RAIRO. RECHERCHE OPÉRATIONNELLE

P. RICHARD

C. PROUST

Solving scheduling problems using Petri nets and constraint logic programming

RAIRO. Recherche opérationnelle, tome 32, n° 2 (1998), p. 125-143

<http://www.numdam.org/item?id=RO_1998__32_2_125_0>

Recherche opérationnelle/Operations Research
(vol. 32, n° 2, 1998, p. 125 à 143)

SOLVING SCHEDULING PROBLEMS USING PETRI NETS AND CONSTRAINT LOGIC PROGRAMMING (*)

by P. RICHARD and C. PROUST

Communicated by Philippe CHRETIENNE

Abstract. – *This paper presents an approach to solve scheduling problems from a Petri net model. A timed Petri net describes feasible sequences and schedules of operations. The net is then translated into a CHIP program. Build-in solver of the constraint logic programming language is used to solve the associated scheduling problem. The implementation of the OPTNET software and some results are described.* © Elsevier, Paris

Keywords: Scheduling, Petri nets, Constraint Logic Programming.

Résumé. – *Cet article présente une approche pour résoudre des problèmes d'ordonnancement depuis un modèle réseau de Petri. Un réseau de Petri temporisé modélise les séquences et les ordonnancements réalisables d'opérations. Le réseau est ensuite transcrit en un programme CHIP. Le solveur intégré du langage de programmation logique avec contrainte est utilisé pour résoudre le problème d'ordonnancement associé. L'implémentation du logiciel OPTNET et quelques résultats sont décrits.*© Elsevier, Paris

Mots clés : Ordonnancement, réseaux de Petri, Programmation Logique avec Contrainte.

1. INTRODUCTION TO SCHEDULING PROBLEMS

Carlier *et al.* (1987) say that we deal with a scheduling problem when "we must program the execution of a realisation by assigning resources to tasks and by fixing their execution dates [...]. The tasks are the common denominator of scheduling problems, their definitions are neither always immediate, nor trivial". These problems are hard to solve, most of them are NP-hard (Lawler *et al.*, 1989).

Traditional approaches is made through a mathematical classification of problems as Baker (1974), GOThA (1993). The notation of problems is: n/m/A/B, where n is the number of jobs, m is the number of machines, A

(*) Received October 1995.

Laboratoire d'Informatique, École d'Ingénieurs en Informatique pour l'Industrie, Université de Tours, 64, avenue J. Portalis, Technopôle BP 4, F. 37913, Tours, France.
e-mail:richardp@univ-tours.fr.

is the flow of pieces and internal rules of processing, and B is the measure of performance. Of course, we can model analytically those problems in a classical operational research way (*see* for instance, for the flowshop problem, Stafford *et al.* (1990)) to use standard resolution tools. But, on one hand, that does not necessarily allow to use the properties of the studied problem to face its resolution complexity (a specific branch and bound approach allows it), and on the other hand, the initial model can't be easily extended to take new constraints into account. The efficiency is then synonym of a lack of genericity even if we actually see some attemps to remedy it, as Foure *et al.* (1993). Classical methods used to solve these problems are enumerative methods (branch and bound, dynamic programming...), and neighborhood methods (tabu search, simulated annealing, genetic algorithms...).

But previous approaches failed to bridge the gap between the theory and practice of scheduling (Bauer *et al.,* 1991). In consequence, other approaches have been made: simulation techniques, artificial intelligence methods... The readers can refer to Bauer *et al.* (1991) and the special number of IJPR (1988) for a review of these techniques. More recently, the development of constraint programming seems to be interesting in the modelling of scheduling problems (Le Pape, 1994a). Constraints are added to a programming language through an extension of the language as CHIP (Van Hentenryck, 1989) or Prolog III (Colmeraurer, 1990), or as a library, as the Ilog Solver C++ library (Le Pape, 1994b). In order to deal with scheduling problems, CHIP and the Ilog library introduce symbolic constraints to represent resources, Aggoun *et al.* (1993) with CHIP and Le Pape (1994b) with Ilog schedule. The modelling power of these kind of tools is very important, but the resolution abilities are not well known. But those approaches reduce the gap between theory and practice.

All approaches reviewed before are bottum-up, in the sense that a resolution tool is applied to a scheduling problem. Another category of approach is to determine the set of feasible solutions of the scheduling problem without any optimization concern (Baptiste *et al.* (1991), Roubellat *et al.* (1994), Levy *et al.* (1994)). For example in Baptiste *et al.* (1994), feasible sequences are modeled with a P-Q-R tree, and then exploited with an extension of a CLP language designed by Zidoum *et al.* (1994), but without any optimization objective.

Our approach to solve scheduling problem is top-down. The first step is the modelling of the scheduling problem with Petri nets (Murata, 1989). The modelling is independant from any resolution tool. In a second step, among emerging resolution tools, we gainst interest with CLP.

First, we recall the basis of Petri nets. In second, we present relation between Petri nets and the scheduling theory. The third part concerns the modelling of scheduling problems with Petri nets. Without lost of generality, we take our examples in the classical flow-shop problem (Johnson, 1954) and its extensions (Proust, 1992). After deduction of the properties of the model, we describe the transcription of the obtained net into a CLP program in CHIP, and the resolution step. Then we present the software implementation and some results.

2. PETRI NETS RECALLED

A Petri net (PN) is a bipartite graph in which the vertices are places and transitions. Weighted edges link places and transitions. More formally, a Petri net is a 4-tuple <P,T,In,Out> where:

P a finite set of places ($|P| = m$)

T a finite set of transitions $|T| = n$)

In an input function $P \times T \rightarrow N$

$\text{In}(p,t) = 0$ if p does not precede t else the weight of the edge

Out an output function $P \times T \rightarrow N$

$\text{Out}(p,t) = 0$ if p is not behind t else the weight of the edge.

