Gestion hiérarchisée de systèmes de production discrets : une approche basée sur les réseaux de Petri

Liming Wang

d'ordre

THÈSE

présentée à

L'UNIVERSITÉ DE METZ
FACULTE DES SCIENCES
FR MATHÉMATIQUES, INFORMATIQUE, MÉCANIQUE

pour l'obtention du titre de

DOCTEUR

Spécialité :

SCIENCES DE L'INGÉNIEUR
(Mention : AUTOMATIQUE)

par

Liming WANG

Sujet de la thèse :

GESTION HIÉRARCHISÉE DE SYSTÈMES DE PRODUCTION DISCRETS: UNE APPROCHE BASÉE SUR LES RÉSEAUX DE PETRI

Soutenue le 1995 devant le jury composé de :

M.	Claude LAURENT	Rapporteurs
M.	Bernard MUTEL	
M.	François VERNADAT	
Mlle	Marie-Claude PORTMANN	Examinateurs

Gestion Hiérarchisée de Systèmes de Production Discrets: Une Approche Basée sur Les Réseaux de Petri

Liming WANG

Table des matières

Table des figures

Liste des tableaux

Avant-propos

Arrière-plan de la Thèse

Dans un appel aux communications d'un congrès international (*IJCAI'95: Intelligent Manufacturing Systems*) sur les systèmes de production, on peut lire:

> Global competition and rapid technological advances in communication, computing, and flexible machinery are bringing about unprecedented changes in manufacturing and management practices. The global manufacturing company of the future will have to be *lean, customer-driven, environment-conscious.* It will have to be capable of rapidly adapting its products, processes and alliances in reaction to changes in market demands, technologies, raw material availabilities, legislations, etc.. As companies strive to attain these elusive objectives, they are turning to radically new manufacturing philosophies and concepts, e.g., Lean Manufacturing, Agile Manufacturing, Virtual Manufacturing, Holonic Manufacturing, etc.. It is expected that Artificial Intelligence, Operations Research, Control Theory, and Information Technology will play a key role in delivering the decision support tools required to turn these visionary concepts and philosophies into practice.

Ce résumé des tendances montre bien que la complexité des systèmes de production va en croissant, et donc qu'une modélisation minutieuse prend tous les jours plus d'importance. Cela justifie les travaux que nous présentons ici.

Cette thèse a été préparée au sein du projet SAGEP de l'INRIA-Lorraine sous la direction scientifique de Monsieur Jean-Marie PROTH, Directeur de Recherche à l'INRIA.

Le projet SAGEP a pour objectif l'étude des problèmes induits par le contrôle des systèmes dynamiques, et en particulier des systèmes à événements discrets. Le domaine d'application privilégié de nos travaux est celui des systèmes de production industriels.

Contribution de la Thèse

La gestion de la production aboutit à l'ordonnancement. L'ordonnancement consiste à affecter les tâches aux ressources, et à décider les dates auxquelles ces tâches vont débuter, en tenant compte des contraintes inhérentes à la production manufacturière. Un tel problème est généralement NP-difficile, ce qui interdit de le résoudre sur un horizon suffisant pour prendre en compte les fluctuations de la demande. Un moyen possible pour contourner cette difficulté est d'utiliser une approche hiérarchique et de faire évoluer le système en suivant le principe des plans glissants.

Nous verrons, dans le chapitre 2, qu'une abondante littérature existe sur ce sujet. Cependant, une approche systématique de la conception d'une hiérarchie reste à trouver. Il nous semble que les réseaux de Petri, et plus particulièrement un type de réseaux de Petri appelé réseaux de Petri à sorties contrôlables (CO nets en anglais; CO signifie "Controllable-Output") sont particulièrement efficaces pour nous accompagner dans cette démarche.

Dans notre travail, nous présentons d'abord un exemple simple de hiérarchie à deux niveaux. Il permet d'illustrer les avantages et les inconvénients d'une approche hiérarchique.

Nous passons ensuite à une étude approfondie des "CO nets", et nous montrons que les réseaux CO constituent un outil satisfaisant pour notre approche, laquelle est à la fois modulaire et hiérarchique. Nous développons une méthode de simplification des modules et montrons comment intégrer les modules simplifiés pour conserver les propriétés qualitatives souhaitées.

Enfin, nous établissons une démarche descendante qui consiste à résoudre le modèle global simplifié sur un horizon suffisamment grand, puis à se servir de la solution obtenue comme contrainte à satisfaire par les modules détaillés du niveau bas sur un horizon réduit.

Ce travail peut être considéré comme une tentative de rationalisation de la gestion hiérarchisée.

Plan de la Thèse

Cette thèse est organisée comme suit:

- Le chapitre 1 est consacré à une présentation générale des réseaux de Petri et de la gestion de production. Les notions de base, ainsi que les approches d'analyse et différentes classes de réseaux de Petri, sont présentées. Les bases de la gestion des systèmes de production discrets sont rappelées dans ce même chapitre.

- Dans le chapitre 2, nous comparons l'approche hiérarchique et l'approche monolithique pour la gestion de production à l'aide d'un exemple simple

mais significatif. Cette étude comparative permet de justifier l'utilisation de l'approche hiérarchique pour la gestion de systèmes complexes. Une approche hiérarchique à deux niveaux pour la planification de la production est également présentée pour illustrer le concept.

- Dans le chapitre 3, nous proposons la notion de réseaux de Petri à sorties contrôlables (réseaux CO) et étudions les propriétés des réseaux CO. Nous étudions aussi la réduction/simplification d'un réseau de Petri. Nous proposons quelques algorithmes pour atteindre cet objectif et, en particulier, nous proposons un algorithme qui permet de traiter les réseaux de Petri généraux.

- Dans le chapitre 4, nous étudions la modélisation modulaire et la gestion hiérarchique des systèmes de production discrets à l'aide des réseaux de Petri. Nous présentons aussi, dans ce chapitre, l'architecture du logiciel HMPS (en anglais: *Hierarchical and Modular approach to production Planning and Scheduling*) développé au sein de l'équipe SAGEP de l'INRIA-Lorraine.

- Dans le chapitre 5, nous évaluons l'approche que nous avons proposée. Selon les résultats des tests affectués, nous constatons que l'approche est efficace pour les problèmes que nous avons rencontrés dans la pratique.

- Le chapitre 6 est la conclusion. Nous y présentons en particulier les extensions possibles de notre travail.

Remerciements

Je tiens à remercier tous les membres du jury.

Je tiens à exprimer toute ma reconnaissance à Monsieur Jean-Marie PROTH pour la confiance qu'il m'a accordée tout au long de cette thèse, ainsi que ses conseils qui m'ont permis de progresser dans mon travail.

J'adresse mes plus vifs remerciements à Monsieur Xiao-Lan XIE, Chargé de Recherches à l'INRIA et habilité à diriger des recherches, pour l'aide qu'il m'a apportée tout au long de ce travail.

J'exprime également ma sympathie à tous les membres[1] du projet SAGEP, ainsi qu'à Evelyne Agostini, pour l'aide et la bonne humeur dont ils font preuve quotidiennement. En particulier, je remercie C. Bouton, F. Chu, V.M. Savi pour leurs aides dans la réalisation du logiciel HMPS. Je remercie aussi P. Fournet-Fayard pour son aide sur l'usage de LaTeX.

1. Julien Antonio, Christophe Bouton, Christel Carmier, Chengbin Chu, Feng Chu, Pierre Fournet-Fayard, Nathalie Sauer, Vanio Murilo Savi, Abdelghani Souilah, François Vernadat, Philippe Wolff

Chapitre 1

Introduction générale

La concurrence internationale, un marché où l'économie d'envergure a remplacé l'économie d'échelle, et une demande de plus en plus forte de qualité, ont conduit à concevoir des systèmes de production de plus en plus sophistiqués. Les maîtres mots sont désormais **automatisation**, **intégration** des fonctions de l'entreprise, **gestion** hiérarchisée de la production, **flexibilité**, c'est-à-dire capacité de passer rapidement d'une production à une autre, et **modularité**, laquelle facilite la restructuration des systèmes de production. Pour faire face à cette situation dans de bonnes conditions de compétitivité, l'industrie fait appel à des systèmes à la fois adaptatifs et automatisés: les **ateliers flexibles** (en anglais: Flexible Manufacturing Systems, en abrégé: FMS). Les systèmes automatisés et flexibles, tels les ateliers flexibles, permettent de fabriquer une grande variété de produits, en petites et moyennes séries, grâce à l'utilisation de ressources multi-tâches. La conséquence de cette évolution est une complexité croissante des systèmes de production et de leur systèmes de commande. Cette évolution a rendu nécessaire le renforcement de la phase de conception préliminaire, encore appelée "étude papier". Cette étude papier englobe l'ensemble des activités qui vont de la spécification des produits que l'on veut fabriquer à la spécification du système qui sera chargé de cette fabrication. Elle contient, sans pour autant s'y limiter, la spécification fonctionnelle, la modélisation, l'évaluation et la spécification de la commande du futur système. [105][114]

Un problème fondamental rencontré dans les systèmes automatisés et flexibles est de trouver le bon compromis entre les deux qualités antagonistes que sont une bonne productivité et une grande flexibilité. Une solution partielle du problème se trouve dans la gestion de ces systèmes.

Selon une évaluation faite dans l'industrie aérospatiale et automobile, les progrès de la recherche au niveau exécution (serveur/actuateur) peuvent conduire à 1-2% de réduction du coût global, alors que les progrès de la recherche au niveau organisateur (planning, ordonnancement) peuvent conduire de 25 à 33% d'amélioration du coût global [55].

La gestion des systèmes de production est un vaste domaine qui s'intéresse

à l'ensemble des décisions à prendre pour assurer le bon fonctionnement d'un système de production. Des outils ont été développés afin d'aider aux prises de décision, du niveau le plus haut (stratégie générale de l'entreprise) au niveau le plus bas (fonctionnement des machines dans l'atelier de fabrication) [34].

Notre travail concerne les systèmes de production discrets. L'outil que nous utilisons est les réseaux de Petri (RdP) [100][109][68].

Puisque l'objectif de cette thèse est d'étudier la gestion des systèmes de production discrets dans un environnement réseaux de Petri, nous présentons d'abord une introduction aux systèmes de production discrets, dans laquelle nous donnons une vue globale de ce domaine. Puis nous introduisons la théorie des réseaux de Petri en nous limitant aux aspects qui seront utiles dans la suite.

La relation entre les réseaux de Petri et les systèmes de production automatisés est clairement expliquée dans des articles [120] et [3].

1.1 Gestion de la production

Une des caractéristiques fondamentales des systèmes de production que l'on rencontre dans l'industrie manufacturière est la nature *discrète* du processus de fabrication. Les occurrences des *événements* qui produisent les changements d'état du système dépendent de plusieurs facteurs: l'ordre partiel entre les différentes tâches imposé par les contraintes technologiques, les durées réelles des opérations (i.e. contraintes temporelles), et la disponibilité des ressources. Un tel système est un système à événements discrets (SED).

On note que les caractéristiques des systèmes automatisés et flexibles se rencontrent dans d'autres domaines, notamment celui des systèmes informatiques. Bien que les paramètres ne soient pas les mêmes, les systèmes informatiques sont également des systèmes multi-tâches et multi-ressources, et les problèmes qui se posent dans ce domaine s'expriment en termes de coordination d'activités parallèles asynchrones, de conflits de ressources, et de contraintes temporelles strictes. C'est pourquoi les outils d'évaluation employés aujourd'hui pour les systèmes de production sont au carrefour de plusieurs domaines d'application [60][105].

Il faut noter cependant que certaines caractéristiques, en particulier l'importance des débits, font que les outils disponibles n'ont pas la même efficacité en production manufacturière et en informatique.

1.1.1 Systèmes à événements discrets

La théorie des systèmes traditionnels est principalement consacrée à l'étude des systèmes *continus*, dont les comportements peuvent être décrits par des équations aux différences finies, des équations différentielles, ou des équations aux dérivées partielles. Malheureusement, il est impossible d'utiliser ces équations pour décrire

le comportement dynamique d'un système à événements discrets et, en particulier, d'un système de production. La même remarque s'applique aux réseaux de communication, aux réseaux d'ordinateurs, aux réseaux de transport: bien que différents des systèmes de production, ils ont des comportements qui en font également des systèmes à événements discrets [27][61] [73] [106] .

Lorsque l'on observe un système à événements discrets, on constate que son état n'évolue qu'en des points discrets du temps. L'évolution d'un SED est donc caractérisée par une séquence d'états et par la durée de chacun d'eux. La combinaison de la nature discrète de l'espace des états et de la nature continue de leur durée est une des difficultés de l'analyse des SEDs. D'autres difficultés sont:

- la présence de perturbations et d'interactions entre les sous-systèmes du système étudié,
- l'explosion de l'espace des états: habituellement le nombre d'états d'un SED croît de manière exponentielle avec le nombre de ses composants,
- la variété des champs d'application des systèmes à événements discrets.

