An hybrid CP/MILP method for scheduling with energy costs

Alain Haït[1]Christian Artigues[2,3]

[1] Université de Toulouse,
Institut Supérieur de l'Aéronautique et de l'Espace,
10 avenue E. Belin F31055 Toulouse, France
E-mail: alain.hait@isae.fr
5pt [1]CNRS; LAAS; 7, avenue du Colonel Roche,
F-31077 Toulouse, France
[2]Université de Toulouse; UPS, INSA, INP, ISAE; LAAS;
F-31077 Toulouse, France
E-mail: christian.artigues@laas.fr

Abstract

This paper deals with energy-related job scheduling for a foundry, in order to minimize the electricity bill. Accounting for energy and human resource constraints leads to better solutions in terms of cost and overall energy consumption. We propose a hybrid heuristic based on a two-step constraint/mathematical programming approach that improves significantly the computation time, compared to the full MILP model.

keywords: Scheduling, Energy, Human resources, Parallel machines, Hybrid approach.

1 Introduction

The problem of energy consumption in industry exists for several decades, but has become much more critical in the beginning of this century. The main reasons are demand increase with the industrialization of China and India, inducing higher prices, and environmental aspects (reliance on fossil energy, global warming). The reflexion about a better energy consumption is a real challenge now and for the future.

Consequently, energy consumption is a matter of concern in the industrial sector. Long range solutions are about alternative sources of energy and design of new types of industrial networks (Gibbs and Deutz, 2007). Waiting for these solutions to be effective, shorter-range responses can be found for industry. Integration of the energy aspects into production guarantees a better use of energy (Cheung and Hui, 2004). For example, the use of co generation, along with a coordination of product and energy operations is a promising issue in the process industry (Agha et al., 2008, 2009; Korhonen, 2002).

These new constraints force to reconsider production planning and scheduling models. From a scheduling point of view, accounting for energy, and more generally *utilities* or *secondary resources* involve changes due to the availability of these resources, their cost or their impact on the duration of the operations. In this paper, we present an example of scheduling for a foundry, with electricity costs, accounting for energy and human resource constraints. Next section presents informally the problem and the related work. Section 3 presents the mathematical model, and in section 4 we propose a constraint/mathematical programming heuristic to solve efficiently the problem. Section 5 shows some experimental results.

2 Problem statement

The addressed problem comes from a pipe-manufacturing plant, presented in Haït et al. (2007), and more precisely, the foundry where metal is melted in induction furnaces and then cast in individual billets.

The foundry has five lines of production (furnaces). Each melting operation has to be assigned to a furnace and scheduled within a time window. Furthermore, it has a variable duration that depends on the power given to the furnace (constrained by physical and operational considerations). A melting job is thus composed of three sequential parts (Fig. 1): loading, heating and unloading. Each furnace has unavailability periods representing operator temporary operator unavailability. While heating can be performed during an unavailability period, loading and unloading operations need the presence of an operator. The durations of loading and unloading are known (D_l and D_u), but heating duration depends on the following conditions:

- melting duration depends on the power given to the furnace, in a range $[P_{min}, P_{max}]$. Melting of job j ends when an amount E_j of energy has been supplied.
- when melting is complete, the temperature must be hold in the furnace until an operator is ready to unload it. To this aim, power P_{hold} is applied to the furnace.

The goal is to minimize the cost of the schedule, depending on the energy consumed and on penalties when the overall power in the foundry exceeds a given subscribed value. The energy consumption is recorded and the power is evaluated every 15' interval by the electricity provider, using a counter.

From a scheduling view-point, this facility can easily be recognized as a parallel machine problem. Parallel machine scheduling without energy considerations has been widely studied (Chen and Sin, 1990), specially because it appears as a

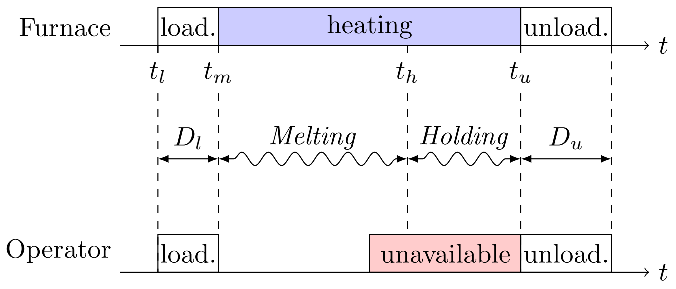

Figure 1: Job description and corresponding operator's tasks.

relaxation of more complex shop or project scheduling problems like the hybrid flow shop scheduling problem or the resource-constrained project scheduling problem. Several methods have been proposed to solve this problem. In Chen and Powell (1999), a column generation strategy is proposed. Pearn et al. (2007) propose a linear program and an efficient heuristic for solving large-size instances of a priority constraints and family setup times problem. Salem et al. (2000) solve the problem with a tree search method. Gacias et al. (2010) propose limited discrepancy-based local search methods for a parallel machine scheduling problem with precedence constraints and sequence-dependent setup times. Neron et al. (2008) compare two different branching schemes and several tree search strategies for the problem with release dates and tails for the makespan minimization case. In Baptiste et al. (2000), a constraint programming-based approach is proposed to solve the weighted number of late jobs problem. In Sadykov and Wosley (2006), An hybrid Integer/Constraint Programming approach is propose to solve a minimum-cost assignment problem. Among the variants presented in the latter, the most effective strategy is to combine a tight and compact, but approximate, mixed integer programming (MIP) formulation with a global constraint testing single machine feasibility.

