Sur la commande des systèmes flexibles de production manufacturière par l'algèbre des dioïdes

Michel ALSABA, Jean-Louis BOIMOND, Sébastien LAHAYE

Laboratoire d'Ingénierie des Systèmes Automatisés (LISA)
62 avenue Notre Dame Du Lac, 49000 ANGERS, France

[alsaba, boimond, lahaye]@istia.univ-angers.fr

Résumé— **Cet article traite de la commande en juste-à-temps des systèmes flexibles de production manufacturière lorsque la trajectoire de l'entrée de référence est connue** *a priori*. **La commande est issue de résultats de base de la théorie de la résiduation dans l'algèbre des dioïdes, une règle d'ordonnancement locale permet de résoudre les conflits éventuels.**

Mots-clés— **Systèmes flexibles de production manufacturière, réseaux de Petri temporisés, graphes d'événements temporisés, graphes d'états temporisés, tas de pièces, ordonnancement, dioïde, résiduation, commande en juste-à-temps.**

I. Introduction

Ce travail a trait à la commande en Juste-A-Temps (JAT) des Systèmes Flexibles de Production Manufacturière (SFPM). Il s'agit de retarder au plus tard les entrées du système telles que ses sorties se produisent avant les dates désirées, données par l'entrée de référence. La commande en JAT proposée nécessite de connaître *a priori* la trajectoire de l'entrée de référence. Les SFPM sont caractérisés par :

1. La production de plusieurs types de pièces selon une séquence fixée de tâches à accomplir,

2. Certaines ressources (tels une machine ou un robot) partagées entre plusieurs tâches.

Pour un tel système, chaque type de pièces passe successivement par des étapes de traitement disposées de façon linéaire, comparable à une ligne de production.

Une méthode de construction systématique d'un SFPM est proposée dans [1] en utilisant les réseaux de Petri. Comme exemple d'illustration, considérons le SFPM, décrit par la figure 1, dans lequel trois types de pièces (A, B, C) sont traités. Une pièce A suit la séquence de tâches suivante : elle passe dans la machine $M1$ (la présence de deux jetons dans la place $P3$ signifie que la machine peut traiter deux pièces simultanément et indépendamment), puis elle passe dans la machine $M2$ (également utilisée pour traiter des pièces B). Une pièce B suit la séquence de tâches suivante : elle passe dans la machine $M3$ (également utilisée pour traiter des pièces C), puis elle passe dans la machine $M2$. Enfin une pièce C est traitée par la machine $M3$.

Il est connu que le calcul d'une commande optimale vis-à-vis du critère de JAT pour le type de systèmes considérés n'est généralement pas possible du fait du problème d'ordonnancement des ressources partagées. Ce problème, analogue à celui de l'ordonnancement d'un atelier de type *jobshop*, est NP-difficile lorsqu'une ressource est partagée

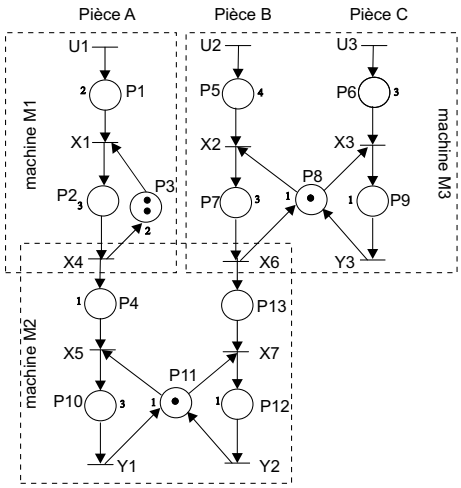

Fig. 1. Exemple d'un SFPM.

par plus de trois tâches [2]. La solution alternative que nous utilisons pour gérer les ressources partagées, est basée sur une règle d'ordonnancement locale.

La méthode proposée nécessite de décomposer les réseaux de Petri en différents graphes, de type graphe d'états ou graphe d'événements :

• Nous utilisons les graphes d'états temporisés (GEtT) pour représenter des phénomènes de choix (voir la machine $M2$, ou $M3$, décrite dans la figure 1). Dans ce papier, on utilise la règle d'ordonnancement des tâches communément appelée *Early Due Date* (EDD), ou règle de Jackson [3]. Cette règle est reconnue pour sa simplicité et fournit des résultats généralement assez bons lorsque le critère porte sur le retard algébrique (*lateness*), ce qui est le cas dans cet article (le retard algébrique correspondant à $L_i = C_i - d_i$ où C_i est la date de fin d'exécution de la tâche i (*completion time*) et d_i est la date d'achèvement souhaitée de la tâche i (*due date*)). Appliqué sur le GEtT élémentaire décrit par la figure 2, où sont seulement représentées deux types de tâches pour des raisons de clarté, cette règle d'ordonnancement est obtenu en rangeant simplement les tâches ($I1O1$, (resp., $I2O2$) correspondant au traitement d'une pièce 1 (resp., pièce 2)) par ordre de dates croissantes d'achèvement souhaitées (indiquées par une entrée de référence). Une fois cet ordonnancement fixé, nous proposons, à travers une modélisation de type *tas* des GEtT saufs (un seul jeton à la fois dans une place), de calculer la commande

optimale vis-à-vis du JAT, au sens où la ressource est attribuée le plus tardivement possible aux tâches.