Pour ces raisons, il est difficile de fournir une approche analytique unifiée dans l'étude des systèmes à événements discrets.

Les modèles des systèmes à événements discrets sont de deux types: les modèles temporisés et les modèles non temporisés. Les modèles non temporisés servent à étudier les propriétés qualitatives telles que l'absence de blocage, la vivacité, la réversibilité etc. (Voir, e.g., [37] [105] [106] [107] [120]). Les modèles temporisés sont utilisés pour l'évaluation des performances et pour l'optimisation du comportement des systèmes.

A ces deux types de modèles correspondent respectivement deux types de travaux: les études logiques et les études des performances. Les études logiques s'adressent à la description causale des comportements des systèmes. Les études des performances s'adressent à l'évaluation des performances des systèmes. Elles sont basées sur la théorie des probabilités et les techniques stochastiques. Le Tableau 1.1 donne une classification des modèles les plus connus [27] [61]. Dans le Tableau 1.1, on voit que les réseaux de Petri sont un outil complet qui permet de supporter la modélisation, l'analyse et l'évaluation des systèmes à événements discrets [105] [130], car les réseaux de Petri peuvent servir à l'étude logique et à l'étude des performances d'un système..

Ramadge et Wonham [106][107] ont introduit la théorie du langage formel et les automates pour étudier les systèmes à événements discrets. Ils ont étudié en particulier le problème de supervision de ces systèmes. Ce problème consiste à déterminer un contrôle aussi peu contraignant que possible pour que le comportement du système satisfasse des conditions imposées décrites par un langage légal. Un langage légal spécifie un ensemble de comportements admissibles. Les concepts du contrôle classique tels que la contrôlabilité, l'observabilité, le contrôle

	Temporisé	Non temporisé
Logique		Automates, Processus récursifs, RdP, etc.
Performance	RdP temporisé, Algèbre max-plus, Simulation, Files d'attente Automates temporisés, etc.	

TAB. 1.1 – *Classification des Modèles des Systèmes à Événements Discrets*

décentralisé et le contrôle hiérarchique sont étudiés. Une extension aux modèles temporisés a été étudiée dans Ostroff et Wonham [97] à l'aide de la théorie de logique temporelle(*real time temporal logic*).

Cohen et al [30][31] ont utilisé l'algèbre max-plus[33] pour modéliser et analyser les systèmes à événements discrets. Ils ont fait apparaître des propriétés similaires entre les modèles d'algèbre max-plus et les systèmes linéaires aux sens du contrôle classique. L'inconvénient principal de ce modèle est sa faible puissance descriptive.

Inan et Varaiya [66] ont proposé des modèles FRP (*Finitely Recursive Process* models). Ces modèles sont basés sur la théorie des CSP (*Communicating Sequential Processes* [63]). Un système à événements discrets est représenté par un ensemble d'équations récurrentes et la trajectoire de l'état du système est générée de manière itérative par rapport au temps. Les modèles FRP ont une grande puissance, ce qui permet de décrire une famille plus importante de systèmes à événements discrets que les techniques précédentes.

L'utilisation des files d'attente pour les problèmes d'analyse et d'évaluation des systèmes de production, et plus particulièrement les ateliers flexibles, fait l'objet d'une abondante littérature [25] [24]. Les réseaux de files d'attente sont adaptés à l'étude du fonctionnement en régime permanent lorsqu'il s'agit d'obtenir des résultats en première approximation sur le comportement du système, ou bien pour résoudre certains problèmes qui se posent à un niveau "haut" de la gestion (problèmes de flux). La littérature abondante qui traite des chaînes de Markov justifie les développements qui concernent les files d'attente. L'extension des processus Markoviens conduit aux processus semi-Markoviens, e.g., GSMP (*Generalized Semi-Markov Processes*)[47][46], dans lesquels un temps de distribution générale est associé à chaque événement. Le concept de dérivée qui joue un rôle prédominant dans la théorie de contrôle classique fait l'objet de nombreuses études

appelées communément l'Analyse des Perturbations (*Perturbation Analysis*)[62]. Le souci principal de l'analyse des perturbations est d'estimer les gradients des fonctions dont les valeurs sont obtenues par simulation.

Largement utilisée dans la pratique, la simulation a l'avantage de s'appliquer à n'importe quel type de système, aussi complexe soit-il. Au niveau de la finesse des résultats obtenus, la simulation reste incontestablement le moyen d'évaluation le plus puissant. Les langages de simulation sont nombreux (plus d'une centaine), mais les plus connus restent WITNESS [64], SIMAN [99], SIMULA [15], GPSS[49], et SLAM II[96]. Ils permettent d'exprimer le modèle sous forme d'un programme dont les entrées représentent le contrôle et les demandes. Si ce programme représente fidèlement le système, son exécution rend possible la connaissance du comportement de ce système sous différentes hypothèses de contrôle. Quand il s'agit d'optimiser les performances du système, la simulation offre donc peu de recours. Sans prétendre aborder en détail les problèmes liés à sa mise en œuvre, on reconnaîtra cependant l'inconvénient majeur de la simulation en raison de son aspect *boîte noire*: on définit des paramètres et on obtient des résultats, mais sans aucune information sur la façon dont les résultats dépendent des valeurs des paramètres. Bien entendu, cela n'est plus vrai si l'on superpose l'analyse des perturbations à la simulations.

Grâce à leurs représentations graphiques faciles à comprendre et à utiliser par les ingénieurs, les réseaux de Petri[1] [23] [92] [100] [109] fournissent un outil simple et efficace pour la modélisation des systèmes à événements discrets. Les réseaux de Petri temporisés permettent de prendre en compte de manière simple les durées des activités d'un système à événements discrets. Les réseaux de Petri ont été utilisés pour la modélisation et l'analyse des systèmes de production automatisés [54] [59] [56] [74] [104] [103] [102] [105] [120], des protocoles de communication [113], etc..

Il est utile de noter que l'exposé que nous venons de présenter n'est pas complet. L'objectif est de donner un aperçu de la recherche autour des systèmes à événements discrets et, en particulier, de la gestion des systèmes de production discrets, laquelle est notre principal intérêt de recherche.

1.1.2 Systèmes de production discrets

Un système de production dépend fortement de son environnement, en particulier de ses fournisseurs et de ses clients. La Figure 1.1 illustre cette dépendance.

D'un côté, les fournisseurs alimentent le système de production suivant les décisions prises par le système de gestion (voir aussi la Figure 1.2). La matière circule ensuite dans l'atelier de production entre les ressources et les stocks. Enfin, les produits finis sont stockés jusqu'à la livraison aux clients.

[1] Nous présentons en plus de détaille dans la section 1.2.

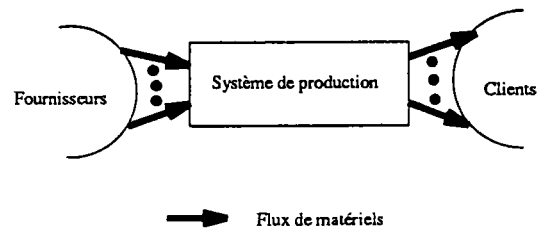


FIG. 1.1 – *Système de production et son environnement*

Un système de production contient toutes les ressources nécessaires, tant humaines que matérielles, qui permettent de transformer la matière première ou/et les composants en produits finis. Les systèmes de production sont organisés et gérés en fonction des demandes et des ressources disponibles.

Il existe une grande diversité de systèmes de production. De nombreuses typologies ont été proposées dans la littérature [91]. Même si elles ne sont pas uniques et peuvent être discutées, deux typologies nous semblent intéressantes pour caractériser un système de production [45].

La première typologie distingue les systèmes dans lesquels la production est déclenchée par les commandes des clients (production à la demande, *make-to-order*), et ceux dont la production peut s'effectuer en anticipant des demandes (production prévisionnelle, *make-to-stock*).

La **production à la demande** concerne principalement les entreprises fabricant une grande variété de produits dont la demande est trop aléatoire, et les entreprises qui ne définissent leurs produits qu'à partir de demandes précises des clients. Les sociétés de sous-traitance sont des exemples parfaits de cette catégorie [34].

La **production prévisionnelle** n'est possible que pour des entreprises qui fabriquent les produits dont la demande reste relativement stable et prévisible. Par exemple, les productions alimentaires sont à classer dans cette catégorie.

Bien entendu des situations intermédiaires existent: dans certaines firmes automobiles les voitures sont commencées sur la base du "*make-to-stock*" et sont complétées au vu des commandes des clients.

La production prévisionnelle conduit à des modèles déterministes ou stochastiques de gestion de stocks. Parmi ces modèles, on trouve en particulier les fameuses quantités économiques, qui assurent un compromis optimal entre les coûts de stockage et les coûts de changement de fabrication [13] [38] [134].

La deuxième typologie est basée sur le type de production. Elle distingue quatre catégories de systèmes [12] [34]:

- **La production unitaire**: dans un système de ce type, la fabrication de chaque produit est longue et coûteuse. La tâche principale consiste à réunir les moyens nécessaires au bon moment et au bon endroit. Ce type de production utilise les techniques de *l'ordonnancement de projets.*

- **La production en petites et moyennes séries:** de volume faible, le produit se déplace dans un atelier de production. Le problème consiste souvent à minimiser, non pas le temps de fabrication d'un seul produit, mais celui de l'ensemble de la production, cet objectif coïncidant souvent avec la minimisation des temps d'attente devant les ressources.

- **La production en grande série:** dans le cas où le nombre de produits similaires à fabriquer dans les mêmes délais devient très important, il est rentable de constituer des chaînes de fabrication (les ressources sont placées dans un ordre précis et inamovible). Celles-ci peuvent entraîner une meilleure productivité. La constitution de chaînes équilibrées est un élément primordial dans la réussite de ce type de production.

- **La production en continu:** ce type de production interdit toute possibilité d'attente entre deux ressources. Il concerne surtout une fabrication nécessitant la manipulation de matières liquides ou gazeuses. Là encore, la définition de la chaîne est souvent l'élément essentiel du bon fonctionnement de la production.

Dans ce travail, nous nous intéressons aux systèmes de production discrets fonctionnant *en prévisionnel ou à la demande* et *en petites et moyennes séries.*

1.1.3 Gestion des systèmes de production discrets

Le système de gestion à pour rôle d'assurer en permanence la bonne utilisation de l'ensemble des moyens de production. Pour bien gérer, les décisions prises doivent être basées sur des informations actualisées (ressources disponibles, situation des clients et des fournisseurs). Un bon système de gestion est caractérisé par sa capacité d'acquisition des informations nécessaires, et sa capacité de prises de décisions [128].

Les interactions entre le système de gestion, le système de production, et son environnement extérieur peuvent être schématisées par la Figure 1.2 (voir aussi la Figure 1.1).

Le système de gestion reçoit les commandes des clients et ou les décisions de production prises en interne, en fonction de celles-ci, planifie la production tout en prenant en compte les contraintes des fournisseurs et la capacité du système. De plus, il gère le système de production en tenant compte des aléas. Le système de gestion inclut parfois les décisions concernant les réapprovisionnements de matières premières et la livraison des produits finis.

Pour un système de production réel et complexe, les décisions sont de plusieurs types, et s'appliquent à différents niveaux[2]. On distingue habituellement trois

[2] En parlant du management de la complexité, MacFarlane [86] dit:

Typical procedures for the management of complexity are aggregation and hierar-

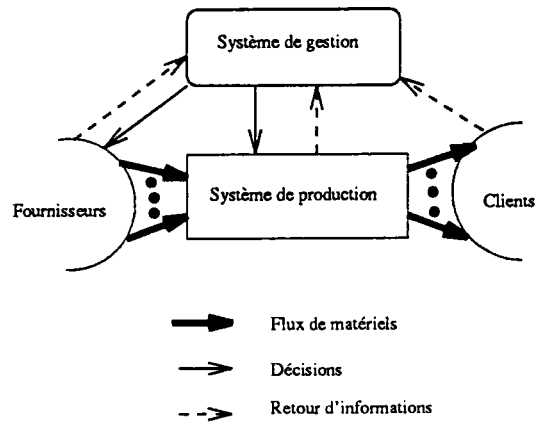


FIG. 1.2 - *Système de gestion, système de production et environnement*

niveaux successifs caractérisés par l'horizon sur lequel les décisions s'appliquent [4] [21].

- **Niveau stratégique**: La conduite générale de l'entreprise est décidée à ce niveau, et les décisions qui en découlent concernent la mise au point des installations de production. On définit entre autres la taille et l'emplacement de nouvelles usines, les modifications en profondeur à apporter aux usines existantes, le recrutement et la formation du personnel, l'acquisition d'équipements neufs, la conception de nouvelles gammes de produits, etc.. Ce sont ces décisions, prises souvent pour plusieurs années, qui vont contraindre et constituer les objectifs des niveaux inférieurs.

