Laboratoire d'Ingénierie des Systèmes Automatisés

Université d'Angers

–Équipe Modèles et Systèmes Dynamiques–

DEA Automatique et Informatique Appliquée

Mémoire de DEA

thème

Modélisation et Simulation d'un Problème d'Ordonnancement

DIB ABDOU

septembre 2005

Responsables de stage : Jean-Louis BOIMOND, Professeur *LISA ANGERS*
Michel ALSABA, Doctorant *LISA ANGERS*

Laboratoire d'Ingénierie des Systèmes Automatisés
FRE 2656 CNRS
62, avenue Notre Dame du Lac
49000 Angers

Remerciements

Le travail présenté dans ce mémoire a été préparé au sein du Laboratoire d'Ingénierie des Systèmes Automatisés (LISA) dirigé par le professeur J. L. Ferrier. Je tiens à lui exprimer mes remerciements pour son accueil en stage de DEA dans les meilleurs conditions.

J'exprime toute ma reconnaissance au Professeur J. L. Boimond pour l'honneur qu'il m'a fait en acceptant de m'encadrer dans ce stage, et pour ses conseils et son aide qui ont permis d'effectuer ce travail.

Je voudrai remercier M. Alsaba pour sa coopération et ses conseils durant ce travail.

Je voudrai également remercier Marie Françoise Girard qui a effectué une partie informatique importante pour les applications.

Je voudrai remercier H. Ghassan pour la correction de l'orthographe.

Mes remerciement s'adressent aux enseignants du DEA, aux membres du LISA, et en particulier aux doctorants, pour leur disponibilité, et leurs conseils.

Table des matières

Introduction

L'ordonnancement est très important pour l'amélioration de la production dans un entreprise, où l'organisation de la production figure incontestablement en bonne place parmi les paramètres qui influencent le système de la production.

L'ordonnancement est un Algorithme qui permet l'organisation méthodique d'un processus (Le Petit Robert. Dictionnaires Le Robert, 1998), ou d'une autre façon, ordonnancer c'est programmer l'exécution d'une réalisation en attribuant des ressources aux tâches et en fixant leurs dates d'exécution (J. Carlier and P. Chrétienne. Problème d'ordonnancement : modélisation/complexité/algorithmes. Masson, 1988).

Dans la terminologie de la recherche opérationnelle, un problème d'ordonnancement désigne tout problème dans lequel l'objectif est l'allocation de ressources au cours du temps, de façon à réaliser un ensemble de tâches [David Rivreau, 1999]. Une autre définition du problème d'ordonnancement peut se formuler de façon identique, quel que soit le domaine d'application : ordonnancer un ensemble de tâches consiste à programmer leur exécution en leur allouant les ressources requises et en fixant leur date de début [Carlier and al., 1993].

Les problèmes de l'ordonnancement sont très variés, ils diffèrent d'un cas à un autre selon la nature des opérations, les caractéristiques des ressources, les contraintes portant sur les opérations, et les critères à optimiser. On peut dire qu'un problème d'ordonnancement constitue un ensemble d'opérations à organiser selon un critère d'optimisation en tenant compte de l'exploitation des ressources disponibles et des contraintes portées sur ces opérations pour effectuer une tâche déterminée.

Si on ajoute des contraintes pour exploiter des ressources, ou pour décrire une séquence de production, le problème d'ordonnancement est dit sous contraintes. Les contraintes sont, en général, des relations entre les caractéristiques temporelles des opérations avec le temps (la précedence des opérations), ou des relations temporelles décrivant les disponibilités des ressources, la complexité des problèmes d'ordonnancement augmente avec celle de l'exploitation des ressources.

L'essentiel pour résoudre un problème d'ordonnancement est de trouver un bon modèle qui décrit bien le problème pour représenter la réalité et qu'il soit suffisamment simple pour être traité, en rappelant qu'un petit changement de modèle peut entraîner de grands changements de complexité, nous allons parler dans cette étude de la modélisation des systèmes d'atelier à tâche par les réseaux de Petri et les graphes disjonctifs. Pour vérifier ces modèles nous allons utiliser le logiciel de SIMAN-ARENA pour faire la simulation des modèles des systèmes d'atelier, et logiciel de MATLAB sera utilisé pour exécuter les algorithmes relatifs avec les lois de commande.

Chapitre 1

Modélisation

1.1 Algèbre des Dioïdes

1.1.1 Dioïdes

Définition 1 (Monoïdes) *L'ensemble M muni d'une loi de composition interne notée $\oplus$ est un monoïde si $\oplus$ est associative, et admettant un élément neutre ε ($\forall$ m ϵM, m $\oplus$ ε = ε $\oplus$ m = m). Si la loi $\oplus$ est commutative, le monoïde (M, $\oplus$, ε) est dit commutatif.*

Définition 2 (Dioïdes) *L'ensemble $\mathcal{D}$ muni de deux lois de composition interne $\oplus$, $\otimes$ est un dioïde si :*
. ($\mathcal{D}$, $\oplus$, ε) est un monoïde commutatif idempotent ($\forall$ a ϵ $\mathcal{D}$, a $\oplus$ a = a),
. ($\mathcal{D}$, $\otimes$, e) est un monoïde,
. La loi $\otimes$ distribue à gauche et à droite par rapport à la loi $\oplus$, ε est un absorbant pour la loi $\otimes$, (a $\otimes$ ε = ε $\otimes$ a = ε).

Si ($\mathcal{D}$, $\otimes$, e) est un monoïde commutatif, le dioïde ($\mathcal{D}$, $\otimes$, $\oplus$) est dit commutatif.

Un dioïde $\mathcal{D}$ peut être muni d'une relation d'ordre (a $\succeq$ b $\iff$ a = a $\oplus$ b) du fait de la structure de monoïde commutatif idempotent ($\mathcal{D}$, $\oplus$), par conséquent, un dioïde est un sup-demi treillis.

Si ($\mathcal{D}$, $\otimes$, $\oplus$) est un dioïde fermé pour les sommes infinies et si la loi $\otimes$ distribue sur les sommes infinies, c'est-à-dire si pour tout c ϵ $\mathcal{D}$ et tout sous-ensemble $\mathcal{A} \subseteq \mathcal{D}$,

$$c \otimes (\bigoplus_{x \in A} x) = \bigoplus_{x \in A} c \otimes x,$$

On dit que le dioïde est complet. Il admet alors une borne supérieur T égal à la somme de ses éléments,

$$\mathcal{T} = \bigoplus_{x \in \mathcal{D}} x,$$

l'élément T est donc absorbant pour l'addition ($\mathcal{T}$ $\oplus$ a = $\mathcal{T}$).
Un dioïde complet a une structure de treillis.

Exemple 1 $\mathbb{R}_{max}$ *est le dioïde commutatif* ($\mathbb{R}$ $\cup$ {- ∞ },max,+) *muni du max (loi additive $\oplus$) et de l'addition usuelle + (loi multiplication $\otimes$). Cette structure est appelée algèbre (max,+).*

Exemple 2 $\mathbb{R}_{max}$ *n'est pas complet. il faut lui ajouter la borne supérieure $\mathcal{T} = +\infty$ sachant que $(\mathcal{T} \otimes \varepsilon) = +\infty + (-\infty) = -\infty = \varepsilon$. le dioïde complet est noté $\overline{\mathbb{R}}_{max} = (\mathbb{R} \cup \{-\infty\} \cup \{+\infty\}, max, +)$.*

Définition 3 (Relation d'équivalence) *On dit que R est une relation d'équivalence sur un ensemble X si les trois notions suivantes sont vérifiées, pour tous $x,\ y,\ z \in X$:*

1. *R est transitive : si $x_R y$ et $y_R z$, alors $x_R z$,*
2. *R est symétrique : $x_R y \Rightarrow y_R x$,*
3. *R est réflexive : $x_R x$.*

Définition 4 (Relation d'ordre) *On dit que R est une relation d'ordre sur un ensemble X si les trois notions suivantes sont vérifiées, pour tous $x,\ y,\ z \in X$:*

1. *R est transitive : si $x_R y$ et $y_R z$, alors $x_R z$,*
2. *R est antisymétrique : $x_R y \Rightarrow y_R x$,*
3. *R est réflexive : $x_R x$.*

On peut définir à l'aide de l'indempotance de la loi additive $\oplus$ une relation d'ordre dans un dioïde, le théorème suivant affirme que cette relation d'ordre est compatible avec les lois du dioïde.

Théorème 1 *(Relation d'ordre) Dans un dioïde $(\mathcal{D}, \otimes, \oplus)$, la relation $\succeq$ définie par*

$$a \succeq b \Leftrightarrow a = a \oplus b,$$

est une relation d'ordre compatible avec les lois additive $\oplus$ et multiplicative $\otimes$, c'est-à-dire,

$$a \succeq b \Rightarrow \forall c \in \mathcal{D}, a \oplus c \succeq b \oplus c,$$

$$a \succeq b \Rightarrow \forall c \in \mathcal{D}, a \otimes c \succeq b \otimes c.$$

La relation d'ordre est dite totale si

$$\forall a, b \in \mathcal{D}, a \succeq b \text{ ou } b \succeq a,$$

une condition nécessaire et suffisante pour que l'ordre d'un dioïde soit total, s'écrit de la façon suivante

$$\forall a, b \in \mathcal{D}, a \oplus b = a \text{ ou } b,$$

le dioïde est alors dit totalement ordonné.

Un dioïde muni de la relation d'ordre définie dans le théorème 1 est un demi-treillis supérieur [Birkhoff, 1940] car tout couple (a,b) admet a $\oplus$ b comme plus petit majorant (ou borne supérieure).

Dans un dioïde complet on peut définir l'opérateur $\wedge$ qui pour toute paire d'éléments (a,b) donne la borne inférieure de la manière suivante :

$$a \wedge b = \bigoplus_{\{x | x \preceq a, x \preceq b\}} x.$$

Cette borne inférieure existe pour tout sous ensemble d'un dioïde complet, ce qui donne au dioïde la structure de treillis complet. l'opérateur $\wedge$ est associatif, commutatif, idempotent et possède également un élément neutre $\mathcal{T}$.

Notation 1 (Dioïde $\overline{\mathbb{Z}}_{max}$) *On note $\overline{\mathbb{Z}}_{max}$ le dioïde complet défini sur $\mathbb{Z} \cup \{-\infty, +\infty\}$ et muni des opérations* max *et* $+$ *respectivement comme $\oplus$-addition et $\otimes$-multiplication.*

Notation 2 (Dioïde $\overline{\mathbb{Z}}_{min}$) *On note $\overline{\mathbb{Z}}_{min}$ le dioïde complet défini sur $\mathbb{Z} \cup \{-\infty, +\infty\}$ et muni des opérations* min *et* $+$ *respectivement comme $\oplus$-addition et $\otimes$-multiplication.*

Définition 5 (Étoile de Kleene) *Soit $(\mathcal{D}, \oplus, \otimes)$ un dioïde, et l'opérateur étoile $*$ est défini par :*

$$a^* = \bigoplus_{i=0}^{+\infty} a^i. \qquad (avec\ a^0 = e).$$

on notera egalement :

$$a^+ = a \oplus a^2 \oplus ... \oplus a^n = \bigoplus_{i=1}^{+\infty} a^i,$$

avec

$$a^* = e \oplus a^+, \quad a^+ = aa^*.$$

Définition 6 (Isotonie) *Une application f d'un dioïde $(\mathcal{D}, \oplus, \otimes)$ dans un autre dioïde $(\mathcal{C}, \oplus, \otimes)$ est dite isotone si :*

$$\forall a, b \in \mathcal{D}, a \succeq b \Rightarrow f(a) \succeq f(b),$$

autrement dit si pour tout $a, b \in \mathcal{D}$, on a :

$$f(a \oplus b) \succeq f(a) \oplus f(b).$$

Définition 7 (Continuité) *Une application f d'un dioïde $(\mathcal{D}, \oplus, \otimes)$ complet dans un autre dioïde $(\mathcal{C}, \oplus, \otimes)$ complet, est dit semi-continue inférieure, respectivement semi-continu supérieure si, pour tout sous-ensemble $\mathcal{B} \in \mathcal{D}$:*

$$f(\bigoplus_{x \in \mathcal{B}} x) = \bigoplus_{x \in \mathcal{B}} f(x),$$

respectivement,

$$f(\bigwedge_{x \in \mathcal{B}} x) = \bigwedge_{x \in \mathcal{B}} f(x).$$

Définition 8 (Homomorphisme de dioïdes) *Soient $\mathcal{D}$ et $\mathcal{C}$ deux dioïdes et $f : \mathcal{D} \mapsto \mathcal{C}$. L'application f est un homomorphisme si $\forall\ (a, b) \in \mathcal{D}^2$:*

$$f(a \oplus b) = f(a) \oplus f(b) \quad et \quad f(\varepsilon) = \varepsilon,$$

$$f(a \otimes b) = f(a) \otimes f(b) \quad et \quad f(e) = e.$$

1.1.2 Dioïdes matriciels

Soit $(\mathcal{D}, \oplus, \otimes)$ un dioïde, soit $A \in \mathcal{D}^{m \times p}$, $B \in \mathcal{D}^{p \times n}$, $C \in \mathcal{D}^{m \times n}$ à coefficients dans $\mathcal{D}$. La somme et le produit des matrices sont définis de la façon suivante :

$$A \oplus B : (A \oplus B)_{ij} = A_{ij} \oplus B_{ij},$$

$$A \otimes B : (A \otimes B)_{ij} = \bigoplus_{k=1}^{p} A_{ik} \otimes B_{kj}.$$

On peut montrer que l'ensemble $\mathcal{D}$ muni de ces deux opérations est un dioïde matriciel, dont l'élément nul noté ε, est la matrice composée exclusivement de ε. L'élément unité est la matrice notée Id_n composée de e sur la diagonale et de ε partout ailleurs.

Remarque 1 *: Même si la relation d'ordre est totale dans le dioïde $\mathcal{D}$, l'ordre dans le dioïde matriciel $\mathcal{D}^{n \times n}$ $(n \geqslant 1)$ est partiel.*

1.1.3 Résolution d'équation dans un dioïdes

On peut résoudre certaines équations dans l'algèbre des dioïdes, on va montrer ici deux types d'équations :

l'équation $x = Ax \oplus B$ où x est l'inconnue,

l'équation $f(x) = b$ où x est l'inconnue et f satisfait les propriétés d'isotonie et de continuité.