Energy consumption management is a critical issue in computer systems, networks and embbeded systems where many (online) algorithmic problems are raised and well studied Irani and Pruhs (2005). On the other hand, these considerations are recent in production. Production scheduling for steel manufacturing has been studied, but few papers focus on energy cost (Nolde and Morari, 2010). Complexity is a major difficulty for the integration of energy constraints to production scheduling (Sec. 3.5) and the literature on the subject is rather sparse. This generally leads to develop heuristics. For example, Boukas et al. (1990) propose a hierarchical approach for scheduling a steel plant subject to a global limitation on the power supplied to the furnaces. Harjunkoski and Grossmann (2001) use a decomposition approach to solve a steel manufacturing scheduling problem with multiple products.

3 Mathematical programming model

In this section, we propose a continuous-time MILP model of the addressed problem. Then this model will be included in a two-level heuristic presented in the next section.

3.1 Scheduling model

Jobs are defined by the operation start times t_l (loading), t_m (melting, $t_m = t_l + D_l$), t_h (holding) and t_u (unloading). These times are continuous variables in the model. Job assignment and sequencing are represented by the following binary variables (the nomenclature is given in Appendix A):

- $x(f,j) = 1$ if furnace f is allocated to job j, 0 otherwise;
- $y(j_1,j_2) = 1$ if job j_1 precedes job j_2, 0 otherwise.

The assignment and sequencing constraints are:

$$\sum_{f\in F} x(f,j) = 1 \quad \forall j \in J \tag{1}$$

$$y(j_1,j_2) + y(j_2,j_1) \le 1 \quad \forall (j_1,j_2) \in J^2, j_1 \neq j_2 \tag{2}$$

$$y(j_1,j_2) + y(j_2,j_1) \ge x(f,j_1) + x(f,j_2) - 1 \qquad \forall f \in F, (j_1,j_2) \in J^2, j_1 \neq j_2 \tag{3}$$

Equation (1) indicates that exactly one furnace is allocated to a job, while (2) represents the antisymmetry of the precedence relation. Finally, (3) ensures that if the same furnace is allocated to two jobs, one precedes the other.

The operation start times depend on release and due dates, operation sequence and precedence constraints:

$$\begin{aligned}
t_l(j) &\ge Rel(j) && \forall j \in J && (4)\\
t_m(j) &= t_l(j) + D_l(j) && \forall j \in J && (5)\\
t_h(j) &\ge t_m(j) + E(j)/P_{\max} && \forall j \in J && (6)\\
t_h(j) &\le t_m(j) + E(j)/P_{\min} && \forall j \in J && (7)\\
t_u(j) &\ge t_h(j) && \forall j \in J && (8)\\
t_u(j) &\le Due(j) - D_u(j) && \forall j \in J && (9)\\
t_l(j_2) &\ge t_u(j_1) + D_u(j_1) - M(1-y(j_1,j_2)) && \forall (j_1,j_2) \in J^2, j_1 \neq j_2 && (10)
\end{aligned}$$

Constraints (5)-(8) set the sequence of operation start times for job j: first t_l, then t_m, t_h and finally t_u (Fig. 1). Note that (6) and (7) use the min/max durations of job j. Constraints (4) and (9) situate job j in the range [*Release date*; *Due date*] and job sequencing is given by (10) according to the precedence variables $y(j_1,j_2)$. The 'Big-M' value in (10) is set to the horizon length.

3.2 Heating operation logic

The time horizon is divided into $I_{\max}$ intervals of uniform duration D. These intervals will be used to measure the consumption of energy and then to estimate the instantaneous power as the mean power on each interval.

3.2.1 Time/interval binary variables.

During the melting of job j, an amount of energy $e_m(j,i)$ is supplied at an interval i. It is the integration of the power given to the furnace over the melting duration $d_m(j,i)$ in this interval. Our model uses energy and duration as variables, but it is not necessary to represent explicitly the power, considered as a constant over the melting duration for each interval (Fig. 2). During the holding phase (waiting for an operator to unload the furnace), the power is set to P_{hold}. The holding duration of job j at interval i is given by the variable $d_h(j,i)$. Melting and holding of a job may appear in the same interval when t_h belongs to this interval (e.g. interval 3 in Fig. 2).

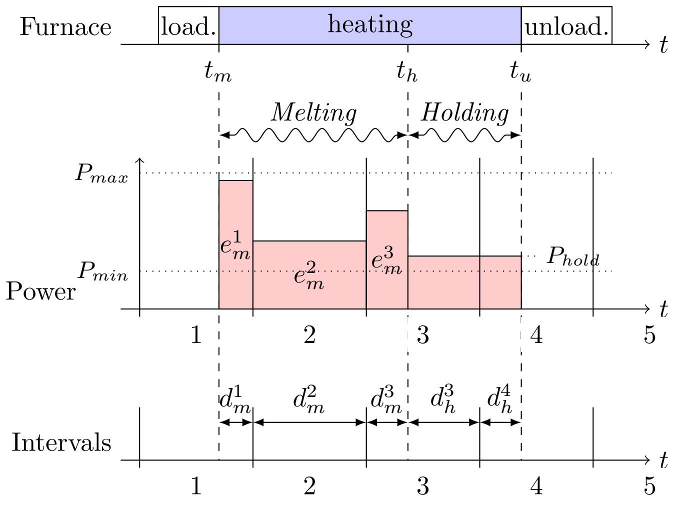

Figure 2: Energy supply by interval: melting and holding.

Durations $d_m(j,i)$ and $d_h(j,i)$ depend on operation start times $t_m(j)$, $t_h(j)$ and $t_u(j)$, and the intervals to which they belong. Binary variables are used to identify the intervals in which energy (for melting or holding) is supplied to the furnace for job j. Variables $z_m(j,i)$ indicate if melting start time of job j occurred before or during interval i. Similarly, $z_h(j,i)$ and $z_u(j,i)$ indicate if holding and unloading start times of job j occurred before interval i. For example, Figure 3 shows $z_m(j,i)$ and $z_h(j,i)$ for the same job as in Figure 2. The melting operation occurs in intervals for which $z_m(j,i) - z_h(j,i) = 1$.

