Conception et pilotage de cellules flexibles à fonctionnement répétitif modélisés par réseaux de Petri

Stéphane Julia

N° d'ordre : 2712 - Année 1997

Thèse

préparée au

Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS

en vue de l'obtention du

Doctorat de l'Université Paul Sabatier de Toulouse

Spécialité : Informatique Industrielle

par

Stéphane JULIA

Maître ès Sciences

Conception et Pilotage de Cellules Flexibles à fonctionnement répétitif modélisées par Réseaux de Petri

---

Soutenue le 16 juillet 1997 devant le jury:

Président	**M.**	**COURVOISIER** *Professeur UPS (Toulouse)*
Rapporteurs	**H.**	**ALLA** *Professeur ENSIEG (Grenoble)*
	J.C.	**GENTINA** *Professeur Ecole Centrale de Lille*
Examinateurs	**G.**	**FONTAN** *Professeur INPT (Toulouse)*
	D.	**NOYES** *Professeur ENIT (Tarbes)*
Directeur de thèse	**R.**	**VALETTE** *Directeur de recherche au LAAS-CNRS (Toulouse)*

Cette thèse a été préparée au LAAS-CNRS
7, Avenue du Colonel Roche, 31077 Toulouse Cedex 4

Rapport LAAS N° 97282

Avant-propos

Les travaux présentés dans ce manuscrit ont été effectués au Laboratoire d'Analyse et d'Architecture des Systèmes du Centre National de la Recherche Scientifique au sein du groupe Systèmes de Production, au "*Centro Federal de educação tecnológica do Paranâ*" (Brésil) au sein du "*Curso de Pós-Graduação em Engenharia Elétrica e Informática Industrial*" et enfin à l'Université Fédérale d'Uberlândia (Brésil) au sein du Département d'Informatique. Je tiens donc à exprimer ma gratitude à Monsieur Alain Costes qui m'a accepté au LAAS, à Monsieur Jean-Claude Hennet, actuel responsable du groupe SP, à Monsieur Maurizio Tazza qui m'a accueilli dans son groupe de recherche au *CPGEI* et enfin à ma femme et à Monsieur Costa Pereira qui m'ont permis de poursuivre mes travaux de recherche à l'Université Fédérale d'Uberlândia. Je tiens enfin à remercier le gouvernement Brésilien par le biais du *CNPq*, l'organisme de recherche Bresilien, qui a financé ces travaux de recherche.

Je tiens à remercier tout particulièrement mon directeur de thèse, Monsieur Robert Valette, qui a accepté de m'encadrer "à distance" compte tenu de la situation peu commune dans laquelle j'ai effectué ces travaux. Ses conseils, ses critiques, ses encouragements, sa sympathie et surtout sa patience m'ont permis de mener cette thèse à son terme.

Je suis extrêmement reconnaissant à Messieurs Hassane Alla et Jean-Claude Gentina pour avoir accepté la charge d'être rapporteurs des travaux présentés. J'exprime également ma reconnaissance à Monsieur Marc Courvoisier pour avoir présidé le jury de cette thèse. Je suis enfin extrêmement reconnaissant à Messieurs Gérard Fontan et Daniel Noyes pour l'intérêt qu'ils ont bien voulu porter à ce travail et pour leur participation au jury en tant qu'examinateurs.

Pour l'appui et l'aide administrative qu'ils m'ont apportés, je tiens à remercier tout particulièrement ma mère, Simone Julia, et Monsieur Serge Attali, directeur de l'UFR-PCA à l'Université Paul Sabatier.

Pour leur formidable accueil au Brésil, je remercie Myriam et Armando.

Pour leur assistance technique, je remercie Flavien Huynh, Bernard Bassil et Henrique.

Pour avoir relu le manuscrit de thèse, je remercie mon Père, Gérard Julia.

Enfin, pour avoir cru en moi, j'exprime ma plus profonde gratitude à ma femme, Rita Maria Da Silva Julia et je dédie ce mémoire à notre fils Etienne Da Silva Julia.

Table des matières

Introduction

Devant l'évolution importante en matière de production dans les pays industrialisés, les ateliers de fabrication doivent pouvoir aujourd'hui s'adapter le plus rapidement possible aux changements et aux évolutions de la production. L'évolution des sciences comme l'informatique ou la robotique a permis la naissance de nouveaux types d'ateliers de fabrication qui répondent bien à ces exigences : les ateliers flexibles. L'objectif principal de ces derniers et de concilier à la fois l'efficacité et la flexibilité qui sont généralement considérées comme des critères contradictoires.

Le cycle de développement d'un atelier flexible est une tâche extrêmement complexe constituée par un ensemble d'étapes où l'on trouve la spécification des produits, la conception et la mise en marche du système, et éventuellement le démantèlement ou l'évolution de ce dernier. Pour aider les opérateurs humains au cours de l'ensemble des étapes du cycle de développement des ateliers flexibles, ceux-ci sont aujourd'hui équipés d'un système informatique (*CIM: Computer Integrated Manufacturing*) intégrant tout un ensemble de fonctions. Elles sont généralement réparties dans trois modules distincts qui sont: *la conception assistée par ordinateur* qui intervient dans la phase de spécification des produits, *l'ingéniérie assistée par ordinateur* qui aide à concevoir le système au travers de nombreux calculs de performance et *la fabrication assistée par ordinateur* qui fait intervenir les fonctions de gestion de la production, de commande en temps réel et de surveillance.

Le principal problème lorsque l'on se place dans le contexte de l'ingéniérie de production est d'être capable de proposer une méthodologie complète de conception, d'analyse et de commande des ateliers flexibles. Devant la difficulté du problème à résoudre, on est généralement amené à étudier séparément les différentes fonctions du système informatique chargé d'assister les ingénieurs au cours de l'ensemble des étapes du cycle de développement de l'atelier flexible alors qu'il existe en réalité un haut degré de corrélation entre toutes ces fonctions. Une règle impérative à observer afin de garantir dans la mesure du possible un développement cohérent et rigoureux est l'utilisation d'une méthodologie formelle. Ceci nous amène à justifier l'approche que nous allons présenter dans ce travail et qui est l'intégration de certaines des fonctionnalités d'un *CIM* à partir d'un même modèle. Nous nous limiterons au cas de cellules flexibles à fonctionnement répétitif et nous aborderons plus particulièrement les problèmes de l'aide au dimensionnement, de l'aide à l'élaboration de politiques de commande et du pilotage temps réel de cellules flexibles. Le travail sera basé sur une approche hybride qui devra combiner d'une part les réseaux de Petri pour la partie modélisation et analyse qualitative, et d'autre part les techniques de résolution de l'Intelligence Artificielle pour tout ce qui sera rattaché à l'analyse sous contrainte des

cellules flexibles.

Le mémoire est divisé en cinq chapitres.

Dans le premier chapitre, nous présentons la problématique de la conception et de la production dans les cellules flexibles. Un bref rappel est fait sur les cellules flexibles et sur le rôle fondamental de l'informatique dans les ateliers de fabrication. Les réseaux de Petri sont présentés comme l'outil principal de modélisation et d'analyse des cellules flexibles. Face à certaines limitations des réseaux de Petri, les techniques d'Intelligence Artificielle sont introduites comme un complément aux réseaux de Petri qui peut être utilisé pour tout ce qui est lié à l'analyse d'un problème sous un ensemble de contraintes. Nous introduisons enfin plus précisément le cadre de notre travail en nous situant dans le contexte d'une approche hybride qui combine les réseaux de Petri pour les parties de modélisation et d'analyse qualitative, et les techniques d'Intelligence Artificielle pour tout ce qui est rattaché à l'exploitation des contraintes que l'on peut déduire du modèle.

Dans le deuxième chapitre, nous présentons la modélisation de cellules flexibles. Les réseaux de Petri sont utilisés pour la représentation des diverses contraintes de synchronisation ainsi que pour l'analyse qualitative du modèle obtenu. Nous présentons ensuite le modèle temporisé utilisé pour modéliser aussi bien les durées d'opération qui sont les paramètres de notre approche, que les durées d'attente des pièces dans les stocks qui sont elles les inconnues de notre approche.

Dans le troisième chapitre, nous présentons l'approche générale pour l'analyse quantitative d'une cellule flexible. C'est ici qu'est défini le régime stationnaire périodique forcé à partir duquel seront déduites des contraintes linéaires. Ces contraintes linéaires seront ensuite utilisées pour calculer par l'intermédiaire de la programmation mathématique des bornes minimales et maximales qui exprimeront les conditions nécessaires d'une production plus ou moins flexible. Cette partie a comme domaine d'application plus particulièrement le problème du dimensionnement dans les cellules flexibles (étape de conception de la cellule ou de sa modification).

Le quatrième chapitre aborde les problèmes de l'ordonnancement et du pilotage temps réel dans les cellules flexibles. Le principe de fonctionnement du joueur de réseau de Petri p-temporel t-temporisé avec retour arrière ("backtracking") utilisé pour le calcul d'un ordonnancement cyclique prévisionnel est présenté. La particularité de cet algorithme est qu'il utilise les bornes calculées au chapitre précédent. Finalement, en nous basant sur le résultat de l'ordonnancement prévisionnel calculé, le modèle de réseau de Petri est utilisé pour un pilotage temps réel et réactif de la cellule au travers de l'utilisation d'un joueur temps réel ne possédant plus de mécanisme de retour arrière et permettant de décrire en temps réel l'enchaînement des traitements à effectuer sur les pièces, de communiquer avec l'environnement extérieur et d'être soumis à des contraintes temporelles explicites.

Dans le dernier chapitre, un exemple d'application illustre l'approche exposée. A partir d'un exemple particulier de cellule flexible de fabrication, les principaux aspects de la méthode sont illustrés: modélisation de l'ensemble des contraintes et analyse qualitative du modèle obtenu, analyse quantitative de la cellule pour le dimensionnement, calcul d'un ordonnancement prévisionnel qui doit servir de base au pilotage temps réel de la cellule.

Chapitre 1

Conception et Production dans les Cellules Flexibles

1.1 Ateliers Flexibles

Devant l'évolution importante en matière de production dans les pays industrialisés, les entreprises doivent faire face à une concurrence internationale qui ne cesse de croître. Le consomateur cherche systématiquement le produit nouveau et les dimensions de la production à l'échelle mondiale font qu'il est nécessaire de combiner une grande flexibilité, similaire à celle des ateliers à tâches avec une grande productivité, similaire à celle des lignes de production. Considérant la complexité d'un atelier de fabrication et les moyens mis en œuvre pour sa réalisation, il est nécessaire que celui-ci ne devienne pas inadapté chaque fois que l'on a à faire face à des modifications brutales du marché. Au contraire, il doit être conçu de telle sorte qu'il s'adapte le plus rapidement possible aux changements et aux évolutions de la production.

Grâce à l'évolution de sciences comme l'informatique ou la robotique, nous voyons aujourd'hui la naissance de nouveaux types d'ateliers de fabrication: les ateliers flexibles. Un des principaux intérêts de ces systèmes est qu'ils essaient de concilier à la fois l'efficacité et la flexibilité. Nous devons souligner la difficulté d'atteindre un tel équilibre lorsque l'on sait que généralement efficacité et flexibilité sont considérés comme des critères contradictoires. En effet, si l'on prend le système extrêmement flexible qu'est l'homme, on réalise que celui-ci en matière d'efficacité de production est parfois limité. De même, les ateliers de type ligne de production, s'ils permettent un taux de production très élevé sont complètement dépourvus de la capacité de s'adapter aux changements.

Généralement nous pouvons distinguer deux types de flexibilité [SI 89]:

- La flexibilité à long terme qui correspond à la possibilité d'introduire rapidement et à peu de frais de nouvelles familles de produits au sein d'un même atelier.

- La flexibilité à court terme qui correspond à la possibilité de produire simultanément plusieurs types de familles de produits.

Bien qu'il n'existe pas de définition formelle de ce qu'est un atelier flexible (chaque atelier ayant des caractéristiques propres), il y a néanmoins un ensemble de composants que l'on retrouve systématiquement au sein de chaque atelier flexible [KU 90] et qui sont:

- Un ensemble de machines programmables ou de stations de travail plus ou moins flexibles. En général, ces machines peuvent réaliser divers types d'opérations.
- Un système de transport automatisé et flexible. Contrairement aux lignes de production, la disposition des machines au sein d'un atelier flexible n'est pas déterminée par la séquence des opérations à effectuer et doit laisser une liberté totale en ce qui concerne le séquencement des opérations. On doit pouvoir atteindre n'importe quelle station de travail de n'importe quel point de l'atelier. En particulier, une telle liberté est nécessaire si l'on veut pouvoir modifier le séquencement des opérations au cours du temps enfin de s'adapter au mieux aux diverses perturbations (comme les pannes de machines par exemple) qui peuvent survenir durant la production.
- Un système de stockage distribué au niveau de chaque station de travail et/ou centralisé au niveau global de l'atelier.
- Un système de décision sophistiqué qui doit décider à chaque instant ce qui doit être fait et sur quelle machine. Les systèmes de conduite et de commande d'ateliers sont en général des systèmes répartis, souvent construits autour d'un réseau local industriel.

Fréquemment, les ateliers flexibles sont structurés en un ensemble de cellules flexibles. Une cellule flexible est constituée d'un ensemble de machines, d'un système de stockage local et d'un système de manipulation automatisé permettant le transfert des pièces à usiner à l'intérieur de la cellule. La manipulation est généralement réalisée à l'aide d'un robot manipulateur. Ce type d'architecture favorise une répartition des tâches à réaliser ; un ensemble de tâches plus ou moins similaires (ou au contraire complémentaires comme dans le cas des îlots) étant attribué à chaque cellule. De plus, le fait de regrouper les opérations au niveau de cellules diminue considérablement les temps de transport à l'intérieur de l'atelier. Plusieurs opérations nécessaires à l'élaboration d'un même produit peuvent par exemple être réalisées au sein d'une même cellule, le système global de transport de l'atelier permettant de faire la connexion entre chaque cellule. Une autre différence entre un atelier et une simple cellule vient du fait que pour un atelier les stocks tampons découplent, dans une certaine mesure, la gestion des cellules ou des ilôts. Par contre, à l'intérieur d'une cellule les opérations de stockage et de manutention doivent être traitées de la même façon que les opérations d'usinage. Les problèmes de gestion des ressources et de synchronisation des opérations sont donc très importants. Enfin, soulignons que pour une cellule, il faut représenter à la fois l'architecture physique du système (la disposition des machines et des équipements) et la (ou les) politique de gestion envisagée. Dans la suite de ce travail, lorsque nous aborderons les problèmes de conception et de production, nous nous limiterons au cas de cellules flexibles.

1.2 Cycle de Vie d'un Atelier Flexible

Le cycle de vie d'un atelier flexible est généralement constitué par un ensemble d'étapes:

- Spécification des produits.
- Conception préliminaire.
- Conception détaillée.
- Production.
- Démantèlement ou évolution du système.

1.2.1 Spécification des Produits

La spécification des produits est la première étape du cycle de développement d'un système de production. Elle est directement issue d'une étude du marché et est à l'origine de la mise en place d'un projet de conception d'un système de production. Précisons qu'avec l'introduction de la notion de flexibilité, les systèmes de production doivent être non seulement capables de réaliser plusieurs types de produits, mais aussi de suivre l'évolution du marché et par conséquent de modifier progressivement les produits réalisés initialement au sein du système de production. Tout celà a pour conséquence de rendre cette étape beaucoup plus complexe que lorsqu'il s'agissait de produire un seul type de produit à grande échelle et sur une importante période de temps. Nous devons être capable de définir toute une gamme de produits divers avec une certaine anticipation dans le temps afin de prévoir à l'avance les fonctionnalités que devra avoir le système pour réaliser au mieux la production. Une des fonctions importantes de cette étape va être la définition pour chaque type de produit de l'ensemble des séquences d'opérations possibles pour sa réalisation c'est-à-dire la définition des gammes de fabrication.

1.2.2 Conception Préliminaire

Durant la phase de conception préliminaire, on va :

- choisir les ressources parmi celles qui sont disponibles (machines, système de transport, robots etc...) afin d'être capable de réaliser la production définie à l'étape précédente,
- étudier la meilleure disposition des ressources sur la surface disponible (agencement du système),
- choisir le système de gestion de production que l'on appliquera au système physique,

- évaluer le comportement dynamique du système afin d'obtenir tout un ensemble d'indices de performance qui vont nous permettre de vérifier si le système répond bien au cahier des charges défini à l'étape de spécification des produits. On utilise généralement à ce niveau la simulation ainsi qu'éventuellement certaines techniques analytiques d'évaluation de performance.

Tout au long de cette phase de conception préliminaire, il est important d'essayer dans la mesure du possible d'optimiser tout un ensemble de critères qui caractérisent une production efficace, de bonne qualité et correspondant à un prix de revient le plus compétitif possible. Précisons que ce problème est loin d'être trivial puisqu'il correspond à un problème d'optimisation multicritère.

1.2.3 Conception Détaillée

La conception détaillée consiste à réaliser l'implantation physique du système de production. Après avoir formé le personnel qui va participer au bon fonctionnement du système, on va mettre en marche le système en mode ralenti afin de réaliser les derniers réglages avant la mise en marche définitive.

1.2.4 Production

La production correspond au fonctionnement normal du système. On retrouve ici des opérations comme la gestion de la production, la conduite temps réel, la surveillance et la maintenance du système. Du fait de la complexité d'un atelier flexible, une étude globale du système est difficilement possible et la structure de décision de l'atelier est généralement décomposée en une hiérarchie de niveaux d'abstraction [ER 86, SI 89]. Le but d'une telle décomposition est de ramener la résolution d'un problème de grande dimension aux caractéristiques variées, à une suite de résolutions de problèmes de taille raisonnable ayant des caractéristiques plus homogènes. Le but ici n'est plus de retenir une solution parmi un ensemble d'alternatives mais plutôt de restreindre progressivement l'ensemble des alternatives possibles en utilisant les connaissances propres à chaque niveau. Les niveaux de décision classiques sont les suivants:

Planification

A ce niveau de la production, on doit établir un plan de production agrégé en fonction d'une estimation de la demande des clients et de la prévision de l'évolution du marché. En particulier, c'est ici que l'on définit les divers taux de production i.e. le nombre de produits à traiter par semaine ou par mois. On fait aussi une pré-allocation des diverses ressources afin de diminuer l'explosion combinatoire qui peut se produire au niveau de l'ordonnancement du fait de la flexibilité des machines qui peuvent généralement traiter plus d'un type d'opérations.

Ordonnancement

C'est à ce niveau que doit être élaboré le plan de fabrication détaillé. On définit l'ordre d'exécution de chaque opération sur chaque machine tout en tenant compte des contraintes imposées par le niveau de décision supérieur (planification). Nous retrouvons comme contraintes en particulier le respect des taux de production et la pré-allocation des ressources. Parce que chaque machine peut exécuter diverses opérations, l'explosion combinatoire du nombre d'alternatives possibles dans le cas des ateliers flexibles est énorme. L'ordonnancement est parfois donné comme une séquence d'opérations parfaitements ordonnées. On considèrera alors que l'on a défini un ordonnancement explicite. Une telle stratégie est plutôt à éviter parce que non seulement elle met en jeu des temps de calcul très importants, mais en plus elle n'est pas du tout flexible. Une autre solution est d'élaborer un ensemble de règles qui définissent chaque fois qu'une machine devient libre l'opération suivante. Une telle stratégie peut être considérée comme la définition d'un ordonancement implicite. Généralement c'est une approche mixte combinant ces deux stratégies qui donnera le meilleur résultat.

Supervision et Pilotage en Temps Réel

Les deux fonctions principales sont la surveillance et la prise de décision en temps réel.

La fonction de surveillance permet de détecter en temps réel tous les comportements anormaux ou non nominaux (panne de machine etc.) [SA 87, CO 89] et d'effectuer un diagnostic efficace afin d'orienter au mieux l'équipe de manutention.

La fonction de prise de décision en temps réel est chargée de résoudre en temps réel les indéterminismes qui existent encore. Le système doit respecter au mieux l'ordonnancement prévisionnel qui a été calculé à l'étape précédente et doit être capable dans le cas d'un comportement anormal de faire une redistribution des tâches à effectuer au sein du système en temps réel afin que la production ne soit pas interrompue à chaque événement imprévisible. Soulignons que lorsque l'ordonnancement est défini de façon implicite au travers de règles de décisions, on pourra alors parler d'ordonnancement temps réel.

Coordination des Sous-Systèmes

La fonction principale de ce niveau est de s'occuper de la commande en temps réel des sous-systèmes de l'atelier. Il s'agit en particulier de la gestion du système de transport permettant de faire la liaison entre les diverses cellules.

Commande Locale

C'est à ce niveau qu'est implémentée la commande en temps réel des machines, des robots, etc..., le plus souvent au travers d'automates programmables [CO 86] et de commandes numériques. Soulignons qu'aucune décision n'est prise à ce niveau et que les contraintes de temps sont généralement extrêmement fortes puisque nous nous trouvons dans le niveau le plus bas.

1.2.5 Démantèlement ou Evolution du Système

Cette étape peut être considérée comme la fin du cycle de vie d'un atelier flexible. Généralement, elle intervient lorsque les perturbations au niveau du marché sont si importantes que le système implanté n'est plus capable de répondre efficacement aux besoins de la production. Ou bien on modifiera le système afin de l'adapter à la nouvelle production, ou bien on le démantellera et on mettra en place un nouveau projet de conception d'un système de production plus actualisé.

Dans la suite de ce travail, nous allons nous consacrer aux étapes de conception et de production des cellules flexibles. Nous aborderons plus particulièrement les problèmes d'évaluation de performance pour le dimensionnement, d'ordonnancement, et de pilotage temps réel. Dans la suite de ce chapitre, après avoir montré le rôle fondamental de l'informatique dans les ateliers flexibles, nous présenterons ensuite les diverses approches issues de l'informatique qui peuvent nous aider à résoudre au mieux ces problèmes.

1.3 Rôle de l'Informatique dans les Ateliers Flexibles

L'application de l'informatique au secteur industriel et tout particulièrement aux ateliers de fabrication constitue aujourd'hui un domaine de recherche en pleine expansion. En particulier, les progrès considérables de l'informatique ces dernières années ont permis l'introduction d'un haut degré d'automatisation dans l'industrie manufacturière qui s'est traduit par:

- une production élevée et régulière,
- l'amélioration de la qualité des produits,
- l'amélioration des conditions de travail et de sécurité.

Les ateliers flexibles sont aujourd'hui équipés d'un système informatique (*CIM: Computer Integrated Manufacturing*) intégrant tout un ensemble de fonctions permettant d'automatiser ou d'assister l'ensemble des étapes du cycle de développement. En général, les diverses fonctions d'un *CIM* sont réparties dans trois modules distincts qui sont:

- la *Conception Assistée par Ordinateur* (*CAD: Computer Aided Design*).
- l'*Ingéniérie Assistée par Ordinateur* (*CAE: Computer Aided Engineering*).
- la *Fabrication Assistée par Ordinateur* (*CAM: Computer Aided Manufacturing*).

1.3.1 Conception Assistée par Ordinateur

La *Conception Assistée par Ordinateur* intervient dans la phase de *spécification des produits*. Ce module est utilisé pour la description des produits, pour le choix du matériel qui va être utilisé et pour définir la façon dont le matériel va être utilisé pour réaliser les produits. En particulier, c'est à ce niveau que l'on va définir les différentes familles de produits en fonction de la similitude des opérations qui vont leur être appliquées. De cette définition vont être produites les différentes gammes de fabrication possibles. Soulignons que par la suite, l'établissement simple et efficace d'une séquence admissible (ordonnancement) va essentiellement dépendre de la façon subtile dont ont été regroupés les produits à l'intérieur de familles distinctes. *A. Kusiak* dans [KU 94] donne justement à ce propos des règles de spécification des produits qui doivent permettre par la suite de simplifier l'ordonnancement de la production.

1.3.2 Ingéniérie Assistée par Ordinateur

L'*Ingéniérie Assistée par Ordinateur* permet d'assister la phase de *conception préliminaire*. On va essentiellement à ce niveau établir les performances du système de production à l'aide de techniques de simulation ou de méthodes analytiques. Précisons que les performances du système vont être étroitement liées au choix du type de gestion de production qui va être appliqué.

1.3.3 Fabrication Assistée par Ordinateur

La *Fabrication Assistée par Ordinateur* va directement intervenir durant la phase de *production*. On trouve à ce niveau les fonctions de gestion de production, de commande en temps réel et de surveillance. L'élaboration des programmes pour les machines outils à commande numérique fait également partie de la fabrication assistée par ordinateur.

Nous allons maintenant présenter les modèles et techniques de résolution qui sont utilisés au sein des *CAE* et des *CAM* puisque ce sont les seuls modules directements concernés par les phases de conception et la production des cellules flexibles.

1.4 Utilisation des Réseaux de Petri dans le Cadre des Cellules Flexibles

Les réseaux de Petri ont pour origine la thèse de Carl Adam Petri intitulée *Communication avec des Automates* qui a été présentée en 1962 à l'Université de Darmstadt. C'est à partir de ce travail qu'Anatol W. Holt aidé par un groupe de chercheurs du MIT (*Massachussetts Institute of Technology*) a développé les bases de la théorie des réseaux de Petri entre 1968 et 1976. Les réseaux de Petri sont aujourd'hui considérés comme un outil de représentation formel qui permet la modélisation, l'analyse et le contrôle des systèmes à événements discrets qui comprennent des activités parallèles,

concurrentes et asynchrones. Les références suivantes traitent de la définition des réseaux de Petri et de leurs propriétés: [PE 66], [BR 83], [DA 92], [MU 89], [PE 81], [RE 90], et [RE 85].

Une cellule flexible peut être vue comme un système à événements discrets. De plus, son architecture distribuée met en jeu des problèmes de parallélisme, de concurrence et de synchronisation. Les réseaux de Petri donc, de par leur nature à traiter ces types de problèmes, sont parfaitement adaptés à la spécification, à la modélisation, à la validation et à la mise en œuvre d'une cellule flexible. Nous allons voir maintenant plus en détail comment utiliser les réseaux de Petri pour la conception, l'ordonnancement et le pilotage de cellules flexibles.

1.4.1 Utilisation des Réseaux de Petri pour la Conception

La conception (*CAE*) d'une cellule flexible comme d'un atelier repose sur les tâches suivantes:

- modélisation du système,
- analyse qualitative du modèle,
- analyse quantitative du modèle.

Modélisation

Comme nous l'avons vu, à l'intérieur d'une cellule les opérations de stockage et de manutention doivent être traitées de la même façon que les opérations d'usinage. De plus, il faut représenter par le même modèle aussi bien l'architecture physique de la cellule que la politique de gestion qui va lui être appliquée. Les principaux avantages des réseaux de Petri pour la modélisation sont alors les suivants:

- Leur aptitude à modéliser le vrai parallélisme [DA 92].
- La possibilité de modéliser la cellule de façon progressive en ayant une approche par raffinement (*top-down*) ou en appliquant une approche modulaire (bottom-up) qui grâce à l'utilisation de sous-réseaux bien formés (ils vérifient les bonnes propriétés [DA 92]) permet une réduction de l'effort de vérification des bonnes propriétés du modèle global (voir par exemple [VA 82]). Soulignons ici que l'approche par raffinement (*top-down*) sera plutôt utilisée lorsque le système considéré sera un atelier et qu'une certaine hiérarchie entre les niveaux de contrôle devra apparaître. Certains modèles comme les Statecharts (Harel) par exemple pourront alors être mieux adaptés. Par contre l'approche modulaire (bottom-up) rentre bien dans le cadre de représentation d'une cellule et ne demande pas d'adaptation particulière de la définition de base des réseaux de Petri.
- La simplicité de représentation des divers synchronismes existant dans une cellule. Par exemple, le modèle d'une cellule à laquelle est appliquée une politique

kanban est aisément représenté par un réseau de Petri [MA 91] alors qu'une telle prise en compte de la synchronisation n'est absolument pas possible si l'on considère par exemple les réseaux de files d'attente [AL 85] qui sont plutôt utilisés dans la phase de conception préliminaire d'un atelier.

- Leur caractère graphique qui est fondamental lorsque l'on fait de la simulation afin de pouvoir suivre directement le comportement dynamique de la cellule.

- Le fait de pouvoir modéliser avec les réseaux de Petri simultanément le système physique et le système de contrôle lorsque sont considérés des systèmes à fonctionnement cyclique [SV 94].

Même en considérant des approches modulaires, il est parfois difficile de modéliser à l'aide d'un simple réseau places/transitions une cellule dans toute sa complexité du fait de l'explosion des états du système. Face à ce problème, de nouveaux modèles dit de haut-niveau ont été définis: réseaux de Petri colorés [JE 90], réseaux de Petri prédicats/transitions [GE 87], réseaux de Petri à objets [SB 94]. L'utilisation de "couleurs" qui différencient les jetons permet ainsi d'obtenir un modèle plus compact. Dans la plupart des cas, c'est l'utilisation d'un modèle de haut-niveau combiné à une approche modulaire ou à une approche par raffinement qui nous permettra de modéliser des systèmes aussi complexes que les cellules ou ateliers flexibles (voir par exemple [AU 94]).

La principale fonction d'un *CAE* est d'évaluer le comportement dynamique d'un atelier au niveau de la conception préliminaire afin d'obtenir tout un ensemble d'indices de performance (taux de production, durée cyclique, indices d'utilisation des ressources etc. [TA 87]) et de pouvoir considérer les possibles alternatives de production (disposition des cellules de fabrication, type de système de transport, nombre de machines, capacités de stockage des divers magasins etc.). Nous parlerons alors d'*évaluation de performance.* L'évaluation de performance d'une cellule n'est possible que dans la mesure où le modèle considéré est capable de prendre en compte les durées d'opération. Dans un réseau de Petri autonome [DA 92], le temps apparaît de façon implicite et qualitative (on peut représenter par exemple l'ordre des opérations). Pour l'étude du comportement dynamique, il est nécessaire de prendre en compte le temps de façon explicite et quantitative. De ce fait, de nombreux modèles qui permettent une intégration simple et efficace du temps ont été définis. Les principaux sont:

- Les réseaux de Petri p-temporisés [?]. Le temps est représenté par des durées (nombres rationnels positifs ou nuls) associées aux places du modèle.

- Les réseaux de Petri t-temporisés [RA 74]. Le temps est représenté par des durées (nombres rationnels positifs ou nuls) associées aux transitions du modèle.

- Les réseaux de Petri temporels [ME 74]. Le temps est représenté par un intervalle $[(d_t)_{min}, (d_t)_{max}]$ associé à la transition t. $(d_t)_{min}$ est la durée de sensibilisation minimale de t. $(d_t)_{max}$ est la durée de sensibilisation maximale. Le temps est donc donné ici sous la forme de durées de sensibilisation associées aux transitions. L'avantage des réseaux temporels est qu'ils sont plus généraux

que les réseaux temporisés. Ils permettent par exemple de décrire les "chiens de garde", ce que les réseaux temporisés ne permettent pas.

- Les réseaux de Petri p-temporels. Le temps est représenté par un couple de dates $((d_p)_{min}, (d_p)_{max})$ associé à la place p. $(d_p)_{min}$ représente la date à laquelle un jeton se trouvant dans p devient disponible pour le tir d'une transition. $(d_p)_{max}$ représente la date avant laquelle le jeton de p doit être utilisé pour le tir d'une transition, sous peine de violation de contrainte et de fonctionnement anormal du réseau. *Khansa* dans [KH 96] utilise ce modèle pour modéliser des durées d'opération incertaines qui sont représentées sous forme d'intervalles associés aux places du réseau.

- Les réseaux de Petri stochastiques [FL 84]. Un temps aléatoire est associé au franchissement d'une transition. On modélise ainsi des phénomènes qui ne peuvent pas être bien modélisés avec des durées constantes comme par exemple le temps de bon fonctionnement d'une machine entre deux pannes. L'hypothèse la plus courante est que les temporisations sont distribuées selon une loi exponentielle, ce qui permet d'associer au réseau un processus de Markov homogène.

Analyse qualitative

Un des avantages des réseaux de Petri est leur capacité à être analysés qualitativement. On peut ainsi vérifier certaines propriétés au niveau du modèle. Parmi ces propriétés, nous pouvons mettre en évidence d'une part ce que nous appelons les "bonnes propriétés" et d'autre part les composantes conservatives et répétitives stationnaires.

Les bonnes propriétés ont comme caractéristique principale d'être fortement liées au marquage initial du réseau. Leurs définitions impliquent des considérations sur l'ensemble des marquages accessibles à partir du marquage initial. Les plus importantes lorsque l'on utilise les réseaux de Petri comme modèle d'une cellule flexible sont celles qui garantissent qu'un réseau est:

- k-borné (le concept de réseau borné correspond au fait qu'un système est toujours limité, ce qui est le cas par exemple lorsque l'on représente les stocks d'une cellule),

- vivant (un réseau de Petri vivant garantit qu'aucun blocage ne peut être provoqué par la structure du réseau),

- réinitialisable (un réseau est réinitialisable généralement lorsqu'il modélise un système à fonctionnement répétitif, ce qui est le cas dans les cellules flexibles auxquelles sont appliqués des gestions de production de type cyclique).

Les méthodes d'analyse de ces propriétés font appel essentiellement à des algorithmes d'énumération des marquages et à des algorithmes de réduction du modèle.

Les composantes conservatives et répétitives stationnaires, contrairement aux bonnes propriétés, sont des propriétés qui découlent directement de la structure du réseau,

c'est-à-dire qui ne dépendent pas du marquage initial. Une composante conservative, d'un point de vue graphique, définit un sous-réseau de Petri qui correspond généralement à la représentation d'une contrainte à respecter au sein de la cellule. Un invariant de transition est une séquence de franchissements de transitions qui ne modifie pas le marquage du réseau. Un tel invariant correspond à des séquences cycliques d'événements qui peuvent être répétées indéfiniment et qui existent lorsque l'on considère des cellules à fonctionnement cyclique. Les composantes conservatives et répétitives stationnaires peuvent être calculées de façon systématique grâce à la résolution d'un système linéaire faisant intervenir une simple méthode de Gauss (voir [MA 81] par exemple).

Une présentation plus complète de l'ensemble des propriétés qualitatives des réseaux de Petri peut être consultée dans [MU 89].

L'analyse qualitative des réseaux de Petri places/transitions est un domaine dans lequel il existe déjà de très bon résultats. Le problème à prendre en compte est qu'une cellule flexible est généralement représentée par un réseau de Petri de haut niveau interprété non autonome [DA 92] et que l'analyse ne peut être faite que sur le réseau autonome sous-jacent. Néanmoins, même ainsi, les résultats obtenus permettent de valider certaines propriétés, ce qui n'est pas toujours possible avec d'autres outils de modélisation.

Analyse quantitative

Généralement, l'évaluation de performance d'une cellule est faite par simulation. Même si de très nombreux langages et formalismes de simulation existent, les réseaux de Petri restent, de par leur caractère rigoureux et formel, l'outil de modélisation se rapprochant le plus de la nature réelle d'une cellule flexible. De plus, leur représentation graphique permet de visualiser le comportement dynamique du système pendant la simulation. On utilise en principe au niveau d'une simulation un modèle de haut niveau qui inclut des durées stochastiques qui rendent possible la simulation de perturbations comme les pannes par exemple.

Une autre approche est d'aborder l'évaluation de performance au travers de méthodes analytiques. Lorsque le modèle utilisé est un réseau de Petri stochastique, la méthode est basée sur la génération d'un processus de Markov [AL 85]. On est obligé dans ce cas de passer par une énumération des marquages qui n'est possible que pour des réseaux bornés, et est de complexitée exponentielle. Par conséquent, nous ne pouvons considérer pour des raisons d'explosion combinatoire que des réseaux de taille raisonnable qui modélisent difficilement une cellule dans toute sa complexité. Un des avantages à souligner, lorsque l'on utilise les réseaux stochastiques, est la possibilité de faire de l'analyse de sensibilité (voir [VI 94] par exemple).

Les réseaux de Petri temporisés (dans le cas non stochastique) sont eux aussi bien adaptés à l'évaluation de performance de cellules flexibles. La méthode analytique employée est basée sur la recherche du circuit critique dans un réseau de Petri t-temporisé et sur le calcul du temps de cycle associé. *Ramamoorthy et Ho* ont défini ce temps de cycle dans [RA 80]. Par la suite, de nombreux chercheurs ont utilisé cette même notion de temps de cycle pour obtenir par une approche analytique les

performances d'ateliers de fabrication du type *atelier à tâche* (job-shop) [HI 88, HI 89, AK 93, MA 85]. La principale limitation de cette approche vient du fait que le modèle considéré est un graphe d'événements (chaque place a exactement une transition d'entrée et une transition de sortie) qui ne permet la modélisation du partage de ressources que si tous les conflits ont été résolus au préalable par l'introduction d'une relation d'ordre total. Ce type de modèle ne peut donc être utilisé pour décrire une cellule flexible que si l'on considère un ordonnancement particulier. L'avantage des réseaux de Petri qui permettent de représenter un ensemble de séquences admissibles [SI 89] plutôt qu'un ordonnancement unique n'est alors pas exploité.

Le caractère non-linéaire d'un modèle plus général qu'un graphe d'événements va forcément rendre difficile l'évaluation de performance réalisée par une méthode analytique. On peut toutefois espérer mettre en évidence certaines bornes qui constitueront des conditions nécessaires de réalisation de la production et qui pourront être utilisées dans la phase de conception préliminaire. Par exemple, *Tazza* dans [TA 87] aborde de façon analytique l'étude du comportement dynamique d'un atelier flexible modélisé par un réseau de Petri p-temporisé qui met en jeu des problèmes d'allocation de ressources. La méthode se fonde en particulier sur la définition d'une période de base d'utilisation des ressources.

Certains travaux sur l'évaluation de performance d'ateliers de fabrication non-cycliques ont déjà été réalisés (voir [SV 94]). Leur faiblesse réside dans le fait que l'évaluation de performance dépend très fortement du calcul d'un ordonnancement particulier mettant en jeu des algorithmes plutôt lourds. De plus, le résultat obtenu hors ligne ne sera pas forcément utilisable en ligne au niveau de la supervision et du pilotage en temps réel de l'atelier car l'évaluation hors ligne se base sur des valeurs moyennes autour desquelles des variations non négligeables peuvent avoir lieu.

Toutes ces remarques nous montrent les efforts qui doivent encore être fournis en particulier en ce qui concerne les méthodes analytiques pour l'évaluation de performance des ateliers de fabrication modélisés par des réseaux plus généraux que les graphes d'événements.

1.4.2 Utilisation des Réseaux de Petri pour la Production

Nous allons maintenant voir en quoi les réseaux de Petri peuvent être intéressants pour aborder les problèmes d'ordonnancement et de pilotage temps réel.

Ordonnancement

Si l'on considère un ordonnancement comme une suite d'opérations à effectuer sur une pièce afin d'obtenir un produit fini, alors les réseaux de Petri t-temporisés sont particulièrement bien adaptés à la modélisation de l'ordonnancement d'une cellule flexible. En effet, il est extrêmement naturel d'associer aux transitions du réseau les opérations ou tâches et de représenter les ressources par des jetons se trouvant dans des places.

Lorsque l'on parle d'ordonnancement, on fait généralement référence à une séquence d'opérations particulière. Si tel est le cas, alors les graphes d'événements

constituent un modèle suffisant pour décrire une solution, en particulier dans le cas d'ordonnancements cycliques. Une fois l'ordonnancement modélisé par le réseau, il est possible pour le régime stationnaire périodique d'obtenir la performance optimale grâce au calcul du temps de cycle [RA 80]. L'obtention de l'ordonnancement cyclique optimal ou sous-optimal est obtenu à partir d'algorithmes d'optimisation dont l'efficacité dépend généralement de l'heuristique choisie. En partant d'un modèle assez général permettant la représentation de ressources partagées et la flexibilité de routage, lorsque les taux de production sont fixés, en appliquant des algorithmes d'optimisation pour la recherche du cycle optimal, on arrive à lever progressivement tous les indéterminismes et à transformer le modèle général de départ en un graphe d'événements complètement déterministe [CA 96]. Il a aussi été montré qu'il existe une représentation algébrique des graphes d'événements et que les équations obtenues sont linéaires si l'on considère des algèbres non-conventionnelles [COH 89, DU 83]. Cette représentation algébrique permet en particulier de trouver le marquage initial du réseau tel que le temps de cycle soit minimum pour un régime stationnaire périodique. L'inconvénient d'une telle approche est qu'elle ne permet d'étudier que des systèmes ayant un fonctionnement complètement cyclique et répétitif. Or, une cellule flexible n'aura jamais un comportement strictement cyclique et répétitif. De plus, les graphes d'événements, n'autorisent pas la représentation de ressources partagées qui est une notion fondamentale de la flexibilité au sein d'une cellule. Par conséquent, dans le cas de cellules flexibles, il est préférable de considérer un modèle plus général que les graphes d'événements.

Dans la mesure où un modèle plus général qu'un graphe d'événements représente de façon agrégée un ensemble de séquences admissibles, nous pourrons parler de modélisation d'un ordonnancement flexible et dans ce cas, l'approche la plus naturelle fera intervenir deux étapes fondamentales [LE1 94, LE2 94, AZ 94]:

- La modélisation de l'ordonnancement flexible de l'atelier ou de la cellule à l'aide d'un réseau de Petri général t-temporisé [RA 80]. On représente en particulier les gammes de fabrication et les ressources flexibles.

- La recherche de l'ordonnancement optimal à l'aide d'un algorithme du type *Branch and Bound*. Cet algorithme fait une recherche, en parcourant le graphe des marquages accessibles, de la solution qui minimise le temps de cycle. Du fait de la rapide explosion combinatoire, des heuristiques sont généralement utilisées afin de diminuer le temps de calcul. La solution obtenue est alors optimale ou sous-optimale.

Le résultat de ces deux étapes est un ordonnancement explicite qui sera malheureusement difficilement exploitable lorsque l'on arrivera au niveau *coordination globale* de la cellule. Soulignons de plus que dans ce type d'approche, les transitions sont généralement tirées au plus tôt, ce qui peut facilement mener à un ordonnancement qui ne soit pas optimal comme l'ont montré *Silva et Valette* dans [SI 89] sur un exemple particulier.

Une autre solution qui parait mieux adaptée à la nature d'une cellule flexible est d'élaborer un ensemble de règles plus ou moins heuristiques qui définissent chaque fois

qu'une machine devient libre l'opération suivante à effectuer. Ces règles remplacent l'algorithme *Branch and Bound* très couteux en calcul et sont appliquées en temps réel au réseau de Petri t-temporisé au niveau de la coordination globale de l'atelier. On parle alors d'ordonnancement implicite. La faiblesse de cette approche réside bien sûr dans la difficulté d'obtenir de bonnes performances, ou du moins de comparer les performances obtenues vis-à-vis d'une solution optimale.

Les réseaux de Petri sont donc bien adaptés à la modélisation d'ordonnancements flexibles. Par contre, il reste encore beaucoup de travail à faire si l'on veut être capable de calculer en temps réel un ordonnancement qui soit à la fois robuste et efficace. Enfin, l'intérêt des réseaux de Petri étant justement de pouvoir conserver une certaine cohérence entre les divers niveaux d'un CIM, il serait intéressant pour la diminution des calculs que les résultats obtenus à l'étape d'évaluation de performance soient mieux exploités à l'étape d'ordonnancement.

Supervision et Pilotage en Temps Réel

Pour pouvoir surveiller et piloter une cellule, on doit pouvoir comparer à tout instant son état réel avec celui du modèle qui la représente. Le modèle doit prendre en compte toute la complexité du système. De ce fait, un réseau de haut niveau plus général qu'un réseau places/transitions doit généralement être considéré. On peut par exemple utiliser les réseaux prédicats/transitions [BA 90]. *Atabakhche* dans [AT1 87] définit un réseau de Petri à décision particulièrement bien adapté au pilotage temps réel. Ce modèle est dérivé des réseaux à objets [SB 94]. La mise en œuvre se fait à l'aide d'un joueur de réseau de Petri qui est un mécanisme d'inférence particulier appliqué aux réseaux de Petri [AT1 87, AT2 87, SI 89]. Cet algorithme résout en temps réel tous les conflits existant en exécutant l'ordonnancement explicite ou implicite qui a été défini à l'étape *ordonnancement*.

Les réseaux de Petri, en plus d'être utilisés dans l'exécution de l'ordonnancement temps réel, jouent un rôle essentiel au niveau de la surveillance des ateliers et des cellules. En effet, le réseau interprété décrit en temps réel le comportement dynamique du système. Il peut être utilisé comme modèle du procédé commandé dans le cadre du principe classique de la supervision/surveillance décrit figure 1.1 ([CO 91]). A chaque changement significatif de l'état du procédé (dans notre cas, essentiellement des fins d'opérations) le module de détection vérifie que l'état du modèle de l'atelier est compatible et effectue une mise à jour de son état. Le module de décision déduit alors quelle action (ou quelles actions) entreprendre. Il s'agira essentiellement de débuts d'opérations dans notre cas.

Si ce modèle d'atelier est un graphe d'événements, le module de décision ne pourra pas remettre en cause l'ordre d'exécution des opérations sur les ressources. Si ce modèle est un réseau avec conflit, alors c'est le module de détection qui aura du mal à travailler en cohérence avec un ordonnancement exprimé sous la forme d'un graphe d'événements.

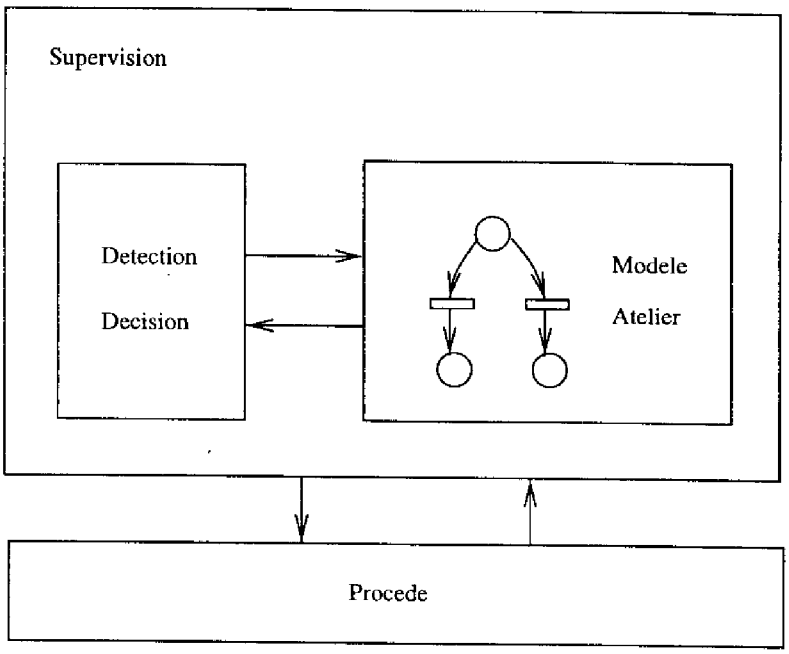

Figure 1.1: Supervision et Pilotage d'un Atelier

1.4.3 Limitations des Réseaux de Petri

Les réseaux de Petri présentent de nombreux avantages pour les cellules flexibles qui sont :

- La possibilité de définir un ordonnancement flexible dont les conflits sont résolus en temps-réel par le niveau supervision et pilotage. Le joueur de réseau de Petri permet de mettre en évidence et d'analyser l'indéterminisme résiduel dans l'état courant. Il doit être complété par des politiques de décision en temps réel.
- L'utilisation d'une même famille d'outils de spécification et de modélisation à pratiquement tous les niveaux d'un CIM. Il est ainsi beaucoup plus facile de garantir une certaine cohérence entre les divers niveaux du système et d'avoir une approche plus intégrée de l'ensemble des fonctions d'un CIM.

D'un autre côté, de nombreux efforts doivent encore être fait pour:

- avoir une approche plus structurée au niveau de la modélisation en considérant les différentes contraintes qui existent au sein d'un même réseau et en étant capable de contrôler les effets que peuvent avoir l'ajout ou le retrait d'une contrainte sur le reste du réseau aussi bien d'un point de vue qualitatif que quantitatif,
- avoir des algorithmes de calcul de performance plus efficaces qui considèrent aussi bien le régime stationnaire périodique que les régimes perturbés,

- avoir une approche de l'ordonnancement qui permette d'atteindre un meilleur compromis entre efficacité et flexibilité et être capable de l'associer de façon cohérente et efficace au pilotage en temps réel,
- être capable d'avoir une approche cohérente entre l'ordonnancement prévisionnel et le pilotage en temps réel,
- être capable de diminuer les temps de calcul de l'analyse des performances en intégrant mieux les résultats obtenus à chaque niveau.

Revenons sur le dernier point de cette énumération pour l'expliciter. Il est rare que les résultats de l'évaluation de performance obtenus pendant la conception soient utilisés pour l'ordonnancement, puis pour le pilotage en temps réel. Généralement, l'algorithme d'ordonnancement ignore totalement les résultats de l'analyse quantitative et recalcule inutilement des performances qui étaient déjà connues au sein du système intégré que représente le CIM.

Les réseaux de Petri ne peuvent exprimer que des contraintes de précédence au sens large (relations d'ordre partiel, exclusions mutuelles, synchronisations). S'ils permettent de mettre en évidence les situations de conflit, ils ne permettent pas de spécifier les politiques de décision. Les techniques d'intelligence artificielle sont par contre largement utilisées pour les problèmes d'aide à la décision.