Résolution de l'équation $x = ax \oplus b$ ***:***
Soit $\mathcal{D}$ un dioïde complet, l'équation :

$$x = ax \oplus b, \tag{1.1}$$

définie dans $\mathcal{D}$ admet $x = a^*b$ comme plus petite solution.

Preuve : Tout d'abord il faut montrer que $x = a^*b$ est une solution de l'équation (1.1) :

$$a(a^*b) \oplus b = a^+b \oplus b = (a^+ \oplus e)b = a^*b,$$

et puis on montre que a^*b est le minorant des solutions de (1.1) :

$$\begin{aligned} x = ax \oplus b = a(ax \oplus b) \oplus b &= a^2x \oplus ab \oplus b \\ &= a^3x \oplus a^2b \oplus ab \oplus b \\ &\vdots \\ &= a^nx \oplus (a^{n-1}b \oplus ... \oplus ab \oplus b) \\ &\succeq a^{n-1}b \oplus ... \oplus ab \oplus b \qquad (n \geqslant 1) \end{aligned}$$

on en déduit que toute solution de (1.1) est supérieure ou égale à a^*b. De la même façon on peut montrer que l'équation $x = xa \oplus b$ admet ba^* comme plus petite solution.

Résolution de l'équation $f(x) = b$ ***:*** Le problème qui se pose ici est l'inversion de l'application f dans l'équation

$$f(x) = b, \tag{1.2}$$

car les lois $\oplus$ *et* $\otimes$ ne sont pas toujours inversibles, ce qui rend les application non inversibles en général dans un dioïde.

Néanmoins, la théorie de la résiduation permit de définir des "pseudo-inverses" pour des application définies sur des treillis, et plus particulièrement sur des dioïdes.

1.1.4 Théorie de la résiduation

cette théorie permet d'établir, lorsqu'elle existe, la plus grande solution de l'inéquation

$$f(x) \preceq b. \tag{1.3}$$

La théorie de la résiduation fournit la plus grande solution de l'équation (1.3), cette solution n'est que la borne supérieure de l'ensemble $\{x \in \mathcal{D} \mid f(x) \preceq b\}$. On note cette solution $f^\sharp(b)$, $f^\sharp$ sera appelée l'application résiduée de f.

Définition 9 (Applications résiduables) *Une application $f : (\mathcal{D}, \oplus, \otimes) \to (\mathcal{C}, \oplus, \otimes)$ avec une relation d'ordre $\preceq$ définie sur ces deux dioïde est dite résiduable si l'équation (1.3) admet une plus grande solution dans $\mathcal{D}$ pour tout $b \in \mathcal{C}$.*

Notation 3 *On va utiliser la notation $(\mathcal{D}, \preceq)$ pour un dioïde $(\mathcal{D}, \oplus, \otimes)$ avec une relation d'ordre $\preceq$.*

Théorème 2 *[Baccelli et al, 2001] Soit $f : (\mathcal{D}, \preceq) \to (\mathcal{C}, \preceq)$ une application isotone, les deux notions suivantes sont équivalentes :*

(i) pour tout $b \in \mathcal{C}$, l'équation admet une plus grande sous solution de l'équation $f(x) = b$,

(ii) $f(\varepsilon) = \varepsilon$ *et* f *est semi-continue intérieurement,*

(iii) il exist une application isotone $f^\sharp : \mathcal{C} \to \mathcal{D}$ *telle que :*

$$f \circ f^\sharp \preceq Id_{\mathcal{C}},$$

$$f^\sharp \circ f \succeq Id_{\mathcal{D}}.$$

En conséquence $f^\sharp$ *est unique, si* f *satisfait ces propriétés, on dit que* f *est résiduable et* $f^\sharp$ *est appelée la résiduée de* f*.*

La résiduation de l'addition et de la multiplication : Soit $\mathrm{T}_a, \mathrm{L}_a, \mathrm{R}_a$ trois applications d'un dioïde $\mathcal{D}$ à lui même, soient :

$$T_a : x \to a \oplus x \text{ (translation par a)},$$

$$L_a : x \to a \otimes x \text{ (multiplication par a à gauch)},$$

$$R_a : x \to x \otimes a \text{ (multiplication par a à droite)}.$$

On a :

$$T_a \circ T_b = T_b \circ T_a = T_{a \oplus b} = T_a \oplus T_b,$$

puisque $\oplus$ est associative, on a :

$$L_a \circ L_b = L_{ab},$$

et de plus :

$$L_a \circ R_b = R_b \circ L_a.$$

La multiplication est distributive par rapport aux sommes infinies, ce qui implique que :

$$L_a(\bigoplus_{x \in \mathcal{D}} x) = \bigoplus_{x \in \mathcal{D}} L_a(x),$$

de même pour R_a. L_a et R_a sont donc semi-continues inférieure, avec $L_a(\varepsilon) = \varepsilon$ *et* $R_a(\varepsilon) = \varepsilon$, d'après le théorème 2, les applications L_a et R_a sont résiduables et leur applications résiduées sont notées respectivement $L_a^\sharp et R_a^\sharp$:

$$L_a^\sharp(x) = a \backslash x,$$

$$R_a^\sharp(x) = x / a.$$

Extension de la résiduation au cas matriciel : Soient L_A et R_B deux applications définies sur les ensembles des matrices à coefficients dans le dioïde complet $(\mathcal{D}, \oplus, \otimes)$ comme suivant :

$$L_A : \mathcal{D}^{p \times q} \to \mathcal{D}^{n \times q} : X \mapsto AX \quad (X \mapsto A \otimes X) : A \in \mathcal{D}^{n \times p},$$

$$R_{A'} : \mathcal{D}^{q \times p} \to \mathcal{D}^{q \times n} : X \mapsto XA' \quad (X \mapsto X \otimes A') : A' \in \mathcal{D}^{p \times n}.$$

Les équations $AX = B$ et $XA' = B'$ admettent comme plus grandes sous-solutions respectivement :

$$L_A^\sharp(B) = A \backslash B \quad et \quad L_{A'}^\sharp(B') = B' / A',$$

les coefficients de ces matrices se calculent de la façon suivante :

$$(A \backslash B)_{ij} = \bigwedge_{l=1}^{n} A_{li} \backslash B_{lj}, \qquad i = 1, ..., p \quad j = 1, ..., q,$$

$$(B' / A')_{ij} = \bigwedge_{l=1}^{n} B'_{li} / A'_{lj}, \qquad i = 1, ..., q \quad j = 1, ..., p.$$

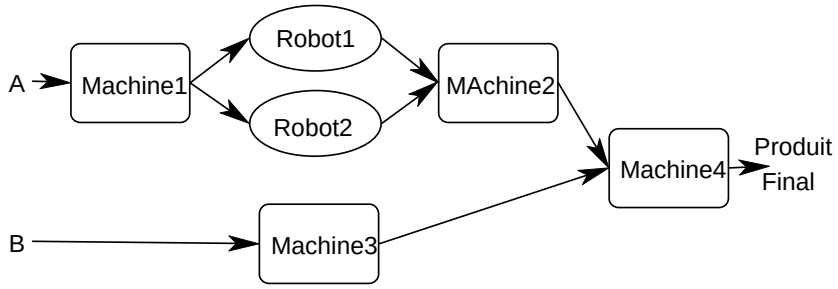


FIG. 1.1 – Un système de production

1.2 Modélisation avec les réseaux de Petri

1.2.1 Réseaux de Petri

Les réseaux de Petri sont des outils de modélisation graphiques et mathématiques efficaces pour plusieurs applications, ils ont une grande importance pour la représentation des systèmes de traitement d'information, de phénomènes de synchronisation et de concurrence [David and Alla, 1989], [Murata, 1989], Les RdP sont aussi des outils mathématique pouvent établir des équations d'état, des équations algébriques, et autres modèles mathématiques décrivant le fonctionnement des systèmes.

Les réseaux de Petri sont un cas particulier des graphes orientés, avec un état initial appelé *le marquage initial* (M_0), ils sont constitué de deux types de noueds, appelés *places et transition*, et les arcs sont orientés. Les places sont représentées par des cercles, les transitions sont représentées par des bars ou des rectangles, et les arcs sont marqués par des poids.

Le changement du marquage dans un réseau de Petri est effectué selon les règles de franchissement suivantes :

1 on dit q'une transition est valide si toute place d'entrée est marquée par un nombre de jetons plus grand ou égal au poids de l'arc connectant la place à cette transition.

2 le franchissement d'une transition valide, dépend du temps de démarrage d'un événement.

3 le franchissement d'une transition enlève des jetons de chaque place d'entrée selon les poids des arcs venant de ces places, et rajoute des jetons à chaque place de sortie selon les poids des arcs allant à ces places.

Pour pouvoir étudier la dynamique des systèmes modélisés par les réseaux de Petri, on associe à chaque transition x un opérateur appelé dateur $x(k)$ qui désigne la date du tir d'un événement numéro k, les dateurs sont des applications croissantes définies sur les domaines événementiels (ensemble $\mathbb{Z}$) vers le dioïde $\overline{\mathbb{R}}_{max}$ ou $\overline{\mathbb{Z}}_{max}$.

1.2.2 Représentation avec les réseaux de Petri

On va illustrer ici une application de réseaux de Petri (RdP) en utilisant l'exemple d'un système de fabrication automatisé présenté en figure 1.1 [Kurkovsky and Loganantharaj,].

Le système se compose de quatre machines (Machine 1, Machine 2, Machine 3 et Machine 4) et de deux robots (Robot 1 et Robot 2), ce système traite deux types de pièces (A et B). La séquence de traitement de la pièce A commence par la Machine 1 et puis par la Machine 2, la pièce B se traite seulement sur la Machine 3, la Machine 4 prend deux pièces A et une pièce B, elle fait l'assemblage de ces pièces pour arriver au produit final, Robot 1 et Robot 2 font les opérations de recharge et de décharge entre les Machines 1 et 2, la figure 1.2 montre la représentation en réseaux de Petri [Kurkovsky and Loganantharaj,].

La disponibilité des pièces A et B est représentée par les jetons dans les places p_1 *et* p_5 respectivement, le traitement sur la Machine 1 est représenté par la transition t_1 et son produit est représenté par un jeton dans la place p_2. Les transitions t_2 *et* t_3 sont valides quand un jeton au moins existe dans la place p_2.

Nous pouvons associer les temps de traitement des pièces sur chaque machine, en associant à chaque machines un délai, les temps d'attente des pièces pendant la fabrication peuvent être associés aux places

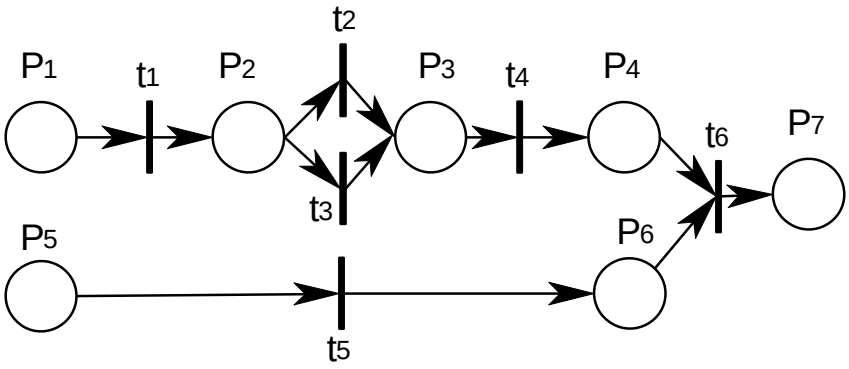


FIG. 1.2 – Une représentation de la système de production Fig 1.1 par les réseaux de Petri

qui ne peuvent contribuer à la validation des transitions avant que le temps d'attente associé à cette place s'écoule, comme on peut associer le temps nécessaire pour qu'un jeton se transporte entre une transition et une place à l'arc qui connecte la transition et la place intendue, avec ces associations temporelles on peut créer des relations de temporisation modélisantes le fonctionnement dynamique des systèmes représentés par RdP dans la figure 1.2.

Il y a deux manières pour effectuer cette modélisation temporelle, la première avec l'utilisation des équations où la variable est le nombre de franchissement des transition à la date t, c'est-à-dire, on associe à chaque transition un compteur pour compter le nombre des fois que cette transition a été franchie. Une deuxième manière de modélisation est d'associer à chacune des transitions une fonction dateur qui enregistre la date l'occurrence du franchissement des transitions.

Dans la suite on va parler de ces deux manières de modélisation des graphes d'événement temporisés (RdP où chaque place a une transition amont et une transition aval) car, on peut les représenter par un modèle linéaire sous forme canonique dans l'algèbres $(max, +)$.

1.2.3 Modèle en équations aux *dateurs* (domaine événmentiel)

Un dateur $x(.)$ est une fonction associée à la transition x, et $x(k)$ désigne la date de l'activation numéro k de cette transition x. Soit la figure 1.3, qui représente un système de fabrication de deux machines (M_1, M_2).

Forme implicite : En suivant les règles de franchissements décrites avant en 1.2.1, les équations aux dateurs modélisant ce système seront :

$$\begin{array}{rcl} x_1(k) & = & max[x_2(k-1), u_2(k)], \\ x_2(k) & = & x_1(k)+1, \\ x_3(k) & = & max[x_2(k), x_4(k-1), u_1(k)+2], \\ x_4(k) & = & x_3(k)+2, \\ Y(k) & = & x_4(k)+3. \end{array}$$

Il s'agit d'un ensemble d'équations linéaires dans l'algèbre $\overline{\mathbb{Z}}_{max}$ de la forme ***ARMA***[1] générale suivante :

$$X(k) = \underbrace{\bigoplus_{i=1}^{n} A_i \; X(k-i)}_{AR} \; \oplus \; \underbrace{\bigoplus_{l=1}^{m} B_l \; U(k-l)}_{MA} . \tag{1.4}$$

[1]ARMA pour "Auto Regressive - Moving Average" et en français "Auto Régressive - Moyenne Mobile".

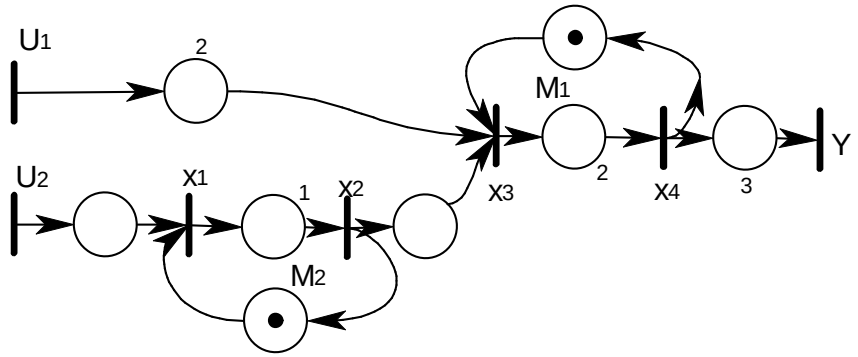


FIG. 1.3 – Réseau de Petri d'un système de production.