- **Niveau tactique**: Les décisions prises à ce niveau correspondent à un ensemble de décisions à moyen terme (horizon variant entre 3 mois et 2 ans en général). La capacité de production a été fixée par le niveau stratégique. A partir des commandes fermes des clients et des prévisions des demandes, les

---

chisation. ···. Nature, in the aeons of experiment over which evolution has taken place, has discovered that aggregation, modularisation and hierarchisation are effective ways of managing complexity, building systems capable of highly effective interaction with their environment by assembling the total complexity necessary by an incremental approach. In an effectively functioning hierarchy, the interactions between systems or units at the lower levels is such as to create a reduced level of complexity at the level perceived above. This reduction of externally perceived complexity proceeds up the hierarchy till the top level perceive the aggregated effect of all the lower levels in terms of a single entity with a manageable level of perceived complexity. ··· . The price to be paid for a reduction of externally perceived complexity is two-fold:

- A loss of detailed information;
- An appropriate degree of complexity in the sub-systems.

décisions tactiques définissent un plan de fabrication. Ce plan est établi à partir d'informations globales sur les capacités. Dans ce plan, les quantités à produire par période sont calculées de façon à répondre aux demandes au moindre coût, compte tenu des stocks initiaux disponibles et d'autres critères qui peuvent venir s'y ajouter. Certaines perturbations, telles que des fluctuations de la demande ou des aléas de production, viennent parfois compliquer le problème.

- **Niveau opérationnel**: Les décisions prises à ce niveau concernent l'utilisation des moyens de l'atelier afin de produire, dans les délais, les quantités fixées aux niveaux supérieurs. Elles gèrent le déroulement quotidien du processus de fabrication dans le respect de décisions tactiques. Les décisions opérationnelles assurent l'ordonnancement des opérations sur les machines, l'affectation des opérateurs aux machines, la livraison des produits finis, etc.. Elles tiennent compte de tous les détails du fonctionnement du système.

La figure 1.3 schématise les interactions entre ces décisions prises à différents niveaux.

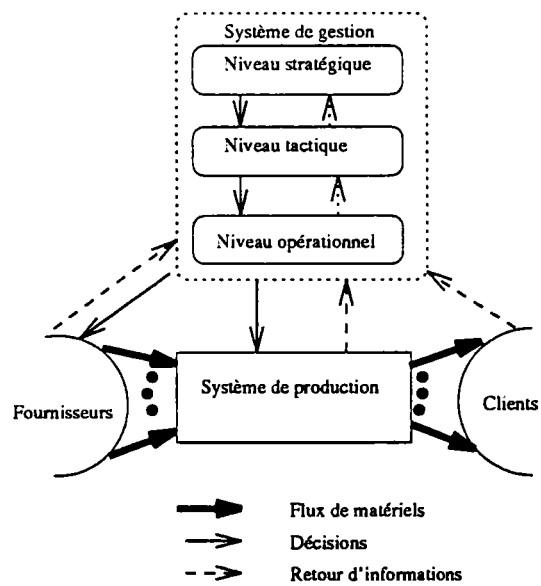


FIG. 1.3 – *Une structure décisionnelle à trois niveaux*

Dans ce travail nous nous intéressons aux décisions tactiques et opérationnelles.

1.2 Introduction aux réseaux de Petri

Les réseaux de Petri (RdP) [23] [37] [68] [92] [100] [105] [109] constituent un outil puissant de description des systèmes concurrents et parallèles évoluant

dans le temps de façon discrète, déterministe ou stochastique. Ils permettent une modélisation simple des systèmes de production à des fins d'analyse mathématique ou de simulation. Pour la simulation, on utilise souvent des réseaux de Petri complexes et généralement pauvres en propriétés mathématiques, comme par exemple les réseaux de Petri colorés ou à prédicats [68]. Pour l'analyse mathématique, des réseaux plus simples, mais riches en propriétés, comme par exemple les réseaux de Petri élémentaires, sont utilisés. Puisque notre étude concerne l'analyse mathématique des systèmes de production, nous nous limiterons aux réseaux de Petri élémentaires.

1.2.1 Notions de base

Un réseau de Petri est un graphe biparti composé de deux types de sommets: les *places* et les *transitions*. Des arcs relient les places aux transitions, ou les transitions aux places. Un arc ne relie jamais deux sommets de même nature. Généralement, les places sont représentées par des cercles et les transitions par des rectangles (ou des barres). Chaque place peut contenir un ou plusieurs *jetons*, représentés par des points. Comme nous les verrons plus loin, ces jetons permettent de modéliser la dynamique du système. Le *marquage* d'un RdP est un vecteur à composantes entières positives ou nulles, et dont la dimension est égale au nombre de places. Le $n^{ème}$ composante de ce vecteur représente le nombre de jetons qui figurent dans la place n du RdP.

Plus formellement, un réseau de Petri est un 5-tuple $G = (P,\ T,\ F,\ W,\ M_0)$ où:

- P est l'ensemble fini des places
- T est l'ensemble fini des transitions
- F est l'ensemble fini des arcs reliant les places aux transitions et les transitions aux places. $F \subseteq (P \times T) \cup (T \times P)$
- $W : F \rightarrow \mathbf{N}$ est la fonction poids attachée aux arcs, où $\mathbf{N}$ est l'ensemble des entiers naturels.
- $M_0 : P \rightarrow \mathbf{N} \cup \{0\}$ est le marquage initial.

Notons que $T \cap P = \emptyset$.

Le RdP sans son marquage est noté $PN = (P,\ T,\ F,\ W)$. Donc $G = (PN,\ M_0)$. Lorsque tous les poids des arcs sont égaux à 1, le RdP est dit *ordinaire* . Si M_0 est le marquage initial d'un RdP G, $M_0(p)$ est le nombre de jetons contenus dans la place p de G à l'instant initial.

Dans la suite, nous utilisons les notations suivantes:

- ${}^\bullet t$ est l'ensemble des places d'entrée de la transition t, c'est-à-dire l'ensemble des places p telles que $(p, t) \in F$.

- $t^\bullet$ est l'ensemble des places de sortie de la transition t, c'est-à-dire l'ensemble des places p telles que $(t,p) \in F$.

- ${}^\bullet p$ est l'ensemble des transitions d'entrée de la place p, c'est-à-dire l'ensemble des transitions t telles que $(t,p) \in F$.

- $p^\bullet$ est l'ensemble des transitions de sortie de la place p, c'est-à-dire l'ensemble des transitions t telles que $(p,t) \in F$.

Une transition $t \in T$ est dite *franchissable* (ou *tirable*) pour un marquage M si, quelque soit $p \in {}^\bullet t$, $M(p) \geq W(p,t)$. En d'autres termes, une transition est franchissable si chacune de ses places d'entrée contient un nombre de jetons au moins égal au poids de l'arc qui la relie à la transition. Une transition t peut être *tirée* (on dit encore *mise à feu* ou *franchie*) si elle est tirable.

Le franchissement ou la mise à feu d'une transition t consiste à:

- retirer $W(p,t)$ jetons de chaque place $p \in {}^\bullet t$,

- ajouter $W(t,p)$ jetons dans toutes les places $p \in t^\bullet$.

Ainsi, à partir du marquage M, le franchissement d'une transition t conduit au nouveau marquage M' défini comme suit:

$$\forall p \in P, \quad M'(p) = \begin{cases} M(p) - W(p,t) & \text{si} \quad p \in {}^\bullet t \\ M(p) + W(t,p) & \text{si} \quad p \in t^\bullet \\ M(p) & \text{sinon} \end{cases} \tag{1.1}$$

Nous noterons $M[t > M'$ le franchissement de t qui transforme M en M'. Nous noterons aussi $M[t >$ le fait que t est franchissable pour le marquage M.

A partir des notions définies précédemment, il est clair que le marquage peut évoluer par tirs successifs de transitions. On dira alors que la séquence de transitions $\sigma = (t^1, t^2, \cdots, t^s)$ est *franchissable* pour le marquage M^0 si les marquages successifs $(M^1, M^2, \cdots, M^s)$ vérifient

$$M^{k-1}[t^k > M^k \qquad \text{pour } k = 1, 2, \cdots, s$$

Nous noterons en abrégé $M^0[\sigma > M^s$ pour signifier que M^s est le marquage atteint à partir de M^0 par le franchissement de la séquence σ.

Etant donné deux marquages M et M', nous dirons que M' est *accessible* à partir de M si il existe une séquence franchissable σ telle que $M[\sigma > M'$. L'ensemble de tous les marquages accessibles à partir de M sera noté $R(M)$.

A partir d'une séquence de transitions σ, on peut construire un vecteur $\bar{\sigma}$, appelé *vecteur de comptage* de la séquence, dont la $j^{ième}$ composante représente le nombre d'occurrences de la transition t_j dans σ.

Si la séquence σ est franchissable, le marquage M' atteint à partir de M par le tir de σ vérifie la relation algébrique suivante:

$$M' = M + C \times \bar{\sigma} \tag{1.2}$$

C'est l'*équation fondamentale* ou l'*équation d'état* du réseau de Petri. Rappelons que $C = [c_{ij}]$, $i = 1, 2, ..., |P|$, $j = 1, 2, ..., |T|$ est la matrice d'incidence du réseau, définie de la manière suivante:

$$C_{ij} = \begin{cases} W(t_j, p_i) & \text{si } (t_j, p_i) \in F \\ -W(p_i, t_j) & \text{si } (p_i, t_j) \in F \\ 0 & \text{sinon} \end{cases} \tag{1.3}$$

Notons que l'équation 1.2 permet de calculer simplement le marquage atteint par une séquence σ quelconque, sans qu'il soit possible pour autant d'en déduire si la séquence σ est effectivement tirable. Par ailleurs le vecteur de comptage $\bar{\sigma}$ ne donne pas l'ordre dans lequel les transitions sont tirées.

Définition 1.1 (graphe d'événements) *Un graphe d'événements est un RdP élémentaire dans lequel:*

(i) chaque place a exactement une transition d'entrée et une transition de sortie;

(ii) tous les arcs sont pondérés à 1.

Définition 1.2 (Transitions sources et transitions puits) *Une transition sans place d'entrée est appelée* transition source. *Une telle transition est toujours tirable. Une transition sans place de sortie est appelée* transition puits. *Une telle transition peut être tirée si elle est tirable. Si une telle transition est tirée, les jetons sont prélevés dans la (les) place(s) d'entrée suivant les règles habituelles, mais aucun jeton n'est produit en sortie.*

Nous verrons ultérieurement[3] qu'une transition source est souvent utilisée, dans un modèle d'atelier, pour modéliser l'entrée des matières premières (ou des produits semi-finis) dans le système. De même, une transition puits est souvent utilisée pour modéliser la sortie de produits finis (ou de produits semi-finis) du système.

Définition 1.3 (T-invariant) *Un $|T| \times 1$ vecteur y tel que $y_i \in \mathbf{N} \cup \{0\}$, et $y \neq 0$, pour $i = 1, 2, \cdots, |T|$ est appelé un T-invariant (ou T-semi-flot) si $C \times y = 0$ où C est la matrice d'incidence.*

[3] Voir chapitre 4 et annexe C

Remarquons que, dans l'ensemble des T-invariants, si une composante donnée est nulle, cela signifie qu'il n'est pas possible de revenir au marquage initial en franchissant une séquence contenant la transition correspondant à cette composante. Dans le cas d'un atelier, cette situation signifie que l'état initial, qui peut être par exemple un état nécessaire à une mise au point ou à un réglage, ne pourra jamais être retrouvé si l'on effectue certaines opérations. Il y aurait, dans ce cas, une erreur de conception [105].

Définition 1.4 (T-invariant minimal) *Un T-invariant minimal est un T-invariant y tel qu'il n'existe pas un T-invariant x tel que $x \leq y$.*

Un T-invariant minimal correspond à une gamme de fabrication. Utiliser un T-invariant qui est une combinaison linéaire de T-invariants minimaux revient à décider d'utiliser différentes gammes dans des proportions données ou, en d'autres termes, de prendre *a priori* une partie des décisions de gestion [105].