Dans ce qui suit, nous allons présenter les principales techniques d'Intelligence Artificielle qui sont utilisées dans l'Ingéniérie de Production et qui peuvent nous aider à surmonter les limitations imposées par les réseaux de Petri au niveau conception et production des cellules flexibles.

1.5 Utilisation de l'Intelligence Artificielle dans le Cadre des Cellules Flexibles

1.5.1 Rappel sur les Langages de Programmation de Contraintes et sur les CSP (Constraint Satisfaction Problems)

Il existe aujourd'hui dans les domaines de l'Informatique et de l'Intelligence Artificielle une tentative de simplifier et de rendre plus naturelle l'écriture d'un programme en déchargeant le programmeur de la tâche de conception d'algorithmes spécifiques de résolution. C'est dans ce but qu'ont été développés des langages déclaratifs comme Prolog par exemple. Dans un langage impératif comme Pascal ou C, l'utilisateur doit mettre en œuvre un algorithme aboutissant à la solution de façon systématique et toujours de la même manière. Avec les langages déclaratifs, le programmeur doit seulement établir un ensemble de relations entre un ensemble d'objets, un moteur d'inférence plus ou moins général étant chargé d'exploiter ces informations pour obtenir une solution. Un type de langages de programmation déclaratifs appelés Langages de Programmation de Contraintes est particulièrement intéressant pour aider à la

décision dans des systèmes complexes tels que les ateliers flexibles. Avant tout, nous allons donner quelques définitions [LE 88]:

- Une contrainte exprime une relation entre un ensemble d'objets.
- Un langage de contraintes est un language utilisé pour décrire les objets et les contraintes.
- Un programme de contraintes est un programme écrit à partir d'un langage de contraintes. Ce programme va définir un ensemble d'objets et l'ensemble des contraintes qui leurs sont appliquées.
- Un système de résolution de contraintes (constraint satisfaction system) trouve les valeurs des objets qui vont vérifier les relations à l'aide de méthodes de résolution appelées techniques de résolution de contraintes (constraint satisfaction techniques).

On peut voir que dans ces définitions apparaissent deux parties distinctes : le programme de contraintes et le système de résolution de contraintes. On rencontre un fonctionnement similaire dans Prolog qui sépare les règles (le programme Prolog) du mécanisme de contrôle (moteur d'inférence de Prolog). Les avantages de cette séparation sont :

- La possibilité de programmer les contraintes et les règles sans connaître exactement le principe de résolution qui va être utilisé, évitant ainsi une perte de temps avec des questions qui ne concernent pas directement la formulation du problème.
- La possibilité de résoudre divers problèmes avec un même programme qui représente de façon simple les diverses contraintes que le système doit respecter.
- La simplicité de la modification d'un programme, en retirant ou ajoutant certaines contraintes.
- La facilité de décrire des systèmes complexes de façon simple et naturelle en utilisant seulement des informations locales.
- Simplifier la tâche humaine de description d'un problème à un ordinateur.

D'un autre côté, le désavantage d'un système de résolution de contraintes vient du fait que pour être efficace, il doit être très spécialisé, ce qui restreint son application à des systèmes qui ont des caractéristiques similaires. C'est la raison pour laquelle, il n'existe pas un système de résolution de contraintes générique capable de résoudre efficacement un problème quelconque. En particulier, le mélange de variables booléennes (ou énumératives) et de variables réelles pose souvent des problèmes.

Généralement, un programme de contraintes peut être vu comme un graphe [LE 88, GU 89] et le plus simple et le plus commun des mécanismes de résolution

de contraintes est appelé *propagation locale.* Les règles de résolution impliquent seulement des informations locales à chaque nœud du graphe et résultent en une cohérence locale des objets.

Les autres mécanismes de résolution de contraintes sont:

- les mécanismes de relaxation quand le graphe a des cycles et quand ses contraintes sont incohérentes (sans solution),
- les mécanismes de transformation de graphes,
- la programmation linéaire quand les contraintes sont linéaires,
- les mécanismes de résolution de l'Intelligence Artificielle comme les techniques de recherche utilisées par des démonstrateurs automatiques de théorèmes.

Des exemples de langages de programmation de contraintes qui sont aujourd'hui commercialisés comme CLP(R), Prolog III, CHIP V4 et ILOG Solver/Schedule sont présentés dans [ES 95].

Pour mieux exploiter la spécificité des problèmes posés par la programmation sous contrainte, l'approche CSP (Constraint Satisfaction Problems) offre à la fois un cadre de représentation général et rigoureux permettant de formuler simplement de nombreux problèmes combinatoires, et une palette de techniques de résolution de ces problèmes. Cette approche a comme principal avantage d'être totalement indépendante de la sémantique des contraintes et de pouvoir définir formellement, indépendamment du domaine d'application, ce qu'est un problème de décision sous contrainte. Les principales techniques de résolution de l'approche CSP sont :

- les techniques de filtrage par propagation de contraintes,
- le backtrack avec l'aide d'heuristiques afin de réduire la complexité des problèmes qui sont généralement NP-complet,
- la recherche de classes de problèmes polynomiaux et l'exploitation de ces classes pour l'élaboration de méthode de décomposition de problèmes.

Devant le caractère trop restrictif de l'approche CSP qui interdit la prise en compte des caractéristiques flexibles ou incertaines de certains problèmes, *H. Fargier* dans [FA 94] a défini le nouveau formalisme FCSP (Fuzzy Constraint Satisfaction Problems) qui généralise l'approche CSP, tant au niveau théorique qu'algorithmique. Cette nouvelle approche qui est fondé sur la théorie des ensembles flous et la théorie des possibilités autorise le traitement de contraintes souples ou à priorité, de contraintes incertaines, ainsi que la prise en compte de paramètres non-contrôlables.

1.5.2 Analyse sous Contraintes des Ateliers Flexibles

Même si le concept de flexibilité est relativement nouveau et informel, il se traduit souvent par la définition d'un ensemble de solutions acceptables vis à vis du problème considéré et non pas par la définition d'une solution unique et optimale comme le fait

généralement la recherche opérationnelle. Les solutions acceptables sont alors données au travers d'un ensemble de contraintes qui doivent être satisfaites [ER 88] quel que soit l'état du système. Ce moyen de décrire un atelier flexible comme un ensemble de contraintes a déjà été développé dans diverses approches telles que:

- ISIS [FO 87],
- OPIS [SM 88],
- MASCOT [ES 87],
- MASCOT 2 [LO 91].

Par exemple, MASCOT, qui a été implémenté en Prolog II, a pour tâche de résoudre l'ordonnancement d'un atelier flexible. L'écriture des clauses de Horn en Prolog II correspond aux contraintes d'un programme de contraintes et le moteur d'inférence de Prolog II correspond au système de résolution de contraintes. Il s'agit ici d'analyse sous contraintes de problèmes d'ordonnancement [ER 76] dans la mesure où l'objectif n'est pas de générer une solution unique mais plutôt un ensemble de solutions qui caractérise une famille d'ordonnancements respectant les contraintes du problème indépendamment de tout critère. Les contraintes prises en compte portent sur la disponibilité des ressources et sur les intervalles de temps alloués. La recherche est axée sur l'obtention de conditions nécessaires d'admissibilité que tout ordonnancement solution du problème doit satisfaire. Cette démarche n'aboutit pas à la caractérisation de l'ensemble des ordonnancements admissibles, mais à celle d'un ensemble d'ordonnancements contenant tous les ordonnancements admissibles.

Les travaux qui ont permis la réalisation du logiciel MASCOT ont été enrichis au niveau de MASCOT II par une nouvelle approche qui réside dans l'utilisation de contraintes de nature énergétique c'est-à-dire de variables qui sont obtenues par le produit d'un intervalle de temps par des ressources. Elles permettent de représenter de façon agrégée des caractéristiques temporelles et de ressources.

Il peut être intéressant d'établir que la modélisation de problèmes d'ordonnancement à l'aide de graphes non-conjonctifs [LO 91] comme c'est le cas dans MASCOT II sont en fait des CSP et que les règles d'analyse sous contraintes utilisées en ordonnancement peuvent être interprétées vis à vis des méthodes de filtrage des CSP en terme de cohérence locale.

Le modèle FCSP aussi a été appliqué aux problèmes d'ordonnancement de production de type ateliers à tâches (job-shop). Il permet en particulier de représenter des durées d'opérations flexibles ou imprécises.

Par rapport à notre problème qui est l'étude et l'analyse de cellules flexibles au niveau conception et production, ces approches présentent un certain nombre d'inconvénients. En effet, les applications actuelles de l'analyse sous contraintes concernent essentiellement des ateliers non cycliques. On a un ensemble de tâches avec des dates au plus tôt et au plus tard sur lesquelles on raisonne. Dans le cas de cellules flexibles, on travaillera plutôt à partir de taux de production à respecter. Dans un graphe non conjonctif, il faut autant de nœuds que de tâches. Par exemple, si un type de pièce doit être fabriqué n fois, il faudra n nœuds pour chacune des opérations nécessaires.

Un graphe non conjonctif est donc peu adapté au pilotage temps réel de cellules avec un fonctionnement cyclique.

Tout ceci doit nous amener à considérer des approches mixtes qui combinent les réseaux de Petri et les techniques de l'Intelligence Artificielle utilisées dans l'Ingéniérie de Production.

1.6 Approches Hybrides dans le Cadre de Cellules Flexibles

1.6.1 Réseaux de Petri et Systèmes de Règles

Un réseau de Petri peut être considéré comme un système de règles ou système de production de connaissance [VA 91] formé:

- d'une base de faits (contexte courant et base de règles),
- d'un mécanisme d'inférence.

Le mécanisme d'inférence le plus élémentaire consiste à ranger l'ensemble des règles dans une liste et à parcourir cette liste séquentiellement. Dès que la condition d'une règle est vraie dans le contexte courant (faits vérifiés au moment de l'inférence), la règle est appliquée. Généralement, dans un contexte donné, plusieurs règles peuvent être applicables et le résultat de la déduction peut être différent suivant que l'on choisit d'appliquer d'abord telle ou telle règle ou de les appliquer toutes simultanément. Si l'on considère par exemple Prolog, le mécanisme élémentaire choisit la première rencontrée sans même détecter que d'autres possibilités existent. De façon générale, lorsque plusieurs règles sont applicables, on parle alors de conflit et c'est à l'aide d'un contrôle que celui-ci est résolu.

Dans un réseau de Petri:

- l'ensemble des transitions avec leurs règles de franchissement sont la base de règles,
- le marquage initial est le contexte initial,
- la résolution des conflits fondée sur les notions de transitions parallèles (l'ordre de franchissement est indifférent), de transitions en conflit (le résultat dépendra de l'ordre des franchissements) et sur le franchissement des transitions correspondent au mécanisme d'inférence.

L'une des motivations pour l'utilisation de l'Intelligence Artificielle dans le cadre des cellules flexibles est d'aider à prendre des décisions lors du pilotage en temps réel de ces dernières. Au niveau d'un réseau de Petri, ces décisions correspondent essentiellement à la résolution des conflits. Il peut alors être intéressant d'utiliser le formalisme des systèmes de règles pour mettre en œuvre aussi bien le joueur de réseaux de Petri que le mécanisme d'aide à la décision. Certains travaux ont déjà suivi

cette approche [AT1 87, AT2 87, BA 90]. Toutefois, ils n'ont pour l'instant développé que la mise en œuvre du joueur sous la forme d'un système de règles. Le mécanisme d'aide à la décision pour la résolution des conflits n'a pas été détaillé.

1.6.2 Réseaux de Petri et Logique

Toujours dans le contexte conception et production de cellules flexibles, en plus de la mise en œuvre d'un réseau de Petri sous la forme d'un système de règles pour aider à résoudre les conflits pendant le pilotage temps réel, nous sommes tout particulièrement concernés par l'aide au dimensionnement et à la conception d'une politique de conduite (un ordonnancement flexible). Pour cela, il faut raisonner sur le réseau de Petri qui est le mieux approprié pour modéliser l'ensemble des contraintes existant au sein d'une cellule flexible. Par conséquent, dans le cadre d'approches hybrides combinant réseaux de Petri et techniques d'Intelligence Artificielle, nous devons aller au delà d'une simple similitude syntaxique avec les systèmes de règles et se poser le problème des relations existant entre la logique et les réseaux de Petri.

Il y a déjà plusieurs années que la parenté qui existe entre les réseaux de Petri et les systèmes de règles a été considérée par de nombreux chercheurs. Néanmoins, ce n'est que récemment qu'ont été formalisés certains résultats qui établissent les liens directs qui existent entre les réseaux de Petri places/transitions et la logique propositionnelle et entre les réseaux de Petri prédicats/transitions et la logique du premier ordre. *T. Murata* et *G. Peterka* ont prouvé que certains programmes logiques pouvaient être traduits en un réseau de Petri prédicats/transitions [MU 88, PE 89]. Parallèlement, de nombreux articles combinant les réseaux de Petri et les techniques d'Intelligence Artificielle dans le domaine des *CIM* ont été publiés [VA 93]. Le résultat le plus significatif dans ce domaine a sûrement été celui qui a montré que la logique classique n'était pas cohérente avec le concept de ressource [GI 87] qui est justement une des notions fondamentales de la théorie des réseaux de Petri. Si on considère le calcul des séquents en logique classique, on se rend compte que deux règles structurelles sont responsables de cette incohérence:

- la règle d'affaiblissement (celle qui correspond à la notion de monotonie) qui dit que si A implique B alors A et C implique B,
- la règle de contraction qui dit que si A et A implique B alors nous pouvons simplement dire que A implique B.

Si les propositions A, B et C sont interprétées comme des ressources, l'affaiblissement veut dire que les ressources peuvent disparaître sans être utilisées et la contraction veut dire qu'il n'y a pas de différence entre avoir une ou deux ressources.

Pour prendre en compte les ressources, une nouvelle logique (fondée sur un nouveau calcul des séquents) a été proposée par *J.Y. Girard* [GI 87]: la logique linéaire. Avec cette logique, de la proposition "A implique B" ne peut être déduit que la proposition "A et C implique B et C". De la même façon, la règle de contraction n'est plus valide. Il a été montré que les réseaux de Petri correspondaient au fragment multiplicatif de cette nouvelle logique [BR 89, GUN 89].

Les CSP ont été développés dans le cadre de la logique classique. Pour aborder la gestion et le pilotage des ateliers il faut manipuler des ressources. A cause de l'incohérence entre la notion de ressource et la logique classique, il peut être avantageux d'associer un modèle de type réseau de Petri à un CSP. Souvent des modèles graphiques complémentaires sont utilisés soit pour écrire les équations décrivant les contraintes [RI 95], soit pour guider les calculs effectués dans le formalisme CSP. Par exemple on peut utiliser les graphes potentiel-tâches [ES 87, LO 91] si l'on considère un nombre fini de tâches sur un horizon donné. Nous pouvons également décrire les utilisations de ressources sous la forme d'un réseau de Petri. Nous venons de voir que cela revient à utiliser une logique plus adéquate que la logique classique pour les contraintes portant sur les ressources. C'est cette approche que nous retiendrons dans le contexte des cellules flexibles où le concept de ressource partagée est fondamental. Si nous considérons que généralement les sémantiques d'un programme logique et d'un réseau de Petri sont différentes, il pourra être intéressant de combiner les deux approches afin d'obtenir un modèle plus compréhensible en particulier dans les domaines qui nous concernent: le dimensionnement, la conception des politiques de conduite et le pilotage des cellules flexibles.

1.6.3 Réseaux de Petri et Formalismes de Contraintes

Même si les approches hybrides qui mettent en jeu les réseaux de Petri et l'Intelligence Artificielle sont de plus en plus utilisées au niveau des ateliers de fabrication, les réseaux de Petri ont rarement été directement considérés comme un formalisme de contraintes. Ce n'est que très récemment que certains chercheurs ont commencé à aborder des approches combinant par exemple les réseaux de Petri et les langages de programmation de contraintes.

R. Pascal dans [RI 95] transcrit un réseau de Petri dans le langage de programmation logique avec propagation de contraintes CHIP [CO 93]. L'objectif est de modéliser des problèmes d'ordonnancement avec des réseaux de Petri t-temporisés [CA 87] sans se préoccuper de l'outil de résolution. Le réseau de Petri est décomposé et représenté sous forme de clauses CHIP. Le programme calcule alors l'ordonnancement minimisant la durée globale d'opération (makespan). L'avantage de cette approche est sa souplesse: on peut ajouter et relaxer des contraintes, ce qui laisse des possibilités d'intervention de l'utilisateur sur le processus de décision. En revanche, il est difficile d'évaluer l'efficacité de résolution d'une telle approche qui peut mettre en jeu des temps de calcul plus ou moins importants selon la taille du réseau.

Yim dans [YI 95] définit un nouveau modèle mathématique qui cherche à unifier d'une part les réseaux de Petri et d'autre part les CSP. A partir de ce modèle, un système appelé MINOS est utilisé pour implémenter en Prolog III un algorithme de recherche de séquences basé sur l'arbre des marquages accessibles et qui peut être utilisé pour l'ordonnancement. Il y a bien sûr le problème de l'explosion combinatoire associée à ce type d'approche puisqu'il n'y a pas de stratégie de recherche particulière. Pour trouver une solution vérifiant toutes les contraintes il faudra forcément un très grand nombre de retours en arrière ("backtrack").

Il est sûrement intéressant d'aller plus loin et de ne pas simplement s'arrêter à

la simple traduction d'un réseau de Petri en un CSP. Il peut être plus efficace de travailler directement sur le réseau de Petri en exploitant ses caractéristiques propres et en isolant par exemple certains sous-réseaux spécifiques du modèle global à partir desquels nous pourrons déduire tout un ensemble d'équations linéaires. Il sera alors très simple de travailler à partir de ces contraintes linéaires en utilisant par exemple la programmation mathématique pour établir des conditions nécessaires de réalisation de la production qui réduiront le domaine de recherche des solutions admissibles. De plus, en appliquant directement des techniques de résolution de contraintes au réseau nous pourrons à la fois évaluer et optimiser l'efficacité de résolution de l'approche adoptée.

1.7 Intégration des diverses fonctions d'un CIM

Lorsque l'on se place dans le contexte de l'ingéniérie de production, on voit que le principal problème est d'être capable de proposer une méthodologie complète de conception, d'analyse et de commande des systèmes flexibles de production manufacturière. Devant la difficulté du problème à résoudre, on est généralement amené à étudier séparément les différentes fonctions du système informatique chargé d'assister l'ensemble des étapes du cycle de vie du système de production alors qu'il existe en réalité un haut degré de corrélation entre toutes ces étapes. Parmi les nombreux projets qui cherchent à intégrer au mieux le plus de fonctionnalités d'un CIM nous pouvons citer:

- Le système ANALYTICE du CEPGEI/CEFET (Curitiba-Brésil) [TA] d'aide au développement d'ateliers flexibles. Les principales activités de ce système interviennent dans la phase de spécification des produits ainsi que dans la phase de conception préliminaire du système de production au travers de techniques de simulation pour l'évaluation des performances. En particulier, ce travail recouvre plutôt les fonctions intervenant au niveau des *CAD* et *CAE*.

- La méthode GRAI de l'université de Bordeaux I qui propose une méthode structurée pour l'analyse et la conception des systèmes décisionnels de gestion de production [RO 93]. Cette approche propose des outils graphiques qui permettent de mener une analyse du système décisionnel d'une entreprise et de tirer un modèle des activités. On est donc à l'interface entre la conception et l'ordonnancement.

- L'outil informatique d'ordonnancement ORDO initialement développé au LAAS et qui vise à caractériser un ensemble admissible de solutions d'ordonnancement par l'utilisation de contraintes suffisantes d'admissibilité [BI 93] et à son utilisation pour le pilotage en temps réel.

- Le projet SCOOP développé au LAAS qui est un système interactif d'aide à la décision pour l'ordonnancement d'atelier [ES 94] et basé sur l'analyse sous contrainte. Ici aussi, on est d'une certaine manière entre l'ordonnancement prévisionnel et le pilotage.

- Les travaux du LAG (Laboratoire d'Automatique de Grenoble) sur la problématique de la conduite des systèmes flexibles de production manufacturière avec en particulier une approche mixte objets/réseaux de Petri qui est fondée sur un modèle de représentation de la commande des ateliers flexibles basé sur une classe enrichie des réseaux de Petri: les réseaux de Petri de commande orientés objet. Il s'agit de l'intégration des niveaux ordonnancement et pilotage en temps réel, plutôt du point de vue informatique.

- Dans le même cadre, on peut citer la thèse de Cyril Briand [BR 94].

- L'approche CODECO (conduite décentralisée coordonnée d'ateliers) qui s'appuie sur la définition d'une structure de commande basée sur une décomposition en plusieurs niveaux de décision [KA 85].

- L'approche: conduite hiérarchisée par la méthode de la commande des flux [LO 93] qui préconise la définition d'une structure de commande organisée en trois niveaux hiérarchiques: un niveau ordonnancement prévisionnel, un niveau ordonnancement temps réel et un niveau supervision.

- Les travaux du Rensselaer Polytechnic Institute of Troy dont l'approche proposée se base sur une modélisation par réseaux de Petri de bas niveau [ZH 92] pour obtenir un réseau de commande qui fournit toutes les bonnes propriétés indispensables à la cohérence de la conduite. Il s'agit plutôt d'intégrer la commande locale avec le pilotage.

- Le projet CASPAIN (conception assistée de systèmes de production automatisés en Industrie Manufacturière) du LAIL (Laboratoire d'Automatique et d'Informatique Industrielle de Lille) qui a pour objectif de proposer une méthodologie complète d'analyse, de conception et d'implémentation des systèmes flexibles de production manufacturière [CA 97].

La principale difficultée des approches citées est de définir les principes qui permettent d'assurer un développement cohérent et rigoureux d'un système de conduite d'atelier. Une règle impérative à observer est l'utilisation d'une méthodologie formelle pour le développement du CIM. L'exemple de l'approche CASPAIN illustre bien ce principe. Dans cette approche, initialement les niveaux de gestion de production ont été sous-estimés. Ceci a abouti à un système qui ne répondait que partiellement aux besoins. Pour améliorer l'approche, l'équipe CASPAIN a ensuite utilisé les réseaux de Petri de haut niveau en tant que modèle de la commande et s'est orientée vers un contexte de gestion de production réel en proposant une surveillance des défaillances et un ordonnancement prévisionnel de l'ensemble des opérations à mettre en oeuvre [TO 92, EL 93, AU 94, OH 95, CA 97]. Même si l'approche tend à être plus formelle, elle atteint néanmoins assez vite certaines limitations. Par exemple Ausfelder dans [AU 94] présente une méthode pour la modélisation hiérarchique de la commande des systèmes flexibles de manufacture basée sur l'utilisation d'un réseau de Petri de haut niveau qui permet la représentation de ressources partagées. D'un autre côté, toujours dans le cadre du même projet CASPAIN, Camus dans [CA 97] propose une

approche pour la conduite de systèmes flexibles de production qui se base sur le calcul préalable d'un ordonnancement prévisionnel cyclique optimal et sur la représentation de ce dernier à l'aide de graphes d'événements. Ce même graphe d'événements est alors utilisé comme modèle de pilotage et perd justement ce que le modèle de Ausfelder avait de bon c'est-à-dire la représentation des comportements flexibles décrits par les ressources partagées. En transformant progressivement un modèle qui comprenait des indéterminismes (les ressources partagées) en un modèle complètement déterministe et optimal (dans le cas des graphes d'événements tous les conflits sont résolus), on a forcément une perte d'information qui serait pourtant nécessaire en cas de fonctionnement perturbé (ce qui est souvent le cas des systèmes réels) pour lequel le système de prise de décision temps réel doit pouvoir réagir instantanément en se basant sur l'ensemble des états admissibles du système. De plus, les graphes d'événements de Camus perdent l'avantage du modèle plus général d'Ausfelder qui permet le découplage des contraintes de gammes et des contraintes de ressources. En effet, les contraintes de gammes doivent être respectées même en fonctionnement perturbé alors que les contraintes de ressources peuvent être modifiées, en particulier en ce qui concerne l'ordre des opérations sur les ressources. Lorsque l'ordonnancement est donné sous la forme d'un graphe d'événements, on peut imaginer que le passage du fonctionnement optimal au fonctionnement dégradé se fait en passant d'un graphe d'événement à un autre. Ce passage est simple s'il se produit en début de cycle et que le début de cycle correspond à un état où aucune opération n'est en cours. Dans le cas contraire, il pose de sérieux problèmes. Supposons par exemple que l'on ait un premier graphe d'événements qui décrive la séquence d'opérations suivante pour une machine M donnée:

$o_{1,1}, o_{1,2}|o_{2,1}, o_{2,2}$

Les deux premières opérations sont appliquées à des pièces de type p_1 sur une première gamme de fabrication et les deux dernières à des pièces de type p_2 sur une deuxième gamme. Supposons qu'à la fin de l'opération $o_{1,2}$ une perturbation se produise au sein de l'atelier et que l'on soit forcé de commuter sur un nouveau graphe d'événements pour lequel la nouvelle séquence d'opération calculée soit:

$o_{1,1}|o_{2,1}, o_{2,2}|o_{1,2}$

Comme les graphes d'événements ne découplent pas les contraintes de gammes des contraintes de ressources il ne sera pas possible de passer instantanément d'un graphe à l'autre. En effet dans le deuxième graphe, il n'y a pas d'état pour lequel $o_{1,1}$ et $o_{1,2}$ sont terminées et $o_{2,1}$ et $o_{2,2}$ pas commencées. Il faudra forcément considérer un transitoire de durée non négligeable et pas simple à calculer. C'est pour ces raisons que dans notre travail, nous chercherons à utiliser exactement le même modèle pour l'ordonnancement et pour le pilotage en temps réel, et que ce modèle ne sera pas un graphe d'événements.

L'obtention d'une commande à la fois robuste et efficace ne sera possible que dans la mesure où il existera une grande réactivité non seulement entre les divers niveaux de commande du CAM mais aussi entre toutes les fonctionnalités du CIM. Par exemple, les raisons d'un ordonnancement aux performances peu satisfaisantes peuvent être aussi bien issues d'une mauvaise étape de planification que d'un mauvais choix lors de la définition des familles de produits au niveau du CAD. Ceci nous amène

à justifier l'approche que nous allons présenter dans ce mémoire qui a pour objectif principal l'intégration de la plupart des fonctionnalités d'un CIM à partir d'un même modèle. Nous nous limiterons aux parties CAE et CAM du CIM. Le modèle devra être suffisamment expressif pour modéliser une cellule flexible dans toute sa complexité et posséder des techniques d'analyse qui pourront répondre aux divers besoins de l'ensemble des fonctionnalités mises en jeux.

Comme nous l'avons dit précédemment, les réseaux de Petri autorisent la représentation de ressources partagées. Il est ainsi possible de modéliser de façon agrégée un ordonnancement flexible qui, plutôt que de représenter une séquence d'opérations optimale, définit un ensemble de solutions acceptables qui caractérisent le comportement flexible du système. Ils permettent aussi le calcul d'invariants de places et de transitions qui sont deux propriétés essentielles lorsque l'on considère des cellules à fonctionnement cyclique.

Les techniques de l'Intelligence Artificielle possèdent des mécanismes de retour arrière ("backtracking") indispensables pour l'exploration systématique d'un ensemble de trajectoires afin de savoir si ces dernières vérifient ou non les contraintes du problème. De plus, les langages utilisés en Intelligence Artificielle et particulièrement les langages de programmation de contraintes donnent une représentation simple et naturelle de certaines contraintes qui peuvent exister au sein d'une cellule flexible. Ils possèdent aussi des techniques d'analyse et de résolution de contraintes extrêmement efficaces.

Il semble que l'analyse sous contraintes soit une approche très prometteuse lorsque sont considérés des systèmes possédants de hauts degrés de flexibilité. Toutefois, comme nous venons de le voir, la notion de ressource consommable et partageable entre des opérations n'est pas très cohérente avec les propositions de la logique classique. Nous allons donc nous baser dans la suite de ce travail sur une approche hybride qui combine d'une part les réseaux de Petri pour la partie modélisation et analyse qualitative, et d'autre part les techniques de résolution de l'Intelligence Artificielle pour tout ce qui est rattaché à l'analyse sous contrainte des cellules flexibles ou du moins pour l'exploitation des contraintes linéaires que l'on pourra éventuellement déduire du modèle.

La suite de ce travail va être consacrée à la présentation d'une telle approche pour l'aide au dimensionnement, l'aide à l'élaboration de politiques de contrôle et le pilotage en temps réel de cellules flexibles.

Chapitre 2

Modélisation d'une Cellule Flexible

Nous avons vu dans le premier chapitre qu'un besoin important est d'avoir un modèle si possible unique et utilisable pour la plupart des phases du cycle de vie du système. Nous allons donc présenter une approche pour construire un modèle susceptible de servir pour la conception (dimensionnement), l'élaboration d'une politique de gestion et le pilotage en temps réel. Comme c'est un problème très complexe, nous nous restreindrons au cas d'une cellule dans un atelier manufacturier pour un fonctionnement cyclique.

2.1 Modélisation des Contraintes de Synchronisation

En réponse à tout ce qui a été dit précédemment, nous choisissons d'avoir une approche modulaire en ne considérant que des informations locales qui seront vues comme un ensemble de contraintes, chaque contrainte représentant seulement un aspect partiel du système. De plus, pour pouvoir laisser la place au décideur humain, pour pouvoir utiliser des approches de type Intelligence Artificielle (analyse sous-contraintes) sur le modèle et conserver de la flexibilité au niveau du pilotage temps réel, ce dernier devra comprendre une part d'indéterminisme (c'est-à-dire laissant de la flexibilité). Le modèle choisi sera donc un réseau de Petri plus général qu'un graphe d'événements.

Après avoir donné la définition des gammes et des sous-gammes de fabrication, nous présenterons les divers types de contraintes qui peuvent exister au sein d'une cellule.

2.1.1 Gamme de Fabrication

Dans une cellule flexible, il existe une ou plusieurs gammes de fabrication qui permettent de réaliser un même produit. Une gamme de fabrication décrit la séquence des opérations qui doivent être effectuées sur une pièce afin de réaliser un produit.

On peut la définir de la façon suivante:

soit le réseau de Petri ordinaire non marqué $g =< S_g, O_g, Pre_g, Post_g >$ [DA 92], g définit une gamme de fabrication si et seulement si:

- $O_g = \{o_{g,1}, o_{g,2}, ..., o_{g,n_g}\}$ est un ensemble fini et non vide de n_g transitions avec n_g le nombre d'opérations discrètes associées à la gamme g.
- $S_g = \{s_{g,1}, s_{g,2}, ..., s_{g,n_g}, s_{g,n_g+1}\}$ est un ensemble fini et non vide de $n_g + 1$ places avec $n_g + 1$ le nombre de stocks associés à la gamme g.
- $Pre_g : S_g \times O_g \rightarrow \{0, 1\}$ est l'application d'incidence avant telle que pour $i \neq j$, $Pre_g(s_{g,i}, o_{g,i}) = 1$ et $Pre_g(s_{g,j}, o_{g,i}) = 0$.

  Ceci signifie qu'il y a un stock $s_{g,i}$ à l'entrée de chaque opération $o_{g,i}$.
- $Post_g : S_g \times O_g \rightarrow \{0, 1\}$ est l'application d'incidence arrière telle que pour $i \neq j$, $Post_g(s_{g,i+1}, o_{g,i}) = 1$ et $Post_g(s_{g,j+1}, o_{g,i}) = 0$.

  Ceci signifie qu'il y a un stock $s_{g,i+1}$ à la sortie de chaque opération $o_{g,i}$.

On interprète la transition $o_{g,i}$ comme la i^{eme} opération d'usinage ou de transport de la gamme g et la place $s_{g,i}$ comme le i^{eme} stock de la gamme g dans lequel sont déposés les pièces en attente de subir l'opération associée à la transition $o_{g,i}$. Lorsque nous aurons plus d'une gamme par cellule, nous noterons la première gamme $g1$, la deuxième $g2$ etc.

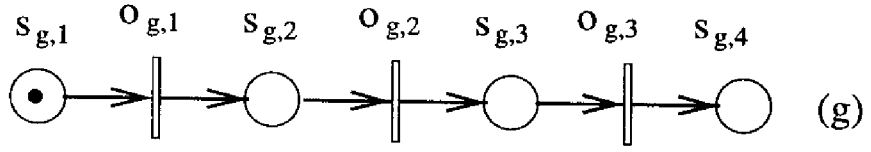

Figure 2.1: Gamme de Fabrication g

Le réseau de la figure 2.1 représente la suite des opérations de la gamme g qui doivent être effectuées sur la pièce modélisée par le jeton se trouvant dans le buffer d'entrée $s_{g,1}$. Le produit fini sera déposé dans le buffer de sortie $s_{g,4}$. $o_{g,1}, o_{g,2}$ et $o_{g,3}$ représentent les opérations et $s_{g,2}$ et $s_{g,3}$ les stocks intermédiaires. Soulignons que ce réseau ne décrit pas de façon explicite un intervalle de temps entre les opérations $o_{g,1}$ et $o_{g,2}$ par exemple. Il explicite uniquement le fait que la date de début d'opération de $o_{g,2}$ doit être supérieure à la date de fin d'opération de $o_{g,1}$.

2.1.2 Sous-Gamme de Fabrication

Une sous-gamme de fabrication est définie de la façon suivante:

soit le réseau de Petri ordinaire non marqué $sg =< S_{sg}, O_{sg}, Pre_{sg}, Post_{sg} >$, sg définit une sous-gamme de fabrication d'une gamme g si et seulement si:

- $O_{sg} = \{o_{g,m_{sg}+1}, o_{g,m_{sg}+2}, \ldots, o_{g,m_{sg}+n_{sg}}\}$

  est un ensemble fini et non vide de n_{sg} transitions de O_g avec $m_{sg} \in N$ et $n_{sg} \in N^*$ ($O_{sg} \subseteq O_g$).

  n_{sg} représente le nombre d'opérations discrètes associées à la sous-gamme sg.

  $m_{sg} + 1$ représente le numéro de la première opération de la sous-gamme sg à l'intérieur de la gamme g.

- $S_{sg} = \{s_{g,m_{sg}+2}, s_{g,m_{sg}+3}, \ldots, s_{g,m_{sg}+n_{sg}}\}$

  est un ensemble fini de $n_{sg} - 1$ places de S_g ($S_{sg} \subseteq S_g$).

  $n_{sg} = 1$ correspond au cas où cet ensemble est vide.

  $n_{sg} - 1$ représente le nombre de stocks intermédiaires associés à la sous-gamme sg.

  $m_{sg} + 2$ représente le numéro du premier stock intermédiaire de la sous-gamme sg à l'intérieur de la gamme g.

- $Pre_{sg} : S_{sg} \times O_{sg} \rightarrow \{0, 1\}$ est l'application d'incidence avant telle que pour $i \neq j$, $Pre_{sg}(s_{g,i}, o_{g,i}) = 1$ et $Pre_{sg}(s_{g,j}, o_{g,i}) = 0$.

  Ceci signifie qu'il y a un stock $s_{g,i}$ à l'entrée de chaque opération $o_{g,i}$ à l'exception de la première opération $o_{g,m_{sg}+1}$ de la sous-gamme sg.

- $Post_{sg} : S_{sg} \times O_{sg} \rightarrow \{0, 1\}$ est l'application d'incidence arrière telle que pour $i \neq j$, $Post_{sg}(s_{g,i+1}, o_{g,i}) = 1$ et $Post_{sg}(s_{g,j+1}, o_{g,i}) = 0$.

  Ceci signifie qu'il y a un stock $s_{g,i+1}$ à la sortie de chaque opération $o_{g,i}$ à l'exception de la dernière opération $o_{g,m_{sg}+n_{sg}}$ de la sous-gamme sg.

La différence fondamentale entre une gamme et une sous-gamme est qu'une gamme commence par une place "stock d'entrée" et termine par une place "stock de sortie" alors qu'une sous-gamme commence par une transition opération et termine par une autre transition opération (les transitions d'entrée et de sortie sont les mêmes lorsque la sous-gamme se réduit à une seule opération). Lorsque nous considèrerons plus d'une sous-gamme associée à la gamme $g1$ par exemple, nous noterons la première sous-gamme $s1g1$, la deuxième $s2g1$ etc.

Les réseaux de la figure 2.2 représentent deux exemples de sous-gammes $s1g$ et $s2g$ obtenues à partir de la gamme g de la figure 2.1.

2.1.3 Contrainte Kanban

Si nous considérons le modèle de la sous-gamme de fabrication que nous avons défini, un nombre illimité de pièces peuvent se trouver dans un même stock intermédiaire. Il est évident que pour une cellule réelle nous devons prendre en compte le fait que les stocks intermédiaires ont une capacité de stockage limitée. De plus, si ces stocks sont indispensables pour assurer une certaine flexibilité (un blocage sur une machine amont ne se traduit pas immédiatement par un blocage de la machine

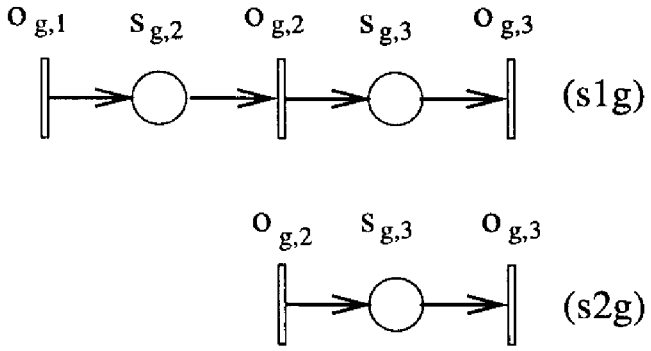

Figure 2.2: Sous-Gammes de Fabrication $s1g$ et $s2g$

aval), ils ont un coût. Enfin, dans les ateliers d'usinage les pièces sont en général fixées sur des palettes et ces palettes sont en nombre limité. Les politiques de type Kanban [MA 91] étant bien appropriées pour réguler les stocks intermédiaires d'une cellule flexible, nous pouvons considérer une contrainte réunissant dans un même sous-réseau une sous-gamme de fabrication et une politique Kanban que nous définissons de la façon suivante:

soit le réseau de Petri ordinaire marqué $C_k =< S_k, O_k, Pre_k, Post_k, M_k >$,

C_k associe une sous-gamme de fabrication sg à une politique Kanban si et seulement si:

- $S_k = S_{sg} \cup \{s_k\}$
- $O_k = O_{sg}$
- $$Pre_k \begin{cases} Pre_{sg} : S_{sg} \times O_{sg} \rightarrow \{0,1\} \\ Pre_{s_k} : \{s_k\} \times O_{sg} \rightarrow \{0,1\} \end{cases}$$

  est l'application d'incidence avant telle que pour $i \neq m_{sg} + 1$,

  $Pre_{s_k}(s_k, o_{g,m_{sg}+1}) = 1$ et $Pre_{s_k}(s_k, o_{g,i}) = 0$
- $$Post_k \begin{cases} Post_{sg} : S_{sg} \times O_{sg} \rightarrow \{0,1\} \\ Post_{s_k} : \{s_k\} \times O_{sg} \rightarrow \{0,1\} \end{cases}$$

  est l'application d'incidence arrière telle que pour $i \neq m_{sg} + n_{sg}$,

  $Post_{s_k}(s_k, o_{g,m_{sg}+n_{sg}}) = 1$ et $Post_{s_k}(s_k, o_{g,i}) = 0$
- $M_k : S_k \rightarrow N$ est l'application marquage initial telle que

  $M_k(s_k) = m_{s_k}$ et $M_k(s_{g,i}) = 0$ pour $i \in \{m_{sg} + 2, ..., m_{sg} + n_{sg}\}$.

  m_{s_k} est le nombre d'étiquettes Kanban qui se trouvent dans la place s_k.

Nous noterons la première place Kanban de la cellule s_{k1}, la deuxième s_{k2} etc.

Pour que la première opération $o_{g,m_{sg}+1}$ de la sous-gamme sg soit exécutée, il faut qu'un espace soit disponible dans l'atelier, c'est-à-dire qu'au moins un jeton se trouve dans s_k. C'est donc le marquage initial de s_k qui va définir la capacité des stocks intermédiaires de la sous-gamme sg.

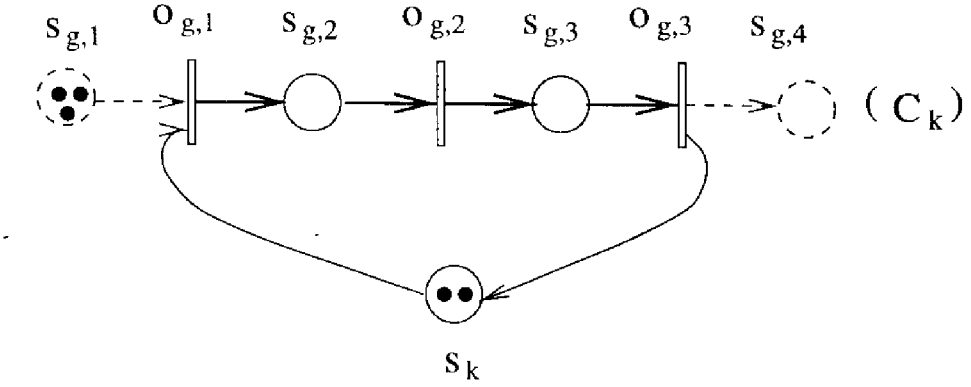

Figure 2.3: Contrainte Kanban C_k

Si nous considérons la figure 2.3, les deux jetons se trouvant dans la place s_k représentent le nombre de pièces du buffer d'entrée $s_{g,1}$ de la gamme g qui peuvent être simultanément traitées. Ces étiquettes Kanban doivent être vues comme des signaux de contrôle qui autorisent le traitement de nouvelles pièces en fonction des capacités des stocks intermédiaires $s_{g,2}$ et $s_{g,3}$ de la sous-gamme $s1g$. Les stocks $s_{g,2}$ et $s_{g,3}$ seront bornés à 2.

2.1.4 Contrainte de Ressource

Comme nous l'avons dit dans le premier chapitre, si l'ordre des opérations est complètement défini alors le réseau de Petri qui modélise cette contrainte est un graphe d'événements [HI 89, RA 80]: chaque place a seulement un arc d'entrée et un arc de sortie. Dans cette classe de réseaux de Petri, il n'y a pas de conflits et donc pas de décision à prendre. Le franchissement au plus tôt des transitions franchissables mène toujours à la séquence la plus courte. Pour exprimer un fonctionnement plus flexible, il faut pouvoir associer différentes opérations à un même ensemble de ressources sans définir l'ordre dans lequel ces opérations seront réalisées afin de pouvoir définir un ordonnancement flexible. La contrainte qui décrit une telle politique de contrôle peut être définie de la façon suivante:

soit le réseau de Petri ordinaire marqué $C_r =< S_r, O_r, Pre_r, Post_r, M_r >$,
C_r définit une contrainte de ressource si et seulement si:

- $S_r = \bigcup_{\alpha=1}^{n_r} S_{(sg)_\alpha} \cup \{s_r\}$

  avec s_r une place ressource et n_r le nombre des sous-gammes $(sg)_\alpha$ attachées à la place ressource s_r (i.e. le nombre de sous-gammes à considérer au niveau de cette contrainte).

Les sous-gammes concernées par le partage d'une ressource n'appartiennent pas nécessairement toutes à des gammes différentes. Toutefois, si deux sous-gammes $(sg)_1$ et $(sg)_2$ appartiennent à une même gamme, elles doivent être disjointes. Dans tous les cas nous avons donc:

$S_{(sg)_i} \cap S_{(sg)_j} = \emptyset$ pour $i \neq j$.

- $O_r = \bigcup_{\alpha=1}^{n_r} O_{(sg)_\alpha}$

  avec de la même façon $O_{(sg)_i} \cap O_{(sg)_j} = \emptyset$ pour $i \neq j$.

- $$Pre_r \begin{cases} Pre_{(sg)_1} : S_{(sg)_1} \times O_{(sg)_1} \rightarrow \{0,1\} \\ Pre_{(sg)_2} : S_{(sg)_2} \times O_{(sg)_2} \rightarrow \{0,1\} \\ \vdots \\ Pre_{(sg)_{n_r}} : S_{(sg)_{n_r}} \times O_{(sg)_{n_r}} \rightarrow \{0,1\} \\ Pre_{s_r} : \{s_r\} \times O_r \rightarrow \{0,1\} \end{cases}$$

  est l'application d'incidence avant telle que pour $i \neq m_{(sg)_\alpha} + 1$,

  $Pre_{s_r}(s_r, o_{(g)_\alpha, m_{(sg)_\alpha}+1}) = 1$ et $Pre_{s_r}(s_r, o_{(g)_\alpha, i}) = 0$.

  Ceci signifie que s_r est la place d'entrée de la première transition (opération) de chaque sous-gamme $(sg)_\alpha$.

  $(g)_\alpha$ est la gamme à partir de laquelle la sous-gamme $(sg)_\alpha$ a été définie.

- $$Post_r \begin{cases} Post_{(sg)_1} : S_{(sg)_1} \times O_{(sg)_1} \rightarrow \{0,1\} \\ Post_{(sg)_2} : S_{(sg)_2} \times O_{(sg)_2} \rightarrow \{0,1\} \\ \vdots \\ Post_{(sg)_{n_r}} : S_{(sg)_{n_r}} \times O_{(sg)_{n_r}} \rightarrow \{0,1\} \\ Post_{s_r} : \{s_r\} \times O_r \rightarrow \{0,1\} \end{cases}$$

  est l'application d'incidence arrière telle que pour $i \neq m_{(sg)_\alpha} + n_{(sg)_\alpha}$,

  $Post_{s_r}(s_r, o_{(g)_\alpha, m_{(sg)_\alpha}+n_{(sg)_\alpha}}) = 1$ et $Post_{s_r}(s_r, o_{(g)_\alpha, i}) = 0$.

  Ceci signifie que s_r est la place de sortie de la dernière transition (opération) de chaque sous-gamme $(sg)_\alpha$.

- $M_r : S_r \rightarrow N$ est l'application marquage initial telle que

  $M_r(s_r) = m_{s_r}$ et $M_r(s_{(g)_\alpha, i}) = 0$

  pour $i \in \{m_{(sg)_\alpha} + 2, ..., m_{(sg)_\alpha} + n_{(sg)_\alpha}\}$.

  m_{s_r} est le nombre de ressources de même type se trouvant dans la place s_r.

Nous noterons la première place ressource de la cellule s_{r1}, la deuxième s_{r2} etc.

Les jetons de s_{r2} de la figure 2.4 représentent des ressources flexibles qui peuvent être utilisées pour traiter les opérations $o_{g1,1}$, $o_{g1,2}$ et $o_{g1,3}$ de la sous-gamme $(sg)_1$,

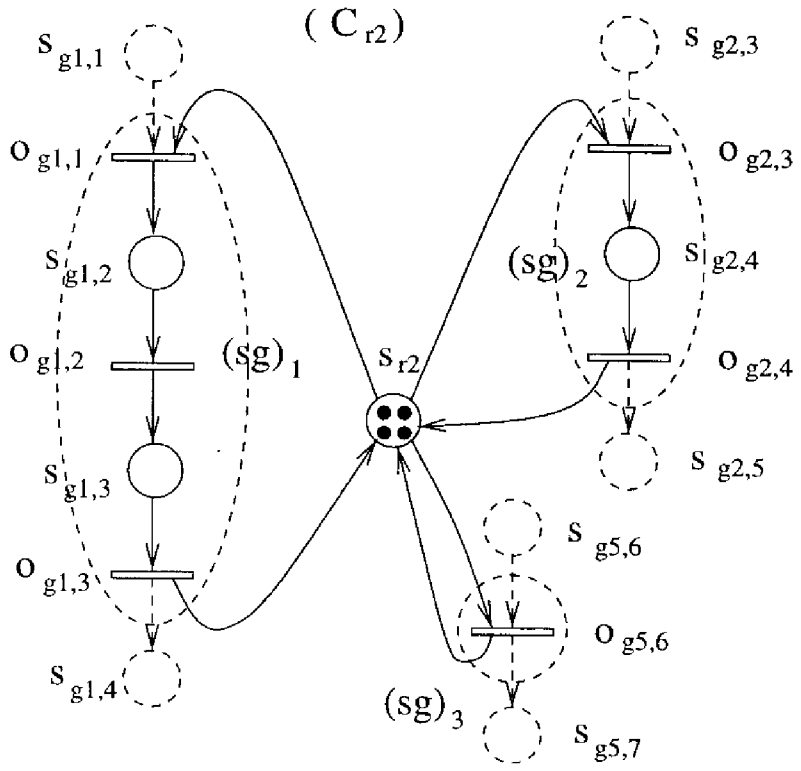

Figure 2.4: Contrainte de Ressource C_{r2}

ou les opérations $o_{g2,3}$ et $o_{g2,4}$ de $(sg)_2$, ou l'opération $o_{g5,6}$ de $(sg)_3$. La place s_{r2} qui est partagée entre trois sous-gammes permet de représenter un "ordonnancement flexible" dans la mesure où l'ordre dans lequel sont exécutées les opérations n'est spécifié que partiellement.