Les équations aux dateurs sont :

$$X(k) = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 1 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 2 & \varepsilon \end{pmatrix} X(k) \oplus \begin{pmatrix} \varepsilon & e & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & e \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{pmatrix} X(k-1) \oplus \begin{pmatrix} \varepsilon & e \\ \varepsilon & \varepsilon \\ 2 & \varepsilon \\ \varepsilon & \varepsilon \end{pmatrix} U(k),$$

$$Y(k) = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & 3 \end{pmatrix} X(k),$$

avec :

$$X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}; \quad et \quad U = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}.$$

Ce système matriciel peut s'écrire sous la forme suivante :

$$\begin{cases} X(k) &= A_0X(k) \oplus A_1X(k-1) \oplus B_0U(k), \\ Y(k) &= C_0X(k). \end{cases} \tag{1.5}$$

cette équation est une forme implicite en X.

Forme *ARMA* explicite (suppression de la forme implicite) : On peut passer de la forme implicite de l'équation (1.5) à la forme ARMA explicite suivante :

$$\begin{cases} X(k) &= AX(k-1) \oplus BU(k), \\ Y(k) &= CX(k). \end{cases} \tag{1.6}$$

avec $A = A_0^*A_1$ $\quad et \quad$ $B = A_0^*B$. Pour l'exemple précédent, on obtient :

$$A_0 = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 1 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 2 & \varepsilon \end{pmatrix} \Rightarrow A_0^* = \begin{pmatrix} e & \varepsilon & \varepsilon & \varepsilon \\ 1 & e & \varepsilon & \varepsilon \\ e & e & e & \varepsilon \\ 2 & 2 & 2 & e \end{pmatrix},$$

soient :

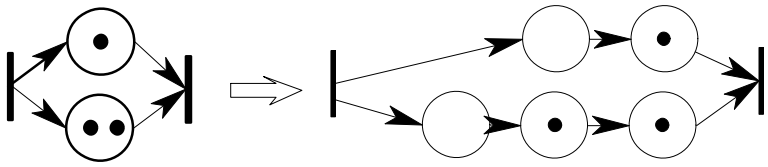
$$A = \begin{pmatrix} \varepsilon & e & \varepsilon & \varepsilon \\ \varepsilon & 1 & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon & e \\ \varepsilon & 2 & \varepsilon & 2 \end{pmatrix}, \qquad B = \begin{pmatrix} \varepsilon & e \\ \varepsilon & 1 \\ 2 & e \\ 4 & 2 \end{pmatrix} \qquad et \qquad C = C_0 = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & 3 \end{pmatrix}.$$

Forme récurrente *"Markovienne" ou forme d'état* : il s'agit de réduire le retard à 1 sur la partie AR, et à 0 sur la partie MA dans l'équation (1.4), par une extension du vecteur d'état [Cohen, 1995], qui nous ramène à un système de la forme :

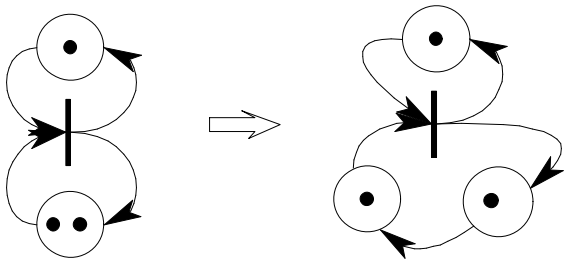
$$X(k) = A_0X(k) \oplus A_1X(k-1) \oplus B_0U(k),$$

$$Y(k) = C_0X(k).$$

ce qui peut être interprété en pratique en étendant le graphe afin que chaque place ne contient initialement qu'une seul jeton (figure 1.4).



Réduction d'une partie MA



Réduction d'une partie AR

FIG. 1.4 – Réduction des retards dans la forme Markovienne

1.2.4 Modèle en équations aux *compteurs* (domaine temporel)

Un compteur $x(t)$ est une fonction $t \rightarrow x(t)$ associée à la transition x, qui indique le numéro de la dernière activation de cette transition x survenue avant la date t. Cette fonction est la fonction duale de la fonction dateur (monotone non décroissante) [Cohen, 1995]. Pour préserver la linéarité des équations en passant par la fonction réciproque du dateur, on se place dans une autre algèbre, l'algèbre $\overline{\mathbb{Z}}_{min}$.

Forme implicite : Les équations aux compteurs décrivant le système représenté dans la figure 1.3 sont :

$$\begin{array}{rcl} x_1(t) & = & min[1 + x_2(t), u_2(t)], \\ x_2(t) & = & x_1(t-1), \\ x_3(t) & = & min[x_2(t), 1 + x_4(t), u_1(t-2)], \\ x_4(t) & = & x_3(t-2), \\ Y(t) & = & x_4(t-3), \end{array}$$

ce qui donne la notation matricielle suivante :

$$X(t) = \begin{pmatrix} \varepsilon & 1 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon & 1 \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{pmatrix} X(t) \oplus \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ e & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{pmatrix} X(t-1) \oplus \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & e & \varepsilon \end{pmatrix} X(t-2)$$

$$\oplus \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{pmatrix} X(t-3) \oplus \begin{pmatrix} \varepsilon & e \\ \varepsilon & \varepsilon \\ \varepsilon & \varepsilon \\ \varepsilon & \varepsilon \end{pmatrix} U(t) \oplus \begin{pmatrix} \varepsilon & \varepsilon \\ \varepsilon & \varepsilon \\ e & \varepsilon \\ \varepsilon & \varepsilon \end{pmatrix} U(t-2),$$

$$Y(t) = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & e \end{pmatrix} X(t-3).$$

soit :

$$\left\{ \begin{array}{rcl} X(t) & = & A_0\ X(t) + A_1\ X(t-1) + A_2\ X(t-2) + A_3\ X(t-3) + B_0\ U(t) + B_2\ U(t-2), \\ Y(t) & = & C_3 X(t-3). \end{array} \right. \tag{1.7}$$

où :

$$X(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{pmatrix} \quad et \quad U(t) = \begin{pmatrix} u_1(t) \\ u_2(t) \end{pmatrix}.$$

qui est de la forme ARMA dans l'équation (1.4) en remplaçant k par t et sachant que les calculs matriciels s'effectuent dans l'algèbre $\overline{\mathbb{Z}}_{min}$.

Forme explicite : On obtient la forme explicite en résolvant la forme implicite et en sélectionnant la plus petite solution au sens de l'algèbre (dans l'algèbre $\overline{\mathbb{Z}}_{min}$ $a = a \oplus b \Leftrightarrow a = min(a, b)$), ce qui donne la forme explicite suivante :

$$\begin{cases} X(t) &= A_0^* \; A_2 \; X(t-2) + A_0^* \; A_3 \; X(t-3) + A_0^* \; B_0 \; U(t) + A_0^* \; B_2 \; U(t-2), \\ Y(t) &= C_3 \; X(t-3). \end{cases}$$

Forme récurrente : Comme precédament en étendant le graphe afin que chaque place ne contient initialement qu'un seul jeton, nous obtenons :

$$X(t) = AX(t-1) \oplus BU(t),$$

$$Y(t) = CX(t).$$

1.2.5 Modèle exprimé sous forme de *séries formelles*

Il est possible de modéliser les GET *via* des séries formelles en γ dans le dioïde $\overline{\mathbb{Z}}_{max}$ et en δ dans le dioïde $\overline{\mathbb{Z}}_{min}$, ce qui aboutit à une représentation bidimentionnelle dans le dioïde $\mathcal{M}_{in}^{ax}[\gamma, \delta]$ avec ces deux variables commutatives, γ et δ à exposants dans $\mathbb{Z}$ [Cohen,1995].

Transformée en γ (Dioïde $\overline{\mathbb{Z}}_{max}$) : Pour un dateur X(.), on définit la série formelle en γ comme suit :

$$X(\gamma) = \bigoplus_{k \in \mathbb{Z}} X(k) \otimes \gamma^k,$$

nous avons :

$$\gamma \otimes X(\gamma) = \bigoplus_{k \in \mathbb{Z}} X(k) \otimes \gamma^{k+1} = \bigoplus_{k \in \mathbb{Z}} X(k-1) \otimes \gamma^k,$$

γ s'assimile à un opérateur de retard, *i.e.*, $X(k-1) = \gamma \; X(k)$.

Définition 10 (Dioïde $\overline{\mathbb{Z}}_{max}$ $[\gamma]$) *L'ensemble des séries formelles en γ à exposants dans $\mathbb{Z}$ et à coefficients dans $\overline{\mathbb{Z}}_{max}$ a une structure de dioïde, pour élément neutre de l'addition, $\varepsilon(\gamma) = \bigoplus_{k \in \mathbb{Z}} \varepsilon \gamma^k$ $(\varepsilon = -\infty)$ et pour élément neutre de la multiplication, $e(\gamma) = e\gamma^0 (e = 0)$. Dans ce dioïde la somme et le produit sont définis en séries formelles en γ comme suit :*

$$X_1(\gamma) \oplus X_2(\gamma) = \bigoplus_{k \in \mathbb{Z}} \left[X_1(k) \oplus X_2(k) \right] \gamma^k,$$

$$X_1(\gamma) \otimes X_2(\gamma) = \bigoplus_{k \in \mathbb{Z}} \bigotimes_{j \in \mathbb{Z}} \left[X_1(j) \oplus X_2(k-j) \right] \gamma^k.$$

On peut donc reformuler le système d'équations (1.6) de la façon suivante :

$$\begin{cases} X(\gamma) &= \gamma \; A \; X(\gamma) \oplus B \; U(\gamma), \\ Y(\gamma) &= \quad C \; X(\gamma). \end{cases} \tag{1.8}$$

En remplaçant $X(\gamma)$ dans $Y(\gamma)$, et en appliquant la théorie de la résiduation, on obtient :

$$Y(\gamma) = C(\gamma A)^* B\ U(\gamma) = H\ U(\gamma), \tag{1.9}$$

qui est une relation de transfert caractérisant le comportement entrée/sortie du système où H est *la matrice de transfert.*

Remarque 2 *: Dans un GET les trajectoires sont monotones croissantes, si X(.) un dateur d'un GET (la date d'occurrence de X(k) est supérieur à X(k-1)) donc :*

$$\forall k \in \mathbb{Z}\ :\ X(k) \succeq X(k-1) \Leftrightarrow X(k) = X(k) \oplus X(k-1),$$

qui peut être formulé avec γ par :

$$X(\gamma) \geqslant \gamma X(\gamma) \Leftrightarrow X(\gamma) = \gamma^* X(\gamma).$$

d'où la transformée en γ d'un trajectoire monotone $\gamma^* X(\gamma)$, et la multiplication par γ^* d'un trajectoire non monotone donne une trajectoire monotone croissant.

Remarque 3 *: L'ensemble des trajectoires monotones forme un dioïde, noté $\gamma^* \overline{\mathbb{Z}}_{max}[\gamma]$ telque :*
- *$\gamma^n \oplus \gamma^m = \gamma^{min(n,m)}$,*
- *l'élément neutre pour l'addition est $\gamma^* \varepsilon(\gamma) = \varepsilon \oplus \varepsilon\gamma \oplus \varepsilon\gamma^2 \oplus ... \oplus \varepsilon\gamma^{+\infty}$,*
- *l'élément neutre pour la multiplication est $\gamma^* e(\gamma) = e \oplus e\gamma \oplus e\gamma^2 \oplus ... \oplus e\gamma^{+\infty}$.*

Transformée en δ (Dioïde $\overline{\mathbb{Z}}_{min}$) : De la même manière, on définit la transformée en série formelle en δ pour un compteur X(.) comme suit :

$$X(\delta) = \bigoplus_{t \in \mathbb{Z}} X(t) \otimes \delta^t.$$

Définition 11 (Dioïde $\overline{\mathbb{Z}}_{min}$ [δ) / *L'ensemble des séries formelles en δ à exposants dans $\mathbb{Z}$ et à coefficients dans $\overline{\mathbb{Z}}_{min}$ a une structure de dioïde, avec pour élément neutre de l'addition, $\varepsilon(\delta) = \bigoplus_{t \in \mathbb{Z}} \varepsilon\delta^t$ $(\varepsilon = +\infty)$ et pour élément neutre de la multiplication, $e(\delta) = e\delta^0 (e = 0)$. Dans ce dioïde la somme et le produit sont définis en séries formelles en δ comme suivant :*

$$X_1(\delta) \oplus X_2(\delta) = \bigoplus_{t \in \mathbb{Z}} [X_1(t) \oplus X_2(t)]\ \delta^t,$$

$$X_1(\delta) \otimes X_2(\delta) = \bigoplus_{t \in \mathbb{Z}} \bigotimes_{j \in \mathbb{Z}} [X_1(j) \oplus X_2(t-j)]\ \delta^t,$$

nous pouvons donc écrire le système d'équations (1.8) en fonction de la variable δ :

$$\begin{cases} X(\delta) &= \delta\ A\ X(\delta) \oplus B\ U(\delta), \\ Y(\delta) &= C\ X(\delta). \end{cases} \tag{1.10}$$

Il en résulte :

$$Y(\delta) = C(\delta A)^* B\ U(\delta) = H\ U(\delta) \tag{1.11}$$

où H est appelé la matrice de transfert d'entrée/sortie du système en δ.

Remarque 4 *: Si X(.) est un compteur d'un GET (le nombre d'événements à l'instant t+1 supérieur au nombre d'événements à l'instant t,* i.e. *:*

$$\forall t \in \mathbb{Z}\ :\ X(t) \succeq X(t+1) \Leftrightarrow X(t) = X(t+1) \oplus X(t),$$

qui peut se transformer avec δ en :

$$X(\delta) = \delta^{-1} X(\delta) \oplus X(\delta) \Leftrightarrow X(\delta) = (\delta^{-1})^* X(\gamma),$$

d'où la forme générale d'un trajectoire monotone en δ exprimé par la relation $(\delta^{-1})^* X(\delta)$.