The dynamical behaviour of a PN is made by the flow of marks (or tokens) in its places. The marking function is $M : P \rightarrow N.M(p)$ is the number of marks in the place p.

A transition t is enabled if and only if every input place of t has a number of marks greater or equal to the weight of the corresponding edge. Firing a sequence of transitions follows the state equation below: $M' = M + C\sigma$

C is the incidence function: C =Out-In

σ is the firing vector of the (firing) sequence. It is an n vector in which a k in i^{th} position indicates that transition i is fired k times.

We note $\{\bullet p\}$ (respectively $\{p\bullet\}$) the set of input (respectively output) transitions of the place p.

An extension of the basic model adds a firing time (d_t) to each transition (t) (called Timed Petri Nets: TPN). A transition is enabled if places are marked at time u. The firing of a transition is decomposed in two phases: instantaneous withdrawal of tokens in input places of t at time u; instantaneous addition of tokens in output places of t at date $u + d_t$. Below, we deal with a particular class of TPN: timed marked graphs (TMG). Adding time to transitions of

a PN was made by Ramchandani (1974). In the following, we assume that no transition is firing at the date $t = 0$.

DEFINITION: A timed marked graph is a 3-tuple $G =< R, d, M^0 >$ in which:

R is a marked graph, *i.e.* every place has exactly one input and one output $|p_\bullet| = |{}_\bullet p| = 1$ and all weights are equal to one.

d is the time mapping $d : T \rightarrow N^*$

M^0 is the initial marking. We note p_{ij} the place between the transition t_i and t_j, and M_{ij} the initial marking of p_{ij}.

The firing time of a transition is no longer atomic in this model, but we assume that no transition can have more than one fire in progress (it is usually ensure by associating places, in input and output of each transition, marked by one token). The parallelism of transition firing is effective. The marking at date u of the PN is not sufficient to describe the state of the net. It is necessary to know the date of firing of transitions in progress (the residual firing times). Thus, the notion of firing sequence is not sufficient to determine the net evolution for a span of time. For that, Chrétienne (1983) introduces the notions of controlled execution and state. Before defining the notion of state, we must define the residual firing time of a transition $R_t(u)$ where x_t^n is the date of the n-th fire of transition t:

$$\begin{aligned} R_t(u) &= 0 && \text{if } 0 \leq u \leq x_t^1; \\ R_t(u) &= x_t^n + d_t - u && \text{if } x_t^n < u < x_t^n + d_t; \\ R_t(u) &= 0 && \text{if } x_t^n + d_t \leq u \leq x_t^{n+1}. \end{aligned}$$

DEFINITION: a state is a couple $(M(u), R(u))$ in which

$$\begin{aligned} M(u) &= (M_p(u))_{p \in P} && \text{the marking vector at date } u \\ R(u) &= (R_t(u))_{t \in T} && \text{the residual time vector at date } u. \end{aligned}$$

DEFINITION: a controlled execution is the step function of the firing dates series.

The characteristic vector $N = (N_t)_{t \in T}$ of a controlled execution stores the number of initiations at date u. A controlled execution is said feasible if the behaviour of the net insures that the marking remains nonnegative at any date u. A controlled execution can be represented by a Gantt diagram (*fig.* 1).

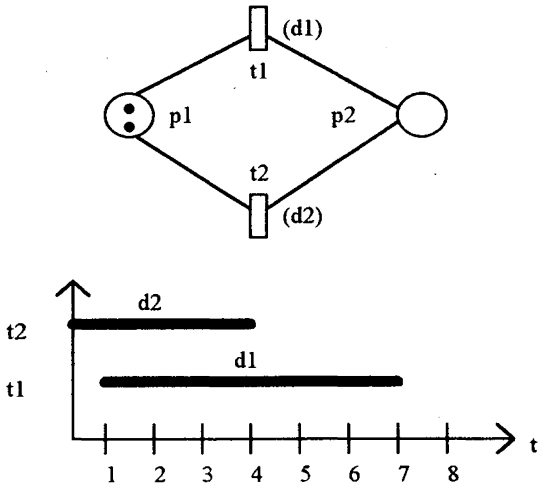


Figure 1. – A TPN and a feasible controlled execution.

Chrétienne (1983) has shown that constraints on firing dates in a TMG are potentials constraints:

THEOREM 1: *A controlled execution of a TMG is feasible iff the firing dates follow the inequalities*

$$\forall t_i,\ N_{t_i} > 0,\ x^1_{t_i} \geq 0,\ \forall p_{ij} \in P,\ N_{t_j} \geq 0,$$
$$\forall n \left\{ \begin{array}{c} n \geq 1 \text{ et} \\ n + M_{ij} \leq N_{t_j} \end{array} \right\} \Rightarrow \left\{ \begin{array}{c} n \leq N_{t_i} \\ x_j^{n+M_{ij}} \geq x_i^n + d_i \end{array} \right\} \quad (1)$$

This result is easy to understand. Let us consider an elementary part of a marked graph (a place p_{ij} with its two transitions (t_i, t_j), its form follows necessary $|p_\bullet| = |{}_\bullet p| = 1$. t_j can be simultaneously fired M_{ij} times, at date $t = 0$, unless t_i was fired once. Afterwards, every firing of t_j must be preceded by a firing of t_i, to bring a mark in p_{ij} (enabling t_j).

3. SCHEDULING AND PETRI NETS

Petri nets (PN) allows the modelling of scheduling problems constraints with a homogenous graphical formalism. They define central problems of the scheduling theory. For instance, TMG generalize the PERT/CPM methods

to cyclic problems (DiCesare *et al.* (1993). Usually, the places (P) represent stocks and resources; the transitions (T) represent operations; the tokens (or marks) are tasks and amount of resources.