Nous remarquons que la définition syntaxique de la contrainte C_k est un cas particulier de la définition syntaxique de la contrainte C_r. En effet, les capacités des stocks intermédiaires des sous-gammes de fabrication pourraient être considérées comme des ressources nécessaires à la réalisation de la production. Néanmoins, nous pouvons associer une sémantique différente à chacune de ces deux contraintes du point de vue de la production. Plus il y aura d'étiquettes dans les places Kanban et plus il y aura de chance pour que l'on sature l'ensemble des ressources. Par contre, il risque d'y avoir de nombreux en-cours dans la cellule. S'il y a peu d'étiquettes Kanban le contrôle des en-cours sera plus simple mais le système perdra de sa flexibilité et de nombreuses ressources seront sous utilisées. Il est donc important de différencier ces deux types de contraintes et de définir un nombre d'étiquettes Kanban lors de la conception de la cellule qui permette de limiter les en-cours tout en laissant au système un degré de flexibilité acceptable.

2.1.5 Contrainte Cyclique

Il est nécessaire d'avoir une contrainte qui définisse les exigences de la production sous la forme d'un taux de production associé à chaque type de pièce. La contrainte qui va nous permettre de définir une politique de production cyclique a été présentée pour la première fois par *Ohl* [OH 94]. Nous la définissons de la façon suivante:

soit N_g gammes décrites par les réseaux gi pour $i \in \{1, ..., N_g\}$,
soit le réseau de Petri généralisé marqué [DA 92] $C_c =< S_c, O_c, Pre_c, Post_c, M_c >$,
C_c définit une politique de production cyclique si et seulement si:

- $S_c = \{c_{g1,1}, c_{g1,2}, ..., c_{gi,1}, c_{gi,2}, ..., c_{g_{N_g},1}, c_{g_{N_g},2}\}$ est un ensemble fini est non nul de $2N_g$ places.
- $O_c = \{o_{g1,1}, o_{g2,1}, ..., o_{g_{N_g},1}\} \cup \{t_s\}$ est un ensemble fini de $N_g + 1$ transitions. Les éléments de cet ensemble représentent les premières opérations de chacune des N_g gammes plus la transition de synchronisation t_s exprimant la contrainte cyclique.
- $Pre_c : S_c \times O_c \rightarrow N$ est l'application d'incidence avant telle que:
  $Pre_c(t_s, c_{gi,2}) = pd_{gi}$
  $Pre_c(t_s, c_{gi,1}) = 0$
  $Pre_c(o_{gi,1}, c_{gi,1}) = 1$
  $Pre_c(o_{gi,1}, c_{gj,1}) = 0$ si $j \neq i$
  $Pre_c(o_{gi,1}, c_{gj,2}) = 0$ pour $j \in \{1, ..., N_g\}$
- $Post_c : S_c \times O_c \rightarrow N$ est l'application d'incidence arrière telle que:
  $Post_c(t_s, c_{gi,1}) = pd_{gi}$
  $Post_c(t_s, c_{gi,2}) = 0$
  $Post_c(o_{gi,1}, c_{gi,2}) = 1$
  $Post_c(o_{gi,1}, c_{gj,2}) = 0$ si $j \neq i$
  $Post_c(o_{gi,1}, c_{gj,1}) = 0$ pour $j \in \{1, ..., N_g\}$
- $M_c : S_c \rightarrow N$ est l'application marquage initial telle que
  $M_c(c_{gi,1}) \geq pd_{gi}$ et $M_c(c_{gi,2}) = 0$.

Le réseau de la figure 2.5 représente un exemple de politique de production cyclique. Cette contrainte modélise le fait que la cellule doit traiter pd_{g1} pièces du buffer d'entrée $s_{g1,1}$, pd_{g2} pièces du buffer d'entrée $s_{g2,1}$, et pd_{g3} pièces du buffer d'entrée $s_{g3,1}$. Le signe "+" dans les places d'entrée des gammes $g1$, $g2$ et $g3$ signifie que les buffers d'entrée de la cellule sont alimentés en permanence. t_s est la transition de synchronisation qui impose la politique cyclique.

Il est possible de considérer au niveau de cet exemple que les stocks d'entrée $s_{g1,1}$ et $s_{g2,1}$ soient alimentés par des pièces de même type. Si tel est le cas, cela revient à considérer que ces deux stocks sont en fait au niveau de la cellule un même et

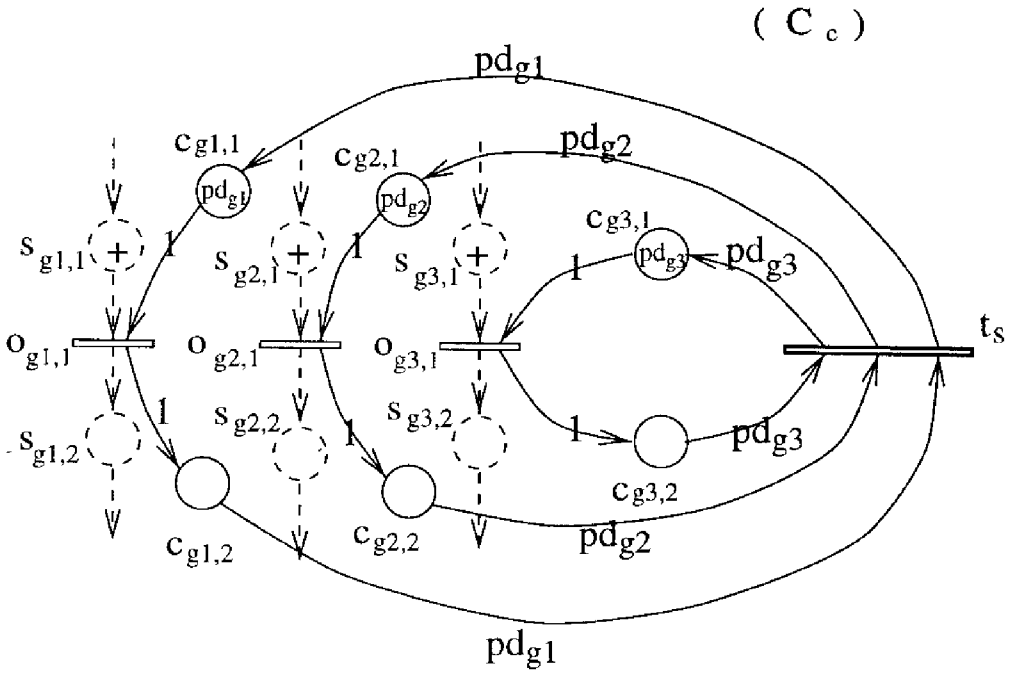

Figure 2.5: Contrainte Cyclique C_c

unique stock d'entrée qui alimente deux sous-gammes distinctes. Comme ce stock est alimenté en permanence, nous pouvons, afin de rester en accord avec la définition de la gamme de fabrication qui a été donnée, le diviser en deux stocks distincts $s_{g1,1}$ et $s_{g2,1}$ qui sont eux même alimentés en permanence. Ceci correspond à la situation où un même type de pièce peut être traité sur des gammes distinctes. On dit alors que l'on a de la flexibilité au niveau du routage des pièces. Les taux de production associés aux pièces des buffers d'entrée $(s_{g1,1}, s_{g2,1})$ d'une part, et aux pièces du buffer $s_{g3,1}$ d'autre part pour un cycle de production sont définis de la façon suivante:

$$\tau_{(s_{g1,1},s_{g2,1})} = \frac{pd_{g1} + pd_{g2}}{pd_{g1} + pd_{g2} + pd_{g3}}$$

$$\tau_{(s_{g3,1})} = \frac{pd_{g3}}{pd_{g1} + pd_{g2} + pd_{g3}}$$

2.1.6 Construction et Propriétés du Modèle Global

Construction du Modèle Global

Le modèle global de la cellule est obtenu par des fusions de branches (suites de places et de transitions) dans le cas des contraintes de type C_k et C_r, et par des fusions de transitions dans le cas de la contrainte cyclique C_c. Pour être plus exact, précisons que pour les contraintes C_k et C_r, les branches fusionnées seront les sous-gammes $sjgi$ des gammes gi communes à C_k et à C_r, et pour la contrainte C_c les transitions fusionnées seront les transitions opérations $o_{gi,1}$ des gammes gi pour $i \in \{1, ..., N_g\}$.

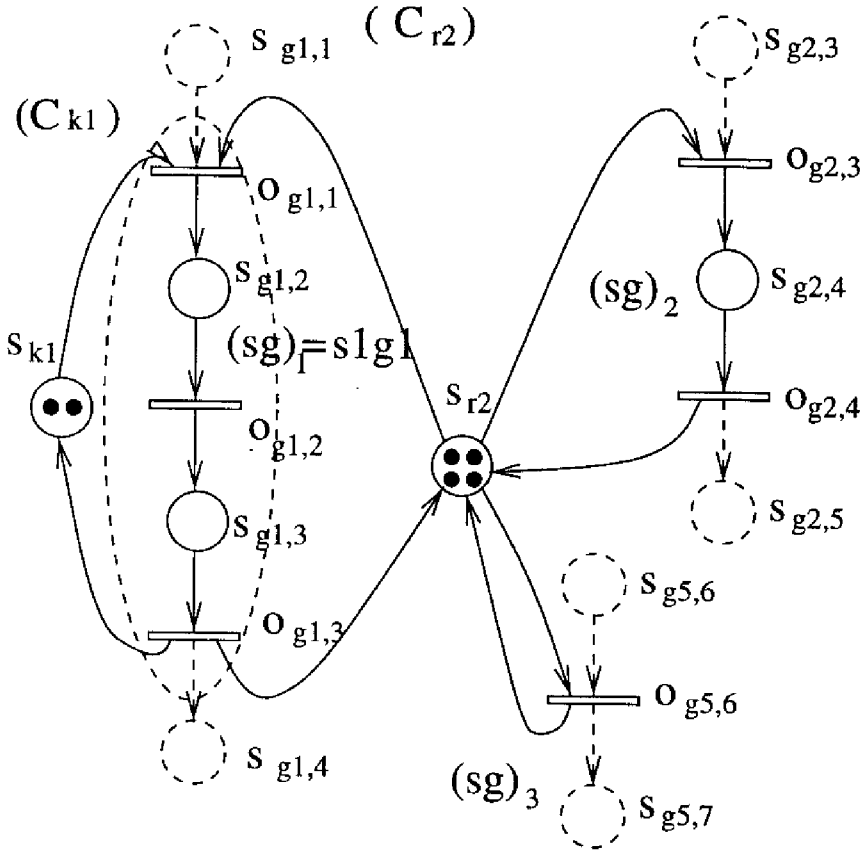

Figure 2.6: Fusion de C_{k1} avec C_{r2}

La figure 2.6 est un exemple de fusion d'une contrainte C_{k1} avec la contrainte C_{r2} suivant la sous-gamme $s1g1$.

La figure 2.7 est un autre exemple de fusion d'une contrainte C_{k2} avec la contrainte C_c au niveau de la transition opération $o_{g2,1}$.

Nous allons voir maintenant quelles sont les conséquences de telles fusions sur les propriétés du modèle global de la cellule.

Composantes Conservatives (invariants de places)

Une composante conservative, d'un point de vue graphique, définit un sous-réseau de Petri qui correspond généralement à la représentation d'une contrainte à respecter au sein d'un système de production. Toutes les contraintes qui ont été définies jusqu'à présent sont des *graphes d'états* ; chaque transition a exactement une place d'entrée et une place de sortie. Or, les graphes d'états sont des invariants de places [MU 89]. Par conséquent, chacune des contraintes est bien un invariant de places auquel peut être associé une composante conservative.

Il a été montré dans [VA 85, NA 86] qu'en fusionnant les transitions communes de sous-réseaux de Petri pour construire un réseau global, les invariants de places des sous-réseaux sont conservés, c'est-à-dire qu'ils sont des invariants de places du réseau

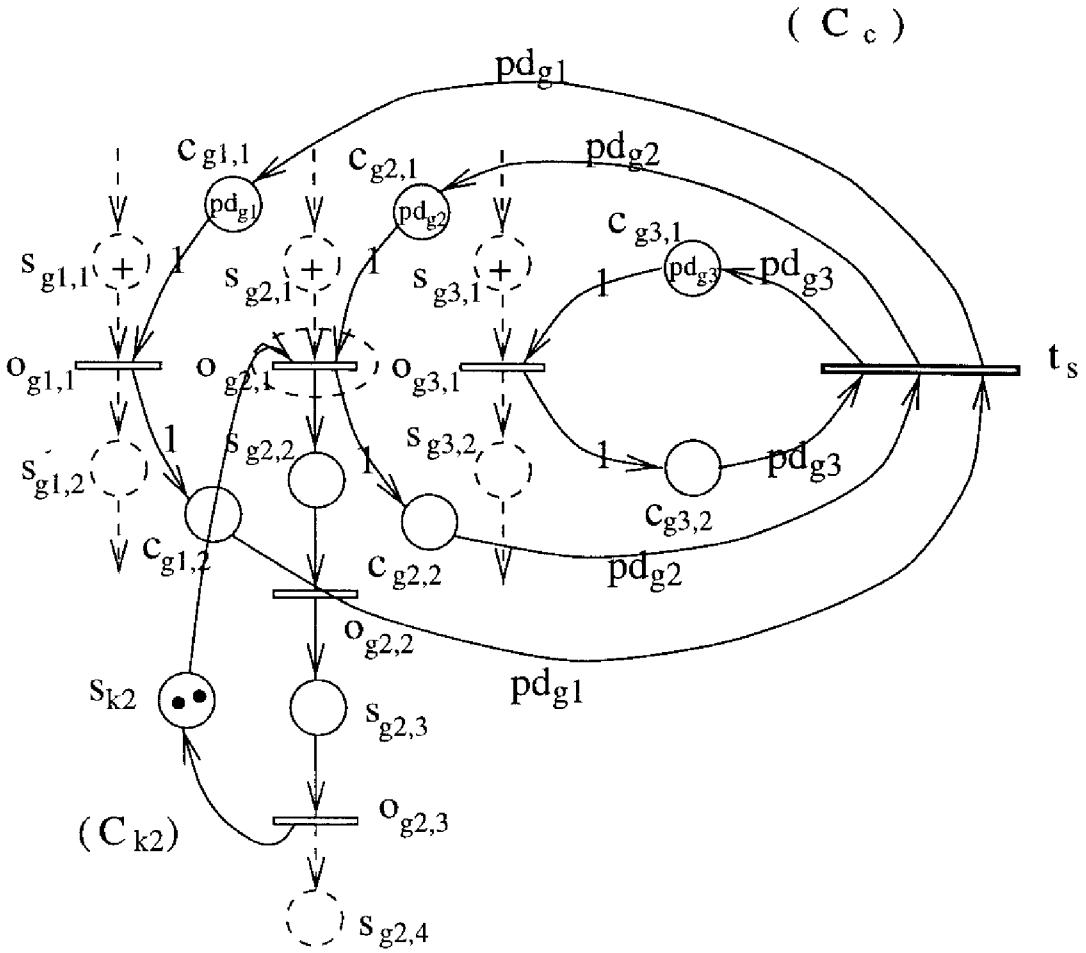

Figure 2.7: Fusion de C_c avec C_{k2}

global. Nous allons montrer par une preuve analogue à celle de [VA 85, NA 86] que ce résultat reste valide pour la fusion de branches (lorsque les branches commencent et se terminent par des transitions comme c'est le cas lors de la fusion d'une contrainte du type C_k avec une contrainte du type C_r).

Dans ce qui suit, comme nous nous replaçons dans un cadre général, nous utilisons des notations différentes. Tout d'abord rappelons la définition de la matrice d'incidence C d'un réseau de Petri R.

On appelle matrice d'incidence avant la matrice
$C^- = [c^-_{i,j}]$ où $c^-_{i,j} = Pre(pi, tj)$
avec pi une place de R et tj une transition de R.
On appelle matrice d'incidence arrière la matrice
$C^+ = [c^+_{i,j}]$ où $c^+_{i,j} = Post(pi, tj)$.
On appelle matrice d'incidence la matrice
$C = C^+ - C^- = [c_{i,j}]$.

Considérons deux réseaux de Petri R_1 de matrice d'incidence C_1 et R_2 de matrice d'incidence C_2. La fusion de ces deux réseaux produit le réseau R de matrice d'incidence C.

Pour le réseau R_1 nous avons:

- $t1$ l'ensemble des transitions internes,
- $p1$ l'ensemble des places internes,
- $t12$ l'ensemble des transitions fusionnant avec des transitions du réseau R_2,
- $p12$ l'ensemble des places fusionnant avec des places du réseau R_2.

Pour le réseau R_2 nous avons:

- $t2$ l'ensemble des transitions internes,
- $p2$ l'ensemble des places internes,
- $t12$ l'ensemble des transitions fusionnant avec des transitions du réseau R_1,
- $p12$ l'ensemble des places fusionnant avec des places du réseau R_1.

La seule restriction au processus de fusion est que les places de l'ensemble $p12$ ne sont connectées qu'aux transitions de l'ensemble $t12$, ce qui est bien le cas si l'on considère la définition des sous-gammes qui a été donnée. Si l'on construit la matrice $C = [c_{i,j}]$ d'incidence du réseau R, on obtient le schéma décrit par la figure 2.8.

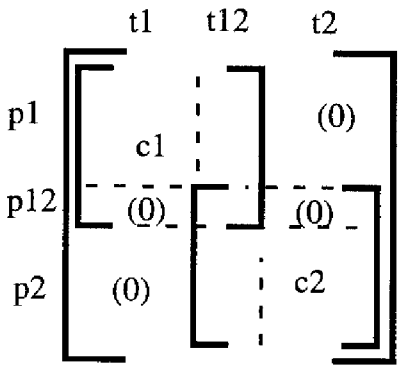

Figure 2.8: Matrice C

Les composantes conservatives sont solutions de

$$f^T.C = 0 \tag{2.1}$$

C'est-à-dire que l'on a autant d'inconnues que de places (les composantes de f) et autant d'équations que de transitions (les colonnes de la matrice C). L'équation correspondant à la colonne i s'écrit:

$$\sum c_{i,j}.f_i = 0 \tag{2.2}$$

Soit $f1$ une composante conservative du réseau R_1. Nous avons

$$f1^T.C1 = 0 \tag{2.3}$$

Considérons un vecteur f formé de la façon suivante:

- pour les composantes correspondant aux places des ensembles $p1$ et $p12$, on prend les valeurs correspondantes de $f1$,
- pour les composantes correspondant aux places de $p2$, on choisit la valeur 0.

Les équations correspondantes aux transitions des ensembles $t1$ et $t12$ redonnent les équations du système 2.3 puisqu'elles s'écrivent:

$$\sum c_{i,j}.f_i = 0 \tag{2.4}$$

avec les termes $f_i = 0$ pour toutes les composantes correspondantes à l'ensemble $p2$.

Les équations correspondant aux transitions de l'ensemble $t2$ sont toutes dégénérées sous la forme

$$0 = 0 \tag{2.5}$$

puisque les éléments $c_{i,j}$ sont tous nuls pour i appartenant aux ensembles $p1$ et $p12$ et que les f_i sont tous nuls pour les i appartenant à l'ensemble $p2$.

Cela veut dire que les composantes conservatives (invariants de places) du réseau R_1 sont nécessairement des composantes conservatives du réseau R.

Il est clair qu'un raisonnement analogue peut être fait pour les composantes conservatives de R_2.

Tout cela nous amène à la conclusion que chacune des contraintes que nous avons défini (contrainte Kanban, contrainte de ressource, contrainte cyclique) apparaîtra dans le réseau global sous la forme d'un invariant de places.

Composantes Répétitives Stationnaires (invariants de transitions)

Au niveau d'un réseau de Petri, une séquence cyclique d'événements se traduit par un invariant de transitions. Comme nous nous limitons dans ce travail à l'étude des systèmes de production cyclique, le réseau global qui va modéliser la cellule devra décrire un invariant de transitions qui spécifiera exactement combien de fois chaque transition sera franchie durant un cycle de production.

La construction du modèle se fait en considérant tout d'abord la contrainte cyclique C_c qui impose les taux de production et plus particulièrement qui définit le régime cyclique.

Soit I_c la matrice d'incidence de C_c.
Les composantes répétitives stationnaires sont solution de

$$I_c.r = 0 \tag{2.6}$$

avec r le vecteur caractéristique d'une séquence de franchissement de transitions. Le système d'équation correspondant peut s'écrire sous la forme:

$$\begin{cases} pd_{g1}.r_s = r_{g1,1} \\ pd_{g2}.r_s = r_{g2,1} \\ \vdots \\ pd_{gN_g}.r_s = r_{gN_g,1} \end{cases} \tag{2.7}$$

ce qui veut dire que si $r_s = 1$ c'est-à-dire si la transition de synchronisation t_s de O_c est franchie une fois dans la composante répétitive stationnaire r (pour un cycle), alors la transition $o_{g1,1}$ de O_c sera franchie pd_{g1} fois au cours de ce cycle, la transition $o_{g2,1}$ sera franchie pd_{g2} fois etc.

De ce résultat, nous déduisons que la contrainte C_c est un invariant de transitions et nous pourrons dire de façon exacte en fonction du marquage du réseau combien de fois chacune des transitions de C_c sera franchie pour qu'un cycle se complète.

A partir de cette contrainte, on fusionne ensuite de façon incrémentale une à une les contraintes de type C_k. La seule restriction au processus de fusion est que la nouvelle contrainte fusionnée ait au moins une de ses transitions qui appartienne au modèle incrémental de la cellule.

Soit I_k la matrice d'incidence de la contrainte C_k.

Le résultat de la fusion de I_k et de I_c au niveau du calcul des composantes répétitives stationnaires donne un ensemble d'équations supplémentaires:

$$\begin{cases} r_{gi,m_{sjgi}+1} = r_{gi,m_{sjgi}+2} \\ r_{gi,m_{sjgi}+2} = r_{gi,m_{sjgi}+3} \\ \vdots \\ r_{gi,m_{sjgi}+n_{sjgi}} = r_{gi,m_{sjgi}+1} \end{cases} \tag{2.8}$$

ce qui veut dire que si une des transitions de S_k est franchie une fois pour un cycle, alors les autres transitions de S_k seront franchies chacune une fois pour que le cycle se complète. Comme au moins une des transitions de S_k appartient au modèle incrémental de la cellule avant fusion de la nouvelle contrainte, nous savons qu'au niveau du modèle obtenu, après fusion, chacune des transitions de S_k sera franchie durant un cycle complet pd_{gi} fois si t_s est franchie une fois seulement.

Lorsque toutes les contraintes de type C_k sont fusionnées au modèle incrémental, il ne reste plus qu'à inclure les contraintes de type C_r. Ces contraintes sont les contraintes introduites par les ressources. Elles correspondent à des sous-réseaux qui définissent une composante répétitive stationnaire par sous-gamme. Il suffit donc de prendre la valeur pd_{gi} pour toutes les transitions des sous-gammes issues de gi pour trouver une composante compatible avec la solution du réseau global.

Après avoir fusionné toutes les contraintes en suivant le raisonnement précédent, nous arrivons au résultat suivant:

- le modèle global de la cellule est un invariant de transitions,
- si pour un cycle la transition t_s est franchie une fois, alors les transitions des gammes gi du modèle global seront franchies pd_{gi} fois chacune.

Les contraintes qui définissent le modèle de la cellule ont donc les fréquences d'occurrence de leurs transitions qui sont imposées par la contrainte cyclique C_c.

Bonnes Propriétés

Il existe au niveau du modèle global de la cellule une couverture de composantes conservatives et répétitives stationnaires. De ce résultat, nous pouvons déduire d'une part que le réseau qui modélise la cellule est k-borné (mis à part les places initiales et finales des gammes) quel que soit son marquage initial. D'autre part, le réseau est recouvert par un invariant de transitions exprimant la possibilité d'un régime cyclique. Ce régime cyclique, induit par le sous-réseau décrivant la politique de production cyclique, respecte les proportions désirées entre les pièces des diverses gammes (la transition t_s synchronisant les diverses exécutions).

Enfin, comme nous nous restreindrons au cas d'un réseau global vivant, il est important suivant le cas particulier considéré de vérifier qu'il n'existe pas un entrelacement mauvais d'utilisation de ressources. Il existe en effet la possibilité d'introduire durant la construction du modèle un siphon pouvant se vider et provoquer un blocage mortel. Il faut donc faire l'analyse du modèle de la cellule pour savoir s'il est vivant, soit par réduction quand la structure est simple, soit en montrant qu'il n'y a aucun siphon susceptible de se vider [EZ 93]. Dans ce dernier cas, il faut spécifier lors de l'utilisation simultanée de ressources que les ressources soient demandées simultanément ou qu'elles soient toujours demandées dans le même ordre.

Il est important de souligner que la démarche proposée ne garantit pas l'existence effective d'une séquence de franchissements de transitions correspondant au cycle. En effet, si l'ensemble des sous-réseaux définit un ensemble de contraintes contradictoires, le réseau global ne sera pas vivant et il est possible qu'aucune séquence franchissable ne corresponde à l'invariant de transitions définissant le cycle. Néanmoins, de par le mécanisme de construction retenu, le réseau de Petri obtenu a une structure très particulière, semblable aux réseaux de la classe S^3PR (Systèmes de Processus Séquentiels Simples avec Ressources) [EZ 93, AB 96]. Il est donc clair qu'en imposant certaines restrictions sur l'ordre d'utilisation des ressources dans les gammes, on pourrait obtenir un réseau vivant par construction. La démarche serait toutefois très restrictive. Il est également clair [AB 96] qu'une recherche a posteriori des verrous sur le réseau global permettrait d'établir si ce réseau est vivant ou non et éventuellement de le modifier pour qu'il le devienne. Toutefois, le but de notre travail est de caractériser des politiques de conduite par un ensemble de conditions nécessaires. Il est plus important pour nous d'introduire le plus tôt possible le temps de façon explicite pour trouver des conditions nécessaires riches vis-à-vis du processus de décision pour la conception, que de prouver qu'une séquence correspondant à la composante répétitive stationnaire est effectivement franchissable. Cette preuve serait en effet beaucoup plus coûteuse que l'analyse quantitative que nous allons présenter par la suite et très probablement

elle donnerait moins d'informations à l'opérateur humain. En effet, dans le cas des cellules flexibles, vu l'existence de stocks intermédaires entre les opérations, les cas de blocage dûs aux ressources sont rares et en général ce sont les contraintes temporelles explicites qui déterminent les politiques de pilotage.

Nous allons maintenant introduire explicitement le temps dans notre modèle.

2.2 Modèle temporel/temporisé

2.2.1 Modèle t-temporisé

Une cellule de fabrication est un système qui dépend du temps. En effet, il s'écoule une durée non nulle entre le début et la fin d'une opération. Par conséquent, le modèle utilisé pour modéliser la cellule devra être capable de prendre en compte les durées d'opération. Il existe divers modèles qui permettent la représentation du temps: les réseaux de Petri temporisés, les réseaux de Petri temporels, les réseaux de Petri stochastiques etc... Les réseaux de Petri temporisés qui sont essentiellement utilisés dans les problèmes d'évaluation de performances et d'ordonnancement permettent de modéliser les durées soit en les associant aux places [TA 87] (réseaux p-temporisés), soit en les associant aux transitions [HI 89, RA 80] (réseaux t-temporisés). Dans les contraintes définies précédemment, les opérations ont été associées aux transitions des gammes de fabrication. Nous allons donc utiliser pour modéliser les durées d'opération (durées de transport, durées d'usinage etc.) les réseaux t-temporisés.

Définition 2.2.1 *Un réseau de Petri t-temporisé est un doublet* $< R, tempo >$ *tel que:*

- *R est un réseau de Petri marqué,*
- *tempo est une application de l'ensemble des transitions dans l'ensemble des nombres réels positifs ou nuls; si t est une transition alors* $tempo(t) = d_t$ *est la temporisation associée à t.*

Le principe de fonctionnement d'un réseau t-temporisé comme celui des réseaux de Petri ordinaires, n'impose pas le franchissement des transitions au plus tôt. Quand une transition est sensibilisée on peut choisir de la franchir immédiatement ou choisir d'attendre un certain temps. Considérons le réseau décrit par la figure 2.9. Après le franchissement de la transition t_1, un jeton est déposé dans la place p_2. A partir de cet instant, on peut décider de franchir t_2 à tout instant mais non nécessairement dès l'instant où t_2 est franchissable. Si après une durée arbitraire, durée pendant laquelle le jeton reste dans la place p_2 dans l'état non réservé, on décide de franchir t_2, alors le jeton est mis dans un état réservé. Après une durée d_{t_2} (temporisation associée à t_2), la transition est effectivement franchie, le jeton est retiré de p_2, et un jeton se trouvant dans l'état non réservé est déposé dans p_3.

Maintenant que nous avons rappelé ce qu'est un réseau t-temporisé, revenons au contexte de notre travail pour discuter des valeurs des durées que nous allons associer aux transitions. En effet, une durée d'opération $d_{o_{gi,j}}$ associée à une transition

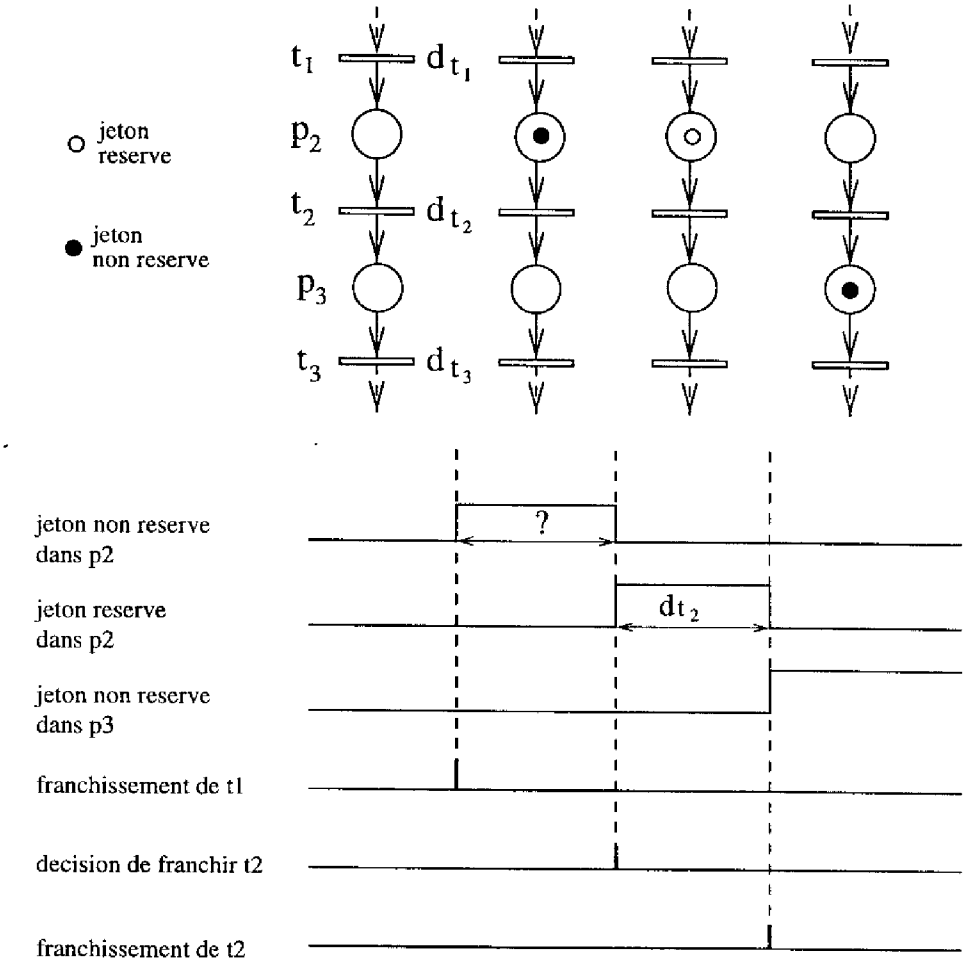

Figure 2.9: Réseau de Petri t-temporisé

$o_{gi,j}$ d'une gamme gi appartient à un intervalle $[(d_{o_{gi,j}})_{min}, (d_{o_{gi,j}})_{max}]$. $(d_{o_{gi,j}})_{min}$ représente la durée minimale de réalisation de l'opération $o_{gi,j}$ et $(d_{o_{gi,j}})_{max}$ représente la durée maximale. Par exemple, si l'opération $o_{gi,j}$ nécessite l'utilisation d'un bras manipulateur, il est évident que selon la position du bras dans la cellule la durée $d_{o_{gi,j}}$ sera plus ou moins longue. Si nous nous intéressons à l'évaluation des performances d'une cellule dans le contexte de l'analyse sous contrainte, l'important sera de mettre en évidence les comportements limites au-delà desquels les contraintes ne seront plus vérifiées. Les valeurs $(d_{o_{gi,j}})_{max}$ seront alors les valeurs significatives. En effet, un comportement obtenu dans le cas où est considéré la durée maximale $(d_{o_{gi,j}})_{max}$ reste valide même si la durée réelle $d_{o_{gi,j}}$ est inférieure à $(d_{o_{gi,j}})_{max}$. Il suffira d'augmenter la durée pendant laquelle le jeton reste non réservé dans la place d'entrée de la transition correspondante. Nous pourrons nous intéresser aussi à des politiques avec de nombreux *"en-cours"* pour favoriser la flexibilité et la valeur moyenne sera alors la plus significative.

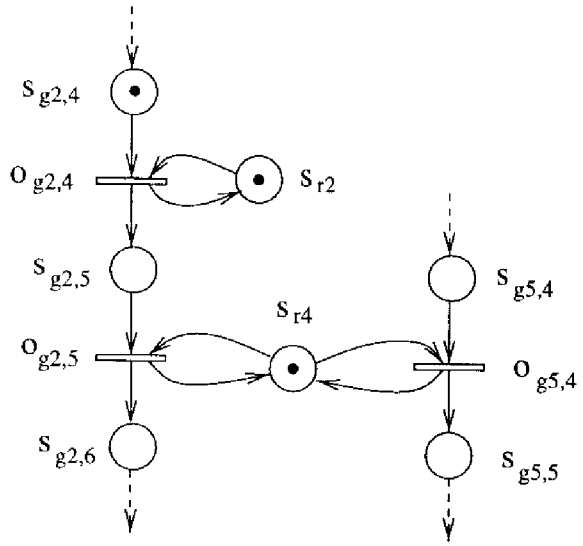

Figure 2.10: Modèle t-temporisé

Considérons par exemple la figure 2.10. Le jeton de la place s_{r2} modélise une machine qui est utilisée pour réaliser l'opération $o_{g2,4}$, et le jeton de la place s_{r4} modélise un bras manipulateur qui est utilisé pour transporter les pièces du stock de sortie $s_{g2,5}$ au stock d'entrée $s_{g2,6}$. Une durée d'usinage $d_{o_{g2,4}}$ peut être associée à la transition $o_{g2,4}$ et une durée de transport $d_{o_{g2,5}}$ peut être associée à la transition $o_{g2,5}$.

Les durées d'opération associées aux transitions représentent une partie de l'ensemble des contraintes à respecter. Elles correspondent à des paramètres de notre problème de conception de cellule et sont définies par des valeurs fixes. Selon le type de régime cyclique considéré (régime libre ou régime forcé), nous choisirons les valeurs maximales ou les valeurs moyennes des durées d'opération. Ce point sera présenté plus en détail dans le chapitre suivant lorsque sera défini le régime cyclique d'une cellule flexible.

2.2.2 Modèle p-t-temporisé

Pour le travail que nous développons, le modèle précédent présente un inconvénient majeur: si les paramètres de notre problème sont bien explicités par l'intermédaire des durées associées aux transitions, par contre les variables de décision qui sont les durées d'attente des jetons non réservés dans les places, ne sont pas explicitées. Il est donc nécessaire d'associer des variables correspondant aux attentes dans les places avant le franchissement des transitions. Cette association existe dans les réseaux de Petri p-temporisés, et comme nous voulons conserver les durées associées aux transitions, nous obtenons les réseaux p-t-temporisés.

Définition 2.2.2 *Un réseau de Petri p-t-temporisé est un triplet* $< R, tempo\_p, tempo\_t >$ *tel que:*

- *R est un réseau de Petri marqué,*

- *tempo_p est une application de l'ensemble des places dans l'ensemble des nombres réels positifs ou nuls. Si p est une place alors tempo_p(p) = d_p est la temporisation associée à p.*
- *tempo_t est une application de l'ensemble des transitions dans l'ensemble des nombres réels positifs ou nuls. Si t est une transition alors tempo_t(t) = d_t est la temporisation associée à t.*

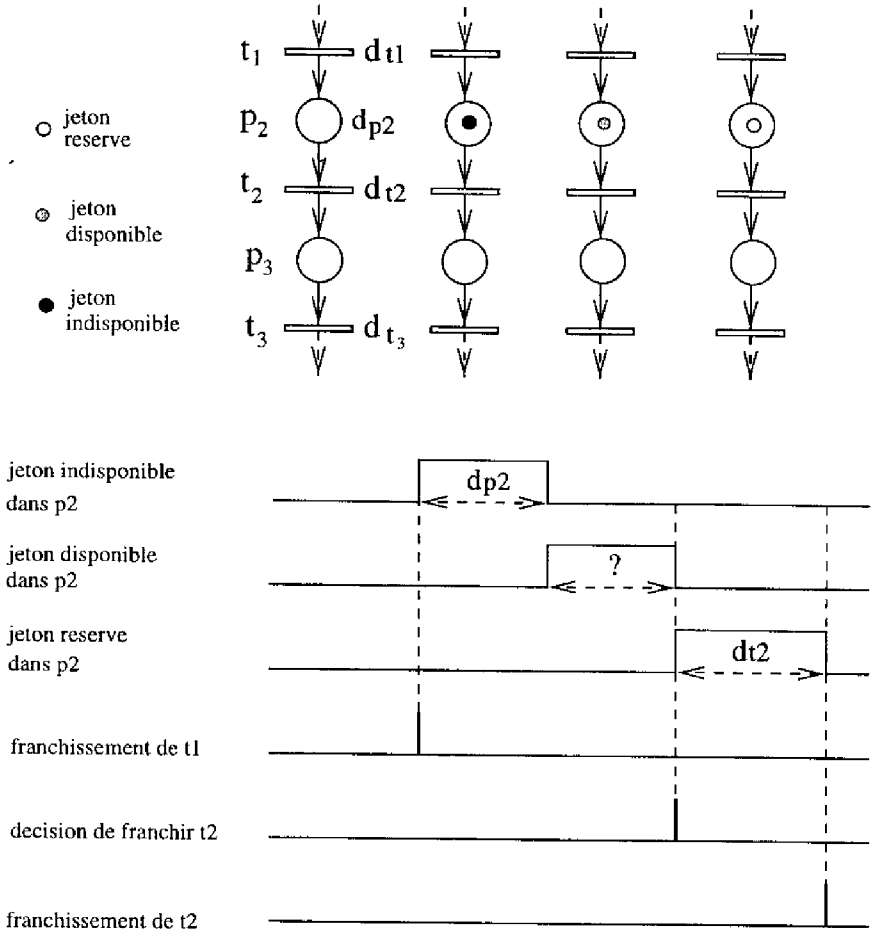

Figure 2.11: Réseau de Petri p-t-temporisé

Considérons le réseau décrit par la figure 2.11. Après le franchissement de la transition t_1, un jeton est déposé dans la place p_2 et mis dans l'état indisponible. Après la durée d_{p_2} (temporisation associée à p_2), le jeton passe dans l'état disponible. A partir de cet instant, on peut décider de franchir t_2 à tout instant mais non nécessairement dès l'instant où t_2 est franchissable. Si après une durée arbitraire, durée pendant laquelle le jeton reste dans la place p_2 dans l'état disponible, on décide de franchir t_2, alors le jeton est mis dans l'état réservé. Après une durée d_{t_2} (temporisation associée à t_2), la transition est effectivement franchie, le jeton est retiré de p_2, et un jeton se trouvant dans l'état indisponible est déposé dans p_3.

Si on choisit de franchir les transitions au plus tôt (ce qui n'est pas imposé par le modèle), le temps de séjour du jeton dans la place p_2 avant d'être réservé est exactement égal à d_{p_2}. Le choix d'une valeur pour cette variable imposera alors un certain séquencement des franchissements de transition et permettra par exemple de réserver une ressource pour une tâche plus urgente. Par contre, on peut remarquer que l'on n'aura aucune flexibilité. Une fois la valeur d_{p_2} choisie, il n'y a plus aucun indéterminisme et la séquence de franchissement sera totalement fixée.

Examinons maintenant le cas où l'on ne se restreint pas à franchir les transitions au plus tôt. Le temps de séjour du jeton dans la place p_2 sans être réservé est maintenant égal à d_{p_2} augmenté du temps de séjour dans l'état disponible et non réservé. Avec cette approche, il est possible de retarder l'exécution d'une opération moins urgente en associant une durée d'attente à la place d'entrée de la transition correspondante et de conserver une certaine flexibilité puisque le jeton peut séjourner dans la place pendant un temps excédent cette durée. Mais il s'agit d'une flexibilité excessive et non contrôlée puisque le franchissement de la transition peut être retardé indéfiniment. Pour pouvoir exactement exprimer ce que nous voulons, il nous faut donc la notion d'attente minimale et d'attente maximale existante dans les réseaux de Petri temporels, mais que cette attente soit associée aux places.

2.2.3 Modèle p-temporel t-temporisé

Le modèle de réseau de Petri p-temporel présenté par *Khansa* dans [KH 96] est particulièrement bien adapté pour associer des intervalles de temps aux places. Dans notre cas ces intervalles ne seront pas des durées d'opération mais des durées d'attente qui seront exploitées pour l'ordonnancement des opérations de la cellule, mais cela ne changera pas la définition formelle du modèle. Puisque les durées d'opération sont attachées aux transitions du modèle, notre modèle ne sera pas simplement p-temporel mais plus exactement p-temporel t-temporisé.

Définition 2.2.3 *Un réseau de Petri p-temporel t-temporisé est un triplet* $< R, I_p, tempo\_t >$ *tel que:*

- *R est un réseau de Petri marqué,*
- I_p *est une application définie de la façon suivante:*
  $I_p : P \rightarrow (Q^+ \cup 0) \times (Q^+ \cup \infty)$
  $p_i \rightarrow I_{p_i} = [a_i, b_i]$ *avec* $0 \leq a_i \leq b_i$
- *tempo_t est une application de l'ensemble des transitions dans l'ensemble des nombres réels positifs ou nuls*

Comme dans les modèles précédents, si t est une transition alors $tempo\_t(t) = d_t$ est la temporisation associée à t.

$I_{p_i} = [a_i, b_i]$ définit l'intervalle statique d'attente durant laquelle un jeton reste dans la place p_i: a_i est la durée minimale avant laquelle le jeton de p_i reste dans l'état non disponible. b_i est la durée maximale avant laquelle le jeton doit être réservé.

L'évolution dynamique d'un réseau p-temporel t-temporisé dépend du marquage des jetons et de leurs situations temporelles. A un instant particulier, l'état du réseau est complètement donné par le triplet $E =< M, D_p, D_t >$ tel que:

- $M = M_n + M_r$ avec M_n marquage du réseau par les jetons non réservés (indisponibles + disponibles) et M_r marquage du réseau par les jetons réservés.
- D_p est l'application intervalle de temps dynamique qui associe pour chaque jeton non réservé j d'une place p_i un intervalle $[a_i^j, b_i^j]$ délimitant la date à laquelle le jeton peut passer dans l'état réservé. Si à la date δ le jeton j arrive dans la place p_i à laquelle est associé l'intervalle statique $[a_i, b_i]$ alors à la date $\delta + \theta$ $(a_i \leq \theta \leq b_i)$ l'intervalle dynamique associé à j est $[a_i^j, b_i^j] = [max(a_i - \theta, 0), b_i - \theta]$.
- D_t est l'application qui associe pour chaque jeton j réservé pour le tir d'une transition t_i la date δ_i^j à laquelle t_i sera effectivement franchie.

Pour bien illustrer cette notion, considérons à nouveau le réseau décrit par la figure 2.11. Au lieu d'avoir la temporisation d_{p_2} associée à la place p_2 nous avons maintenant l'intervalle statique $[(d_{p_2})_{min}; (d_{p_2})_{max}]$ associé à cette place. Après le franchissement de la transition t_1, le jeton est déposé dans la place p_2 à la date δ_0 et mis dans l'état indisponible. A la date $\delta_1 = \delta_0 + (d_{p_2})_{min}$, le jeton passe dans l'état disponible. A partir de cet instant on peut décider de franchir t_2 puisque le jeton est passé dans l'état disponible. A la date δ_2 qui est telle que $\delta_1 \leq \delta_2 \leq \delta_0 + (d_{p_2})_{max}$, on décide de franchir t_2 et le jeton est mis dans l'état réservé. A la date $\delta_2 + d_{t_2}$, la transition est effectivement franchie, le jeton est retiré de p_2, et un jeton se trouvant dans l'état indisponible est déposé dans p_3.

Les bornes a_i et b_i de l'intervalle statique associé à chaque place du réseau permettent de retrouver des équivalents aux dates de début au plus tôt et dates de début au plus tard qui sont classiques en ordonnancement non cyclique. Dans notre modèle, ceci s'illustre par des contraintes sur le franchissement des transitions. Soient par exemple p_i et p_j les deux places d'entrée d'une transition t. Soit δ_i la date d'apparition d'un jeton dans la place p_i. La contrainte I_{p_i} impose le franchissement de t dans l'intervalle $[\delta_i + a_i, \delta_i + b_i]$. Soit δ_j la date d'apparition d'un jeton dans la place p_j. La contrainte I_{p_j} impose le franchissement de t dans l'intervalle $[\delta_j + a_j, \delta_j + b_j]$. t devra donc être franchie dans l'intervalle que l'on obtient en faisant l'intersection de $[\delta_i + a_i, \delta_i + b_i]$ avec $[\delta_j + a_j, \delta_j + b_j]$ puisque les deux jetons qui se trouvent dans les places d'entrée de t doivent être dans l'état disponible pour sensibiliser la transition et passer dans l'état réservé. Si cette intersection est nulle, cela veut dire que les contraintes sont contradictoires. Dans la pratique cela voudra dire qu'une contrainte (au moins) devra être violée. Il faudra relâcher les contraintes du problème et augmenter la flexibilité. Nous reviendrons plus en détail sur la notion de conflit pour un réseau p-temporel t-temporisé lorsque nous présenterons l'algorithme du joueur.

Maintenant que nous avons défini le modèle p-temporel t-temporisé, replaçons ce nouveau modèle dans le contexte de la modélisation de cellules flexibles.

Si les temporisations associées aux transitions représentent toujours les durées d'opération, les durées d'attente pendant lesquelles les jetons peuvent rester dans

les places dans l'état non réservé peuvent avoir plusieurs significations suivant la sémantique des places auxquelles elles sont associées.

Lorsque l'on considère une place stock intermédiaire $s_{gi,j}$ d'une gamme gi, la durée d'attente associée au jeton de cette place, pour un ordonnancement donné, représente le temps qu'une pièce, se trouvant dans ce stock, doit attendre pour qu'une ressource nécessaire à la réalisation de l'opération suivante $o_{gi,j}$ soit disponible et lui soit affectée. Par exemple, si à l'instant où une pièce est déposée dans le stock $s_{g2,5}$ de la figure 2.10, la ressource s_{r4} est déjà réservée pour l'opération $o_{g5,4}$, alors on devra attacher à $s_{g2,5}$ une durée d'attente non nulle qui représentera le temps d'attente de la pièce dans $s_{g2,5}$ avant que la ressource s_{r4} soit disponible. Rappelons que les ressources peuvent être partagées par plusieurs opérations et que par conséquent elles ne sont pas forcément disponibles à tout instant.

La durée d'attente attachée à un jeton d'une place ressource s_{ri} représente la durée pendant laquelle la ressource reste au repos entre deux opérations. Par exemple, si après avoir terminé l'opération $o_{g2,5}$ il n'y a plus de jetons dans les places $s_{g2,5}$ et $s_{g5,4}$, alors la ressource s_{r4} reste au repos en attendant qu'une nouvelle pièce soit déposée dans $s_{g2.5}$ ou dans $s_{g5,4}$. Le fonctionnement le plus efficace de la cellule est celui pour lequel les durées attachées aux jetons ressource sont nulles c'est-à-dire lorsque les ressources sont saturées. Il faut néanmoins remarquer que si les ressources sont saturées, on ne dispose plus d'aucune flexibilité.

Finalement, d'une façon beaucoup plus générale, les durées d'attente associées aux jetons de certaines places peuvent être considérées comme un moyen d'introduire de la flexibilité au niveau de la politique de contrôle du système. Ces durées peuvent par exemple prendre en compte des contraintes supplémentaires qui ne sont pas directement formalisées dans notre approche. Par exemple, si nous considérons la figure 2.10, le démarrage de l'opération $o_{g2,4}$ peut nécessiter une intervention humaine que l'on pourra traduire par une durée d'attente supplémentaire attachée au jeton de $s_{g2,4}$ qui retardera la date de début de l'opération $o_{g2,4}$.

Les durées d'attente associées aux jetons des places représenteront les variables à définir de notre approche. Rappelons que notre objectif est d'établir un bon compromis entre efficacité et flexibilité. Il sera donc extrêmement important d'analyser les domaines dans lesquels ces durées prendront leurs valeurs afin de définir une politique de contrôle à la fois robuste et efficace.

Nous allons maintenant présenter un exemple simple qui va illustrer plus en détail les avantages et limitations de chacun des modèles temporels que nous venons de définir .

