

The Homogeneous Non-Idling Scheduling Problem

Alain QUILLIOT
 Laboratoire LIMOS CNRS
 Université Blaise Pascal, Bat ISIMA,
 Campus des Cézeaux, BP 10125,
 63173 Aubière, France Email:
 alain.quilliot@isima.fr

Philippe CHRETIENNE
 Université de Paris VI
 Laboratoire LIP6 CNRS
 75005 Paris, France
 Email: philippe.chretienne@lip6.fr

Abstract—This paper is about multi-processor scheduling with non idling constraints, i.e constraints which forbid interruption in the use of the processors. We first reformulate the problem, while using a notion of pyramidal shape, and next apply matching techniques in order to get a min-max feasibility characterization criterion, which allows us to derive a polynomial algorithm for the related existence problem and for the Makespan Minimization related problem.

I. INTRODUCTION

MOST scheduling problems assume that no cost is incurred when a machine waits between the completion of a job and the start of the next job. Moreover, it is well-known that such waiting delays are often necessary to get optimality, whatever be the related performance criterion. This is the key feature which explains why list algorithms, which do not allow a machine to wait for a more urgent job, do not generally yield optimal schedules. However, it may happen that in some applications such that those described in [5], the cost of making a running machine stop and restart later is so high that a non idling constraint is put on the machine so that only schedules without any intermediate delays are accepted. For instance, if the machine is an oven which must heat different and non compatible pieces of work at a given high temperature, keeping the required temperature of the oven while it is empty may clearly become too costly. Problems concerning power management policies may also yields similar constraints [6], where for example each idling period has a cost and the total cost has to be minimized [1]. Note that the non idling constraint will not necessarily ensure full machine utilization but will remove the cost of machine re-starts, maybe at the price of processing the jobs later.

Contrarily to the well-known no-wait constraints in job scheduling, where no idle time is allowed between the successive operations of a same job, the non-idling machine constraints have just begun to receive research attention in the literature. To the best of our knowledge, the first work on such problems concerns the earliness-tardiness single machine scheduling problem with no unforced idle time, where a Branch and Bound approach has been developed [4]. More recently, some aspects of the impact of the non-idling constraints on the complexity of single-machine

scheduling problems as well as the important role played by the earliest starting time of a non-idling schedule has been studied in [2]. Moreover, in [3], a Branch and Bound efficient method has been developed for the basic one-machine non-idling scheduling problem.

In this paper, we study the basic homogeneous m -machine non-idling problem, where weakly dependant unit-time jobs have to be scheduled within time windows in such a way that the non-idling constraint be satisfied not only for each machine but also for every subset of machines. In section 2, the problem, as well as the key notions of *pyramidal structure*, *k-hole*, *m-matching*, *pre-schedule*, *k-schedule* and *schedule* are defined. Section 3 is devoted to structural analysis and to the derivation of feasibility criteria. It first examines the case of a *m-matching* and next the case of a *pre-schedule*, and finally provides a necessary and sufficient condition for an instance to admit at least a feasible schedule. Section 4 provides a polynomial algorithm which solves both the related existence problem and the makespan minimization problem.

II. NOTATIONS, PROBLEM DEFINITION AND REFORMULATION

A. Main Notations: Time-Units and Intervals

We consider the discrete time space $\mathbf{N} = \{0, \dots, +\infty\}$, each element of it being called a *time-unit*. A subset Ω of \mathbf{N} is an *interval* if it is made of a finite number of consecutive time-units. The smallest (respectively largest) time-unit of an interval Ω is denoted $\min(\Omega)$ (respectively $\max(\Omega)$). If p and q are two distinct natural numbers, the interval whose bounds are p and q is denoted by $I(p, q)$ (note that we may have $p < q$ or $q < p$). Interval Ω_2 is said to *dominate* interval Ω_1 if $\max(\Omega_1) + 1 < \min(\Omega_2)$: such a situation is denoted by $\Omega_1 \text{ Dom } \Omega_2$. If $\Omega_1 \text{ Dom } \Omega_2$, then we denote by $\text{Mid}(\Omega_1, \Omega_2)$ the (non empty) interval $\{\max(\Omega_1) + 1, \dots, \min(\Omega_2) - 1\}$. We say that two intervals are *connected* if their union is an interval.

B. The Feasibility Homogenous Non-idling Scheduling Problem

We now suppose that we are given a set $J = \{J_1, \dots, J_n\}$ of n unit-time jobs that are to be processed on a set $M = \{M_1, \dots,$

M_m of m identical machines. Job J_i must be executed inside a given time-window $F(i) = \{r_i, \dots, d_i\}$ which is an interval. It will be convenient to denote r_{min} (respectively d_{max}) the smallest r_i (respectively the smallest d_i) and by H the interval $\{r_{min}, \dots, d_{max}\}$. The jobs are also constrained by a *weak precedence* relation denoted by \ll where $J_i \ll J_j$ means that J_j must not be performed before J_i . Jobs are further constrained by the so-called homogeneous non-idling constraint (*HNI* in short) which imposes that, for any subset M' of M , the time units at which the machines of M' are busy define an interval. Then a *schedule* of the job set M is a pair (T, μ) , where T and μ are two functions, which assign, to any job J_i , respectively a time-unit $T(i)$ and a machine $\mu(i)$. The schedule (T, m) is said to be *feasible* if:

- For any job J_i , $T(i) \in F(i)$;
- For any pair of jobs J_i, J_j , such that $J_i \ll J_j$, we have $T(i) \leq T(j)$;
- For any pair of jobs J_i, J_j , we have either $T(i) \neq T(j)$ or $\mu(i) \neq \mu(j)$;
- The *HNI* condition is satisfied: for any subset M' of M , the set $\{t \in \mathbb{N}, \text{ such that there exist } i = 1..n, \text{ with } T(i) = t \text{ and } T(i) \in M'\}$ is an interval.