Petri nets have been used to deal with cyclic scheduling. For instance Ramchandani (1974), Chrétienne (1983), Hillion *et al.* (1989), Munier (1993), Julia *et al.* (1995). They have been also used to study acyclic scheduling, for instance Chu (1993), Lee (1994), Cheng *et al.* (1994), Proth (1994), Richard *et al.* (1994). The optimized performance criterion is not modelized with the TPN, as it is the case with other models (*i.e.* mathematical programming). The total completion time (Cmax) is defined by ending time of firing transition: $Cmax = \max_{t \in T} (x_t^{N_t} + d_t)$ where $N = (N_t)$ is the characteristic vector (the fire number of transitions) to consider for the scheduling period. A scheduling criterion is regular, in scheduling theory, if it is non-decreasing in function of the tasks ending dates. $Cmax$ is obviously regular since a controlled execution is a non-decreasing step function.

In order to deal with due dates, an integer p_t can be associated to each transition, as Richard *et al.* (1995a). Thus, the following criteria can be defined: lateness ($Lmax$), $Lmax = \max_{t \in T} (\max_{n=1 \ldots n_t} (x_t^n + p_t - n \,.\, d_t))$, and tardiness ($Tmax$), $Tmax = \max_{t \in T} (\max_{n=1 \ldots n_t} (x_t^n + p_t - n \,.\, d_t), 0)$. These performance criteria are very used in industrial problems (Baker, 1974, and French, 1982).

Carlier *et al.* (1988) study the scheduling of firing sequences of TPN. In that paper, the parallel is made between the notions of tasks sequences and firing sequences on one hand, and the notions of schedules and controlled executions on other hand. Classically, in scheduling theory, the study of performance criterion allows to limit the search space of feasible solutions. For instance, for any regular criterion, it is only necessary to consider the schedules for which the operations execute as early as possible on each machine (semi-active schedules). That allows to associate one schedule to each sequence. Carlier *et al.* (1988) say that the same reasoning is applicable for TPN when the criterion studied is regular. The set of feasible solutions is then the set of earliest controlled executions associated to each feasible firing sequence.

The generalization of scheduling algorithms is obtained by showing equivalence between performance criteria. For instance, minimizing $Cmax$ is equivalent to maximize the average number of processing machines ($\overline{N}_p$) during the scheduling period. We show the same result with TPN concepts.

PROPERTY 1: *Following performance criteria are equivalent:*

$$\text{(i) } \min Cmax\,; \quad \text{(ii) } \max \overline{R}_t; \text{(iii) } \max \overline{N}_p$$

where

$$\overline{R}_t = \frac{1}{Cmax} \int_0^{C\max} R_t(u)\,du,\; N_p(u) = \sum_{t \in T} \delta_t(u)$$

and $\delta_t(u) = 1$ *if* $R_t(u) > 0$ *and 0 otherwise.*

Proof: (a) $\min Cmax \Leftrightarrow \max \overline{R}_t$, $\forall t\, \overline{R}_t = \frac{1}{Cmax} \int_0^{Cmax} R_t(u)\,du = \frac{1}{Cmax} \sum_{i=1}^{N_t} \int_{x_t^i}^{x_t^i + d_t} (x_t^i + d_t - u)\,du = \frac{N_t\, d_t^2}{2\, Cmax}$.

(b) $\min Cmax \Leftrightarrow \max \overline{N}_p$, $\overline{N}_p = \frac{1}{Cmax} \sum_{t \in T} \int_0^{Cmax} \delta_t(u)\,du = \frac{1}{Cmax} \sum_{t \in T} N_t\, d_t$.

Those results allow us to think that PN constitutes a homogenous model to study scheduling problems, both on modelling and theoretical point of view.

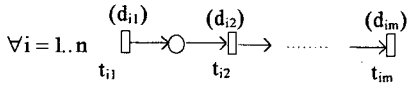
4. MODELLING OF SCHEDULING PROBLEMS – THE FLOWSHOP EXAMPLE

The modelling of scheduling problems with TPN consists on a design of a net where only the feasible schedules are reachable (feasible controlled executions). Then, come two choices: using high level PN or elementary PN. The former gives abstractions that are usually obtained with a loss of properties (DiCesare *et al.* 1993). For example Valentin *et al.* (1994) uses a high level Petri net to study the job-shop problems. Thus, the use of elementary PN is often prefered. But the model is too large to be wholly designed without a method. We have decided to model each constraint and extension of the basic flowshop problem (Proust, 1992) by different nets. The global net is obtained by a synthesis of all the set of designed nets. The constraints modelling of the flowshop problem family is given below. Note that, since for all regular performance criterion, it is only necessary to consider earliest controlled executions, we can add transitions with nought delay to model particular events (starting or ending of operations). The modelling of constraints is given Figure 2.

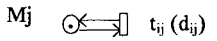
Two synthesis techniques of PN have been proposed: the bottum-up and the top down techniques. The first technique merges the two parts of independant nets into one. The second technique makes a decomposition

(a) A task i viewed as a single operation outside the shop

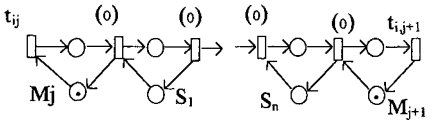
(b) A task is a sequence of operations. Places symbolize stocks between machines



(c) Machines are mutual exclusion places.



(d) In a permutation problem, the tasks pass in the same order on every machines. (stocks between machines are fifo managed). $M_1..M_m$, $S_1..S_n$ are shared places.

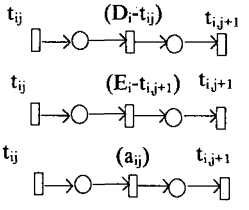


(e) Delays between operations.

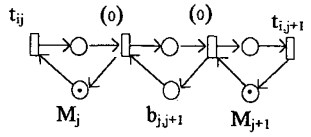
(e1) Start lag Di, the start of $t_{i,j+1}$ can't occur before the starting time $t_{ij}+D_i$. (we assume $D_i \geq d_i$)

(e2) Stop lag E_i : the end of $t_{i,j+1}$ can't happen before the end of $t_{ij}+E_i$ (we assume $E_i \geq d_j$).