2.2.4 Exemple

Le réseau de la figure 2.12 représente un exemple très simple de cellule flexible. Cette cellule met en jeu deux gammes de fabrication $g1$ et $g2$ et par conséquent deux contraintes Kanban C_{k1} et C_{k2}. Pour simplifier le problème nous n'avons représenté que la ressource partagée modélisée par le jeton de s_{r1}. Les autres ressources associées aux transitions opération $o_{g1,2}$, $o_{g1,4}$, $o_{g2,2}$, $o_{g2,4}$ ne sont pas partagées et n'ayant pas d'influence sur le comportement flexible de la cellule n'apparaissent pas sur le modèle.

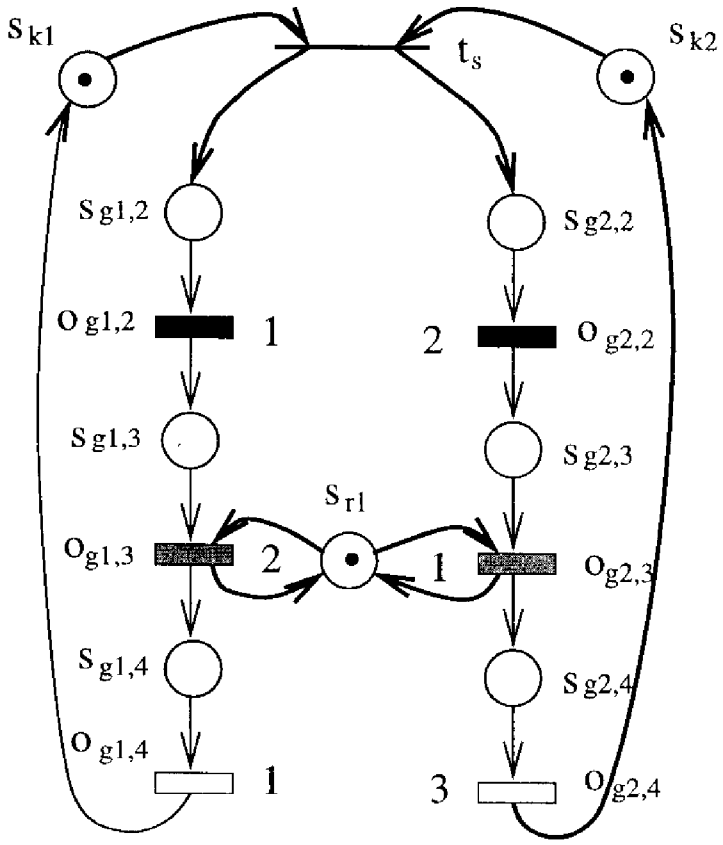

Figure 2.12: Cellule Flexible

Il n'y a donc qu'une contrainte ressource C_{r1} à considérer. La politique de production cyclique étant extrêmement simple dans ce cas, nous avons directement fusionné la transition de synchronisation t_s de la contrainte cyclique C_c avec les transitions opérations $o_{g1,1}$ et $o_{g2,1}$ de début de gamme que nous supposons de durée nulle. Les temporisations associées aux transitions sont celles qui sont directement exprimées sur la figure.

Si nous choisissons un simple modèle t-temporisé avec franchissement des transitions au plus tôt, nous obtenons le diagramme temporel 2.13 qui montre que le temps de cycle est égal à 7.

Nous choisissons maintenant un modèle p-t-temporisé. On s'autorise à maintenir les jetons de certaines places dans l'état indisponible. Nous choisissons encore la stratégie de franchissement au plus tôt. Les variables $tempo\_p(p_i)$ associées aux places p_i serviront à définir un plan de fabrication sans aucun indéterminisme (c'est-à-dire définissant une seule séquence de franchissement sans ambiguité). Par exemple, si $tempo\_p(s_{g1,3}) = 1,5$ et si toutes les autres temporisations associées aux places sont nulles, alors on obtiendra le diagramme temporel 2.14. On remarque que maintenant le temps de cycle est de 6. La décision d'imposer une attente minimale de 1,5 dans la place $s_{g1,3}$ est donc une bonne décision. On remarque également que les jetons séjournerons dans $s_{g1,3}$ pendant 1,5 en étant non disponibles puis pendant 0,5

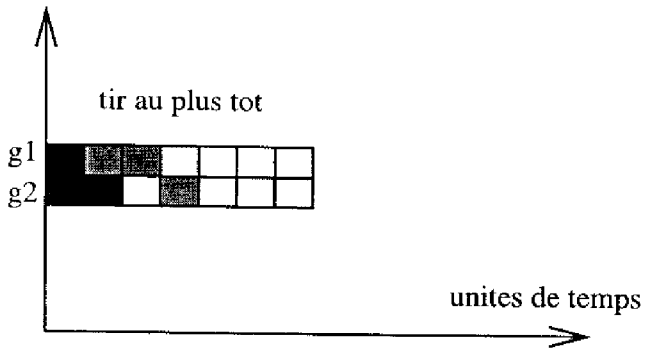

Figure 2.13: Tir au plus Tôt

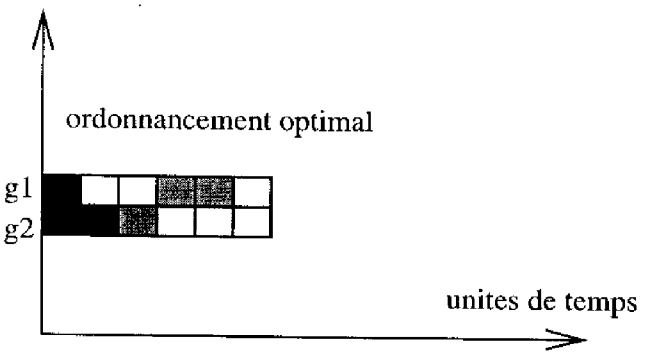

Figure 2.14: Ordonnancement Optimal 1

en étant disponibles et non réservés puisque la transition $o_{g1,3}$ n'est sensibilisée que lorsque l'opération $o_{g1,3}$ se termine et que le jeton ressource de s_{r1} devient à nouveau disponible.

On retrouve le même diagramme temporel (et le même temps de cycle optimal) en choisissant pour $tempo\_p(s_{g1,3})$ n'importe quelle valeur x telle que $1 < x \leq 2$.

Si on choisit $tempo\_p(s_{g1,3}) = 0$ et $tempo\_p(s_{g1,2}) = y$ avec $1 < y \leq 2$, on retrouve encore le temps de cycle optimal mais un diagramme différent cette fois (figure 2.15). Si $y = 1,5$ par exemple, le jeton restera dans la place $s_{g1,2}$ dans l'état indisponible pendant $1,5$ et dans la place $s_{g1,3}$ dans l'état disponible et non réservé pendant $0,5$.

Cet exemple illustre bien le fait que le choix des valeurs des variables $tempo\_p$ pour chaque place permet de définir un ordonnancement qui peut être meilleur que le franchissement au plus tôt dans un réseau t-temporisé, mais il illustre également le fait que cela peut amener à se restreindre arbitrairement à une seule séquence et à un fonctionnement sans flexibilité alors que plusieurs séquences seraient tout aussi bonnes.

Nous pouvons bien sûr ne pas nous restreindre au franchissement au plus tôt pour donner plus de flexibilité au système. Néanmoins, le relâchement de la contrainte du franchissement au plus tôt sera trop brutal puisque rien n'empêchera alors un jeton disponible de rester dans un place pour une durée illimitée. Il est donc indispensable d'introduire la notion de durée d'attente maximale qui existe dans les réseaux de Petri

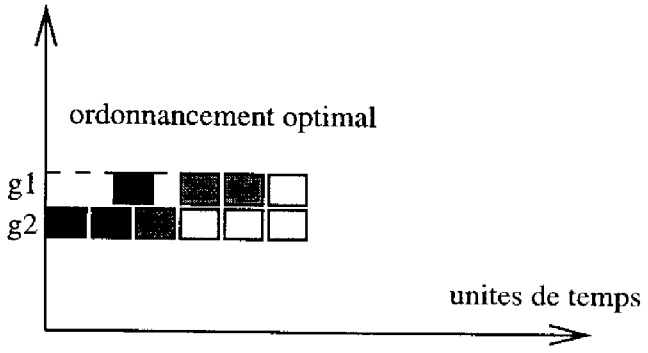

Figure 2.15: Ordonnancement Optimal 2

p-temporels t-temporisés.

De façon générale, au niveau d'un ordonnancement cyclique, afin d'obtenir la performance optimale correspondant à la durée de cycle minimale, il ne sera pas forcément intéressant de franchir les transitions du modèle temporisé au plus tôt (dès que la transition est franchissable vis à vis du marquage). Cette remarque illustre le fait que le réseau temporisé, plutôt que de définir une politique de contrôle, définit simplement un ensemble de contraintes qui doivent être respectées par la politique de contrôle. Afin d'avoir un contrôle sur la flexibilité introduite sans avoir pour autant à pratiquer une politique de tir au plus tôt, des contraintes temporelles supplémentaires doivent être associées au simple modèle t-temporisé. Ces contraintes seront définies sous forme d'intervalles associés aux places du réseau et délimiteront en particulier la flexibilité introduite, d'où l'utilisation dans la suite de ce travail d'un modèle p-temporel t-temporisé. En plus du modèle p-temporel t-temporisé, il sera indispensable d'avoir une politique de contrôle robuste et efficace à appliquer au modèle temporel et permettant de définir des séquences optimales ou sous-optimales respectant les contraintes et pouvant être calculées aussi bien au niveau d'ordonnancements prévisionnels que d'ordonnancements temps réel utilisés pour le pilotage de la cellule.

Dans l'exemple ci dessus, on peut caractériser un ensemble de bonnes solutions en choisissant $tempo\_p(s_{g1,2}) \in [0;1[$ et $tempo\_p(s_{g1,3}) \in [0;1[$.

Dans ce qui suit, après avoir défini le fonctionnement cyclique particulier qui sera appliqué au modèle, nous présenterons comment analyser quantitativement le modèle que nous avons présenté dans ce chapitre et nous formulerons comment à partir des résultats obtenus nous pourrons définir une politique de contrôle à la fois robuste, efficace et respectant les contraintes imposées par le modèle.

Chapitre 3

Approche Générale pour l'Analyse Quantitative d'une Cellule Flexible

3.1 Régime Stationnaire Périodique Forcé et Analyse sous Contrainte

Conformément à ce que nous avons dit en conclusion du premier chapitre, nous allons baser notre approche dans le contexte de l'analyse sous contrainte [ER 86, ES 87, LO 91] appliquée au modèle p-temporel t-temporisé que nous avons présenté dans le deuxième chapitre. Le principe sera d'encadrer par deux ensembles de contraintes un ensemble de fonctionnements acceptables. Les contraintes à respecter seront obtenues à partir d'un régime périodique particulier : **le régime stationnaire périodique forcé.** Nous le définissons de la façon suivante :

Pour un régime stationnaire périodique forcé, si t_0 *est l'instant initial,* E_0 *l'état initial du réseau p-temporel t-temporisé et* π *la période de production, alors l'état du réseau auquel est appliqué ce régime sera* E_0 *aux instants:* $t_0, t_0 + \pi, t_0 + 2\pi, ..., t_0 + n\pi, etc....$

Nous rappelons que l'état du réseau c'est le marquage courant, plus pour chaque jeton non réservé son intervalle de visibilité, plus pour chaque transition en cours de franchissement la durée restante.

La particularité de ce régime est que nous devons retrouver les conditions initiales (E_0) dès la première période de fonctionnement. Un tel fonctionnement existe dans la mesure où grâce à l'existence de durées d'attente associées aux places, nous ne sommes pas obligés de ne considérer que le fonctionnement à vitesse maximale puisque nous ne franchissons pas nécessairement au plus tôt les transitions du réseau. Suivant la valeur de la période de production π, nous parlerons soit d'un régime périodique forcé optimiste, soit d'un régime périodique forcé pessimiste.

Le régime périodique forcé optimiste sera tel que la valeur de la période π soit minimale pour un ensemble de t-temporisations données. Il n'est pas sûr que l'atelier

puisse effectivement fonctionner de la sorte à cause de possibles perturbations qui pourront par exemple retarder les dates de début de certaines opérations. Néanmoins, ce régime nous permettra en particulier de calculer une durée de cycle minimale: pour la politique d'affectation des ressources et de lancement décrite par le réseau de Petri, il sera impossible de faire mieux. Nous calculerons une borne minimale pour ce régime en ne prenant en compte qu'une partie des contraintes pour éviter l'explosion combinatoire. Cela nous permettra de connaître les opérations critiques et la marge existante sur les autres opérations puisque nous aurons des valeurs pour les durées d'attente associées aux places.

Pour être sûr de pouvoir exécuter une séquence d'opérations dans tous les cas, nous imposerons une durée d'attente (marge de sécurité) minimale pour certaines opérations (en particulier les opérations critiques) en associant des durées d'attente non nulles aux places d'entrée de ces opérations. L'introduction de ces contraintes supplémentaires se traduira au niveau du fonctionnement cyclique par une valeur de la période π plus importante qui exprimera une augmentation de la flexibilité et une diminution de l'efficacité. Nous parlerons alors d'un régime périodique forcé pessimiste.

A partir de ces deux régimes, une politique d'utilisation de la flexibilité sera définie afin de résoudre en temps réel les conflits d'allocation de ressources non résolus par le réseau de Petri. Notre démarche cherche donc à aider à la fois à la conception de la cellule (combien de machines) grâce aux indices de performance qui seront obtenus au travers de la valeur de la période de production et à la conception du pilotage (ou ordonnancement réactif en temps réel) sous la forme de règles pour l'exploitation de la flexibilité.

Nous allons maintenant présenter plus en détail ces régimes stationnaires périodiques forcés au travers de la définition formelle de la période de production π.

3.2 Mise en Equation

3.2.1 Spécification de la Politique de Production Cyclique

Comme nous l'avons dit dans le chapitre précédent lors du calcul de l'invariant de transitions, la construction du modèle se fait en considérant tout d'abord la contrainte cyclique C_c qui définit le régime cyclique. Nous nous limiterons au calcul de l'invariant de transitions qui passe par la transition t_s de la contrainte C_c avec un poids égal à 1 ce qui signifie que pour un cycle, la transition t_s sera franchie une fois et les autres transitions $o_{gi,1}$ $(i \in \{1, ..., N_g\})$ pd_{gi} fois chacune.

Soit n_p le nombre de types de produits différents p_j $(j \in \{1, ..., n_p\})$ qui peuvent être traités par une même cellule. N_g est le nombre de gammes g_i $(i \in \{1, ..., N_g\})$ de la cellule. Il existe n_{p_j} gammes $(g)_\beta$ $(\beta \in \{1, ..., n_{p_j}\})$ au sein de la cellule à partir desquelles peut être réalisé un même produit p_j. Si pour un cycle de production N_{p_j} représente le nombre de produits traités de même type p_j et N_p le nombre de produits traités tous types confondus, alors le taux de production associé aux pièces p_j est défini de la façon suivante:

$$\tau_{p_j} = \frac{N_{p_j}}{N_p}$$

ce qui est équivalent à:

$$N_{p_j} = N_p.\tau_{p_j} \tag{3.1}$$

Le taux τ_{p_j} est une donnée du problème qui est fournie généralement dès la phase de conception de la cellule. Il représente la proportion de produits p_j qui doivent être réalisés à long terme vis à vis de la production globale de la cellule. Considérant la contrainte définie par l'équation 3.1, N_p doit être choisi de telle façon que N_{p_j} soit à valeur entière $\forall j \in \{1, ..., n_p\}$.

En se basant sur l'invariant de transitions qui recouvre le modèle de la cellule, le nombre de produits p_j à traiter pour un cycle de production en fonction des poids pd_{gi} de la contrainte cyclique C_c est :

$$N_{p_j} = \sum_{\beta=1}^{n_{p_j}} pd_{(g)_\beta}$$

Après avoir défini des quantités N_p et N_{p_j} qui soient en accord avec les taux τ_{p_j} associés aux produits p_j, nous pouvons alors introduire les contraintes

$$\sum_{\beta=1}^{n_{p_j}} pd_{(g)_\beta} = N_{p_j} \tag{3.2}$$

pour $j \in \{1, ..., n_p\}$ qui définissent les domaines dans lesquels les poids pd_{gi} de la contraintes C_c prennent leurs valeurs afin de respecter les taux imposés par les exigences de la production.

Puisque nous avons pour une cellule N_g gammes de fabrication et n_p types de produits différents qui peuvent être réalisés, la spécification de la production pour un cycle de production se fera au travers des n_p équations linéaires 3.2 et des N_g variables entières pd_{gi}.

Si nous considérons par exemple la politique de production cyclique représentée sur la figure 2.5, pd_{g1}, pd_{g2} et pd_{g3} sont les variables entières qui vont définir combien de produits devront être réalisés sur les gammes de fabrication $g1$, $g2$ et $g3$ pour un cycle. Supposons que la cellule à laquelle est appliquée cette politique cyclique soit capable de réaliser deux types de produits p_1 et p_2. Le type p_1 peut être réalisé aussi bien sur la gamme $g1$ que sur la gamme $g2$, et le type p_2 seulement sur la gamme $g3$. Les quantités à produire sont telles que $\tau_{p_1} = 65\%$ et $\tau_{p_2} = 35\%$. Le plus petit nombre de produits qui peuvent être réalisés pour un cycle de fabrication (période de production) afin de respecter ces taux est $N_p = 20$ avec $N_{p_1} = 13$ et $N_{p_2} = 7$. Une autre spécification de la production respectant toujours ces taux pourrait être par exemple $N_p = 40$ avec $N_{p_1} = 26$ et $N_{p_2} = 14$. Si nous considérons le cas $N_p = 20$ alors nous devons introduire les contraintes suivantes:

$$pd_{g1} + pd_{g2} = 13 \tag{3.3}$$

$$pd_{g3} = 7$$

La flexibilité de gamme (choix entre plusieurs gammes possibles) de la cellule sera ainsi exprimée par l'équation 3.3 qui représente l'ensemble au sein duquel les variables entières pd_{g1} et pd_{g2} prendront leurs valeurs.

3.2.2 Période de Production pour une Contrainte Kanban

Rappelons que l'invariant de transitions qui recouvre le modèle global de la cellule est tel que si pour un cycle la transition t_s de la contrainte C_c est franchie une fois seulement alors les transitions d'une gamme g sont franchies pd_g fois chacune.

Une contrainte Kanban C_k est un invariant de places composé de n_{sg} places et de n_{sg} transitions qui sont franchies pd_g fois chacune pour un cycle puisqu'elles appartiennent toutes à la même gamme g.

Si cet invariant de places ne contient qu'un jeton i.e. $M_k(s_k) = 1$, que nous considérions un régime stationnaire périodique forcé, et que la spécification de la production par la contrainte cyclique C_c soit telle qu'un seul produit soit traité sur la sous-gamme sg durant une période de production cyclique i.e. $pd_g = 1$, alors le jeton de C_k passe d'une place à l'autre en franchissant successivement chaque transition une fois seulement et la période se termine lorsque le jeton se retrouve dans son état initial i.e. $M_k(s_k) = 1$. Ceci se traduit par l'expression de la période π:

$$\pi = \sum_{i=1}^{n_{sg}} d_{o_{g,m_{sg}+i}} + \sum_{i=2}^{n_{sg}} d_{s_{g,m_{sg}+i}} + d_{s_k}$$

avec $d_{o_{g,m_{sg}+i}}$ la t-temporisation associée à la transition $o_{g,m_{sg}+i}$ de O_k et $d_{s_{g,m_{sg}+i}}$ le temps d'occupation de la place $s_{g,m_{sg}+i}$ de S_k par le jeton (dans l'état non réservé) de C_k pendant une période π (idem pour d_{s_k}).

Si la politique de production cyclique décrite par C_c est telle que pd_g ($\neq 1$) produits soient traités suivant la gamme g comprenant la sous-gamme sg pendant une période, alors nous devons franchir chaque transition pd_g fois et l'expression de la période est:

$$\pi = pd_g \sum_{i=1}^{n_{sg}} d_{o_{g,m_{sg}+i}} + \sum_{i=2}^{n_{sg}} d_{s_{g,m_{sg}+i}} + d_{s_k}$$

$d_{s_{g,m_{sg}+i}}$ est le temps d'occupation global de la place $s_{g,m_{sg}+i}$ par le jeton dans l'état non réservé durant une période complète π (idem pour d_{s_k}).

Dans ce qui suit, la variable

$$d_{C_k} = \sum_{i=2}^{n_{sg}} d_{s_{g,m_{sg}+i}} + d_{s_k}$$

représentera le temps global d'occupation des places de la contrainte C_k par un jeton (dans l'état non réservé) appartenant à cet invariant de places durant une période complète π.

Si nous considérons le cas le plus général où nous avons plus d'un jeton par invariant de places C_k i.e. $M_k(s_k) = m_{s_k} \neq 1$, alors il y a dans le meilleur des cas autant d'évolutions simultanées possibles à l'intérieur de la sous gamme sg qu'il y a de jetons dans la mesure bien sûr où $m_{s_k} \leq pd_g$ (le nombre d'étiquettes Kanban doit être inférieur ou égal au nombre de pièces qui parcourent la gamme g pendant une période de production) et où pd_g est un multiple de m_{s_k}. Pour qu'une période se complète, il faut que l'on retrouve le marquage initial $M_k(s_k) = m_{s_k}$ après le franchissement des n_{sg} transitions par les m_{s_k} jetons de C_k et la période associée à la contrainte Kanban doit donc vérifier:

$$\pi = \frac{pd_g}{m_{s_k}} \sum_{i=1}^{n_{sg}} d_{o_{g,m_{sg}+i}} + d_{C_k} \tag{3.4}$$

avec le rapport $\frac{pd_g}{m_{s_k}}$ à valeur entière. d_{C_k} représente la durée d'attente globale admissible d'une étiquette Kanban de la contrainte C_k pendant une période de production ou de façon plus formelle la durée globale pendant laquelle un jeton de l'invariant de places C_k reste dans l'état non réservé pendant l'exécution de la séquence de tir de l'invariant de transitions qui passe par ce sous-réseau.

Considérons l'exemple de la figure 2.3 en supposant que la politique de production imposée par la contrainte C_c soit telle que quatre pièces du buffer d'entrée $s_{g,1}$ soient traitées sur cette gamme pendant une période. Nous avons deux jetons dans la place s_k. Par conséquent, nous pouvons déduire de cette contrainte l'équation suivante:

$$\pi = \frac{4}{2}(d_{o_{g,1}} + d_{o_{g,2}} + d_{o_{g,3}}) + d_{C_k}$$

π et d_{C_k} constituant bien sûr les inconnues réelles positives (ou nulles pour d_{C_k}) de cette équation linéaire.

Si pd_g n'est pas un multiple de m_{s_k} alors le rapport $\frac{pd_g}{m_{s_k}}$ de l'équation 3.4 n'est pas à valeur entière et l'expression de la période donnée par cette équation n'est plus cohérente avec la notion de régime stationnaire périodique forcé. En effet, lorsqu'une étiquette Kanban a commencé un cycle, on doit franchir chaque transition de la contrainte C_k un nombre fini et entier de fois et retrouver le marquage initial.

On suppose que chaque jeton franchit la séquence le même nombre de fois. Reconsidérons l'exemple de la figure 2.3 en supposant cette fois que la politique de production soit telle que cinq pièces du buffer d'entrée $s_{g,1}$ soient traitées pendant une période. Comme nous avons deux jetons dans la place s_k l'expression de la période devient:

$$\pi = \frac{5}{2}(d_{o_{g,1}} + d_{o_{g,2}} + d_{o_{g,3}}) + d_{C_k}$$

Le rapport $\frac{5}{2}$ n'est pas à valeur entière et l'expression de π n'est plus cohérente avec la notion de régime stationnaire périodique forcé. Le rapport $\frac{5}{2}$ peut alors être vu que comme le nombre moyen de franchissements des transitions du réseau lorsque l'on considère un *régime stationnaire périodique libre* [DA 92].

Une solution afin de pouvoir utiliser la formule 3.4 pour le régime stationnaire périodique forcé lorsque pd_g n'est pas un multiple de m_{s_k} est de déplier la gamme g

qui comprend la sous-gamme sg de la contrainte C_k en deux gammes $g^{(1)}$ et $g^{(2)}$. Nous devrons alors considérer au lieu d'une contrainte C_k deux contraintes $C_{k^{(1)}}$ et $C_{k^{(2)}}$. La contrainte $C_{k^{(1)}}$ aura comme marquage initial $M_{k^{(1)}}(s_{k^{(1)}}) = M_k(s_k) - 1$ et $C_{k^{(2)}}$ aura $M_{k^{(2)}}(s_{k^{(2)}}) = 1$. Nous devrons aussi considérer au niveau de la contrainte cyclique C_c deux poids $pd_{g^{(1)}}$ et $pd_{g^{(2)}}$ qui remplaceront le poids pd_g tels que $pd_{g^{(1)}} + pd_{g^{(2)}} = pd_g$.

Par exemple, si nous déplions la gamme g de la contrainte C_k (figure 2.3) en deux gammes $g^{(1)}$ et $g^{(2)}$, nous obtenons les deux contraintes $C_{k^{(1)}}$ et $C_{k^{(2)}}$ de la figure 3.1 avec $M_{k^{(1)}}(s_{k^{(1)}}) = 1$ et $M_{k^{(2)}}(s_{k^{(2)}}) = 1$.

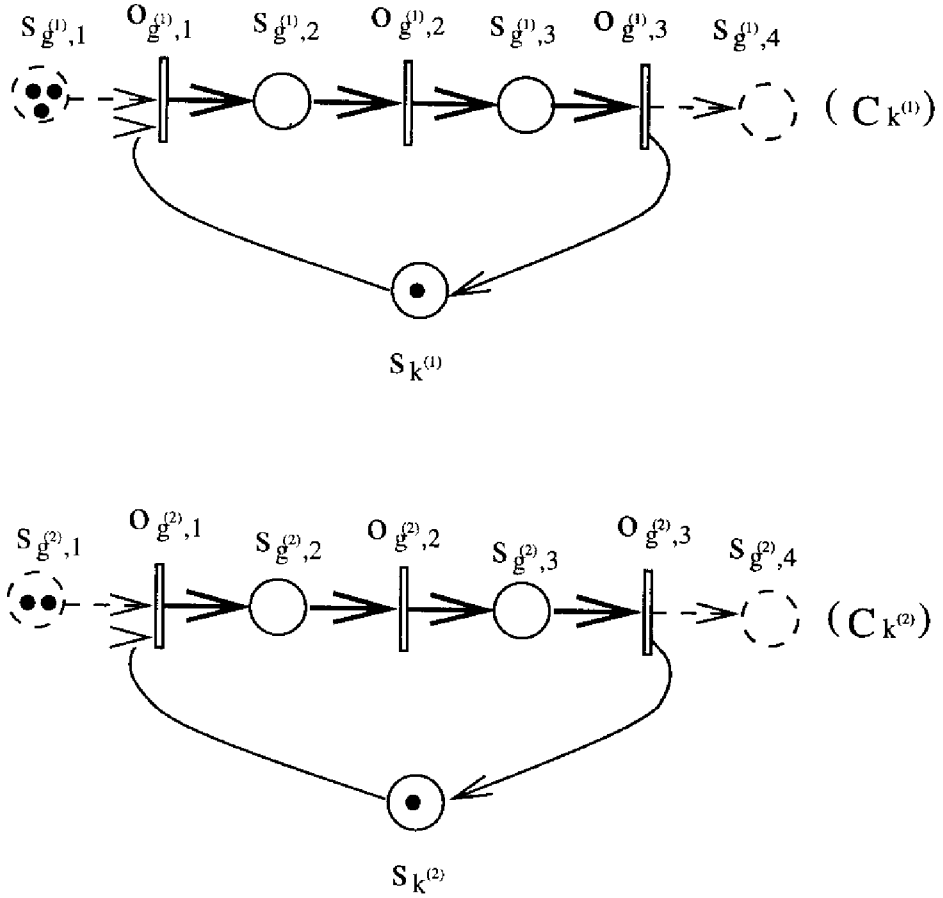

Figure 3.1: Dépliage de C_k en $C_{k^{(1)}}$ et $C_{k^{(2)}}$

Si la spécification de la production par C_c est telle que $pd_{g^{(1)}} = 3$ et $pd_{g^{(2)}} = 2$ de façon à bien respecter la contrainte $pd_{g^{(1)}} + pd_{g^{(2)}} = pd_g = 5$, alors nous pouvons déduire de la contrainte $C_{k^{(1)}}$ l'équation:

$$\pi = 3(d_{o_{g^{(1)},1}} + d_{o_{g^{(1)},2}} + d_{o_{g^{(1)},3}}) + d_{C_{k^{(1)}}} \tag{3.5}$$

et de la contrainte $C_{k^{(2)}}$ l'équation:

$$\pi = 2(d_{o_{g^{(2)},1}} + d_{o_{g^{(2)},2}} + d_{o_{g^{(2)},3}}) + d_{C_{k^{(2)}}} \tag{3.6}$$

Les équations déduites du réseau déplié C_k ont donc bien leurs rapports $\frac{pd_{g^{(1)}}}{m_{s_{k^{(1)}}}}$ et $\frac{pd_{g^{(2)}}}{m_{s_{k^{(2)}}}}$ à valeur entière. En particulier, l'équation 3.5 signifie que trois produits sont traités par l'étiquette Kanban de $C_{k^{(1)}}$ et l'équation 3.6 que deux produits sont traités par l'étiquette de $C_{k^{(2)}}$.

Dans le cas le plus général, les poids de la contrainte cyclique C_c ne sont pas connus et l'équation déduite du réseau de la figure 2.3 s'écrit:

$$\pi = \frac{(d_{o_{g,1}} + d_{o_{g,2}} + d_{o_{g,3}})}{2}.pd_g + d_{C_k}$$

Nous aurons alors à considérer les deux inconnues réelles π et d_{C_k} et l'inconnue entière pd_g.

3.2.3 Période de Production pour une Contrainte de Ressource

Rappelons qu'une contrainte de ressource C_r est un invariant de places composé d'une place ressource s_r et de n_r sous-gammes $(sg)_\alpha$. Chaque sous-gamme $(sg)_\alpha$ est composée de $(n_{(sg)_\alpha}-1)$ places $s_{(g)_\alpha,i}$ pour $i \in \{m_{(sg)_\alpha}+2, ..., m_{(sg)_\alpha}+n_{(sg)_\alpha}\}$ et de $n_{(sg)_\alpha}$ transitions $o_{(g)_\alpha,i}$ pour $i \in \{m_{(sg)_\alpha}+1, ..., m_{(sg)_\alpha}+n_{(sg)_\alpha}\}$. L'invariant de transitions qui recouvre le modèle de la cellule nous indique que pour un cycle, chaque transition $o_{(g)_\alpha,i}$ appartenant à une gamme $(g)_\alpha$ sera franchie $pd_{(g)_\alpha}$ fois.

Si l'invariant de places C_r ne contient qu'un jeton i.e. $M_r(s_r) = 1$, que nous considérions un régime stationnaire périodique forcé, et que la spécification de la production par la contrainte C_c soit telle que $pd_{(g)_\alpha} = 1$ i.e. qu'un seul produit soit traité sur chaque sous-gamme $(sg)_\alpha$ issue de la gamme $(g)_\alpha$, alors pour qu'une période se complète, le jeton de C_r doit franchir chaque transition $o_{(g)_\alpha,i}$ une fois seulement pour chacune des n_r sous-gammes $(sg)_\alpha$ appartenant à la contrainte C_r. Ceci va se traduire par l'expression de la période:

$$\pi = \sum_{\alpha=1}^{n_r} \sum_{i=m_{(sg)_\alpha+1}}^{m_{(sg)_\alpha}+n_{(sg)_\alpha}} d_{o_{(g)_\alpha,i}} + \sum_{\alpha=1}^{n_r} \sum_{i=m_{(sg)_\alpha+2}}^{m_{(sg)_\alpha}+n_{(sg)_\alpha}} d_{s_{(g)_\alpha,i}} + d_{s_r}$$

avec $d_{o_{(g)_\alpha,i}}$ la t-temporisation associée à la transition $o_{(g)_\alpha,i}$ de $O_{(sg)_\alpha}$ et $d_{s_{(g)_\alpha,i}}$ le temps d'occupation de la place $s_{(g)_\alpha,i}$ de $S_{(sg)_\alpha}$ par le jeton dans l'état non réservé de C_r pendant une période π (idem pour d_{s_r}).

Si la politique de production cyclique décrite par C_c est telle que $pd_{(g)_\alpha}$ $(\neq 1)$ produits soient traités suivant la gamme $(g)_\alpha$ comprenant la sous-gamme $(sg)_\alpha$ pendant une période, alors nous devons franchir chaque transition de $(sg)_\alpha$ $pd_{(g)_\alpha}$ fois et l'expression de la période devient:

$$\pi = \sum_{\alpha=1}^{n_r} pd_{(g)_\alpha} \sum_{i=m_{(sg)_\alpha+1}}^{m_{(sg)_\alpha}+n_{(sg)_\alpha}} d_{o_{(g)_\alpha,i}} + \sum_{\alpha=1}^{n_r} \sum_{i=m_{(sg)_\alpha+2}}^{m_{(sg)_\alpha}+n_{(sg)_\alpha}} d_{s_{(g)_\alpha,i}} + d_{s_r}$$

$d_{s_{(g)_\alpha,i}}$ est le temps d'occupation global de la place $s_{(g)_\alpha,i}$ par le jeton dans l'état non réservé durant une période complète π (idem pour d_{s_r}).

Dans ce qui suit, la variable

$$d_{C_r} = \sum_{\alpha=1}^{n_r} \sum_{i=m_{(sg)_\alpha+2}}^{m_{(sg)_\alpha}+n_{(sg)_\alpha}} d_{s_{(g)_\alpha,i}} + d_{s_r}$$

représentera le temps global d'occupation des places de la contrainte C_r par un jeton dans l'état non réservé appartenant à cet invariant de places durant une période complète π.

Si nous considérons le cas le plus général où nous avons plus d'un jeton par invariant de places C_r i.e. $M_r(s_r) = m_{s_r} \neq 1$, alors il est nécessaire de considérer le parallélisme qui peut exister entre ces jetons puisque suivant le cas plusieurs ressources appartenant au pool de ressources peuvent être utilisées simultanément. Dans le meilleur des cas, nous arriverons à répartir exactement la même quantité de travail sur chaque ressource i.e. toutes les ressources du pool seront utilisées (se trouveront dans l'état réservé) pendant exactement la même durée et la période associée à la contrainte de ressource devra donc vérifier:

$$\pi = \frac{1}{m_{s_r}} [\sum_{\alpha=1}^{n_r} pd_{(g)_\alpha} \sum_{i=m_{(sg)_\alpha+1}}^{m_{(sg)_\alpha}+n_{(sg)_\alpha}} d_{o_{(g)_\alpha,i}}] + d_{C_r} \tag{3.7}$$

d_{C_r} représente la durée d'attente globale admissible d'une ressource du pool pendant une période de production ou de façon plus formelle la durée globale pendant laquelle un jeton de l'invariant de places C_r peut rester dans l'état non réservé avant que la séquence de tir de l'invariant de transitions qui passe par ce sous-réseau se termine.

L'équation 3.7 exprime le fait que la quantité de travail qui peut être fait par le pool de ressources durant une période π est limitée: c'est une sorte de contrainte d'énergie [LO 91]. Comme nous considérons la quantité globale du travail qui peut être effectué dans le meilleur des cas par le pool de ressources, nous n'avons qu'une condition nécessaire qu'il ne sera pas toujours possible de désagréger. L'avantage sera de pouvoir considérer tout un ensemble d'alternatives possibles sans avoir pour autant à les énumérer. Nous éviterons ainsi les problèmes d'explosion combinatoire. En effet, si nous devions prendre en compte tous les ordres de passage possibles des ressources sur chacune des n_r sous-gammes $(sg)_\alpha$, il serait alors nécessaire de considérer l'ensemble des graphes d'événements équivalents à la contrainte C_r et nous perdrions de plus tout l'intérêt du caractère flexible de la contrainte, les conflits de ressources devant plutôt être traités en temps réel en fonction des aléas rencontrés durant le fonctionnement effectif de la cellule.

Même en travaillant avec des valeurs moyennes et approximatives, l'expression de la période π définie par l'équation 3.7 n'aura de sens pour un régime stationnaire périodique forcé que si l'on respecte la contrainte

$$\sum_{\alpha=1}^{n_r} pd_{(g)_\alpha} \geq m_{s_r}$$

qui signifie que dans le meilleur des cas, la répartition du travail effectué par un pool de ressources C_r est telle que pour une période nous n'avons pas plus de

ressources que de tâches à effectuer. Il est en effet clair que le fait d'associer deux ressources à une tâche qui n'en nécessite qu'une ne va pas diviser le temps opératoire par deux.

Enfin, nous pouvons remarquer que cette approche n'a d'intérêt que dans le cas de cellules flexibles c'est-à-dire pour lesquelles $n_r > 1$. Si le pool de ressources n'est concerné que par une seule sous-gamme i.e. $n_r = 1$ alors il pourra être éventuellement plus intéressant d'assimiler la contrainte C_r à une contrainte Kanban C_k et d'appliquer l'approche utilisée dans le paragraphe précédent (i.e. l'équation 3.4+le modèle déplié si le nombre de pièces à traiter n'est pas un multiple du nombre de ressources du pool) qui donnera une expression exacte de la période pour le régime stationnaire périodique forcé.

Considérons l'exemple de la figure 2.4 en supposant que la politique de production soit telle que trois produits soient traités sur $(sg)_1$, deux sur $(sg)_2$ et trois sur $(sg)_3$. Nous avons quatre jetons dans la place s_{r2}. Nous pouvons donc déduire de cette contrainte l'équation suivante:

$$\pi = \frac{3}{4}(d_{o_{g1,1}} + d_{o_{g1,2}} + d_{o_{g1,3}}) + \frac{2}{4}(d_{o_{g2,3}} + d_{o_{g2,4}}) + \frac{3}{4}(d_{o_{g5,6}}) + d_{C_{r2}}$$

π et $d_{C_{r2}}$ constituant bien sûr les inconnues réelles positives (ou nulles pour $d_{C_{r2}}$) de cette équation linéaire.

Dans le cas le plus général, les poids de la contrainte cyclique C_c ne sont pas connus et l'équation dérivée du réseau de la figure 2.4 s'écrit:

$$\pi = \frac{(d_{o_{g1,1}} + d_{o_{g1,2}} + d_{o_{g1,3}})}{4}.pd_{g1} + \frac{(d_{o_{g2,3}} + d_{o_{g2,4}})}{4}.pd_{g2} + \frac{(d_{o_{g5,6}})}{4}.pd_{g5} + d_{C_{r2}}$$

Nous aurons alors à considérer les deux inconnues réelles π et $d_{C_{r2}}$ et les trois inconnues entières pd_{g1}, pd_{g2}, et pd_{g5}.

3.3 Analyse des Contraintes

3.3.1 Démarche

Le but de la modélisation est d'aider le concepteur à dimensionner la cellule (nombre de ressources et quantité d'en-cours) et à planifier des séquences opératoires efficaces (ordre de passage des pièces sur les machines). Si l'on part trop tôt sur une méthode de recherche de séquence optimale ou quasi-optimale, le concepteur connaîtra exactement les performances de la cellule (le temps de cycle π) pour cette séquence, mais n'aura aucune évaluation de son comportement pour d'autres séquences et, surtout, ne sera pas capable de savoir sur quel paramètre agir pour modifier les performances si elles ne sont pas satisfaisantes. L'ensemble des contraintes que nous venons de définir (correspondant à des politiques Kanban et à des allocations de ressources) sont agrégées. Elles ne donnent que des conditions nécessaires qui doivent être vérifiées. Néanmoins, l'influence des variables de décision telles que les quantités de ressources (m_{s_r} dans les équations) ou la taille des en-cours (m_{s_k} dans les équations) est claire et leurs conséquences facilement analysables du point de vue de

l'efficacité (valeur de π), de la flexibilité (marges temporelles données par les variables d_{C_k}) et de la criticité des ressources (d_{C_r}). Nous allons maintenant proposer une procédure d'analyse de ces contraintes pour aider à la conception et à l'ordonnancement de cellules flexibles.

3.3.2 Borne Minimale

Comme les temps d'occupation des places par les jetons dans l'état non réservé sont des durées d'attente qui ne peuvent pas être négatives, nous devons introduire la contrainte supplémentaire

$$d_{p_i} \geq 0$$

associée à chaque jeton se trouvant dans une place p_i du modèle global de la cellule. Nous avons donc au niveau des durées d'attente agrégées associées aux jetons des invariants de places:

$$d_{C_k} \geq 0$$

et

$$d_{C_r} \geq 0$$

Le régime stationnaire périodique forcé optimiste correspond à l'efficacité maximale de la cellule vis à vis des taux de production qui sont fixés par la contrainte cyclique C_c. L'efficacité maximale est obtenue pour la valeur minimale de la période π compte tenu de l'ensemble des contraintes à respecter. C'est donc au travers de la minimisation de π sous l'ensemble des contraintes linéaires définies par des équations de la forme 3.2, 3.4 et 3.7 que nous obtiendrons la performance optimale correspondant au régime stationnaire périodique forcé optimiste. Puisque les équations linéaires peuvent avoir des variables entières, un outil approprié à la résolution de ce type de problème peut être la programmation linéaire en variables mixtes [MI 83, NE 88] qui combine une méthode de résolution de type *simplexe* qui travaille sur des valeurs réelles et un algorithme de type *branch and bound* qui permet de définir des valeurs entières pour lesquelles le critère est optimal.

Après la minimisation de π sous l'ensemble des contraintes linéaires, nous obtenons:

- La valeur de la période minimale envisageable π_{min} à partir de laquelle nous pouvons calculer certains indices de performance comme par exemple la quantité de produits d'un même type qui peut être réalisée par unité de temps. Il est intéressant tant pour le dimensionnement de la cellule que pour la gestion de la production de connaître la performance optimale envisageable de la cellule afin de savoir si nous pourrons respecter les exigences de la production. En particulier, cette borne minimale π_{min} peut être utilisée comme une condition nécessaire à respecter dans tous les cas. Quelle que soit la valeur de la période

π à la fin d'un cycle de production, et quel que soit l'ordre d'exécution des opérations, nous devrons vérifier:

$$\pi \geq \pi_{min} \tag{3.8}$$

- L'équilibrage optimal entre les variantes de gammes au sein de la cellule défini par les valeurs des poids pd_{gi} $(i \in 1, ..., N_g)$ de la contrainte cyclique C_c, ces poids étant présents lors de la minimisation de la période dans les équations de type 3.2.

- Les temps d'occupation des places par les jetons de chaque invariant de places (sous une forme agégée) lorsque le temps de cycle est π_{min}. Ces temps d'occupation correspondront aux durées d'attentes agrégées minimales $(d_{C_k})_{min}$ et $(d_{C_r})_{min}$ exprimées lors de la minimisation de la période par les variables réelles d_{C_k} et d_{C_r} des équations de type 3.4 et 3.7. Ces bornes minimales pourront elles aussi être considérées comme des conditions nécessaires à respecter dans tous les cas. En effet, à la fin d'un cycle de production, nous devrons vérifier:

$$d_{C_k} \geq (d_{C_k})_{min}$$

pour les invariants de places de type C_k, et des contraintes de la forme:

$$d_{C_r} \geq (d_{C_r})_{min}$$

pour les invariants de places de type C_r.

La borne π_{min} est atteignable (i.e. un ordonnancement optimal existe) lorsque le modèle global est un graphe d'événements (qui ne comporte donc pas de flexibilité) et que le régime stationnaire périodique libre est considéré [DA 92, RA 80]. En effet, seul l'ordonnancement optimal permettra d'atteindre la valeur π_{min} sous la forme d'une séquence de tir de transitions cyclique (un invariant de transitions) menant d'un marquage M_0 (non nécessairement équivalent au marquage initial du réseau) à l'instant t_0 à un marquage identique M_0 à l'instant $t_0 + \pi_{min}$. Dans notre cas, nous pourrons seulement écrire la condition nécessaire de réalisation de la production 3.8 et éventuellement remettre en cause, si cette condition n'est pas respectée, soit la production envisagée, soit le dimensionnement de la cellule.

La grandeur d_{C_k} est un indicateur de l'utilisation des en-cours. S'il est nul, toute diminution des en-cours dans la sous-gamme correspondante se traduira par un allongement du cycle. La grandeur d_{C_r} est un indicateur de la criticité de la ressource correspondante. S'il est nul, tout incident sur cette ressource aura une conséquence immédiate sur la durée du cycle.

3.3.3 Borne Maximale

Les équations linéaires 3.4 et 3.7 sont déduites à partir de contraintes locales et ne peuvent donc pas prendre en compte toute la globalité du système qu'est une cellule flexible. En particulier, elles ne prennent pas en compte tous les problèmes de synchronisation qui existent au niveau du modèle global et qui peuvent facilement retarder la date de début de certaines opérations pour le régime forcé.

Il doit être important aussi d'inclure de possibles perturbations durant le fonctionnement réel de la cellule comme par exemple des durées supplémentaires d'usinage qui n'apparaissent pas directement sur les t-temporisations du modèle. Les critères d'efficacité et de flexibilité peuvent paraître contradictoires pour un fonctionnement normal mais ils ne le sont pas forcément pour un fonctionnement perturbé devant faire face aux aléas de la production. Il est généralement préférable pour l'efficacité du système de ne pas chercher à respecter une séquence optimale préétablie et d'utiliser les ressources pour d'autres opérations plutôt que de les faire attendre pour des opérations ayant pris du retard.

D'un autre côté, si nous donnons trop de flexibilité au système et que nous ne considérions comme seule contrainte à respecter que

$$\pi \geq \pi_{min}$$

alors nous risquons de perdre beaucoup en efficacité et d'avoir une performance qui ne respecte plus du tout les exigences de la production.

Afin de rendre notre modèle plus robuste vis à vis du fonctionnement réel de la cellule tout en conservant un indice d'efficacité acceptable, nous introduisons des exigences de marges minimales sur certaines opérations en introduisant des contraintes de la forme:

$$d_{p_i} \geq d_f \tag{3.9}$$

avec d_f une marge minimale associée à la place p_i. Nous pouvons ainsi retarder le début de certaines opérations et introduire un degré de flexibilité que nous pouvons contrôler. Comme au niveau des équations linéaires les durées d'attente sont sous une forme agrégée, cette contrainte s'écrit

$$d_{C_k} \geq d_{f_k} \tag{3.10}$$

si p_i appartient à l'invariant de places C_k et

$$d_{C_r} \geq d_{f_r} \tag{3.11}$$

si p_i appartient à l' invariant de places C_r.

d_{f_k} représente la durée d'attente minimale agrégée associée à un jeton (étiquette Kanban) de l'invariant de places C_k et d_{f_r} la durée d'attente minimale agrégée associée à un jeton (ressource) de l'invariant de places C_r.

Il est évident que la principale difficulté de cette approche est de définir les d_{f_k} et d_{f_r} utilisés dans les équations 3.10 et 3.11 pour spécifier la flexibilité que l'on peut donner au système. Chaque cellule possède des caractéristiques propres qui ne peuvent

pas toujours être mises en équation et il est intéressant de pouvoir agir simplement sur le modèle en fonction de l'expérience acquise pour trouver le meilleur compromis possible entre efficacité et flexibilité.

Si nous ne considérons que le modèle théorique de la cellule, nous pouvons en spécifiant des contraintes de la forme 3.10 et 3.11 essayer de prendre en compte les possibles retards qui peuvent exister du fait des diverses synchronisations du modèle global. Considérons par exemple une opération $o_{gi,j}$ d'une gamme gi. Si cette opération nécessite l'utilisation d'une ressource saturée d'une place s_r (nous saurons qu'une ressource est saturée après le calcul de π_{min} si $(d_{C_r})_{min}$ a sa valeur très proche de zéro) alors nous pouvons imposer une marge minimale sur cette opération en associant à sa place d'entrée $s_{gi,j}$ la contrainte

$$d_{s_{gi,j}} \geq \delta_r max$$

avec $\delta_r max$ la durée d'utilisation la plus longue de la ressource sur une autre gamme par exemple.

D'une façon beaucoup plus générale, lorsque nous considèrerons le fonctionnement réel de la cellule, nous pourrons à l'aide de contraintes de type 3.9 prendre en compte toutes sortes de perturbations dont les conséquences directes sont généralement de retarder les débuts de certaines opérations. Cela sera vrai en particulier pour les opérations critiques et il sera alors intéressant d'associer a priori des durées d'attente non nulles aux places d'entrée de ces opérations. L'avantage d'une telle approche sera essentiellement de pouvoir ajouter ou relaxer des contraintes dynamiquement durant l'étape de résolution du problème. Nous laisserons ainsi des possibilités d'intervention de l'opérateur sur le processus de décision.