• Nous utilisons les graphes d'événements temporisés (GEvT) pour représenter des phénomènes de synchronisation. Des résultats bien connus sur la résiduation dans l'algèbre $(\max, +)$ permettent le calcul d'une commande optimale en JAT pour de tels graphes, voir [4], [5]. Il est important de noter que les GEvT ne sont pas nécessairement saufs (voir la machine $M1$ décrite dans la figure 1), contrairement aux GEtT supposés saufs.

Une telle approche de modélisation nous paraît préférable à une transformation globale d'un réseau de Petri temporisé en un GEvT, comme proposée dans [6], [7]. En effet, la transformation d'un réseaux de Petri temporisé en un GEvT équivalent nécessite, pour chaque entrée de référence donnée, un arbitrage *a priori* des conflits, ce qui peut s'avérer coûteux en temps de calcul de par la dimension éventuellement importante du GEvT généré. Au contraire, la représentation d'un réseaux de Petri temporisé est indépendante de l'ordonnancement, et par conséquent, de l'entrée de référence, ce qui est particulièrement intéressant dans notre cas, au sens où n'importe quelle trajectoire d'entrée de référence peut être considérée.

Cet article est organisé comme suit. Des rappels sur les dioïdes, la théorie de la résiduation, les réseaux de Petri et les modèles de type *tas* sont présentés dans la deuxième section. Nous traitons de la modélisation et de la commande des SFPM dans la troisième section, ce qui mène à décomposer les réseaux de Petri en considérant deux types de graphes, à savoir les GEtT et les GEvT. L'algorithme de contrôle est appliqué au système représenté par la figure 1 à titre d'illustration.

II. Préliminaires

A. Dioïde

Voir [4], [8, §4] pour une présentation exhaustive de la théorie des dioïdes.

Un *dioïde* $\mathcal{D}$ est un ensemble muni de deux lois de composition internes, notées $\oplus$ (addition) et $\otimes$ (multiplication), associatives et ayant chacune un élément neutre, noté respectivement ε et e, telles que $\oplus$ est commutative et idempotente (*i.e.*, $a \oplus a = a$). De plus, la loi $\otimes$ est distributive par rapport à la loi $\oplus$, et l'élément neutre ε est absorbant pour le produit (*i.e.*, $\varepsilon \otimes a = a \otimes \varepsilon = \varepsilon$). Notons que le symbole $\otimes$ est souvent omis.

Un dioïde $\mathcal{D}$ est dit *complet* s'il est fermé pour les sommes infinies et si la multiplication est distributive sur les sommes infinies, c'est-à-dire, si :

$$\forall c \in \mathcal{D} \text{ et } \forall \mathcal{A} \subseteq \mathcal{D}, c \otimes \left(\bigoplus_{x \in \mathcal{A}} x \right) = \bigoplus_{x \in \mathcal{A}} c \otimes x.$$

La borne supérieure, notée $\top$, d'un dioïde complet est la somme de tous les éléments du dioïde, elle est absorbante pour l'addition (*i.e.*, $\forall a, \top \oplus a = \top$).

Une *relation d'ordre*, notée $\succeq$, peut être associée à un dioïde $\mathcal{D}$ de par l'équivalence suivante :
$\forall\, a, b \in \mathcal{D},\ a \succeq b \Leftrightarrow a = a \oplus b$.
Cet ordre confère au dioïde complet une structure de treillis complet. Ainsi, nous pouvons introduire un opérateur Inf, noté $\wedge$, vérifiant : $\forall\ a, b \in \mathcal{D},\ a \succeq b \Leftrightarrow b = a \wedge b$.

L'ensemble $\mathbb{R} \cup \{\pm\infty\}$, muni du maximum comme opérateur $\oplus$ et de l'addition classique comme opérateur $\otimes$, est un dioïde complet, habituellement noté $\overline{\mathbb{R}}_{\max}$, avec $\varepsilon = -\infty, e = 0$ et $\top = +\infty$.

Si $\mathcal{D}$ est un dioïde, l'ensemble $\mathcal{D}^{n \times n}$ constitué des matrices de dimension $n \times n$ à coefficients dans $\mathcal{D}$, où la somme et le produit matriciels sont définis par :

$$(A \oplus B)_{ij} = A_{ij} \oplus B_{ij} \quad , \quad (A \otimes B)_{ij} = \bigoplus_{k=1}^{n} A_{ik} \otimes B_{kj},$$

est un dioïde.

B. Théorie de la résiduation

Une présentation complète de cette théorie est proposée dans [9]. Voir [8, §4.4] pour son utilisation dans l'algèbre des dioïdes.
La théorie de la résiduation fournit, sous certaines hypothèses, *la plus grande* solution à l'inégalité $f(x) \preceq b$, où f est une application isotone (*i.e.*, $a \preceq b \Rightarrow f(a) \preceq f(b)$) définie sur des ensembles ordonnés.

Soit l'application isotone $f : \mathcal{D} \to \mathcal{F}$, avec $(\mathcal{D}, \preceq_{\mathcal{D}})$ et $(\mathcal{F}, \preceq_{\mathcal{F}})$ deux ensembles ordonnés. L'application f est dite *résiduable* si pour tout $y \in \mathcal{F}$, la borne supérieure du sous-ensemble $\{x \in \mathcal{D} \mid f(x) \preceq_{\mathcal{F}} y\}$ existe et appartient à ce sous-ensemble.