For any such a schedule (T, μ) , we define the *active time-unit set* of the function T as the image set of T , that means as the subset $ACT(T)$ of \mathbb{N} defined by: $ACT(T) = \{t \in \mathbb{N} \text{ such that there exists at least some index value } i = 1..n, \text{ with } T(i) = t\}$.

Then we define the related *Feasibility Homogeneous Non-Idling Scheduling Problem* $NON-IDLE_0 = (P, HNI|p_i = 1, r_i, d_i, preceq|-)$ as the problem which consists in deciding whether the given instance (J, F, \ll, m) admits at least one feasible schedule. If the answer is yes, we say that this instance is *feasible*.

It must be pointed out that the precedence relation *preceq* which we handle here has not the same meaning as the classical one, since when we set $J_i \ll J_j$, we allow J_i and J_j to be processed at the same time-unit. Clearly, due to the machine constraint, there is no difference when $m = 1$.

It is also of interest to notice that the problem $(P, HNI|p_i = 1, r_i, d_i, preceq|-)$, where *prec* is the usual precedence relation, is NP-Complete, since the NP-Complete $(P|p_i = 1, prec|C_{max} \leq d)$, polynomially reduces to the problem $(P, HNI|p_i = 1, r_i, d_i, preceq|-)$, by adding $(md - n)$ filling jobs and setting $r_i = 1$ and $d_i = d$ for all the jobs.

C A Reformulation Through Pyramidal Shape Functions

Let $I = (J, F, \ll, m)$ be an instance of $NON-IDLE_0$ and (T, μ) some schedule of I . If $t \in \mathbb{N}$, we denote by $n_T(t) = \text{Card}(T^{-1}(t))$ the number of jobs which are scheduled at time-unit t according to T , and we call this function the *resource*

profile function of the schedule. We say that this function $t \rightarrow n_T(t)$ has a *pyramidal shape* if for any time units t, t', t'' such that $t' < t < t''$, we have $\text{Inf}(n_T(t'), n_T(t'')) \leq n_T(t)$ (see figure 1).

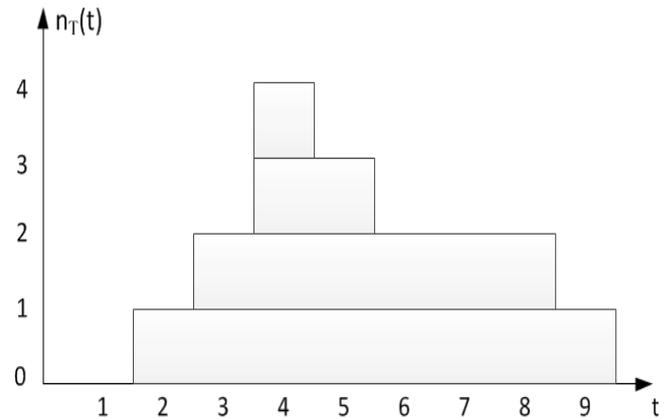


Figure 1: A pyramidal function $n_T(t)$

Then we say that (T, μ) is a *flat schedule* of the instance I , if for any time unit t such that $n_T(t) > 0$, we have $\mu(t) = \{M_1, \dots, M_{n_T(t)}\}$. Clearly any feasible schedule (T, μ) can be turned, through reassignment of the jobs onto the machines, into a flat feasible schedule, and, in the case of a flat schedule, the knowledge of T determines μ . So, the following statement reformulates the $NON-IDLE_0$ as a problem which only involves the time function T , subject to some *pyramidal shape* property related to the profile function $t \rightarrow n_T(t)$.

Theorem 1: *Solving the $NON-IDLE_0$ problem in the case of instance $I = (J, F, \ll, m)$ only means computing the function T , which with any job J_i associates some time-unit $T(i)$ in such a way that:*

1. For any job J_i , $T(i) \in F(i)$;
2. For any pair of jobs J_i, J_j , such that $J_i \ll J_j$, we have $T(i) \leq T(j)$;
3. For any time-unit t , $n_T(t) = \text{Card}(T^{-1}(t)) \leq m = \text{the number of machines}$;
4. The function $t \rightarrow n_T(t)$ has a pyramidal shape.

A function T which satisfies 1 and 3 above will be called a *m-matching*. In case it also satisfies 2, it will be called a *pre-schedule*. In case it satisfies all conditions 1..4, we shall also call T a *feasible schedule* of the $NON-IDLE_0$ of the instance $I = (J, F, \ll, m)$.

D. The Makespan Minimization Homogenous Non-Idling Scheduling Problem

Let $I = (J, F, \ll, m)$ as above, and let T be some feasible schedule for I . The *Makespan* of T is the cardinality of its active time-unit set $\text{Act}(T)$. Then the *Makespan Minimization*

tion Homogeneous Non-Idling Problem NON-IDLE_1 comes as follows:

NON-IDLE_1 : {Compute a feasible schedule T of $I = (J, F, \ll, m)$ with a minimal makespan $\text{Card}(\text{Act}(T))$ }.