Remarque 5 *: Comme dans la remarque 3, l'ensemble des trajectoires monotones dans le dioïde* $\overline{\mathbb{Z}}_{min}$ *forme un dioïde noté* $\delta^*\overline{\mathbb{Z}}_{max}[\delta]$ *telque :*
- $\delta^t \oplus \delta^r = \delta^{max(t,r)}$,
- *l'élément neutre pour l'addition est* $\delta^*\varepsilon(\delta) = \varepsilon \oplus \varepsilon\delta \oplus \varepsilon\delta^2 \oplus ... \oplus \varepsilon\gamma^{+\infty}$ *avec* $\varepsilon = +\infty$,
- *l'élément neutre pour la multiplication est* $\delta^* e(\delta) = e \oplus e\delta \oplus e\delta^2 \oplus ... \oplus e\delta^{+\infty}$.

Modélisation Bi-dimentionnelle en γ ***et*** δ ***sur*** $\mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]$: La modélisation bi-dimentionnelle permet de prendre en compte simultanément les deux domaines (événementiel, temporel), ce qui aboutit à modéliser un GET *via* une seule représentation formelle à deux variables (γ *et* δ).

$\mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]$ est un dioïde complet dont les éléments sont désignés par des séries en (γ *et* δ) avec des coefficients dans $\mathbb{Z}$. La manipulation de ces éléments se fait selon les règles de somme et de produit * (précédement définies), avec les règles de simplification suivantes :
- $\gamma^n\ \delta^t \oplus \gamma^m\ \delta^t = \gamma^{max(n,m)}\ \delta^t$,
- $\gamma^n\ \delta^t \oplus \gamma^n\ \delta^r = \gamma^n\ \delta^{min(n,r)}$,

Dans la suite et pour résoudre les problèmes dans ce dioïde, nous allons utiliser le programme *SCILAB*[2] où les opérations (somme et produit) et la manipulation graphique sont déjà programmés.

On va modéliser notre exemple de GET présenté dans la figure 1.3 dans le dioïde $\mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]$. commençons par transformer les décalages événementielles et temporelles en γ *et* δ pour obtenir les équations suivantes :

$$\begin{aligned} x_1 &= \gamma x_2 \oplus u_2, \\ x_2 &= \delta x_1, \\ x_3 &= x_2 \oplus \gamma x_4 \oplus \delta^2 u_1, \\ x_4 &= \delta^2 x_3, \\ y &= \delta^3 x_4, \end{aligned}$$

et sous une notation matricielle :

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} \varepsilon & \gamma & \varepsilon & \varepsilon \\ \delta & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon & \gamma \\ \varepsilon & \varepsilon & \delta^2 & \varepsilon \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \oplus \begin{pmatrix} \varepsilon & e \\ \varepsilon & \varepsilon \\ \delta^2 & \varepsilon \\ \varepsilon & \varepsilon \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix},$$

$$y = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \delta^3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}.$$

ce qui correspond à la forme standard suivante :

$$\begin{aligned} x &= A\ x \oplus B\ u, \\ y &= C\ x. \end{aligned}$$

1.3 Le système *Atelier à tâche (Job-Shop)*

Un système *Atelier à tâche* est un type spécifique de système de production composé de machines avec des tâches à effectuer sur ces machines. Chaque tâche (job) est composée de plusieurs opérations et sera effectuée par une procédure de visite à ces machines. Les temps consommés par les tâches sur ces machines sont aussi définis [Hillion and Proth, 1989].

Pour bien expliquer ce qu'est un système d'atelier à tâche, citon un exemple [Cohen, 1995], [Hillion and Proth, 1989] qui représente un système de production de trois machines (M_1, M_2, M_3) dans un usine qui produit trois types de pièces (tâches) (P_1, P_2, P_3). Chaque pièce doit subir un certain nombre d'opérations sur les machines dans un ordre précis avec un temps de traitement défini sur chaque machine, soient :

$$P_1 : M_1(1)^3 \rightarrow M_2(3) \rightarrow M_3(3),$$

[2] C'est un logiciel comme le *MATLAB* de plus il est muni des outils de l'algèbre maxplus
[3] le temps de traitement

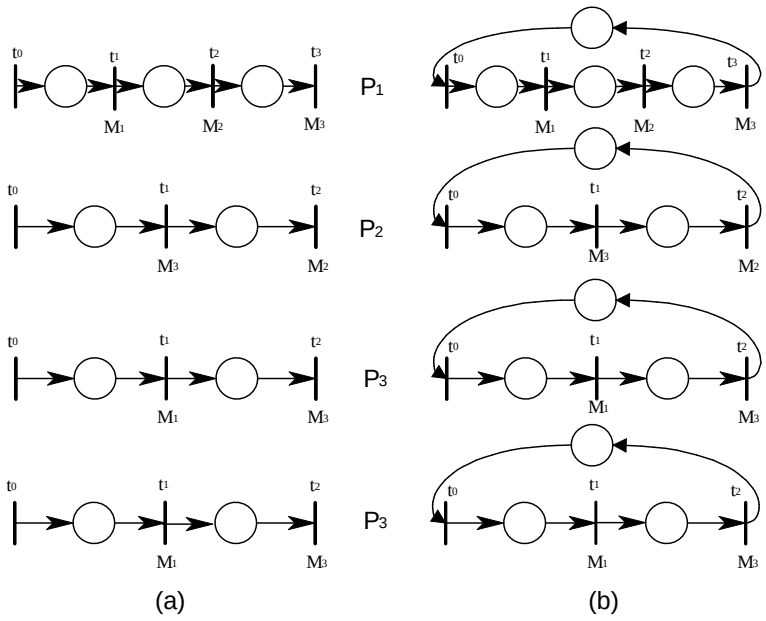


FIG. 1.5 – (a) Représentation des routes des tâches par des séries de transitions,

(b)Représentation de la circulation des tâches.

$$P_2 : M_3(1) \rightarrow M_2(2),$$
$$P_3 : M_1(2) \rightarrow M_3(1).$$

les séquences de traitement

Si on ajoute un ratio de fabrication $1/1/2$ pour les trois pièces, il faudra donc produire simultanément 25% de pièce P_1, 25% de pièce P_2 et 50% de pièce P_3, ce qui nous conduit à *la séquence complète de production* $P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_3$, on constate l'ordre de passage de cette séquence pour chaque machine est :

$$M_1 : P_1 \rightarrow P_3 \rightarrow P_3,$$
$$M_2 : P_1 \rightarrow P_2,$$
$$M_3 : P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_3.$$

les séquences de traitement

Une nouvelle pièce n'est introduite que lorsque la dernière opération pour la pièce en cours est accomplie. Les opérations de déchargement/chargement peuvent être représentées par un passage sur une machine avec une durée de passage si nécessaire.

1.3.1 Représentation des systèmes d'Atelier à tâches par les réseaux de Petri

Une représentation graphique des systèmes d'atelier à tâche, est celle basée sur les réseaux de Petri et est décrite par Guy Cohen, Hervé P. Hillion et Jean-Marie Proth. détaillons les étapes suivies.

Première étape : Ils représentent la route de chaque tâche dans la séquence complète de production par une série de transitions, (chaque transition représente une exécution d'une opération de cette tâche), figure 1.5.a, avec les durées d'exécution (t_i) sur les machines correspondantes (M_i). La première transition est une transition immédiate sans temps de passage pour charger une tâche au système, un jeton circulant représente une tâche, une place correspond à une place tampon (buffer place).

Deuxième étape (Circuit de traitement) : ce circuit représente le fonctionnement circulaire de la séquence de fabrication (traitement), en supposant que la tâche se répète au moment où elle sera accomplie. La figure 1.5.b représente le circuit de traitement comme une boucle fermée, et la place de fermeture est appelée une

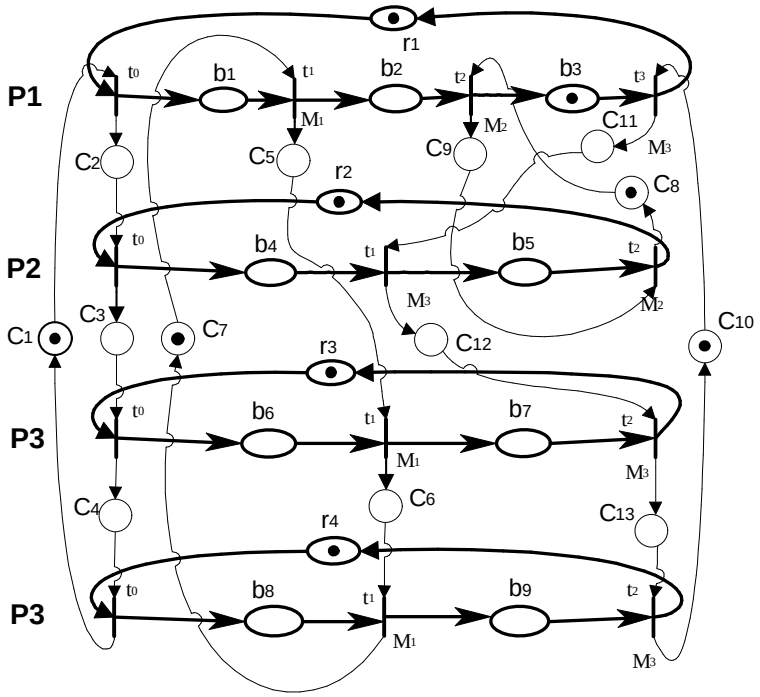


FIG. 1.6 – Représentation complète du système d'atelier.

ressource.

Troisième étape (Circuit de commande) : ce circuit représente la séquence qui commande les machines en exécutant les tâches (séquence de production). Ce circuit est vérifié par la connection des transitions qui correspondent aux opérations exécutées par la même machine (dans un circuit unique) en respectant l'ordre d'exécution des tâches sur cette machine.

Le jeton circulant dans un circuit de traitement représente une tâche à exécuter. Un jeton circulant dans le circuit de commande représente une condition (la disponibilité d'une machine pour exécuter une tâche), ce qui implique l'existante d'un seul jeton dans chaque circuit de commande, car, une machine n'exécute qu'une seule tâche à la fois.

La figure 1.6 montre la représentation complète du système d'atelier précédent, où une place tampon, une place de ressource et une place de commande sont marquées respectivement par (b, r et c). La position initiale dans un circuit de commande est donnée par la première tâche de la séquence. La distribution de jetons dans le circuit de traitement représente un état initial.

La représentation de l'atelier correspond à un graphe d'événements temporisé, dont les règles et les conditions s'appliquent ici. De plus, les réseaux représentant un atelier sont fortement connectés grâce aux circuits de commande qui relient les circuits de traitement entre eux. On peut donc utiliser les résultats obtenus par Commoner et *al*, dans l'ouvrage (Marked Directed Graphs, 1971) où l'on peut mentionner les notions suivantes :

. Le nombre total de jetons dans un circuit élémentaire[4] est constant. De même l'existence d'un jeton, dans chaque circuit évite les cas de blocages.

. A chaque circuit élémentaire γ, on associe le temps du circuit : $C(\gamma) = (\mu(\gamma)/M(\gamma))$ où $\mu(\gamma)$ représente le temps de franchissements des transitions du circuit γ et $M(\gamma)$ le nombre total des jetons dans ce circuit, le circuit qui a le plus grand temps sera appelé le *circuit critique*.

[4]Par définition un circuit élémentaire est un chemin dirigé d'une transition ou place vers cette transition ou cette place sans répétition des transitions ou places.

Problème du taux maximal de traitement [Hillion and Proth, 1989] : Dans la représentation des systèmes de l'atelier en réseau de Petri, nous avons vu qu'il y a une distribution finie des ressources (un marquage initial des places de ressources) qui permet aux tâches d'être traitées dès le début (des jetons dans les places tampons-buffers), ce qui conduit à l'utilisation du système d'atelier en cas de productivité maximale.

Preuve [Hillion and Proth, 1989] : Avec notre représentation par GET, la preuve est plus facile.
Soit π la durée maximale des circuits sur tous les circuits élémentaires, c'est-à-dire,

$$\pi = max\ _{\gamma}\ (C(\gamma)),$$

soit :

$$\pi = max(max\ _{\gamma_c}\ (C(\gamma_c)),\ max\ _{\gamma \neq \gamma_c}(C(\gamma))),$$

où γ_c représente un circuit de commande et $\gamma \neq \gamma_c$ représente les autres types de circuits (circuits de traitement et circuits mixtes[5]). Par définition $C_0 = max\ _{\gamma_c}(C(\gamma_c))$ représente le taux de l'utilisation maximale des machines, un circuit γ qui n'est pas un circuit de commande a au moins une place tampon ou une place de ressource, en ajoutant un nombre suffisant et limité de jetons dans cette place on pourra toujours atteindre $C(\gamma) < C_0$.

Hillion et Proth prouvent aussi, que la séquence des tâches peut se traiter en performance maximal si on démarre d'un marquage initial où chaque place tampon et chaque place ressource contient exactement un jeton.

Preuve : Soit $\gamma = (p_1 t_1 p_2, ..., p_n t_n)$ est un circuit mais pas de type de circuit de commande où p_1 n'est pas une place de commande, on note par $\mu(t_i)$ le temps de franchissement de la transition t_i et par $\varphi(t_i)$ la machine exécutante l'opération correspondante, on peut alors diviser le circuit γ comme suivant :

$$\gamma = (p_{s(1)}, \gamma_1, p_{s(2)}, ..., p_{s(k)}, \gamma_k)$$

γ_i et $s(k)$ sont définis itérativement comme suivant :

$$s(1) = 1, \quad s(i+1) = s(i) + j \qquad\qquad i = 1, 2, ..., k$$

$$\gamma_i = (t_{s(i)}\ p_{s(i)+1} ... p_{s(i)+j-1}\ t_{s(i)+j-1}),$$

où j est déterminé uniquement par :

$$\varphi(t_{s(i)}) = \varphi(t_{s(i)+1}) = ... = \varphi(t_{s(i)+j-1}),$$

$$\varphi(t_{s(i)}) \neq \varphi(t_{s(i)+j-1}).$$

Autrement dit, γ_i (pour i = 1,...,k) représente un chemin (d'une seule transition) dans le même circuit de commande, où $p_{s(i)}$ soit une place tampon ou une place de ressource.
Si on définit $\mu(\gamma_i)$ comme la somme des temps des transitions appartenant au chemin (γ_i), on aura pour un circuit γ :

$$\mu(\gamma) = \sum_{i=1}^{k} \mu(\gamma_i),$$

Tant que les transitions de (γ_i) sont des transitions de même circuit de commande, et tant que le contenu de n'importe quel circuit de commande est un jeton, nous avons pour $i = 1, 2, ..., k$:

$$\mu(\gamma_i) \leqslant C_0 = max\ _{\gamma_c}(C(\gamma)),$$

d'où :

$$\mu(\gamma) \leqslant kC_0. \tag{1.12}$$

Prenons en compte que chaque place tampon et chaque place de ressource a exactement un jeton, alors :

$$M(\gamma) \geqslant \sum_{i=1}^{k} M(p_{s(i)}) = k, \tag{1.13}$$

[5]des circuits qui ont des noeds partagés en même temps avec les circuits de traitement et les circuits de commande.

finalement, pour tout $\gamma \neq \gamma_c$, les deux équations 1.18 et 1.19 donnent :

$$C(\gamma) = (\mu(\gamma)/M(\gamma)) \leqslant C_0, \tag{1.14}$$

ce qui confirme aussi que ce taux maximal de traitement des tâches est verifiable.