Théoreme 1: [8, §4.4.2] Soit $f : \mathcal{D} \to \mathcal{F}$ une application isotone d'un dioïde complet $(\mathcal{D}, \preceq_{\mathcal{D}})$ vers un dioïde complet $(\mathcal{F}, \preceq_{\mathcal{F}})$. Sont équivalents :

(i) L'application f est *résiduable*.

(ii) Il existe une unique application isotone $f^\sharp : \mathcal{F} \to \mathcal{D}$, appelée application *résiduée* de f, telle que $f \circ f^\sharp \preceq_{\mathcal{F}} id_{\mathcal{F}}$ et $f^\sharp \circ f \succeq_{\mathcal{D}} id_{\mathcal{D}}$, où $id_{\mathcal{F}}$ et $id_{\mathcal{D}}$ sont respectivement les applications identités dans $\mathcal{F}$ et $\mathcal{D}$.

Les applications $L_a : x \mapsto ax$ et $R_a : x \mapsto xa$ définies sur un dioïde complet $\mathcal{D}$ sont résiduables. Leurs résiduées sont notées respectivement $L_a^\sharp(x) = a\backslash x$ et $R_a^\sharp(x) = x/a$.

Soient $A \in \mathcal{D}^{m \times n}, B \in \mathcal{D}^{m \times p}, C \in \mathcal{D}^{n \times p}$, la résiduation matricielle en fonction de la résiduation scalaire est définie par :

$$A\backslash B \in \mathcal{D}^{n \times p} \text{ où } (A\backslash B)_{ij} = \bigwedge_{l=1}^{m} A_{li}\backslash B_{lj},$$

$$B/C \in \mathcal{D}^{m \times n} \text{ où } (B/C)_{ij} = \bigwedge_{k=1}^{p} B_{ik}/C_{jk}.$$

De plus, on a les relations suivantes :
$B/(AC) = (B/C)/A \in \mathcal{D}^{m \times m}$,
$(AC)\backslash B = C\backslash(A\backslash B) \in \mathcal{D}^{p \times p}$.

C. Réseaux de Petri

Voir [10] pour une présentation détaillée des réseaux de Petri.

Les Réseaux de Petri Temporisés (RdPT) considérés ont des temporisations associées uniquement aux places.

Un RdPT est un 5-uplet $\mathcal{G} = (\mathcal{T}, \mathcal{P}, \mathcal{F}, \mathcal{M}, \tau)$, où $\mathcal{T}$ est un ensemble fini de transitions, $\mathcal{P}$ est un ensemble fini de places, $\mathcal{F} \subseteq (\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P})$ est un ensemble d'arcs, $M : \mathcal{P} \to \mathbb{N}$ est le marquage initial des places et $\tau : \mathcal{P} \to \mathbb{N}$ est le temps de séjour minimum d'un jeton dans les places. Nous notons $x^\bullet$ (resp., ${}^\bullet x$) l'ensemble des successeurs (resp., prédécesseurs) immédiats d'un noeud (place ou transition) x. Le *langage* du réseau de Petri $\mathcal{G}$ est l'ensemble $L \subset \mathcal{T}^*$ des séquences commençant par le marquage

initial M et se terminant par un marquage atteignable, où $\mathcal{T}^*$ est l'ensemble des mots sur l'alphabet $\mathcal{T}$.
Les transitions sont franchies selon la règle suivante. Supposons que la transition T_i soit franchissable à l'instant t, le franchissement de T_i se déroule en deux étapes :

1. A l'instant t, un jeton est retiré de chaque place $\in {}^\bullet T_i$.

2. A l'instant t, un jeton est ajouté dans chaque place $p \in T_i^\bullet$. Il peut ainsi contribuer au franchissement des transitions dans $p^\bullet$ après l'instant $t + \tau_p$, où τ_p correspond au temps de séjour minimum d'un jeton dans la place p.

Notons que cette règle de franchissement correspond à un fonctionnement *au plus tôt* des RdPT.

D. Modèle et automate de type tas

Une présentation exhaustive des automates de type *tas* est proposée dans [11] où il est notamment montré que les modèles de type *tas* sont particulièrement intéressants pour évaluer successivement un grand nombre d'ordonnancements possibles dans un RdPT sauf.

Ces modèles sont vus comme des automates $(\max, +)$, lesquels permettent le calcul du *makespan* à travers le calcul de la hauteur dite de *tas de pièces*, ceci avec une complexité indépendante de l'ordonnancement. Les modèles de type *tas* sont utilisés ici pour modéliser les GEtT en vue de proposer une méthode de commande, en ce sens on s'intéresse davantage au problème d'inversion lié à ce type de modèle.
L'axe vertical permettant de représenter un tas de pièces indique la hauteur du tas, l'axe horizontal représente un nombre fini de *slots*. Une *pièce* est un bloc solide (éventuellement non connecté) qui occupe une partie des *slots*, avec des contours supérieur et inférieur, un exemple est donné dans la figure 3. Pour une séquence ordonnée de pièces est associé un tas correspondant à l'empilement des pièces. Une pièce occupe la position la plus basse possible par rapport au sol et aux pièces éventuellement déjà empilées.