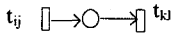
(e3) transport time a_{ij} between the machine j and j+1 for task i.



(f) Limited stock capacity between machines j and j+1 ($b_{j,j+1}$). The places $b_{j,j+1}$ are initialized with $b_{j,j+1}$ tokens. Mj and M_{j+1} are the machine constraints .



(g) General precedence constraint. We insert a place between the two transitions (operations) in precedence relation.



(h) Setup s_{ij} and remove time r_{ij} of tools which are not sequence dependent. (Snsd,Rnsd)

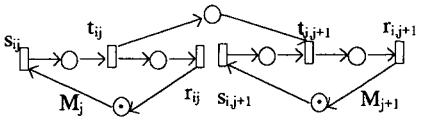


Figure 2. – The modelling of shop scheduling constraints.

(stepwise refinements) of places and/or transitions in subnets. Those principal techniques are summarized by DiCesare *et al.* (1993).

In presence of highly shared resources, modelling with one of those approaches is impossible. A combination of those two techniques must be used (hybrid approach), but then the systematic feature of the modelling

requires a synthesis procedure. It expresses the order and the use of synthesis rules. We give below the rules which we are going to use (*fig.* 3). The origin of the rule is indicated between brackets without mentioning the exact references. In the following N1 N2 represents the merge of the nets N1 and N2.

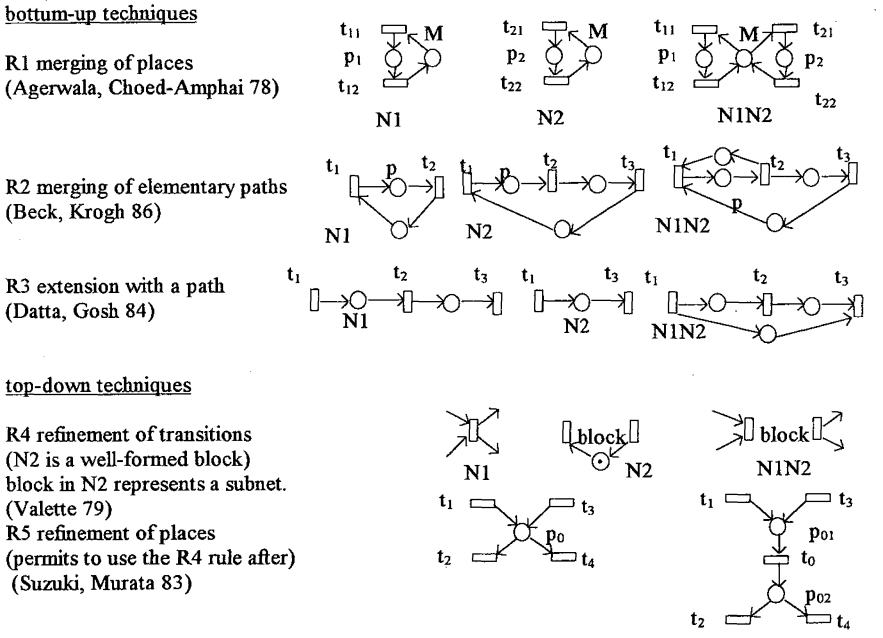


Figure 3. – Synthesis techniques of Petri nets.

Example 1: The necessity to use the two techniques is illustrated for the modelling of limited buffer capacity between machines (*fig.* 4). Since the net is timed, that modelling means that the two machines are serialized. It can

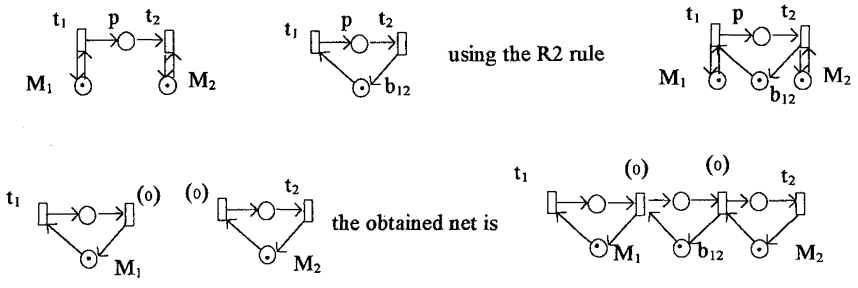


Figure 4. – The need of an hybrid synthesis approach.

be corrected if t_1 and t_2 are decomposed to set the events: start of operation t_2 and end of operations t_1 (using the R4 rule).

The same approach must be used to model fifo stocks. The modelling of problem $n/m/F$, constraints/$Cmax$ follows the synthesis procedure given hereafter:

Synthesis procedure

step 1: modelling each task as in (a)

step 2: setting the sequence of operations for each task as in (b)

step 3: modelling resource constraints with R2 (c)

step 4: modelling setup time and remove time of tools with R4 (h)

step 5: modelling constraints of limited stocks capacity (f)

- 5.1. building the limitation of the stock with R2
- 5.2. decomposing the input and output transitions of the stock with R4

step 6: modelling fifo stocks (d)

- 6.1. using R4 to insert a transition with a zero duration after t_1
- 6.2. decomposing the place between t_0 and t_1 to insert the stock (f) with R5
- 6.3. placing mutual exclusion loop of stocks parts with R2

step 7: modelling the lateness with R2 (e)

step 8: modelling precedence constraints with R3 (g)

step 9: merging resource places and stock places with R1.

Example 2: modelling of $n/3/F$, $b_{j,j+1}$, a_{ij}/Cmax with the above synthesis procedure. For each job i, we have following steps (*fig.* 5).