Définition 1.5 (support de T-invariant) *Soit y un T-invariant. L'ensemble $\|y\| = \{t \mid y[t] > 0\}$ des transitions est appelé le support du T-invariant y, où $y[t]$ est la coordonnée de y correspondant à la transition t.*

Définition 1.6 (support minimal) *Soit y un T-invariant. Son support $\|y\|$ est dit minimal si $\|y\|$ ne contient pas (au sens strict) de supports d'autres T-invariants.*

Définition 1.7 (T-invariant à support minimal) *Un T-invariant y correspondant à un support minimal est appelé un T-invariant à support minimal.*

Définition 1.8 (consistance) *Un RdP G est dit consistant si il existe un marquage initial M_0 et une séquence de transitions franchissables σ qui contient chaque transition au moins une fois et dont le franchissement conduit à nouveau au marquage initial M_0, i.e., $M_0[\sigma > M_0$ et $\bar{\sigma} > 0$. Il a été démontré que G est consistant si et seulement si il existe un T-invariant strictement positif pour G.*

La consistance est une propriété structurelle[4] liée à la réversibilité et aux états d'accueil[5]. Elle caractérise l'existence d'une séquence de transitions qui ramène le marquage d'un RdP réversible au marquage initial.

Définition 1.9 (réversibilité) *Un RdP G est dit réversible si l'on peut toujours revenir au marquage initial quelque soit le marquage atteint. i.e., $M_0 \in R(M)$ pour tout $M \in R(M_0)$.*

---

[4] Les propriétés structurelles ne dépendent que de la structure du RdP, et non de la manière dont les jetons évoluent dans le réseau. En termes de production, les propriétés structurelles dépendent de l'architecture du système, et non de la manière dont il est géré. On comprend l'importance des propriétés structurelles lorsqu'on conçoit un système: elles permettent de garantir des comportements ultérieurs indépendamment de la manière dont le système sera géré, laquelle n'est pas connue à ce moment là.

[5] Un marquage M d'un RdP G est dit *état d'accueil* si il peut être atteint à partir de tous les marquages atteignables, i.e., $M \in R(M')$ quelque soit $M' \in R(M_0)$.

La réversibilité est une propriété compormentale[6], qui caractérise la possibilité de relancer un système soumis à panne, après avoir solutionné des dysfonctionnements.

Définition 1.10 (vivacité) *Un RdP G est dit vivant si chacune de ses transitions est vivante. Une transition t d'un RdP G est dite vivante si elle peut être franchie quel que soit le marquage atteint, i.e., $\forall M \in R(M_0)$, $\exists M' \in R(M)$ tel que t soit franchissable pour M'.*

La vivacité est aussi une propriété comportementale. Dans un système de production, il est fréquent que des activités manufacturières se déroulent en parallèle. Cela exige la synchronisation de ces activités et le partage des ressources. Une mauvaise synchronisation et un partage inadapté des ressources conduisent non seulement à une utilisation inefficace du système, mais peuvent également se traduire par un blocage total ou partiel des activités. L'étude de la vivacité d'un modèle de RdP garantit l'absence de blocage dans le système de production considéré.

Définition 1.11 (bornitude) *Une place p de G est dite k-bornée si le nombre de jetons dans cette place ne dépasse jamais k. i.e.. $M(p) \leq k$, $\forall M \in R(M_0)$.*

Une place p de G est dite bornée si elle est k-bornée pour un certain nombre entier $k > 0$.

G est dit k-borné si le nombre de jetons dans chaque place ne dépasse pas k, i.e., $M(p) \leq k$, $\forall p \in P$ et $\forall M \in R(M_0)$. Autrement dit, G est dit k-borné si chaque place de G est k-bornée.

Un RdP G est dit borné s'il est k-borné pour un certain nombre entier $k > 0$.

Un RdP G est dit sain s'il est 1-borné, i.e., $M(p) \leq 1$, $\forall p \in P$ et $\forall M \in R(M_0)$.

Certaines places d'un RdP, modèle d'un système de fabrication, représentent des zones de stockage. D'autres places contiennent des jetons qui représentent des ressources de fabrication. Il est souvent souhaitable de savoir si le nombre de jetons dans ces places est limité: cela permet de dimensionner le système de production correspondant, ou de découvrir certaines erreurs de conception. Par exemple, si un modèle n'est pas borné, on pourra voir s'accumuler des en-cours dans le système de fabrication correspondant, ce qui bien entendu n'est pas le reflet d'une conception efficace dans le cas d'un système automatisé.

Pour être complet, nous donnons la définition d'un P-invariant qui est symétrique à la notion de T-invariant:

Définition 1.12 (P-invariant) *Un $|P| \times 1$ vecteur x tel que $x_i \in \mathbf{N} \cup \{0\}$, et $x \neq 0$, pour $i = 1, 2, \cdots, |P|$ est appelé P-invariant (ou P-semi-flot) si $x' \times C = 0$ où x' est la transposée du vecteur x et C est la matrice d'incidence.*

[6]. Les propriétés compormentales sont souvent difficiles à étudier. Par contre, la plupart des propriétés structurelles peuvent être aisément vérifiées à l'aide des techniques algébriques. Sous certaines conditions, des propriétés structurelles impliquent des propriétés compormentales.

Nous aurons aussi besoin de la notion suivante:

Définition 1.13 (Conflit) *Deux transitions t_1 et t_2 sont en conflit structurel si elles ont au moins une place commune en entrée, i.e., ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset$. Elles sont en conflit effectif pour un marquage M si, de plus: $M[t_1>$, $M[t_2>$ et $\exists p$ t.q. $M(p) < W(p,t_1) + W(p,t_2)$. Un conflit effectif correspond à un choix exclusif entre deux franchissements.*

1.2.2 Méthodes d'analyse générales

Arbres des marquages atteignables et arbre de recouvrement:[100] [105] [92] [109]

Soit G un RdP muni d'un marquage initial M_0. L'objectif de l'arbre des marquages atteignables est de découvrir tous les marquages que l'on peut atteindre à partir de M_0. Un arbre des marquages atteignables est une aborescence dont les nœuds sont les marquages atteignables à partir de M_0, et dont chaque arc représente le tirage d'une transition. La racine de l'aborescence représente M_0.

Notons que chaque marquage conduit à autant de marquages qu'il y a de transitions tirables à partir de ce marquage, et que le même marquage peut se retrouver à différents endroits de l'aborescence. Notons également qu'un arbre des marquages atteignables se développe indéfiniment dans la plupart des cas.

Pour éviter d'aboutir à un arbre qui se développe indéfiniment, il a été décidé:

(i) qu'un marquage qui a été précédemment rencontré est marqué par "old"; un nœud marqué "old" sera une feuille de l'aborescence;

(ii) que si un marquage M^* obtenu est tel qu'il existe, sur le chemin qui mène de la racine M_0 à M^*, un marquage M tel que $M^*(p) \geq M(p)$ pour toutes les places p du RdP, et si $M^*(p) > M(p)$ pour au moins une place, alors le marquage de cette place est marqué "ω" (ω peut se comprendre comme signifiant l'infini). Ce marquage restera ω dans tous les développements suivants, et la règle (i) s'applique également aux marquages contenant le symbole ω. Bien entendu, $\omega + k = \omega$, $\omega - k = \omega$, quelque soit l'entier k.

Un *arbre de recouvrement* est un arbre des marquages atteignables ajoutant les règles suivantes aux règles (i) et (ii):

(iii) un nœud correspondant à un marquage tel qu'aucune transition n'est tirable sera marqué "DeadEnd" et constituera une feuille de l'aborescence;

(iv) tous les nœuds qui ne sont pas marqués "old" et qui admettent au moins un descendant sont marqué "new".

L'arbre de recouvrement contient moins d'informations que l'arbre des marquages atteignables, mais reste de taille limitée. L'arbre des marquages atteignables et l'arbre de recouvrement sont donc des outils d'analyse des RdP.

Les conclusions suivantes peuvent être tirées de l'arbre de recouvrement et de l'arbre des marquages atteignables:

- Un RdP est borné si et seulement si aucun des marquages correspondant aux nœuds de l'arbre de recouvrement ne contient le symbole ω. On comprend que si un RdP modélise un système de production, il est nécessaire d'identifier les situations qui augmentent les marquages indéfiniment. L'arbre de recouvrement est un moyen de détecter ces situations.

- L'arbre de recouvrement permet de détecter les transitions qui ne sont pas tirées, ou qui ne sont plus tirées à partir d'un certain point d'évolution. Cela permet de mettre en évidence les fonctionnalités d'un système de production qui ne sont pas actives, ou qui deviennent inactives au bout d'un certain temps, pour certains états initiaux du système et certaines séquences de décisions.

- Lorsque le système est borné, l'arbre des marquages atteignables donne l'ensemble des états qui peuvent être atteints les uns à partir des autres, et la manière de réaliser ces transformations. Du point de vue des systèmes de production, l'arbre des états atteignables fournit donc toutes les évolutions possibles du système connaissant son état initial représenté par M_0.

Matrice d'incidence et équation d'état[100] [105] [92] [109]

La matrice d'incidence d'un RdP est définie par la relation 1.3 et l'équation d'état est donnée par l'équation 1.2.

Une colonne de la matrice d'incidence correspond aux modifications apportées aux places lors du franchissement de la transition correspondante. Mais la matrice d'incidence est indépendante du marquage. Elle ne nous donne donc aucun renseignement sur la possibilité de franchir une transition donnée.

L'équation d'état donnée en 1.2 ne garantit pas que σ soit tirable. Elle permet simplement de trouver le marquage atteint lorsqu'on connaît le marquage initial M_0 et la séquence tirable σ.

Il n'est pas possible de représenter une boucle dans une matrice d'incidence car cela exigerait de mettre à l'intersection de la ligne correspondant à la place et de la colonne correspondant à la transition concernée par cette boucle à la fois +1 et -1. Pour modéliser le fait que plusieurs opérations ne peuvent se dérouler simultanément sur la même machine, nous attacherons une boucle à toute transition dans les modèles des systèmes de production comportant des transitions temporisées. Cela ne nous empêchera pas d'utiliser la matrice d'incidence dans tout calcul ne faisant pas intervenir le temps.

Méthodes de réduction [100] [105] [92] [109]

Un des problèmes que l'on rencontre lorsque l'on utilise les RdP pour modéliser et analyser les systèmes de production est la taille généralement importante des modèles obtenus (voir chapitre 3).

Le tableau 1.2 résume les méthodes de réduction qui agissent localement sur le modèle [105]. Elles sont de deux types, à savoir *les méthodes de transformation* et *les méthodes de synthèse* .

Méthodes de transformation	sur les places	simplification des places redondantes
		fusion de places doublées
		fusion de places équivalentes
		suppression de places implicites
	sur les transitions	post-fusion
		fusion latérale
		pré-fusion
Techniques de synthèse	techniques ascendantes	combinaison de places
		combinaison de chemins élémentaires
	techniques descendantes	affinage des transitions
		affinage des places

TAB. 1.2 – *Méthodes de réduction*

Une place redondante est une place dont le marquage n'influence pas le tirage de ses transitions de sortie. Dans la pratique, le marquage d'une telle place est lié aux marquages des places qui ont les mêmes transitions de sortie.

Deux places p_1 et p_2 sont structurellement doublées et peuvent être fusionnées si aucun jeton n'arrive dans p_1, p_2 avant qu'un jeton ne soit arrivé dans chacune des places des ${}^\bullet t$, où les transitions t sont les éléments de $p_1^\bullet$, $p_2^\bullet$.

Deux places p_1 et p_2 sont équivalentes si et seulement si il existe deux transitions t_1 et t_2 telles que les conditions suivantes sont vérifiées: (i) p_i $(i = 1, 2)$ est une place d'entrée de t_i, (ii) p_i $(i = 1, 2)$ n'est pas une place d'entrée de t_{3-i}, (iii) p_1 et p_2 ne sont places d'entrée d'aucune autre transition que t_1 et t_2, (iv) t_1 et t_2 ont mêmes places d'entrée et de sortie, excepté pour ce qui concerne p_1 et p_2, ou n'ont pas de place d'entrée ou de sortie, (v) p_1 et p_2 sont places de sortie d'au moins une transition. Cette transition n'est pas nécessairement la même pour les deux places.

Une place implicite est une place dont le marquage n'est jamais un obstacle au franchissement des transitions dont elle est place d'entrée, c'est-à-dire que lorsque les autres places entrées d'une telle transition ont un marquage suffisant pour

permettre le franchissement, alors le marquage de la place considérée est lui aussi toujours suffisant pour permettre ce franchissement.

Les méthodes de transformation partent d'un réseau de Petri de taille importante, appliquent les règles de transformation pour obtenir un réseau de faible taille, et déduisent les propriétés du réseau initial des propriétés du réseau de faible taille. Les règles de transformation qui préservent les propriétés souhaitées ont été proposées dans [14], [80], [81]. L'inconvénient de ces méthodes réside dans la difficulté que l'on rencontre pour déterminer les sous-réseaux réductibles.