The following equations give the relations between the melting start time

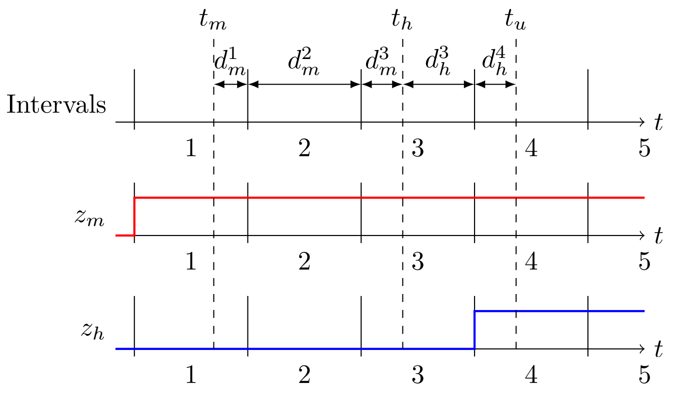

Figure 3: Time/interval binary variables.

and binary variable z_m for all job j in J, according to the above description. Constraints for z_h and z_u are similar.

$$t_m(j) \geq D.i(1 - z_m(j,i)) \qquad i = 1, \ldots, I_{\max} - 1 \tag{11}$$

$$t_m(j) \leq D.i + M(1 - z_m(j,i)) \qquad i = 1, \ldots, I_{\max} - 1 \tag{12}$$

$$z_m(j, i+1) \geq z_m(j,i) \qquad i = 1, \ldots, I_{\max} - 1 \tag{13}$$

$$z_m(j, I_{\max}) = 1 \tag{14}$$

The sequence of operations and the sequence of jobs induce additional constraints between time/interval variables. Equations (15) and (16) are based on the melting-holding-unloading sequence; (17) is related to precedence constraints.

$$z_m(j,i) \geq z_h(j, i+1) \quad \forall j \in J, i \in [1..I_{\max} - 1] \tag{15}$$

$$z_h(j,i) \geq z_u(j,i) \quad \forall j \in J, i \in I \tag{16}$$

$$z_u(j_1, i) \geq z_m(j_2, i) - (1 - y(j_1, j_2)) \qquad \forall (j_1, j_2) \in J^2, i \in I \tag{17}$$

3.2.2 Melting duration and energy.

The melting operation of a job, between t_m and t_h, is decomposed in intervals (Fig. 2). Melting duration $d_m(j,i)$ of any interval i lies in $[0, D]$. It is zero if $z_h(j,i) - z_m(j,i) = 0$ (Fig. 3, (18)). The sum of all interval durations is equal to the overall melting duration (19).

$$0 \leq d_m(j,i) \leq D(z_m(j,i) - z_h(j,i)) \quad \forall j \in J, i \in I \tag{18}$$

$$\sum_{i \in I} d_m(j,i) = t_h(j) - t_m(j) \quad \forall j \in J \tag{19}$$

Note that when $t_m(j)$ exactly matches the end of an interval i ($t_m(j) = D.i$), $z_m(j,i)$ can either take the value 0 or 1 ((11) and (12)). This is not a problem because in this case $d_m(j,i) = 0$ whatever the value of $z_m(j,i)$ (19).

The duration of melting $d_m(j,i)$ for an interval i depends on the position of the melting and holding start times with respect to this interval. Figure 4 summarizes the possible configurations of t_m and t_h over an interval i. The corresponding value of d_m is the intersection of interval i and $[t_m, t_h]$.

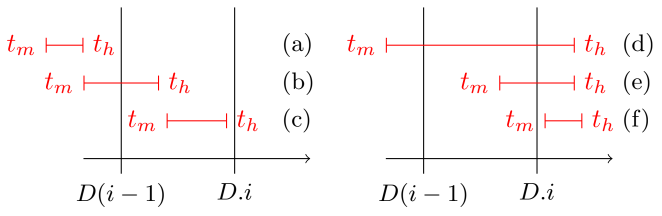

Figure 4: Intersections of a melting operation and an interval.

The following constraints set the value of $d_m(j,i)$ according to these configurations, for all i in $\{2, \ldots, I_{\max} - 1\}$ and j in J:

$$d_m(j,i) \geq D(z_m(j,i-1) - z_h(j,i+1)) \tag{20}$$

$$d_m(j,i) \geq D.i(1 - z_m(j,i-1)) - t_m(j) - D.z_h(j,i+1) \tag{21}$$

$$d_m(j,i) \geq t_h(j) - D.i + D.z_m(j,i-1) - M(1 - z_h(j,i+1)) \tag{22}$$

Constraints (20), (21) and (22) respectively match configurations (d), (e) and (b). The other cases are solved by (18) and (19).

For each interval, the amount of energy provided to a job depends on the melting duration and the supplied power. The melting ends when the required energy quantity $E(j)$ is reached.

$$P_{\min}.d_m(j,i) \leq e_m(j,i) \leq P_{\max}.d_m(j,i) \quad \forall j \in J, i \in I \tag{23}$$

$$\sum_{i \in I} e_m(j,i) = E(j) \quad \forall j \in J \tag{24}$$

3.3 Operator unavailability

Unavailability may occur when the number of furnaces exceeds the number of operators. In this case, at the end of the melting operation the material stays in the furnace and the temperature is hold, involving extra energy consumption at power P_{hold}, until an operator become available to unload the furnace. These human resource constraints are represented by unavailabilities, modelled as operations on a furnace. Operator unavailability o is defined by its start time $t_s(o)$, end time $t_e(o)$ and the binary parameter $X_o(o, f) = 1$ if unavailability o is on furnace f, 0 otherwise. An unavailability cannot overlap a loading or unloading operation on the same furnace, but can overlap a melting operation. Consequently, according to the position of loading and unloading, three configurations of a job j and an unavailability o are possible (Fig. 5).

