A Multi-Machine Scheduling Problem with Global Non Idling Constraint

Philippe Chrétienne, Samuel Deleplanque, Alain Quilliot

A Multi-Machine Scheduling Problem with Global Non Idling Constraint

Philippe Chretienne*, Samuel Deleplanque, Alain Quilliot****

**LIP6, PARIS VI, 75015 PARIS*
***LIMOS, UMR CNRS 6158, Bat. ISIMA,, Labex IMOBS3*
BLAISE PASCAL University, France (e-mail: quilliot@isima.fr)

Abstract: We deal here with a multi-machine scheduling problem with non idling constraints, i.e constraints which forbids the interruption of the use of the processors. We reformulate the problem, while using specific pyramidal profile functions, next get a min-max feasibility criterion and finally derive exact polynomial algorithms for both the feasibility problem and the makespan optimization problem.

Keywords: multi-machine scheduling, non idling constraint, bipartite graph matching.

1. 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 happens that in some applications such (see Landis 1985), 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 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 (see Irani and Al. 2005), where for example each idling period has a cost and the total cost has to be minimized (see Baptiste 2005). Note that the non idling constraint does not ensure full machine utilization but 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 been scarcely studied. To the best of our knowledge, the first work on such problems concerns (see Valente and Al. 2005) the earliness-tardiness single machine scheduling problem with no unforced idle time. More recently, the impact of the non-idling constraints on the complexity of single-machine scheduling problems as well as the role played by the earliest starting time of a non-idling schedule has been studied in Chretienne 2008. Moreover, in Jouglet 2012, a Branch/Bound method has been developed for the one-machine non-idling scheduling problem.

In this paper, we study the basic global *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 profile function*, *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.

2. A PYRAMIDAL FORMULATION of the MULTI-MACHINE SCHEDULING PROBLEM with NON IDLING CONSTRAINTS

2.1 Notations: Time Representation

N denotes the discrete time space $\{0,.., +\infty\}$, whose elements are called *time-units*. A subset Ω of **N** is an *interval* if it is made 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 $I(p, q)$. Interval Ω_2 *dominates* interval Ω_1 if $max(\Omega_1) + 1 < min(\Omega_2)$, which we denote denoted by Ω_1 *Dom* Ω_2. If Ω_1 *Dom* Ω_2, then we denote by $Mid(\Omega_1, \Omega_2)$ the (non empty) interval $\{max(\Omega_1) +1 ,.., min(\Omega_2) - 1\}$. Two intervals are *connected* if their union is an interval.

2.2. The Feasibility Multi-Machine Scheduling Problem with Global Non Idling Constraints

We now suppose that we are given a set $J = \{J_1, .. , J_n\}$ of n unit-time jobs that are to be processed on a set $M = \{M_1, .., M_m\}$ of m identical machines. Job J_i must be executed inside a given time-window $F(i) = \{r_i, .. , d_i\}$ 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 $\{r_{min}, .., d_{max}\}$. The jobs are also constrained by a *weak* precedence relation denoted by << where $J_i << J_j$ 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 J_i, J_j, such that $J_i << J_i$, we have $T(i) \leq T(j)$;
- For any pair 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 ∈ **N**, such that there exist i = 1..n, with $T(i) = t$ 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 **N** defined by: $ACT(T) = \{t \in \mathbf{N}$ such that there exists at least some index value $i = 1..n$, with $T(i) = t\}$.

Then we define the related *Feasibility Multi-Machine Scheduling Problem with Global Non Idling Constraints* 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, <<, 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 $J_i << J_i$, 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.

2.3 A Pyramidal Reformulation of the NON-IDLE$_0$ Problem.

Let $I = (J, F, <<, m)$ be an instance of NON-IDLE$_0$ and (T, μ) some schedule of I. If $t \in \mathbf{N}$, we denote by $n_T(t)$ = 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 $t \rightarrow n_T(t)$ is a *pyramidal profile function* if for any time units t, t', t'' such that $t' < t < t''$, we have Inf($n_T(t')$, $n_T(t'')$) $\leq n_T(t)$ (see figure 1).