Les méthodes de synthèse construisent les modèles RdP de manière systématique et progressive afin de préserver les propriétés souhaitées tout au long du processus de conception. L'idée est d'adopter un processus de conception qui préserve les propriétés au lieu de vérifier les propriétés après avoir obtenu le modèle RdP. Deux types d'approches existent: *les méthodes ascendantes* et *les méthodes descendantes.*

Une méthode descendante part d'un modèle agrégé du système global, et l'affine progressivement pour introduire de plus en plus de détails. L'affinement principal est de substituer un réseaux de Petri bien formé à une place ou transition (voir, e.g., [122], [126], [132]). Cette approche est bien adaptée pour modéliser les systèmes composés de sous-systèmes presque indépendants. Pour les systèmes de production composés de sous-systèmes fortement liés à cause de ressources partagées, il est difficile de trouver des modèles agrégés de taille acceptable.

Une approche ascendante part des modèles des sous-systèmes (aussi appelés modules) et intègre des modules par fusion de places ou de transitions communes (voir, e.g., [2], [5] [72] [95] [116] [132]). Pour les approches ascendantes générales, la faiblesse provient de la difficulté de déterminer les conditions dans lesquelles l'intégration préserve les propriétés souhaitées. Les approches modulaires et les approches incrémentales [28] sont également des approches ascendantes.

Même si les méthodes de réduction tiennent une place non négligeable dans la théorie des RdP et si elles s'appliquent aux RdP dans lesquels les transitions ne sont pas temporisées, elles exigent, dans le cas des RdP temporisés, des conditions supplémentaires qui ne sont que très rarement vérifiées en pratique. C'est la raison pour laquelle nous proposons plus loin une approche modulaire (voir chapitre 4), qui est beaucoup plus naturelle dans l'industrie, pour résoudre les problèmes liés à la taille. Nous verrons qu'il est possible, pour certaines applications et sous certaines hypothèses, de diviser le système à étudier en modules de taille raisonnable, de simplifier ces modules, puis d'en faire la synthèse d'une manière qui permet de déduire les propriétés du système complet des propriétés des modules.

1.2.3 Réseaux de Petri temporisés

Dans la littérature, on trouve trois types de temporisations:

- la temporisation des transitions;

- la temporisation des places;

- la temporisation de certains arcs

Ramchandani [108] fut le premier à introduire les réseaux de Petri temporisés. Dans son modèle appelé RdP T-temporisés, il associe une durée à chaque franchissement de transition. Dans le cas du modèle étudié par Sifakis [119], la temporisation est associée aux places et non aux transitions. Ces réseaux sont appelés réseaux P-temporisés et, dans ce cas, les jetons ont un temps de séjour minimal dans chaque place. Stark [121] et Hanisch [51] ont proposés une autre approche pour la prise en compte du temps: la temporisation est associée à certains arcs, appelés "arc temporisés". Les réseaux de Petri "arc temporisés" sont utilisés pour modéliser et analyser certains processus chimiques [52]. Nous nous intéressons aux processus discrets. Comme l'a montré Sifakis [119], les modèles de RdP T-temporisés et de RdP P-temporisés sont équivalents. Dans la suite, nous préférons utiliser la temporisation des transitions qui nous semble plus parlante lorsqu'il s'agit d'activités de fabrication.

Supposons que le temps associé à une transition t soit θ_t, et que le franchissement de t débute à l'instant T_0. Alors, franchir la transition t consiste à:

1. retirer $W(p,t)$ jetons de tout $p \in^\bullet t$ (i.e., de toutes les places d'entrée de t) à l'instant T_0;

2. ajouter $W(t,p)$ jetons dans tout $p \in t^\bullet$ (i.e., dans toutes les places de sortie de t) à l'instant $T_0 + \theta_t$.

Entre les instants T_0 et $T_0 + \theta_t$, les jetons sont supposés séjourner dans la transition. Cela représente, dans notre approche, le séjour de pièces sur une machine au cours de leur transformation ou de leur assemblage. Cependant, les propriétés des réseaux de Petri s'énoncent en supposant que les jetons concernés par un franchissement font partie des places d'entrée tant que le franchissement n'est pas terminé.

1.3 L'approche proposée

Lorsque les demandes qui s'appliquent à un système de production varient au cours du temps, nous disons que le système est à fonctionnement non cyclique. Les performances d'un tel système dépendent de sa gestion, i.e., de la planification et de l'ordonnancement de la production. Les systèmes à fonctionnement non cyclique ne peuvent plus être modélisés à l'aide de graphes d'événements [105] [130].

L'approche que nous adoptons pour l'étude des systèmes à fonctionnement non cyclique est modulaire et hiérarchique [54] [105] [130].

Pour modéliser et analyser un système de production réel, qui est souvent complexe, nous adoptons une approche modulaire (cf. chapitre 4 et annexe C) qui consiste à:

1. Décomposer le système de production en un ensemble de modules de faible taille permettant une modélisation et une analyse simple et efficace;
2. Modéliser les modules à l'aide des réseaux de Petri;
3. Analyser les propriétés qualitatives souhaitées pour les systèmes de production (bornitude, vivacité, réversibilité, consistance, flexibilité, etc.);
4. Evaluer les propriétés quantitatives de chaque module (productivité, taux d'utilisation des ressources, niveaux moyens des stocks, etc.). Cette étape comprend la conception d'un système local d'aide à la décision;
5. Intégrer les modèles des modules de manières à préserver les propriétés qualitatives souhaitées:
6. Déduire les propriétés quantitatives du système intégré de celles des modules.

En pratique, le découpage d'un système de production en modules dépend du système considéré. Il peut être résolu par une approche combinant l'expertise et les techniques de classification automatique [105].

L'utilisation d'une approche modulaire nécessite des:

- *transitions d'entrée*, pour représenter l'arrivée de produits ou de matières premières dans un module:
- *transitions de sortie*, pour représenter le départ de produits finis du module;
- *places d'interface*, pour relier les différents modules.

Pour gérer le système, nous adoptons l'approche hiérarchique qui comprend une hiérarchie à deux niveaux. Dans le prochain chapitre (chapitre 2), nous montrerons les avantages et les inconvénients de l'approche hiérarchique en étudiant la planification d'un système de production simple. Comme un outil complet, RdP sera utilisé, dans les chapitres suivants, pour la modélisation modulaire et la gestion hiérarchique de systèmes de production discrets.

Chapitre 2

Une hiérarchie à deux niveaux

2.1 Approche hiérarchique et approche globale

Les systèmes de production classiques sont généralement de taille importante, ce qui rend leur gestion complexe. En planification de la production, on distingue deux approches fondamentalement différentes: l'approche globale et l'approche hiérarchique.

2.1.1 Approche globale

Cette approche utilise un modèle monolithique qui décrit le problème de planification de manière détaillée. Elle fournit toutes les décisions sur l'horizon complet.

L'approche globale est très limitée pour les raisons suivantes:

- Le problème de planification est couramment formulé comme un problème de programmation linéaire en variables mixtes. Les procédures par séparation et évaluation, qui utilise parfois la technique de relaxation lagrangienne pour le calcul des bornes inférieures, sont souvent mises en œuvre pour obtenir la solution exacte [69]. Ces procédures sont souvent inacceptables en raison d'un temps de calcul exponentiel. Des méthodes approximatives basées sur des techniques de décomposition sont proposées [75], mais ne garantissent pas l'obtention d'une solution admissible.

- Beaucoup de données sont prévisionnelles, ce qui conduit à des résultats sujets à caution lorsqu'on fait des prévisions détaillées sur un horizon important.

2.1.2 Approche hiérarchique

L'approche hiérarchique est proposée par de nombreux auteurs[1]. Cette approche décompose le problème de planification en sous-problèmes. Chaque sous-problème est lié à un niveau de la hiérarchie. A chaque niveau, les entités sont des agrégats des entités du niveau immédiatement inférieur. L'horizon de planification décroît en descendant dans la hiérarchie. Les décisions correspondant à ces agrégats sont désagrégées au niveau immédiatement inférieur.

Plusieurs efforts ont été fait dans le domaine de la gestion hiérarchisée de systèmes de production. Voir e.g., [19], [20], [16], [39], [18], [50], [1], [17], [110], [93]. Plusieurs modèles ont été étudié, e.g., le modèle de Hax et Meal [57], le modèle de Axaster [6] [7] [8], le modèle de Tsubone, Matsuura, et Tsutsu [125], le modèle de Saad [112], le modèle de Thompson, Davis et Watanabe [124] [123], le modèle de Inman et Jones [67].

Des synthèses de la littérature sur l'approche hiérarchique de la production peut être trouvées dans [42] [36] [94], ainsi que dans les thèses de Xie[128], Meier[90], Nagi[93], Libosvar[83], et Mehra[89].

Xie [128], après avoir introduit la notion de configuration, a étudié l'ordonnancement en temps réel d'un atelier flexible par une approche hiérarchique à trois niveaux: séquencement de configurations, contrôle de flux et ordonnancement en temps réel. La notion de configuration a pour but de décrire un état du système permettant de fabriquer un ensemble de produits. Les machines de l'atelier sont sujettes à pannes.

Meier [90] s'intéresse aux problèmes posés par la gestion de production d'un atelier flexible. Une procédure hiérarchique à deux niveaux a été développée pour la planification et la commande de la production.

Nagi [93] a étudié la conception et l'exécution d'un système de gestion de production hierarchisée. L'agrégation spatiale et temporelle, la consistance et la désagrégation sont étudiées.

Libosvar [83] s'intéresse à la gestion des flux de production. Ce type de décisions est à prendre au niveau le plus élevé d'un système de gestion de production hiérarchisée.

Mehra [89] a proposé une méthode hiérarchique à deux niveaux pour aborder le problème de planification d'un système de production. L'agrégation des produits, l'agrégation des machines et l'agrégation du temps sont considérés simultanément. Cette méthodes a été appliquée à un atelier de *Pangborn Corporation* (USA).

Afin d'introduire les bases de l'approche hiérarchique, nous étudions, dans ce chapitre, la planification d'un système de production simple. L'objectif est de montrer comment la valeur de la fonction objectif varie avec le coût de stockage des produits dans la cellule, ce qui permet de se faire une idée des avantages et des inconvénients de l'approche hiérarchique.

[1] Voir e.g.,[6] [7] [8] [9] [10] [16] [17] [19] [20] [22] [84] [36] [39] [40] [42] [43] [50] [53] [57] [77] [82] [87] [94] [110] [111] [112] [123] [124] [125] [127] [131] [135] [89] [90] [93] [128] [83]

2.2 Un exemple illustratif

Considérons un système à 2 machines M_1 et M_2 qui fabrique 4 types de produits P_1, P_2, P_3, et P_4. Ce système est illustré par la Figure 2.1.

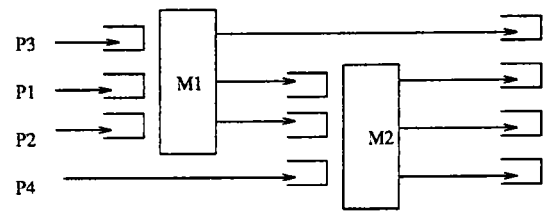


FIG. 2.1 – *Un exemple*

On fait les hypothèses suivantes:

1. A chaque type de produit est associé un routage qui indique la séquence des machines à visiter pour effectuer sa fabrication.

2. Pour chaque type de produit, le dernier stock est utilisé pour stocker les produits finis.

3. Un coût de stockage et un coût de rupture sont associés à chaque stock de produits finis. On suppose que ces coûts sont déterministes et qu'ils sont connus à priori. La rupture de stock n'est pas autorisée pour les produits en cours de fabrication.

4. La durée de chaque opération est déterministe et connue à priori.

5. Les matières premières arrivent dans le système sur un mode "juste-à-temps". Par conséquent, on peut supprimer les premiers stocks de chaque type de produit, et on obtient la Figure 2.2.

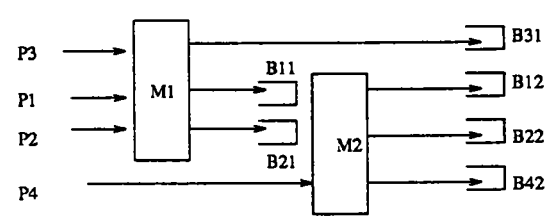


FIG. 2.2 – *Le système obtenu en supprimant les premiers stocks*

2.3 Modélisation des systèmes généraux

Dans ce paragraphe, on généralise l'exemple précédent et on présente des définitions supplémentaires.

2.3.1 Définitions et notations

Nous utilisons plusieurs définitions:

1. **Horizon:** $\mathbb{T} = T_1 \cup T_2 \cup \cdots \cup T_K$ est une période de longueur $K \times H$. K est le nombre de périodes élémentaires. H est la longueur de chaque période élémentaire. Dans la suite, on suppose que la durée de chaque période élémentaire est la même. L'horizon est l'extrémité de la période $\mathbb{T}$, c'est à dire l'instant $K \times H$ si l'instant initial est l'instatnt 0.