Un modèle de type *tas* est un 5-uplet $\mathcal{H} = (\mathcal{T}, \mathcal{R}, R, l, u)$, où :

- $\mathcal{T}$ est un ensemble fini de pièces.
- $\mathcal{R}$ est un ensemble fini de *slots*.
- $R : \mathcal{T} \to \mathcal{P}(\mathcal{R})$ correspond au sous-ensemble des *slots* occupés par chaque pièce.
- $l : \mathcal{T} \times \mathcal{R} \to \mathbb{R} \cup \{-\infty\}$ donne la hauteur du contour inférieur de chaque pièce pour les différents *slots*.
- $u : \mathcal{T} \times \mathcal{R} \to \mathbb{R} \cup \{-\infty\}$ donne la hauteur du contour supérieur de chaque pièce pour les différents *slots*. Par construction, nous avons : $u \geq l$.

Par convention, $l(a, r) = u(a, r) = -\infty$ si $r \notin R(a)$ et $min_{r \in R(a)} l(a, r) = 0$ (ce qui revient à dire qu'une pièce ne peut pas aller plus bas que le niveau du sol).

Le mot $\omega = a_1 ... a_k \in \mathcal{T}^*$ de longueur k est interprété comme le *tas* obtenu en empilant les k pièces $a_1, ..., a_k$ (dans cet ordre). Nous définissons le *contour supérieur* du *tas* ω comme le vecteur ligne $x_\mathcal{H}(\omega)$, de dimension $card(\mathcal{R})$ où $x_\mathcal{H}(\omega)_r$ est la hauteur du *tas* relativement au *slot* r. Par exemple, on a relativement au tas de pièces correspondant au mot $I1O1$ représenté sur la figure 3 : $x_\mathcal{H}(I1O1)_1 = \tau_1, x_\mathcal{H}(I1O1)_2 = \tau + \tau_1, x_\mathcal{H}(I1O1)_3 = -\infty$.
L'hypothèse d'un plan horizontal, correspondant au sol, impose que $x_\mathcal{H}(e) = (0, ..., 0)$ où e est le mot vide. Ainsi, la *hauteur* du *tas* ω est $y_\mathcal{H}(\omega) = max_{r \in \mathcal{R}} x_\mathcal{H}(\omega)_r$.

Théoreme 2: [11] Soit $\mathcal{H} = (\mathcal{T}, \mathcal{R}, R, l, u)$ un modèle de type *tas*. L'automate de type *tas* $(\mathcal{R}, {I^t}_\mathcal{R}, I_\mathcal{R}, \mathcal{M})$, associé au modèle de type *tas* $\mathcal{H}$, reconnaît le contour supérieur $x_\mathcal{H}$ et la hauteur $y_\mathcal{H}$, ce qui signifie que $\forall \omega \in \mathcal{T}^*$:

- $x_\mathcal{H}(\omega) = {I^t}_\mathcal{R} \mathcal{M}(\omega)$,
- $y_\mathcal{H}(\omega) = {I^t}_\mathcal{R} \mathcal{M}(\omega) I_\mathcal{R}$,

où :
-$I_\mathcal{R}$ est un vecteur colonne de dimension $\mathcal{R}$ dont les éléments sont égaux à e,
-$\mathcal{M}$ est le morphisme[1] défini par :
$\mathcal{M} : \mathcal{T}^* \to \overline{\mathbb{R}}_{\max}^{\mathcal{R} \times \mathcal{R}}, a \mapsto \mathcal{M}(a) = I \oplus [\tilde{l}(a, .)]^t u(a, .)$,
où I est la matrice identité définie par $I_{ii} = e, I_{ij} = \varepsilon, i \neq j$, et $\tilde{l}(a, i) = -l(a, i)$ si $l(a, i) \neq \varepsilon$ et $\tilde{l}(a, i) = \varepsilon$ autrement.

On vérifie que, de façon équivalente, on a :

$$\left\{ \begin{array}{lcl} x_\mathcal{H}(e) & = & {I^t}_\mathcal{R}, \\ x_\mathcal{H}(\omega a) & = & x_\mathcal{H}(\omega) \mathcal{M}(a), \\ y_\mathcal{H}(\omega) & = & x_\mathcal{H}(\omega) I_\mathcal{R}. \end{array} \right. \quad (1)$$

Une interprétation possible d'un RdPT par un modèle de type *tas* consiste à considérer les pièces comme des tâches et les *slots* comme des ressources. Chaque tâche "a" nécessite, pour être réalisée, un sous-ensemble de ressources (donné par $R(a)$) durant un certain temps (égal à $u(a, r) - l(a, r)$ pour la ressource $r \in R(a)$). En fait, dans le cadre des RdPT, $x_\mathcal{H}$ et $y_\mathcal{H}$ correspondent à des dateurs au sens où $x_\mathcal{H}(\omega)_r$ représente le temps d'achèvement de la tâche r à l'issu de l'application du mot ω.

Théoreme 3: [11] Soit $\mathcal{G} = (\mathcal{T}, \mathcal{P}, \mathcal{F}, M, \tau)$ un RdPT sauf auquel est associé un langage L, alors le modèle de type *tas* $\mathcal{H} = (\mathcal{T}, \mathcal{P}, R, l, u)$, où :

- $\forall a \in \mathcal{T}, R(a) = a^\bullet \cup {}^\bullet a$,
- $\forall a \in \mathcal{T}, \forall p \in a^\bullet, u(a, p) = \tau_p$,
- $\forall a \in \mathcal{T}, \forall p \in {}^\bullet a \backslash a^\bullet, u(a, p) = 0$, où "\" est la soustraction des ensembles,
- $\forall a \in \mathcal{T}, \forall p \in R(a), l(a, p) = 0$,

est tel que :
$\forall \omega \in L, x_\mathcal{G}(\omega) = x_\mathcal{H}(\omega), y_\mathcal{G}(\omega) = y_\mathcal{H}(\omega)$,
ce qui signifie que les fonctions *dateurs* du vecteur $x_\mathcal{G}$ et du scalaire $y_\mathcal{G}$ du RdPT $\mathcal{G}$ coïncident respectivement avec le contour supérieur $x_\mathcal{H}$ et la hauteur $y_\mathcal{H}$ du *tas* associé au modèle $\mathcal{H}$.

