solve to optimality an NP-complete problem, namely PARTITION, which is a version of the SUBSETSUM problem with $B = \sum_j a_j/2$.

Given an arbitrary instance of PARTITION, we construct an instance I of problem $1\,|Ona(2)|\,C_{\max}$ with $p(N) = \sum_{i=1}^{n+1} p_i \geq \max\{\Delta_1, \Delta_2\}$ and with $n+1$ jobs $J_1, \ldots, J_n, J_{n+1}$ as follows: the processing times are $p_j = a_j$, for all $j \in \{1, 2, \ldots, n\}$, and we add an extra job $p_{n+1} = \max a_j + 1$. The two ONA intervals are $(B, B + p_{n+1})$ and $(p(N), 2p(N))$.

Observe, as in the proof of Theorem 1, that PARTITION has a solution if and only if, for the constructed instance I of problem $1\,|Ona(2)|\,C_{\max}$, an optimal schedule S^* satisfies $C_{\max}(S^*) = p(N)$. Due to the second ONA period occurring at time $p(N)$, which can not be overlapped, it implies that any schedule for a negative instance of PARTITION has a makespan strictly greater than $2p(N)$. Now assume that there exists an r-approximation algorithm A with $r \leq 2$, returning the value $C_{\max}$ for the instance I. If I is constructed from a negative instance of partition, then $C_{\max} > 2p(N)$. If I is constructed from a positive instance of partition, then $C_{\max} \leq rC_{\max}(S^*) \leq 2p(N)$. Hence, algorithm A can decide PARTITION in polynomial time, which implies $\mathcal{P} = \mathcal{NP}$. ■

Notice that this result excludes the case $\rho_l = 2$ for $\mathcal{K} = 2$ in Theorem 4.

4 λ-bounded periods

So far, we have considered bounded periods, *i.e.* $p(N) \geq \Delta = \max_{q=1}^{\mathcal{K}} \Delta_q$. This is a rather weak condition. There are instances where none of the periods can be covered. The poor performance ratio for list schedules may in part be due to this fact. In order to increase the number of covered periods, we define λ-*bounded periods* by the condition :

$$p(N) \geq \lambda \, \Delta$$

The previous theory refers now to 1-bounded periods.

Lemma 5 *For instances with λ-bounded periods, λ large enough, semi-active schedules have a performance ratio not greater than 2.*

Proof: Let $\lambda > \frac{s_{\mathcal{K}}}{\Delta}$. Then $C^*_{max} \geq \sum p_j > (\frac{s_{\mathcal{K}}}{\Delta})\Delta = s_{\mathcal{K}}$ and $C^*_{max} \geq s_{\mathcal{K}} + \Delta_{\mathcal{K}}$. Therefore, $C_{max} \leq s_{\mathcal{K}} + \Delta_{\mathcal{K}} + \sum p_j \leq 2C^*_{\max}$. ∎

Theorem 6 *For problem $1|Ona(\mathcal{K})|C_{max}$ with λ-bounded periods, list scheduling algorithms have a performance ratio satisfying*

$$\rho_L \leq 1 + \frac{2\mathcal{K}}{\lambda}$$

In particular, we get a constant bound $\rho_L \leq 3$ for $\lambda \geq \mathcal{K}$. Also the makespan of any list schedule tends to C^*_{max} as $\lambda \to \infty$. Previously, we had $\lambda = 1$. Here we obtain the bound $\rho_L \leq 1 + 2\mathcal{K}$ which is weaker than the ones in Theorem 4.

Proof: Consider a list schedule. Let A_i denote the inactivity period in $[s_{i-1} + \Delta_{i-1}, s_i + \Delta_i)$ with $s_0 = 0$, $\Delta_0 = 0$. Since we consider only semi-active schedules, we have at most one such period per interval.

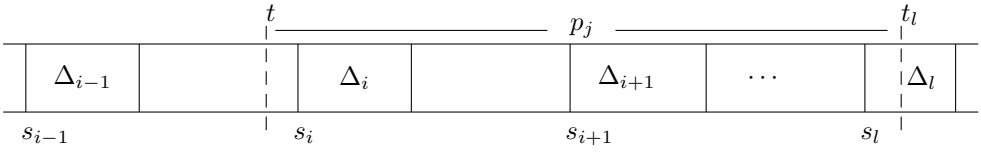

Figure 3: Position of idle times

Let $t \in [s_{i-1} + \Delta_{i-1}, s_i]$ be the beginning of an inactivity period A_i. Suppose job J_j of length p_j is the first job placed after time t in the schedule. Since J_j cannot start at t, we must have $t + p_j = t_l \in (s_l, s_l + \Delta_l)$ for some $l \geq i$. We may distinguish two cases (See Figure 3).