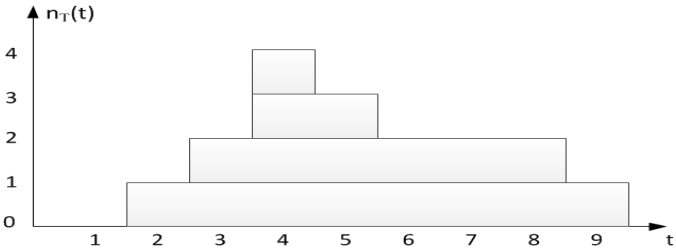

Figure 1: A pyramidal profile function $n_T(t)$

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 $\mu(t) = \{M_1, .., M_{nT(t)}\}$. 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 provides a reformulation of the NON-IDLE$_0$ problem 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 $I = (J, F, <<, m)$ 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 $J_i << J_i$, we have $T(i) \leq T(j)$;*
3. *For any time-unit t, $n_T(t) = Card(T^{-1}(t)) \leq m$ = 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 $I = (J, F, <<, m)$.

2.4. The Makespan Minimization NON-IDLE$_1$ Problem

Let $I = (J, F, <<, m)$ as above, and let T be some feasible schedule for I. The Makespan of T is the cardinality of its active time-unit set **Act**(T). Then the *Makespan Minimization Multi-Machine Problem with Global Non-Idling Problem* NON-IDLE$_1$ comes as follows:

NON-IDLE$_1$: {*Compute a feasible schedule T of $I = (J, F, <<, m)$ with a minimal makespan Card(Act(T))*}.

Notice that, while setting this problem, we do not require our schedule T to start at instant 0, and, also, that it would be possible to take into account costs $C_{j,t}$ related, for any job j, to the date t when j is run.

III. STRUCTURAL ANALYSIS of NON-IDLE$_0$.

In order to proceed to the analysis of the feasibility problem NON-IDLE$_0$, we need to introduce some additional concepts.

3.1. Blocks, Holes, k-schedules, Time Window Stability

Let T some pre-schedule of the NON-IDLE$_0$ instance $I = (J, F, <<, m)$. 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 $\Omega \subseteq ACT(T) = \{s(T), .., e(T)\}$ is a *T-block* if every job J_i which is scheduled inside Ω is such that $F(i) \subset \Omega$. A time-unit t is then a *k-hole* for T, where k is some positive number, if there exists time-units t, t', 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. Clearly, a feasible schedule is a m-schedule and conversely.

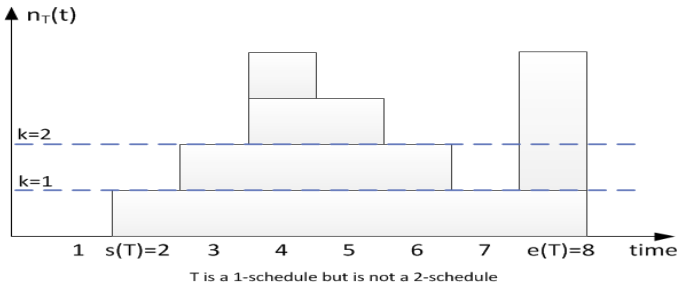

Figure 2: k holes

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

Proposition 1: Let *$I = (J, F, <<, m)$ be an instance of the NON-IDLE$_0$ problem. There exists an instance $I' = (J, F', <<, m)$, 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 << 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 Konig-Hall Theorem related to *matchings in bipartite graphs* 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.

3.2. Existence of a m-matching and of a pre-schedule.

Let $I = (J, F, <<, m)$ 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 Konig-Hall Theorem related to the existence of generalized matching in bipartite graphs that:

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

The next property mainly derive from the stability of F. It will help us in dealing with the precedence relation <<.

Proposition 3: *The instance $(J, F, <<, m)$ admits a pre-schedule if an 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 << precedence relation.

3.3. Existence of A Feasible Schedule.

Let $I = (J, F, <<, m)$ our NON-IDLE$_0$ instance. If the condition provided by Proposition 3 is satisfied, we easily become able 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 $I = (J, F, <<, m)$ 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 $I = (J, F, <<, m)$ 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 $\lambda(\Omega) \geq \lceil \Sigma_{t \in \Omega} n_T(t)/\text{Card}(\Omega) \rceil$, 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 Ω_1 *Dom* Ω_2. If we denote by $\mu(\Omega_1, \Omega_2)$ the value $\text{Card}(Mid(\Omega_1, \Omega_2)).\text{Inf}(\lambda(\Omega_1), \lambda(\Omega_2))$, then we get:

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

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, <<, m) and let Ω_1 and Ω_2 be two connected T-blocks. Then $\Omega_1 \cup \Omega_2$ is a T-block and $\lambda(\Omega_1 \cup \Omega_2) \geq Sup(\lambda(\Omega_1), \lambda(\Omega_2))$.*

Lemma 4: *Let T be some m-matching of I = (J, F, <<, m), let Ω_1 be a T-block, and let Ω_1 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 $\lambda(\Omega_1) > \lambda(\Omega_1)$).*

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 $H = \{r_{min}, .., d_{max}\}$ 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 $\gamma = (t_0, .., t_k)$ 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 $\Sigma_{s = 1..k} |t_s - t_{s-1}|$.