2. **Machines:** $\mathbb{M} = \{M_1, M_2, \cdots, M_m\}$, m est le nombre de machines considérées.

3. **Types de produits:** $\mathbb{P} = \{P_1, P_2, \cdots, P_n\}$, n est le nombre de types de produits considérés.

4. **Stocks:** $\mathbb{B} = \{B_{ij} \mid P_i \text{ visite } M_j\}$. B_{ij} est le stock dédié au produit P_i à la sortie de la machine M_j. On a deux types de stocks: les stocks internes et les stocks terminaux (i.e., stocks contenant des produits finis).

5. **Opérations:** $\mathbb{O} = \{< P_i, M_j > \mid P_i \text{ visite } M_j, \forall i = 1, \cdots, n; \forall j = 1, \cdots, m\}$ est l'ensemble des opérations dans le système.

Pour chaque $i = 1, \cdots, n$, on définit $\mathcal{O}_i = \{< P_i, M_j > \mid P_i \text{ visite } M_j, \forall j = 1, \cdots, m\}$ comme l'ensemble des opérations nécessaire pour la production de P_i.

Pour chaque $j = 1, \cdots, m$, on définit $\mathbb{I}_j = \{P_i \mid P_i \text{ visite } M_j, \forall i = 1, \cdots, n\}$ comme l'ensemble des types de produits qui visitent la machine M_j.

On définit aussi:

- $\mathcal{O}^{pre} = \{(i,j) \mid \text{le rang de l'opération } < P_i, M_j > \in \mathcal{O}_i \text{ est inférieur à } |\mathcal{O}_i|\}$
- $\mathcal{O}^{fin} = \{(i,j) \mid \text{le rang de l'opération } < P_i, M_j > \in \mathcal{O}_i \text{ est égal à } |\mathcal{O}_i|\}$

On note $< P_i, M_j > \prec < P_i, M_l >$ pour signifier que l'opération $< P_i, M_j >$ est suivie par l'opération $< P_i, M_l >$.

Les paramètres suivants sont utilisés:

t_{ij}: temps nécessaire pour exécuter l'opération $< P_i, M_j >$;

α_{ij}: coût de stockage associé aux stocks $B_{ij} \in \mathbb{B}$, $\forall i,\ j$;

β_i: coût de rupture associé aux stocks $B_{ij} \in \mathbb{B}$, $\forall (i,j) \in \mathcal{O}^{fin}, \forall i$.

On note aussi $d_i(k)$ la demande de P_i à la fin de la période T_k.
Les variables d'état sont:

$x_{ij}(k)$: niveau du stock B_{ij} à la fin de la période T_k.

Les variables de contrôle sont:

$u_{ij}(k)$: nombre de produits de type P_i qui doivent subir l'opération $< P_i, M_j >$ durant la période T_k.

2.3.2 Positionnement du problème général

Avec les notations précédentes et les deux notations conventionnelles suivantes:

$$x^+ = max(x, 0) \qquad x^- = -min(x, 0)$$

le problème de planification de la production peut s'écrire comme suit:

$$Min \sum_k \sum_i \{ \sum_{j|(i,j)\in\mathcal{O}^{pre}} \alpha_{ij} \times x_{ij}(k) + \sum_{j|(i,j)\in\mathcal{O}^{fin}} [\alpha_{ij} \times x_{ij}^+(k) + \beta_i \times x_{ij}^-(k)] \} \quad (2.1)$$

sous les contraintes suivantes:

- dynamique d'état:

$$\begin{aligned} x_{ij}(k) &= x_{ij}(k-1) + u_{ij}(k) - u_{il}(k), \quad (2.2) \\ &\forall k,\ \forall (i,j) \in \mathcal{O}^{pre}, (i,l) \in \mathcal{O}^{pre},\ < P_i, M_j > \prec < P_i, M_l > \end{aligned}$$

$$x_{ij}(k) = x_{ij}(k-1) + u_{ij}(k) - d_i(k), \qquad \forall k,\ \forall (i,j) \in \mathcal{O}^{fin} \quad (2.3)$$

- contraintes de capacité:

$$\sum_{i:P_i\in\mathbb{I}_j} t_{ij} \times u_{ij}(k) \leq H, \qquad \forall j,\ \forall k \quad (2.4)$$

- contraintes sur les variables:

$$x_{ij}(k) \begin{cases} \geq 0, & \text{si } k > 0,\ \forall (i,j) \in \mathcal{O}^{pre} \\ = x_{ij}(0), & \text{si } k = 0 \end{cases} \quad (2.5)$$

$$x_{ij}(k-1) \geq u_{ij}(k) \geq 0,\ \forall i,\ \forall j,\ \forall k \quad (2.6)$$

Le critère (2.1) signifie que la fonction objectif est de minimiser la somme des coûts de stockage et des coûts de rupture sur l'horizon de planification.

Les contraintes d'états distinguent deux cas: stocks internes (2.2) et stocks terminaux (2.3) contenant les produits finis.

Les contraintes de capacité (2.4) indiquent que, pour chaque machine, la somme des temps d'exécution des opérations prévues sur une période élémentaire ne doit pas excéder la durée de cette période.

Les contraintes (2.5) et (2.6) indiquent d'une part qu'il ne peut y avoir de rupture dans les stocks internes, et d'autre part que la production d'une machine durant une période élémentaire ne peut excéder le contenu des stocks en amont au début de la période.

2.4 Modélisation du système illustratif

2.4.1 L'exemple re-visité

Afin de faciliter l'étude du système illustratif, nous re-étiquetons tous les buffers et nous étiquetons toutes les opérations, ce qui est schématisé dans la Figure 2.3 dérivée de la Figure 2.2.

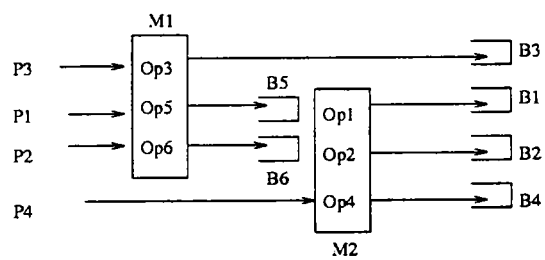


FIG. 2.3 – *Notations définitives*

Ainsi, on a les paramètres suivants:

- $t_1, \cdots, t_6$: temps nécessaire pour exécuter les opérations $Op1, \cdots, Op6$;
- $d_1(k), \cdots, d_4(k)$: demande de $P_1, \cdots, P_4$;
- $x_1(0), \cdots, x_6(0)$: niveaux initiaux des stocks $B_1, \cdots, B_6$;
- $\beta_1, \cdots, \beta_4$: coûts de rupture pour les types de produits $P_1, \cdots, P_4$ dans les stocks $B_1, \cdots, B_4$;
- $\alpha_1, \cdots, \alpha_6$: coûts de stockage des produits dans les stocks $B_1, \cdots, B_6$.

On a encore des variables suivantes:

- $x_1(k), \cdots, x_6(k)$: niveau des stocks $B_1, \cdots, B_6$ à la fin de période k;
- $u_1(k), \cdots, u_6(k)$: nombre d'opérations $Op1, \cdots, Op6$ exécutées durant la période k.

2.4.2 Formulation du problème donné en exemple

Avec les paramètres et les variables précédents, on peut poser le problème relatif à notre exemple. Il s'écrit:

$$Min \sum_{k=1}^{K} \{ \sum_{i=1}^{6} \alpha_i \times x_i^+(k) + \sum_{i=4}^{4} \beta_i \times x_i^-(k) \} \tag{2.7}$$

Sous les contraintes (pour tout $k = 1, \cdots, K$):

- dynamique d'état:

$$x_i(k) = x_i(k-1) + u_i(k) - d_i(k), i = 1, 2, 3, 4 \tag{2.8}$$

$$x_5(k) = x_5(k-1) + u_5(k) - u_1(k) \tag{2.9}$$

$$x_6(k) = x_6(k-1) + u_6(k) - u_2(k) \tag{2.10}$$

- contraintes de capacité:

$$t_1 u_1(k) + t_2 u_2(k) + t_4 u_4(k) \leq H \tag{2.11}$$

$$t_3 u_3(k) + t_5 u_5(k) + t_6 u_6(k) \leq H \tag{2.12}$$

- contraintes sur les variables:

$$x_i(k) = x_i(0), k = 0, \ i = 1, \cdots, 6 \tag{2.13}$$

$$x_5(k) \geq 0 \tag{2.14}$$

$$x_6(k) \geq 0 \tag{2.15}$$

$$0 \leq u_1(k) \leq x_5(k-1) \tag{2.16}$$

$$0 \leq u_2(k) \leq x_6(k-1) \tag{2.17}$$

$$u_3(k), u_4(k), u_5(k), u_6(k) \geq 0 \tag{2.18}$$

Dans ce cas spécifique, la fonction objectif (2.7) à minimiser est la somme des coûts de stockage et des coûts de rupture pour les 4 types de produits sur l'horizon de planification.

Les contraintes (2.8) à (2.18) ont été expliquées lorsque le problème général a été introduit.

2.5 Le modèle du niveau haut

Au niveau haut, les 2 machines M_1 et M_2 sont groupées en une cellule, et les 4 types de produits sont groupés en deux familles de produits: P_1 et P_2 sont dans la famille F_1, P_3 et P_4 sont dans la famille F_2. Cette situation est schématisée dans la Figure 2.4.

Soient $\bar{K}$ nombre de sous-périodes, $\bar{H}$ la longueur de chaque sous-période, donc, $K \times H = \bar{K} \times \bar{H}$.

Soient aussi

T_1, T_2 les temps nécessaires estimés au niveau haut pour la production d'une unité de chacune des familles F_1 et F_2 respectivement.

$D_1(\kappa)$, $D_2(\kappa)$ les demandes des familles F_1 et F_2 respectivement à la fin de sous-période κ.

$X_1(0)$, $X_2(0)$ les stocks initiaux pour F_1 et F_2 respectivement.

$\bar{\beta}_1$, $\bar{\beta}_2$ les coûts de rupture pour F_1 et F_2 respectivement.

$\bar{\alpha}_1$, $\bar{\alpha}_2$ les coûts de stockage pour F_1 et F_2 respectivement.

$X_1(\kappa)$, $X_2(\kappa)$ les niveaux des stocks de F_1 et F_2 respectivement au début de la sous-période κ.

$U_1(\kappa)$, $U_2(\kappa)$ les quantités de F_1 et F_2 produites respectivement pendant la sous-période κ.

On a besoins de spécifier les paramètres suivants afin de pouvoir construire le modèle du niveau haut: $\bar{K}$; $\bar{H}$; T_1, T_2: $D_1(\kappa)$, $D_2(\kappa)$; $X_1(\kappa)$, $X_2(\kappa)$; $\bar{\beta}_1$, $\bar{\beta}_2$; $\bar{\alpha}_1$, $\bar{\alpha}_2$.

Une des approches dont on peut disposer pour évaluer ces paramètres se trouve dans [58]. Pour le cas spécifique étudié, nous obtenons facilement:

(i) $K = 30$, $\bar{K} = 6$; $\bar{H} = 5 \times H$.