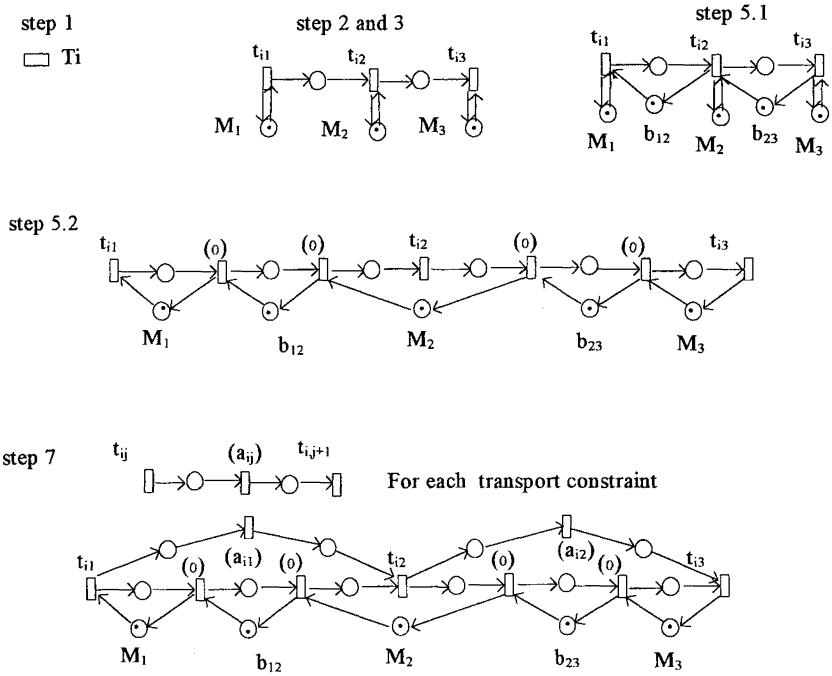
PROPERTY 2: *The global net, after synthesis, can be decomposed into:*

– *timed marked graph,*

– *shared resources constrained by n processes,* (i.e. *structure of figure* 6).

These kinds of shared resources are also used in the modelling approach of flexible manufacturing system by Zhou *et al.* (1991), and are called "parallel mutual exclusion" resources. But these authors focus on the validation of the properties as deadlock, liveness and reversibility.

The introduction of renewable resource constraints modify the structure of the net, *i.e.* it is no longer a marked graph. We show with an example that the sufficient condition, of the theorem 1, is no longer true. If t_i and t_j



step 9 : merging of places M_1 , M_2 , M_3 , b_{12} , b_{23} of the n nets gives the final one. For instance, the obtained net with two generic jobs i and j is :

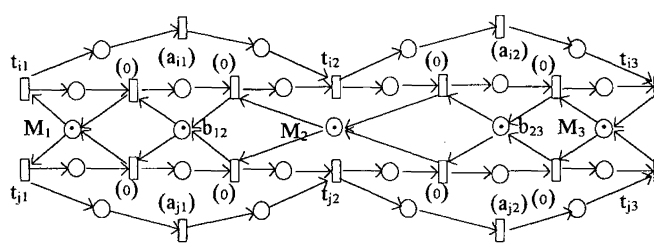


Figure 5. – Example using the synthesis procedure.

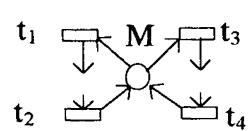


Figure 6. – Net structure of a shared renewable resource.

verify the linear inequalities, we have $x_j^n \geq \nu_j$ and $x_i^n \geq \nu_i$ where v_i and v_j are the bounds of the n-th fire of t_i and t_j. If t_i is fired first, the bound v_j becomes $v_j + d_i$ and conversely, if t_j is fired first, v_i becomes $v_i + d_j$. Therefore, there is no linearity of the inequalities according to the ordering of the allocation of the resources on the jobs. Besides, the necessary condition still holds since $\forall t_i$, $d_i \in \mathbb{N}$. In fact, resource conflicts delay firing dates.

5. SOLVING WITH CONSTRAINT LOGIC PROGRAMMING

Solving scheduling problems consists on the computation of optimal controlled execution with a given characteristic vector. Among emergent resolution techniques we focus our interest on logic programming with constraint propagation (CLP). We use the CHIP language (Van Hentenryck, 1989, and Cosytec, 1993).

CLP does not restrict itself to the manipulation of symbolic terms, and extends the mecanisms of logic programming to different domains: booleans, rationals, finite domains. A finite domain variable is an integer variable which takes its values in a finite non empty set. Linear terms are built with these variables and + and × operators. A constraint is the comparison of two linear terms with classical arithmetic comparators. Symbolic constraints, specially designed for specific problems, are also predefined. For instance the cumulative constraint of the CHIP language is designed to serve in the resolution of scheduling problems (Aggoun *et al.*, 1993).

A CLP program is classically divided in two parts. In the first one, variables are declared and the constraints are set. In the second one, values are assigned to variables, such that the optimum is reached (with a build in branch and bound technique). The disjunctive constraints generated by a shared resource, can be set in different ways. An efficient one is the build-in cumulative constraint of the CHIP language.

The PN graph, which models the scheduling problem, is represented in the CHIP language by a set of clauses. Some additive clauses define the list of places, transitions, firing durations, the initial marking. The part of the model, which is a marked graph, is represented by the set of place/transition incidence relations. In fact a clause has the form *edge(rel,p,t)* where *rel* takes its values in {in, out}. The resource part of the model is coded like *edge_res(p,treq,trel,v)* where *treq* and *trel* are the request and the release transitions for the resource p and its required quantity v. These correspondances are summarized in Figure 7.