(1) Interval Δ_i is covered by J_j. Then $s_l + \Delta_l - t_l \leq s_i - t$ and the length of A_i satisfies $|A_i| = s_l + \Delta_l - t_l < \Delta_l$.

(2) Δ_i is not covered by J_j. Then $s_l + \Delta_l - t_l > s_i - t$ and also $s_i - t < \Delta_l$. Hence $|A_i| = s_i - t + \Delta_i \leq \Delta_i + \Delta_l$.

In all cases, we have $|A_i| \leq 2\Delta$. Therefore, $C_{max} \leq \sum |A_i| + \sum p_j \leq 2\mathcal{K}\Delta + \sum p_j \leq (1 + 2\mathcal{K}/\lambda)p(N) \leq (1 + 2\mathcal{K}/\lambda)C^*_{max}$ which completes the proof. ∎

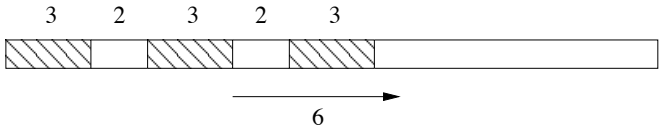

Figure 4: An example with a task that cannot finish in $[0, s_{\mathcal{K}}]$

Increasing the value of λ is usually the result of adding more jobs and big jobs. However, it is always possible to construct large jobs, larger than the ONA periods, that cannot be placed and finish in interval $[0, s_{\mathcal{K}}]$, see example in Figure 4. If such a job exists, then $C^*_{\max} > s_{\mathcal{K}}$ and we obtain, as in the proof of Lemma 5, the performance bound of 2 for any semi-active schedule. We add a final remark. Even if all jobs J_j have processing times $p_j \geq \Delta_q$ for all $q \in \{1, 2, \ldots, \mathcal{K}\}$, where $\mathcal{K}$ cannot be reduced, we are not sure that all ONA periods are covered in an optimal solution (see Figure 5).

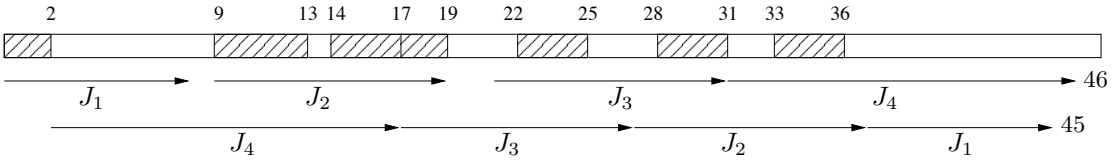

Figure 5: An example with an ONA period not covered in the optimal solutions

5 Conclusion

We have introduced a new scheduling model, in which the operator non-availability periods have to be respected. The resulting class of problems has richer combinatorial features than scheduling problems with the traditional machine non-availability periods. We establish properties and performance ratios for list scheduling algorithms. Still, there is a wide range of problems for further study. This includes proving tightness of all known ratios and the search of improved approximation algorithms. We also generalize the concept of bounded periods to so-called λ-bounded periods and obtain list algorithms with constant performance ratios. Problems with other objective functions and in other machine environments are also worth studying.

Acknowledgement

This research has been financed in part by the INTAS network 03-51-5501. We also thank N. Mete, M.Sc. student in Grenoble, for having constructed the example in Figure 5.

References

[1] J. Breit, G. Schmidt, V.A. Strusevich, Non-preemptive two-machine open shop scheduling with non-availability constraints *Mathematical Methods of Operations Research* 34 (2003) 217–234.

[2] P. Brucker, C. Dhaenens-Flipo, S. Knust, S.A. Kravchenko, F. Werner, Complexity results for parallel machine problems with a single server, *Journal of Scheduling*, 5 (2002) 429-457.

[3] V. Lebacque, N. Brauner, B. Celse, G. Finke, C. Rapine. Planification d'expériences dans l'industrie chimique. In: *Colloque IPI 2006*, Allevard, France, 2006.

[4] N.G. Hall, C. Potts, C. Sriskandarajah. Parallel machine scheduling with a common server. In *Discrete Applied Mathematics*, 102 (2000) 223-243.

[5] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*, Springer, Berlin et al., 2004.

[6] C.-Y. Lee, Machine scheduling with an availability constraint, *Journal of Global Optimization* 9 (1996) 395–416.

[7] C.-Y. Lee, Machine scheduling with availability constraints, in J. Y.-T. Leung (Editor), Handbook of Scheduling: Algorithms, Models and Performance Analysis, Chapman & Hall/CRC, London, 2004, pp. 22-1 – 22-13.