Lorsque les contraintes supplémentaires de type 3.10 ou 3.11 sont définies, nous les incorporons à l'ensemble des contraintes déjà existantes et nous minimisons la période π en considérant les poids pd_{gi} de la contrainte C_c qui ont été calculés à l'étape précédente. Nous obtenons alors:

- La valeur de la période maximale acceptable π_{max}. Cette valeur maximale exprimera un comportement sous-optimal comportant plus de flexibilité dans la mesure où cette borne correspondra à une performance plus en accord avec le fonctionnement réel de la cellule et délimitera la flexibilité qui pourra être donnée au système sans pour autant perdre toute efficacité. Nous parlerons alors d'un régime stationnaire périodique forcé pessimiste et la politique de conduite retenue devra aboutir à une période π vérifiant:

$$\pi_{min} \leq \pi \leq \pi_{max} \tag{3.12}$$

- Les temps d'occupation des places par les jetons de chaque invariant de places (sous une forme agrégée) lorsque le temps de cycle est π_{max}. Ces temps d'occupation correspondront aux durées d'attente agrégées maximales $(d_{C_k})_{max}$ et $(d_{C_r})_{max}$. A la fin d'un cycle de production, nous devrons vérifier:

$$(d_{C_k})_{min} \leq d_{C_k} \leq (d_{C_k})_{max}$$

pour les invariants de places de type C_k, et:

$$(d_{C_r})_{min} \leq d_{C_r} \leq (d_{C_r})_{max}$$

pour les invariants de places de type C_r.

Précisons que dans ce cas, les contraintes mises en jeu lors de la minimisation de π sont extrêmement simple. En particulier, elles ne contiennent plus de variables entières. Il ne sera donc pas nécessaire d'utiliser une méthode aussi complexe que la programmation mathématique pour le calcul de π_{max}.

3.4 Calcul d'une Séquence Admissible?

Les bornes π_{min} et π_{max} ne sont que des conditions nécessaires de réalisation de la production. Elles ne garantissent pas qu'il existe une séquence admissible qui vérifie bien la contrainte 3.12. La recherche d'une telle séquence est essentielle pour nous indiquer de façon précise les zones du système où il y a un manque de flexibilité qui se traduit lors du calcul de l'ordonnancement soit par un échec dans la zone considérée (violation d'une contrainte) soit simplement par la consommation totale de la flexibilité (c'est-à-dire que la durée d'attente maximale associée à un jeton d'un invariant de places est utilisée dans son intégralité). Comme cette approche a pour objectif final de définir une politique de décision en temps réel qui soit à la fois robuste et efficace, il est absolument indispensable de vérifier au préalable l'existence d'une "bonne solution". Nous allons donc exposer dans le chapitre suivant le principe de fonctionnement d'un joueur de réseau de Petri p-temporel t-temporisé pour le calcul d'un ordonnancement prévisionnel qui devra nous donner les garanties suffisantes à l'élaboration d'une politique de décision temps réel bien adaptée aux exigences de la production et à la cellule utilisée.

Chapitre 4

Ordonnancement et Pilotage Temps Réel d'une Cellule Flexible

4.1 Réseaux de Petri et Mécanismes d'Aide à la Décision

Si nous considérons l'ensemble des synchronisations mises en jeu au niveau d'une cellule flexible alors les réseaux de Petri sont un modèle particulièrement bien adapté puisqu'ils permettent de modéliser les conflits existants dans toute leur complexité. En d'autres termes, ils autorisent au travers d'une représentation simple et compacte d'énumérer toutes les possibilités d'ordonnancement des diverses opérations de la cellule.

D'un autre côté, les mécanismes d'aide à la décision en présence d'un ensemble de contraintes temporelles explicites qu'un réseau de Petri peut représenter se résument à des simulations. Bien sûr, les mécanismes logiques d'allocation de ressources sont formellement spécifiés et les situations de conflits (situations dans lesquelles des décisions doivent être prises) sont mises en évidence [AT1 87]. Toutefois, les situations de conflits sont analysées par ordre de temps croissant et non par ordre de criticité (les conflits ayant le plus de conséquence seraient alors résolus les premiers indépendamment de leur date d'occurrence).

La logique, pour sa part, ainsi que les diverses techniques d'Intelligence Artificielle n'abordent pas la notion de conflit avec la même richesse que les réseaux de Petri mais elles permettent de construire des mécanismes de décision très efficaces utilisant des procédures de propagation de contraintes et de retour arrière ("backtracking"). D'une façon générale, lorsque l'on raisonne pour prendre une décision dans une situation donnée, cela relève plutôt de la logique classique.

Au-delà de la nécessité de résoudre l'ensemble des conflits existants, soulignons que les décisions prises doivent essentiellement être fondées sur des considérations temporelles pour avoir une influence bénéfique sur la production, ce qui justifie l'utilisation d'un modèle temporel/temporisé et son analyse préalable afin qu'une certaine connaissance globale puisse être utilisée lors de la résolution d'un conflit local.

Ceci nous amène à la conclusion que pour l'ordonnancement de cellules flexibles,

il sera intéressant d'avoir une approche hybride qui combine les réseaux de Petri et certaines techniques de l'Intelligence Artificielle utilisées dans les systèmes d'aide à la décision. Nous utiliserons le formalisme des réseaux de Petri (p-temporel t-temporisé) pour mettre en évidence les situations de conflit, mais les intervalles de visibilité des jetons dans les places ne seront pas uniquement utilisés pour effectuer une simulation. Ces intervalles seront prédéfinis par une heuristique de désagrégation des contraintes agrégées définies au chapitre précédent, puis seront considérés comme des contraintes à respecter lors de la simulation du réseau de Petri par un *"joueur"*. En particulier, ces contraintes supplémentaires aideront à prendre des décisions dans les situations de conflit: souvent un seul ordre d'exécution sera possible, ou alors un seul ordre ne réduira pas les intervalles de façon anormale. Comme nous partirons des contraintes agrégées globales (calculées dans le chapitre précédent), la résolution d'un conflit ne dépendra pas que du passé comme dans la simulation. Les contraintes futures seront prises en compte, même si cela n'est possible que de façon implicite et partielle. En un mot, nous allons définir un algorithme de joueur de réseau de Petri p-temporel t-temporisé qui aura pour objectif la recherche d'une séquence d'opérations admissible compte tenu du respect de l'ensemble des contraintes associées au réseau.

4.2 Contraintes Temporelles Attachées au Modèle de la Cellule

Comme nous l'avons dit, les durées d'opération sont représentées par des t-temporisations attachées aux transitions du modèle; notre modèle est donc t-temporisé. Si nous ne considérons qu'un modèle t-temporisé, nous n'aurons pas d'indication particulière sur la durée pendant laquelle un jeton se trouvant dans une place p peut rester dans l'état non réservé, même si la transition de sortie t de cette place est immédiatement franchissable (pas d'hypothèse de franchissement au plus tôt). Le jeton d'une place p peut rester dans l'état non réservé pendant une durée $d_p \in [0; +\infty[$.

Lorsque l'on calcule un ordonnancement par simple simulation du réseau de Petri, pour ne pas avoir à considérer tous les conflits qui peuvent exister sur cet intervalle, on applique une politique de tir au plus tôt pour la recherche d'une séquence minimisant, dans la mesure du possible, la durée cyclique, ce qui a pour effet de réduire considérablement l'espace de recherche et bien sûr de faire disparaitre bon nombre de solutions parfois mieux adaptées.

Une autre solution consiste à construire tous les graphes d'événements possibles exprimant des ordres stricts d'utilisation des machines, puis à choisir le meilleur. Le résultat de cette approche, en plus de mettre en œuvre des calculs très lourds, reste très difficilement exploitable au niveau du pilotage temps réel d'une cellule. La solution n'est en général pas robuste aux aléas qui existent lors du fonctionnement temps réel puisque l'on ne peut pas remettre en cause l'ordre d'exécution des opérations sur les machines (le graphe d'événements est figé).

Afin de réduire l'espace de recherche tout en laissant un degré de flexibilité susceptible de rendre la politique de contrôle plus robuste, nous avons vu l'intérêt d'associer à chaque place un intervalle min/max. Ceci nous a amené dans le deuxième chapitre

à définir les réseaux de Petri p-temporels t-temporisés qui ont un intervalle statique attaché à chaque place du modèle.

Après l'analyse quantitative du modèle p-temporel t-temporisé effectuée dans le troisième chapitre (le calcul des marges temporelles minimales et maximales associées à chaque invariant de places du modèle), nous n'avons d'information que sur la durée d'attente globale associée à chaque contrainte de la cellule pour un cycle de fabrication ou de façon plus formelle la durée d'occupation des places de l'invariant correspondant par l'ensemble des jetons se trouvant dans l'état indisponible sur une période et appartenant à cet invariant de places. Ainsi, même en ayant calculé des bornes définissant en particulier le maximum de flexibilité exploitable pour l'ordonnancement des opérations, nous aurons toujours à considérer une marge très importante à l'intérieur de laquelle existera un trop grand nombre de séquences possibles si on envisage la possibilité de consommer toute cette marge dès la première opération.

Puisque le modèle utilisé est p-temporel (t-temporisé), nous pouvons désagréger pour un invariant de places donné les marges temporelles associées en distribuant directement sur chaque place sous forme de bornes min/max une fraction des durées globales d'attente minimales et maximales. Des contraintes temporelles plus précises peuvent et doivent ainsi être directement attachées aux places du modèle sous forme d'intervalles statiques associés aux places. Le rôle de ces contraintes supplémentaires sera de restreindre l'espace de recherche et de rendre plus simple pour le mécanisme de décision (le joueur) la recherche d'une solution admissible à la fois robuste et efficace. Cette désagrégation sera fondée sur une heuristique.

4.2.1 Contraintes Temporelles pour une Contrainte Kanban

L'invariant de transitions qui recouvre le modèle global de la cellule nous indique qu'une transition $o_{g,m_{sg}+i}$ d'une contrainte Kanban C_k est franchie pd_g fois pour un cycle de fabrication. La place d'entrée $s_{g,m_{sg}+i}$ de cette transition est donc traversée successivement par pd_g jetons pour qu'un cycle se complète et nous devons attacher à chacun de ces jetons une durée d'attente $d_{s_{g,m_{sg}+i}}$ qui doit appartenir à l'intervalle statique $[(d_{s_{g,m_{sg}+i}})_{min}; (d_{s_{g,m_{sg}+i}})_{max}]$ qui est associé à la place $s_{g,m_{sg}+i}$. Cette durée d'attente représente l'attente du jeton avant sa réservation pour le franchissement d'une transition et décrit la stratégie de pilotage choisie.

Il n'y a rien a priori dans la définition d'un réseau p-temporel qui spécifie que les valeurs minimale $(d_{s_{g,m_{sg}+i}})_{min}$ et maximale $(d_{s_{g,m_{sg}+i}})_{max}$ de l'intervalle statique soient à valeur constante. Dans notre cas, cet intervalle doit dépendre de la place à laquelle il est attaché et aussi de la position courante dans l'exécution de la séquence cyclique décrivant le cycle de fabrication. Nous notons la durée d'attente du k^{ieme} jeton qui occupe la place $s_{g,m_{sg}+i}$

$$(d_{s_{g,m_{sg}+i}})^k,$$

elle doit appartenir à l'intervalle statique noté

$$[(d_{s_{g,m_{sg}+i}})_{min^k}; (d_{s_{g,m_{sg}+i}})_{max^k}], \tag{4.1}$$

et nous avons

$$k \in \{1, ..., pd_g\}.$$

Si la place d'entrée de $o_{g,m_{sg}+i}$ est la place Kanban s_k alors cette dernière sera traversée successivement par pd_g jetons et la fin du cycle sera indiquée par le retour de l'ensemble des jetons dans cette place afin de retrouver le marquage initial. En plus des pd_g durées d'attente qui vont précéder les pd_g débuts d'opération $o_{g,m_{sg}+i}$, nous devons prendre en compte la durée d'attente globale de fin de cycle qui est modélisée par une durée d'attente supplémentaire associée à chacune des étiquettes Kanban puisque le régime stationnaire cyclique est forcé et que l'on doit retrouver l'état initial à la fin de chaque cycle. Si la contrainte Kanban C_k contient m_{s_k} jetons ($M_k(s_k) = m_{s_k}$ est un multiple de pd_g) alors nous notons la durée d'attente du k^{ieme} jeton qui occupe la place s_k

$$(d_{s_k})^k,$$

elle doit appartenir à l'intervalle statique noté

$$[(d_{s_k})_{min^k}; (d_{s_k})_{max^k}], \tag{4.2}$$

et nous avons

$$k \in \{1, ..., pd_g, pd_g + 1, ..., pd_g + m_{s_k}\}.$$

En résumé, nous devons donc attacher à une place stock intermédiaire pd_g intervalles statiques distincts et à la place Kanban $pd_g + m_{s_k}$ intervalles statiques distincts. Nous supposons ici que les étiquettes Kanban se trouvent toutes dans la place s_k pour le marquage initial bien sûr. Dans le cas général, ce seront les places marquées initialement qui auront un intervalle supplémentaire pour chaque jeton initial.

L'ensemble des contraintes temporelles à considérer au niveau de l'ordonnancement des opérations de la cellule pour une contrainte Kanban C_k sont donc:

- La marge temporelle agrégée qui vérifie $(d_{C_k})_{min} \leq d_{C_k} \leq (d_{C_k})_{max}$ et qui est associée à chacune des étiquettes Kanban de C_k.
- L'ensemble des intervalles statiques associés à l'ensemble des places de la contrainte Kanban. Ces intervalles sont vus comme des contraintes désagrégées qui ont pour objectif de réduire le domaine de décision à l'intérieur duquel doivent être résolus les conflits ressource.

C'est l'opérateur qui devra définir "manuellement" les intervalles statiques (c'est-à-dire définir les mécanismes de désagrégation) afin de diriger la recherche d'une séquence admissible. Ces intervalles vont dépendre bien évidemment de la place à laquelle ils appartiennent et de leur position dans le cycle. Nous devrons éviter que ces contraintes désagrégées ne servent à rien ou qu'elles soient de façon évidente incohérentes avec les contraintes agrégées. La politique de désagrégation sera de toute

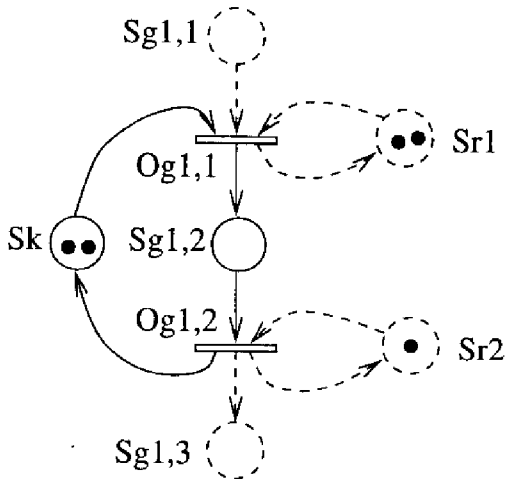

Figure 4.1: Exemple 1 de Contraintes Temporelles associées à une Contrainte Kanban

façon fonction d'heuristiques basées sur la connaissance du système par le concepteur de la cellule.

Considérons la contrainte Kanban représentée par la figure 4.1. La marge temporelle associée à chacune des étiquettes Kanban est telle que: $2 \leq d_{C_k} \leq 4$. La contrainte cyclique qui fixe les taux de production de la cellule est telle que $pd_{g1} = 2$ c'est-à-dire que pour un cycle de fabrication deux pièces seront traitées suivant la gamme $g1$. L'état initial du cycle au niveau de la contrainte Kanban représentée par la figure 4.1 est tel que les deux étiquettes en début de cycle se trouvent dans la place s_k dans l'état non réservé. Pour un cycle, la place $s_{g1,2}$ sera traversée par deux étiquettes Kanban et nous devons associer à cette place deux intervalles statiques. Comme la place s_k est la place de début de cycle, nous devons associer à cette place quatre intervalles statiques (les deux derniers représentant le retour des jetons dans la place s_k en fin de cycle).

Nous choisissons comme intervalles statiques associés à $s_{g1,2}$:

$[0;2]_1$

$[0;2]_2$

En effet, comme il n'y a qu'une ressource en s_{r2}, si les deux jetons Kanban sollicitent en même temps la ressource pour effectuer l'opération $o_{g1,2}$ sur une pièce, alors une des deux attendra dans le pire des cas la durée de l'opération $d_{o_{g1,2}} = 2$ quel que soit l'état initial du système.

Comme nous avons deux ressources en s_{r1} nous n'attendrons jamais en s_k pour débuter l'opération $o_{g1,1}$ et les deux premiers intervalles statiques associés à s_k sont donc:

$[0;0]_1$

$[0;0]_2$

En fin de cycle les étiquettes Kanban devront attendre dans la place s_k dans l'état indisponible jusqu'à ce que toutes les étiquettes Kanban du réseau global de la cellule soient dans leur place de début de cycle. Nous choisissons donc comme intervalles:

$[4;4]_3$

$[4;4]_4$

puisqu'au-delà de la valeur 4 nous aurons de toute façon violation de la marge temporelle associée à cette contrainte. Cela veut dire que les jetons resteront dans l'état indisponible, sans sensibiliser de nouvelle transition, jusqu'au franchissement de la transition de fin de cycle qui provoquera la réinitialisation de tous les intervalles.

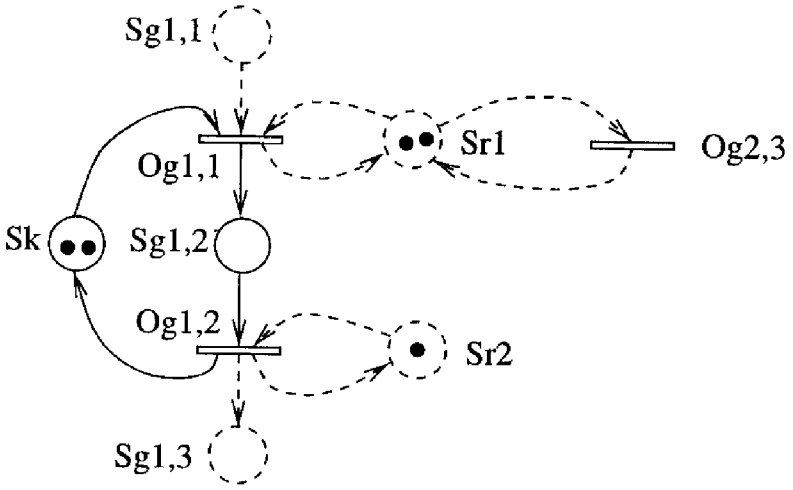

Figure 4.2: Exemple 2 de Contraintes Temporelles Associées à une Contrainte Kanban

Les ressources de s_{r1} sont maintenant partagées entre les opérations $o_{g1,1}$ et $o_{g2,3}$. La durée d'opération associée à $o_{g2,3}$ est: $d_{o_{g2,3}} = 4$.

Si une étiquette Kanban de s_k sollicite une ressource de s_{r1} pour effectuer l'opération $o_{g1,1}$ mais que la ressource est déjà réservée pour l'opération $o_{g2,3}$, alors on attendra dans le pire des cas en s_k la durée d'opération $d_{o_{g2,3}} = 4$. Nous choisissons donc comme deux premiers intervalles associés à s_k:

$[0;4]_1$

$[0;4]_2$

Nous voyons qu'en choisissant comme borne maximale de ces intervalles la valeur 4, nous pouvons consommer toute la marge temporelle associée aux étiquettes Kanban dès le début du cycle et nous n'aurons plus aucune flexibilité par la suite. En particulier nous pourrons avoir une violation de contrainte si nous attendons en $s_{g1,2}$. Pour mieux répartir la flexibilité sur tout le cycle, nous choisissons plutôt comme intervalles de début de cycle associés à s_k:

$[0;2]_1$

$[0;2]_2$

(ceux associés à $s_{g1,2}$ restent les mêmes). Ainsi, toute la flexibilité ne pourra pas être consommée dès le début du cycle. De plus, si à un instant particulier apparaît un conflit ressource entre $o_{g1,1}$ et $o_{g2,3}$, alors le mécanisme de résolution de conflit devra impérativement donner la priorité à l'opération $o_{g1,1}$ puisque sinon il y aura forcément par la suite une violation de contrainte temporelle en s_k. Nous reviendrons plus en détail sur ce point lorsque nous présenterons la procédure de résolution de conflit.

4.2.2 Contraintes Temporelles pour une Contrainte de Ressource

Une ressource disponible peut très bien ne pas être utilisée immédiatement; soit parce qu'aucune opération l'utilisant n'est à exécuter, soit parce que l'on a décidé de retarder une opération pour affecter ultérieurement la ressource à une opération plus urgente. Cela se traduit par une durée d'attente d_{s_r} attachée à chaque jeton ressource de s_r et qui doit appartenir à l'intervalle statique $[(d_{s_r})_{min}; (d_{s_r})_{max}]$ associé à la place s_r. Pendant cette durée, la ressource correspondante est en attente et n'est pas allouée pour l'exécution d'une opération.

Une place ressource s_r d'une contrainte C_r est place d'entrée des n_r premières opérations $o_{(g)_\alpha, m_{(sg)_\alpha}+1}$ de n_r sous-gammes $(sg)_\alpha$. Chacune de ces opérations est exécutée $pd_{(g)_\alpha}$ fois pour qu'un cycle se complète. Nous devons donc considérer en tout $\sum_{\alpha=1}^{n_r} pd_{(g)_\alpha}$ durées d'attente associées à l'ensemble des jetons ressource qui seront utilisés pour l'exécution des premières opérations des n_r sous-gammes $(sg)_\alpha$.

Si la contrainte C_r contient m_{s_r} jetons ($M_r(s_r) = m_{s_r}$) alors la fin du cycle est indiquée par le retour de l'ensemble de ces jetons dans la place s_r afin de retrouver le marquage initial. Nous devons donc prendre en compte la durée d'attente de fin de cycle qui est modélisée par une durée d'attente supplémentaire associée à chacun des jetons ressource puisque le régime stationnaire cyclique est forcé et que l'on doit retrouver le marquage initial à la fin de chaque cycle (nous supposons que chaque jeton est au moins utilisé une fois).

Nous notons la durée d'attente du k^{ieme} jeton qui occupe la place s_r

$$(d_{s_r})^k,$$

elle doit appartenir à l'intervalle statique noté

$$[(d_{s_r})_{min^k}; (d_{s_r})_{max^k}], \tag{4.3}$$

et nous avons

$$k \in \{1, ..., \sum_{\alpha=1}^{n_r} pd_{(g)_\alpha}, \sum_{\alpha=1}^{n_r} pd_{(g)_\alpha} + 1, ..., \sum_{\alpha=1}^{n_r} pd_{(g)_\alpha} + m_{s_r}\}$$

En résumé, nous devons donc attacher à une place s_r, $\sum_{\alpha=1}^{n_r} pd_{(g)_\alpha} + m_{s_r}$ intervalles statiques distincts pour chaque cycle de fabrication.

Après le calcul des marges temporelles minimales et maximales qui a été présenté dans le chapitre précédent, nous avons une marge temporelle moyenne d_{C_r} associée à chaque ressource qui doit vérifier:

$$(d_{C_r})_{min} \leq d_{C_r} \leq (d_{C_r})_{max}$$

Comme le travail est difficilement réparti de façon égale sur chacune des ressources puisqu'une cellule flexible est un système à événements discrets, lorsque nous avons plusieurs ressources dans un même pool (place s_r), nous devons associer à la contrainte C_r une marge temporelle agrégée D_{C_r} qui doit vérifier:

$$m_{s_r}.(d_{C_r})_{min} \leq D_{C_r} \leq m_{s_r}.(d_{C_r})_{max} \quad (4.4)$$

Cette marge agrégée délimite l'attente globale de non utilisation de l'ensemble des ressources du pool sur une période.

Les contraintes temporelles statiques à considérer au niveau de l'ordonnancement des opérations de la cellule pour une contrainte de ressource C_r sont donc:

- La marge temporelle agrégée totale qui vérifie 4.4.
- L'ensemble des intervalles statiques associés aux places des sous-gammes utilisant la ressource s_r (attente des jetons entre deux opérations sans libération de la ressource) et à la place ressource s_r.

Ici aussi, c'est l'opérateur qui devra définir "manuellement" les intervalles statiques afin de diriger la recherche d'une séquence admissible.

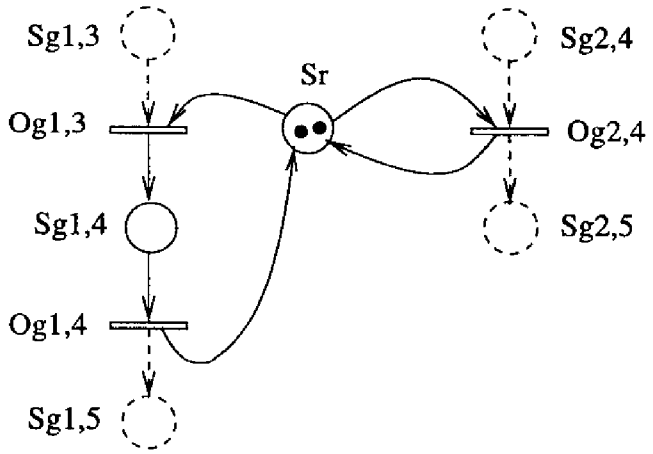

Figure 4.3: Contraintes Temporelles Associées à une Contrainte de Ressource

Considérons la contrainte de ressource représentée par la figure 4.3. La marge temporelle agrégée totale associée à la contrainte C_r est telle que: $0 \leq D_{C_r} \leq 7$. La contrainte cyclique qui fixe les taux de production de la cellule est telle que $pd_{g1} = 2$ et $pd_{g2} = 1$ c'est-à-dire que pour un cycle de fabrication deux pièces seront traitées suivant la gamme $g1$ et une pièce sera traitée suivant la gamme $g2$. L'état initial du cycle pour la contrainte de ressource représentée par la figure 4.3 est tel que les deux ressources en début de cycle se trouvent dans la place s_r dans l'état non réservé. Pour un cycle, la place $s_{g1,4}$ sera traversée par deux jetons ressource et nous devons associer à cette place deux intervalles statiques. Comme la place s_r est la place de début de cycle, nous devons associer à cette place cinq intervalles statiques. Deux pour les deux débuts d'opération $o_{g1,3}$, un pour le début d'opération $o_{g2,4}$ et les deux derniers représentant le retour des jetons ressource dans la place s_r en fin de cycle.

Nous voyons que la valeur minimale de la marge temporelle agrégée totale de C_r est 0, ce qui veut dire que les ressources du pool sont saturées. Par conséquent, elles

devront être dans l'état réservé pratiquement durant tout le cycle. Nous choisissons donc comme intervalles statiques associés à $s_{g1,4}$:

$[0;0]_1$

$[0;0]_2$

Pour laisser le maximum de flexibilité au niveau de l'ordre d'utilisation des ressources du pool s_r, nous associons à s_r les intervalles:

$[0;7]_1$

$[0;7]_2$

$[0;7]_3$

Cela veut dire que la politique d'utilisation des ressources pour les opérations $o_{g1,3}$, $o_{g1,4}$ et $o_{g2,4}$ sera plutôt fixée par l'ordre de passage des étiquettes Kanban dans les stocks des gammes. Ce sera plutôt au niveau des intervalles statiques associés aux places "stock intermédiaire" que l'on raisonnera pour résoudre les possibles conflits ressource. En effet, il n'y aurait pas beaucoup de sens à diminuer la flexibilité associée aux ressources du pool.

En fin de cycle, les jetons ressource devront attendre dans la place s_r dans l'état indisponible jusqu'à ce que toutes les étiquettes Kanban du réseau global de la cellule soient dans leur place de début de cycle. Nous choisissons donc comme intervalles:

$[7;7]_4$

$[7;7]_5$

puisqu'au-delà de la valeur 7 nous aurons de toute façon violation de la marge temporelle associée à cette contrainte.

Puisque les ressources de s_r sont saturées, elles devront pouvoir être utilisées pour les opérations $o_{g1,3}$ et $o_{g2,4}$ dès qu'elles sont disponibles. Il faut qu'il y ait si possible toujours en attente une pièce dans $s_{g1,3}$ (un jeton disponible) et dans $s_{g2,4}$. Il sera donc important que la valeur des bornes maximales des intervalles statiques associés à ces places soit la plus grande possible. Par conséquent, le choix des intervalles associés aux places d'entrée des transitions opération utilisant des ressources saturées, même si ces places n'appartiennent pas directement à une contrainte de ressource, peut dépendre aussi bien de la marge temporelle associée à la contrainte Kanban à laquelle ces places appartiennent, qu'à la valeur de la marge temporelle associée à la contrainte ressource à laquelle appartiennent les ressources saturées utilisées.

4.3 Notion de Conflit pour un Réseau de Petri p-temporel t-temporisé

Les réseaux de Petri permettent de représenter les conflits existant au sein d'une cellule flexible dans toute leur complexité, c'est-à-dire qu'ils permettent d'énumérer toutes les possibilités d'ordonnancement des opérations. Dans un réseau de Petri autonome, le temps n'est pas explicité et l'information importante que donne le réseau de Petri est que l'on sait quelles sont les opérations qui peuvent être exécutées en parallèle et quelles sont celles qui sont en exclusion mutuelle (conflit dans le sens des transitions en conflit pour un marquage). Quand on explicite le temps, le problème est plus complexe car l'état du système n'est pas simplement donné par le marquage

courant. La notion de conflit peut donc changer selon le modèle de réseau de Petri utilisé. Nous allons rappeler dans un premier temps certaines définitions de base. Nous présenterons ensuite la notion de conflit pour un réseau p-temporel t-temporisé qui est le modèle à partir duquel sera calculé l'ordonnancement des opérations de la cellule.

4.3.1 Conflit pour un Réseau de Petri Autonome

Définition 4.3.1 Conflit structurel: *Deux transitions $t1$ et $t2$ sont en conflit structurel si et seulement si elles ont au moins une place p d'entrée en commun:*

$$\exists p \quad Pre(p, t1).Pre(p, t2) \neq 0$$

Définition 4.3.2 Conflit effectif: *Deux transitions $t1$ et $t2$ sont en conflit effectif pour un marquage M si et seulement si elles sont en conflit structurel et que:*

$$M \geq Pre(., t1)$$
$$M \geq Pre(., t2)$$

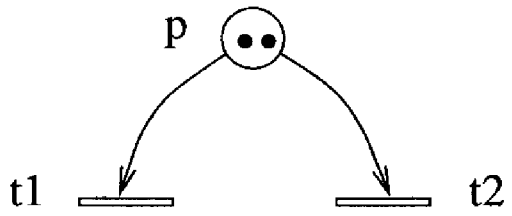

Figure 4.4: Notion de conflit pour un réseau de Petri autonome

S'il y a un conflit structurel et que le marquage M soit tel que les deux transitions soient sensibilisées alors nous aurons plusieurs alternatives à prendre en compte. Considérons par exemple le réseau autonome de la figure 4.4. Comme nous avons deux jetons dans la place d'entrée p, il y a une décision irréversible à prendre entre trois alternatives: franchir deux fois $t1$, ou franchir deux fois $t2$, ou franchir une fois $t1$ et une fois $t2$. Si le réseau de Petri décrit un ensemble de processus partageant un ensemble de ressources, il y aura trois séquencements possibles entre lesquels il faudra choisir.

4.3.2 Conflit pour un Réseau de Petri t-temporisé

L'état d'un système décrit par un tel réseau est donné par le marquage courant associé au temps écoulé pour chaque jeton réservé. Connaissant le temps associé aux transitions on en déduit la date de fin de franchissement. La dynamique ne sera en fait complètement définie que si l'on précise si les jetons sont réservés dès que les transitions deviennent sensibilisées (tir au plus tôt). Dans le cas contraire, la durée pendant laquelle un jeton d'une place p peut rester dans l'état non réservé avant le franchissement d'une transition t qu'il sensibilise appartient à l'intervalle $[0; +\infty[$.

C'est pourquoi, pour ne pas avoir à considérer l'ensemble des conflits pouvant exister sur un tel intervalle, généralement, pour les problèmes d'ordonnancement modélisés par des réseaux de Petri t-temporisés, seule la stratégie de tir au plus tôt des transitions est prise en compte [LE1 94, LE2 94, AZ 94]. Dès qu'un jeton sensibilise une transition t à laquelle est attachée une durée d_t, il est immédiatement réservé pour le tir de cette transition qui aura lieu après que la durée d_t se soit écoulée. Si nous ne considérons que les jetons non réservés d'un réseau t-temporisé dans le cadre d'une stratégie de tir au plus tôt alors la définition de *conflit effectif* des réseaux de Petri autonomes reste vraie pour les réseaux de Petri t-temporisés.

4.3.3 Conflit pour un Réseau de Petri p-temporel t-temporisé

La notion de conflit pour un réseau p-temporel t-temporisé, dans la mesure où la stratégie de tir au plus tôt n'est pas la seule à considérer, devient plus complexe. Les conflits pour une ressource sont pris en considération durant un intervalle de temps et non pas seulement à un instant précis et il est possible de générer des séquences pour lesquelles les transitions ne sont pas nécessairement tirées dès que les jetons sont présents dans les places d'entrée. En jouant sur les dates de disponibilité des jetons, on peux faire varier de façon continue la date de franchissement des transitions qu'ils sensibilisent.

Nous allons maintenant proposer plusieurs définitions se rapportant directement à la notion de conflit d'un réseau de Petri p-temporel t-temporisé.

Définition 4.3.3 Intervalle de visibilité d'un jeton dans une place: *Un intervalle de visibilité* $[(\delta_p)_{min}; (\delta_p)_{max}]$ *associé à un jeton d'une place p d'un réseau p-temporel t-temporisé définit la date au plus tôt* $(\delta_p)_{min}$ *à partir de laquelle le jeton se trouvant dans la place p devient disponible et peut être réservé pour le franchissement d'une transition t de sortie de la place p, et la date au plus tard* $(\delta_p)_{max}$ *après laquelle on a "mort" du jeton qui ne peut donc plus être utilisé pour le tir d'aucune transition.*

Précisons en nous basant sur la définition du modèle de Khansa [KH 96] que l'on a "mort" d'un jeton lorsque pendant l'intervalle $[(\delta_p)_{min}; (\delta_p)_{max}]$ associé à un jeton d'une place p aucune transition de sortie de cette place n'est franchissable. Dans le contexte de notre problème, cela voudra dire que l'on aura viol d'une contrainte temporelle et que dans ce cas, par rapport à l'ordonnancement des opérations de la cellule, on devra soit modifier la stratégie de désagrégation des marges temporelles associées aux invariants de places du modèle (augmenter la valeur de la borne maximale de l'intervalle statique à partir duquel est calculé l'intervalle de visibilité du jeton et diminuer la borne maximale d'un autre intervalle statique associé à une autre place du même invariant de places), soit faire un retour arrière (backtrack) au niveau de la séquence calculée et chercher dans une autre direction une autre séquence admissible.

Définition 4.3.4 Intervalle de sensibilisation d'une transition: *Si une transition t possède n places d'entrée contenant chacune un ou plusieurs jetons alors un*

intervalle de sensibilisation de cette transition $[(\theta_t)_{min}, (\theta_t)_{max}]$ *est obtenu en choisissant pour chacune des n places un jeton et son intervalle de visibilité et en faisant l'intersection de ces intervalles de visibilité.*

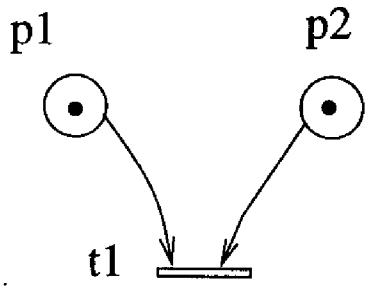

Figure 4.5: Premier exemple d'intervalle de sensibilisation d'une transition

Considérons le réseau de la figure 4.5. Si les intervalles de visibilité associé au jeton de p_1 et de p_2 sont
$[(\delta_{p_1})_{min}; (\delta_{p_1})_{max}] = [3; 6]$
$[(\delta_{p_2})_{min}; (\delta_{p_2})_{max}] = [4; 7]$
alors l'intervalle de sensibilisation de t_1 est
$[(\theta_{t_1})_{min}, (\theta_{t_1})_{max}] = [4; 6]$.

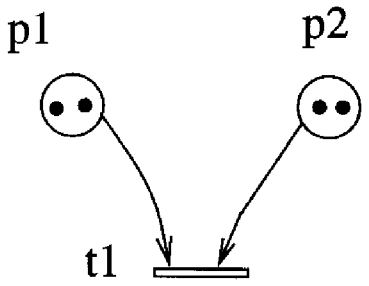

Figure 4.6: Deuxième exemple d'intervalle de sensibilisation d'une transition

Considérons maintenant le réseau de la figure 4.6. Les intervalles de visibilité sont les suivant:
$[(\delta_{p_1})_{min}; (\delta_{p_1})_{max}]_1 = [3; 6]$
$[(\delta_{p_1})_{min}; (\delta_{p_1})_{max}]_2 = [8; 15]$
$[(\delta_{p_2})_{min}; (\delta_{p_2})_{max}]_1 = [4; 7]$
$[(\delta_{p_2})_{min}; (\delta_{p_2})_{max}]_2 = [5; 10]$
Comme nous avons deux intervalles de visibilité pour chaque place, nous devons faire un choix. Comme nous le voyons sur la figure 4.7, si nous faisons l'intersection de $[(\delta_{p_1})_{min}; (\delta_{p_1})_{max}]_1$ et de $[(\delta_{p_2})_{min}; (\delta_{p_2})_{max}]_1$ alors l'intervalle de sensibilisation est toujours $[4; 6]$. Nous pouvons aussi choisir de faire l'intersection de $[(\delta_{p_1})_{min}; (\delta_{p_1})_{max}]_1$ avec $[(\delta_{p_2})_{min}; (\delta_{p_2})_{max}]_2$ et l'intervalle de sensibilisation est alors $[5; 6]$. Si nous faisons

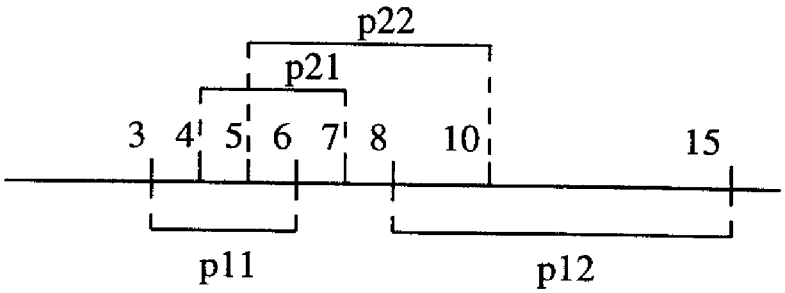

Figure 4.7: Ensemble des intervalles possibles

ce choix, l'intersection entre $[(\delta_{p_1})_{min};(\delta_{p_1})_{max}]_2$ et $[(\delta_{p_2})_{min};(\delta_{p_2})_{max}]_1$ sera égale à l'ensemble vide. Nous ne pourrons franchir pour ce marquage du réseau la transition t_1 qu'une fois et nous aurons une violation des contraintes temporelles puisque l'on aura "mort" d'un des jetons de p_1 (idem pour le jeton de p_2). Donc nous ne conserverons que la première solution puisque notre approche est basée sur l'analyse sous contraintes et que l'objectif est la recherche d'une solution acceptable respectant l'ensemble des contraintes. Le premier intervalle de sensibilisation de t_1 sera $[4;6]$ et le deuxième $[8;10]$.

Définition 4.3.5 Intervalle de temps de conflit: *Si deux transitions t_1 et t_2 sont en conflit structurel alors l'intervalle de temps de conflit de ces transitions est obtenu en faisant l'intersection des intervalles de sensibilisation des transitions t_1 et t_2. Pendant cet intervalle de temps, t_1 et t_2 sont en conflit effectif pour le marquage. Hors de cet intervalle, elles ne sont plus en conflit effectif.*

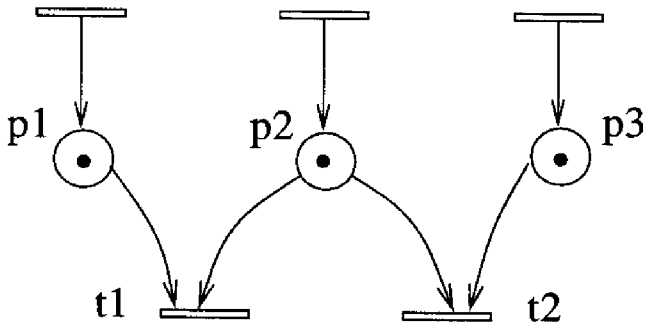

Figure 4.8: Notion de conflit pour un réseau de Petri p-temporel t-temporisé

Considérons l'exemple de la figure 4.8. Les intervalles statiques associés aux places sont:

$[(d_{p1})_{min};(d_{p1})_{max}] = [1,6]$,
$[(d_{p2})_{min};(d_{p2})_{max}] = [0,7]$,
$[(d_{p3})_{min};(d_{p3})_{max}] = [2,6]$.

A la date 0 un jeton arrive en p_1, à la date 2 un jeton arrive en p_3, et à la date 3 un jeton arrive en p_2. Nous pouvons en déduire les intervalles de visibilité associés aux jetons des places p_1, p_2 et p_3:

$[(\delta_{p1})_{min}; (\delta_{p1})_{max}] = [1, 6]$,
$[(\delta_{p2})_{min}; (\delta_{p2})_{max}] = [3, 10]$,
$[(\delta_{p3})_{min}; (\delta_{p3})_{max}] = [4, 8]$.

Les intervalles de sensibilisation sont $[3, 6]$ pour t_1 et $[4, 8]$ pour t_2. L'intervalle de conflit associé au couple (t_1, t_2) est donc $[4, 6]$.

4.4 Résolution de Conflit pour l'Ordonnancement et le Pilotage

4.4.1 Obtention d'un Etat Isolant un Conflit

Lorsqu'un jeton de type "pièce" sensibilise une transition qui est elle même en conflit structurel avec d'autres transitions, le système de décision associé au joueur chargé d'ordonner les opérations doit travailler sur un horizon temporel minimal qui est l'intervalle de visibilité du jeton concerné. Il faut comparer l'ensemble des séquences possibles sur cet horizon.

Si l'intervalle de visibilité du jeton est $[(\delta_p)_{min^k}; (\delta_p)_{max^k}]$ alors il faut calculer sur cet intervalle les dates possibles d'arrivée d'autres jetons dans les places d'entrée des transitions en conflit structurel avec la transition sensibilisée. Ces jetons sont ceux qui pour arriver dans les places d'entrée des transitions en conflit avant la date $(\delta_p)_{max^k}$ auront à franchir au préalable un certain nombre de transitions.

En fonction des diverses dates d'arrivée, il est alors possible de définir les divers intervalles de visibilité et d'établir les intervalles possibles de conflit.

Un problème se pose si certaines transitions à franchir sont elles-mêmes en conflit avec d'autres transitions et que le calcul des intervalles de visibilité pour résoudre un conflit donné dépend lui même de la résolution d'autres conflits. Ce problème peut apparaître même dans le cadre d'une stratégie de résolution des conflits par ordre de temps croissant. En effet, un état est défini par un marquage et par l'ensemble des intervalles de visibilité de l'ensemble des jetons de ce marquage. Pour chaque transition on a un intervalle de sensibilisation et ce n'est pas parce que la borne inférieure de l'intervalle de sensibilisation d'une transition t_i est inférieure à celle de t_j que t_i sera franchie avant t_j. L'ordre ne sera imposé que si la borne maximale de t_i est inférieure à la borne minimale de t_j.

A partir d'un état donné, la recherche des conflits possibles pour une transition donnée se fera alors d'une part en considérant les jetons qui, pour une stratégie de tir au plus tôt, n'ont à franchir que des transitions n'étant pas en conflit structurel et, d'autre part en considérant les jetons qui, pour une stratégie de tir au plus tard, n'ont à franchir que des transitions en conflit structurel.

Après avoir cherché les possibilités de conflit pour toutes les transitions sensibilisées à partir de l'état considéré, nous prendrons en considération le conflit dont la borne minimale est la plus petite. Cette stratégie nous permettra de découpler les conflits. Elle peut avoir pour conséquence de perdre certaines séquences résultant de la résolution simultanée de plusieurs conflits. Il ne faut toutefois pas oublier que nous

cherchons un compromis entre une politique robuste et une politique efficace. Dans ce cadre nous ne souhaitons nous éloigner d'une politique de franchissement au plus tôt que lorsque la perte d'efficacité est importante. Cette considération justifie notre choix.

Si deux transitions t_1 et t_2 sont en conflit structurel et qu'une des deux transitions est sensibilisée par un jeton se trouvant dans une de ses places p d'entrée à la date $(\delta_p)_{min^k}$ (borne inférieure de l'intervalle de visibilité du jeton qui sensibilise la transition) alors la procédure à appliquer pour établir l'ensemble des intervalles de visibilité qui devront être considérés est la suivante:

Etape 1: Simuler sur l'horizon temporel minimal $[(\delta_p)_{min^k}; (\delta_p)_{max^k}]$ pour les tirs au plus tôt l'évolution des jetons du réseau qui évoluent sur les parties sans choix du réseau i.e. les jetons dont les décisions de tir au plus tôt ne mettent pas en jeu de transitions en conflit structurel, et simuler sur le même horizon l'évolution des jetons dont les décisions de tir au plus tard ne mettent en jeu que des transitions en conflit structurel.

Etape 2: Calculer pour les jetons qui arrivent dans les places d'entrée des transitions en conflit structurel les intervalles de visibilité en fonction des dates d'arrivée des jetons et des intervalles statiques associés à ces places.

A la fin de cette procédure, les divers intervalles de visibilité des jetons sont définis et les intervalles de conflit peuvent être calculés.

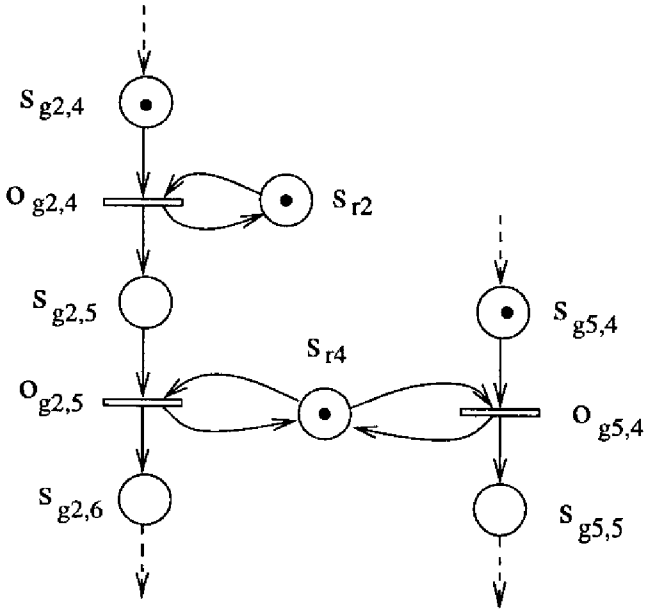

Figure 4.9: Premier Exemple de Calcul des Intervalles de Visibilité

Considérons le réseau de la figure 4.9. A la date $\delta_1 = 7$ un jeton arrive dans la place $s_{g5,4}$ et un autre dans la place $s_{g2,4}$. Les intervalles statiques associés aux places sont:

$$[(d_{s_{g2,4}})_{min^2}; (d_{s_{g2,4}})_{max^2}] = [0,0]$$
$$[(d_{s_{g2,5}})_{min^2}; (d_{s_{g2,5}})_{max^2}] = [0,4]$$
$$[(d_{s_{g5,4}})_{min^1}; (d_{s_{g5,4}})_{max^1}] = [0,5]$$

Les jetons ressource étaient déjà dans les places ressources avant la date $\delta_1 = 7$ et les intervalles de visibilité associés sont:

$$[(\delta_{s_{r2}})_{min^2}; (\delta_{s_{r2}})_{max^2}] = [0,10]$$
$$[(\delta_{s_{r4}})_{min^2}; (\delta_{s_{r4}})_{max^2}] = [0,12]$$

On a comme durée d'opération attachée à la transition $o_{g2,4}$:

$$d_{o_{g2,4}} = 3$$

Les transitions $o_{g2,5}$ et $o_{g5,4}$ sont en conflit structurel. Les places d'entrée de ces transitions sont $s_{g2,5}$, s_{r4} et $s_{g5,4}$. On a déjà l'intervalle de visibilité du jeton ressource se trouvant dans s_{r4}. L'intervalle de visibilité du jeton se trouvant dans $s_{g5,4}$ est:

$$[(\delta_{s_{g5,4}})_{min^1}; (\delta_{s_{g5,4}})_{max^1}] = [0+7, 5+7] = [7,12]$$

L'intervalle de visibilité du jeton se trouvant dans $s_{g2,4}$ est:

$$[(\delta_{s_{g2,4}})_{min^2}; (\delta_{s_{g2,4}})_{max^2}] = [0+7, 0+7] = [7,7]$$

Si nous franchissons la transition $o_{g2,4}$ à la date $\delta_1 = 7$, le jeton se trouvant dans la place $s_{g2,4}$ arrivera dans la place $s_{g2,5}$ à la date:

$$\delta_2 = \delta_1 + d_{o_{g2,4}} = 7 + 3 = 10$$

et l'intervalle de visibilité de ce jeton dans la place $s_{g2,5}$ sera:

$$[(\delta_{s_{g2,5}})_{min^2}; (\delta_{s_{g2,5}})_{max^2}] = [0+10, 4+10] = [10,14]$$

L'intervalle de sensibilisation de $o_{g2,5}$ est $[10,12]$ et celui de $o_{g5,4}$ est $[7,12]$. L'intervalle de temps de conflit associé au couple $(o_{g2,5}, o_{g5,4})$ est alors $[10,12]$. Cet intervalle n'est pas nul et nous ne pourrons pas ignorer ce conflit vis-à-vis du joueur.