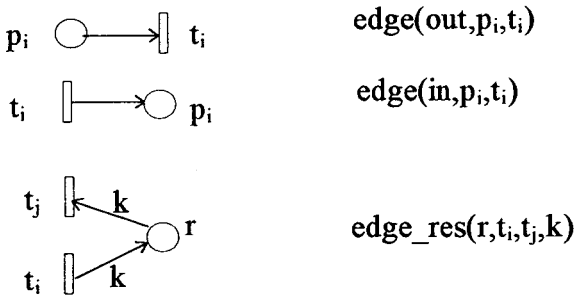
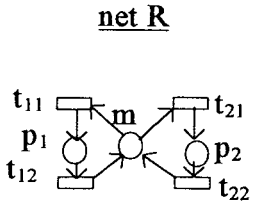


Figure 7. – Transcription rules of the net into CLP clauses.

Other lists are added to complete the data (places, resources, durations, marking, characteristic vector).

Example 3: The example herafter shows the transcription of a net in the CHIP language (Figure 8).

Chrétienne (1983) presents an algorithm to compute the completion time of M fires in a TMG. The principle is to unfold the marked graph by considering every fire of the same transition as a new vertex. In this new graph, called the developped graph, every edge represents a potential constraint between each fire. It is a generalization of the conjunctive graph, as defined by Roy (1970), on which the set of minimal potential is build. The optimum is reached by taking the critical path (the path for which the sum of the potentials is maximal). The use of the CLP will avoid us to build the developped graph by using directly the inequality of theorem 1. Moreover, we can deal with both disjunctive and conjunctive constraints (Richard *et al.,* 1995b).



clauses

edge(out,p_1,t_{11})
edge(in,p_1,t_{12})
edge(out,p_2,t_{21})
edge(in,p_2,t_{22})
edge_res(m,t_{12},t_{11},1)
edge_res(m,t_{22},t_{21},1)

Figure 8. – Transcription of a net into a CHIP program.

The resolution of the scheduling problem uses a finite domain variable for each fire occurence. In the particular case of the flowshop, each transition is fired only once. Constraints on firing dates of a marked graph are potential constraints. For every input place p of t, we set an inequality constraint: $x_t^1 \geq 0$ if the marking of p is equal to $1\,(m\,(p) = 1)$, else $x_t^1 \geq x_{t'}^1 + d_{t'}$, where t' is the input transition of p. An algorithm which sets potential constraints in the CHIP language for marked graph is given by Richard (1994). Resource constraints use the built-in cumulative constraint. We note before edge_res $(p,\ X\,t,\ Y\,t,\ V\,t)$, where $X\,t$, $Y\,t$, $V\,t$ are finite domain variables. Parameters of cumulative are: Starts (list of the demand dates of the resource: $X\,t$), Durations (a list of non instanciated finite domain variables), Ends (list of release dates of the resource: $Y\,t + Dt$), Quantity (list of the quantity of resource required for the operation: $V\,t$), High (the total amont of the resource – it is the initial marking of the place p). In the above example, the constraint set will be: cumulative $([x_{t_{11}}^1,\ x_{t_{21}}^1],\ [D1,\ D2],\ [x_{t_{12}}^1,\ x_{t_{22}}^1],\ [1,1],\ 1)$. Constraints computing the $Cmax$ are $\forall\, t \in T,\ Cmax \geq x_t^{N_t} + d_t$. The start of the CHIP solver, associated to finite domain, is made by assigning values to free variables. The constraint propagation phasis restricts as many domains as possible. The skeleton of the algorithm is given hereafter:

Skeleton of the algorithm:

step 1: build the list of firing dates (finite domain variables)

step 2: set potential constraints

set renewable resource constraints

set constraints for the $Cmax$ computation

step 3: optimize with a built-in branch and bound procedure technique

We describe the algorithm which sets the potential constraints in an imperative form to make understanding easier (Figure 9). The constraints that we must set take the form of the theorem 1. The number of fires, fixed by the user, allows to build the characteristic vector of the controlled execution. N_i is the number of fire of t_i.

Cavalier *et al.* (1995) have developped a prototype, called OPTNET. We report some result hereafter.

6. IMPLEMENTATION AND RESULTS

The software developped is OPTNET (OPTimization of Petri NETs) with the CHIP language on a Sun Sparc Station under Solaris 2.2. The net

```
Algorithm potential-constraints
Begin
    For i←1 à n do
        For all j∈Γ+(i) do
        /* successor of ti in the graph */
        k←nj
            While k-Mij > 0 do
                set contraint {x_i^(k-Mij) + d_i ≤ x_j^k}
                k←k-1
            End while
        End for
    End for
End
```

Figure 9. – Algorithm which set potential constraints.

is described in a textual form as we seen in the previous sections. But there's no problem to design the net through a graphical interface, by using transcription rules defined below. The global architecture of the software is given Figure10.

The graphic interface is used to parametrize the net (duration associated to transitions, the number of tokens in each place), and the resolution characteristics (bound of time, presentation of Gantt diagrams). Gantt diagrams can be drawn with one line per transition, or one line for a set of transitions (*i.e.* a job), which is set with the graphical interface. During the resolution step, each solution found is drawn in the Gantt diagram window. To avoid long resolution time, the user can provide a maximal resolution time after which the solver must stop. When the time is elapsed, the solver is stopped and the best solution that was found is given. In general, the optimal solution is found fastly, but it takes a lot of time for CHIP to prove its optimality. For our tests, we use the flow-shop problem. It is well known

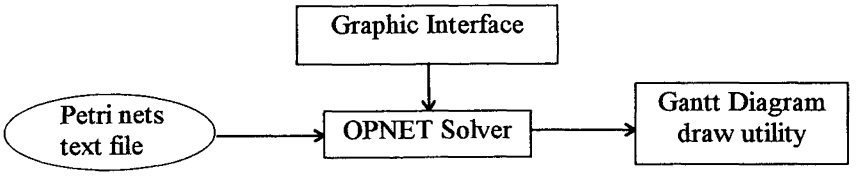


Figure 10. – OPTNET software architecture.

that permutation schedules (*i.e.* stocks between machines are managed in fifo) are dominant (a subset of optimal schedules) for the $n/2/F/$Regular Rule and for the $n/3/F/Cmax$ problems (Baker, 1974). So, most resolution techniques (optimal or heuristic) limit the search space to the permutation schedules. Optnet doesn't exploit this kind of dominance properties, and searches a solution in the general case.