This propagation path γ is *monotone* if the sequence $(t_0, .., t_k)$ 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 f $\gamma = (t_0, .., t_k)$ is a propagation path of *G*(*T*), which is fitted and provided with the labeling $\sigma = (J_{i(0)}, .., J_{i(k-1)})$, then we brcome able to modify the *m*-matching *T* and get another *m*-matching $T' = Trans(T, \gamma, \sigma)$ by setting $T'(J_{i(p)}) = t_{p+1}$ for any $p = 0,.., k\text{-}1$.

Let *T* be a pre-schedule and let $\gamma = (t_0, .., t_k)$ be a propagation path of the graph *G*(*T*). If γ has at least one labeling σ such that $T' = Trans(T, \gamma, \sigma)$ is a pre-schedule, then γ is said to be *compatible* with the precedence relation << (*<<-compatible* in short). Then two following lemmas show the way no-cross propagation paths may turn a pre-schedule into another one:

Lemma 5: *Let T be a pre-schedule and let us assume that $\gamma = (t_0, .., t_k)$ 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 $\gamma = (t_0, .., t_k)$ is a propagation path of G(T) from $t_0 = u$ to $t_k = v$, where $Card(T^{-1}(t_k)) < m$. Then there exists a no-cross and <<-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 $\Sigma_{i = 1..k} \left| t_i - t_{i-1} \right|$, 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 <<-compatible and let $\sigma = (J_{i(0)}, .., J_{i(k-1)})$ 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 <<-compatible, << is violated in $T' = Trans(T, \gamma, \sigma)$ because an inversion either 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 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 a NON-IDLE$_0$ instance $I = (J, F, <<, m)$ so that it admits some feasible schedule.

Theorem 2: *The instance I = (J, F, <<, m) of NON-IDLE$_0$ is feasible if and only if:*

1. *For any interval Ω of **N**, $Card(J(\Omega)) \leq m.Card(\Omega)$.*
2. *For any sequence $(\Omega_1,..\Omega_p)$ of intervals of N, such that $\Omega_1 Dom…Dom\ \Omega_p$,we have:*
   $Card(J - \cup_{s = 1..p} J(\Omega_s)) \geq \Sigma_{s = 1..p-1}\ \mu(\Omega_s, \Omega_{s+1})$.

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 dealing with the "if"part, 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 $l = 0..k\text{-}1$. So we adapt the definition of the quantity $\mu(\Omega_1,\Omega_2)$, where Ω_1 and Ω_2 are two time intervals such that Ω_1 *Dom* Ω_2, to *k*-schedules by setting: $\mu^*(\Omega_1,\Omega_2, k) = \text{Card}(Mid(\Omega_1, \Omega_2)).\text{Inf}(\lambda(\Omega_1), \lambda(\Omega_2, k)$. Then one checks:

Lemma 8: *Let Ω_1 and Ω_2 be two time intervals such that Ω_1 DomΩ_2. In any k-schedule of I = (J, F, <<, m), at least $\mu^*(\Omega_1,\Omega_2, k)$ jobs are scheduled in the interval Mid(Ω_1, Ω_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, and to prove the following inductive extension of Theorem 2 to *k*-schedules :

Inductive Formulation of Theorem 2: *The instance I = (J, F, <<, m) of NON-IDLE$_0$ admits at least one k-schedule if and only if the following two conditions are satisfied:*

1. *For any interval Ω of **N**, Card(J(Ω)) ≤ m.Card(Ω).*
2. *For any sequence (Ω_1,..Ω_p) of intervals of **N**, such that Ω_1 Dom… Dom Ω_p ,we have: Card(J - $\cup_{s = 1..p}$ J(Ω_s)) ≥ $\Sigma_{s = 1..p-1}$ $\mu^*(\Omega_s, \Omega_{s+1},k)$.*

In order get it, we proceed by induction on *k*. More precisely, we show that if an instance *I* = (*J*, *F*, <<, *m*) of the NON-IDLE$_0$ 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 *T* of *I*, such that:

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

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

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

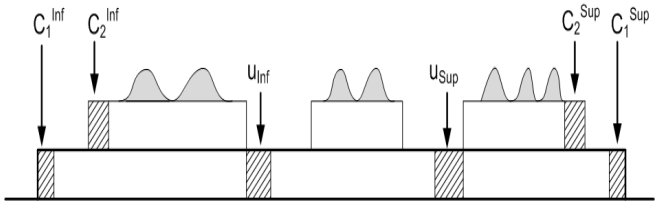

Figure 3: t -> $n_T(t)$ segmentation

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

- from *t* which satisfies $n_T(t) \geq k + 2$ to a *k*-hole; (E1)
- from C^{Sup}_p or C^{Inf}_p to a *k*-hole; (E2)
- from time-units C^{Sup}_p to $C^{Sup}_j + 1$, $1 \leq j < p$; (E3)
- from time-units C^{Inf}_p to $C^{Inf}_j + 1$, $1 \leq j < p$; (E4)

From what precedes, we deduce 3 families of intervals:

- the *right* family $L_1..L_l$, which we get from the *T*-blocks $A_T(C^{Sup}_p)$, 1 ≤ *p* ≤ *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: $\Sigma_{t \in Mid(Li, Li+1)}$ $n_T(t)$ = $\lambda(L_i)$.Card($Mid(L_i, L_{i+1})$) ≤ $\mu^*(L_i, L_{i+1}, k+1)$;
- the *left* family $L^*_1..L^*_h$, which we get from the *T*-blocks $A_T(C^{Inf}_p)$,1 ≤ *p* ≤ *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: $\Sigma_{t \in Mid(L^*i, L^*i+1)}$ $n_T(t)$ = $\lambda(L^*_i)$.Card($Mid(L^*_i, L^*_{i+1})$ ≤ $\mu^*(L^*_i, L^*_{i+1}, k+1)$;
- 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: $\Sigma_{t \in Mid(Ji, Ji+1)}$ $n_T(t)$ = $\lambda(J_i)$.Card($Mid(J_i, J_{i+1})$ ≤ $\mu^*(J_i, J_{i+1}, k+1)$;

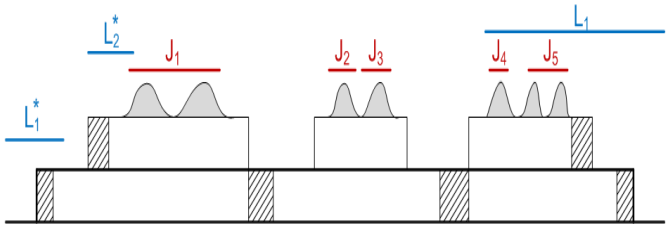

Figure 4: t -> $n_T(t)$ Interval Decomposition

Because of (E1, .., E4), those intervals define distinct connected blocks, with non empty space between them.

Still, since we may have $L^*_1 \cap J_1 \neq$ Nil or $L_l \cap J_q \neq$ 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 *Dom* .. *Dom* M_r and number of jobs which is in $\cup_{j = 1..r}$ $J(M_j)$ is not large enough to avoid the existence of a *k*-hole. Then we conclude.

IV. POLYNOMIAL ALGORITHMS

4.1. A Exact Polynomial Algorithm for the NON-IDLE$_0$ 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. It allows us to get conditions which might be used as halting test inside the "while" loop of a recognition algorithm, and to derive a exact polynomial algorithm which solves NON-IDLE$_0$, 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: *PP*(*U*, *V*) is the set of propagation paths of *G*(*T*) which start in *U* and end into *V*, *Hole*(*k*) is the set of time-units *t* which are *k*-holes, and *Top*(*k*) is the set of time-units *t* which satisfies $n_T(u) \geq k + 2$. Then we may state:

Theorem 3. *Let $I = (J, F, <<, m)$ be an instance of NON-IDLE$_0$, and k in $\{1, .., m-1\}$. 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. *$PP(Top(k) \cup \{C^{Inf}_1,..,C^{Inf}_{k+1}\} \cup \{C^{Sup}_1,..,C^{Sup}_{k+1}\}), Hole(k))$ is empty;*
3. *$PP(Top(k), \{C^{Inf}_1 -1,..,C^{Inf}_{k+1} -1\} \cup \{C^{Sup}_1 +1,.., C^{Sup}_{k+1} +1\})$ is empty;*
4. *For $j = 2..k+1$, $PP(\{C^{Sup}_j\}, \{ C^{Sup}_{j-1} + 1\})$ is empty;*
5. *For $j = 2..k+1$, $PP(\{C^{Inf}_j\}, \{ C^{Inf}_{j-1} - 1\})$ is empty;*

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

This result gives rise to the following algorithm SEARCH-SCHEDULE(J, F, $<<$, m) which provides, for any $I = (J, F, <<, m)$, either a feasible schedule of I or a k-schedule T which satisfies 1..5 above: $Hole(k)$, $Top(k)$, C^{Sup}_p , C^{Inf}_p denote here variables related with the current k-schedule T.