Notice that, while setting this problem, we do not require our schedule T to start at instant 0.

III. STRUCTURAL RESULTS FOR THE NON-IDLE_0 FEASIBILITY PROBLEM

In order to proceed to the analysis of the feasibility problem NON-IDLE_0 , we need to introduce some additional concepts.

A. T -Block, k -holes, k -Schedules.

Let T some pre-schedule of the NON-IDLE_0 instance $I = (J, F, \ll, m)$. We denote by $s(T)$ (respectively $e(T)$) the smallest (respectively largest) time-unit such that at least one job is performed at time-unit t . We say that an interval $\Omega \subseteq \text{ACT}(T) = \{s(T), \dots, e(T)\}$ is a T -block if every job J_i which is scheduled inside Ω is such that $F(i) \subseteq \Omega$. A time-unit t is then a k -hole for T , where k is some positive number, if there exists time-units t', t'' such that:

- $t' < t < t''$;
- $\text{Inf}(n_T(t'), n_T(t'')) > n_T(t) = k$ (in the following figure 2, time-unit 6 is a 2-hole and time-unit 7 is a 1-hole).

Clearly, T is a feasible schedule if only if, for any k in $\{0..m-1\}$, it has no k -hole. This leads us to introduce the intermediate notion of k -schedule: the pre-schedule T is a k -schedule if T has no l -hole for $l = 0..k-1$, or, equivalently, if the function $t \rightarrow \text{Inf}(k, n_T(t))$ has a pyramidal shape. The time diagram of Figure 2 represents a pre-schedule which is a 1-schedule, but not a 2-schedule since time-unit 7 is a 1-hole. Obviously, a feasible schedule is a m -schedule and conversely.

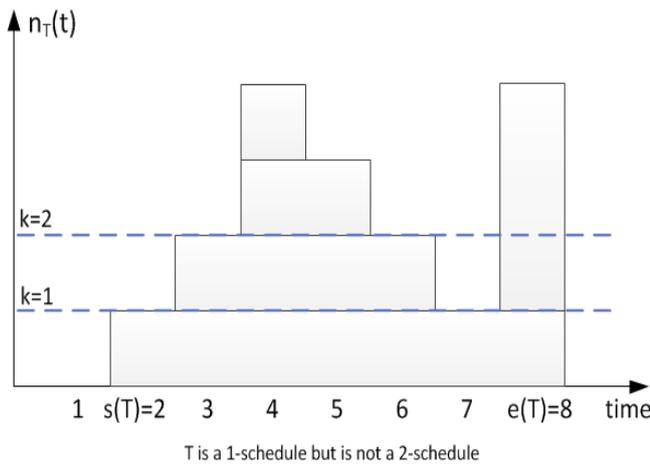


Figure 2: k holes

B. Time-windows stability.

The family of time-windows $F(i)$, $i = 1..n$, is *stable* with respect to the precedence relation \ll if, for any pair of jobs J_i, J_j such that $J_i \ll J_j$, we have: $r_i \leq r_j$ and $d_i \leq d_j$. The following result, which is almost obvious, shows that we may assume, without any loss of generality, that our input family of time windows is stable with respect to the precedence relation \ll :

Proposition 1: Let $I = (J, F, \ll, m)$ be an instance of the NON-IDLE_0 problem. There exists an instance $I' = (J, F', \ll, m)$, which may be obtained from I through constraint propagation, which admits the same set of feasible solution as f , which is such that: F' is stable with respect to \ll and, for any I , $F'(i) \subseteq F(i)$.

Since the goal of this paper is mainly to provide a characterization of the feasible instances of the NON-IDLE_0 problem, together with recognition and makespan minimization algorithms, we proceed in several steps. First, we use the König-Hall Theorem in order to characterize the instances which admits a m -matching. Then, we show that any m -matching may be turned into a pre-schedule. We keep on by identifying a structural property which is going to make possible turning this pre-schedule into a feasible schedule. Finally, we translate this mathematical characterization into a recognition algorithm.

C. Existence of a m -matching.

Let $I = (J, F, \ll, m)$ some NON-IDLE_0 instance. For any interval Ω , we denote by $J(\Omega)$ the set of all jobs J_i , such that $F(i) \subseteq \Omega$. Then one may derive in a straightforward way from classical König-Hall Theorem related to the existence of generalized matching in bipartite graphs that:

Proposition 2: The instance (J, F, \ll, m) admits a m -matching if and only if, for any interval Ω of N , we have $\text{Card}(J(\Omega)) \leq m \cdot \text{Card}(\Omega)$.

D. Existence of a Pre-schedule.

The next property is mainly due to our assumption about the stability of F . It will be essential for us when we shall have to deal with the precedence relation \ll .

Proposition 3: The instance (J, F, \ll, m) admits a pre-schedule if and only if it admits a m -matching.

Principle of the proof: starting from a m -matching T , one only has to iteratively exchange $T(i)$, $T(j)$ values in order to make T compatible with the \ll precedence relation.