Il résulte de ces deux théorèmes que les dates de franchissement des transitions d'un RdPT sauf sont reconnus par un automate de type *tas*.

Par exemple, considérons le GEtT décrit par la figure 2, lequel peut correspondre aux machines M2 ou M3, représentées dans la figure 1.

Le modèle de type *tas* associé est défini par :
$\mathcal{T} = \{I1, I2, O1, O2\}, \mathcal{R} = \mathcal{P} = \{P1, P, P2\}$;
$R(I1) = R(O1) = \{P1, P\}, R(I2) = R(O2) = \{P, P2\}$;
$u(I1, .) = [\tau_1, 0, -\infty], l(I1, .) = [0, 0, -\infty]$,
$u(I2, .) = [-\infty, 0, \tau_2], l(I2, .) = [-\infty, 0, 0]$,

[1] $\mathcal{M}$ est un morphisme si $\mathcal{M}(a \oplus b) = \mathcal{M}(a) \oplus \mathcal{M}(b)$.

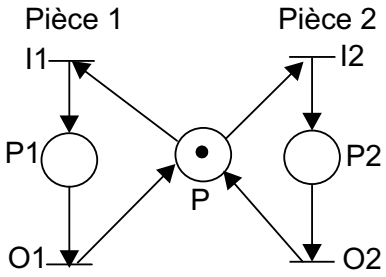

Fig. 2. Un GEtT élémentaire.

$u(O1,.) = [0, \tau, -\infty], l(O1,.) = [0, 0, -\infty]$,
$u(O2,.) = [-\infty, \tau, 0], l(O2,.) = [-\infty, 0, 0]$.

Les tas de pièces associés aux mots $I1, O1$ et $I1O1$ sont représentés dans la figure 3. Par exemple, nous pouvons déduire directement les valeurs $x_{\mathcal{H}}(I1O1) = [\tau_1, \tau_1\tau, -\infty]$ et $y_{\mathcal{H}}(I1O1) = \tau_1\tau$, où $\tau_1\tau$ correspond à $\tau_1 + \tau$ dans l'algèbre usuelle.

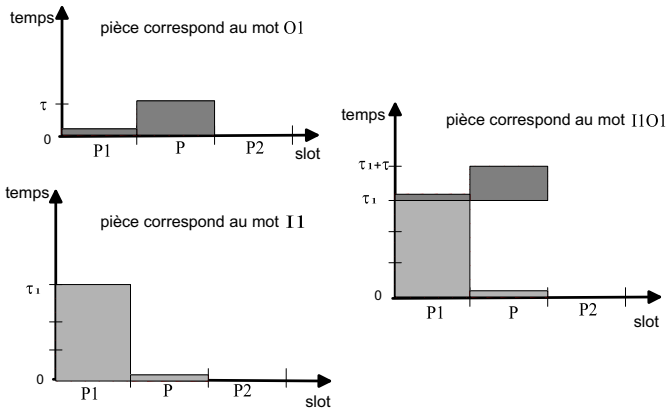

Fig. 3. Tas de pièces associés aux mots $I1, O1$ et $I1O1$.

L'automate de type *tas* correspondant est défini par le 4-uplet $(\mathcal{P}, (e\ e\ e), (e\ e\ e)^t, \mathcal{M})$, où $\mathcal{M}$ est tel que :

$$\mathcal{M}(I1) = I \oplus \begin{pmatrix} -e \\ -e \\ \varepsilon \end{pmatrix} \begin{pmatrix} \tau_1 & e & \varepsilon \end{pmatrix} = \begin{pmatrix} \tau_1 & e & \varepsilon \\ \tau_1 & e & \varepsilon \\ \varepsilon & \varepsilon & e \end{pmatrix},$$

$$\mathcal{M}(O1) = I \oplus \begin{pmatrix} -e \\ -e \\ \varepsilon \end{pmatrix} \begin{pmatrix} e & \tau & \varepsilon \end{pmatrix} = \begin{pmatrix} e & \tau & \varepsilon \\ e & \tau & \varepsilon \\ \varepsilon & \varepsilon & e \end{pmatrix},$$

de la même manière, on a :

$$\mathcal{M}(I2) = \begin{pmatrix} e & \varepsilon & \varepsilon \\ \varepsilon & e & \tau_2 \\ \varepsilon & e & \tau_2 \end{pmatrix}, \mathcal{M}(O2) = \begin{pmatrix} e & \varepsilon & \varepsilon \\ \varepsilon & \tau & e \\ \varepsilon & \tau & e \end{pmatrix}.$$

Il en résulte que $\mathcal{M}(I1O1) = \mathcal{M}(I1)\,\mathcal{M}(O1) =$

$$\begin{pmatrix} \tau_1 & e & \varepsilon \\ \tau_1 & e & \varepsilon \\ \varepsilon & \varepsilon & e \end{pmatrix} \begin{pmatrix} e & \tau & \varepsilon \\ e & \tau & \varepsilon \\ \varepsilon & \varepsilon & e \end{pmatrix} = \begin{pmatrix} \tau_1 & \tau_1\tau & \varepsilon \\ \tau_1 & \tau_1\tau & \varepsilon \\ \varepsilon & \varepsilon & e \end{pmatrix}.$$

et $\mathcal{M}(I2O2) = \begin{pmatrix} e & \varepsilon & \varepsilon \\ \varepsilon & \tau_2\tau & \tau_2 \\ \varepsilon & \tau_2\tau & \tau_2 \end{pmatrix}$.