Ces derniers résultats sont intéressants par ce que, d'une part ils munissent un état initial[6] conduisant au taux maximal de traitement de tâches, et d'autre part, ils donnent le nombre nécessaire de ressources pour achever le taux maximal de production.

Problème de minimisation des tâches en traitement $(Pmtt)$: Soit C_0 le temps du circuit pour un taux de traitement maximal, on va utiliser la notation $M_{min}(\gamma)$ comme le nombre minimal de jetons dans γ avec $C(\gamma) \leqslant C_0$ et la notation M_0 pour le marquage initial, d'après l'analyse précédente, la condition suivante est nécessaire et suffisante pour assurer l'utilisation du taux de traitement maximal :

$$M_0(\gamma) \geqslant M_{min}(\gamma) \quad (\forall\ \gamma \text{ un circuit élémentaire}). \tag{1.15}$$

Alors, si u est le nombre totale des places dans une représentation de système d'atelier en réseau de Petri, nous pouvons formuler le problème de minimisation de tâches pour un taux maximal de traitement, comme suivant :

$$\underbrace{\left[\sum_{j=1}^{u} M_0(p_j) \text{ minimisable si } \ M_0(\gamma) \geqslant M_{min}(\gamma)\right]}_{Pmtt1} \quad (\forall\gamma \text{ un circuit élémentaire}),$$

ce qui signifie la minimisation du nombre total de jetons dans le réseau. Soit la matrice $D = [d_{ij}]_{s\times u}$ tel que :

- u est le nombre total des places $(p_1, p_2, ..., p_u)$,
- s est le nombre total des circuits élémentaires $(\gamma_1, \gamma_2, ..., \gamma_s)$,
- D est la matrice d'incidence des circuits définie par :

$$d_{ij} = \begin{cases} 1 \ si \ p_j \ \in \ \gamma_i, \\ 0 \ sinon. \end{cases}$$

Soit $x_j = M_0(p_j) \quad (j = 1, ..., u)$ et $a_i = M_{min}(\gamma_i) \quad (i = 1, ..., s)$, on note $X = [x_1, x_2, ..., x_u]^t$ et $A = [a_1, a_2, ..., a_s]^t$ alors, le problème $Pmtt1$ peut se formuler en un problème $Pmtt2$ comme suit :

$$\underbrace{[min(x_1 + x_2 + ... + x_u) \ \text{ soumis à } \ DX \geqslant A]}_{Pmtt2} \ \ et \begin{cases} p_j \text{ une place de commande,} \\ x_j = 1 \ \ si \ M_0(p_j) = 1, \\ x_j = 0 \ \ si \ M_0(p_j) = 0. \end{cases}$$

Les deux problèmes $Pmtt1$ *et* $Pmtt2$ sont des problèmes de programmation linéaire en nombres entiers, on obtient leurs solutions optimales en utilisant des algorithmes heuristiques.

1.3.2 La représentation des systèmes d'Atelier à tâches en graphes disjonctives [Brucker,2001]

Grâce à leur structure, les graphes disjonctives peuvent être utilisés pour représenter certains systèmes d'atelier.
Un graphe disjonctif G est un triplet $G = (V, C, D)$ où :

V est l'ensemble des noeuds qui représente les opérations de toutes les tâches, il y a deux noeuds spéciaux, un noeud source $0 \in V$, et un noeud destination $* \in V$. On associe à chaque noeud un poids représentant le temps de traitement ceux des noeuds 0 et * sont nuls.

C est l'ensemble des arcs dirigés et conjonctifs qui représentent les relations de priorité entre les opérations et tâches. Il existe des arcs conjonctifs reliant la source aux opérations suivantes sans prédécesseur, et des arcs conjonctifs reliants les opérations à la destination sans successeur.

D est l'ensemble des arcs non dirigés et disjonctifs, comme un arc entre deux opérations appartenant à la même tâche et non connectées par une chaîne des arcs conjonctifs, ou comme un arc entre deux opérations traitées sur la même machine et qui ne sont pas connectées par une chaîne des arcs conjonctifs.

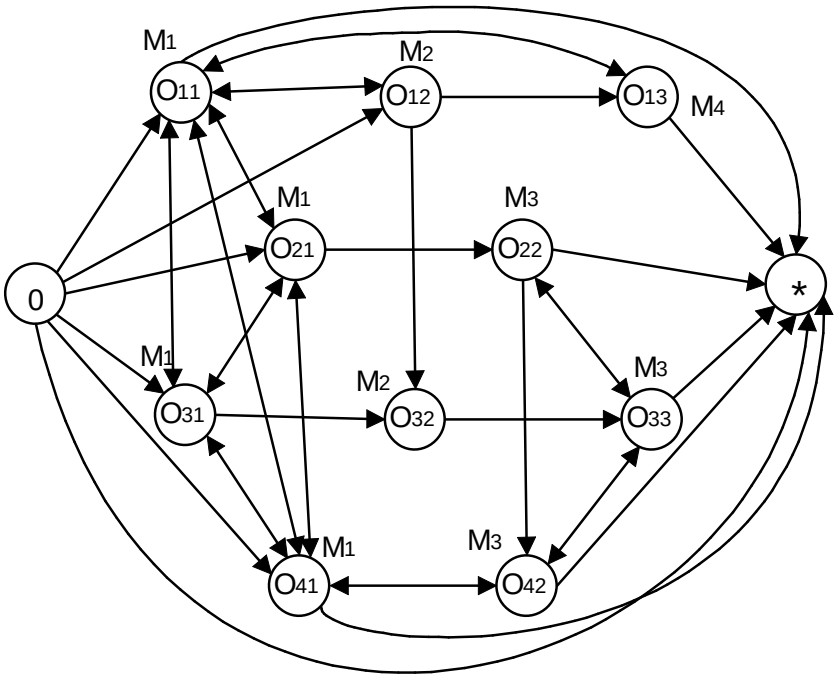


FIG. 1.7 – graphe disjonctif pour un système d'atelier général.

La figure 1.7 montre un graphe disjonctive pour un système d'atelier général de 4 machines à 4 tâches. Chaque noeud est marqué par deux étiquette, une représentant l'opération et l'autre représentant la machine sur la quelle cette opération sera effectuée.

Problème du taux maximal du traitement : On va introduire ici la notion du temps nécessaire pour finir une tâche C (finishing time), ce qui signifie que chaque tâche J_i a besoin du temps C_i pour l'effectuer. Ceci nous aide à construire une fonction du coût du traitement d'une tâche J_i par $f_i(C_i)$. On peut ainsi définir la fonction du coût total de deux façon :

$$f_{max}(C) = MAX_i \ f_i(C_i) \qquad i = 1, ..., n, \tag{1.16}$$

$$\sum f_i(C) = \sum_{i=1}^{n} f_i(C). \tag{1.17}$$

Nous pouvons affecter à chaque tâche une priorité, ou une importance, qui sera représentée par des poids rajoutés dans la fonction coût : pour chaque tâche J_i, qui a un temps d'execution C_i nous exprimons sa priorité comme des poids w_i. la fonction coût devient :

$$\sum f_i(C_i) = \sum_{i=1}^{n} \ w_i \ f_i(C_i). \tag{1.18}$$

Il y a d'autres fonctions coût à base de C_i qui utilisent des notions différentes comme par exemple :

$L_i = C_i - d_i$	retard
$E_i = max\{0, d_i - C_i\}$	tôt
$T_i = max\{0, C_i - d_i\}$	lent
$D_i = \lvert C_i - d_i \rvert$	déviation absolue
$S_i = (C_i - d_i)^2$	déviation carrée
$U_i = \begin{cases} 0 \ si \ C_i \leqslant d_i \\ 1 \ sinon \end{cases}$	conditionnel

[6]concernant les systèmes de production d'atelier à tâches, pour chaque machine $M_i (i = 1, ..., n)$ l'ensemble des tâches correspondant à la séquence de traitement sur M_i est initialement en attente pour se traiter.

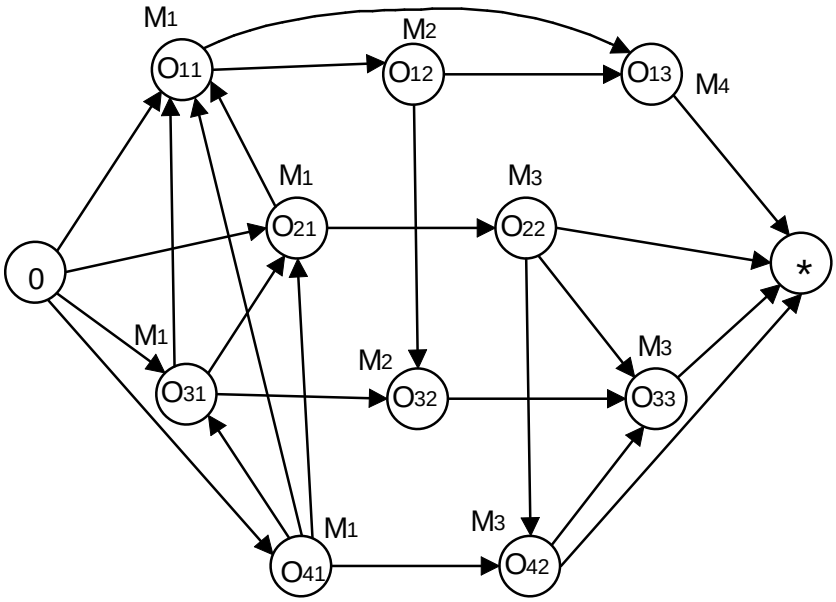


FIG. 1.8 – choix complet du graphe représenté dans la figure 1.7.

La recherche de taux maximal de traitement dans un graphe disjonctive est équivalente à la recherche d'un chemin à un coût minimal, obtenu par l'optimisation de la fonction du coût en utilisant des algorithmes heuristiques. Cela peut se traduire par la transformation des arcs disjonctifs non dirigés en des arcs dirigés selon le chemin obtenu d'après l'optimisation de la fonction du coût. Nous appelons l'ensemble des arcs dirigés un choix S, et celui des arcs disjonctifs transformés aux arcs dirigés sera appelé un choix fixé, mais un choix sera complet[7] si :

– chaque arc disjonctif est fixé,
– le graphe résultant $G(S) = (V, C \cup S)$ est acyclique.

La figure 1.8 montre un choix complet du graphe présenté dans la figure 1.7.

Les règles d'élimination des opérations **:** Pour illustrer l'outil d'élimination nous allons parler de l'idée principale du choix immédiat développé par Carlier et Pinson (1989,1991,1994) [Pinson, 1994] :
Soit I un ensemble des opérations à traiter sur une machine, notre but est de caractériser des séquences partielles et particulières d'opérations sur un sous ensemble de I qui n'amène pas à une solution globale du problème du système d'atelier et par conséquent, il est possible, soit de fixer quelques séquences alternatives, ou de placer quelques opérations par rapport aux autres.

1.3.3 Problème d'ordonnancement du système d'atelier [É. Pinson]

Les problèmes d'ordonnancement d'un système d'atelier sont définis comme suivant : nous avons n tâches ($i = 1, ..., n$), et m machines ($M_1, ..., M_m$), chaque tâche se compose d'une séquence d'opérations $O_{ij}(j = 1, ..., n_i)$ avec un temps de traitement p_{ij} à l'opération O_{ij}. Chaque opération doit être traitée sur la machine $\mu_{ij} \in \{M_1, ..., M_m\}$. La préemption de n'importe quelle opération est interdite, le but est de rechercher une séquence de traitement pour chaque machine minimisant une fonction particulière non décroissante (critère) dépend du temps d'execution de la tâche [Pinson, 1994]. Par exemple, le critère $C_{max} = max_{j=1,n}\ C_j$ où C_j indique le temps nécessaire pour finir l'execution de la dernière opération de la tâche j ($j = 1, ..., n$).

La $\mathcal{NP}$-dureté des problèmes d'ordonnancement du système d'atelier **:** quelques cas particuliers du problème d'ordonnancement ont des solutions en temps polynômial, les problème de deux machines avec $n_i \leqslant 3$ et les problèmes de trois machines avec $n_i \leqslant 2$ sont $\mathcal{NP}$-dures d'après (Lenstra et al. 1977) et (Gonzalez et Sahni 1978) [Pinson, 1994], mais les problèmes de trois machines sont fortement $\mathcal{NP}$-dures, et

[7] un choix qui contient tous les séquences de traitement des opérations

des études plus récentes (Stoskov 1991), ont montré que les problèmes des systèmes d'atelier de trois tâches sont $\mathcal{NP}$-dures [É. Pinson].

Chapitre 2

Simulation

La simulation sur ordinateur est un outil très utile pour bien comprendre le principe de fonctionnement d'un système. Elle aide à observer et à comprendre le fonctionnement en temps réel et à analyser plusieurs scénarios du fonctionnement.
On va parler dans ce chapitre de l'importance du langage de simulation comme *SIMAN-ARENA* pour la simulation des réseaux industriels, et comment peuvent être utilisés pour simuler les réseaux de Petri et les systèmes d'atelier.

2.1 Introduction au logiciel *SIMAN-ARENA*

Le logiciel *ARENA* est désigné essentiellement pour la création des modèles animés et pour représenter virtuellement n'importe quel système. La modélisation d'un système se fait en plaçant des objets ou des blocs (appelés modules) représentant ses comportements.

2.2 Le modèle informatique fondamental d'*ARENA*"template"

Le modèle informatique d'*ARENA* est une collection de plus de 60 modules. Il y a des modules représentants des ressources, des queues, des inspections, et des interfaces avec d'autres fichiers. Dans le domaine industriel, nous retrouvons des modules qui ont des caractéristiques comme le temps d'arrêt d'une machine, des algorithmes d'entretien, les convoyeurs (synchrone et asynchronous) et des dispositifs de transportation.