E. Existence of a Feasible Schedule.

Let $I = (J, F, \ll, m)$ our NON-IDLE_0 instance. If the condition provided by Proposition 3 is satisfied, we easily become to produce some pre-schedule T . However, such a pre-schedule may have k -holes and thus may not fit the "pyramidal" property required for the feasible schedules. This section will provide an additional necessary and sufficient condition so that an instance $I = (J, F, \ll, m)$ admits some feasible schedule. Before deriving this condition, we

first give two simple lower bound properties which must be met by such a feasible schedule and then introduce the notion of propagation path which will prove to be a quite useful tool either to transform a pre-schedule into a feasible schedule or to prove the no existence of such a feasible schedule.

Let Ω be an interval of \mathbf{N} . We denote by $Int(\Omega)$ the set of intervals which are contained into Ω and by $\lambda(\Omega)$ the integer value $\text{Sup}_{\omega \in Int(\Omega)} \lceil \text{Card}(J(\omega)) / \text{Card}(\omega) \rceil$. Then the meaning of those definitions comes in an immediate way through the almost trivial following lemma:

Lemma 1: *Let T some be m -matching of $I = (J, F, \ll, m)$ and let Ω be an interval of \mathbf{N} . There must exist at least one time-unit in Ω such that at least $\lambda(\Omega)$ machines are busy.*

We understand the main role of $\lambda(\Omega)$ is to provide us with a lower bound of the number of machines which are going to be necessary if we want to succeed in scheduling the jobs of $J(\Omega)$. Notice that if Ω is a T -block, then we have $\lambda(\Omega) \geq \lceil \sum_{t \in \Omega} n_T(t) / \text{Card}(\Omega) \rceil$, since, in this case, $J(\Omega)$ is exactly the set of the jobs which are scheduled inside Ω .

Let us assume now that Ω_1 and Ω_2 are two intervals of \mathbf{N} such that $\Omega_1 \text{ Dom } \Omega_2$. If we denote by $\mu(\Omega_1, \Omega_2)$ the value $\text{Card}(\text{Mid}(\Omega_1, \Omega_2)) \cdot \text{Inf}(\lambda(\Omega_1), \lambda(\Omega_2))$, then we easily see that we get:

Lemma 2: *In any feasible schedule of $I = (J, F, \ll, m)$, at least $\mu(\Omega_1, \Omega_2)$ jobs are scheduled in the interval $\text{Mid}(\Omega_1, \Omega_2)$.*

Also, the following two properties, which come in a straightforward way and which are related to T -blocks, will be useful in order to derive the main characterization result:

Lemma 3: *Let T be some m -matching of $I = (J, F, \ll, m)$ and let Ω_1 and Ω_2 be two connected T -blocks. Then $\Omega_1 \cup \Omega_2$ is a T -block and $\lambda(\Omega_1 \cup \Omega_2) \geq \text{Sup}(\lambda(\Omega_1), \lambda(\Omega_2))$.*

Lemma 4: *Let T be some m -matching of $I = (J, F, \ll, m)$, let Ω_1 be a T -block, and let Ω_2 be an interval such that $\Omega_1 \cap \Omega_2$ is empty. If, for any pair (u, t) in $\Omega_1 * \Omega_2$, $n_T(u) \geq n_T(t)$ (respectively $n_T(u) > n_T(t)$), then $\lambda(\Omega_1) \geq \lambda(\Omega_2)$ (respectively $\lambda(\Omega_1) > \lambda(\Omega_2)$).*

Given a m -matching T , we now define what is a propagation path of T . We first define the *propagation graph* $G = (H, E(T))$ as the labeled directed graph whose node set is the interval $H = \{r_{\min}, \dots, d_{\max}\}$ of the possible values for T , and the arc set $E(T)$ is defined by:

- $[t, t'] \in E(T)$ iff $t \neq t'$ and there is at least one job J_i which is scheduled at t and which is such that $t' \in F(i)$.
- If job J_i is scheduled at t and $t' \in F(i)$, then J_i is said to be a *label* of the arc $[t, t']$.

Then, a *propagation path* of T is an elementary path $\gamma = (t_0, \dots, t_k)$ of $G(T)$. The sub-path of γ from t_i to t_j will be denoted by $\gamma(t_i, t_j)$. The *length* $L(\gamma)$ of γ is the value $k + 1$

(that means the number of vertices of γ), and the *extended length* $L^*(\gamma)$ of γ is the sum $\sum_{s=1..k} |t_s - t_{s-1}|$.

This propagation path γ is *monotone* if the sequence (t_0, \dots, t_k) is either decreasing or increasing. It is *no-cross* if for any $r \in \{1..k\}$, we have either $t_r > \text{Sup}_{i=1..r-1} t_i$ or $t_r > \text{Inf}_{i=1..r-1} t_i$. Clearly, the no-cross property may be viewed as a weak version of monotonicity. It is *labeled* when all its arcs are assigned with labels, that means when, with any arc in γ , we decided to associate some job J_i whose value $T(i)$ is likely to be modified through shiftpropagation along γ .

The path γ is said to be *fitted* if it is labeled and satisfies $\text{Card}(T^{-1}(t_k)) < m$. Of course, we understand that if $\gamma = (t_0, \dots, t_k)$ is a propagation path of $G(T)$, which is fitted and provided with the labeling $\sigma = (J_{i(0)}, \dots, J_{i(k-1)})$, then we become able to modify the m -matching T and get another m -matching $T' = \text{Trans}(T, \gamma, \sigma)$ by setting $T'(J_{i(p)}) = t_{p+1}$ for any $p = 0, \dots, k-1$.