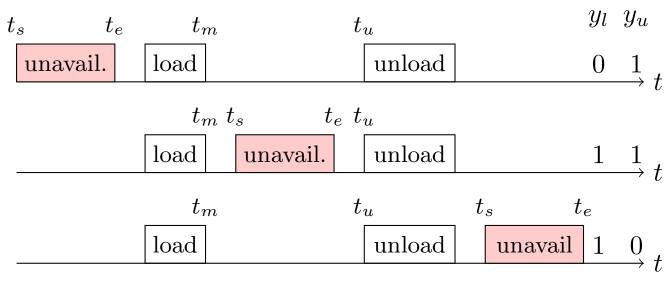

Figure 5: Relative positions job/unavailability.

Binary variables $y_l(j, o)$ and $y_u(j, o)$ represent the relative position of loading/unloading and unavailability (Fig. 5). At least one is equal to 1 if the unavailability and the job are on the same furnace:

$$y_l(j, o) + y_u(j, o) \geq X_o(o, f) + x(f, j) - 1 \tag{25}$$

The following set of constraints links operations start times with unavailability start and end times, for all j in J and o in O:

$$t_m(j) \leq t_s(o) + M(1 - y_l(j, o)) \tag{26}$$
$$t_m(j) - D_l(j) \geq t_e(o) - M(1 + y_l(j, o) - y_u(j, o)) \tag{27}$$
$$t_u(j) + D_u(j) \leq t_s(o) + M(1 - y_l(j, o) + y_u(j, o)) \tag{28}$$
$$t_u(j) \geq t_e(o) - M(1 - y_u(j, o)) \tag{29}$$

Constraints (26) to (29) give the relative position of $t_m(j)$, $t_u(j)$, $t_s(j)$ and $t_e(j)$ for the cases $y_l = 1$, $y_l = 0$ and $y_u = 1$, $y_u = 0$ and $y_l = 1$, and $y_u = 1$ respectively (Fig. 5).

3.4 Objective function

If there is no operator available to unload the furnace, holding is carried out with a power P_{hold} during d_h. Thus the overall energy for an interval is the sum of melting and holding energy on each furnace.

$$se(i) = \sum_{j \in J} (e_m(j,i) + P_{hold}.d_h(i,j)) \quad \forall i \in I \tag{30}$$

Value $se(i)$ is then used to evaluate the instantaneous power, as the mean power over interval i. Power overrun $ov(i)$ is noticed on an interval when this power exceeds the subscribed power P_s.

$$ov(i) \geq se(i)/D - P_s \quad \forall i \in I \tag{31}$$

$$ov(i) \geq 0 \quad \forall i \in I \tag{32}$$

The objective function (33) minimizes the cost of the energy consumption and the overrun cost:

$$\min \sum_{i \in I} (\alpha.se(i) + \beta.ov(i)) \tag{33}$$

Finally, given the hypothesis that $\sum_{i \in I} e_m(j,i) = E_j$ whatever the melting profile, the objective function can be written as the sum of the holding energy cost and the overrun cost:

$$\min \sum_{i \in I} (\alpha.P_{hold}.\sum_{j \in J} d_h(i,j) + \beta.ov(i)) \tag{34}$$

3.5 Problem complexity

The problem under consideration in this paper is strongly NP-hard. This can be shown by considering a special case where the number of machines is larger than the number of jobs, there are no machine unavailability, there are zero loading and unloading times, $D = 1$ and $P_{\min} = P_{\max}$. In this case answering the question whether no overrun is observed is NP-complete in the strong sense as we obtain the multiprocessor task problem $P|r_i, d_i; size_i|-$ (Drozdowski, 1996).

4 A heuristic hybrid method

The integer linear program proposed in the preceding section is only able to solve small problems. To deal with real-life problems, we propose a two-step heuristic method based on the decomposition of the problem into two decision steps (Fig. 6).

During the first step, sequencing of jobs on the furnaces is performed with fixed job durations. Since it may happen that no feasible solution exists considering the due dates, due date violation is admitted but the objective is to minimize the maximal tardiness. Hence the problem resorts to a parallel machine problem which machine availability, release dates and maximal tardiness criterion. A particularity of the problem is that unavailability concerns only the loading and unloading parts of the jobs.

During the second step, the job sequence is fixed on the furnaces according to the preceding step and the jobs are precisely scheduled while the power setting of each furnace during each interval determines the precise duration of each job. The objective function is the same as in Section 3.4 with an additional term to highly penalize due date violations.

For the first iteration, step 1 is performed by considering fixed melting durations of the jobs corresponding to the maximal power P_{max} given to the furnace. Then we close the loop by using at step 1 the durations given by step 2. The process is interrupted if the objective function of step 2 is not better than the one of the previous iteration, and if the tardiness is not improved.

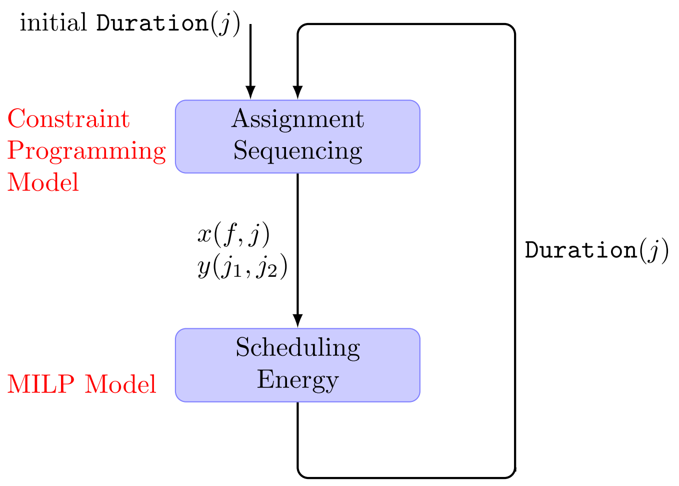

Figure 6: Hybrid approach.