Pour développer un tel modèle de simulation, il faut choisir le module qui convient, on l'installe dans le plan du modèle et on affecte à chaque paramètre du module sa valeur en utilisant le message de sollicitation comme expliqué dans la figure 2.1.

L'analyse des entrées donne la possibilité de faire une analyse temporelle ou historique qui peuvent être inclues directement dans le modèle. L'analyse des sorties est utilisée pour afficher et analyser les résultats pendant ou après la simulation à l'aide de plusieurs moyens, soit par l'affichage direct des valeurs des variables de l'application ou des variables introduites pour observer un tel état pendant la simulation en plusieurs options, niveaux (levels), histogrammes, plot, soit par la génération d'un rapport contenant tous les valeurs des variables choisies pour étudier l'état statistique d'une application, ou soit par un moyen représentatif de la dynamique qui montre directement, le mouvement des changements en forme de mouvement des entités ou des événements [Takus and Profozich, 1998].

Une extension d'*ARENA* appelée RT est disponible pour les applications utilisant un modèle de simulation en interaction avec des applications externes [K. V. Nagarajan and Vial, 2002], cette interaction sera faite par un système de message en ligne, par exemple, le modèle de la simulation a un système logique qui envoie des tâches en temps réel à un système de contrôle, un messages sera mis en queue jusqu'à l'accomplissement du message précédent, et un message sera retourné au modèle de la simulation pour faire une demande ou pour faire une affirmation d'accomplissement d'une tâche, l'animation peut être ici comme un moniteur en temps réel pour montrer la dynamique du modèle de la simulation.

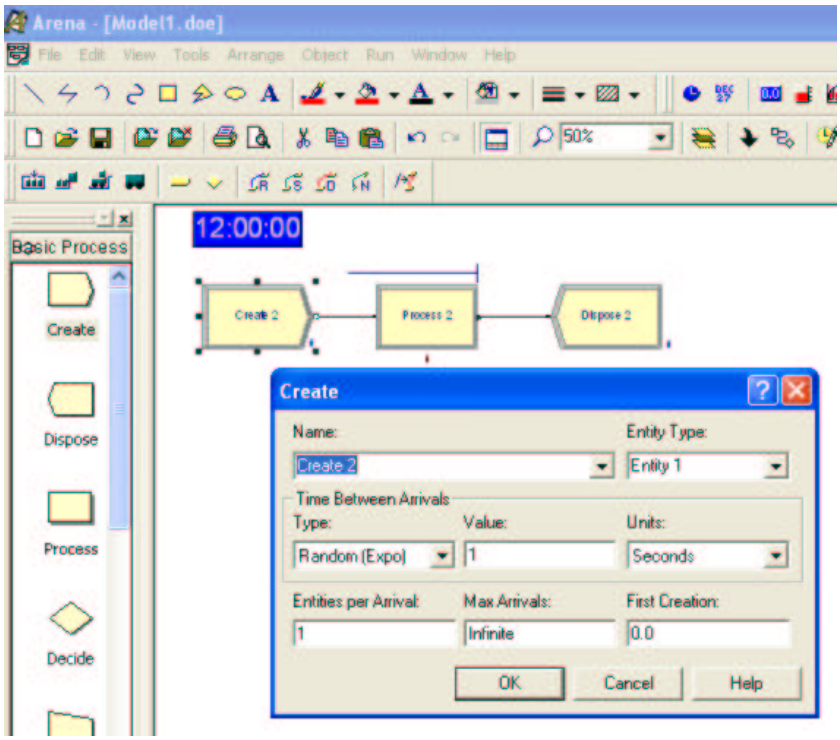


FIG. 2.1 – modèle de la simulation dans *ARENA*.

2.3 La simulation des réseaux de Petri par le logiciel *SIMAN-ARENA*

Un réseau de Petri est un graphe composé de places et de transitions, la place est spécifiée par sa capacité et la durée de séjour qu'un jeton doit y passer, ce qu'on peut traduire dans le modèle de *SIMAN-ARENA* par un module de délai. Les transitions seront modélisées par exemple, avec le module (MATCH) pour représenter une transition de synchronisation avec une possibilité de changement du nombre d'entrées en ajoutant des modules (QUEUE) à l'entrée du module précédent pour la synchronisation (figure 2.2).

La figure 2.2 a montre qu'il n'y a pas de liaison entre les modules QUEUE et MATCH. Dans ce cas là comme dans d'autres cas on peut faire les liaisons à l'aide des étiquettes (Label), qui doivent être définies dans le module de départ et le module d'arrivé (dans le champ nécessaire de la liste sollicité du module).

Pour un cas de conflit il faut tenir compte des conditions de distribution des jetons, et dans ce cas là, *ARENA* nous offre le module BRANCH avec un nombre de sorties en conflit contrôlable et une possibilité de lancer ou d'arrêter une sortie selon la condition évaluée en utilisant quatre types de conditions, *si, avec, sinon et toujours* (if, with, else and always) (figure 2.2 c). Par ailleurs on peut implémenter des conditions en expressions composées des relations mathématiques et logiques des variables de l'application concernée ou des variables d'observation ajoutées dans le modèle, en choisissant le terme "build expression" du message sollicité en appuyant sur le bouton droit du souris.

La figure 2.3 montre le modèle de la simulation d'*ARENA* de notre exemple de RdP présenté dans la figure 1.3 où il apparaît le nouveau module CREATE qui sert à créer des jetons suivant les dates d'entrée déterminées dans la liste de sollicitation concernant ce module.

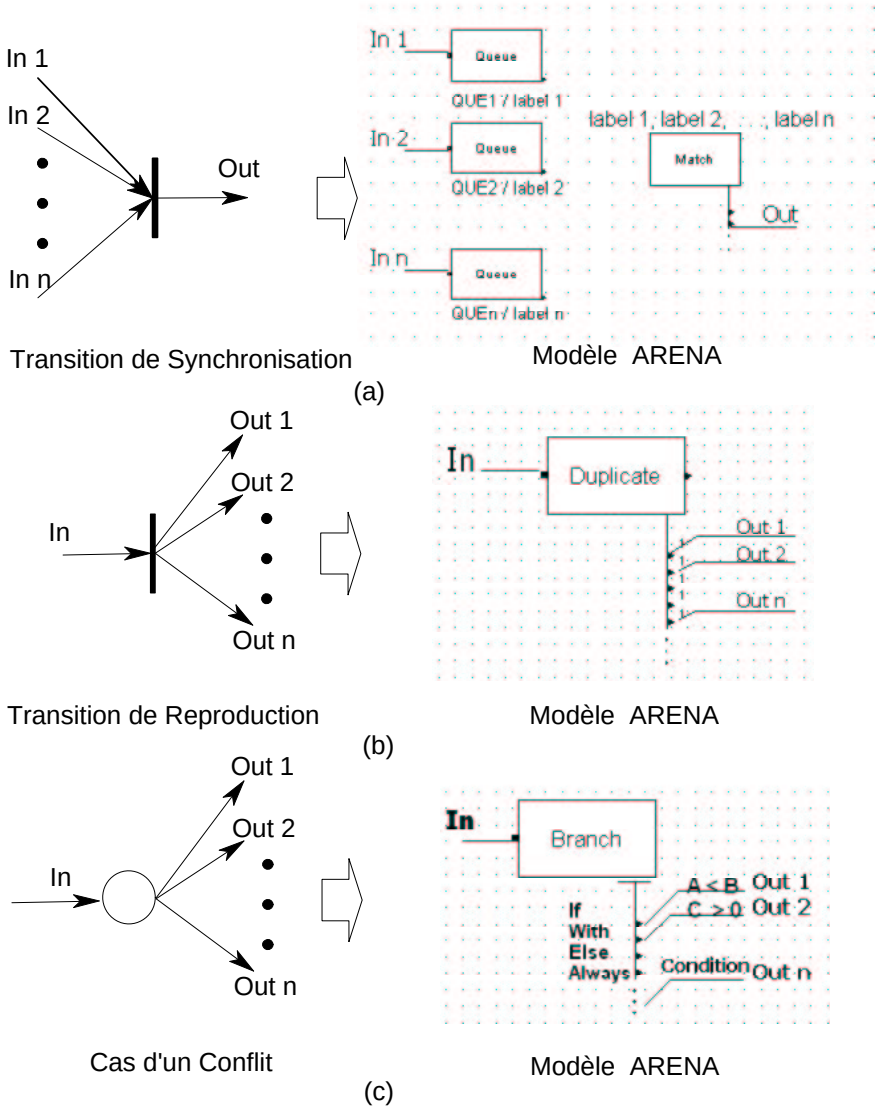


FIG. 2.2 – Modélisation des éléments de RdP en *ARENA*.

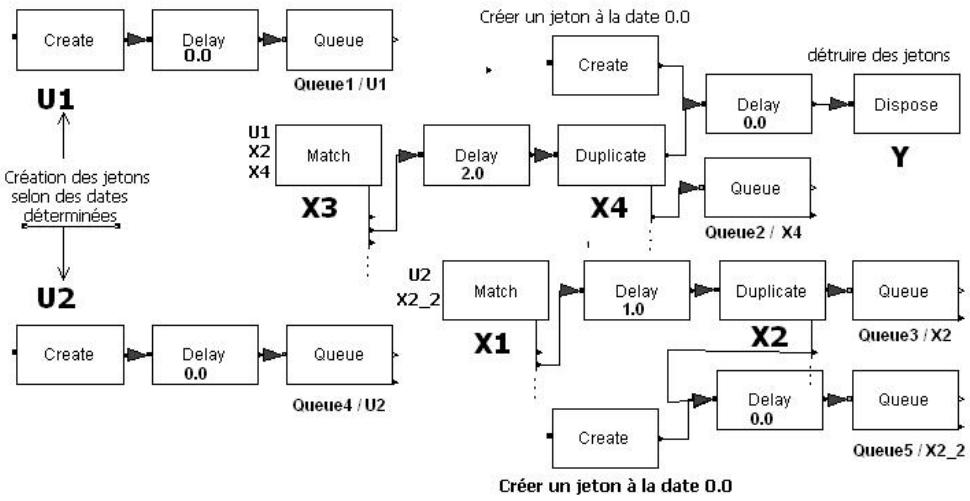
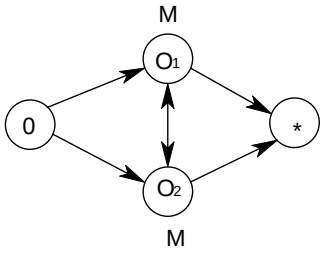
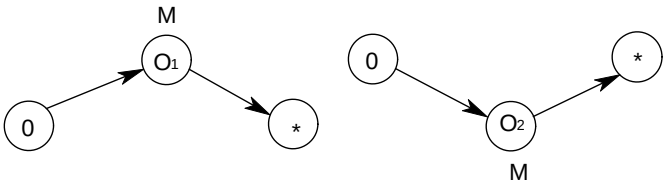


FIG. 2.3 – Un modèle de simulation dans *SIMAN-ARENA*.



a : Un système d'atelier représenté par un graphe disjonctif.



b : Deux choix du graphe disjonctif précédent.

FIG. 2.4 – Un graphe disjonctif avec deux de ses choix possibles.

2.4 La simulation des systèmes d'atelier avec le logiciel *SIMAN-ARENA*

Nous avons parler dans le paragraphe 1.3, de la possibilité de représentater des systèmes d'atelier par des réseaux de Petri et par des graphes disjonctifs. Pour les systèmes d'atelier représentés par les réseaux de Petri on peut suivre la même démarche que nous avons suivi dans le paragraphe précédent 2.3 pour les simuler. Pour ceux représentés par les graphes disjonctifs, il est plus facile à simuler les choix du graphe qu'ils forment une solution et qu'ils contiennent toutes les contraintes nécessaires, on va traiter un exemple de graphe disjonctif représentant un système d'atelier, figure 2.4.a. Ce système se compose d'une seule machine qui traite deux tâches, en considérant que la machine ne peut traiter qu'une seule tâche à la fois, de plus, une tâche ne peut entrer dans ce système qu'après la sortie de la tâche en cours d'exécution, ce qui aboutit aux choix présentés dans la figure 2.4.b.

Les deux choix représentent deux séquences de traitement, l'entrée des tâches au noeud 0, le traitement sur la machine aux noeuds M/O_1 et M/O_2 et la sortie des tâches sera fait au noeud 0. Puisque le temps de traitement se caractérise par le temps d'entrée d'une tâche, le temps de traitement sur la machine et le temps de sortie, nous pourrons modéliser les deux circuits du traitement dans *SIMAN-ARENA* par une série des délais représentant les trois temps précédents, et le choix entre ces deux séquence de traitement (séries de délais) sera fait par un algorithme contrôlant le chargement el le déchargement des tâches selon les contraintes et l'ordonnancement.

2.5 Conclusion

A travers ce chapitre nous avons introduit les bases de la simulation dans *SIMAN-ARENA* pour les systèmes d'atelier.

Chapitre 3

Application

Dans ce chapitre nous allons considérer une application de système d'atelier où apparaît un problème d'ordonnancement. Nous allons tout d'abord définir le problème, puis on va le modéliser par les réseau de Petri et par les graphes disjonctifs pour chercher une solution. Pour vérifier la solution nous allons faire une simulation dans les deux cas (modèle par les réseau de Petri,modèle par les graphes disjonctifs).

3.1 Définition du problème

Il s'agit d'un système d'atelier, proposé par Michel Alsaba et Jean Louis Boimond, composé de deux lignes de production (deux tâches) et d'une seule machine (figure 3.1), comme deux trains sur une seule voie, ou deux systèmes de communication sur un seul canal. La première tâche a besoin d'un prétraitement de 3 secondes et de traitement sur la machine d'une seconde. La deuxième tâche a besoin d'une seconde de prétraitement et d'une seconde de traitement sur la machine. Le rechargement des tâches demande une seconde pour la première tâche mais avec un délai de 5 secondes pour le rechargement suivant d'une autre tâche de même type, et la deuxième tâche demande 3 secondes pour être rechargée.