III. Modélisation et commande des SFPM

Nous traitons de la commande en JAT des SFPM : étant données les dates de franchissement désirées des transitions de sortie, définie par la fonction *dateur* $Z = \{z(k)\}_{k=0,...,k_f}$, il s'agit de trouver les dates de franchissement *au plus tard* des transitions d'entrée $U = \{u(k)\}_{k=0,...,k_f}$ telles que les franchissements des transitions de sortie $Y = \{y(k)\}_{k=0,...,k_f}$ se produisent *avant* les dates désirées. Dans un contexte de production, une telle commande revient à satisfaire la demande des clients tout en minimisant les stocks internes. Ce problème de poursuite de sortie (classique en automatique) est résolu de façon optimale par l'intermédiaire de la théorie de la résiduation dans le cas des GEvT [4], [5].

Dans la méthode que nous proposons, les RdPT sont décomposés en GEvT et en GEtT afin de séparer les phénomènes de synchronisation des phénomènes de choix. La modélisation et la commande de ces graphes sont décrites dans les deux sections suivantes. La loi de commande appliquée dans le cadre des GEtT est telle que l'ordonnancement de la ressource partagée est obtenu en rangeant les tâches par dates croissantes d'achèvements souhaités indiqués par l'entrée de référence. Cette régle est appelée la règle EDD, ou la règle de Jackson [3].

A. Modélisation et commande des GEvT

Les GEvT sont particulièrement adaptés pour modéliser les phénomènes de synchronisation et correspondent à des systèmes dynamiques linéaires dans l'algèbre des dioïdes (voir [4], [8]). Le comportement des GEvT est représenté, dans le cas présent, dans le dioïde $\overline{\mathbb{R}}_{\max}$. Aussi, pour un GEvT donné, correspondra, par la suite, la représentation d'état suivante :

$$\begin{cases} x(k) &= Ax(k-1) \oplus Bu(k), \\ y(k) &= Cx(k). \end{cases} \tag{2}$$

Pour une entrée de référence Z définie *a priori*, la commande optimale en JAT, notée u_{JAT}, pour le modèle décrit par les équations (2) est défini par les équations d'états-adjoints suivantes :

$$\begin{cases} \xi(k) &= A^{\backprime}\xi(k+1) \wedge C^{\backprime}Z(k), \\ u_{JAT}(k) &= B^{\backprime}\xi(k), \end{cases} \tag{3}$$

où ξ représente le vecteur d'états-adjoints (voir [8, §5.6]). Une extension au cas des GEvT non-stationnaires est proposée dans [12].

B. Modélisation et commande des GEtT

Un automate de type *tas* peut être vu comme un système $(\max, +)$-linéaire dont la dynamique est pilotée par des lettres. Pour une séquence donnée de lettres $a_1...a_k$, nous posons $x(k) = x(a_1...a_k)$, les équations (1) correspondent alors à la relation récurrente suivante à l'étape k :

$$\begin{cases} x(k) &= x(k-1)\mathcal{M}(a_k), \\ y(k) &= x(k)F, \end{cases} \tag{4}$$

pour $k \geq 1$ et $x(0) = (0, ..., 0)$.
Considérons le GEtT élémentaire de la figure 2 où $I1, I2$ sont des transitions d'entrée et $O1, O2$ sont des transitions internes (et de sorties) du graphe. Les composantes du vecteur d'état $x(k)$ sont les dates de disponibilité du k-ième jeton dans les places $P1, P, P2$.
De telles équations d'état représentent un système non-stationnaire en posant $A(k) = \mathcal{M}(a_k)$ la matrice associée à

l'étape k, on aura $a_k = I1O1$ ou $I2O2$ selon le choix de la séquence fait à l'étape k. Soient $F_1 = (e, \varepsilon, \varepsilon), F_2 = (\varepsilon, \varepsilon, e)$, les vecteurs *d'interface* qui permettent de relier les composantes des vecteurs d'états, de type *dateurs de transitions*, aux modèles de type *tas*, dont les composantes des vecteurs d'états sont des *dateurs de places* comme décrits précédemment.

Finalement, en ajoutant des entrées au modèle, les équations non-autonomes deviennent :

$$\left\{ \begin{array}{lcl} x(k) & = & x(k-1)A(k) \oplus u(k)B(k), \\ y(k) & = & x(k)C(k). \end{array} \right. \quad (5)$$

où $u = (I1, I2), x = (x_{P1}, x_P, x_{P2}), y = (O1, O2)$,
$B(k) = F_1\mathcal{M}(I1O1)$, resp., $F_2\mathcal{M}(I2O2), C(k) = F_1^t$, resp., F_2^t, selon la séquence $I1O1$, resp., $I2O2$ réalisée à l'étape k.