Let T be a pre-schedule and let $\gamma = (t_0, \dots, t_k)$ be a propagation path of the graph $G(T)$. If γ has at least one labeling σ such that $T' = \text{Trans}(T, \gamma, \sigma)$ is a pre-schedule, then γ is said to be *compatible* with the precedence relation \ll (\ll -compatible in short).

The two following lemmas are fundamental tools, which show that no-cross propagation paths allow to transform a pre-schedule into another one:

Lemma 5: *Let T be a pre-schedule and let us assume that $\gamma = (t_0, \dots, t_k)$ is a propagation path of $G(T)$ from $t_0 = u$ to $t_k = v$. Then there exists a no-cross propagation path from u to v in $G(T)$.*

Proof: Assume that γ is not no-cross and let t_r ($2 \leq r \leq q$) be the first node of γ such that $\min_{i=1..r-1} t_i < t_r < \max_{i=1..r-1} t_i$. From the definition of t_r , we know that there is a smallest index s ($0 \leq s \leq r - 2$) such that t_r belongs to $I(t_s, t_{s+1})$. Thus $[t_s, t_r]$ is an arc of $G(T)$ and the concatenation $\gamma(t_0, t_s)$. $[t_s, t_r]$ is no-cross. The above transformation may then be iterated while the current propagation path is not no-cross.

Lemma 6: *Let T be a pre-schedule and let us assume that $\gamma = (t_0, \dots, t_k)$ is a propagation path of $G(T)$ from $t_0 = u$ to $t_k = v$, where $\text{Card}(T^{-1}(t_k)) < m$. Then there exists a no-cross and \ll -compatible propagation path from u to v in $G(T)$.*

Sketch of the proof. Define the extended length of such a path γ in $G(T)$ as the sum $\sum_{i=1..k} |t_i - t_{i-1}|$, and consider a propagation path γ from u to v with minimal extended length and whose length is maximal among the paths with minimal extended length. Assume also that γ is not \ll -compatible and let $\sigma = (J_{i(0)}, \dots, J_{i(k-1)})$ be a labeling of γ . The jobs of the labeling are called the *moving* jobs, while the other jobs are called the *static* jobs. Since T is a pre-schedule and γ is not \ll -compatible, \ll is violated in $T' = \text{Trans}(T, \gamma, \sigma)$ either because an inversion has been created between a static job J' scheduled at t' and a moving job $J_{i(s)}$ scheduled at t_{s+1} or

between two moving jobs $J_{i(s)}$ and $J_{i(r)}$ respectively scheduled at time t_{s+1} and t_{r+1} . In both case, one checks that it is possible to rearrange path γ in order to get a contradiction on the minimality of γ .

Let T be a pre-schedule and let u be a time-unit such that $n_T(u) > 0$. The next lemma, whose prove is essentially routine, provides us with an important property of the set $A_T(u)$ of the time-units that may be reached from u by the propagation paths of $G(T)$.

Lemma 7: *The set $A_T(u)$ is a T -block.*

We are now able to describe and state the structural condition which must be met by an instance $I = (J, F, \ll, m)$ of the NON-IDLE₀ problem so that it admits some feasible schedule.

Theorem 2: *The instance $I = (J, F, \ll, m)$ of NON-IDLE₀ is feasible if and only if the following two conditions are satisfied:*

1. For any interval Ω of N , $\text{Card}(J(\Omega)) \leq m \cdot \text{Card}(\Omega)$.
2. For any sequence $(\Omega_1, \dots, \Omega_p)$ of intervals of N , such that $\Omega_1 \text{ Dom } \dots \text{ Dom } \Omega_p$, we have:

$$\text{Card}(J - \cup_{s=1..p} J(\Omega_s)) \geq \sum_{s=1..p-1} \mu(\Omega_s, \Omega_{s+1}).$$

Sketch of the Proof.

The ‘‘only if’’ part of the proof is a straightforward consequence of propositions 2 and 4 and lemmas 1, 2, 3

Before starting the proof of the part ‘‘if’’, let us recall that, for any $k = 1..m$, a pre-schedule T is a k -schedule if T has no l -hole for any $l = 0..k-1$. So we may adapt the definition of the quantity $\mu(\Omega_1, \Omega_2)$, where Ω_1 and Ω_2 are two time intervals such that $\Omega_1 \text{ Dom } \Omega_2$, to k -schedules by setting: $\mu^*(\Omega_1, \Omega_2, k) = \text{Card}(\text{Mid}(\Omega_1, \Omega_2)) \cdot \text{Inf}(\lambda(\Omega_1), \lambda(\Omega_2), k)$. Then one easily checks:

Lemma 8: *Let Ω_1 and Ω_2 be two time intervals such that $\Omega_1 \text{ Dom } \Omega_2$. In any k -schedule of $I = (J, F, \ll, m)$, at least $\mu^*(\Omega_1, \Omega_2, k)$ jobs are scheduled in the interval $\text{Mid}(\Omega_1, \Omega_2)$.*

In order to prove the ‘‘if’’ part, we extend the statement of Theorem 2 to k -schedules and prove it by induction on k . Using k -schedules allow us to try to perform an inductive reasoning in order to prove Theorem 2. As a matter of fact, what we have to do is to prove the following statement, which may be viewed as an inductive extension of Theorem 2 to k -schedules :