Step 1 of the proposed heuristic corresponds to solving an almost standard parallel machine scheduling problem. Any specialized algorithm can be adapted to solve this problem. We propose a constraint programming approach to tackle this problem. Step 2 of the proposed method involves continuous start times and power intensity variables. Linear programming is a particularly well-suited technique to tackle the problem. In Section 4.1, we present the constraint programming framework. In Section 4.2, we present the link with the integer linear programming part.

4.1 Modeling and solving the furnace assignment and job sequencing problem

We propose to use a commercial constraint programming modeling language and solver (IBM ILOG OPL 6.3/CP Optimizer 2.3) to cope with the parallel machine problem corresponding to step 1. The OPL language provides high

```
// Tasks
dvar interval loading [j in J]
   in ReleaseDate[j]..maxint
   size LoadDuration[j];
dvar interval melting [j in J]
   in ReleaseDate[j]+LoadDuration[j]..maxint
   size Duration[j] ;
dvar interval unloading [j in J]
   in ReleaseDate[j]+LoadDuration[j]+Duration[j]..maxint
   size UnloadDuration[j];
dvar interval unavail [o in O]
   in unavailEarliestStart[o]..unavailLatestEnd[o]
   size UnavailDuration[o];

// Optional tasks: possible assignements of a furnace to a task
dvar interval alt_loading [f in F] [j in J]
   optional in ReleaseDate[j]..maxint
   size LoadDuration[j];
dvar interval alt_melting [f in F]  [j in J]
   optional in ReleaseDate[j]+LoadDuration[j]..maxint
   size Duration[j] ;
dvar interval alt_unloading [f in F] [j in J]
   optional in ReleaseDate[j]+LoadDuration[j]+Duration[j]..maxint
   size UnloadDuration[j];
```

Figure 7: Constraint programming: task declarations.

level primitives to model scheduling components. We provide the OPL code in Figures 7, 8 and 9. As a first category of high level scheduling components, task declaration is displayed in Figure 7. Job loading, melting and unloading, and furnace unavailabilities are defined as tasks (type `dvar interval`) specifying for each of them the time windows (field `in`) and the duration (field `size`). Furthermore optional tasks are associated to each loading, melting and unloading tasks to model the furnace assignment problem, so that there exists an optional task per loading, melting and unloading operation and candidate furnace. As indicated above, `Duration[j]` for the melting part of job j is initially set to minimal value $E(j)/P_{max}$.

In Figure 8, two sets of sequences (type `dvar sequence`) are declared as a second high level scheduling component category. The first set comprises a sequence declared per furnace f, gathering each optional loading, melting and unloading task assigned to f. The second set also contains a sequence of tasks per furnace f, gathering the optional loading and unloading operations assigned to f and the unavailability tasks assigned to f. It is used to manage the operator availability constraints.

Figure 9 now describes the constraints on the scheduling components and the objective function. First the objective is maximal tardiness minimization as the due dates are relaxed. Then, for each job the loading, melting and unloading tasks are constrained to be synchronized: melting operation of a job starts at the end of loading and unloading starts at the end of melting. The next two sets of constraints are about assignment: we define each loading, melting and unloading task as an alternative among all corresponding optional tasks. It follows that for a given loading, melting or unloading task, only one optional

```
// Sequence of jobs on furnace f
dvar sequence jobSequence[f in F] in
   append(all(j in J) alt_loading[f][j],
          all(j in J) alt_melting[f][j],
          all(j in J) alt_unloading[f][j]);

// Sequence of tasks requiring an operator on furnace f
dvar sequence unavailSequence[f in F] in
   append(all(j in J) alt_loading[f][j],
          all(j in J) alt_unloading[f][j],
          all(o in O: UnavailFurnace[o]==f)unavail[o]);
```

Figure 8: Constraint programming: sequence declarations.

```
minimize max(j in J) maxl(0,endOf(unloading[j])-DueDate[j]);

subject to{
  // Job tasks synchronization
  forall(j in J) {
    startAtEnd(melting[j],loading[t],0);
    startAtEnd(unloading[t],melting[t],0);
  }
  // Furnace assignment
  forall(j in J) {
    alternative(loading[j], all(f in F) alt_loading[f][j]);
    alternative(melting[j], all(f in F) alt_melting[f][j]);
    alternative(unload[j], all(f in F) alt_unloading[f][j]);
  }
  // All the tasks of a job on the same furnace
  forall(j in J,f in F) {
    presenceOf(alt_loading[f][j])==presenceOf(alt_melting[f][j]);
    presenceOf(alt_melting[f][j])==presenceOf(alt_unloading[f][j]);
  }
  // Furnaces viewed as disjunctive resources
  forall(f in F) {
    noOverlap(jobSequence[f]);
    noOverlap(unavailSequence[f]);
  }
}
```

Figure 9: Constraint programming: objective and constraints.

task will be activated. Then we state that the optional loading, melting and unloading tasks of a same job must be assigned to the same furnace. The last set of constraints describes how the two sets of sequences are handled. In this case, no overlapping of the tasks belonging to each sequence is allowed.

Once written in OPL, the parallel machine problem can be solved by the IBM ILOG CP Optimizer, a commercial constraint programming solver embedding precedence and resource constraint propagation techniques and an efficient self-adapting large neighborhood search method dedicated to scheduling problems (Laborie, 2009). A time limit is set and the best solution found within the time limit is returned. Note that modeling unavailabilities as tasks allows to define a time window for positioning each unavailability period.