We report some results, in figure 11, one the $n/2/F/Cmax$, the $n/2/F, Snsd, Rnsd/Cmax$, and the $n/m/F/Cmax$ ($m \geq 3$) problems. Those two first problems are solved optimally with polynomials algorithm J, defined by Johnson (1954) and SH designed by Sule *et al.* (1983). Cavalier *et al.* (1995) have implemented them with the CHIP language too.

Polynomial Problems	OPTNET (ms)	Classical algorithms (ms)
Pb1 (4/2/F/Cmax)	230	59.7 (J)
Pb2 (5/2/F/Cmax)	510	74.4 (J)
Pb3 (6/2/F/Cmax)	300	94.8 (J)
Pb4 (2/2/F,Snsd,Rnsd/Cmax)	873	26.2 (SH)
Pb5(3/2/F,Snsd,Rnsd/Cmax)	342285	39.8 (SH)
Pb6(4/2/F,Snsd,Rnsd/Cmax)	NS*	56.4 (SH)

Strongly NP-hard Problems	OPTNET (ms)
Pb7 (3/3/F/Cmax)	320
Pb8 (4/3/F/Cmax)	370
Pb9 (4/4/F/Cmax)	230
Pb10 (4/5/F/Cmax)	800
Pb11(4/9/F/Cmax)	2680
Pb12(5/4/F/Cmax)	4600

NS* : not solve after one day of computing

Figure 11. – Results.

We can note that the resolution time with Optnet is very large in comparison with the polynomial algorithm. But, as we noted before, the polynomial algorithm exploits the dominance property of permutation schedules. That considerably reduces the search space. Optnet has difficulties to solve a $n/2/F, Snsd, Rnsd/Cmax$ because the durations of the operations in the cumulative constraint are not known at the beginning of the resolution phasis. In fact, the durations are set with constraints. In that case, solving

optimally with CHIP is much cheaper. The resolution time increases mainly with the number of jobs. Default search strategies are not adapted to the resolution of scheduling problems. So, it seems necessary to take into account the specific constraints of the problem, as regular criterion and permutation schedules properties, if we want to solve optimally problems such as: $n/m/F, Snsd, Rnsd, r_i$, lag times, limited buffer capacity,.../$Cmax$ (even for small size problems). OPTNET is a flexible resolution tool with a top-down approach. In this way, Petri nets are used to specify the set of feasible schedules with a graphical tool.

7. CONCLUSION

We have presented a homogeneous approach to solve (shop) scheduling problems submitted to various constraints. The PN advantages have been explained in the first part, then a modelling technique has been exposed, which is based on a hybrid synthesis procedure. And then, the resolution of the modeled problem has been obtained by using the CHIP CLP language. But, according to our experience, default search strategies of CLP language are, in general, not adapted to the resolution of NP-hard problems. The use of constraint and solver as black boxes forbids efficient implementations.

But, in most cases the optimum is not required. As Ackoff (1977) said "preoccupation with optimization leads to a withdrawal from reality". Industrial problems require more flexibility than optimality. So a perspective of our work is to design a computer aided approach to take into account human decisions.

ACKNOWLEDGMENT

The authors are very grateful to the anonymous reviewers for their important and constructive comments.

REFERENCES

R. L. Ackoff, Optimization + objectivity = opt out, *European Journal of Operational Research,* 1977, *1.*

A. Aggoun and N. Beldeceanu, Extending CHIP in order to solve complex scheduling and placement problems, *Mathematical and computer modelling,* 1993, *17,* (7), pp. 57-73.

K. R. Baker, *Introduction to sequencing and scheduling*, John Wiley & Sons.

A. Bauer, R. Bowden, J. Browne, J. Duggan and G. Lyons, *Shop floor Control Systems – From design to implementation,* 1991, Chapman & Hall.

P. BAPTISTE, C. H. CHO, J. FAVREL et M. ZOUHRI, Une caractérisation analytique des ordonnancements admissibles sous contraintes hétérogènes en flow-shop, *RAIRO-APII*, 1991, *25*, pp. 87-102.

P. BAPTISTE, B. LEGEARD and H. ZIDOUM, Sequences constraint solver based on P-Q-R trees, ILPS'94, New York, 1994, 14 p.

J. CARLIER and P. CHRÉTIENNE, Problèmes d'ordonnancement (modélisation/complexité/algorithmes), Masson, collection Études et Recherche en Informatique, 1987.

J. CARLIER and P. CHRÉTIENNE, Timed Petri net Schedules, *Advances on Petri nets,* 1988, Springer Verlag.

C. CAVALIER, N. JACQUET and P. RICHARD, Problèmes d'ordonnancement en Flowshop: Réseaux de Petri et PLC, *Rapport de Projet de Fin d'études de l'École d'Ingénieurs en Informatique pour l'Industrie (E31)*, Université de Tours, 1995, France.

C. W. CHENG T. H. SUN and L. C. FU, Petri net based modelling and scheduling of flexible manufacturing system, IEEE International Conference on Robotics and Automation, 1994, pp. 513-520.

P. CHRÉTIENNE, *Réseaux de Petri temporisés,* thèse d'état, Paris VI, 1983.

F. CHU, J. M. PROTH and V. M. SAVI, Ordonnancement basé sur les réseaux de Petri, *Rapport de Recherche INRIA*, 1993, n° 1960.

A. COLMERAUER, An introduction to Prolog III, *Communications of the ACM*, 1990, *33*, (7), pp. 69-90.

Cosytec, 1993, CHIP V4, *user manual*, Cosytec.