3.2 Modélisation et simulation à base de réseaux de Petri

3.2.1 Représentation et Modélisation

M. Alsaba et J. L. Boimond représentent leur système d'atelier par le réseau de Petri de la figure 3.2. Cette représentation montre deux parties entrée et sortie et trois ressources (P_4, P_2, P_5). Les deux parties (entrée,sortie) sont de type GET, on peut donc appliquer toutes les règles contrôlant le mouvement des jetons dans un GET à chaque partie indépendamment, et comme nous cherchons la meilleur séquence de distribution des tâches il suffit tout d'abord de chercher la plus courte séquence de circulation des tâches dans ces deux parties, qui est équivalant à résiduer ces deux GET. Après résiduation, on obtient les résultats suivants :

$$\begin{array}{lcl} u_1 & = & x_1 - 1, \\ u_2 & = & x_2 - 3, \\ x_3 & = & y_1 - 3, \\ x_4 & = & min(y_2 - 2, y_1 - 4). \end{array} \tag{3.1}$$

Concernant les trois ressources (P_4, P_2, P_5), la ressource P_2 est une ressource partagé par P_4 et P_5. Du point de vue industriel les auteurs tentent de traiter ce problème de ressource partagée comme un problème de satisfaction d'une demande du client, ce qui les a amené à suivre la stratégie du juste à temps (en anglais, JIT just in time) pour diriger l'exploitation de la ressource P_2 ver l'une de P_4 ou P_5 et ils ont développé l'algorithme basé sur la résiduation suivant :

$$Soit \;\; Z = \begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix} = \begin{pmatrix} Y_{1ref} \\ Y_{2ref} \end{pmatrix} \text{la sortie de référence.}$$

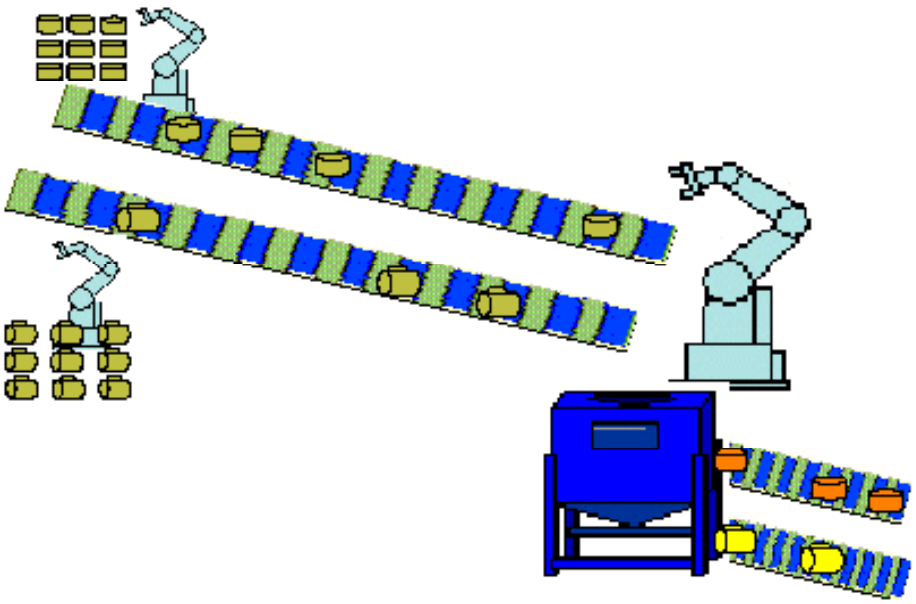


FIG. 3.1 – Système de production proposé par Michel Alsaba et Jean Louis Boimond.

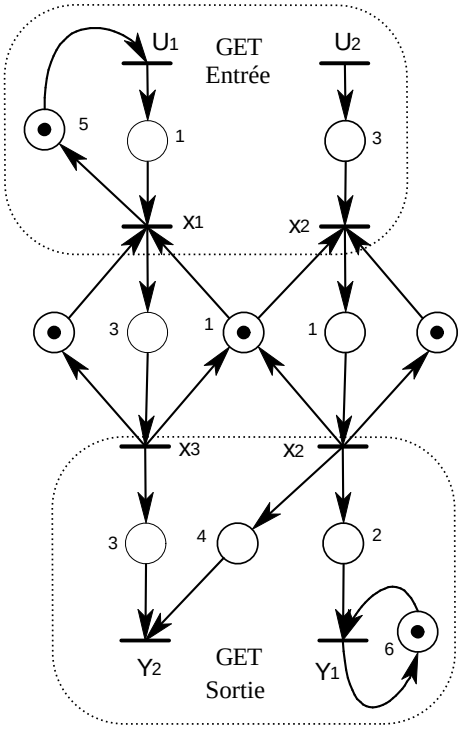


FIG. 3.2 – Représentation du système de la figure 3.1 par un réseau de Petri.

Si $Z_1(l) \preceq Z_2(h)$ alors,

$$\left\{\begin{array}{lcl} \xi(k) & = & (\xi(k+1) \wedge Z_2(h) \not\phi \begin{pmatrix} \varepsilon \\ \varepsilon \\ e \end{pmatrix}) \not\phi \begin{pmatrix} e & \varepsilon & \varepsilon \\ \varepsilon & 2 & 1 \\ \varepsilon & 2 & 1 \end{pmatrix} \\ x_2(h) & = & \xi(k) \not\phi (\varepsilon, \varepsilon, e) \\ h & = & h-1 \end{array}\right. \tag{3.2}$$

sinon

$$\left\{\begin{array}{lcl} \xi(k) & = & (\xi(k+1) \wedge Z_1(h) \not\phi \begin{pmatrix} e \\ \varepsilon \\ \varepsilon \end{pmatrix}) \not\phi \begin{pmatrix} 3 & 4 & \varepsilon \\ 3 & 4 & \varepsilon \\ \varepsilon & \varepsilon & e \end{pmatrix} \\ x_1(l) & = & \xi(k) \not\phi (e, \varepsilon, \varepsilon) \\ l & = & l-1 \end{array}\right. \tag{3.3}$$

Notons que, quand la date de délivrance des deux lignes est identique, la comparaison $Z_1(l) \preceq Z_2(h)$ donne la priorité à la seconde ligne de production (x_2, x_4).

3.2.2 Simulation du problème par *SIMAN-ARENA*

Le modèle de la simulation : Dans la figure 3.2, quatre transitions se présentent, deux transitions de synchronisation dont le sous-modèles est dans la figure 3.3.a, et les deux transitions de reproduction acceptent le sous-modèles décrit dans la figure 3.3.b, et les places seront modélisés par les modules de DELAY (cf. paragraph 2.3), le modèle de la simulation dans *SIMAN-ARENA* a la forme de la figure 3.4.

Monsieur J. L. Boimond propose une autre représentation du système de traitement sur la machine (ressource partagé) disposant de l'exemple du robot, en modélisant les deux tapis par deux queues (module SEIZE) avec des délai du temps pour le prétraitement des tâches (modules Delay44 et Delay46, figure3.2.2), le robot par le module (RELEASE) qui fait relâcher de ces deux queues suivant l'algorithme d'ordonnancement précédent, et le temps d'exécution par un délai (modules Delay45).

La commande du système : Puisqu'on va commander ce système par un vecteur d'entrées dépendant de l'algorithme JIT et des sorties de référence (Z), et comme l'algorithme est composé des relations mathématiques et des relations de l'algèbre MAX^+, on aura besoin d'un outil pour appliquer l'algorithme et calculer les entrées de commande d'après les sorties références. Il faut de plus que cet outil ait la possibilité d'envoyer les commandes à *SIMAN-ARENA* et de recueillir les sorties pour recalculer de nouveau le vecteur de commande suivant ou pour montrer les résultats.

Cet outil est basé sur le logiciel (*MATLAB*), qui a la plupart des outils mathématiques. Puisque l'algèbre MAX^+ n'est pas bien défini, on va créer les fonctions nécessaires pour développer l'algorithme, comme les deux fonctions (MULTIROND.m) et (POINTROND.m) qui font les deux opérations respectivement $(\otimes, \odot)$, et les deux fonctions (residL AparB.m) et (residR AparB.m) qui font aussi les deux opérations respectivement $(A \backslash B, A \not/ B)$.

L'interface MATLAB $\leftrightarrow$ SIMAN-ARENA : Le logiciel *SIMAN-ARENA* permet de s'interfacer avec d'autres logiciels à travers la socket numéro (4334). Cette socket peut être contrôlée par le modèle de la simulation à l'aide de deux modules, le premier est le module TASKS de type ELEMENT qui permet d'envoyer un message en chaîne de caractères à l'extérieur procédé si un jeton ou un événement passe par le module DELAY. Le deuxième est le module STATION de type AdvancedTransfer par lequel on peut recevoir des messages en chaîne de caractères de l'extérieur, le module STATION est contrôlé par le module ARRIVAL de type ELEMENT à l'aide duquel on définit les entrées par des numéros donnés aux stations, en utilisant le message sollicité de ce module, nous considérons que la chaîne de caractères acceptée comme entrée est de la forme (numéro d'entrée, la date d'exécution) et pour celle de la sortie nous acceptons la forme (le numéro de la sortie, le numéro de l'événement, la date de sortie).

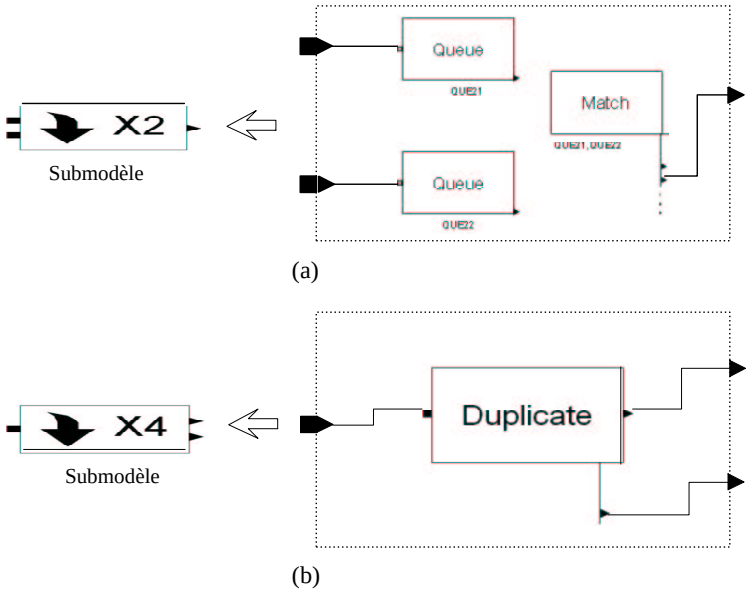


FIG. 3.3 – Représentation des transitions par des sousmodèles.

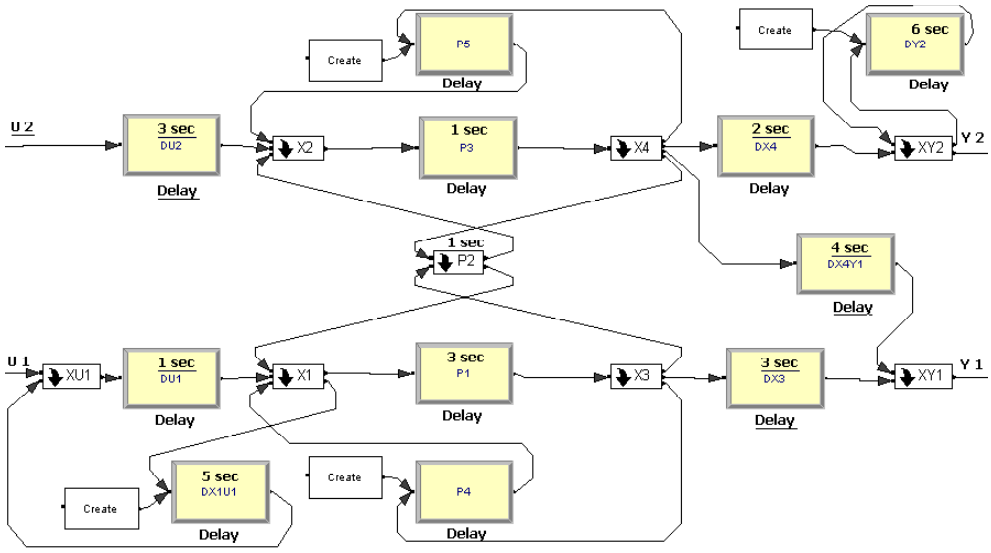


FIG. 3.4 – Modèle de la simulation de la figure 3.2 dans *SIMAN-ARENA*.

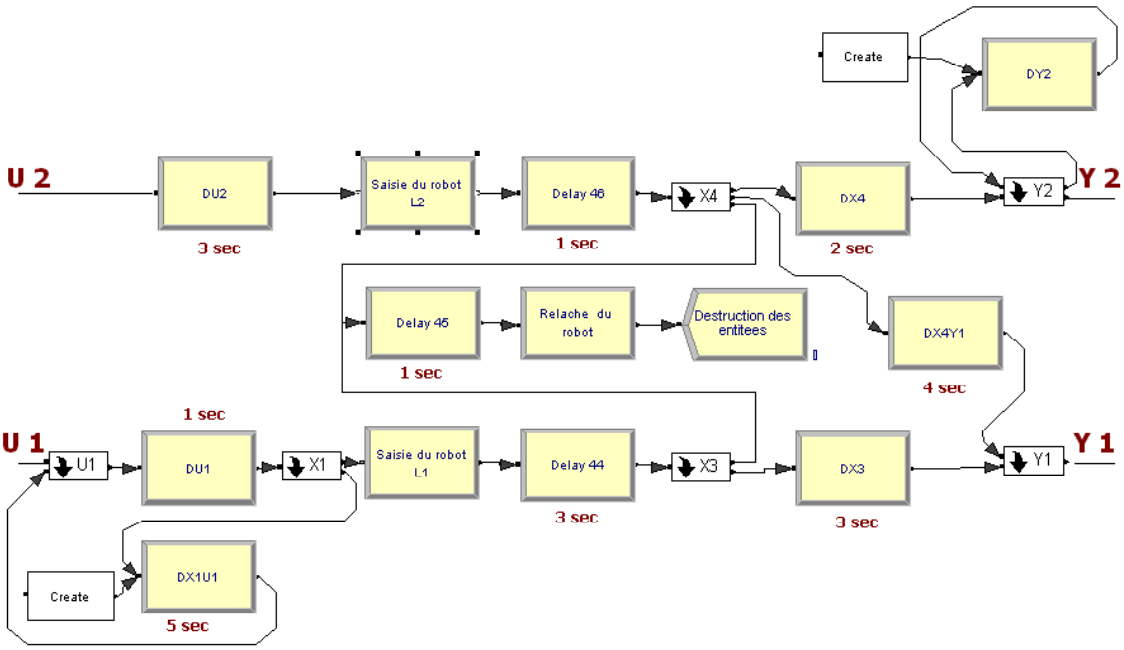


FIG. 3.5 – Autre modèle de la simulation de la figure 3.2 dans *SIMAN-ARENA*.