Algorithm SEARCH-SCHEDULE(J, F, $<<$, m):
T <- *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 *SEARCH-SCHEDULE* <- *Fail* Else
  T <- *Pre-Schedule*(J, F, m, $<<$, T); (**Turn T into a pre-schedule through proposition 3**)
  k <- Sup $_{l = 0..m}$ l such that T is a l-schedule; Not *Stop*;
  While $k < m$ and Not *Stop* do Search, according to this order, for a propagation path γ in:
  - $PP(Top(k) \cup \{C^{Inf}_1,..,C^{Inf}_{k+1}\} \cup \{C^{Sup}_1,..,C^{Sup}_{k+1}\}, Hole(k))$;
  - $PP(Top(k), \{C^{Sup}_1 +1,.., C^{Sup}_{k+1} +1\} \cup \{C^{Inf}_1 -1,.., C^{Inf}_{k+1} -1\})$;
  - $\cup_{j = 2..k+1} PP(\{C^{Sup}_j, C^{Inf}_j\}, \{C^{Inf}_{j-1} -1 , C^{Sup}_{j-1} +1\})$;

  If *Failure*(Search) (*γ does exist*) then *Stop*
  Else Let σ be a label of γ; T <- Trans(T, γ, σ);
  If $k = m$ then *SEARCH-SCHEDULE* <- T else *SEARCH-SCHEDULE* <- *Fail*;

Theorem 4. *SEARCH-SCHEDULE solves, in an exact way, the NON-IDLE$_0$ problem in polynomial time*.

Sketch of the proof: as for correctness, it derives from the proof of Theorem 2. As for complexity, one 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 the number of time the path search instruction for a given k, may be polynomialy bounded in n and k. This provides us with the key argument for the time-polynomiality of our algorithm.

4.2. *An Exact Polynomial Algorithm for the NON-IDLE$_1$ 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 $I = (J, F, <<, m)$
Output: a no idle feasible schedule or a *Failure* signal;
Initialize T through SEARCH-SCHEDULE;
If *Failure*(Initialize) then *Failure* Else
  Not *Stop*;
  While Not *Stop* do
    Δ <- *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-Aux <- SEARCH-SCHEDULE(J, F_Δ, $<<$, m);
    If *T-Aux* $\neq$ *Failure* then T <- *T-Aux* Else
      For any job $J_i \in J$, set $F_\Delta(i) = F(i) \cap \{ t_1,.., t_2-1\}$;
      T-Aux <- SEARCH-SCHEDULE(J, F_Δ, $<<$, m);
      If *T-Aux* $\neq$ *Failure* then T <- *T-Aux* Else *Stop*.
  Makespan-Min-No-Idle-Schedule <- T ;

Theorem 5: *The above Makespan-Min-No-Idle-Schedule algorithm solves, in an exact way, the Makespan 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 $ACT(T) = [a, b]$, 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

We just studied a variant of the m-machine non-idling problem: a structural feasibility criterion as well as polynomial algorithms have been provided. However, several questions about the complexity of more general problems with the same HNI constraints are still open: it would be interesting to get the complexity status of the variant of the problem which corresponds to the case when the non-idling constraint has only to be satisfied on each machine.

REFERENCES

Baptiste.P. (2005): "Scheduling unit tasks to minimize the number of idle periods"; *Research Report*, CNRS LIX Lab..

Chretienne.P (2008): "Single-machine scheduling without intermediate delays"; *Disc. App. Maths* 13-156, p 2543-2550.

Jouglet.A (2012): "Single-machine scheduling with no-idle time and release dates to minimize a regular criterion", *Journal of Scheduling* 15 (2), p 217-238.

Valente .J.M. Alves.R (2005): "An exact approach to early/tardy scheduling ", *Computers/OR* 32, p 2905-2917.

Landis.K (1983): "Group technology and cellular manufacturing in Westvaco", *Project Report in IOM* 581, School of Business, University of Southern California.

Irani.S, Pruhs.K (2005): "Algorithmic problems in power management", *ACM Press*, vol 36, p 63-76, New York, USA.