Inductive Formulation of Theorem 2: *The instance $I = (J, F, \ll, m)$ of NON-IDLE₀ admits at least one k -schedule if and only if the following two conditions are satisfied:*

1. For any interval Ω of N , $\text{Card}(J(\Omega)) \leq m \cdot \text{Card}(\Omega)$.
2. For any sequence $(\Omega_1, \dots, \Omega_p)$ of intervals of N , such that $\Omega_1 \text{ Dom } \dots \text{ Dom } \Omega_p$, we have:

$$\text{Card}(J - \cup_{s=1..p} J(\Omega_s)) \geq \sum_{s=1..p-1} \mu^*(\Omega_s, \Omega_{s+1}, k).$$

In order to prove this last statement, we proceed by induction on k . More precisely, we show that if an instance $I = (J, F, \ll, m)$ of the NON-IDLE₀ problem has at least a k -schedule and no $(k+1)$ -schedule, then there exists a sequence of intervals that does not satisfy condition 2 of the above statement. In order to get such a sequence, we consider a k -schedule of I , which is *doubly* minimum in the following sense:

- T has a minimum number of k -holes;
- the vector $(N_T(m), \dots, N_T(1))$, where $N_T(j)$ is the number of time-units t such that $n_T(t) = j$, is lexicographically minimum.

Thus, T may be viewed as the flattest k -schedule with a minimum number of k -holes.

The graph of the piecewise constant function $t \rightarrow n_T(t)$ may be decomposed into 3 parts:

- a *left* part, which is a *climbing stair* (increasing piecewise constant function) for which we denote by C^{Inf_p} , $1 \leq p \leq k+1$, the smallest time-unit at which the stair height is at least equal to p ;
- a *right* part, which is a *descending stair* (piecewise constant decreasing function) for which we denote by C^{Sup_p} , $1 \leq p \leq k+1$, the largest time-unit at which the stair height is at least equal to p ;
- a *medium* part, which is made of the time-units u such that $n_T(u) \geq k+1$, and which is such that at least one time-unit is a k -hole.

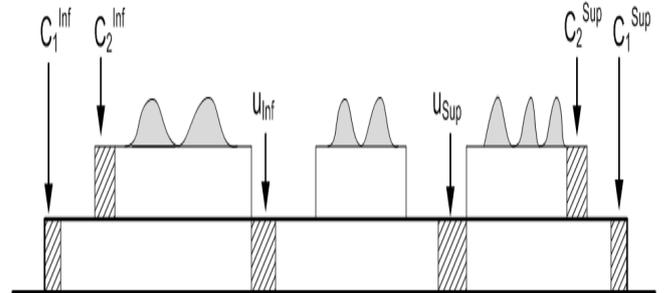


Figure 3: $t \rightarrow n_T(t)$ segmentation

From the minimality of T , we easily get that the propagation graph $G(T)$ is such that there is no propagation path:

- from a time-unit t which satisfies $n_T(t) \geq k+2$ to a time-unit u which is a k -hole; (E1)
- from a time-unit C^{Sup_p} or C^{Inf_p} to a time-unit u which is a k -hole; (E2)
- from a time-unit C^{Sup_p} to a time-unit $C^{\text{Sup}_j} + 1$, $1 \leq j < p$; (E3)

- from a time-unit C_p^{inf} to a time-unit $C_p^{\text{inf}} + 1$, $1 \leq j < p$; (E4)

From what precedes, we deduce 3 families of intervals:

- the *right* family $L_1..L_l$, which we get from the T -blocks $A_T(C_p^{\text{sup}})$, $1 \leq p \leq k+1$, by merging those intervals which are connected. By using lemmas 7 and 8, we may check that, for any $i = 1..l - 1$:

$$\begin{aligned} \sum_{t \in \text{Mid}(L_i, L_{i+1})} n_T(t) &= \lambda(L_i) \cdot \text{Card}(\text{Mid}(L_i, L_{i+1})) \\ &\leq \mu^*(L_i, L_{i+1}, k+1); \end{aligned}$$

- the *left* family $L^*_1..L^*_h$, which we get from the T -blocks $A_T(C_p^{\text{inf}})$, $1 \leq p \leq k+1$, by merging those intervals which are connected. By using lemmas 7 and 8, we may check that, for any $i = 1..h - 1$:

$$\begin{aligned} \sum_{t \in \text{Mid}(L^*_i, L^*_{i+1})} n_T(t) &= \\ \lambda(L^*_i) \cdot \text{Card}(\text{Mid}(L^*_i, L^*_{i+1})) & \\ &\leq \mu^*(L^*_i, L^*_{i+1}, k+1); \end{aligned}$$

- the *medium* family $J_1..J_q$, which we get from the T -blocks $A_T(u)$, u such that $n_T(u) \geq k + 2$, by merging those intervals which are connected. By using lemmas 7 and 8, we check that, for any $i = 1..l - 1$:

$$\begin{aligned} \sum_{t \in \text{Mid}(J_i, J_{i+1})} n_T(t) &= \lambda(J_i) \cdot \text{Card}(\text{Mid}(J_i, J_{i+1})) \\ &\leq \mu^*(J_i, J_{i+1}, k+1); \end{aligned}$$

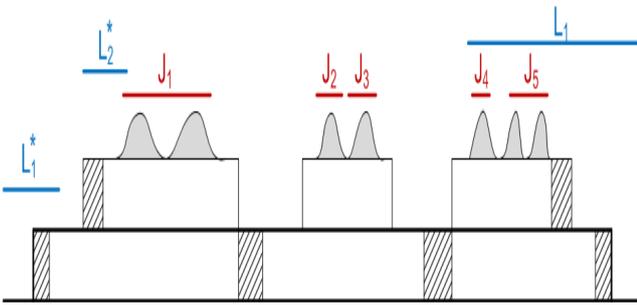


Figure 4: $t \rightarrow n_T(t)$ Interval Decomposition

The key point is that, because of (E1, ..., E4), those intervals define several distinct connected blocks, with non empty space between them (we must find at least the k -holes, which are going to separate those intervals).