Considérons maintenant le réseau de la figure 4.10. A la date $\delta_1 = 7$ un jeton arrive dans la place $s_{g5,4}$ et un autre dans la place $s_{g2,4}$. Les intervalles statiques associés aux places sont:

$$[(d_{s_{g2,4}})_{min^2}; (d_{s_{g2,4}})_{max^2}] = [0,4]$$
$$[(d_{s_{g2,5}})_{min^2}; (d_{s_{g2,5}})_{max^2}] = [0,4]$$
$$[(d_{s_{g5,4}})_{min^1}; (d_{s_{g5,4}})_{max^1}] = [0,5]$$

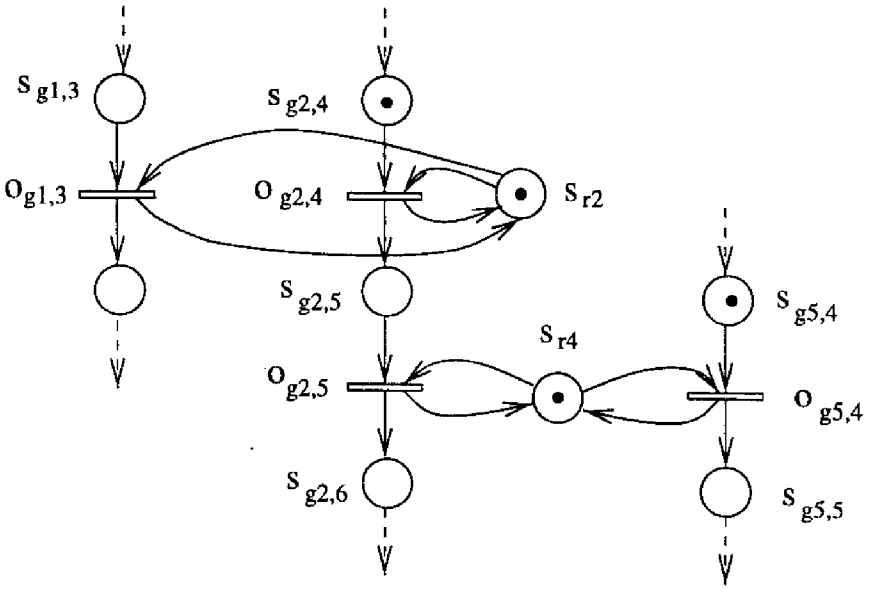

Figure 4.10: Deuxième Exemple de Calcul des Intervalles de Visibilité

Les intervalles de visibilité des jetons ressource sont les même que dans l'exemple précédent. On a comme durée d'opération $d_{o_{g2,4}} = 4$. L'intervalle de visibilité du jeton de $s_{g5,4}$ est le même que dans l'exemple précédent. L'intervalle de visibilité du jeton de $s_{g2,4}$ est:

$$[(\delta_{s_{g2,4}})_{min^2}; (\delta_{s_{g2,4}})_{max^2}] = [0+7, 4+7] = [7, 11]$$

et l'intervalle de sensibilité associé à $o_{g2,4}$ est donc: [7; 10].

Nous ne sommes plus sûr de la date d'arrivée du jeton dans la place $s_{g2,5}$. En effet, la transition $o_{g2,4}$ est en conflit structurel avec $o_{g1,3}$. Nous ne savons pas a priori s'il peut y avoir sur l'horizon temporel minimal considéré l'apparition d'un intervalle de temps de conflit associé au couple $(o_{g1,3}, o_{g2,4})$. Par conséquent, la date d'arrivée du jeton dans $s_{g2,5}$ est calculée pour le tir au plus tard de $o_{g2,4}$. Si nous franchissons la transition $o_{g2,4}$ à la date $\delta_2 = 10$, le jeton se trouvant dans la place $s_{g2,4}$ arrivera dans la place $s_{g2,5}$ à la date:

$$\delta_3 = \delta_2 + d_{o_{g2,4}} = 10 + 4 = 14$$

et l'intervalle de visibilité de ce jeton dans la place $s_{g2,5}$ sera:

$$[(\delta_{s_{g2,5}})_{min^2}; (\delta_{s_{g2,5}})_{max^2}] = [0+14, 4+14] = [14, 18]$$

Comme l'intervalle de visibilité de s_{r4} est [0; 12], à ce niveau de la simulation du réseau nous ne pouvons pas mettre en évidence d'intervalle de sensibilisation pour $o_{g2,5}$. Sur l'horizon temporel minimal, il n'existe donc pas d'intervalle de temps de conflit associé au couple $(o_{g2,5}, o_{g5,4})$ et nous pouvons débuter le tir de $o_{g5,4}$ à la date $\delta_1 = 7$.

Nous pourrions essayer de traiter simultanément les conflits possibles associés aux couples de transitions $(o_{g1,3}, o_{g2,4})$ et $(o_{g2,5}, o_{g5,4})$, mais, comme nous l'avons déjà dit, la philosophie de cette approche est de trouver une solution admissible respectant un ensemble de contraintes et non pas la recherche de la solution optimale qui mettrait en jeu au préalable l'énumération de l'ensemble des solutions admissibles. Ceci aurait pour conséquence une explosion des états admissibles et la solution optimale ne serait pas forcément exploitable de toute façon par la suite au niveau du pilotage temps réel de la cellule.

4.4.2 Stratégie de Résolution des Conflits

Rappelons que notre objectif n'est pas la recherche d'une séquence qui serait optimale vis-à-vis du temps d'exécution, mais plutôt la recherche d'une séquence admissible cohérente avec l'ensemble des contraintes temporelles qui ont été définies. La séquence obtenue devra être robuste aux petites perturbations (comme le retard du début d'une opération par exemple) et devra pouvoir être calculée en temps réel au niveau du pilotage de la cellule. Elle ne devra donc pas trop s'éloigner d'un fonctionnement correspondant au franchissement au plus tôt des transitions. L'heuristique utilisée dans la recherche de la séquence doit donc être simple sans pour autant travailler sur les seuls marquages courants. En particulier, le système de décision associé au joueur de réseau de Petri chargé d'ordonner les opérations doit travailler sur un horizon temporel minimal défini à partir des intervalles de visibilité des pièces se trouvant dans les places d'entrée des transitions en conflit. Il doit comparer toutes les séquences possibles sur cet horizon et prendre la meilleure décision suivant un ensemble de critères qui sont:

- respect des contraintes sur cet horizon,
- maximisation des marges temporelles restantes.

En général, les décisions possibles ne peuvent préserver certaines marges que par la consommation d'autres marges. Les choix entre les diverses politiques doivent donc être basés sur des heuristiques.

Supposons que l'on vienne de franchir une transition et que, après le calcul des intervalles de visibilité et des intervalles de conflit présentés au paragraphe précédent, on se trouve dans la situation suivante: deux transitions t_1 et t_2 sont en conflit (i.e. l'intervalle de conflit associé au couple (t_1, t_2) n'est pas nul). La procédure à appliquer pour résoudre le conflit et indiquer la première transition qui doit être tirée est alors la suivante:

Etape 1: $i \leftarrow 1$, $j \leftarrow 2$

Etape 2: Si l'intervalle de sensibilisation de la transition t_i est $[(\theta_{t_i})_{min}, (\theta_{t_i})_{max}]$ alors nous pouvons tirer t_i au plus tôt à la date $(\theta_{t_i})_{min}$. La date de fin de tir de t_i est alors: $(\theta_{t_i})_f = (\theta_{t_i})_{min} + d_{t_i}$.

Etape 3: Si $(\theta_{t_i})_f \in [(\delta_{p_j})_{min}; (\delta_{p_j})_{max}]$ (la date de fin de tir de t_i appartient à l'intervalle de visibilité du jeton de la place p_j qui est la place d'entrée de t_j dans laquelle se trouve une pièce en attente de subir l'opération associée à la transition t_j) alors le tir au plus tôt de t_i est solution du problème.

Etape 4: Si $i = 1$ alors $i \leftarrow 2$, $j \leftarrow 1$ et retour à l'étape 2. Si $i = 2$ alors aller à l'étape 5.

Etape 5: S'il existe plusieurs solutions au problème, choisir la solution qui maximise la flexibilité de la cellule (i.e. celle qui laisse le plus de marge sur les places d'entrée des transitions en conflits). S'il n'y a pas de solution au problème alors on a échec de la procédure.

Cette procédure peut être généralisée au cas de plus de deux transitions en conflit. Nous devrons alors calculer le tir au plus tôt de chacune des transitions, vérifier qu'il n'empêche pas le tir des autres transitions et choisir la solution qui maximise la flexibilité. La notion de flexibilité n'étant pas définie de façon formelle, l'opérateur devra définir suivant le cas le critère le plus approprié à la cellule considérée (i.e. celui qui permettra la meilleure prise en compte des possibles perturbations).

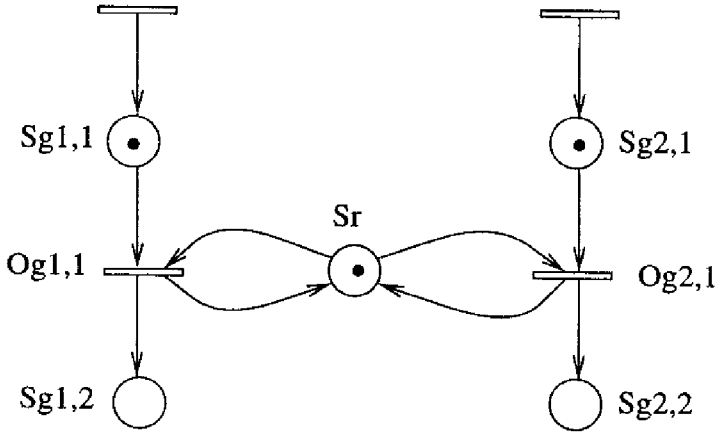

Figure 4.11: Résolution de Conflit

Considérons l'exemple de la figure 4.11. Les intervalles de visibilité sont:
$[(\delta_{s_{g1,1}})_{min}; (\delta_{s_{g1,1}})_{max}] = [1, 6]$,
$[(\delta_{s_r})_{min}; (\delta_{s_r})_{max}] = [0, 10]$,
$[(\delta_{s_{g2,1}})_{min}; (\delta_{s_{g2,1}})_{max}] = [4, 8]$.
Les intervalles de sensibilisation sont $[1, 6]$ pour $o_{g1,1}$ et $[4, 8]$ pour $o_{g2,1}$. L'intervalle de conflit associé au couple $(o_{g1,1}, o_{g2,1})$ est donc $[4, 6]$.

Supposons que les durées des opérations soient $d_{o_{g1,1}} = 5$ et $d_{o_{g2,1}} = 1$.

Si $o_{g1,1}$ est la première transition à être tirée alors la date de fin de tir de $o_{g1,1}$ est $(\theta_{o_{g1,1}})_f = 1 + 5 = 6$. $(\theta_{o_{g1,1}})_f \in [4, 8]$ qui est l'intervalle de visibilité du jeton se trouvant dans la place $s_{g2,1}$. Le tir au plus tôt de $o_{g1,1}$ est donc solution du problème. En appliquant une telle stratégie, nous laissons une marge de 2 pour le tir de $o_{g2,1}$.

Si $o_{g2,1}$ est la première transition à être tirée alors la date de fin de tir de $o_{g2,1}$ est $(\theta_{o_{g2,1}})_f = 4 + 1 = 5$. $(\theta_{o_{g2,1}})_f \in [1,6]$ qui est l'intervalle de visibilité du jeton se trouvant dans la place $s_{g1,1}$. Le tir au plus tôt de $o_{g2,1}$ est donc solution du problème. En appliquant cette stratégie, nous laissons une marge de 1 pour le tir de $o_{g1,1}$.

La première solution (tir de $o_{g1,1}$ avant tir de $o_{g2,1}$) est donc un peu plus robuste. Nous la choisirons comme solution retenue. Néanmoins, l'autre solution ne doit pas être écartée. C'est une solution acceptable qui pourra être reprise si la séquence $o_{g1,1}|o_{g2,1}$ est la cause d'une violation de contrainte qui peut apparaître plus tard dans l'exécution du joueur. Nous devons en effet considérer que du fait du très grand nombre de contraintes simultanées à prendre en compte, il sera de toute façon difficile de définir un critère de choix bien clair, et dans ce cas, même si une solution doit être choisie, les autres ne doivent pas être éliminées pour autant et doivent être gardées en réserve pour éventuellement être utilisées ultérieurement.

Supposons maintenant que les durées des opérations soient $d_{o_{g1,1}} = 4$ et $d_{o_{g2,1}} = 3$.

Si $o_{g1,1}$ est la première transition à être tirée alors la date de fin de tir de $o_{g1,1}$ est $(\theta_{o_{g1,1}})_f = 1 + 4 = 5$. $(\theta_{o_{g1,1}})_f \in [4,8]$ qui est l'intervalle de visibilité du jeton se trouvant dans la place $s_{g2,1}$. Le tir au plus tôt de $o_{g1,1}$ est donc solution du problème. En appliquant cette stratégie, nous laissons une marge de 3 pour le tir de $o_{g2,1}$.

Si $o_{g2,1}$ est la première transition à être tirée alors la date de fin de tir de $o_{g2,1}$ est $(\theta_{o_{g2,1}})_f = 4 + 3 = 7$. $(\theta_{o_{g2,1}})_f \notin [1,6]$ qui est l'intervalle de visibilité du jeton se trouvant dans la place $s_{g1,1}$. Le tir au plus tôt de $o_{g2,1}$ n'est donc pas solution du problème. Par conséquent nous devons de toute façon franchir $o_{g1,1}$ avant $o_{g2,1}$ sinon il y aura violation de contrainte.

Pour finir, supposons que l'on ait toujours les durées d'opération $d_{o_{g1,1}} = 4$ et $d_{o_{g2,1}} = 3$, mais que cette fois il y ait deux jetons ressources dans la place s_r dont les intervalles de visibilité sont $[0,10]$ et $[3,14]$. Nous avons deux possibilités d'intervalle de sensibilité pour $o_{g1,1}$ qui sont $[1,6]$ et $[3,6]$ selon la ressource choisie, et la même possibilité $[4,8]$ pour $o_{g2,1}$ quelle que soit la ressource choisie.

Si on tire $o_{g2,1}$ au plus tôt alors la date de fin de tir de $o_{g2,1}$ est $(\theta_{o_{g2,1}})_f = 4+3 = 7$. $(\theta_{o_{g2,1}})_f \notin [1,6]$ et nous ne pouvons pas utiliser la même ressource pour le tir de $o_{g1,1}$. Comme l'intervalle de sensibilisation de $o_{g2,1}$ est le même quelle que soit la ressource considérée, nous choisissons pour sensibiliser $o_{g1,1}$ la ressource dont l'intervalle de visibilité associé est $[0;10]$ et nous pouvons débuter le tir de $o_{g1,1}$ à la date 1 dès que le jeton en $s_{g1,1}$ devient disponible. Si nous avions choisi l'autre ressource dont l'intervalle de visibilité associé est $[3;14]$ alors le tir de $o_{g1,1}$ n'aurait pu débuter qu'à la date 3 et la consommation de marge aurait été supérieure.

L'introduction d'une seconde ressource a donc permis de relacher les contraintes temporelles et a ainsi rendu possible le tir au plus tôt des deux transitions. Dans ce cas, puisque les deux ressources sensibilisent les transitions $o_{g1,1}$ et $o_{g2,1}$ sur des intervalles de sensibilité différents mais non nuls, il est normal que le conflit disparaisse. Il subsiste néanmoins la stratégie d'ordonnancement à prendre en compte et sa répercussion sur la performance globale du système.

4.5 Joueur de Réseau de Petri p-temporel t-temporisé

Nous allons maintenant présenter l'algorithme du joueur de réseau de Petri p-temporel t-temporisé qui est utilisé pour la recherche d'une séquence d'opérations admissible compte tenu de l'ensemble des contraintes à respecter.

L'algorithme possède une procédure de décision qui doit être utilisée chaque fois qu'un conflit apparaît i.e. qu'un intervalle de temps de conflit non nul est associé à un couple de transitions.

Il possède aussi un mécanisme de retour arrière ("backtracking"). Il ne peut donc être utilisé qu'au niveau d'un ordonnancement prévisionnel et non pour la supervision et le pilotage en temps réel.

Précisons enfin qu'il utilise un échéancier qui contient une suite d'événements ordonnés dans le temps. Ces événements sont d'une part les bornes minimales et maximales des intervalles de visibilité des jetons non réservés et d'autre part les dates de fin de franchissement des transitions associées aux jetons réservés.

L'état courant est caractérisé par la date courante, la liste des jetons qui sont dans l'état non réservé avec leur intervalle de visibilité associé et la liste des transitions en cours de franchissement avec les jetons associés et la date de fin de franchissement.

4.5.1 Algorithme

Etape 1:

1.1 Mémoriser le marquage initial.

1.2 Pour l'origine des temps $\theta \leftarrow 0$, calculer les intervalles de visibilité de chaque jeton.

1.3 Insérer dans l'échéancier les bornes minimales et maximales de ces intervalles.

1.4 Mettre tous les jetons dans l'état indisponible.

1.5 Mettre à zéro la marge temporelle d_{C_k} associée à chacune des étiquettes Kanban.

1.6 Mettre à zéro la marge temporelle agrégée totale D_{C_r} associée à chacune des contraintes de ressources C_r.

1.7 Mémoriser l'état initial du réseau dans la pile.

Etape 2:

2.1 Considérer le premier événement de l'échéancier.

2.2 Calculer la différence δ entre l'instant courant et l'instant précédent.

2.3 Calculer la date courante $\theta \leftarrow \theta + \delta$.

2.4. Appeler la procédure d'actualisation des valeurs des marges temporelles.

2.5 Mémoriser l'état du réseau dans la pile.

Etape 3: Si le premier événement de l'échéancier correspond à la date de fin de tir d'une transition:

3.1 Franchir la transition.

3.2 Appeler la procédure de fin de cycle.

3.3 Calculer les nouveaux intervalles de visibilité des jetons utilisés dans le tir de la transition.

3.4 Insérer dans l'échéancier les nouvelles bornes minimales et maximales de ces intervalles.

3.5 Mettre les jetons utilisés dans le tir de la transition dans l'état *indisponible.*

3.6 Mémoriser l'état du réseau dans la pile.

3.7 Aller à l'étape 2.

Etape 4: Si le premier événement de l'échéancier correspond à la borne minimale d'un intervalle de visibilité:

4.1 Mettre le jeton auquel est associé cet intervalle dans l'état disponible.

4.2 Si une transition différente de la transition de synchronisation t_s est sensibilisée par les jetons *disponibles*:

4.2.1 Si la transition sensibilisée n'est pas en conflit structurel:

4.2.1.1 Mettre les jetons qui la sensibilisent dans l'état *réservé* et enlever de l'échéancier les bornes maximales des intervalles de visibilité associés à ces jetons.

4.2.1.2 Insérer dans l'échéancier la date de fin de tir de la transition.

4.2.1.3 Mémoriser l'état du réseau dans la pile.

4.2.2 Si la transition sensibilisée est en conflit structurel:

4.2.2.1 Appeler la procédure de résolution de conflit.

4.2.2.2 Mémoriser l'état du réseau et indiquer la solution choisie par la procédure dans la pile.

4.2.2.3 Si la transition sensibilisée est solution de la procédure:

4.2.2.2.1 Mettre les jetons qui la sensibilisent dans l'état *réservé* et enlever de l'échéancier les bornes maximales des intervalles de visibilité associés à ces jetons.

4.2.2.2.2 Insérer dans l'échéancier la date de fin de tir de la transition.

4.2.2.2.3 Mémoriser l'état du réseau dans la pile.

4.2.3 Aller à l'étape 4.2

4.3 Aller à l'étape 2.

Etape 5: Si le premier événement de l'échéancier correspond à la borne maximale d'un intervalle de visibilité:

5.1 Envoyer un signal d'alarme à l'opérateur pour indiquer qu'il y a violation de contrainte.

5.2 Appeler la procédure de diagnostic.

Précisons que la procédure de résolution de conflit ne propose, vis à vis de l'algorithme du joueur, que deux solutions admissibles: ou on franchit la transition sensi-

bilisée immédiatement, ou on ne franchit pas la transition et on laisse les jetons qui la sensibilisent dans l'état disponible (la ou les ressources nécessaires au tir de cette transition restent dans l'état disponible pour le tir d'une autre transition ultérieurement).

4.5.2 Procédure de Fin de Cycle

C'est cette procédure qui doit tester après chaque événement si le réseau a atteint l'état initial ou en d'autres termes l'état de fin de cycle. Il suffit pour cela que chacune des étiquettes Kanban se trouve dans sa place de début de cycle et que l'intervalle statique associé à cette place soit un intervalle de fin de cycle c'est-à-dire un intervalle qui force le jeton à rester dans l'état indisponible jusqu'à ce que la fin de cycle soit détectée. Lorsque l'ensemble des jetons Kanban sont tous revenus dans leur place de début de cycle et qu'ils sont en attente dans l'état indisponible, alors on franchit la transition de synchronisation t_s de la contrainte cyclique C_c et on va à l'étape 1.

Il se peut pour des raisons d'efficacité de la production que l'état de début de cycle du réseau de Petri comprenne des opérations en cours c'est-à-dire des jetons réservés depuis une certaine durée. L'état de début de cycle ne sera alors plus défini par le seul marquage réseau. Si tel est le cas, il est nécessaire d'apporter certaines modifications à l'algorithme du joueur. L'étape 1 de l'algorithme devient:

1.1 Initialiser l'origine des temps $\theta \leftarrow 0$.

1.2 Mémoriser le marquage initial et l'état de chaque jeton (indisponible ou disponible ou réservé).

1.3 Insérer dans l'échéancier les événements correpondant aux bornes minimales et maximales des intervalles de visibilité des jetons indisponibles.

1.4 Insérer dans l'échéancier les événements correspondant aux bornes maximales des intervalles de visibilité des jetons disponibles.

1.5 Insérer dans l'échéancier les événements correspondant aux dates de fin de tir des transitions associées aux jetons réservés.

1.5 Mettre à zéro la marge temporelle d_{C_k} associée à chacune des étiquettes Kanban.

1.6 Mettre à zéro la marge temporelle agrégée moyenne D_{C_r} associée à chacune des contraintes de ressources C_r.

1.7 Mémoriser l'état initial du réseau dans la pile.

Comme le test de fin de cycle n'est vérifié qu'après chaque événement (i.e. lorsqu'un événement de l'échéancier devient vrai), nous devons introduire dans l'échéancier des événements de fin de cycle associés à des jetons qui se trouvent dans l'état réservé.

Considérons par exemple la partie de réseau représentée par la figure 4.12. Supposons que le jeton de $s_{g2,4}$ soit une étiquette Kanban d'une contrainte C_{k2}. En début de cycle ce jeton se trouve dans la place $s_{g2,4}$ dans l'état réservé depuis une durée de

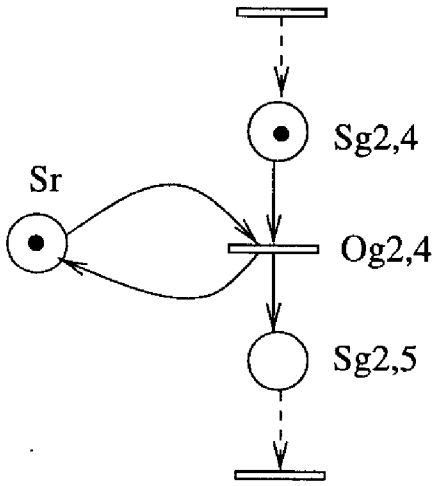

Figure 4.12: Evénement de fin de cycle

3. On a $d_{og2,4} = 7$. Pour pouvoir tester cet état au niveau du joueur, on doit introduire dans l'échéancier un événement de fin de cycle qui nous indique la réservation du jeton depuis une durée de 3. Après l'arrivée du jeton dans la place $s_{g2,4}$, pour le dernier passage de ce jeton dans cette place par rapport au cycle (c'est-à-dire que c'est le dernier intervalle statique du cycle qui est associé à cette place vis à vis de ce jeton), lorsqu'à la date δ le jeton passe dans l'état réservé, alors on introduit dans l'échéancier la date $\delta+3$ de fin de cycle associée à ce jeton. Nous pourrons ainsi savoir quand ce jeton aura atteint son état de fin de cycle. Il faudra rajouter entre l'étape 2 et 3 de l'algorithme l'étape suivante:

Si le premier événement de l'échéancier correspond à la date de fin de cycle associée à un jeton réservé alors appeler la procédure de fin de cycle.

Soulignons enfin que pour chaque contrainte de synchronisation considérée (contraintes Kanban et de ressource), nous aurons pour chaque jeton réservé dans l'état initial un intervalle statique de moins à définir puisqu'en début de cycle et en fin de cycle le jeton sera dans l'état réservé et non dans l'état indisponible.

4.5.3 Procédure de Diagnostic

La procédure de diagnostic lance un programme qui identifie le type de contrainte violée et qui propose plusieurs alternatives qui sont:

- S'il n'y a pas de contre ordre de la part de l'opérateur, on fait un retour arrière ("backtracking") jusqu'au dernier état du réseau qui a mis en jeu la procédure de résolution de conflit pour lequel plus d'un choix était possible (i.e. on fait un

retour arrière jusqu'au dernier conflit ressource dont la résolution a été fondée sur l'utilisation d'une heuristique), et on choisit une autre solution.

- On fait une relaxation de certaines contraintes et on repart dans un état choisi par l'opérateur.
- On fait une réinitialisation de l'algorithme avec un marquage initial différent.

4.5.4 Procédure d'Actualisation des Marges Temporelles

Etape 1:

1.1 Pour chacune des étiquettes Kanban qui se trouvent dans l'état non réservé, on calcule la nouvelle valeur de la marge temporelle associée qui est:

$d_{C_k} \leftarrow d_{C_k} + \delta$.

1.2 Si une au moins des nouvelles durées résiduelles $((d_{C_k})_{max} - d_{C_k})$ est à valeur négative:

1.2.1 Envoyer un signal d'alarme à l'opérateur pour indiquer que l'on a une violation de contrainte.

1.2.2 Appeler la procédure de diagnostic.

Etape 2:

2.1 Pour les marges temporelles agrégées D_{C_r} associées aux contraintes de ressources C_r, η_r étant le nombre de jetons non réservés par invariant de places C_r, les nouvelles valeurs des marges temporelles sont:

$D_{C_r} \leftarrow D_{C_r} + \eta_r.\delta$.

2.2 Si une au moins des nouvelles durées résiduelles $(m_{s_r}.(d_{C_r})_{max} - D_{C_r})$ est à valeur négative:

2.2.1 Envoyer un signal d'alarme à l'opérateur pour indiquer que l'on a une violation de contrainte

2.2.2 Appeler la procédure de diagnostic.

Le mécanisme de retour arrière suit le même modèle de fonctionnement utilisé en Prolog par exemple. A chaque pas de l'algorithme, on mémorise l'état du réseau dans une pile et lorsque l'on a une violation de contrainte, on remonte dans la pile jusqu'au dernier état du réseau qui met en jeu un conflit de ressource caractérisé par l'appel de la procédure de résolution de conflit. Bien sûr, tous les états postérieurs à cet état doivent être effacés dans la pile.

Pour le diagnostic, il peut y avoir intervention de l'opérateur. En effet, c'est ce dernier qui a la responsibilité du choix des contraintes temporelles statiques associées au

modèle p-temporel t-temporisé (les marges temporelles associées à chacun des invariants de places et les intervalles statiques associés à chacune des places du réseau) et du choix de l'état initial du réseau. Il devra donc être capable de suivre l'évolution du joueur et de diagnostiquer la raison de certaines violations de contraintes. Rappelons que la ou les solutions obtenues seront par la suite exploitées au niveau du pilotage temps réel de la cellule et devront donc d'une part répondre aux exigences de la production et d'autre part être suffisamment robustes pour prendre en compte les aléas de la production. L'opérateur doit donc avoir une totale liberté d'intervention lors de l'évolution du joueur soit pour relaxer certaines contraintes statiques trop rigides, soit pour relancer l'algorithme à partir d'un état initial différent qui peut conduire à une meilleure solution. Bien sûr, s'il n'y a pas d'intervention de la part de l'opérateur durant le diagnostic, on applique automatiquement la technique de retour arrière.

A la fin de l'algorithme, nous obtenons comme solution une séquence d'opérations admissible que nous essaierons de suivre au niveau du pilotage temps réel. Le pilotage ou ordonnancement réactif va être fait à partir d'un algorithme de joueur de réseau de Petri qui va fonctionner suivant le même principe que celui qui vient d'être présenté. La différence fondamentale de son fonctionnement va résider dans le fait qu'il n'aura pas de mécanisme de retour arrière puisque les décisions devront être prises en temps réel. Pour que la solution obtenue par le joueur p-temporel t-temporisé puisse donc être exploitée au niveau du pilotage, il faut que celle-ci ne comporte pas de retour arrière. Il est donc important pour l'opérateur à chaque fois qu'il y a un retour à un état antérieur de resserrer certaines contraintes afin que les mauvais choix lors de l'apparition de conflits soient rendus impossibles.

Considérons toujours l'exemple de la figure 4.11 dans le cas où:
$d_{o_{g1,1}} = 5$ et $d_{o_{g2,1}} = 1$.
Les intervalles de visibilité des jetons sont:
$[(\delta_{s_{g1,1}})_{min}; (\delta_{s_{g1,1}})_{max}] = [1, 6]$,
$[(\delta_{s_r})_{min}; (\delta_{s_r})_{max}] = [0, 10]$,
$[(\delta_{s_{g2,1}})_{min}; (\delta_{s_{g2,1}})_{max}] = [4, 8]$.

Les intervalles de sensibilisation sont $[1, 6]$ pour $o_{g1,1}$ et $[4, 8]$ pour $o_{g2,1}$ et l'intervalle de conflit associé au couple $(o_{g1,1}, o_{g2,1})$ est $[4, 6]$. Nous avons vu qu'en appliquant la procédure de résolution de conflit, la solution peut être aussi bien la séquence $o_{g1,1}|, o_{g2,1}$ que la séquence $o_{g2,1}|, o_{g1,1}$. Si la solution $o_{g1,1}|o_{g2,1}$ est choisie, qu'il y a par la suite violation de contrainte, et que le retour arrière nous ramène à ce même conflit, alors nous devrons éliminer la première solution est choisir la solution $o_{g2,1}|o_{g1,1}$. Si nous ne modifions pas les contraintes, l'algorithme du joueur de réseau de Petri qui sera utilisé au niveau du pilotage de la cellule pourra à nouveau choisir la solution $o_{g1,1}|o_{g2,1}$ et ne pourra plus par la suite faire de retour arrière. L'opérateur doit donc tirer les conséquences de ce retour arrière et modifier les contraintes en conséquence. Nous pouvons modifier par exemple l'intervalle de visibilité du jeton se trouvant dans $s_{g1,1}$ en agissant simplement sur les valeurs de l'intervalle statique associé à $s_{g1,1}$ à partir duquel a été calculé l'intervalle de visibilité associé au jeton de cette place. Si le nouvel intervalle est $[(\delta_{s_{g1,1}})_{min}; (\delta_{s_{g1,1}})_{max}] = [4, 6]$, l'intervalle de sensibilisation de $o_{g1,1}$ n'est plus $[1, 6]$ mais $[4, 6]$, l'intervalle de sensibilisation de $o_{g2,1}$ reste le même et l'intervalle de temps de conflit est toujours $[4, 6]$. Lorsque nous appliquons la pro-

cédure de résolution de conflit, nous avons pour la séquence $o_{g1,1}|o_{g2,1}$ comme date de fin de tir de $o_{g1,1}$: $(\theta_{o_{g1,1}})_f = 4 + 5 = 9$. $(\theta_{o_{g1,1}})_f$ n'appartient plus à l'intervalle de visibilité du jeton de $s_{g2,1}$ et la séquence $o_{g1,1}|o_{g2,1}$ ne peut donc plus être retenue par la procédure de résolution de conflit. Après cette modification, si nous relançons l'algorithme du joueur, nous n'aurons plus de retour arrière directement provoquer par la résolution de ce conflit et la solution donnée par le joueur sera directement exploitable pour le pilotage de la cellule.

Par rapport à ce qui a été dit au début de ce chapitre sur l'intérêt d'une approche hybride qui combine les réseaux de Petri et les techniques d'Intelligence Artificielle, nous devons souligner que le principe de résolution de l'algorithme du joueur de réseau de Petri p-temporel t-temporisé peut être situé dans le contexte des techniques de filtrage de l'approche *CSP* (Constraint Satisfaction Problems) [FA 94] en terme de cohérence locale dans la mesure où si une des marges temporelles associée aux invariants de places du modèle est violée (durée résiduelle à valeur négative) au niveau des noeuds (conflit ressource) de l'arbre d'exploration des solutions, alors il ne sera pas nécessaire de poursuivre la recherche plus loin dans cette direction. Ceci vient du fait que de façon implicite, en nous basant sur des contraintes locales, on cherche en réalité une séquence pour laquelle la période vérifie $\pi_{min} \leq \pi \leq \pi_{max}$ en sachant que π prend en compte l'aspect global et non local du système et qu'il faudrait poursuivre la recherche beaucoup plus loin dans une même direction avant d'arriver à une incohérence sur π de façon explicite.

Le mécanisme de retour arrière ("backtracking") qui permet de revenir à un état précédent du réseau à chaque fois qu'apparaît une incohérence dans les contraintes a pour objectif de reprendre la recherche dans une autre direction. Ce principe de résolution est une technique classique de l'Intelligence Artificielle qui est utilisée par exemple dans le moteur d'inférence de Prolog.

Enfin, la possibilité de suivi et d'intervention de l'opérateur durant l'évolution de l'algorithme est un moyen d'analyser les conséquences des choix qui sont fait sur les variables de décision telles que les marges temporelles, les intervalles statiques ou les conditions initiales du système. Un algorithme plus classique issu de la recherche opérationnelle utilisé pour la recherche d'une séquence optimale ne permettrait pas à l'opérateur en charge du bon fonctionnement de la cellule de savoir sur quel paramètre agir pour modifier les performances.

4.6 Pilotage Temps Réel de la Cellule

4.6.1 Principe de Fonctionnement

Le modèle utilisé pour le pilotage temps réel de la cellule doit pouvoir décrire en temps réel l'enchaînement des traitements à effectuer sur les pièces, communiquer avec l'environnement extérieur et être soumis à des contraintes temporelles explicites. Il pourrait être intéressant d'utiliser un modèle de haut niveau comme les réseaux à Objets [SB 94] par exemple afin d'obtenir une description compacte et structurée

qui associe les données au réseau de Petri. Le modèle que nous avons défini dans le chapitre 2 n'est pas de haut niveau, néanmoins, le simple fait d'associer des intervalles de visibilité aux jetons entraîne leur individualisation.

Dans le cas de systèmes qui interagissent avec leur environnement comme c'est le cas des cellules flexibles, on doit faire évoluer le modèle en synchronisme avec le système physique en associant des conditions supplémentaires de franchissement associées aux transitions et des actions associées aux transitions. Ces actions et ces conditions font intervenir des données ou des événements extérieurs comme des capteurs, des actionneurs, des réceptions ou des émissions de messages.

Le principe de la supervision fondée sur un modèle a été brièvement présenté au chapitre 1 (paragraphe 1.4.2). Le bloc "modèle" de la figure 1.1 doit être extrêmement proche du modèle de la cellule que nous avons utilisé dans les étapes de conception et de génération d'un ordonnancement flexible puisque l'un des buts principaux de notre travail est de renforcer la cohérence entre l'ordonnancement prévisionnel et le pilotage réactif. Nous avons associé aux transitions $o_{gi,j}$ des opérations de fabrication qui sont

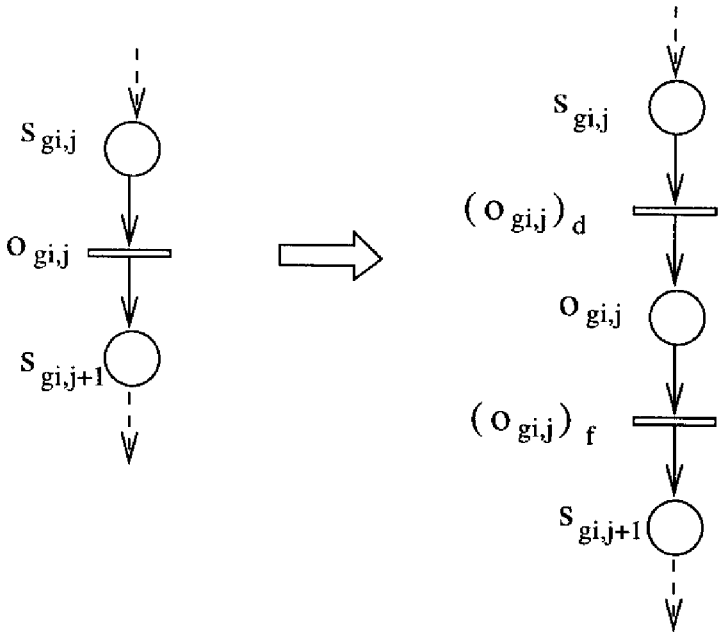

Figure 4.13: Association d'une opération à une transition

indivisibles et ininterruptibles. De telles transitions sont en fait des abréviations pour des séquences élémentaires formées d'une transition de début d'opération $(o_{gi,j})_d$, d'une place $o_{gi,j}$ décrivant l'opération en cours et d'une transition de fin d'opération $(o_{gi,j})_f$ (voir figure 4.13). Cette place supplémentaire $o_{gi,j}$ est une place substituable au sens des règles de réduction et les propriétés du réseau sous-jacent ne sont pas modifiées. Les événements associés aux transitions $(o_{gi,j})_d$ et $(o_{gi,j})_f$ sont de durée nulle et le modèle n'est plus à ce moment p-temporel t-temporisé mais simplement p-temporel. Pour représenter la durée d'opération $d_{o_{gi,j}}$ qui était associée à la transition du modèle p-temporel t-temporisé, nous pouvons associer à la nouvelle place $o_{gi,j}$

l'intervalle statique

$$[d_{o_{gi,j}}; d_{o_{gi,j}}]$$

Comme le pilotage concerne une cellule réelle, nous ne connaîtrons pas avec certitude les durées d'usinage et il pourra être plus judicieux de choisir l'intervalle

$$[d_{o_{gi,j}} - \delta; d_{o_{gi,j}} + \delta]$$

la valeur de δ correspondant à une marge d'erreur existant entre la durée prévue de l'opération et la durée réelle.

Comme nous n'avons plus, après transformation du modèle, de t-temporisations associées aux transitions du modèle, le réseau est simplement p-temporel et les jetons sont soit dans l'état indisponible, soit dans l'état disponible conformément à la définition des réseaux p-temporels qui est proposée dans [KH 96].

Soulignons enfin qu'au niveau des places stocks intermédiaires, nous conserverons les intervalles statiques qui auront été choisis à la fin du calcul de l'ordonnancement prévisionnel du réseau p-temporel t-temporisé.

Le superviseur doit également considérer les marges temporelles maximales associées à chaque invariant de places du réseau puisqu'elles ne sont pas directement comprises dans la définition du réseau p-temporel. Nous devons aussi prendre en compte les informations du monde extérieur, c'est-à-dire les messages envoyés aux actionneurs de la cellule qui sont associés aux transitions de début d'opération, et les messages reçus des capteurs de la cellule qui sont associés aux transitions de fin d'opération. La figure 4.14 présente un exemple de communication entre le modèle qui représente la cellule et la cellule physique.

Comme le pilotage de la cellule est réalisé en temps réel, le superviseur doit pouvoir utiliser la mesure du temps donnée par l'horloge temps réel et il doit réaliser des calculs sur l'ensemble des données afin d'établir les conditions et les traitements ou actions qui seront associées aux transitions et aux places du réseau. Il y a en particulier la vérification de l'ensemble des contraintes temporelles (marges temporelles et intervalles de visibilité des jetons) avec l'envoi d'un signal d'alarme à l'opérateur en cas d'une violation de contrainte. Nous avons aussi les envois de messages aux actionneurs pour démarrer les opérations de la cellule et les réceptions de messages des capteurs pour indiquer les fins d'opération au modèle de la cellule.

Nous allons voir maintenant comment est réalisé le pilotage temps réel de la cellule à l'aide d'un joueur de réseau de Petri p-temporel qui spécifie les liens entre le modèle de la cellule, l'ensemble des données et le système physique.

4.6.2 Joueur Temps Réel

Le joueur temps réel appliqué au réseau p-temporel permet à la fois de synchroniser le marquage du réseau avec l'état de la cellule et également de déduire les commandes à effectuer. En particulier, il doit:

- détecter les instants où se produisent les événements externes (fins d'opération) en décodant les messages reçus de la cellule,

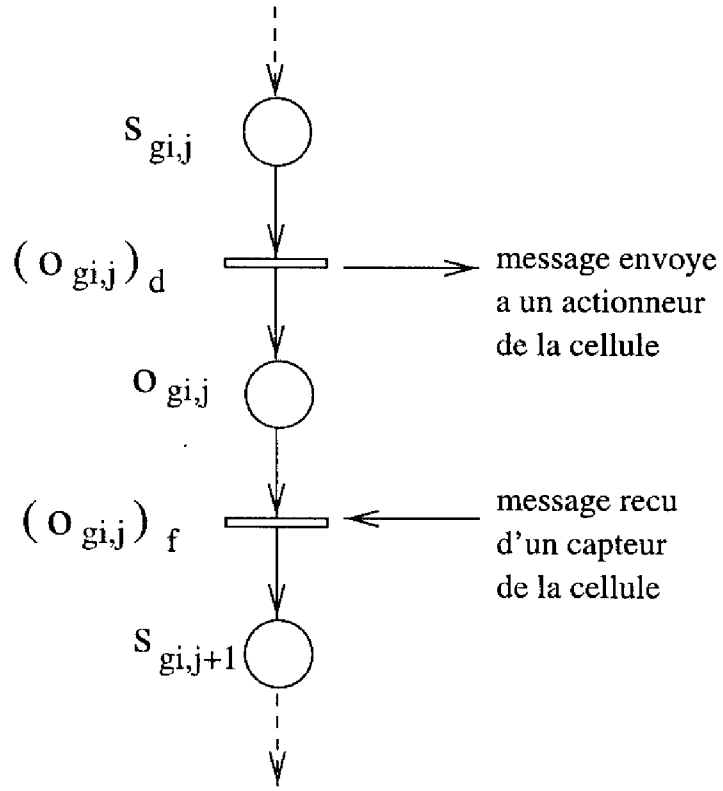

Figure 4.14: Communication entre le système modélisé et le monde extérieur

- déterminer quand les transitions correspondant aux débuts d'opération doivent être franchies et envoyer les commandes correspondantes,
- détecter les instants où les jetons deviennent disponibles,
- détecter les instants où le système entre dans un fonctionnement anormal (i.e. lorsque se produit une violation de contrainte).

Précisons enfin que l'accroissement du temps est piloté par l'horloge temps réel.

L'algorithme du joueur est le suivant:

Etape 1:
1.1 Mémoriser le marquage initial.
1.2 Pour l'origine des temps, calculer les intervalles de visibilité de chaque jeton.
1.3 Insérer dans l'échéancier les bornes minimales et maximales de ces intervalles.
1.4 Mettre tous les jetons dans l'état indisponible.
1.5 Mettre à zéro la marge temporelle d_{C_k} associée à chacune des étiquettes Kanban.

1.6 Mettre à zéro la marge temporelle agrégée moyenne D_{C_r} associée à chacune des contraintes de ressources C_r.

1.7 Mettre à zéro toutes les variables de fin d'opération qui sont associées aux transitions de fin d'opération.

Etape 2:

2.1 Si un événement de l'échéancier devient vrai:

2.1.1 Noter la nouvelle date courante.

2.1.2 Calculer la différence δ entre l'instant courant et l'instant précédent.

2.1.3 Appeler la procédure d'actualisation des valeurs des marges temporelles.

2.1.4 Aller à l'étape 3.

2.2 Aller à l'étape 5.

Etape 3: Si l'événement de l'échéancier qui devient vrai est la borne maximale d'un intervalle de visibilité:

3.1 Envoyer un signal d'alarme à l'opérateur pour indiquer que l'on passe dans un état anormal.

3.2 Appeler la procédure de diagnostic.

Etape 4: Si l'événement de l'échéancier qui devient vrai est la borne minimale d'un intervalle de visibilité:

4.1 Faire passer le jeton auquel est associé cette borne dans l'état *disponible*.

4.2 Si une transition de début d'opération est sensibilisée par les jetons disponibles du réseau:

4.2.1 Si la transition sensibilisée n'est pas en conflit structurel:

4.2.1.1 Franchir la transition.

4.2.1.2 Envoyer le message de début d'opération.

4.2.1.3 Appeler la procédure de fin de cycle.

4.2.1.4 Calculer les nouveaux intervalles de visibilité des jetons utilisés dans le tir de la transition.

4.2.1.5 Insérer dans l'échéancier les nouvelles bornes minimales et maximales de ces intervalles.

4.2.1.6 Mettre les jetons utilisés dans le tir de la transition dans l'état *indisponible*.

4.2.2 Si la transition sensibilisée est en conflit structurel:

4.2.2.1 Appeler la procédure de résolution de conflit.

4.2.2.2 Si la transition sensibilisée est solution de la procédure:

4.2.2.2.1 Franchir la transition.

4.2.2.2.2 Envoyer le message de début d'opération.

4.2.2.2.3 Appeler la procédure de fin de cycle.

4.2.2.2.4 Calculer les nouveaux intervalles de visibilité des jetons utilisés dans le tir de la transition.

4.2.2.2.5 Insérer dans l'échéancier les nouvelles bornes de ces intervalles.

4.2.2.2.6 Mettre les jetons utilisés dans le tir de la transition dans l'état *indisponible.*

4.2.3 Aller à l'étape 4.2.

4.3 Aller à l'étape 2.

Etape 5:

5.1 Si la variable de fin d'opération associée à une transition de fin d'opération devient vraie:

5.1.1 Noter la nouvelle date courante.

5.1.2 Calculer la différence δ entre l'instant courant et l'instant précédent.

5.1.3 Appeler la procédure d'actualisation des valeurs des marges temporelles.

5.1.4 Aller à l'étape 6.

5.2 Aller à l'étape 2.

Etape 6: Si cette transition n'est pas sensibilisée par les jetons disponibles du réseau:

6.1 Envoyer un signal d'alarme à l'opérateur pour indiquer que l'on passe dans un état anormal.

6.2 Appeler la procédure de diagnostic.

Etape 7: Si cette transition est sensibilisée par les jetons disponibles du réseau:

7.1 Si cette transition n'est pas en conflit structurel:

7.1.1 Franchir la transition.

7.1.2 Appeler la procédure de fin de cycle.

7.1.3 Calculer les nouveaux intervalles de visibilité des jetons utilisés dans le tir de la transition.

7.1.4 Insérer dans l'échéancier les nouvelles bornes minimales et maximales de ces intervalles.

7.1.5 Mettre les jetons utilisés dans le tir de la transition dans l'état *indisponible.*

7.1.6 Mettre la variable de fin d'opération associée à la transition à l'état faux.

7.2 Si cette transition est en conflit structurel:

7.2.1 Appeler la procédure de résolution de conflit.

7.2.2 Si la transition sensibilisée est solution de la procédure:

7.2.2.1 Franchir la transition.

7.2.2.2 Appeler la procédure de fin de cycle.

7.2.2.3 Calculer les nouveaux intervalles de visibilité des jetons utilisés dans le tir de la transition.

7.2.2.4 Insérer dans l'échéancier les nouvelles bornes minimales et maximales de ces intervalles.

7.2.2.5 Mettre les jetons utilisés dans le tir de la transition dans l'état *indisponible.*

7.2.2.6 Mettre la variable de fin d'opération associée à la transition à l'état faux.

7.3 Aller à l'étape 5.

Procédure de Fin de Cycle

Cette procédure est la même que pour le joueur avec retour arrière à quelques différences près.

S'il n'y a pas d'opération en cours en début de cycle, nous n'aurons à tester au niveau de cette procédure que des jetons indisponibles se trouvant dans des places "stock".

S'il y a des opérations en cours en début de cycle, nous devrons aussi tester des jetons indisponibles ou disponibles se trouvant dans des places "opération" et auquelles seront associés des intervalles dynamiques. Dans ce cas, l'étape 1 de l'algorithme devient:

1.1 Initialiser l'origine des temps $\theta \leftarrow 0$.

1.2 Mémoriser le marquage initial et l'état de chaque jeton (indisponible ou disponible).

1.3 Insérer dans l'échéancier les événements correpondant aux bornes minimales et maximales des intervalles de visibilité des jetons indisponibles.

1.4 Insérer dans l'échéancier les événements correspondant aux bornes maximales des intervalles de visibilité des jetons disponibles.

1.5 Mettre à zéro la marge temporelle d_{C_k} associée à chacune des étiquettes Kanban.

1.6 Mettre à zéro la marge temporelle agrégée moyenne D_{C_r} associée à chacune des contraintes de ressources C_r.

1.7 Mettre à zéro toutes les variables de fin d'opération qui sont associées aux transitions de fin d'opération.

Comme le test de fin de cycle n'est vérifié qu'après chaque événement (i.e. lorsqu'un événement de l'échéancier devient vrai ou qu'une variable de fin d'opération associée à une transition de fin d'opération devient vraie), nous devons introduire dans l'échéancier des événements de fin de cycle associés aux jetons se trouvant dans les places opération.