D'une façon similaire à [12], nous calculons :
$x(k) = \bigoplus_{j \leq k} u(j)B(j)\Phi(j,k)$
et
$y(k) = \bigoplus_{j \leq k} u(j)B(j)\Phi(j,k)C(k)$,
où $\Phi(j,k)$ est la matrice de transition donnée par :

$$\Phi(j,k) = \begin{cases} \text{non définie} & \text{si } j > k, \\ I & \text{si } j = k, \\ A(j+1)A(j+2)...A(k) & \text{autrement.} \end{cases}$$

Ainsi, nous déduisons la commande en JAT, notée u_{JAT_i}, donnée par :
$u_{JAT_i}(k) = \bigwedge_{k \leq j} Z_i(j) \not\backslash (B(k)\Phi(k,j)C(j))$,
soit,
$u_{JAT_i}(k) = \bigwedge_{k \leq j} Z_i(j) \not\backslash (F_i(k)A(k)A(k+1)..A(j)F_i^t(j))$,
où l'entrée de référence $Z_i(j)$ correspond à $Z_1(j)$, resp., $Z_2(j)$ (c'est-à-dire, la sortie désirée Y_1, resp., Y_2) selon la séquence $I1O1$, resp., $I2O2$, réalisée à l'étape j.
En posant $\xi(k) = \bigwedge_{k \leq j} Z_i(j) \not\backslash (A(k)A(k+1)..A(j)F_i^t(j))$ le vecteur d'états-adjoints nous obtenons (en utilisant II.B) ces équations d'états-adjoints :

$$\left\{ \begin{array}{lcl} \xi(k) & = & \xi(k+1) \not\backslash A(k) \wedge Z_i(k) \not\backslash (A(k)F_i^t(k)), \\ u_{JAT_i}(k) & = & \xi(k) \not\backslash F_i(k), \end{array} \right.$$

Le fait d'utiliser la règle d'ordonnancement de Jackson, laquelle vise à ordonnancer une ressource afin de respecter l'ordre des dates croissantes indiquée par la référence d'entrée, correspond à appliquer l'algorithme de commande suivant à l'étape k (avec $k = l + h$) :
si $Z_1(l) \preceq Z_2(h)$ alors

$$\left\{ \begin{array}{lcl} \xi(k) & = & (\xi(k+1) \wedge Z_2(h) \not\backslash F_2^t) \not\backslash \mathcal{M}(I2O2), \\ \Xi_2(h) & = & \xi(k) \not\backslash F_2, \\ h & = & h - 1, \end{array} \right. \quad (6)$$

sinon

$$\left\{ \begin{array}{lcl} \xi(k) & = & (\xi(k+1) \wedge Z_1(l) \not\backslash F_1^t) \not\backslash \mathcal{M}(I1O1), \\ \Xi_1(l) & = & \xi(k) \not\backslash F_1, \\ l & = & l - 1, \end{array} \right.$$

où $\Xi_1(l)(\text{resp.}\Xi_2(h))$ correspond à la date de franchissement de la transition d'entrée $I1(l)(\text{resp.}I2(h))$. Il apparaît clairement que la règle d'ordonnancement de la ressource partagée réalise une sélection parmi les séquences $I1O1$ et $I2O2$ à chaque étape k de l'algorithme. Notons que le test $Z_1(l) \preceq Z_2(h)$ donne une priorité à la première ligne de production $(I1, O1)$ quand la date de la consigne est la même pour les deux lignes.

En fait, la commande obtenue est équivalente à la commande optimale en JAT du GEvT correspondant au RdPT obtenu en considérant l'ordonnancement utilisé.

C. Modélisation et commande des SFPM

Chaque ligne de production d'un SFPM est un assemblage de GEtT et de GEvT. Considérons, par exemple, le SFPM de la figure 1, il est aisé de déduire sa correspondance en termes de sous-systèmes (*cf.* figure 4).

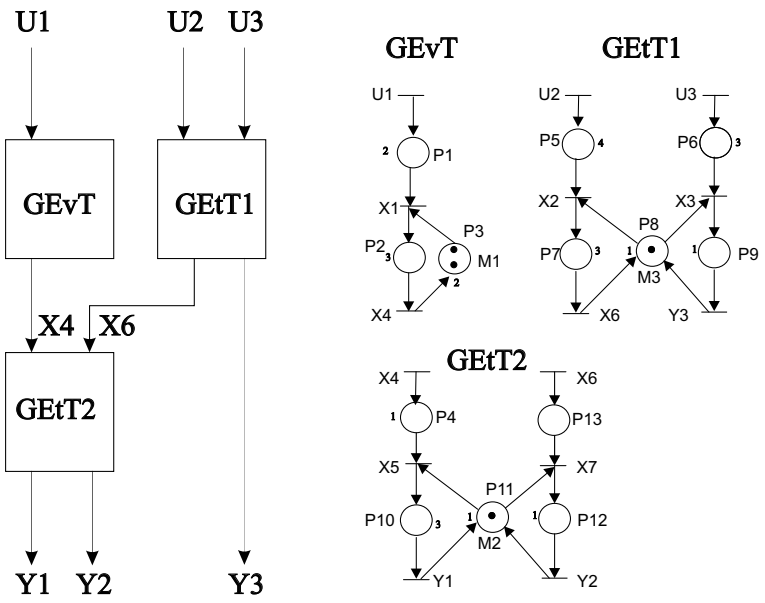

Fig. 4. Diagramme correspondant au SFPM de la figure 1.