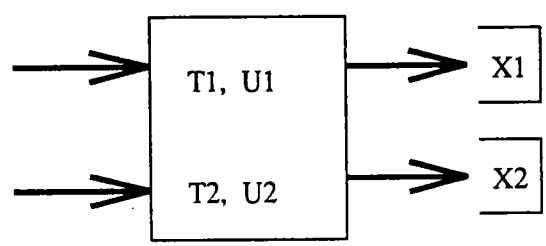


FIG. 2.4 – *Niveau haut*

(ii) La demande de chaque famille pendant une sous-période est égale à la somme des demandes de tous les types de produits (qui appartiennent à la famille) pendant les périodes élémentaires de la sous-périodes:

$$D_1(\kappa) = \sum_{i=1}^{5}\{d_1[5\times(\kappa-1)+i] + d_2[5\times(\kappa-1)+j]\}, \qquad \kappa = 1,\cdots,6$$

$$D_2(\kappa) = \sum_{i=1}^{5}\{d_3[5\times(\kappa-1)+i] + d_4[5\times(\kappa-1)+j]\}, \qquad \kappa = 1,\cdots,6$$

(iii) Il en est de même pour les stocks initiaux:

$$X_1(0) = x_1(0) + x_2(0)$$
$$X_2(0) = x_3(0) + x_4(0)$$

Il n'est pas possible d'évaluer précisément les paramètres $\bar{\beta}_1$, $\bar{\beta}_2$; $\bar{\alpha}_1$, $\bar{\alpha}_2$, car, ils dépendent des quantités de chaque type de produit fabriquées, et ne peuvent donc être obtenus qu'après la résolution du problème de niveau bas qui donne les quantités de production réel de chaque type de produit. Dans [58] ces paramètres sont approximés en considérant les moyennes pondérées par les demandes de chaque type de produits. Dans l'exemple auquel nous nous intéressons, nous avons:

$$\bar{\beta}_1 = \frac{\beta_1 \times \sum_{\kappa=1}^{6}\sum_{i=1}^{5} d_1[5\times(\kappa-1)+i] + \beta_2 \times \sum_{\kappa=1}^{6}\sum_{i=1}^{5} d_2[5\times(\kappa-1)+i]}{\sum_{\kappa=1}^{6}\sum_{i=1}^{5}\{d_1[5\times(\kappa-1)+i] + d_2[5\times(\kappa-1)+i]\}}$$

$$\bar{\beta}_2 = \frac{\beta_3 \times \sum_{\kappa=1}^{6}\sum_{i=1}^{5} d_3[5\times(\kappa-1)+i] + \beta_4 \times \sum_{\kappa=1}^{6}\sum_{i=1}^{5} d_4[5\times(\kappa-1)+i]}{\sum_{\kappa=1}^{6}\sum_{i=1}^{5}\{d_3[5\times(\kappa-1)+i] + d_4[5\times(\kappa-1)+i]\}}$$

qui donne les coûts de rupture estimés pour les familles F_1 et F_2 respectivement, et

$$\bar{\alpha}_1 = \frac{\alpha_1 \times \sum_{\kappa=1}^{6}\sum_{i=1}^{5} d_1[5\times(\kappa-1)+i] + \alpha_2 \times \sum_{\kappa=1}^{6}\sum_{i=1}^{5} d_2[5\times(\kappa-1)+i]}{\sum_{\kappa=1}^{6}\sum_{i=1}^{5}\{d_1[5\times(\kappa-1)+i] + d_2[5\times(\kappa-1)+i]\}}$$

$$\bar{\alpha}_2 = \frac{\alpha_3 \times \sum_{\kappa=1}^{6}\sum_{i=1}^{5} d_3[5\times(\kappa-1)+i] + \alpha_4 \times \sum_{\kappa=1}^{6}\sum_{i=1}^{5} d_4[5\times(\kappa-1)+i]}{\sum_{\kappa=1}^{6}\sum_{i=1}^{5}\{d_3[5\times(\kappa-1)+i] + d_4[5\times(\kappa-1)+i]\}}$$

qui donne les coûts de stockage estimés pour les familles F_1 et F_2 respectivement.

Les temps de fabrication pour les familles sont estimés dans [58] comme suit:

$$\tau_{f,q}^{\kappa} = \frac{z}{z+1-max_{j\in f}|\mathcal{R}_j^q|} \times max_{j\in f, w\in\mathcal{R}_j^q}\, t_{j,w}$$

où z est le nombre des périodes élémentaires dans une sous-période qui est, dans notre cas, $\bar{H}/H$;

$t_{j,w}$ est le temps nécessaire pour exécuter l'opération $< P_j, M_w >$;

$\tau^{\kappa}_{f,q}$ est le valeur à estimer, i.e., le temps nécessaire pour exécuter la $q^{ème}$ opération de la famille f durant la sous-période κ. $\mathcal{R}^q_j$ est l'ensemble des machines qui effectuent la $q^{ème}$ opération de la famille f auquel P_j appartient. Dans notre cas, nous avons:

$$\begin{aligned} T_1 &= \frac{5}{5+1-2} \times max(t_1, t_2, t_5, t_6) \\ T_2 &= \frac{5}{5+1-1} \times max(t_3, t_4) \end{aligned}$$

Après avoir évalué les paramètres précédents, on peut poser le problème au niveau haut comme suit (cf. problème 2.1):

$$Min \sum_{\kappa=1}^{6} \{ \sum_{i=1}^{2} [\bar{\alpha}_i \times \bar{e}_i(\kappa) + \bar{\beta}_i \times \bar{f}_i(\kappa)] \} \tag{2.19}$$

sous les contraintes suivantes (pour tout $\kappa = 1, \cdots, 6,\ i = 1, \cdots, 2$):

$$\begin{aligned} X_i(\kappa) = X_i(\kappa - 1) + U_i(\kappa) - D_i(\kappa), \\ \sum_{i=1}^{2} \{T_i \times U_i(\kappa)\} \leq \bar{H}, \\ \bar{e}_i(\kappa) - X_i(\kappa) \geq 0, \\ \bar{f}_i(\kappa) + X_i(\kappa) \geq 0, \\ U_i(\kappa) \geq 0, \\ \bar{e}_i(\kappa) \geq 0, \\ \bar{f}_i(\kappa) \geq 0, \end{aligned}$$

2.6 Passage du niveau haut au niveau bas

Après avoir construit le modèle du niveau haut et avoir obtenu la planification à ce niveau, on peut utiliser les résultats de cette planification comme contraintes pour le problème du niveau bas. La planification (niveau haut) consiste à déterminer la quantité de chaque famille fabriquée dans chaque cellule durant chaque sous-période. Les décisions prises au niveau bas concernent la production de chaque type de produit sur chaque machine durant chaque période élémentaire. Nous résolvons le problème du niveau bas en deux étapes: *désagrégation des familles*, puis *désagrégation temporelle et spatiale* [58].

2.6.1 Désagrégation des familles de produits

Puisque nous avons deux familles de produits, nous devons résoudre deux problèmes de programmation linéaire pour effectuer cette désagrégation.

Soit $y_i(\kappa)$, $i = 1, \cdots, 4$; $\kappa = 1, \cdots, 6$, la proportion de produits de type P_i dans les familles F_1 ou F_2 pendant la sous-période κ. On note aussi $z_i(k)$, $i = 1, \cdots$; $\kappa = 1, \cdots, 6$ le niveau du stock de produit P_i à la fin de la sous-période κ. Résoudre les deux problèmes ci-dessous conduit à la détermination de $y_i(\kappa)$, $i = 1, \cdots, 4$; $\kappa = 1, \cdots, 6$.

$$Min \sum_{\kappa=1}^{6} \{\sum_{i=1}^{2} [\alpha_i \times e_i(\kappa) + \beta_i \times f_i(\kappa)]\} \tag{2.20}$$

sous les contraintes suivantes (pour tout $\kappa = 1, \cdots, 6$, $i = 1, 2$):

$$x_1(\kappa) = x_1(\kappa - 1) + y_1(\kappa) \times U_1(\kappa) - \sum_{j=1}^{5} d_1(5 \times (\kappa - 1) + j) \tag{2.21}$$

$$x_2(\kappa) = x_2(\kappa - 1) + y_2(\kappa) \times U_1(\kappa) - \sum_{j=1}^{5} d_2(5 \times (\kappa - 1) + j) \tag{2.22}$$

$$y_1(\kappa) + y_2(\kappa) = 1 \tag{2.23}$$

$$e_i(\kappa) \geq x_i(\kappa) \tag{2.24}$$

$$f_i(\kappa) \geq -x_i(\kappa) \tag{2.25}$$

$$y_i(\kappa) \geq 0 \tag{2.26}$$

$$e_i(\kappa) \geq 0 \tag{2.27}$$

$$f_i(\kappa) \geq 0 \tag{2.28}$$

L'objectif (2.20) consiste à minimiser la somme des coûts de stockage et de rupture des produits P_1 et P_2 (qui appartiennent à la famille F_1) sur l'horizon de planification, i.e., sur 6 sous-périodes dans notre cas. Les contraintes (2.21) et (2.22) concernent les niveaux des stocks X_1 et X_2 relatifs à P_1 et P_2 respectivement. Les contraintes (2.23) indiquent que la somme des ratios de P_1 et P_2 dans la famille F_1 est égale à 1. Les inégalités (2.24) à (2.28) sont des contraintes introduites pour transformer le problème initial en un problème de programmation linéaire.

$$Min \sum_{\kappa=1}^{6} \{\sum_{i=3}^{4} [\alpha_i \times e_i(\kappa) + \beta_i \times f_i(\kappa)]\} \tag{2.29}$$

sous les contraintes suivantes (pour tout $\kappa = 1, \cdots, 6$, $i = 3, \cdots, 4$):

$$x_3(\kappa) = x_3(\kappa - 1) + y_3(\kappa) \times U_2(\kappa) - \sum_{j=1}^{5} d_3(5 \times (\kappa - 1) + j) \tag{2.30}$$

$$x_4(\kappa) = x_4(\kappa - 1) + y_4(\kappa) \times U_2(\kappa) - \sum_{j=1}^{5} d_4(5 \times (\kappa - 1) + j) \tag{2.31}$$

$$y_3(\kappa) + y_4(\kappa) = 1 \tag{2.32}$$

$$e_i(\kappa) \geq x_i(\kappa) \tag{2.33}$$

$$f_i(\kappa) \geq -x_i(\kappa) \tag{2.34}$$

$$y_i(\kappa) \geq 0 \tag{2.35}$$

$$e_i(\kappa) \geq 0 \tag{2.36}$$

$$f_i(\kappa) \geq 0 \tag{2.37}$$

Les problèmes (2.29) et (2.20) sont semblables. Le problème (2.29) concerne les produits P_3 et P_4 et le problème (2.20) concerne les produits P_1 et P_2.

2.6.2 Désagrégation temporelle et spatiale

Après avoir obtenu la planification de niveau haut et déterminé les ratios des produits dans les familles, nous passons à la planification de niveau bas. L'objectif est de déterminer la quantité de chaque type de produit à fabriquer pendant chaque période élémentaire sur les machines qu'ils utilisent. La résolution du problème conduit à la détermination des $u_i(l)$, $i = 1.\cdots, 6$; $l = 1, \cdots, 30$.

Le problème s'écrit:

$$Min \sum_{j=5\times(\kappa-1)+1}^{5\times(\kappa-1)+5} \{\sum_{i=1}^{6} \alpha_i \times e_i(j) + \sum_{i=1}^{4} \beta_i \times f_i(j)\} \tag{2.38}$$

sous les contraintes suivantes (pour tout $\kappa = 1, \cdots, 6$; $j = 5 \times (\kappa - 1) + 1, \cdots, 5 \times (\kappa - 1) + 5$):

$$x_i(j) = x_i(j-1) + u_i(j) - d_i(j), \quad i = 1, 2, 3, 4 \tag{2.39}$$

$$x_5(j) = x_5(j-1) + u_5(j) - u_1(j) \tag{2.40}$$

$$x_6(j) = x_6(j-1) + u_6(j) - u_2(j) \tag{2.41}$$

$$t_1 \times u_1(j) + t_2 \times u_2(j) + t_4 \times u_4(j) \leq H \tag{2.42}$$

$$t_3 \times u_3(j) + t_5 \times u_5(j) + t_6 \times u_6(j) \leq H \tag{2.43}$$

$$\sum_{j=5\times(\kappa-1)+1}^{5\times(\kappa-1)+5} u_i(j) = y_i(\kappa) \times U_1(\kappa), \qquad i = 1, 2 \tag{2.44}$$

$$\sum_{j=5\times(\kappa-1)+1}^{5\times(\kappa-1)+5} u_i(j) = y_i(\kappa) \times U_2(\kappa), \qquad i = 3, 4 \tag{2.45}$$

$$0 \leq u_1(j) \leq x_5(j-1) \tag{2.46}$$

$$0 \leq u_2(j) \leq x_6(j-1) \tag{2.47}$$

$$u_3(j), u_4(j), u_5(j), u_6(j) \geq 0 \tag{2.48}$$

$$e_i(j) \geq x_i(j), \qquad i = 1, \cdots, 6 \tag{2.49}$$

$$f_i(j) \geq -x_i(j), \qquad i = 1, \cdots, 4 \tag{2.50}$$

Il s'agit ici (fonction objectif (2.38)) de minimiser la somme des coûts de stockage et de rupture des produits P_1 à P_4 sur l'horizon de planification. Les contraintes (2.39) à (2.41) concernent les stocks B_1 à B_6. Les contraintes (2.42) et (2.43) sont les contraintes de capacité des machines M_1 et M_2 respectivement. Les contraintes (2.44) et (2.45) sont les contraintes provenant du niveau haut. Les contraintes (2.46) à (2.50) sont introduites pour transformer le problème en un problème de programmation linéaire.

2.7 Résultats numériques

On a résolu les problèmes précédents à l'aide du logiciel OSL[2].

2.7.1 Valeurs des paramètres

On choisit les valeurs des paramètres comme suit:

- $K = 30$ est l'horizon de planification contient 30 périodes élémentaires;
- $H = 30$ est la durée de chaque période élémentaire est de 30 unités de temps;
- les temps nécessaires pour exécuter les opération sont donnés dans le tableau 2.1;

t_1	t_2	t_3	t_4	t_5	t_6
1.1	1.3	1.8	2.0	1.7	1.4

TAB. 2.1 - *Temps opératoires*

- $x_i(0) = 10,\ i = 1, \cdots, 6$ sont les stocks initiaux. Ils sont tous égaux à 10.