Considérons par exemple la partie de réseau représentée par la figure 4.15. Supposons que le jeton de $o_{g2,4}$ soit une étiquette Kanban d'une contrainte C_{k2}. En début de cycle ce jeton se trouve dans la place $o_{g2,4}$ dans l'état indisponible depuis une durée de 3. Si l'intervalle statique associé à $o_{g2,4}$ est $[7;7]$ (i.e. durée d'opération de 7) alors il suffit de modifier cet intervalle en début de cycle de la façon suivante:

$[7;7] \rightarrow [(7-3);(7-3)] = [4;4]$

Pour pouvoir tester cet état en fin de cycle au niveau du joueur, on doit introduire dans l'échéancier un événement de fin de cycle qui nous indique que le jeton se trouve dans $o_{g2,4}$ depuis une durée de 3. Après l'arrivée du jeton dans la place $s_{g2,4}$, pour le

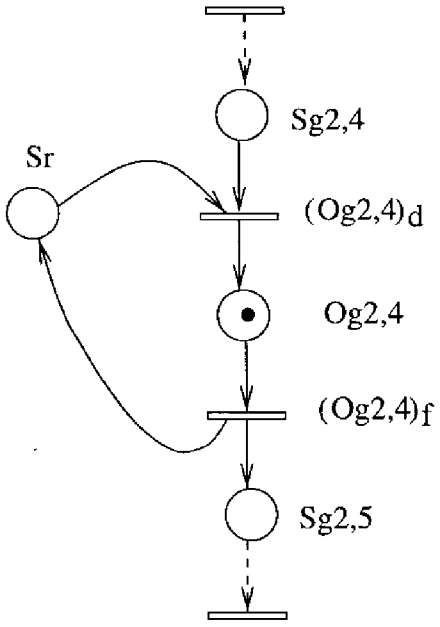

Figure 4.15: Evénement de fin de cycle 2

dernier passage de ce jeton dans cette place par rapport au cycle, lorsqu'à la date δ le jeton franchit $(o_{g2,4})_d$, alors on introduit dans l'échéancier la date $\delta + 3$ de fin de cycle associée à ce jeton. Nous pourrons ainsi savoir quand ce jeton aura atteint son état de fin de cycle. Il faudra rajouter entre l'étape 2 et 3 de l'algorithme l'étape suivante:

Si le premier événement de l'échéancier correspond à la date de fin de cycle associée à un jeton se trouvant dans une place opération alors appeler la procédure de fin de cycle.

Il se peut que l'on ait pris du retard sur d'autres opérations de la cellule et qu'à la date $\delta + 3$ la procédure de fin de cycle ne détecte pas la fin de cycle. Si tel est le cas, on ne peut pas interrompre l'opération $o_{g2,4}$. D'un autre côté, si nous continuons à l'exécuter le jeton de la place $o_{g2,4}$ va évoluer au-delà de son état de fin de cycle. Pour que l'on ne s'éloigne pas trop de la fin du cycle prévue, nous pouvons associer à la place suivante $s_{g2,5}$ un intervalle statique $[d_{max}; d_{max}]$. Ainsi, si nous franchissons la transition $o_{g2,4}$ avant que la fin de cycle ne soit détectée, alors l'intervalle $[d_{max}; d_{max}]$ devra maintenir le jeton dans la place $s_{g2,5}$ dans l'état indisponible jusqu'à la fin du cycle. En début du cycle suivant, si nous voulons nous resynchroniser sur un début de cycle normal, nous devrons modifier le premier intervalle statique associé à la place $s_{g2,5}$. Supposons par exemple que le premier intervalle statique de début de cycle associé à $s_{g2,5}$ soit $[0;4]_1$. Si le cycle précédent ne s'est pas terminé à la bonne date et a pris un retard suffisamment important pour que nous puissions terminer l'opération $o_{g2,4}$, alors nous pouvons modifier l'intervalle statique associé à $s_{g2,5}$ de la

façon suivante:

$[0;4]_1 \rightarrow [0+(7-3);4+(7-3)]_1 = [4;8]$

Procédure de Diagnostic

Cette procédure lance un programme de diagnostic pour détecter l'anomalie et tenter une action de reprise. Les résultats possibles de ce programme sont:

- Relaxation d'une ou de plusieurs contraintes (relaxation de la borne maximale ou minimale des intervalles de visibilité concernés) et retour au fonctionnement normal de l'algorithme.
- Appeler la procédure de fonctionnement anormal.
- Arrêt de l'algorithme du joueur et réinitialisation par l'opérateur dans l'état choisi par lui.

Lorsque l'on a une violation de contrainte au niveau d'un intervalle statique (par exemple, une durée d'opération réelle à cause d'un incident mineur a été plus longue que la durée maximale prévue qui est exprimée par la valeur de la borne maximale de l'intervalle attaché à la place opération correspondante), nous pouvons simplement augmenter la valeur de la borne maximale et continuer à appliquer le joueur normalement (à condition que la marge temporelle globale, c'est-à-dire la durée résiduelle, reste positive). Ce type de violation sera détecté à l'étape 3 et correspondra, soit à une durée d'usinage plus longue que prévue lorsque l'intervalle considéré sera celui d'une place opération, soit à une erreur de décision au niveau de la politique d'ordonnancement lorsque l'intervalle considéré est celui d'une place de type "stock".

Si la violation est directement liée à la valeur maximale d'une marge temporelle (i.e. durée résiduelle associée à un invariant de places négative), alors il est certain par contre que nous ne respecterons pas les taux de production fixés par le cahier des charges. L'erreur peut venir de l'accumulation de petites perturbations (plusieurs dépassements de la valeur de la borne maximale de plusieurs intervalles de visibilité) ou bien simplement d'une mauvaise politique d'ordonnancement. Ce type de violation de contrainte sera détecté par la procédure d'actualisation des marges temporelles et il faudra alors passer en fonctionnement anormal.

Procédure d'Actualisation des Marges Temporelles

Etape 1:

1.1 Pour chacune des étiquettes Kanban qui se trouvent dans l'état non réservé, on calcule la nouvelle valeur de la marge temporelle associée qui est:

$d_{C_k} \leftarrow d_{C_k} + \delta$.

1.2 Si une au moins des nouvelles durées résiduelles $((d_{C_k})_{max} - d_{C_k})$ est à valeur négative:

1.2.1 Envoyer un signal d'alarme à l'opérateur pour indiquer que l'on a une violation de contrainte.

1.2.2 Appeler la procédure de diagnostic.

Etape 2:

2.1 Pour les marges temporelles agrégées D_{C_r} associées aux contraintes de ressources C_r, η_r étant le nombre de jetons non réservés par invariant de places C_r, les nouvelles valeurs des marges temporelles sont:

$D_{C_r} \leftarrow D_{C_r} + \eta_r.\delta$.

2.2 Si une au moins des nouvelles durées résiduelles $(m_{s_r}.(d_{C_r})_{max} - D_{C_r})$ est à valeur négative:

2.2.1 Envoyer un signal d'alarme à l'opérateur pour indiquer que l'on a une violation de contrainte.

2.2.2 Appeler la procédure de diagnostic.

Procédure de Fonctionnement Anormal

Pendant que l'on va diagnostiquer les raisons des violations de contraintes les plus importantes (violation sur les marges par exemple), il est inutile de stopper la production et nous devons passer dans un mode de fonctionnement anormal. La procédure de fonctionnement anormal est la suivante:

Etape 1:

1.1 Passer en mode de fonctionnement anormal: $mode\_n \leftarrow 0$.

1.2 Remplacer tous les intervalles de visibilité par l'intervalle $[0; +\inf[$ à l'exeption des intervalles de fin de cycle.

Etape 2: Appliquer une politique de tir au plus tôt.

Etape 3: Si on reçoit une interruption de l'opérateur alors rebasculer en mode normal: $mode\_n \leftarrow 1$.

Etape 4: Si on atteint le marquage initial (fin de cycle) et que $mode\_n = 1$ alors aller à l'étape 1 du programme principal, sinon revenir à l'étape 2.

En fait, la seule différence lorsque l'on passe en fonctionnement anormal est que l'on relâche toutes les contraintes temporelles et qu'on n'utilise plus la procédure de résolution de conflit. On franchit systématiquement les transitions au plus tôt et lorsqu'un conflit ressource apparaît, on fait un choix arbitraire. Même avec ces modifications, nous fonctionnons toujours en régime périodique stationnaire forcé et

c'est le joueur qui fixe la condition de tir de la transition de synchronisation t_s. C'est pourquoi les intervalles de visibilité de fin de cycle (ceux qui en fin de cycle forcent les jetons à rester dans l'état indisponible) sont différents de l'intervalle $[0; +\infty[$. Au contraire, ils devront être tels que le jeton lorsqu'il arrive dans sa place de fin de cycle reste dans l'état indisponible jusqu'à ce que tous les jetons du réseau aient rejoint leur place de fin de cycle. Comme les performances de la cellule ne répondront plus aux prévisions calculées par la programmation mathématique, il faudra choisir des bornes à valeur suffisamment élevée. Par exemple, nous pourrons choisir pour l'intervalle statique de fin de cycle associé à une place Kanban s_k: $[2.(d_{C_k})_{max}; 2.(d_{C_k})_{max}]$. Comme ces jetons resteront dans ces places jusqu'au franchissement de t_s, il faudra que ces places n'appartiennent à aucune contrainte de ressource. En effet, dans le cas contraire, une ou plusieurs ressources pourraient être bloquées. Cela serait mauvais pour l'efficacité et pourrait éventuellement provoquer un blocage mortel.

En conservant un fonctionnement partiellement synchrone (qui dépend du tir de t_s), il sera très simple de se synchroniser à nouveau sur le fonctionnement normal lorsque la décision de reprise sera envoyée par l'opérateur.

Avec une politique de tir au plus tôt, le rendement de la cellule ne sera pas très bon, mais nous aurons limité la diminution de la production tout en diagnostiquant le problème, et lorsque la réparation du système aura été effectuée, nous pourrons rebasculer en fonctionnement normal sans qu'il y ait eu interruption de la production.

Bien sûr, si la violation de contrainte est liée à une perturbation du système trop importante pour poursuivre la production alors l'opérateur devra interrompre le fonctionnement de la cellule et, après réparation de cette dernière, relancer le joueur. Un problème de ce type est facilement diagnosticable lorsque par exemple, même après avoir augmenté la valeur d'une borne maximale d'un intervalle de visibilité (par exemple, deux fois la valeur de la marge temporelle maximale de l'invariant de places auquel appartient cet intervalle), le fonctionnement anormal persiste.

Lorsqu'un message de fin d'opération associé à une transition de fin d'opération devient vrai mais que le jeton en entrée n'est pas disponible (la borne minimale de l'intervalle de visibilité associée à ce jeton n'a pas été atteinte), cela veut dire que la durée d'opération a été plus courte que prévu et donc que le traitement de la pièce n'a pas été correctement effectué. Nous pouvons alors immédiatement marquer la pièce correspondante pour qu'elle ne passe pas le contrôle de qualité. Ceci est illustré par l'étape 6 de l'algorithme.

Tout cela montre bien l'intérêt d'une approche hybride qui associe la théorie des réseaux de Petri et les problèmes de satisfaction de contraintes au niveau du pilotage temps réel d'une cellule. Le réseau de Petri sans considérations temporelles permet de respecter les contraintes de gamme de fabrication et d'allocation de ressource indépendamment de toute politique de production. Les intervalles temporels associés aux places d'attente permettent d'améliorer le pilotage de la cellule sur un horizon très court (intervalle de visibilité des jetons) et les contraintes temporelles agrégées (marges temporelles associées aux invariants de places) permettent d'analyser le comportement moyen de la cellule sur un cycle. La surveillance en temps réel des contraintes temporelles permet d'assurer une meilleure qualité de production (surveillance des intervalles temporels associés aux places représentant les opérations)

et la détection de toute dérive par rapport à l'ordonnancement prévisionnel (surveillance des intervalles temporels associés aux places d'attente). Cette vérification des contraintes à respecter évite d'exécuter des programmes de diagnostic complexes qui doivent normalement faire de nombreux retours arrières sur les états antérieurs du système. Simplement en observant le type de violation de contrainte, nous avons déjà une bonne idée du problème et du type de compensation à effectuer. Finalement, nous pouvons établir durant l'usinage des pièces des conditions nécessaires de "bon usinage" qui, lorsqu'elles ne sont pas respectées, éliminent les pièces défectueuses sans que d'autres moyens de détection beaucoup plus complexes et généralement liés au contrôle de qualité ne soient utilisés.

La procédure de résolution de conflit utilisée dans l'algorithme du joueur temps réel est la même que celle utilisée dans le joueur avec retour arrière. Comme le modèle n'est plus p-temporel t-temporisé mais simplement p-temporel, il suffit de remplacer les durées d'opération qui étaient attachées aux transitions par les durées d'opération qui sont maintenant attachées aux places d'opération. Si on a un intervalle min/max attaché à la place opération, il suffit de choisir comme durée d'opération la durée maximale. Comme nous travaillons dans le contexte de l'analyse sous contrainte, il suffit de garantir que la solution est acceptable dans la pire des situations pour garantir qu'elle le sera aussi dans le meilleur des cas. En dehors de ces différences avec le modèle p-temporel t-temporisé, le principe reste le même puisqu'en choisissant la valeur maximale comme durée d'opération on peut se ramener à un modèle p-temporel t-temporisé pour lequel les t-temporisations seraient les valeurs maximales des durées d'opération prévues. Rappelons que même si la procédure de résolution de conflit peut fournir plus de deux solutions, au niveau du joueur temps réel seules deux solutions sont à considérer: ou bien on franchit immédiatement la transition sensibilisée par les jetons disponibles, ou bien on laisse les jetons qui la sensibilisent dans l'état disponible et on franchira la transition ultérieurement.

En ce qui concerne la réinitialisation de l'algorithme en début de cycle de fabrication, il est bon de préciser que lorsque l'on revient à l'étape 1 pour recommencer un nouveau cycle, il y a une réactualisation des intervalles statiques. Soit par exemple l'intervalle statique : $[(d_p)_{min^k}; (d_p)_{max^k}]$ associé à la place p. Le k^{ieme} jeton qui traversera cette place aura son intervalle de visibilité calculé en fonction de sa date d'arrivée et de cet intervalle. La variable k qui apparait dans l'intervalle statique sera remise à 1 ($k \leftarrow 1$) chaque fois que l'on reviendra à l'étape 1 et les intervalles de visibilité modifiés en conséquence bien sûr.

Chapitre 5

Exemple de Modélisation, d'Analyse et d'Ordonnancement

5.1 Modélisation de la Cellule

5.1.1 Contraintes de Synchronisation

Considérons une cellule capable de réaliser deux types de produits p_1 et p_2. Le produit p_1 peut être réalisé suivant la gamme de fabrication $g1$ ou $g2$ (figure 5.1). Le produit p_2 peut être réalisé suivant la gamme de fabrication $g3$ (figure 5.2). De ces trois gammes de fabrication $g1$, $g2$ et $g3$, nous déduisons les contraintes Kanban C_{k1}, C_{k2} et C_{k3} (figure 5.3). Les places Kanban s_{k1}, s_{k2} et s_{k3} sont utilisées aussi bien pour la gestion du nombre de palettes que pour déterminer la capacité des stocks intermédiaires. En effet, chaque place Kanban s_{ki} contient mi étiquettes Kanban qui autorisent le traitement de nouvelles pièces sur la gamme gi en fonction des capacités des stocks intermédiaires de cette gamme et aussi du nombre de palettes disponibles.

Dans cette cellule, un certain nombre de ressources sont utilisées pour réaliser les diverses opérations des gammes de fabrication.

Les jetons de s_{r1} de la contrainte C_{r1} (figure 5.4) représentent les ressources flexibles utilisées pour traiter les opérations $o_{g1,1}$, $o_{g2,1}$ et $o_{g3,1}$.

Les jetons de s_{r2} de la contrainte C_{r2} (figure 5.5) représentent les ressources utilisées pour traiter les opérations $o_{g1,2}$.

Les jetons de s_{r3} de la contrainte C_{r3} (figure 5.6) représentent les ressources flexibles utilisées pour traiter les opérations $o_{g1,3}$ et $o_{g2,3}$.

Les jetons de s_{r4} de la contrainte C_{r4} (figure 5.7) représentent les ressources flexibles utilisées pour traiter les opérations $o_{g1,4}$, $o_{g2,4}$ et $o_{g3,3}$.

Et enfin, les jetons de s_{r5} de la contrainte C_{r5} (figure 5.8) représentent les ressources flexibles utilisées pour traiter les opérations $o_{g2,2}$ et $o_{g3,2}$.

Le réseau de la figure 5.9 représente la contrainte cyclique C_c qui définit les exigences de la production. pd_{g1}, pd_{g2} et pd_{g3} sont les variables entières qui vont définir combien de produits doivent être réalisés sur les gammes de fabrication $g1$, $g2$ et $g3$ pour un cycle de fabrication.

5.1.2 Modèle Global

Maintenant que les contraintes de synchronisation de la cellule ont été définies, nous pouvons en fusionnant les transitions communes à chaque contrainte établir le modèle global de la cellule sous la forme d'un réseau de Petri unique. Ce réseau est représenté par la figure 5.10. Le signe "+" dans les places d'entrée des gammes g1, g2 et g3 signifie que les buffers d'entrée de la cellule sont alimentés en permanence.

5.1.3 Analyse des Propriétés

Invariants de Places

Conformément à ce qui a été dit dans le chapitre 2, chaque contrainte est un invariant de places et nous avons les relations suivantes:

- $M(s_{k1}) + M(s_{g1,2}) + M(s_{g1,3}) + M(s_{g1,4}) = m1$
- $M(s_{k2}) + M(s_{g2,2}) + M(s_{g2,3}) + M(s_{g2,4}) = m2$
- $M(s_{k3}) + M(s_{g3,2}) + M(s_{g3,3}) = m3$
- $M(s_{r1}) = m4$
- $M(s_{r2}) = m5$
- $M(s_{r3}) = m6$
- $M(s_{r4}) = m7$
- $M(s_{r5}) = m8$
- $M(c_{g1,1}) + M(c_{g1,2}) = m9$
- $M(c_{g2,1}) + M(c_{g2,2}) = m10$
- $M(c_{g3,1}) + M(c_{g3,2}) = m11$

Après la fusion des transitions communes à chaque contrainte, les invariants de places sont conservés et ils sont donc des invariants de places du réseau global.

Invariants de Transitions

Conformément à ce qui a été dit dans le chapitre 2, le modèle global de la cellule est un invariant de transitions tel que les fréquences d'occurence des transitions sont imposées par la contrainte cyclique C_c. Dans le cas du réseau qui est représenté par la figure 5.10, nous pouvons dire en nous basant sur les résultats énoncés dans le chapitre 2 que si pour un cycle de fabrication, la transition de synchronisation t_s est franchie une fois, alors les transitions de la gamme $g1$ seront franchies pd_{g1} fois, les transitions de la gamme $g2$ seront franchies pd_{g2} fois et les transitions de la gamme $g3$ seront franchies pd_{g3} fois.

Bonnes Propriétés

Il existe au niveau du réseau global une couverture de composantes conservatives et répétitives stationnaires. Nous en déduisons que le réseau est borné. Pour montrer que le réseau est vivant et réinitialisable nous pouvons appliquer les règles de réduction d'un réseau de Petri:

Etape 1: On supprime les places ressource (s_{r1}, s_{r2}, s_{r3}, s_{r4}, s_{r5}) qui sont des places implicites dégénérées.

Etape 2: On supprime les places stock intermédiaire ($s_{g1,2}$, $s_{g1,3}$, $s_{g1,4}$, $s_{g2,2}$, $s_{g2,3}$, $s_{g2,4}$, $s_{g3,2}$, $s_{g3,3}$) qui sont des places substituables.

Etape 3: On supprime les places Kanban (s_{k1}, s_{k2}, s_{k3}) qui sont des places implicites dégénérées.

Etape 4: On supprime les places $c_{g1,1}$, $c_{g2,1}$ et $c_{g3,1}$ qui sont des places substituables.

Etape 5: On supprime les places $c_{g1,2}$ et $c_{g2,2}$ qui sont des places implicites dégénérées.

On obtient comme résultat le réseau de la figure 5.11 qui est bien un réseau vivant et réinitialisable et ceci quel que soit son marquage initial pourvu qu'il y ait au moins un jeton (une machine) dans chaque place ressource (pool) et un jeton (une palette) dans chaque place Kanban.

5.1.4 Modèle p-temporel t-temporisé

Les t-temporisations qui représentent les durées d'opération exprimées en "unité de temps" sont les suivantes:

$d_{o_{g1,1}} = 12$
$d_{o_{g1,2}} = 4$
$d_{o_{g1,3}} = 2$
$d_{o_{g1,4}} = 7$
$d_{o_{g2,1}} = 12$
$d_{o_{g2,2}} = 2$
$d_{o_{g2,3}} = 2$
$d_{o_{g2,4}} = 7$
$d_{o_{g3,1}} = 10$
$d_{o_{g3,2}} = 3$
$d_{o_{g3,3}} = 12$

Ces durées d'opération représentent une partie de l'ensemble des contraintes à respecter et correspondent aux paramètres de notre approche.

Les intervalles statiques d'attente associés aux places du réseau p-temporel t-temporisé représentent les variables à définir de notre approche. Ces intervalles exprimeront les durées minimales et maximales pendant lesquelles les jetons resteront dans les places dans l'état *non réservé*. Nous allons analyser maintenant les domaines dans

lesquels ces durées d'attente vont prendre leurs valeurs afin de définir une politique de contrôle à la fois robuste et efficace.

5.2 Analyse Quantitative de la Cellule

5.2.1 Mise en Equation

Spécification de la Production Cyclique

La cellule à laquelle est appliquée la politique cyclique exprimée par la contrainte C_c est capable de réaliser deux types de produits p_1 et p_2. Le type p_1 peut être réalisé aussi bien sur la gamme $g1$ que sur la gamme $g2$, et le type p_2 seulement sur la gamme $g3$. Supposons que les spécifications données par le cahier des charges quant aux proportions de pièces à traiter soient telles que $\tau_{p_1} = 60\%$ et $\tau_{p_2} = 40\%$. Le plus petit nombre de produits qui peuvent être réalisés pour un cycle de fabrication afin de respecter ces taux est $N_p = 5$ avec $N_{p_1} = 3$ et $N_{p_2} = 2$ et nous devons dans ce cas introduire les contraintes suivantes:

$$pd_{g1} + pd_{g2} = 3 \tag{5.1}$$

$$pd_{g3} = 2$$

La flexibilité de gamme (choix entre plusieurs gammes possibles) de la cellule est ainsi exprimée par l'équation 5.1 qui représente l'ensemble au sein duquel les variables entières pd_{g1} et pd_{g2} prendront leurs valeurs. Le marquage de la contrainte C_c sera tel que $m9 = pd_{g1}$, $m10 = pd_{g2}$ et $m11 = pd_{g3}$.

Période pour les Contraintes Kanban

Nous choisissons comme marquage pour les contraintes Kanban: $m1 = m2 = m3 = 2$. Le nombre de jetons dans chaque place Kanban représente le nombre de palettes disponibles ainsi que les capacités de stockage de ces dernières au sein de la cellule. L'expression de la période de fabrication formalisant chacune des contraintes Kanban est alors la suivante:

pour C_{k1}: $\pi = 12,5.pd_{g1} + d_{C_{k1}}$
pour C_{k2}: $\pi = 11,5.pd_{g2} + d_{C_{k2}}$
pour C_{k3}: $\pi = 12,5.pd_{g3} + d_{C_{k3}}$

Période pour les Contraintes de Ressource

Si nous cherchons à évaluer les performances pour un nombre minimal de ressources alors nous avons comme marquage pour les contraintes de ressources: $m4 = m5 = m6 = m7 = m8 = 1$. L'expression de la période de fabrication formalisant chacune des contraintes de ressource est alors la suivante:

pour C_{r1}: $\pi = 12.pd_{g1} + 12.pd_{g2} + 10.pd_{g3} + d_{C_{r1}}$
pour C_{r2}: $\pi = 4.pd_{g1} + d_{C_{r2}}$

pour C_{r3}: $\pi = 2.pd_{g1} + 2.pd_{g2} + d_{C_{r3}}$
pour C_{r4}: $\pi = 7.pd_{g1} + 7.pd_{g2} + 12.pd_{g3} + d_{C_{r4}}$
pour C_{r5}: $\pi = 2.pd_{g2} + 3.pd_{g3} + d_{C_{r5}}$

5.2.2 Analyse des Contraintes Linéaires

Nous allons maintenant analyser l'ensemble des contraintes linéaires afin d'aider à la conception et à l'ordonnancement de la cellule.

Calcul de la Borne Minimale

Le régime stationnaire périodique forcé correspond à l'efficacité maximale de la cellule pour les taux de production fixés par la contrainte cyclique C_c. L'efficacité maximale est obtenue pour la valeur minimale de la période π compte tenu de l'ensemble des contraintes linéaires. Nous devons donc minimiser π sous l'ensemble des contraintes suivantes:

$pd_{g1} + pd_{g2} = 3$
$pd_{g3} = 2$
$\pi = 12,5.pd_{g1} + d_{C_{k1}}$
$\pi = 11,5.pd_{g2} + d_{C_{k2}}$
$\pi = 12,5.pd_{g3} + d_{C_{k3}}$
$\pi = 12.pd_{g1} + 12.pd_{g2} + 10.pd_{g3} + d_{C_{r1}}$
$\pi = 4.pd_{g1} + d_{C_{r2}}$
$\pi = 2.pd_{g1} + 2.pd_{g2} + d_{C_{r3}}$
$\pi = 7.pd_{g1} + 7.pd_{g2} + 12.pd_{g3} + d_{C_{r4}}$
$\pi = 2.pd_{g2} + 3.pd_{g3} + d_{C_{r5}}$

Une des solutions obtenue à partir du logiciel de programmation mathématique *XPRESS* est:

$\pi_{min} = 56$
$pd_{g1} = 1$
$pd_{g2} = 2$
$pd_{g3} = 2$
$(d_{C_{k1}})_{min} = 43,5$
$(d_{C_{k2}})_{min} = 33$
$(d_{C_{k3}})_{min} = 31$
$(d_{C_{r1}})_{min} = 0$
$(d_{C_{r2}})_{min} = 52$
$(d_{C_{r3}})_{min} = 50$
$(d_{C_{r4}})_{min} = 11$

$(d_{C_{r5}})_{min} = 46$

Compte tenu de ce qui a été dit dans le chapitre 3, afin de rester cohérent avec la notion de régime stationnaire périodique forcé, les poids pd_g doivent être des multiples du nombre d'étiquettes Kanban m_{s_k}. Comme nous avons $pd_{g1} = 1$, il n'est pas nécessaire pour cette politique cyclique d'avoir deux étiquettes Kanban dans s_{k1} et au lieu d'avoir $m1 = 2$, nous posons $m1 = 1$. L'équation formalisant la contrainte C_{k1} devient alors:

$\pi = 25 + d_{C_{k1}}$

Après minimisation de π, on a toujours $\pi_{min} = 56$ et $(d_{C_{k1}})_{min} = 31$.

On remarque que la valeur des marges temporelles $d_{C_{r1}}$ et $d_{C_{r4}}$ sont très inférieures aux valeurs des autrés marges de la cellule. Ceci signifie que les ressources des contraintes C_{r1} et C_{r4} fonctionneront dans un état quasi-saturé. Une telle différence montre que la charge de travail au sein de la cellule n'est pas bien répartie puisqu'elle est beaucoup plus importante pour ces deux ressources. Un moyen d'améliorer les performances de la cellule est de mieux répartir la charge de travail et d'ajouter une ressource en s_{r1} et une ressource en s_{r4}.

Le marquage devient alors $m4 = m7 = 2$ et nous devons minimiser π sous l'ensemble des contraintes suivantes:

$pd_{g1} + pd_{g2} = 3$
$pd_{g3} = 2$
$\pi = 25.pd_{g1} + d_{C_{k1}}$
$\pi = 11,5.pd_{g2} + d_{C_{k2}}$
$\pi = 12,5.pd_{g3} + d_{C_{k3}}$
$\pi = 6.pd_{g1} + 6.pd_{g2} + 5.pd_{g3} + d_{C_{r1}}$
$\pi = 4.pd_{g1} + d_{C_{r2}}$
$\pi = 2.pd_{g1} + 2.pd_{g2} + d_{C_{r3}}$
$\pi = 3.5.pd_{g1} + 3.5.pd_{g2} + 6.pd_{g3} + d_{C_{r4}}$
$\pi = 2.pd_{g2} + 3.pd_{g3} + d_{C_{r5}}$

Après minimisation de π, la solution donnée par *XPRESS* est:

$\pi_{min} = 28$
$(d_{C_{k1}})_{min} = 3$
$(d_{C_{k2}})_{min} = 5$
$(d_{C_{k3}})_{min} = 3$
$(d_{C_{r1}})_{min} = 0$
$(d_{C_{r2}})_{min} = 24$
$(d_{C_{r3}})_{min} = 22$
$(d_{C_{r4}})_{min} = 5,5$

$$(d_{C_{r5}})_{min} = 18$$

Nous remarquons que nous avons alors une diminution de moitié de la valeur de la période π et des écarts beaucoup plus raisonnables entre les valeurs des marges temporelles de la cellule.

Calcul de la Borne Maximale

Considérons le modèle théorique de la cellule. Nous pouvons en introduisant des exigences de marges minimales sur certaines opérations essayer de prendre en compte les possibles retards qui peuvent exister du fait des diverses synchronisations du modèle global. De plus, nous rendrons ainsi le modèle plus robuste vis-à-vis du fonctionnement réel de la cellule.

Nous avons:

$$(d_{C_{k1}})_{min} = (d_{C_{k3}})_{min} < (d_{C_{k2}})_{min}$$

En nous basant sur ce résultat, si un conflit ressource apparaît entre une opération de $g1$, une opération de $g2$ et une opération de $g3$, alors nous traiterons en dernier l'opération de $g2$ puisque c'est la contrainte C_{k2} qui a la marge temporelle la plus importante. Manifestement, nous n'avons pas de critère pour décider la façon dont doit être traité le conflit entre l'opération de $g1$ et de $g3$ puisque les marges temporelles sont identiques, et les deux possibilités doivent donc être considérées. L'ordre de résolution des conflits aura probablement pour conséquence d'induire des durées d'attente supplémentaires sur les places des contraintes Kanban. Ceci nous amène à fixer les exigences de marges temporelles minimales suivantes:

$$\begin{aligned}
d_{C_{k1}} &\geq max\{d_{o_{g1,1}}; d_{o_{g1,4}}\} \\
d_{C_{k3}} &\geq max\{d_{o_{g1,1}}; d_{o_{g1,4}}\} \\
d_{C_{k2}} &\geq max\{d_{o_{g1,1}}; d_{o_{g1,3}}; d_{o_{g1,4}}\} + max\{d_{o_{g3,1}}; d_{o_{g3,2}}; d_{o_{g3,3}}\}
\end{aligned}$$

avec

$$\begin{aligned}
&max\{d_{o_{g1,1}}; d_{o_{g1,4}}\} = 12 \\
&max\{d_{o_{g1,1}}; d_{o_{g1,3}}; d_{o_{g1,4}}\} = 12 \\
&max\{d_{o_{g3,1}}; d_{o_{g3,2}}; d_{o_{g3,3}}\} = 12
\end{aligned}$$

La première valeur correspond à la durée d'utilisation la plus longue des ressources communes aux gammes $g3$ et $g1$. La deuxième à la durée d'utilisation la plus longue des ressources communes aux gammes $g2$ et $g1$ sur la gamme $g1$. Et enfin la troisième à la durée d'utilisation la plus longue des ressources communes aux gammes $g2$ et $g3$ sur la gamme $g3$. Ces durées d'utilisation de ressources doivent être vues comme des attentes moyennes et globales associées à l'ensemble des places des contraintes C_{k1}, C_{k2} et C_{k3}. Nous pourrions bien sûr choisir d'autres critères pour fixer les exigences de marges minimales, ces choix dépendant essentiellement de la connaissance de la façon dont se comporte la cellule aussi bien pour le fonctionnement normal que perturbé.

Nous pouvons maintenant minimiser la période π sous l'ensemble des contraintes suivantes:

$$\pi = 25 + d_{C_{k1}}$$
$$\pi = 23 + d_{C_{k2}}$$
$$\pi = 25 + d_{C_{k3}}$$
$$\pi = 28 + d_{C_{r1}}$$
$$\pi = 4 + d_{C_{r2}}$$
$$\pi = 6 + d_{C_{r3}}$$
$$\pi = 22,5 + d_{C_{r4}}$$
$$\pi = 10 + d_{C_{r5}}$$
$$d_{C_{k1}} \geq 12$$
$$d_{C_{k2}} \geq 24$$
$$d_{C_{k3}} \geq 12$$

XPRESS nous donne comme nouvelles valeurs:

$$\pi_{max} = 47$$
$$(d_{C_{k1}})_{max} = 22$$
$$(d_{C_{k2}})_{max} = 24$$
$$(d_{C_{k3}})_{max} = 22$$
$$(d_{C_{r1}})_{max} = 19$$
$$(d_{C_{r2}})_{max} = 43$$
$$(d_{C_{r3}})_{max} = 41$$
$$(d_{C_{r4}})_{max} = 24,5$$
$$(d_{C_{r5}})_{max} = 37$$

Si nous combinons les bornes minimales et maximales, nous avons alors les contraintes suivantes:

$$28 \leq \pi \leq 47$$

$3 \leq d_{C_{k1}} \leq 22$
$5 \leq d_{C_{k2}} \leq 24$
$3 \leq d_{C_{k3}} \leq 22$
$0 \leq d_{C_{r1}} \leq 19$
$24 \leq d_{C_{r2}} \leq 43$
$22 \leq d_{C_{r3}} \leq 41$
$5,5 \leq d_{C_{r4}} \leq 24,5$
$18 \leq d_{C_{r5}} \leq 37$

5.3 Ordonnancement de la Cellule

5.3.1 Définition des Contraintes Statiques

Comme nous l'avons dit dans le chapitre précédent, le travail est difficilement réparti de façon égale sur chacune des ressources d'un même pool. Nous devons donc associer à chaque contrainte de ressource une marge temporelle agrégée et nous avons:

$0 \leq D_{C_{r1}} \leq 38$
$24 \leq D_{C_{r2}} \leq 43$
$22 \leq D_{C_{r3}} \leq 41$
$11 \leq D_{C_{r4}} \leq 49$
$18 \leq D_{C_{r5}} \leq 37$

Pour les contraintes Kanban par contre, la marge temporelle est exactement la même pour chaque étiquette Kanban (chaque jeton pris individuellement dans une contrainte Kanban) et nous conservons donc les contraintes:

$3 \leq d_{C_{k1}} \leq 22$
$5 \leq d_{C_{k2}} \leq 24$
$3 \leq d_{C_{k3}} \leq 22$

On choisit comme marquage initial du réseau global:

$M(c_{g1,1}) = M(s_{k1}) = M(s_{r2}) = M(s_{r3}) = M(s_{r5}) = 1$
$M(c_{g2,1}) = M(c_{g3,1}) = M(s_{k2}) = M(s_{k3}) = M(s_{r1}) = M(s_{r4}) = 2$

Les autres places ne contiennent pas de jetons et leur marquage est égal à zéro.

Comme cet état correspond au démarrage de la cellule, les jetons sont tous dans l'état indisponible.

L'opérateur chargé du bon fonctionnement de la cellule doit définir les intervalles statiques (à partir desquels seront calculés les intervalles de visibilité des jetons et par conséquent les dates de tir des transitions) les mieux adaptés pour l'ordonnancement de la cellule. Il doit au départ définir un premier ensemble d'intervalles qui lui paraissent à peu près bons. Ces intervalles doivent être choisis en fonction des marges temporelles associées aux jetons des invariants de places. En d'autres termes, il faut désagréger les marges temporelles pour réduire le domaine de décision à l'intérieur duquel l'algorithme du joueur devra résoudre les conflits de ressources.

Nous choisissons les intervalles statiques associés aux stocks intermédiaires des gammes $g1$, $g2$ et $g3$ de la façon suivante:

place $s_{g1,2}$: $[0;0]_1$

On choisit cet intervalle puisque pour réaliser l'opération $o_{g1,2}$, nous avons une ressource propre en s_{r2}.

place $s_{g1,3}$: $[0;2]_1$

Le critère de choix de cet intervalle est le suivant: si la ressource nécessaire à l'exécution de $o_{g1,3}$ n'est pas disponible, c'est qu'elle est utilisée pour l'opération $o_{g2,3}$ qui a une durée de 2. Donc dans le meilleur des cas, on n'attendra pas (la ressource sera immédiatement disponible pour le tir de $o_{g1,3}$) et dans le pire des cas (le retard maximum qu'on acceptera sans qu'il y ait violation de contrainte) on attendra que l'opération $o_{g2,3}$ soit terminée. Nous procédons de la même façon pour fixer les intervalles des places suivantes:

place $s_{g1,4}$: $[0;12]_1$
place $s_{g2,2}$: $[0;3]_1$ $[0;3]_2$
place $s_{g2,3}$: $[0;2]_1$ $[0;2]_2$
place $s_{g2,4}$: $[0;12]_1$ $[0;12]_2$
place $s_{g3,2}$: $[0;2]_1$ $[0;2]_2$
place $s_{g3,3}$: $[0;7]_1$ $[0;7]_2$

Nous allons maintenant définir les intervalles statiques associés aux places Kanban. Considérons la place s_{k1}. La marge moyenne consommée par les stocks intermédiaires de la gamme $g1$ compte tenu des intervalles statiques qui ont été définis est:

$$\frac{((d_{s_{g1,2}})_{min^1}+(d_{s_{g1,2}})_{max^1})}{2}+\frac{((d_{s_{g1,3}})_{min^1}+(d_{s_{g1,3}})_{max^1})}{2}+\frac{((d_{s_{g1,4}})_{min^1}+(d_{s_{g1,4}})_{max^1})}{2}=7$$

La marge maximale restante associée au jeton de C_{k1} est alors à peu près:

$$(d_{C_{k1}})_{max}-7=15$$

Nous choisissons donc comme premier intervalle associé à s_{k1}: $[0;15]_1$

Le deuxième intervalle associé à s_{k1} correspondra au retour du jeton dans la place Kanban à la fin du cycle. Le jeton devra rester dans cette place de fin de cycle dans l'état indisponible jusqu'à ce que la fin de cycle soit détectée et la transition de synchronisation t_s franchie. Comme a priori, nous pouvons franchir les transitions de la gamme $g1$ au plus tôt (les bornes minimales des intervalles statiques associés aux places de C_{k1} sont toutes nulles), nous choisissons donc comme deuxième intervalle statique associé à s_{k1}: $[22;22]_2$, avec 22 la valeur de la marge maximale associée au jeton de C_{k1}. Ainsi, si après 22 la fin de cycle n'a pas été détectée, nous aurons de toute façon une violation de contrainte sur la marge associée au jeton de C_{k1}.

Si nous procédons à un raisonnement similaire pour les places des contraintes C_{k2} et C_{k3}, nous trouvons pour l'ensemble des places du modèle les intervalles statiques suivants:

place $s_{g1,2}$: $[0;0]_1$
place $s_{g1,3}$: $[0;2]_1$
place $s_{g1,4}$: $[0;12]_1$
place s_{k1}: $[0;15]_1$ $[22;22]_2$
place $s_{g2,2}$: $[0;3]_1$ $[0;3]_2$
place $s_{g2,3}$: $[0;2]_1$ $[0;2]_2$
place $s_{g2,4}$: $[0;12]_1$ $[0;12]_2$
place s_{k2}: $[0;15]_1$ $[0;15]_2$ $[24;24]_3$ $[24;24]_4$
place $s_{g3,2}$: $[0;2]_1$ $[0;2]_2$
place $s_{g3,3}$: $[0;7]_1$ $[0;7]_2$
place s_{k3}: $[0;15]_1$ $[0;15]_2$ $[22;22]_3$ $[22;22]_4$

Précisons au niveau de ces notations indicées qu'elles se lisent de la façon suivante: par exemple pour la place s_{k3}, le premier intervalle correspond au premier

passage du premier jeton Kanban dans cette place, le deuxième au premier passage du deuxième jeton, le troisième au deuxième passage du premier jeton, et enfin le dernier au deuxième passage du deuxième jeton.

La façon dont nous avons calculé ces intervalles dépend d'une approche heuristique. Il est évident que le choix des critères utilisés pour le calcul de ces intervalles sera d'autant meilleur que la connaissance du système par l'opérateur sera bonne.

Nous ne désagrégeons pas les marges temporelles associées aux jetons ressource pour conserver le maximum de flexibilité, ce qui veut dire que les intervalles statiques associés aux places ressource ont pour borne minimale 0 et pour borne maximale la valeur de la marge agrégée maximale (exemple pour s_{r2} l'intervalle associé quel que soit le passage du jeton dans le cycle est: $[0;43]$).

A partir de ces conditions initiales, l'opérateur peut lancer l'algorithme du joueur. Cet algorithme en utilisant des mécanismes de retour arrière et des mécanismes de relaxation de contraintes doit nous donner une solution admissible ainsi probablement que de nouvelles contraintes temporelles (nouveaux intervalles statiques).

5.3.2 Joueur

Nous allons maintenant parcourir les étapes principales de l'algorithme du joueur et détailler en particulier celles qui mettent en évidence des résolutions de conflits de ressources.

En début de cycle, à la date 0, les transitions $o_{g1,1}$, $o_{g2,1}$ et $o_{g3,1}$ sont en conflit pour l'utilisation des ressources de s_{r1}. Nous devons donc appeler la procédure de résolution de conflit que nous avons présenté au chapitre 4. Cette procédure doit énumérer tous les ordonnancements possibles et se baser sur des considérations temporelles faisant intervenir aussi bien les marges agrégées (associées aux jetons des invariants de places) que les intervalles de visibilité des jetons (obtenus en fonction des dates et des intervalles statiques associés aux places) pour choisir une solution acceptable. Puisque nous devons exécuter une fois $o_{g1,1}$, deux fois $o_{g2,1}$ et deux fois $o_{g3,1}$, nous avons en tout vingt solutions à regarder. Nous ne regarderons que les deux séquences:

$(o_{g3,1} \rightarrow o_{g3,1} \rightarrow o_{g1,1} \rightarrow o_{g2,1} \rightarrow o_{g2,1})$ (stratégie 1)
$(o_{g3,1} \rightarrow o_{g1,1} \rightarrow o_{g3,1} \rightarrow o_{g2,1} \rightarrow o_{g2,1})$ (stratégie 2)

les autres se rapprochant bien trop vite des marges temporelles maximales autorisées. Nous devons donc vérifier ce que nous donnent les stratégies 1 et 2 par rapport aux contraintes temporelles à respecter et faire un choix. Rappelons qu'à la date 0 les marges temporelles agrégées associées aux jetons Kanban sont:

pour C_{k1}: $[3;22]$
pour C_{k2}: $[5;24]$
pour C_{k3}: $[3;22]$

et les intervalles de visibilité des jetons Kanban sont:
pour s_{k1}: $[0;15]_1$
pour s_{k2}: $[0;15]_1$ $[0;15]_2$
pour s_{k3}: $[0;15]_1$ $[0;15]_2$

Stratégie 1:
A la date 0, les deux jetons Kanban de s_{k3} passent dans l'état réservé ainsi que les deux jetons ressource de s_{r1}. Les jetons Kanban de s_{k1} et de s_{k2} restent en attente dans l'état disponible.

A la date 10, on tire la transition $o_{g3,1}$ deux fois, le jeton de s_{k1} passe dans l'état réservé, un jeton de s_{k2} passe dans l'état réservé et l'autre jeton de s_{k2} reste en attente d'une ressource dans l'état disponible. Les marges temporelles agrégées actualisées sont:
pour le jeton de C_{k1}: $[(3-10);(22-10)] = [-7;12]$
pour les deux jetons de C_{k2}: $[(5-10);(24-10)] = [-5;14]$
pour les deux jetons de C_{k3}: $[3;22]$
Nous n'avons pas de violation des intervalles de visibilité associés à s_{k1} et à s_{k2} et l'intervalle actualisé pour le jeton disponible restant de s_{k2} est:
$[0;15-10]_2 = [0;5]_2$.
Nous voyons dès à présent que nous aurons une violation sur cet intervalle lorsque les jetons ressource de s_{r1} seront passés à nouveau dans l'état disponible puisque les opérations $o_{g1,1}$ et $o_{g2,1}$ vont durer 12 unités de temps et que la durée maximale de l'intervalle de visibilité est 5. Nous devons donc dès à présent modifier les intervalles statiques associés au deuxième jeton Kanban de s_{k2}. Comme ce jeton est en retard sur tous les autres en début de cycle, il n'attendra probablement pas en fin de cycle pour une ressource et nous pouvons utiliser par exemple une partie de la marge que nous avons sur l'intervalle $[0;12]_2$ qui est associé à la dernière place $s_{g2,4}$ de la gamme. Pour ne pas avoir de violation de contrainte dès le début de cycle nous effectuons donc les transformations suivantes:
place $s_{g2,4}$: $[0;12]_2 \rightarrow [0;5]_2$
place s_{k2}: $[0;15]_2 \rightarrow [0;22]_2$
Nous avons transféré 7 unités de temps de l'intervalle de la place $s_{g2,4}$ à celui de la place s_{k2} et l'intervalle de visibilité du jeton Kanban disponible de s_{k2} devient:
$[0;12]_2$.

A la date 22, on tire $o_{g1,1}$ et $o_{g2,1}$ et le jeton restant de S_{k2} passe dans l'état réservé. Les marges temporelles agrégées actualisées deviennent:
pour le jeton de C_{k1}: $[-7;12]$
pour le premier jeton de C_{k2}: $[-5;14]$
pour le deuxième jeton de C_{k2}: $[-17;2]$
pour les deux jetons de C_{k3}: $[3;22]$

A la date 34, nous franchissons $o_{g2,1}$.

La ressource s_{r1} étant utilisée en permanence, il n'y a pas lieu de considérer sa marge temporelle comme critère de choix.

Si nous récapitulons nous avons la séquence suivante:
date 10: tir de $o_{g3,1}$ deux fois.
date 22: tir de $o_{g1,1}$ et de $o_{g2,1}$.
date 34: tir de $o_{g2,1}$.

Stratégie 2:
En appliquant le même type de raisonnement à la stratégie 2, nous obtenons la séquence suivante:

date 10: tir de $o_{g3,1}$
date 12: tir de $o_{g1,1}$
date 20: tir de $o_{g3,1}$
date 24: tir de $o_{g2,1}$
date 32: tir de $o_{g2,1}$

A la date 10, les marges temporelles agrégées sont:
pour le jeton de C_{k1}: $[3;22]$
pour les deux jetons de C_{k2}: $[-5;14]$
pour le premier jeton de C_{k3}: $[3;22]$
pour le deuxième jeton de C_{k3}: $[-7;12]$

A la date 12, les marges sont:
pour le jeton de C_{k1}: $[3;22]$
pour les deux jetons de C_{k2}: $[-7;12]$
pour le premier jeton de C_{k3}: $[3;22]$
pour le deuxième jeton de C_{k3}: $[-7;12]$

A la date 20, les marges sont:
pour le jeton de C_{k1}: $[3;22]$
pour le premier jeton de C_{k2}: $[-7;12]$
pour le deuxième jeton de C_{k2}: $[-15;4]$

pour le premier jeton de C_{k3}: $[3;22]$
pour le deuxième jeton de C_{k3}: $[-7;12]$
Par rapport à la situation à la date 22 correspondant à la stratégie 1, il n'y a pas de différence flagrante.

Il y a aussi les modifications suivantes sur les intervalles statiques:
place s_{k2}: $[0;15]_2 \rightarrow [0;20]_2$
place $s_{g2,4}$: $[0;12]_2 \rightarrow [0;7]_2$

Pour la deuxième stratégie, on termine le tir des cinq transitions à la date 32 alors que pour la première stratégie on termine à la date 34. On choisit donc pour ce conflit, la deuxième stratégie. Mais on note qu'il n'y a pas de différence flagrante entre ces deux solutions, c'est-à-dire que l'on peut être amené à effectuer un retour arrière ("backtrack") pour reprendre éventuellement la première stratégie.

En appliquant le même type de raisonnement chaque fois que l'on rencontre un conflit ressource, on obtient à la fin du cycle la séquence exprimée par le diagramme de Gantt de la figure 5.12.