Pour la mise en série des sous-modèles GEtT ou GEvT, il est supposé l'existence d'une zone de stockage de capacité infinie entre chaque sous-modèle (représentées par les places $P1, P4, P5, P6, P13$ dans le SFPM de la figure 1). Cette hypothèse permet alors de calculer la loi de commande par applications successives des algorithmes proposés pour chaque sous-modèle (à travers l'utilisation des équations (3) pour un sous-modèle de type GEvT et (6) pour un sous-modèle de type GEtT).

Étant donnés Z_1, Z_2, nous obtenons ξ_4, ξ_6 en appliquant l'algorithme (6) au GEtT2. Étant donné ξ_4, nous obtenons U_{1JAT} en appliquant l'algorithme (3) au GEvT. Finalement, étant donné ξ_6, Z_3, nous obtenons U_{2JAT}, U_{3JAT} en appliquant l'algorithme (6) au GEtT1. Notons que ξ_2, ξ_3 (resp., ξ_5, ξ_7) donnent les dates où la ressource de la machine M3 (resp., M2) est assignée aux pièces.

D. Application numérique

Appliquons l'algorithme de commande précédent à l'exemple de la figure 1, en considérant l'entrée de référence :

$$Z = \begin{pmatrix} 14 & 32 & 33 & 34 & 35 & 41 \\ \varepsilon & \varepsilon & \varepsilon & 30 & 31 & 43 \\ 10 & 11 & 12 & 13 & 24 & 36 \end{pmatrix}.$$

Nous obtenons en appliquant l'algorithme (6) au GEtT2 :

$$(Z_4, Z_6)^t = \begin{pmatrix} 10 & 19 & 23 & 27 & 31 & 37 \\ \varepsilon & \varepsilon & \varepsilon & 16 & 18 & 42 \end{pmatrix}.$$

Puis, en prenant Z_4 et appliquant l'algorithme (3) au GEvT, nous déduisons l'entrée de commande :

$U_{1JAT} = \begin{pmatrix} 5 & 14 & 18 & 22 & 26 & 32 \end{pmatrix}$.
A présent, en prenant Z_6 et Z_3, et en appliquant l'algorithme (6) au GEtT1, nous obtenons :
$(U_{2JAT}, U_{3JAT})^t = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & 7 & 11 & 35 \\ 0 & 2 & 4 & 6 & 20 & 32 \end{pmatrix}$.
La sortie du système, induit par U_{JAT}, est donnée par :
$Y = \begin{pmatrix} 14 & 23 & 27 & 31 & 35 & 41 \\ \varepsilon & \varepsilon & \varepsilon & 16 & 19 & 43 \\ 4 & 6 & 8 & 10 & 24 & 36 \end{pmatrix}$.
Nous pouvons vérifier que $Y \preceq Z$.

IV. Conclusion et Perspectives

Dans cet article nous avons considéré la commande en juste-à-temps des systèmes flexibles de production manufacturière. A présent, nous travaillons sur la généralisation des classes de systèmes qui peuvent être considérés dans cette méthode de commande.

Références

[1] S. Amar, E. Craye, et J.C Gentina. A Method Of Hierarchical Specification And Prototyping Of FMS. *Proceedings of the IEEE-ETFA*, volume 1, pages 44–49, 1992.

[2] Y.N. Sotskov. The Complexity of Scheduling Problems with 2 and 3 Jobs. *Eur. J. Oper. Res.*, volume 53, pages 326–336, 1991.

[3] J.R. Jackson. Scheduling a Production Line to Minimize Maximum Tardiness. *Research Report 43, Management Science Research Project, University of California. Los Angeles.*, 1955.

[4] G. Cohen, P. Moller, J-P. Quadrat, et M. Viot. Algebraic Tools for the Performance Evaluation of Discrete Event Systems. *IEEE Proceedings : Special issue on Discrete Event Systems*, 77(1) :39–58, January 1989.

[5] E. Menguy, J.-L. Boimond, L. Hardouin, et J.-L. Ferrier. Just-In-Time Control of Timed Event Graphs : Update of Reference Input, Presence of Uncontrollable Input. *IEEE Trans. on Automatic Control*, 45(11) :2155–2158, November 2000.

[6] H. Hillion et J-M. Proth. Performance Evaluation of Job-Shop Systems Using Timed Event-Graphs. *IEEE Trans. on Automatic Control*, 34(1), January 1989.

[7] B. Trouillet et A. Benasser. Cyclic Scheduling Problems with Assemblies : An Approach Based to the Search of Initial Marking in a Marked Graph. *IEEE Trans. on Systems, Man and Cybernetics*, 2002.

[8] F. Baccelli, G. Cohen, G.J. Olsder, et J.-P. Quadrat. *Synchronization and Linearity : An Algebra for Discrete Event Systems.* Wiley and Sons, 1992.

[9] T.S. Blyth et M.F. Janowitz. *Residuation Theory.* Pergamon press, 1972.

[10] T. Murata. Petri Nets : Properties, Analysis and Applications. *Proceedings of the IEEE.*, volume 77(4), pages 541–580, 1989.

[11] S. Gaubert et J. Mairesse. Modeling and Analysis of Timed Petri Nets Using Heaps of Pieces. *IEEE Trans. on Automatic Control*, 44(4) :683–697, April 1999.

[12] S. Lahaye, J.-L. Boimond, et L. Hardouin. Optimal Control of (Min,+) Linear Time-Varying Systems, Petri Nets and Performance Models. Proceedings of PNPM'99, pages 170–178, Zaragoza Spain, 1999.