4.2 Solving the scheduling and power setting problem with fixed furnace assignment and job sequences

In the second stage of the proposed heuristic, the MILP model presented in Section 3 (equations (4) to (34)) is used to set precise job start times and power supply while keeping the job assignments and sequences found at the previous stage. Due dates can be violated but tardiness is highly penalized in order to seek for a feasible final solution. Hence the heuristic does not stop if, for a given iteration, the MILP problem has no solution that respects the due dates. A time limit is set and the best solution found within the time limit is returned. If this solution has no tardiness, no overrun and no holding, the loop is broken because the optimum is reached.

4.3 Variant: initial melting durations for Step 1

The OPL modeling language gives the opportunity to define a job duration as a range between two values. Thus the melting `interval` variables can be defined as a range $[E_j/P_{max};\texttt{maxint}]$, letting the solver determine the adequate duration between the minimal duration and a value that represents the sum of maximal and holding durations. Using this feature at the first iteration may limit the number of iterations if task durations well reflect energy consumption. To this aim, a term is added to tardiness in the objective function of Step 1, that penalizes tasks whose duration is far from the maximal duration. Indeed, when duration is close to the minimum value, the furnace is set to a high power and it could lead to power overrun. On the other hand, if the duration is higher than the maximal duration, holding energy is spent. For the next iterations, the original hybrid approach is used, with the previous CP model at Step 1 relying on the durations given by Step 2.

5 Experimental results

Experimental results give an idea of the interest of the two-level heuristic. The following results are based on the pipe-manufacturing example: we consider 9-hour working days (from 8 am to 5 pm) with operator unavailability during lunch break (from 12:00 am to 1:30 pm) and during the morning and afternoon breaks (15' between 10:00 and 10:30 am, and 15' between 3:00 and 3:30 pm). The flexibility of morning and afternoon breaks allowed to reduce power overrun, inducing a significant cost reduction (Haït et al., 2007). Instances of 36 jobs on 6 furnaces are tested. Job energy requirements and release and due dates were randomly generated. Energy consumption is calculated on 15' periods. Energy cost is \$0,0242 /kWh, and the penalty for overrun is \$32 for each kW over the subscribed power.

All the tests have been performed on a SUN Sunfire server with four Quad-Core AMD Opteron(tm) 2.5 GHz Processors. Parallel CPLEX 12.1 is used to solve the MILP problems. Time limits are set for the hybrid approach thus bounding an iteration duration: 30 s for Step 1 (CP) and 180 s for Step 2 (MILP).

5.1 Solution steps on an illustrative instance

Table 1 shows the solution steps for an illustrative problem instance (data given in appendix B), with MILP model, hybrid model with fixed initial durations and variant hybrid model with variable initial durations. The tables give the objective value, the maximal tardiness, the sum of power overruns and of holding durations, and the computation time.

Table 1: Illustrative instance solved with MILP and Hybrid approaches

	Obj.	$T_{\max}$	$\Sigma ov(i)$	$\Sigma d_h(i,j)$	Time
MILP	650.5	0	0	53.8	1206.8

Hybrid	Obj.	$T_{\max}$	$\Sigma ov(i)$	$\Sigma d_h(i,j)$	Time
Step 1	-	30	-	-	0.11
Step 2	30067311	30	0	25.7	15.48
Step 1	-	30	-	-	0.11
Step 2	650.5	0	0	53.8	6.44
Step 1	-	0	-	-	0.09
Step 2	650.5	0	0	53.8	5.22

Variant	Obj.	$T_{\max}$	$\Sigma ov(i)$	$\Sigma d_h(i,j)$	Time
Step 1	-	0	-	-	30.0
Step 2	650.5	0	0	53.8	6.08
Step 1	-	0	-	-	0.11
Step 2	650.5	0	0	53.8	6.30

The MILP model is solved to optimality in more than 20 minutes. Compared to this solving time, the hybrid approach is very fast. At the first step, the method gives a solution with tardiness, due to the initial values. The assignment and sequencing variables are sent to Step 2, and a first solution is given. The result is big because of the huge penalty given to tardiness. At the second iteration, a solution with tardiness is found again by the CP solver at Step 1, but Step 2 gives then a solution with only a holding duration greater than 0. Note that it is the optimal solution. A third iteration is performed. As nothing is improved, the process ends. The overall solving duration is less than 30 seconds, and no iteration time limit has been reach.

Although the minimal durations are taken to initialize the process for the hybrid approach, the first step of the first iteration does not give a solution with no tardiness (it is only found at iteration 3). This is due to unavailability periods. Fixed melting durations do not necessary fit with these periods so that loading would be performed before an unavailability period and unloading after it. The variant approach attempts to solve this problem by improving the solution quality at the first step or the first iteration. The results are given in the last part of Table 1. The solution with no tardiness is found at the first iteration. Only two iterations are necessary to find the best solution. However, the first step of the first iteration lasts 30 seconds (the search is interrupted by the time limit), and the total duration is higher than the hybrid approach duration. Hence the variant should be interesting if the MILP solving time become longer and if the quality of the final solution is closely linked to the quality of the first iteration solution.

5.2 Method comparison on randomly generated problem instances

A set of 100 problem instances[1] with 36 jobs and 6 furnaces were generated according to the parameters described above, inspired by the industrial case study. Among these, 47 were found feasible by the MILP solver and solved to optimality while the others were proven unfeasible. In this section, we compare the MILP, hybrid and variant models on these instances.

Table 2 summarizes the results of MILP, hybrid and variant approaches for the 47 feasible instances. MILP solving time stays high so that using this model would be difficult in a situation with hundred of jobs. Some instance have overrun or holding durations in their optimal solution.

Table 2: Comparison of the approaches: mean values on 47 feasible instances

	$T_{\max}$	$\Sigma ov(i)$	$\Sigma d_h(i,j)$	Time	Iter.	Optim.
MILP	0	38.2	4.0	5397	-	100%
Hybrid	0.13	38.2	4.6	8.7	1.1	97.8%
Variant	0.13	40.3	4.2	42.8	1.4	91.5%

The hybrid approach is very fast, with a mean solving time less than 10 seconds. Only one instance has not been solved to optimality. Most of the instances have been solved in one iteration. Compared to this excellent result, the variant is less interesting: four instances have not been solved to optimality and the solving time is higher. Note that for some instances, tardiness is greater than 0, so the solution found is not feasible.