La performance de la cellule est donnée par la valeur $\pi = 43$. Cette valeur est proche de la borne maximale $\pi_{max} = 47$. Nous en déduisons que pour cette politique de contrôle, la performance de la cellule n'est pas très bonne. De plus, lors du fonctionnement réel de la cellule, à la moindre perturbation (un retard de début d'opération par exemple), nous risquons de ne pas satisfaire les contraintes temporelles et donc de ne pas respecter les taux de production prévus. Afin d'avoir un fonctionnement à la fois plus robuste et plus efficace, nous devons donc modifier certains paramètres.
La transition de synchronisation t_s n'est franchie que lorsque l'état de fin de cycle (état initial dans ce cas) est détecté par le joueur et ce n'est qu'après le tir de t_s que nous pouvons relancer un nouveau cycle. Il semble donc que pour améliorer les performances de la cellule et rendre la séquence cyclique plus robuste aux aléas de la production, le plus judicieux serait de modifier l'état de fin de cycle.
Au démarrage de la cellule, le réseau sera toujours dans le même état, c'est-à-dire tous les jetons dans l'état indisponible pour le marquage:

$$M(c_{g1,1}) = M(s_{k1}) = M(s_{r2}) = M(s_{r3}) = M(s_{r5}) = 1$$
$$M(c_{g2,1}) = M(c_{g3,1}) = M(s_{k2}) = M(s_{k3}) = M(s_{r1}) = M(s_{r4}) = 2$$

puisque lors de la mise en marche du système nous n'aurons pas d'opérations en cours. Par contre, une fois la production lancée, il ne semble pas justifié d'attendre le retour

de toutes les étiquettes Kanban dans leur place Kanban pour relancer un nouveau cycle (franchir t_s). Puisque pour le régime périodique forcé optimiste les ressources du pool s_{r1} sont saturées, nous pouvons en nous basant sur la séquence calculée essayer de saturer au moins une des deux ressources de s_{r1} pour nous rapprocher du régime périodique forcé optimiste et améliorer la performance du système. Nous pouvons essayer de saturer la deuxième ressource du pool (la ressource s_{r12} sur le diagramme 5.12) puisque c'est la plus utilisée. Lorsque nous terminons l'opération $o_{g2,1}$ à la date 32 (troisième opération exécutée par la ressource s_{r12} du pool s_{r1}), l'état du réseau est alors au niveau des étiquettes Kanban:

$M(s_{k1}) = 1$ et étiquette indisponible,
$M(s_{k3}) = 1$ et étiquette indisponible,
$M(s_{g3,3}) = 1$ et étiquette dans l'état réservé depuis 7,
$M(s_{g2,4}) = 1$ et étiquette dans l'état réservé depuis 4,
$M(s_{g2,2}) = 1$ et étiquette dans l'état réservé depuis 0 (à cet instant précis, l'étiquette de cette place passe de l'état non réservé à l'état réservé).

C'est maintenant cet état qui doit être considéré par le joueur comme condition de franchissement de t_s. Comme les deux étiquettes Kanban de la contrainte C_{k2} sont réservées dans cet état, nous n'avons plus à considérer les intervalles statiques $[24;24]_3$ et $[24;24]_4$ associés à s_{k2}. De même pour C_{k3}, comme une des deux étiquettes est dans l'état réservé, nous n'avons plus à considérer l'intervalle statique de fin de cycle $[22;22]_4$.

En appliquant l'algorithme du joueur p-temporel t-temporisé pour la nouvelle condition de fin de cycle, nous obtenons le diagramme de Gantt de la figure 5.13.

La nouvelle performance de la cellule est donnée par la valeur $\pi = 32$. Cette valeur se trouve bien à l'intérieur de l'intervalle $[\pi_{min};\pi_{max}] = [28;47]$ et la solution est maintenant beaucoup plus robuste puisque la marge restante est beaucoup plus importante.

Au niveau du pilotage temps réel, nous chercherons donc à suivre cette séquence qui semble robuste et efficace et qui a été obtenue sans passer par des méthodes d'optimisation lourdes en calcul. Néanmoins, cette séquence doit seulement être vue comme une des séquences possibles. Le joueur temps réel, toujours en utilisant le même mécanisme de résolution de conflit, pourra choisir d'autres séquences parfois mieux adaptées et respectant toujours les contraintes temporelles lorsqu'apparaîtrons des perturbations non considérées par le joueur avec retour arrière. Si les perturbations sont plus importantes que prévu (c'est-à-dire si on a des violations de contraintes sur les intervalles statiques associés aux places) alors on pourra relacher certaines contraintes temporelles (i.e. augmenter la borne maximale des intervalles statiques

considérés) en utilisant la marge temporelle restante associée à l'invariant de places considéré. Si les perturbations sont si importantes que l'on a violation des marges temporelles associées aux invariants de places alors nous devrons passer en fonctionnement anormal en relachant complètement les contraintes temporelles (i.e. associer l'intervalle $[0; +\infty[$ à toutes les places), en ne considérant plus que les contraintes de gammes, les contraintes de ressources et la contrainte cyclique, et en appliquant une politique de tir au plus tôt. Comme même pour une politique de tir au plus tôt le régime sera toujours un régime périodique forcé (la transition t_s ne sera franchie que lorsque l'état de fin de cycle sera détecté), lorsque le diagnostic sera terminé et la décision de reprise envoyée au système par l'opérateur, nous pourrons très vite commuter en fonctionnement normal et nous n'aurons pas besoin par conséquent d'interrompre la production durant le diagnostic du système.

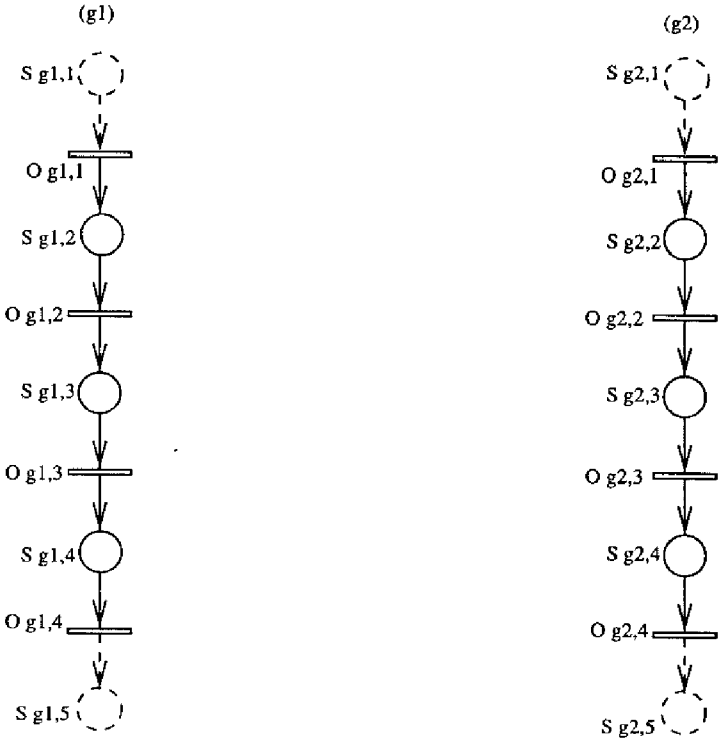

Figure 5.1: Gammes de Fabrication $g1$ et $g2$

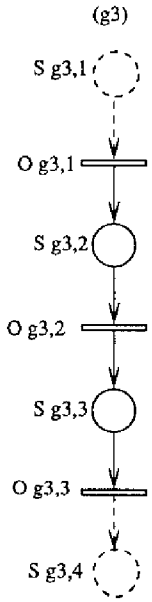

Figure 5.2: Gamme de Fabrication $g3$

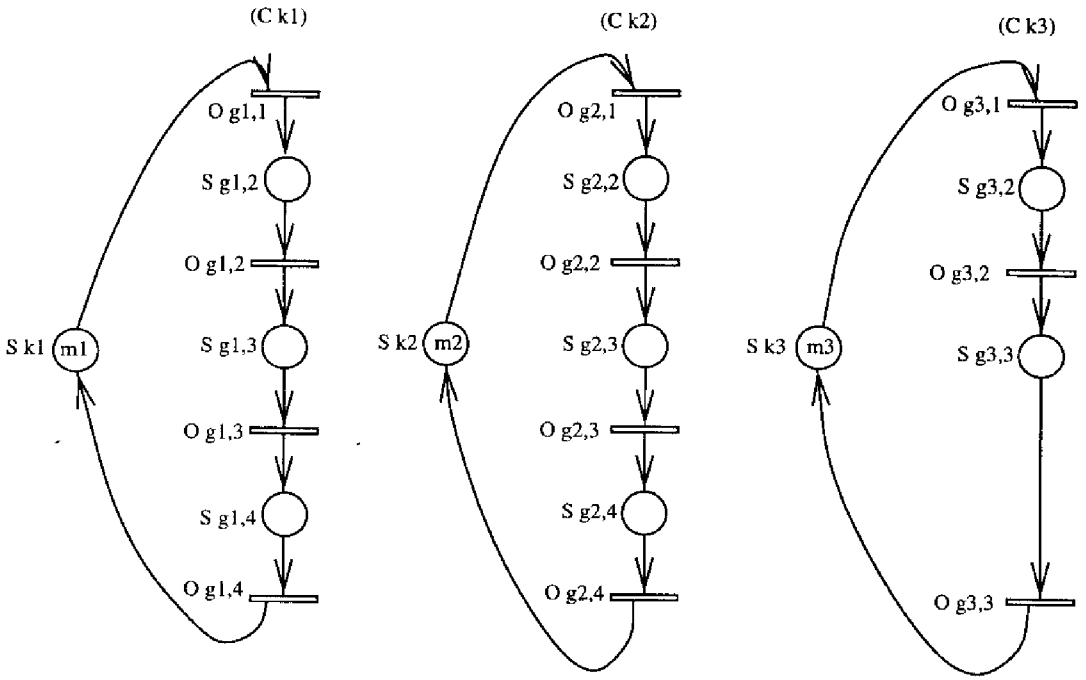

Figure 5.3: Contraintes Kanban C_{k1}, C_{k2} et C_{k3}

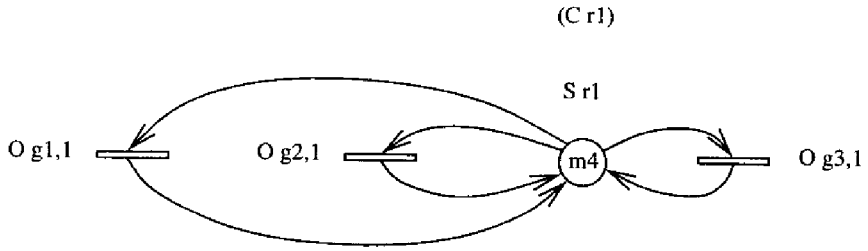

Figure 5.4: Contrainte de Ressource C_{r1}

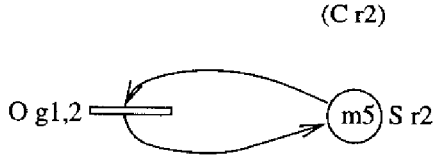

Figure 5.5: Contrainte de Ressource C_{r2}

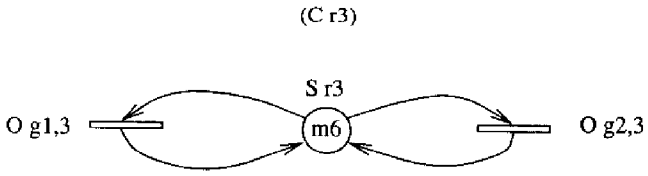

Figure 5.6: Contrainte de Ressource C_{r3}

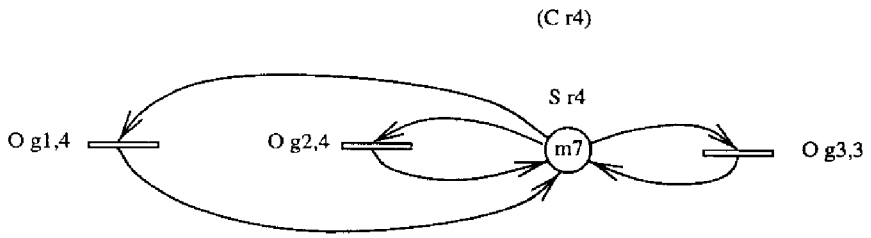

Figure 5.7: Contrainte de Ressource C_{r4}

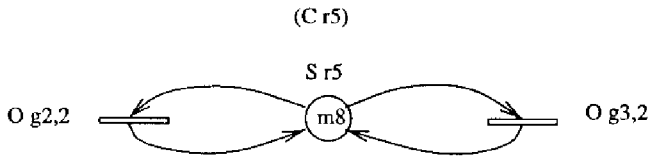

Figure 5.8: Contrainte de Ressource C_{r5}

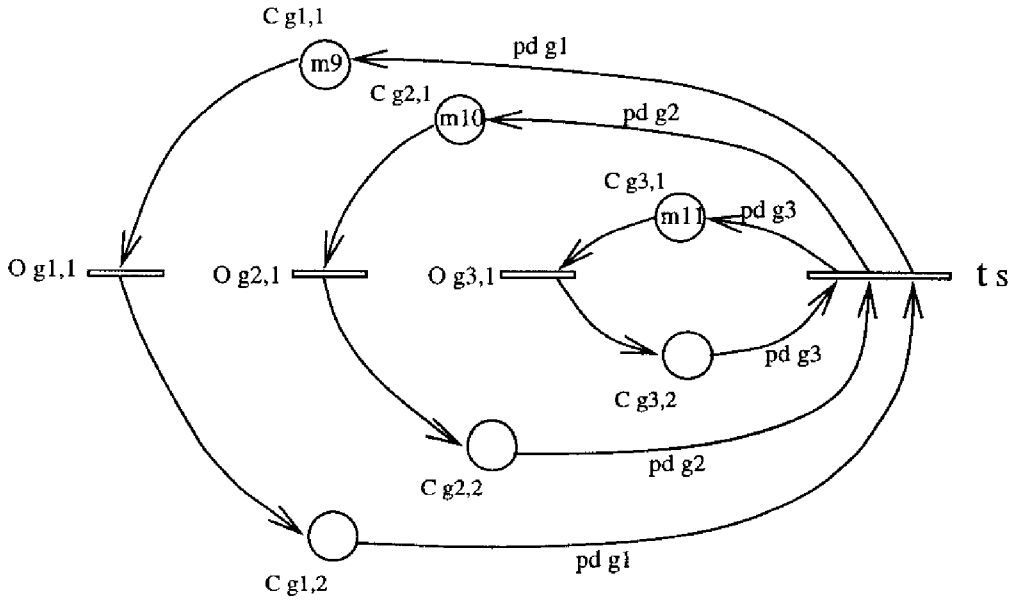

Figure 5.9: Contrainte Cyclique C_c

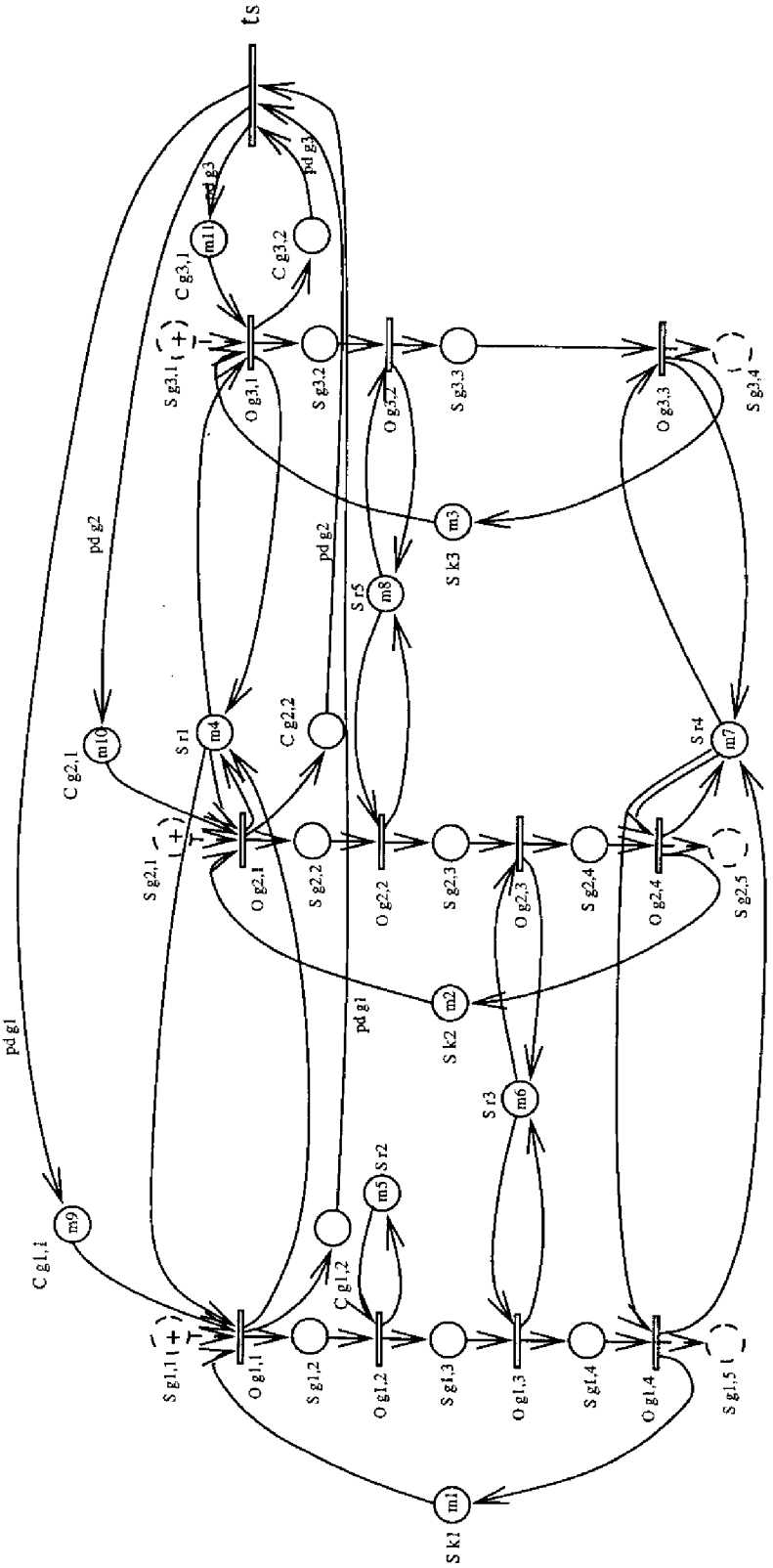

Figure 5.10: Modèle Global

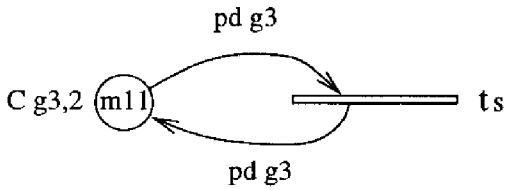

Figure 5.11: Réseau Réduit

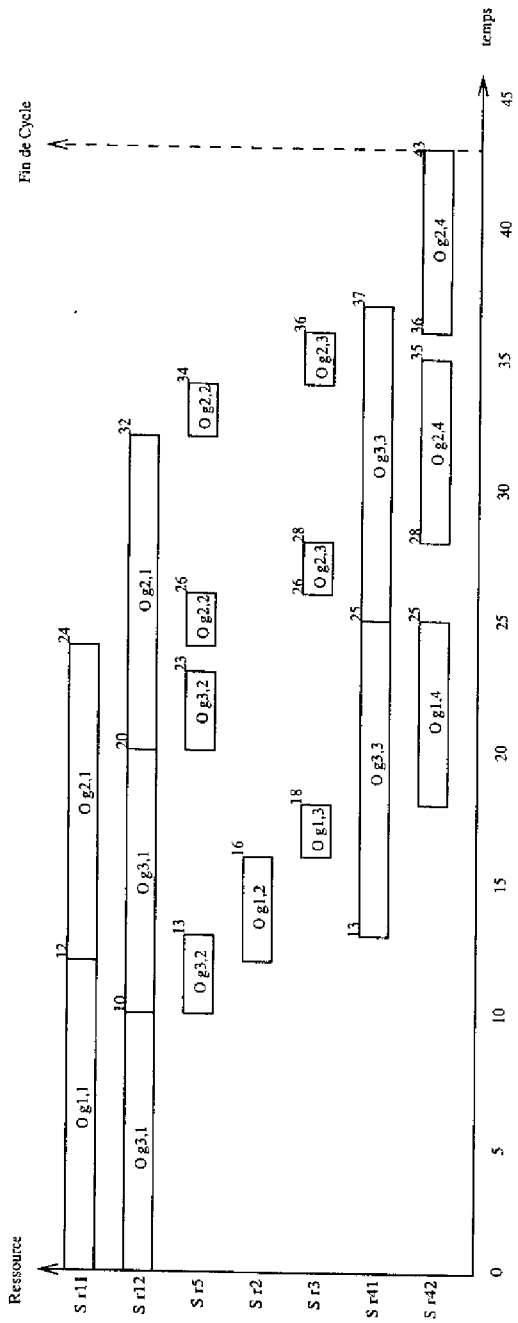

Figure 5.12: Séquence Admissible 1

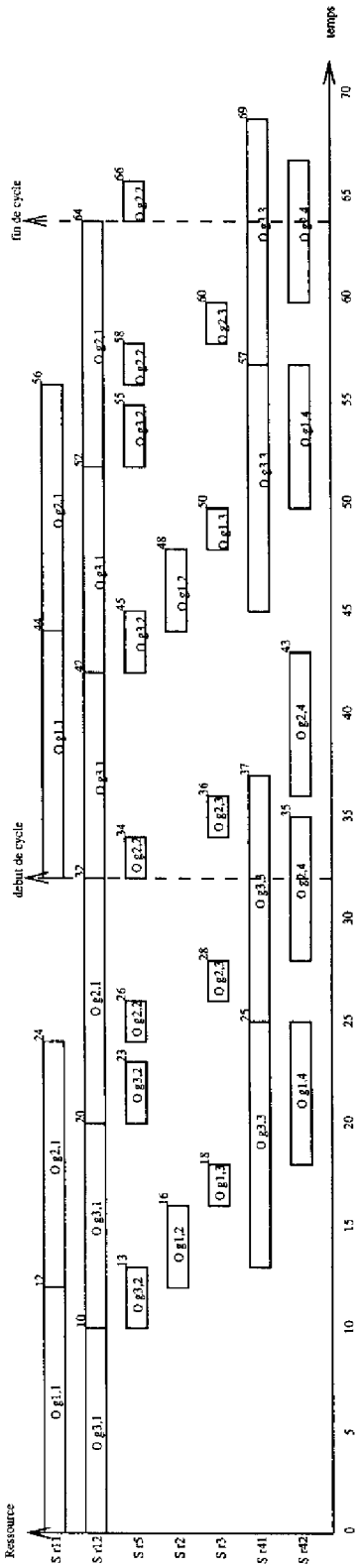

Figure 5.13: Séquence Admissible 2

Conclusion

Dans ce travail, nous avons proposé une méthodologie formelle de conception, d'analyse et de commande d'une cellule de fabrication flexible à fonctionnement cyclique. La particularité de l'approche est l'utilisation d'un même modèle (les réseaux de Petri p-temporels t-temporisés) à pratiquement toutes les phases du cycle de vie du système, ce qui garantit un développement cohérent et rigoureux.

Parmis les avantages qui existent par rapport aux approches plus conventionnelles, nous pouvons souligner:

- Le fait que notre approche puisse être vue comme l'analyse d'un système sous un ensemble de contraintes et non pas comme un simple problème d'optimisation.
- L'intérêt de la définition d'une nouvelle sémantique temporelle attachée aux places du modèle et qui rend possible une meilleure compréhension de l'évolution temporelle des systèmes flexibles et des stratégies de commande qui leurs sont appliquées. En particulier, elle permet de clairement différencier les paramètres du problème des variables de décision.
- La formalisation d'un ensemble d'indicateurs pour le choix d'un bon compromis entre efficacité et flexibilité: les bornes minimales et maximales des marges agrégées pour chaque contrainte Kanban et pour chaque contrainte de ressource (plus généralement chaque invariant de places).
- La traduction des décisions d'ordonnancement sous la forme des durées d'attente entre les opérations qui permet d'éviter la construction d'un graphe d'événements pour exprimer le plan de production et donc de conserver une bonne réactivité lors du pilotage en temps réel.

De plus, la méthode utilisée pour calculer l'ordonnancement aussi bien prévisionnel que temps réel est différente de celles proposées par d'autres auteurs essentiellement parce que notre modèle ne prend pas en compte seulement les durées d'opération; il considère aussi les durées d'attente associées aux places et délimitées par des intervalles. Ainsi, nous n'avons pas à franchir systématiquement les transitions au plus tôt et nous obtenons une méthode de résolution de conflit beaucoup plus riche puisqu'elle prend en compte un plus grand nombre de stratégies d'ordonnancement. La solution retenue doit présenter un bon compromis entre efficacité et robustesse, et comme l'ordonnancement, plutôt que d'être considéré comme une simple séquence d'opérations à suivre, s'exprime au travers d'un joueur (et donc d'un ensemble de règles) possédant

un puissant mécanisme de résolution, il autorise la réactivité indispensable à la prise en compte des aléas de la production lors du fonctionnement réel de la cellule.

Ce travail définissant une nouvelle approche, un certain nombre de ces aspects méritent sûrement d'être examinés plus en profondeur et offrent ainsi des pistes pour des travaux futurs.

Nous avons vu dans le chapitre 4 que pour la désagrégation des contraintes temporelles (la définition des intervalles statiques associés aux places), il est nécessaire de faire appel à des heuristiques qui dépendent de l'exemple considéré. De même, pour la définition de la borne maximale dans le chapitre 3, il n'y a aucune règle systématique à appliquer. C'est l'opérateur chargé du bon fonctionnement du système qui en fonction de l'expérience acquise doit fixer bon nombre de paramètres (durées d'attente minimales globales associées aux places de chacun des invariants de places, durées minimales et maximales des intervalles statiques associés aux places etc.). Il doit être possible de formaliser et d'analyser dans une certaine mesure la sémantique des divers indicateurs de flexibilité utilisés pour résoudre les conflits. Par exemple, les quantités de ressources (m_{s_r} dans les équations) et la taille des en-cours (m_{s_k} dans les équations) peuvent être vues comme des variables de décision dont l'influence est facilement analysable du point de vue de l'efficacité (valeur de la période π), de la flexibilité (marges temporelles données par les variables d_{C_k}) et de la criticité des ressources (marges temporelles données par les variables d_{C_r}), d_{C_k} étant un indicateur de l'utilisation des en-cours et d_{C_r} de la criticité des ressources. Il sera sûrement intéressant en étudiant des cas réels de cellules flexibles de voir s'il n'est pas possible en se basant sur ces indicateurs de définir des règles plus précises et plus systématiques de désagrégation des contraintes. Il pourra être intéressant aussi de développer une interface suffisamment ergonomique pour faire le lien entre l'opérateur et le système d'aide à la décision.

Un autre point à approfondir au niveau du pilotage de la cellule est le passage d'un fonctionnement cyclique prévisible et normal à un fonctionnement perturbé anormal. Il sera intéressant ici de reprendre la contrainte cyclique présentée initialement par *Ohl* dans [OH 94] et d'introduire dans les places $c_{gi,j}$ un nombre de jetons supérieur au poids pd_{gi} ($M_c(c_{gi,j}) > pd_{gi}$) qui fixe le nombre de pièces à traiter par cycle. Il sera ainsi possible de sortir dans une certaine mesure des taux de production fixés lors de régimes perturbés pour lesquels une politique strictement cyclique ne correspond pas à la meilleure des solutions. Dans le cas de régimes très perturbés (une machine qui casse et qui rend une gamme de fabrication inutilisable par exemple), il sera très certainement indispensable d'introduire la notion de jetons négatifs qui nous permettra de passer d'un cycle à l'autre (i.e. de franchir la transition de synchronisation t_s) même lorsqu'une des gammes de la cellule sera rendue inutilisable par l'arrêt d'une machine pendant plusieurs périodes consécutives et que les étiquettes Kanban n'arriveront plus dans leurs places de fin de cycle.

Enfin, tous les liens et similitudes qui existent entre le modèle utilisé et les formalismes de contraintes devraient nous permettre d'implémenter l'approche en utilisant les langages de programmation de contraintes. Il sera ainsi possible de directement tirer parti des techniques de résolution de contraintes de ces langages. La mise en oeuvre du pilotage par l'intermédiaire du joueur temps réel appliqué au cas d'un sys-

tème réel ou au moins en passant par l'utilisation d'un simulateur de réseaux de Petri sera aussi sûrement d'un grand intérêt pour vérifier l'efficacité de l'approche proposée dans ce mémoire.

Bibliographie

[AB 96] I. B. Abdallah: Méthodes d'allocation de ressources dans les systèmes flexibles de production manufacturière fondées sur l'analyse structurelle des réseaux de Petri, Thèse de Doctorat de l'Ecole Centrale de Paris, 4 juillet 1996.

[AK 93] M. Akaza, D. Lee, S. Kumagai, S. Kodama: Application of timed marked graphs to a scheduling problem of production systems including repetitive processes with set-up times, *modern Tools for Manufacturing Systems*, R. Zurawski and T.S. Dillon (Editors), 1993.

[AL 85] M. Alam, D. Gupta, S.I. Ahmad, A. Raouf: Performance modeling and evaluation of flexible manufacturing systems using semi-Markov approach, Manufacturing research and technology 1, Flexible Manufacturing, Recent Developments in FMS, Robotics, CAD/CAM, CIM, Elsevier, p.87-118, 1985.

[AT1 87] H. Atabakhche: Utilisation conjointe de l'intelligence artificielle et des réseaux de Petri: Application au contrôle d'exécution d'un plan de fabrication, Thèse de Doctorat, Université Paul Sabatier, Toulouse, Décembre 1987.

[AT2 87] H. Atabakhche, D. Simonetti-Barbalho, R. Valette, M. Courvoisier: Commande d'ateliers: un compromis est-il possible entre une approche graphique et une approche intelligence artificielle ?, *APII*, p.377-394, 1987.

[AU 94] C. Ausfelder, E. Castelain, JC. Gentina: A Method for Hierarchical Modeling of the Command of Flexible Manufacturing Systems, *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 24, N0. 4, April 1994.

[AZ 94] D. Azzopardi, S. Lloyd: Reduction of search space for scheduling of multi-product batch process plant through Petri net modelling, *2nd IFAC/IFIP/IFURS Workshop on Intelligent Manufacturing Systems*, Vienna, Austria, June 1994.

[BA 90] B. Bako: Mise en oeuvre et simulation du niveau coordination de la commande des ateliers flexibles: une approche mixte réseaux de Petri et systèmes de règles, Thèse de Doctorat, Université Paul Sabatier, Toulouse, Octobre 1990.

[BI 93] J.C. Billaut: Prise en compte des ressources multiples et des temps de préparation dans les problèmes d'ordonnancement en temps réel, Thèse de Doctorat de l'Université Paul Sabatier, Toulouse, décembre 1993.

[BR 83] G.W. Brams: Réseaux de Petri: Théorie et Pratique, Masson, 1983.

[BR 94] C. Briand: Conception orientée-objet et basée réseaux de Petri de la conduite temps réel des systèmes flexibles de production manufacturière, Thèse de Doctorat de l'Université Paul Sabatier, novembre 1994.

[BR 89] C. Brown: Relating Petri Nets to Formulas of Linear Logic, Edinburg Tech., Report ECS-LFCS-89-87, June 1989.

[CA 96] H. Camus, H. Ohl, O. Korbaa, J.C. Gentina: Cyclic schedules in flexible manufacturing systems with flexibilities in operating sequences, Manufacturing and Petri nets, First International Workshop, Osaka, Japan, June 1996.

[CA 97] H. Camus: Conduite de systèmes flexibles de production manufacturière par composition de régimes permanents cycliques: modélisation et évaluation de performances à l'aide des réseaux de Petri, Thèse de doctorat de l'Université des sciences et technologies de Lille, mars 1997.

[CA 87] J. Carlier, P. Chrétienne: Problèmes d'ordonnancement: modélisation/complexité/algorithmes, Masson, 1987.

[CO 91] M. Combacau: Commande et surveillance des systèmes à événements discrets complexes: application aux ateliers flexibles, Thèse de Doctorat de l'Université Paul Sabatier, Toulouse, décembre 1991.

[CO 86] M. Courvoisier et R. Valette: Commande des procédés discontinus- Logique séquentielle, Dunod Université, 1986.

[CO 89] M. Courvoisier, R. Valette, A. Sahraoui, M. Combacau: Specification and implementation techniques for multilevel control of FMS, *CAPE'89*, p.509-516, Tokyo, Japan, october 1989.

[CO 93] Cosytec, CHIP V4.0 reference manual, 1993.

[COH 89] G. Cohen, P. Moller, J. Quadrat, M. Viot: Algebraic Tools for the Performance Evaluation of Discrete Event Systems, *Proceedings of the IEEE*, vol.77, n°1, p.39-57, January 1989.

[DA 92] R. David and H. Alla: Petri Nets and Grafcet, Prentice Hall, 1992.

[DU 83] D. Dubois, K. E. Stecke: Using Petri nets to represent production processes, *Proceedings of the 22ND IEEE conference on decision and control*, December 1983.

[EL 93] S. Elkhattabi: Intégration de la surveillance de bas niveau dans la conception des systèmes à événements discrets: application aux systèmes de production flexibles, Thèse de Doctorat de l'Université de Lille, 1993.

[ER 76] J. Erschler: Analyse sous contraintes et aide à la décision pour certains problèmes d'ordonnancement, Thèse de doctorat d'état, Université Paul Sabatier, Toulouse, 1976.

[ER 86] J. Erschler, P. Esquirol: Decision aid in job-shop scheduling: a knowledge based approach, *IEEE Int. Conf. on Robotics and Automation*, San Francisco, p.1651-1656, 1986.

[ER 88] J. Erschler, G.D. Terssac: Flexibilité et rôle de l'opérateur humain dans l'automatisation intégrée de production, Rapport LAAS n^o 88137, Mars 1988.

[ES 87] P. Esquirol: Règles et processus d'inférence pour l'aide à l'ordonnancement de tâches en présence de contraintes, Thèse de l'Université Paul Sabatier (Toulouse-France), Décembre 1987.

[ES 94] P. Esquirol, P. Lopez, L. Haudot, M. Sicard: Un système coopératif en ordonnancement de production, AGI'94, Colloque Automatique-Génie informatique-Image, Poitiers, juin 1994.

[ES 95] P. Esquirol, P. Lopez, H. Fargier, T. Schiex: Constraint Programming, LAAS report 95209, June 1995.

[EZ 93] J. Ezpeleta, J. Martinez: Synthesis of live models for a class of FMS, *Proceedings of the 1993 Int. Conf. on Robotics and Automation*, Atlanta, May 2-7 1993, p.557-

[FA 94] H. Fargier: Problèmes de satisfaction de contraintes flexibles: application à l'ordonnancement de production, Thèse de l'Université Paul Sabatier (Toulouse-France), Juin 1994.

[FL 84] G. Florin, S. Natkin: Définition formelle des réseaux de Petri stochastiques, Research Report CNAM, Paris, 1984.

[FO 82] C. Forgy: RETE: a fast algorithm for many pattern/many object pattern match problem, *Artificial Intelligence*, N° 19, p.17-37, 1982.

[FO 87] M. Fox: Constraint-Directed Search: A case study of job-shop scheduling, *Research Notes in Artificial Intelligence*, Pitman Publishing, 1987.

[GA 89] H. Garnousset, J.M. Farines, E. Cantú: Efficient Tools for Analysis and Implementation of Manufacturing Systems Modelled by Petri nets with Objects: A production Rules Compilation-based Approach, *IECON'89 Fiftennth Annual Conference of the IEEE Industrial Electronics Society*, Philadelphia, November 1989.

[GE 87] H.J. Genrich: Predicate/transition nets, *Lectures Notes in Computer Science*, Springer Verlag, 1987.

[GI 87] J.Y. Girard: Linear Logic: *Theoretical Computer Science*, 50, 1987.

[GU 89] H.W. Güsgen: CONSAT: a system for Constraint Satisfaction, *Research Notes in Artificial Intelligence*, Pitman Publishing, 1989.

[GUN 89] C. Gunter, V. Gehlot: Nets as tensor theories, *10th International Conference on Application and Theory of Petri nets*, Bonn, Germany, 1989.

[HI 88] H. Hillion, J.M. Proth: Analyse de fabrications non linéaires et répétitives à l'aide de graphes d'événements temporisés, Recherche Opérationnelle, vol. 22, n° 2, p.137-176, 1988.

[HI 89] H. Hillion, J.M. Proth: Performance evaluation of job-shop systems using timed event-graphs, *IEEE Trans. on Automatic Control*, Vol 34, N° 1, p.3-9, January 1989.

[JE 90] K. Jensen: Coloured Petri Nets: A High Level Language for System Design and Analysis, G. Rozenberg (ed.): Advances in Petri nets 1990, *Lecture Notes in Computer Science*, vol. 483, Springer, Berlin Heidelberg New York 1990, pp. 342-416.

[JU 94] S. Julia, R. Valette, M. Tazza: Analysis of the behavior of a manufacturing cell with cyclic feeding policies, *IEEE International Conference on Systems, Man and Cybernetics* , San Antonio, Texas, USA, October 1994.

[JU_1 95] S. Julia, R. Valette, M. Tazza: Computing a feasible schedule under a set of cyclical constraints, *Second International Conference on Industrial Automation*, Nancy, France, June 1995.

[JU_2 95] S. Julia, R. Valette, M. Tazza: Analysis of a manufacturing cell under a set of cyclic constraints, *38TH Midwest Symposium on Circuits and Systems, IEEE Circuits and Systems Society*, Rio de Janeiro, Brazil, August 1995, p. 23-26.

[JU_3 95] S. Julia, R. Valette, R.M. da Silva Julia: Análise sob restrições baseadas em Redes de Petri de uma Célula Flexível de Manufatura, *2º SBAI*, Curitiba, Brasil, 1995.

[KA 85] G. Kallel, X. Pellet, Z. Binder, Conduite décentralisée coordonnée d'atelier, RAIRO APII, vol. 19, 1985, p. 371-387.

[KH 96] W. Khansa, P. Aygalinc, J.P. Denat: Structural analysis of p-time Petri nets, Symposium on discrete events and manufacturing systems, *CESA'96 IMACS Multiconference*, Lille-France, July 1996.

[KU 90] A. Kusiak: Intelligent Manufacturing Systems, Prentice Hall International series in industrial and systems Engineering, 1990.

[KU 94] Andrew Kusiak and Weihua He: Design of components for schedulability, *European Journal of Operational Research*, Vol 76, p. 49-59, 1994.

[LE 88] W. Leler: Constraint Programming Languages: Their Specification and Generation, Addison-Wesley Publishing Company,

[LE1 94] D.Y. Lee, F. DiCesare: Scheduling Flexible Manufacturing Systems using Petri nets and heuristic search, *IEEE Transactions on robotics and automation*, vol. 10, No. 2, p. 123-132, April 1994.

[LE2 94] D.Y. Lee, F. DiCesare: Integrated scheduling of flexible manufacturing systems employing automated guided vehicles, *IEEE Transactions on industrial electronics*, vol. 41, No. 6, p. 602-610, December 1994.

[LO 91] P. Lopez: Approche énergétique pour l'ordonnancement de tâches sous contraintes de temps et de ressources, Thèse de Doctorat, Université Paul Sabatier, Toulouse, Septembre 91.

[LO 93] J. Long: Sur la conduite hiérarchisée des systèmes flexibles de production, Thèse de doctorat de l'Institut National Polytechnique de Grenoble, juin 1993.

[MA 81] J. Martínez, M. Silva: A simple and fast algorithm to obtain all invariants of a generalised Petri net, *Second European Workshop on Application and Theory of Petri nets*, p.301-308, 1981.

[MA 85] J. Magott: Performance evaluation of systems of cyclic sequential processes with mutual exclusion using Petri nets, *Information Processing Letters 21*, p. 229-232, November 1985.

[MA 91] M. Mascolo, Y. Frein, Y. Dallery, R. David: A unified modeling of Kanban Systems using Petri nets, *The international journal of Flexible Manufacturing Systems*, vol.3, p.275-307, 1991.

[ME 74] P. Merlin: A study of recoverability of computer systems, PhD thesis, University of California, Irvine, 1974.

[MI 83] M.Minoux: Programmation Mathématique: théorie et algorithme, Dunod, Août 1983.

[MU 88] T. Murata, D. Zhang: A predicate-transition net model for parallel interpretation of logic programs, *IEEE Transaction on Software Engineering*, vol. 14, N°1, April 1988.

[MU 89] T. Murata: Petri Nets: Properties, Analysis and Applications, *Proceedings of the IEEE*, Vol. 77, N° 4, p. 541-580, April 1989.

[NA 86] Y. Narahari, N. Viswanadham: On the Invariants of coloured Petri nets Advances in Petri nets 1985, *Lecture notes in computer science 222*, Springer Verlag, p.330-345, 1986.

[NE 88] G.L. Nemhauser, L.A. Wolsey: Integer and Combinatorial Optimization, WILEY interscience series in discrete mathematics and optimization, 1988.

[OH 94] H. Ohl, E. Castelain, J.C. Gentina: Synchrony Theory applied to Control Problems in Flexible Manufacturing Systems, *IEEE International Conference on Systems, Man and Cybernetics*, San Antonio, Texas, USA, October 1994.

[OH 95] H. Ohl: Fonctionnements répétitifs de systèmes flexibles de production manufacturière: Analyse et Optimisation des performances à l'aide des réseaux de Petri, Thèse de Doctorat de l'Université de Lille, septembre 1995.

[PE 89] G. Peterka, T. Murata: Proof procedure and answer extraction in Petri net model of logic programs, *IEEE Transaction on Software Engineering*, vol.15, N°2, February 1989.

[PE 81] J.L. Peterson: Petri net theory and the modeling of systems, Prentice-Hall, 1981.

[PE 66] C.A. Petri: Communication with Automata, Technical Report RADC-TR-65-377, 1966.

[RA 74] C. Ramchandani: Analysis of asynchronous coucurrent systems by times Petri net models, Phd. Thesis, MIT, Project MAC TR-120, 1974.

[RA 80] C.V. Ramamoorthy, G.S. Ho: Performance evaluation of asynchronous concurrent systems using Petri nets, *IEEE Trans. on Soft. Eng.*, Vol. SE-6, N° 5, p. 440-449, September 1980.

[RE 85] W. Reisig: Petri Nets: An introduction, Springer-Verlag, 1985.

[RE 90] C. Reutenauer: The Mathematics of Petri nets, Masson and Prentice Hall International, 1990.

[RI 95] P. Richard, J.L. Bouquard, C. Proust: Ordonnancement basé sur les réseaux de Petri et une résolution en CHIP, *2e Int. Conf. on Manufacturing Automation*, (Nancy) 1995.

[RO 93] M. Roboam: La méthode GRAI: principes, outils, démarches et pratiques, Editions Teknea, août 1993.

[SA 87] A.K. Sahraoui: Contribution à la commande et à la surveillance d'ateliers flexibles, Thèse de doctorat, Université Paul Sabatier, Toulouse, Octobre 1987.

[SB 94] C. Sibertin-Blanc: Cooperative nets, *15th International Conference on Application and Theory of Petri Nets, Lecture Notes in Computer Science 815*, p. 471-490, Springer Verlag, 1994.

[SI 77] J. Sifakis: Use of Petri nets for performance evaluation, *3'd International Symposium on modeling and performance evaluation of computer systems*, H. Beilner and E. Gelenbe Eds., Noorth-Holland, 1977.

[SI 89] M. Silva, R. Valette: Petri Nets and Flexible Manufacturing, *Advances in Petri Nets*, G. Rozenberg (ed.), *Lecture Notes of Computer Science*, Springer Verlag, p. 374-417, 1989.

[SM 88] S.F. Smith: A constraint-based framework for reactive management of factory schedule, Intelligent Manufacturing: Proceedings from the first international conference on expert systems and the leading edge in production planning and control, M.D. Oliff editor, The Benjamin/Cummings Publishing Company, 1988.

[SV 94] V.M. Savi: Conception préliminaire des systèmes de production à l'aide des réseaux de Petri: Evaluation des performances, Thèse de Doctorat, Université de Metz, Metz, Mai 1994.

[TA 92] C. Tahon: Systèmes d'aide à la décision pour la conduite des systèmes de production, Habilitation à diriger des travaux de recherche de l'Université de Valenciennes, Janvier 1992.

[TA] M. Tazza, L.A. Kunzle, P.C. Stadzisz: Sistemas Flexíveis de Manufatura: Projeto e Análise, *9° CBA*, UFES-Vitória/ES-Brasil, p.62-81.

[TA 87] M. Tazza: Quantitative analysis of a resourse allocation problem: a net theory based proposal, *Concurrency and nets*, Springer Verlag, p. 511-532, 1987.

[TO 92] A.K.A. Toguyeni: Surveillance et diagnostic en ligne dans les systèmes flexibles de l'industrie manufacturière, Thèse de Doctorat de l'Université de Lille, novembre 1992.

[VA 82] R. Valette, M. Courvoisier, D. Mayeux: Control of flexible production systems and Petri nets, *Application and Theory of Petri nets*, Informatik-Fachberichte, N° 66, Springer Verlag, p. 264-277, 1982.

[VA 85] R. Valette, M. Courvoisier, H. Demmou, JM. Bigou, C. Desclaux: Putting Petri nets to work for controlling flexible manufacturing systems *1985 IEEE ISCAS*, Kyoto Japon, juin 1985, p. 929-932.

[VA 88] R. Valette: Coordination problems in FMS control systems, Coordination Management by Means of Petri Nets, One-Day Workshop, Modena-Italy, April 1988.

[VA 91] R. Valette, B. Bako: *Software Implementation of Petri nets and Compilation of Rule-based Systems* in Lecture Notes in Computer Science "Advances in Petri nets 1991", Vol.524, Ed. G. Rozenberg, Springer Verlag, 1991, pp.296-316.

[VA 93] R. Valette, M. Courvoisier: Petri nets and Artificial Intelligence, International Workshop on Emerging Technologies for Factory Automation p.218-238, North Queensland, Australia, August 1992, Modern Tools for Manufacturing Systems, Manufacturing Research and Technology, Elsevier, vol.18, p.385-405, 1993.

[VI 94] N. Viswanadham, R. Ram: Composite Performance-Dependability Analysis of Cellular Manufacturing Systems, *IEEE Transactions on Robotics and Automation*, vol. 10, n° 2, p.245-258, 1994.

[YI 95] P. Yim, A. Lefort, A. Hébrard: System modelling with hypernets, ETFA 95, Symposium on Emerging Technologies and Factory Automation, Paris, France, October 1995.

[ZH 92] M. Zhou, F. Dicesare, D.L. Rudolph: Design and implementation of a Petri net based supervisor for a flexible manufacturing system, Automatica, vol. 28, no 6, p. 1199-1208, 1992.

Liste des figures

Thèse de Stéphane JULIA

Conception et pilotage de cellules flexibles à fonctionnement répétitif modélisées par réseaux de Petri

Ce travail a pour objet la présentation d'une approche pour l'aide au dimensionnement, l'aide à l'élaboration de politiques de conduite et le pilotage en temps réel de cellules flexibles de fabrication.

L'approche que nous proposons est articulée autour de la modélisation et de l'analyse d'une cellule flexible sous un ensemble de contraintes cycliques.

Le modèle choisi est un réseau de Petri p-temporel t-temporisé plus général qu'un graphe d'événements. Il permet d'une part de découpler les contraintes de gammes des contraintes de ressources, et d'autre part de modéliser aussi bien les durées d'opération associées aux transitions qui sont les paramètres de notre problème, que les durées d'attente des pièces dans les stocks intermédiaires représentés par les places qui sont elles les inconnues du problème.

Après avoir défini un régime stationnaire périodique forcé, des bornes minimales et maximales qui correspondent à des conditions nécessaires de réalisation d'une production plus ou moins flexible sont calculées à l'aide des techniques de la programmation mathématique.

Ces bornes sont utilisées au niveau d'un algorithme de joueur de réseau de Petri p-temporel t-temporisé possédant un mécanisme de retour arrière ("backtrack") afin de calculer une stratégie particulière de réalisation de la production, l'objectif essentiel étant de réaliser un bon compromis entre efficacité et flexibilité.

Finalement, en nous basant sur le résultat de l'ordonnancement prévisionnel calculé, le modèle de réseau de Petri est utilisé pour un pilotage temps réel et réactif de la cellule au travers de l'utilisation d'un joueur temps réel ne possédant plus de mécanisme de retour arrière et permettant de décrire en temps réel l'enchaînement des traitements à effectuer sur les pièces, de communiquer avec l'environnement extérieur et d'être soumis à des contraintes temporelles explicites.

Un exemple d'application illustre l'approche exposée.

Mots Clés: cellule flexible, réseaux de Petri temporels/temporisés, analyse sous contrainte, évaluation de performances, ordonnancement cyclique, pilotage temps réel.

Design and real time control of flexible cells with a cyclic behavior based on a Petri net model

The objective of this work is the development of an approach for aiding the shop sizing, the production management and the real time control of flexible manufacturing cells.

The approach is about the modelization and the analysis of a flexible cell under a set of cyclic constraints.

The selected model is a p-time t-timed Petri net which is more general than an event graph. It allows on the one hand to separate the production routes constraints and the resource allocation constraints and on the other hand to modelize the operation durations associated with the transitions which are the parameters of our problem and the wait durations of parts in intermediary buffers represented by the places which are the unknown of the problem.

After determining a cyclic feeding policy, minimal and maximal bounds which correspond to necessary conditions of realisation of a more or less flexible production are computed by the aid of mathematical programming.

These bounds are used at level of a p-time t-timed Petri net token player algorithm including a backtrack mechanism in order to compute a particular strategy of production realisation, the essential objective being to realize a good trade-off between flexibility and efficiency.

Finally, starting from a computed previsional schedule, the Petri net model is used for the reactive real time control of the cell through the utilisation of a real time token player without any backtrack mechanism and allowing to describe in real time the control sequences of operations carried out on the parts, to communicate with the external environment and to meet the explicit temporal constraints.

An example of application explains the approach.

Keywords: flexible cell, p-time t-timed Petri net, constraint analysis, performance evaluation, cyclic schedule, real time control.