Still, since we may have $L^*_1 \cap J_1 \neq \text{Nil}$ or $L_1 \cap J_q \neq \text{Nil}$, we merge once again the connected intervals of those three families. Then we get a sequence of non connected intervals $M_1..M_r$, of disjoint T -blocks such that:

- $M_1 \text{ Dom } .. \text{ Dom } M_r$;
- The number of jobs which is to in $\cup_{j=1..r} J(M_j)$ is not large enough to avoid the existence of a k -hole.

Then we conclude.

IV. A POLYNOMIAL ALGORITHM FOR THE NON-IDLE₀ FEASIBILITY PROBLEM AND FOR THE NON-IDLE₁ MAKESPAN MINIMIZATION PROBLEM

A. A Polynomial Algorithm for the NON-IDLE₀ Feasibility Problem.

The proof of Theorem 2 is not an algorithmic proof, since it involves an hypothesis about *doubly minimality* which has no algorithmic interpretation. In this section, we first show that a forbidden pattern of intervals may also be found from any k -schedule which satisfies a weaker set of conditions than that of the doubly minimal k -schedule T considered in the proof of Theorem 2. Its is going to allow us to turn this doubly minimality condition into conditions which might be used as halting test inside the “while” loop of a recognition algorithm. Then we derive from this a polynomial algorithm which solves the NON-IDLE₀ Problem, and whose correctness mainly relies on this new set of sufficient conditions.

So, let T be a k -schedule and let U and V be two disjoint sets of time-units: we denote by $PP(U, V)$ the set of propagation paths of $G(T)$ which start in U and end into V . We denote by $Hole(k)$ the set of time-units t which are k -holes, and by $Top(k)$ the set of time-units t which satisfies $n_T(u) \geq k + 2$. Then we may state:

Theorem 3. Let $I = (J, F, \ll, m)$ be an instance of the NON-IDLE₀ problem, and let k be some number in $\{1, \dots, m-1\}$. Let us assume that T is a k -schedule of I such that the following conditions are satisfied:

1. T is not a $(k+1)$ -schedule;
2. $PP(\text{Top}(k) \cup \{C_1^{\text{inf}}, \dots, C_{k+1}^{\text{inf}}\} \cup \{C_1^{\text{sup}}, \dots, C_{k+1}^{\text{sup}}\}, \text{Hole}(k))$ is empty;
3. $PP(\text{Top}(k), \{C_1^{\text{inf}} - 1, \dots, C_{k+1}^{\text{inf}} - 1\} \cup \{C_1^{\text{sup}} + 1, \dots, C_{k+1}^{\text{sup}} + 1\})$ is empty;
4. For $j = 2..k+1$, $PP(\{C_j^{\text{sup}}\}, \{C_{j-1}^{\text{sup}} + 1\})$ is empty;
5. For $j = 2..k+1$, $PP(\{C_j^{\text{inf}}\}, \{C_{j-1}^{\text{inf}} - 1\})$ is empty;

Then the instance I has no $(k+1)$ -schedule.

This result allows us to the following algorithm $\text{SEARCH-SCHEDULE}(J, F, \ll, m)$ which solves the NON-IDLE_0 problem by providing, for any instance $I = (J, F, \ll, m)$ either a feasible schedule of I or a k -schedule T , (with $0 \leq k \leq m-1$) of I which satisfies the conditions 1..5 of the above statement. Notice that $Hole(k)$, $Top(k)$, C^{Sup}_p , C^{Inf}_p denote here variables that contains the values associated with the current k -schedule T .

Algorithm $\text{SEARCH-SCHEDULE}(J, F, \ll, m)$:

$T \leftarrow \text{Matching}(J, F, m)$; (*Computation of an initial m -matching, through a standard matching procedure*)

If T does not exist (*Condition of Propositions 2 and 3*) then $\text{SEARCH-SCHEDULE} \leftarrow \text{Fail}$

Else

$T \leftarrow \text{Pre-Schedule}(J, F, m, \ll, T)$; (*Turn T into a pre-schedule through a sequence of exchanges, according to proposition 3*)

$k \leftarrow \text{Sup}_{l=0..m} l$ such that T is a l -schedule; Not Stop;

While $k < m$ and Not Stop do

Try, according to this order, the existence of a propagation path γ in:

- $PP(Top(k) \cup \{C^{\text{Inf}}_1, \dots, C^{\text{Inf}}_{k+1}\} \cup \{C^{\text{Sup}}_1, \dots, C^{\text{Sup}}_{k+1}\}, Hole(k))$;
- $PP(Top(k), \{C^{\text{Sup}}_1 + 1, \dots, C^{\text{Sup}}_{k+1} + 1\} \cup \{C^{\text{Inf}}_1 - 1, \dots, C^{\text{Inf}}_{k+1} - 1\})$;
- $\cup_{j=2..k+1} PP(\{C^{\text{Sup}}_j, C^{\text{Inf}}_j\}, \{C^{\text{Inf}}_{j-1} - 1, C^{\text{Sup}}_{j-1} + 1\})$;

If $\text{Failure}(\text{Try})$ (*Non existence of γ^*) then Stop

Else

Let σ be a label of γ ;

$T \leftarrow \text{Trans}(T, \gamma, \sigma)$;

If $k = m$ then $\text{SEARCH-SCHEDULE} \leftarrow T$ else $\text{SEARCH-SCHEDULE} \leftarrow \text{Fail}$;

Theorem 4. *The above algorithm SEARCH-SCHEDULE solves the feasibility NON-IDLE_0 problem in polynomial time.*

Sketch of the proof: one follows the proof of Theorem 2, while replacing the *doubly minimality* assumption by the

conditions 1..5 of Theorem 3. As for complexity, it is easy to check that the above algorithm is time polynomial if the encoding size of a time-window $F(i)$ is defined as being proportional to the size of this time windows. But, in fact, it has to be related to the encoding size of both numbers $\min(F(i))$ and $\max(F(i))$. Still, one easily checks that, once an initial m -matching has been computed, time-windows may be restricted in such a way that the size of their union be polynomial bounded by the number n of jobs, and that provides us with the key argument for the time-polynomiality of our algorithm.

B. A Polynomial Algorithm for the NON-IDLE_1 Makespan Minimization Problem.

Makespan minimization is contained into feasibility testing, and comes in a simple way through the following process:

Makespan-Min-No-Idle-Schedule Algorithm.

Input: the instance $I = (J, F, \ll, m)$

Output: a no idle feasible schedule or a *Failure* signal;

Initialize T through the SEARCH-SCHEDULE procedure;

If $\text{Failure}(\text{Initialize})$ then *Failure*

Else

Not Stop;

While Not Stop do

$\Delta \leftarrow \text{Makespan}(T)$;

Let t_1 and t_2 respectively the smallest and largest active time-units according to T ;

For any job $J_i \in J$, set $F_{\Delta}(i) = F(i) \cap \{t_1 + 1, t_2\}$;

$T\text{-Aux} \leftarrow \text{SEARCH-SCHEDULE}(J, F_{\Delta}, \ll, m)$;

If $T\text{-Aux} \neq \text{Failure}$ then $T \leftarrow T\text{-Aux}$

Else

For any job $J_i \in J$, set $F_{\Delta}(i) = F(i) \cap \{t_1, \dots, t_2 - 1\}$;

$T\text{-Aux} \leftarrow \text{SEARCH-SCHEDULE}(J, F_{\Delta}, \ll, m)$;

If $T\text{-Aux} \neq \text{Failure}$ then $T \leftarrow T\text{-Aux}$

Else Stop.

$\text{Makespan-Min-No-Idle-Schedule} \leftarrow T$;

Theorem 5: *The above Makespan-Min-No-Idle-Schedule algorithm solves the Makepan Minimization NON-IDLE_1 Problem in Polynomial Time.*

Sketch of the Proof: the basic point here is that if T is some feasible schedule with active time-unit set $\text{ACT}(T) = [a, b]$, and if there exists a feasible schedule T' with smaller makespan than T , then T' may be computed inside the time-window $[a, b]$.

V. CONCLUSION

In this paper, we have just been studying a variant of the homogeneous m -machine non-idling problem, where weakly

dependent unit-jobs have to be scheduled within their time windows so that the non-idling constraints must be satisfied not only for each machine but for every subset of machines. A structural necessary and sufficient condition for an instance to be feasible has been provided and an algorithm has been developed for the existence problem. This algorithm may be extended to the case when a job dependent cost function is associated with the termination of each job and the makespan has to be minimized. However, several important questions about the complexity of more general problems with the same HNI constraints (non unit jobs, linear costs...) are still open. Also, it would be quite interesting to get the complexity status of the non-homogeneous variant of the problem (more frequent when it comes to applications) which corresponds to the case when the non-idling constraint has only to be satisfied on each machine or on a given subset of machines.

REFERENCES

- [1]. P.Baptiste: "Scheduling unit tasks to minimize the number of idle periods: a polynomial time algorithm for off-line dynamic power management"; *Research Report*, Laboratoire d'Informatique CNRS LIX, 2005.
- [2]. P.Chretienne: "On single-machine scheduling without intermediate delays"; *Discrete Applied Maths* 13-156, p 2543-2550, 2008.
- [3]. A.Jouglet: "Single-machine scheduling with no-idle time and release dates to minimize a regular criterion", *Journal of Scheduling* 15 (2), p 217-238 (2012).
- [4]. J.M.S.Valente and R.A.F.S.Alves: "An exact approach to early/tardy scheduling with release dates", *Computers and Operations Research* 32, p 2905-2917, 2005.
- [5]. K.Landis: "Group technology and cellular manufacturing in the Westvaco Los Angeles VH Department", *Project Report in IOM* 581, School of Business, University of Southern California, 1983.
- [6]. S.Irani, K.Pruhs: "Algorithmic problems in power management", *ACM Press*, vol 36, p 63-76, New York, USA, 2005.