5.3 Comparison on bigger instances

In this section we present how processing time increases with the size of the instances. MILP solving time grows extremely rapidly with the number of jobs,

[1]available upon request

furnaces and intervals, so we do not compare these results with optimal solutions. The results are based on the same one-day shift instances with operator unavailabilities, inspired from the industrial case. Mean values on 50 randomly-generated instances are given.

At first we present in Table 3 the results of hybrid approach for the same problem as previously, but with smaller intervals so that the number of intervals increases as the horizon remains the same. The increase of intervals affects Step 2 (MILP model). Objective values cannot be compared because power overrun are not calculated on the same basis (interval durations). Solving time grows with the number of intervals. This is mainly due to MILP solving at Step 2. In the case of 90 intervals, time limit has been change in order to give the solver enough time to find solutions.

Table 3: Hybrid approach with growing number of intervals

	36	54	72	90
Time	18.6	96.9	336.7	983.1
Iterations	1.94	1.90	2.18	2.81

Then we propose to solve globally two or three consecutive shifts, so the number of tasks will be respectively 72 and 108 instead of 36. The number of furnaces does not change. Table 4 presents the mean solving time and iteration number for 50 instances (feasible or not). The number of iterations does not increase rapidly, whereas the solving time does. Again, this is mainly due to MILP solving. For the 3-shift instances, the 180-second time limit is often reached at Step 2.

Table 4: Hybrid approach with growing number of tasks

	1 shift	2 shifts	3 shifts
Time	18.6	119.5	435.9
Iterations	1.94	2.45	2.63

When the size of the problem increases, the MILP solving at Step 2 of the hybrid approach may take some time, even if it is much better than the full MILP solving. This is due to symmetries in the model. The solution is given in a reduced number of iterations, because the loop is broken as soon as the solution is not improved. Consequently, the increase of the problem size may affect the quality of the solution if the MILP solver does not have enough time to give a good solution. The first action would be to increase the MILP time limit, as done in Table 3 for 90-interval instances. This also claims for an improved variant approach in order to find a better initial solution. CP models that account for energy should be more adapted to the problem (Artigues et al., 2009).

6 Conclusion-Future work

In this paper we presented a mathematical/constraint programming heuristic approach to solve a foundry scheduling problem with energy and human resource constraints. The implemented methods are based on an iterative succession of a sequencing and assignment on parallel machines of jobs with fixed or variable durations via CP and a power adjustment and precise scheduling phase via MILP. The results are satisfactory since the proposed hybrid methods reach near-optimal solutions in reasonable CPU times on the computational experiments we carried-out using realistic data. In particular, the CPU times are drastically reduced compared to the full MILP model. Additional tests will be performed on other instances and other problem variants, like configurations with a number of operators lower than the number of furnaces. Future work will also focus on improving the method. The improvement of the first step may help to get good solutions rapidly. To this aim, this step could take into account explicitly energy constraints.

References

M. Agha, R. Thery, G. Hetreux, A. Haït, and J.-M. Le Lann. Integrated production and utility system approach for optimizing industrial unit operations. *Energy*, 35(2):611–627, 2010.

M. Agha, R. Thery, G. Hetreux, and J.-M. Le Lann. Modèle intégré pour l'ordonnancement d'ateliers batch et la planification de centrales de cogénération. *Conférence Internationale de Modélisation et Simulation, MOSIM08, Paris, France*, 2008.

C. Artigues, P. Lopez, and A. Haït. Scheduling under enregy constraints. *Proceedings of the International Conference on Industrial Engineering and Systems Management (IESM'09)*, 2009.

Ph. Baptiste, A. Jouglet, C. Le Pape, and W. Nuijten. A constraint-based approach to minimize the weighted number of late jobs on parallel machines, 2000. UTC Technical Report 2000/288.

E.-K. Boukas, A. Haurie, and F. Soumis. Hierarchical approach to steel production scheduling under a global energy constraint. *Annals of Operations Research*, 26:289–311, 1990.

Z.-L. Chen and W. B. Powell. Solving parallel machine scheduling problems by column generation. *INFORMS J. on Computing*, 11(1):78–94, 1999.

T. Cheng and C. Sin. A state-of-the-art review of parallel-machine scheduling research. *European Journal of Operational Research*, 47:271–292, 1990.

K.-Y. Cheung and C.-W. Hui. Total-site scheduling for better energy utilization. *Journal of Cleaner Production*, 12:171–184, 2004.

Maciej Drozdowski. Scheduling multiprocessor tasks – an overview. *European Journal of Operational Research*, 94:215–230, 2004.

B. Gacias, C. artigues and P. Lopez. Parallel machine scheduling with precedence constraints and setup times. *Computers and Operations Research*, doi:10.1016/j.cor.2010.03.003, 2010.

D. Gibbs and P. Deutz. Reflexion on implementing industrial ecology through eco-industrial park development. *Journal of Cleaner Production*, 15(17):1683–1695, 2007.

A. Haït, C. Artigues, M. Trepanier, and P. Baptiste. Ordonnancement sous contraintes d'énergie et de ressources humaines. *11e congrès de la Société Française de Génie des Procédés, Saint-Etienne, France, Récents progrès en Génie des Procédés*, 96, 2007.

I. Harjunkoski and I. Grossmann. A decomopsition approach for the scheduling of a steel plant production *Computers and Chemical engineering*, 25:1647–1660, 2001.

S. Irani and K. Pruhs. Algorithmic problems in power management. *SIGACT News*, 36(2):63–76, 2005.

J. Korhonen. A material and energy flow for co-production of heat and power. *Journal of Cleaner Production*, 10:537–544, 2002.