DMATLAB calcule les entrées de commande par l'algorithme (JIT)disposant des sortie de références, et à l'aide des outils de l'algèbre MAXPLUS que nous avons manipuler par les instructions de *MATLAB*. puis *MATLAB* renvoie les messages d'entrées de la commande du modèle de la simulation d'*ARENA*, et recueille le vecteur de sorties créé par le modèle de la simulation avec les entrées de commande. Ce rôle de communication est effectué par un programme crée par Marie-Françoise Gérard, qui se compose de trois fonctions (connection, envoi, déconnection) implémenté en C++. La figure 3.4 montre le modèle de simulation du réseau de Petri présenté dans la figure 3.2, et le modèle de simulation pour la communication avec *MATLAB* est représenté dans la figure 3.5.

3.2.3 Application numérique

Dans cette partie nous allons reprendre les données utilisées par M. Alsaba pour démarrer et vérifier notre exemple de simulation, ces données sont les suivant :

$$Y_{ref} = Z = \begin{pmatrix} 20 & 22 & 23 & 48 \\ 23 & 38 & 39 & 48 \end{pmatrix}.$$

Nous avons obtenu les mêmes résultats que M. Alsaba :

$$U_{jit} = \begin{pmatrix} 4 & 10 & 16 & 41 \\ 0 & 6 & 12 & 37 \end{pmatrix} \quad et \quad Y = \begin{pmatrix} 11 & 17 & 23 & 48 \\ 6 & 12 & 18 & 43 \end{pmatrix},$$

Ce qui montre la compatibilité du modèle de simulation avec le réseau de Petri.

3.3 Modélisation et simulation à base des graphes disjonctifs

3.3.1 Représentation et modélisation

Reformulation du problème : Notre problème se compose de deux tâches (J_1, J_2) à traiter sur une seule machine (M) et sans préemption. Chaque tâche a une seule opération d'où les deux opérations (O_1, O_2) à exécuter pendant les durées d'execution respectivement (P_1, P_2). Nous disposons dans ce problème des dates de sortie des tâches $(t_{*1}(k), t_{*2}(k))$ (les sorties références) et on recherche les dates d'entrée de commande

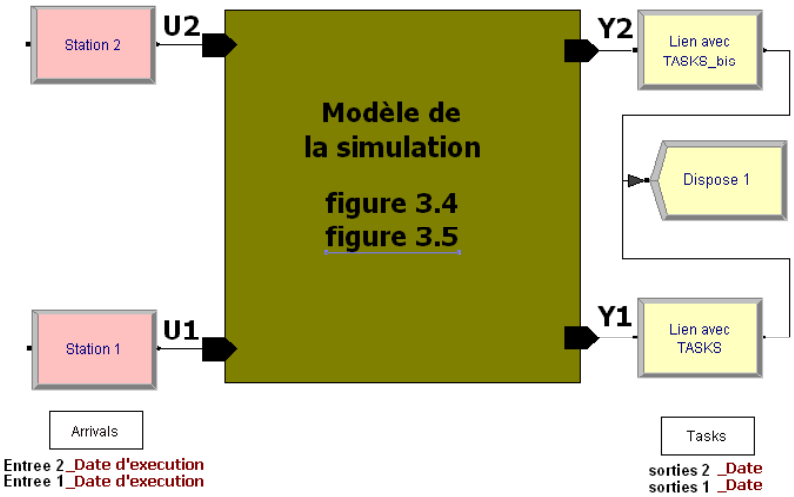


FIG. 3.6 – Modèle de la simulation de la communication entre notre modèle d'*ARENA* (fig 3.4,fig 3.5) et *MATLAB*.

$(t_{01}(k), t_{02}(k))$.

Représentation et modélisation par les graphes disjonctifs : D'après la reformulation précédente, notre graphe disjonctif représentant ce problème se compose de deux noeuds (1,2) d'opérations (O_1, O_2), et un noeud (0) de départ et l'autre de destination (*). On peut représenter ce problème par le graphe disjonctif présenté dans la figure 3.7.a, où cette figure montre les arcs disjonctifs non dirigés entre les deux tâches ce qui signifie que l'une de ces tâches sera traitée avant ou après l'autre sur la machine.

Notations **:** Soit r_i $(respectivement\ q_i)$ la tête (respectivement le queue), soit $l(i,j)$ la plus longue distance entre i et j, on définit $r_i = l(0,i)$ et $q_i = l(i,*)$ où p_i le temps de traitement de l'opération i.

Solution **:** Un ordonnancement dans un graphe disjonctif $G = (G, D)$ est un ensemble des temps de départ $T = t_i \ : \ i \in X$ où X l'ensemble des opérations des tâches [É. Pinson]. Pour construire un ordre des tâches dans une séquence, il faut construire un choix (paragraphe 1.3.2), les deux choix présentés dans la figure 3.7.b, ne montrent pas de contraintes. La précédences des deux tâches dépend de ces deux choix qui déterminent le temps d'exécution total de la séquence de traitement de chaque tâche sans intervention de l'autre. On peut alors faire un ordre de tâches en respectant ces temps d'exécution comme dans les équations 3.4 :

$$\begin{aligned} t_{*1} &\geqslant q_1 + p_1 + r_1 + t_{01}, \\ t_{*2} &\geqslant q_2 + p_2 + r_2 + t_{02}. \end{aligned} \tag{3.4}$$

Dans le cas inverse, pour calculer les dates de départ des tâches en respectant la règle JIT et en prenant compte l'équation 3.4 nous obtenons les temps de départ, équation 3.5 :

$$\begin{aligned} t_{01} &\leqslant t_{*1} - (q_1 + p_1 + r_1), \\ t_{02} &\leqslant t_{*2} - (q_2 + p_2 + r_2). \end{aligned} \tag{3.5}$$

Disposons de l'axe du temps sur le quel nous marquons les dates de sortie des tâches (sortie reference), revenons en arrière en suivant une règle disant : on fait l'ordre de tâches d'après leur sortie (ce qui sorte après, il arrive après), et en faisant un compromis entre les dates de départ comme le montre la figure 3.6, nous pourrons conclure l'algorithme suivant :
Soit n le nombre d'itérations d'ordonnancement qu'on veut faire, soit i et j deux entiers représentant les indices sur les deux vecteurs d'opérations respectivement 1 et 2, alors :

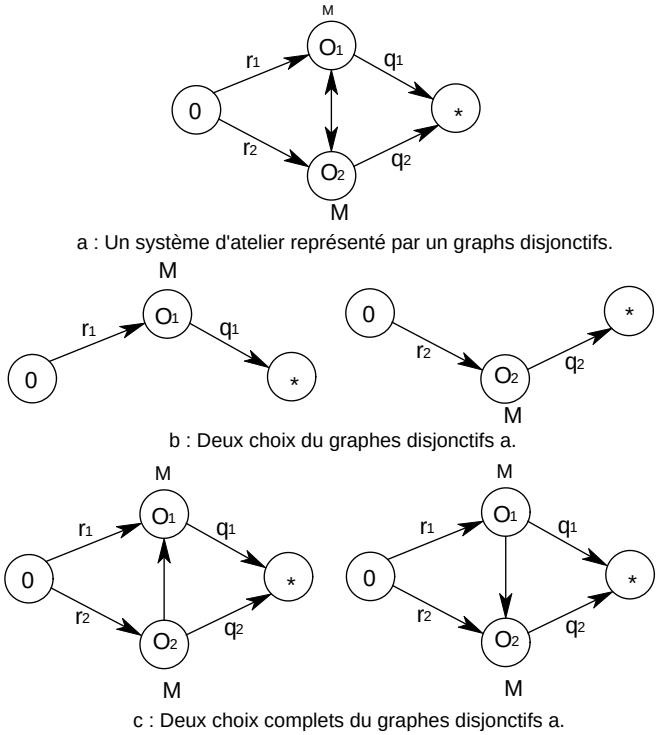


FIG. 3.7 – Un graphe disjonctif et ses choix.

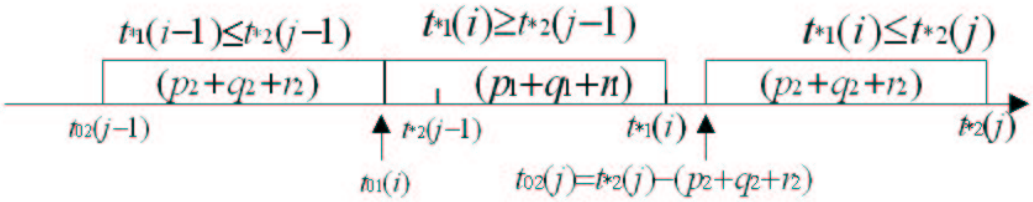


FIG. 3.8 – Ordre temporel des temps de départ des tâches selon les temps de sortie.

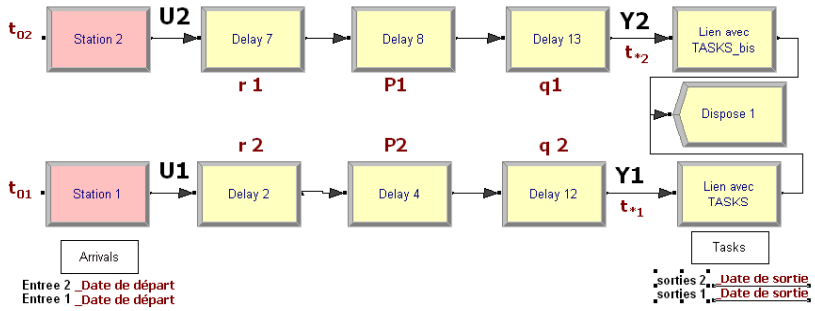


FIG. 3.9 – Modèle de simulation de la figure 3.7.b.

$initialisation\ i \leftarrow n, j \leftarrow n,$
$t_0 \leftarrow max(t_{*1}(n), t_{*2}(n)),$
début,
tantque i ou j inférieur à n faire,
$si\ (t_{*1}(i) \leqslant t_{*2}(j))\ alors,$

$$\left|\begin{array}{rl} t_{02}(j) & \leftarrow min(t_0, t_{*2}(j)) - (p_2 + r_2 + q_2), \\ t_0 & \leftarrow t_{02}(j), \\ j & \leftarrow j - 1, \end{array}\right.$$

$Sinon$

$$\left|\begin{array}{rl} t_{01}(i) & \leftarrow min(t_0, t_{*1}(i)) - (p_1 + r_1 + q_1), \\ t_0 & \leftarrow t_{01}(i), \\ i & \leftarrow i - 1, \end{array}\right.$$

revenir à (début),
$fin.$

Les deux autres choix complets, figure 3.7.c, représentent deux autres solutions, une tâche peut être exécuté avant la sortie de l'autre qui est en cours d'exécution, en respectant qu'une seule tâche peut être exécutée à la fois sur la machine, donc on profite d'éliminer les temps de rechargement et de déchargement. Dans ce cas là, beaucoup de contraintes interviennent de type conjonctif et disjonctif, qui demande l'utilisation des méthodes heuristiques et des méthodes de programmation des entiers mixtes.

3.3.2 Simulation du problème par *SIMAN-ARENA*

D'après ce que nous avons dit au paragraphe 2.4, nous proposons le modèle suivant, figure 2.9, comme un modèle de simulation dans *SIMAN-ARENA*, en dépendant de MATLAB pour programmer l'algorithme précédent et qui va contrôler les modules d'entrées pour faire entrer des tâches selon l'algorithme.

3.3.3 Applications numériques

Supposons que les paramètres du système ont les valeurs suivantes :

$$\begin{array}{ll} r_1 = 1\ sec, & r_2 = 3\ sec. \\ q_1 = 3\ sec, & q_2 = 2\ sec. \\ p_1 = 3\ sec, & p_2 = 1\ sec. \end{array}$$

Soit les dates de sortie données par la matrice :

$$t_{*ref} = \begin{pmatrix} 22 & 30 & 40 & 52 \\ 25 & 27 & 43 & 51 \end{pmatrix}.$$

On obtient les valeurs suivantes pour les dates de départ comme des résultats de la simulation :

$$t_0 = \begin{pmatrix} 0 & 19 & 26 & 45 \\ 7 & 13 & 33 & 39 \end{pmatrix} \quad \Rightarrow \quad t_* = \begin{pmatrix} 7 & 26 & 33 & 52 \\ 13 & 19 & 39 & 45 \end{pmatrix}.$$

3.4 Conclusion

D'après les deux applications précédentes, on peut déduire que :

- Les réseaux de Petri peuvent représenter les détails temporels et la dynamique du système d'atelier.
- Les graphes disjonctifs ne montre pas la dynamique du système mais il représentent plus clairement les contraintes, comme on peut inspirer des solutions des problèmes représentés par ces graphes sans faire de calcules.

Bibliographie

[Birkhoff, 1940] Birkhoff, G. (1940). Lattice theory. Technical Report XXV, American Mathematical Society Colloquim Publications, Providence, Rhode Island.

[Carlier and al., 1993] Carlier, J. and al. (1993). Les problèmes d'ordonnancement, recherche opérationnelle. pages 77–150.

[Cohen, 1995] Cohen, G. (1995). Théorie algébrique des systèmes à Événements discrets. Cours, Centre Automatique et Systèmes, École des Mines, Paris FontainBleu & Rocquencourt INRIA.

[David and Alla, 1989] David, R. and Alla, H. (1989). *Du grafcet aux réseaux de Petri*. Hermès.

[David Rivreau, 1999] David Rivreau, P. (1999). *Problémes d'Ordonnancement Disjonctifs : Règles d'Elimination et Bornes Inférieures*. Thèse, Université de Technologie de Compiègne.

[Hillion and Proth, 1989] Hillion, H. P. and Proth, J. M. (1989). An algebra for network routing problems. *IEEE Transactions on Automatic Control*, 34 :273–294.

[K. V. Nagarajan and Vial, 2002] K. V. Nagarajan, G. A. and Vial, P. (2002). The use of arena simulation software to illustrate network operations in an educational setting using case studies. *University of Wollongong*.

[Kurkovsky and Loganantharaj,] Kurkovsky, S. and Loganantharaj, R. Extension of Petri Nets and its Applications to Model Systems with Imprecise Task Duration.

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

[Pinson, 1994] Pinson, E. (1994). The job shop scheduling problem : A concise survey and some recent developments. *Institute de Mathématique Appliquées, Université Catholique de l'Ouest*, pages 276–293.

[Takus and Profozich, 1998] Takus, D. A. and Profozich, D. M. (1998). Arena software tutorial. *Systems Modeling Corporation, Sewickley, Pennsylvania , U.S.A.*