F. DICESARE, G. HARHALAKIS, J.-M. PROTH, M. SILVA and F. B. VERNADAT, *Practice of Petri nets in manufacturing Systems*, Chapman et Hall, 1993.

R. FOURE D. M. GAY and B. W. KERNIGHAN, *AMPL: A Modelling language for mathematical programming,* The scientific Press, 1993.

S. FRENCH, *Sequencing and scheduling: an introduction to the mathematics of the Job-Shop*, Ellis Horwood, 1982.

GOThA (Groupe d'Ordonnancement Théorique et Appliquée), Les problèmes d'ordonnancement, *RAIRO-Operations Research,* 1993, *27*, (1), pp. 77-150.

H. HILLION and J.-M. PROTH, Performance evaluation of job-shop systems using timed event graphs, *IEEE Transactions on Automatic Control,* 1989, *34*, (1), pp. 3-9.

IJPR, *International Journal of Production Research, Special Artificial intelligence in manufacturing,* 1988, *26*, (5).

S. JULIA R. VALETTE and M. TAZZA, Analysis of the behavior of a manufacturing cell with cyclic feeding policies, *IEEE-SMC,* San Antonio, 1994, pp. 1683-1688.

S. M. JOHNSON, Optimal two- and three-stage production schedules with setup times included, *Naval Research Logistic Quality*, 1954, *1*, (1), pp. 61-68.

E. L. LAWLER, J. K. LENSTRA, A. H. G. RINNOOY Kan and D. B. SHMOYS, Sequencing and Scheduling: algorithms and complexity, *Report BS-R8909, Center for Mathematics and Computer Science,* Amsterdam, The Netherlands, 1989.

D. Y. LEE and F. DICESARE, Scheduling flexible manufacturing systems using petri nets, and heuristic search, *IEEE transactions on robotics and automation*, 1994, *10*, (2), pp. 123-132.

C. LE PAPE, Programmation par contraintes et ordonnancement : historique et perspectives, Tutorial, *9e Congrès Reconnaissance des Formes et Intelligence Artificielle*, Paris, France, 1994a.

C. LE PAPE, Implementation of resource constraints in Ilog Schedule: a library for the development of constraint-based scheduling systems, *Intelligent Systems Engineering,* 1994b, *3*, pp. 55-66.

M.-L. LEVY, P. LOPEZ and B. PRADIN, Characterization of feasible schedules for the flow-shop problem: a decomposition approach, *INRIA European Workshop on Integrated manufacturing Systems Engineering* (IMSE'94), Grenoble, 1994, pp. 307-315.

A. MUNIER, Régime asymptotique des graphes d'événements temporisés généralisés : application à un problème d'assemblage, *RAIRO-APII*, 1993, *27*, (5), pp. 487-513.

T. MURATA, Petri nets: Properties, Analysis and Applications, *Proceedings of the IEEE,* 1989, *77*, (4), pp. 541-580.

J.-M. PROTH and X. XIE, *Les réseaux de Petri pour la conception et la gestion des systèmes de production,* Masson, Paris, 1994.

C. PROUST, Using Johnson's algorithm for solving flow-shop scheduling problems, *Proceedings of the summer school on Scheduling Theory and its Application,* INRIA, Bonnas, 1992.

C. RAMCHANDANI, *Analysis of asynchronous concurrent systems by timed Petri nets,* Ph. D. Thesis, MIT, Cambridge, Project MAC-TR 120, 1974.

P. RICHARD, J. L. BOUQUARD and C. PROUST, Resolution of scheduling problems by modelling with Petri nets and a transcription in CHIP, *INRIA European Workshop on Integrated Manufacturing Systems Engineering* (IMSE'94), Grenoble, 1994, pp. 317-323.

P. RICHARD, J. L. BOUQUARD and C. PROUST, Scheduling based on Petri nets and a resolution with CHIP, *2e International Conference on Manufacturing Automation*, Nancy, 1995a, pp. 135-140.

P. RICHARD, C. CAVALIER, N. JACQUET and C. PROUST, Solving scheduling problems using Petri nets and constraint logic programming, *INRIA/IEEE International Symposium on Emerging Technologies and Factory Automation* (ETFA'95), Paris, 1995b, 9 p.

ROUBELLAT, J. C. BILLAUT and M. VILLAUMIÉ, Ordonnancement d'atelier en temps réel d'Orabaid à Ordo, *Proceedings, journées d'études Ordonnancement et entreprises: applications concrètes et outils pour le futur, CNRS/GdR Automatique/SED/GT3, Toulouse*, 1994, pp. 213-253.

B. ROY, *Algèbre moderne et théorie des graphes,* vol. 2, Dunod.

E. F. STAFFORD, F. T. TSENG, On the Srikar-Gosh MILP model for the N×M SDST flow-shop problem, *International Journal of Production Research,* 1990, *28*, (10), pp. 1817-1830.

D. R. SULE, K. Y. HUANG, Sequencing on two and three machines with setup, processing and removal times separated, *International Journal of Production Research,* 1983, *21*, (5), pp. 723-732.

C. VALENTIN, L. M. AGUILERA, P. LADET and Z. BINDER, A high level Petri nets model of generalized dynamic job-shop, *European Workshop on Integrated manufacturing Systems Engineering* (IMSE'94), INRIA, Grenoble, 1994, pp. 273-280.

Van HENTENRYCK, *Constraint satisfaction in logic programming,* MIT press, 1989.

H. ZIDOUM, P. BAPTISTE and B. LEGEARD, Extension de la PLC pour le traitement des contraintes sur les séquences, *MSCEAI'94*, Rabat, 1994, pp. 285-294.

M. C. ZHOU and F. DICESARE, Parallel and sequential mutual exclusions for Petri net modelling of manufacturing systems with shared resources, *IEEE transactions on Robotics and Automation*, 1991, *7*, (4), pp. 515-527.