P. Laborie. IBM ILOG CP optimizer for detailed scheduling illustrated on three problems. *Proceedings of the 6th International Conference, on Integration of AI and OR Techniques in Con- straint Programming for Combinatorial Optimization Problems (CP-AI-OR'09)*, LNCS 5547: 148–162, 2009.

E. Néron, F. Tercinet, and F. Sourd. Search tree based approches for parallel machine scheduling. *Computers and Operations Research*, 35(4):1127–1137, 2008.

K. Nolde and M Morari. Electrical load tracking scheduling of a steel plant *Computers and Operations Research*, to appear, 2010.

W. L. Pearn, S. H. Chung, and C .M. Lai. Scheduling integrated circuit assembly operations on die bonder. *IEEE Transactions on electronics packaging manufacturing*, 30(2), 2007.

Ruslan Sadykov and Laurence A. Wolsey. Integer programming and constraint programming in solving a multimachine assignment scheduling problem with deadlines and release dates. *INFORMS Journal on Computing*, 18(2):209–217, 2006.

A. Salem, G. C. Anagnostopoulos, and G. Rabadi. A branch-and-bound algorithm for parallel machine scheduling problems. In *Harbour, Maritime & Multimodal Logistics Modeling and Simulation Workshop, Society for Computer Simulation International (SCS)*, Portofino (Italy), 2000.

A Notation

A.1 Parameters

Indices	
j	Job (J set of jobs)
f	Furnace (F set of furnaces)
o	Unavailability (O set of unavailabilities)
i	Interval (I set of intervals)
Power parameters	
$P_{\min}$	Minimal furnace power
$P_{\max}$	Maximal furnace power
P_{hold}	Holding power
P_s	Subscribed power
Job parameters	
$D_l(j)$	Loading duration of job j
$D_u(j)$	Unloading duration of job j
$E(j)$	Energy required for job j
$Rel(j)$	Release date of job j
$Due(j)$	Due date of job j
Interval parameters	
$I_{\max}$	Number of intervals
D	Duration of an interval
Unavailability parameters	
$X_o(o, f)$	Allocation of unavailability o to furnace f

A.2 Continuous variables

Energy-related variables	
$e_m(j,i)$	Melting energy for job j during interval i
$d_m(j,i)$	Melting duration for job j during interval i
$d_h(j,i)$	Holding duration for job j during interval i
$se(i)$	Total energy consumption during interval i
$ov(i)$	Power overrun during interval i
Operation-related variables	
$t_l(j)$	Loading start time of job j
$t_m(j)$	Melting start time of job j
$t_h(j)$	Holding start time of job j
$t_u(j)$	Unloading start time of job j
$t_s(o)$	Unavailability o start time
$t_e(o)$	Unavailability o end time

A.3 Binary variables

Allocation variables	
$x(f,j)$	Allocation of furnace f to job j
Precedence variables	
$y(j_1,j_2)$	Precedence between jobs j_1 and j_2
$y_l(j,o)$	Precedence between loading operation of job j and unavailability o
$y_u(j,o)$	Precedence between unloading operation of job j and unavailability o
Relative time/interval position variables	
$z_m(j,i)$	Melting start time $t_m(j)$ w.r.t. interval i
$z_h(j,i)$	Holding start time $t_h(j)$ w.r.t. interval i
$z_u(j,i)$	Unloading start time $t_u(j)$ w.r.t. interval i

B Instance data

```
NbJobs=36;
NbFurnaces=6;
NbUnavail=18;
Imax=36;
D=25;
Pmin=500;
Pmax=1200;
Phold=500;
Ps=3000;
F1=0.0242;
F2=32;
Dl=[22 14 24 23 19 11 23 18 9 12 23 2 11 7 0 0 4 15
 11 0 10 3 7 1 23 15 0 9 13 20 0 7 24 23 12 10 ];
Du=[23 13 22 25 5 8 21 17 23 16 22 13 3 20 8 14 18 25
 10 17 10 14 4 24 22 5 20 14 23 4 16 18 18 24 17 20 ];
```

```
E=[74831 50285 62357 61938 36862 32296 58587 31681 52931
 18904 55966 39281 23118 40268 69629 31891 43927 21161
 48091 67305 35053 63856 72285 61409 25944 45552 23373
 60472 49108 57937 43054 48575 21695 38225 44270 72462 ];
Rel=[83 395 695 299 222 315 530 112 141 575 409 835 370 84 792 334 182 780
  4 537 90 75 453 349 364 689 715 483 318 42 75 802 517 194 102 441 ];
Due=[755 576 859 500 571 787 749 277 263 850 825 896 503 280 895 743 514 876
 793 683 665 548 612 812 581 772 871 737 748 145 266 885 723 886 736 738 ];
Xo=[
[1 0 0 0 0 0 ]
[1 0 0 0 0 0 ]
[1 0 0 0 0 0 ]
[0 1 0 0 0 0 ]
[0 1 0 0 0 0 ]
[0 1 0 0 0 0 ]
[0 0 1 0 0 0 ]
[0 0 1 0 0 0 ]
[0 0 1 0 0 0 ]
[0 0 0 1 0 0 ]
[0 0 0 1 0 0 ]
[0 0 0 1 0 0 ]
[0 0 0 0 1 0 ]
[0 0 0 0 1 0 ]
[0 0 0 0 1 0 ]
[0 0 0 0 0 1 ]
[0 0 0 0 0 1 ]
[0 0 0 0 0 1 ]
];
So=[400 150 650 400 150 650 400 150 650 400 150 650 400 150 650 400 150 650 ];
Eo=[500 250 750 500 250 750 500 250 750 500 250 750 500 250 750 500 250 750 ];
Do=[100 25 25 100 25 25 100 25 25 100 25 25 100 25 25 100 25 25 ];
```