2.7.2 Résultats et conclusions

On considère la demande de chaque type de produit comme aléatoire. La loi de distribution uniforme, $U[a, b]$, est appliquée à la demande. On désigne la valeur moyenne de cette distribution par ϖ. Selon la sélection des valeurs de a et b, qui

[2] OSL (*Optimisation Subroutine Library*) est un logiciel commercial de IBM. Voir *Optimisation Subroutine Library*, **Guide and Reference**, Release 2, Third Edition, July 1991

sont respectivement les bornes inférieures et supérieures, les demandes peuvent être générées afin que les machines travaillent en surcharge ou en sous-charge.

Dans les Figures 2.5 à 2.9, la coordonnée horizontale représente la valeur des paramètres α_5 et α_6 (on suppose que $\alpha_5 = \alpha_6$), la coordonnée verticale représente la valeur prise par la fonction objectif. Les Figures 2.5 à 2.9 donnent des résultats des cinq groupes d'essais.

- Figure 2.5:
  - les coûts de stockage dans les stocks $B_1, \cdots, B_4$ sont $\alpha_1 = \cdots = \alpha_4 = 10$.
  - les demandes $d_1, \cdots, d_4$ pour les produits $P_1, \cdots, P_4$ sont:

	d_1	d_2	d_3	d_4
ϖ	3.4	3.4	3.3	4.9

TAB. 2.2 – *Essai I: Demandes pour* $P_1, \cdots, P_4$

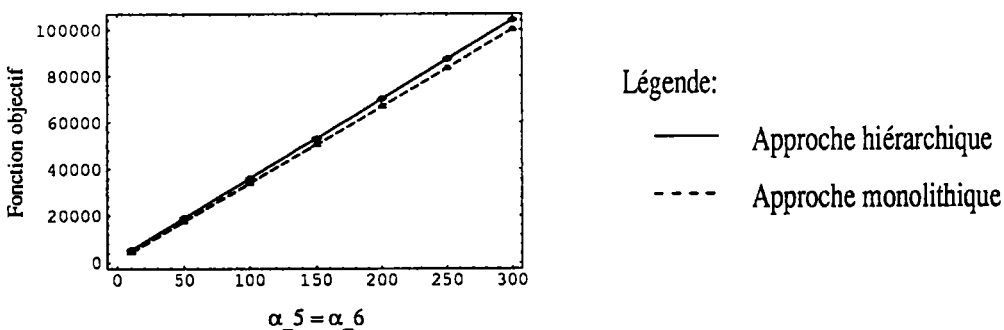


FIG. 2.5 – *Essai I*

- Figure 2.6:
  - les coûts de stockage dans $B_1, \cdots, B_4$ sont $\alpha_1 = \cdots = \alpha_4 = 100$.
  - les demandes $d_1, \cdots, d_4$ pour les produits $P_1, \cdots, P_4$ sont:

	d_1	d_2	d_3	d_4
ϖ	3.4	3.4	3.3	4.9

TAB. 2.3 – *Essai II: Demandes pour* $P_1, \cdots, P_4$

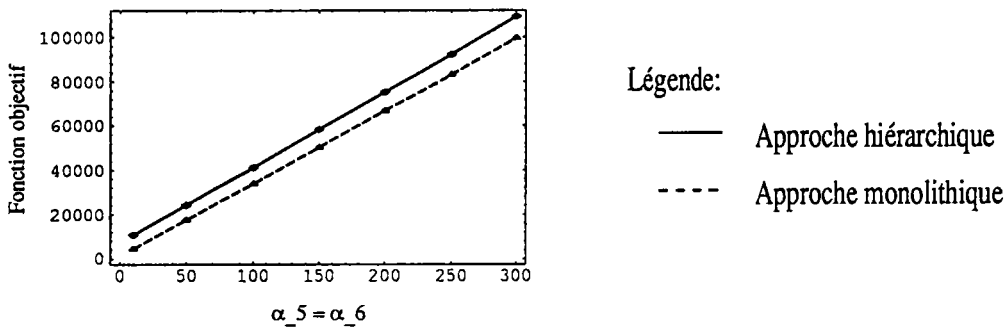


FIG. 2.6 – *Essai II*

- Figure 2.7:
  - les coûts de stockage dans $B_1, \cdots, B_4$ sont $\alpha_1 = \cdots = \alpha_4 = 10$.
  - les demandes $d_1, \cdots, d_4$ pour les produits $P_1, \cdots, P_4$ sont:

	d_1	d_2	d_3	d_4
ϖ	5.4	4.9	4.7	7.8

TAB. 2.4 – *Essai III: Demandes pour* $P_1, \cdots, P_4$

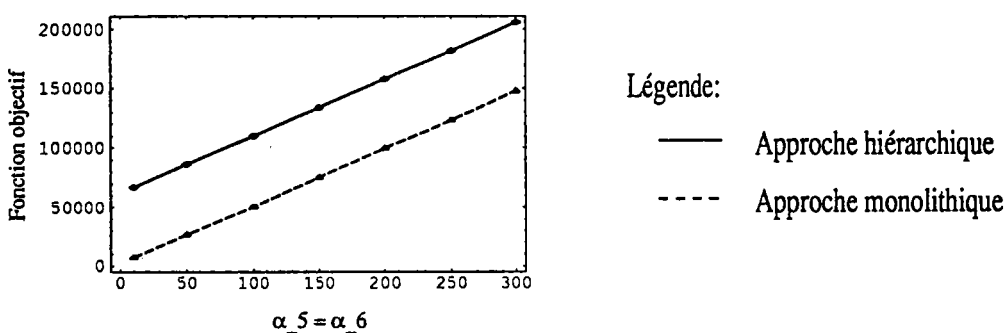


FIG. 2.7 – *Essai III*

- Figure 2.8:
  - les coûts de stockage dans $B_1, \cdots, B_4$ sont $\alpha_1 = \cdots = \alpha_4 = 100$.
  - les demandes $d_1, \cdots, d_4$ pour les produits $P_1, \cdots, P_4$ sont:

	d_1	d_2	d_3	d_4
ϖ	5.4	4.9	4.7	7.8

TAB. 2.5 – *Essai IV: Demandes pour* $P_1, \cdots, P_4$

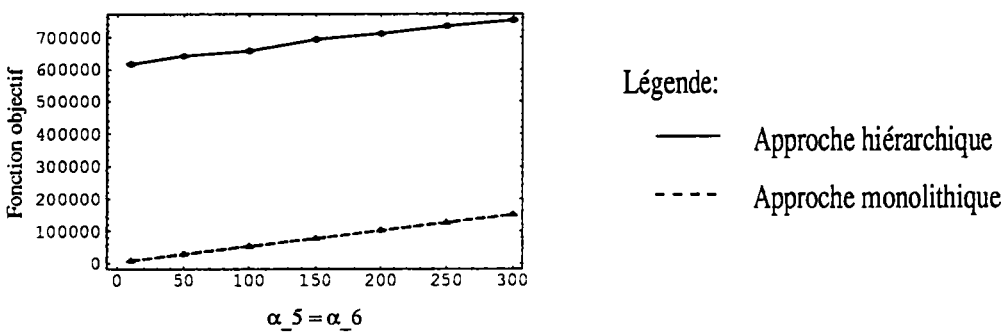


FIG. 2.8 – *Essai IV*

- Figure 2.9:
  - les coûts de stockage dans $B_1, \cdots, B_4$, et les demandes $d_1, \cdots, d_4$ sont les mêmes que pour l'essai IV.
  - on ne fixe pas les temps opératoires du niveau haut, mais on les ajuste itérativement comme dans Mehra [89].

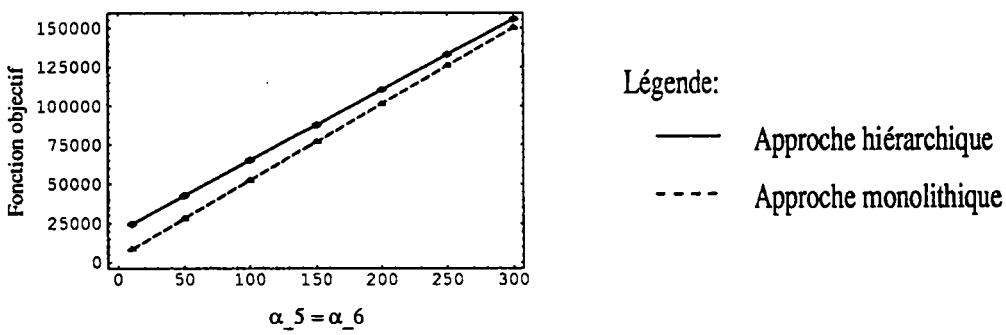


FIG. 2.9 – *Essai V*

2.8 Conclusions

Dans les résultats que nous avons obtenus, nous constatons que:

- l'approche hiérarchique, avec l'estimation conservative du temps au niveau haut, s'adapte bien au cas où les machines travaillent en sous-charge, c'est le cas de l'essai I (Figure 2.5) et de l'essai II (Figure 2.6).
- l'approche hiérarchique, avec l'estimation conservative du temps au niveau haut, se comporte mal lorsque les machines travaillent en surcharge, c'est le cas de l'essai III (Figure 2.7) et de l'essai IV (Figure 2.8).
- dans le cas où le temps estimé au niveau haut évolue itérativement [89], l'approche hiérarchique converge asymptotiquement vers l'approche monolithique (voir Figure 2.9).

Chapitre 3

Réseaux à sorties contrôlables et leur simplification

3.1 Introduction

Dans ce chapitre, nous proposons un outil pour rapprocher des méthodes utilisées pour modéliser et analyser les systèmes de taille importante.

Les résultats que nous avons obtenus constituent la partie modélisation des travaux qui se déroulent autour des systèmes à fonctionnement non cyclique. Leur évaluation interviendra ultérieurement. Plus précisément, nous proposons une classe de réseaux de Petri, que nous appelons les *réseaux de Petri à sorties contrôlables*[1], qui permettent de représenter les modules classiques des systèmes de production. Les réseaux de Petri à sorties contrôlables possèdent les propriétés qualitatives souhaitables lorsqu'il s'agit d'un système de production. Ils sont consistants, vivants et réversibles. Nous présentons également des conditions peu restrictives sous lesquelles l'intégration des modèles des modules préserve ces propriétés. Finalement, nous introduisons les *T-invariants réduits*[2] qui permettent d'obtenir les modèles agrégés des modules dans l'évaluation quantitative du système intégré à l'aide d'une approche hiérarchique.

3.2 Les réseaux de Petri à sorties contrôlables

3.2.1 Définitions

Le concept d'un réseau de Petri à sorties contrôlables (voir la définition A.1 de l'annexe A) étant nouveau et essentiel dans notre travail, nous le rappelons ci-dessous.

---

[1] Pour plus de détails, voir l'annexe A

[2] Pour plus de détails, voir l'annexe B

Définition 3.1 (réseaux de Petri à sorties contrôlables) *Un réseau de Petri $G = (P \cup R, T, F, M_0)$, où P et R sont deux ensembles disjoints de places appelées respectivement places de process et places de ressource, est dit réseau de Petri à sorties contrôlables si les conditions suivantes sont vérifiées:*

H1. $(t,r) \in F \Longleftrightarrow (r,t) \in F, \forall t \in T, \forall r \in R$

H2. $M_0(r) \geq 1, \forall r \in R$

H3. Le sous-réseau $G' = (P, T, F', M'_0)$, où F' est la restriction de F à $(P \times T) \cup (T \times P)$, et M' la restriction de M_0 à P, est un graphe acyclique.

H4. Les extrémités de G sont des transitions. Soit T_{in} l'ensemble des transitions d'entrée et T_{out} l'ensemble des transitions de sortie, i.e.,

$$T_{in} = \{t \mid {}^\bullet t = \emptyset,\ t \in T\}, \qquad T_{out} = \{t \mid t^\bullet = \emptyset,\ t \in T\}$$

H5. G est couvert par un ensemble de T-invariants tels que chacun d'eux correspond à une seule transition de sortie, i.e., $\forall t \in T_{out}$, $\exists y_t \in \mathbf{N}^{|T|}$, tel que

$$C\ y_t = 0.\ \| y_t \| \cap T_{out} = \{t\},\ et \bigcup_{t \in T_{out}} \| y_t \| = T.$$

où C représente la matrice d'incidence du réseau de Petri G.

La condition $H5$ nous a conduit à utiliser le terme "réseau de Petri à sortie contrôlables". Compte tenu du fait que chaque transition de sortie correspond à un T-invariant dont le support ne contient pas d'autres transitions de sortie, le franchissement de chaque transition de sortie peut être complètement indépendant des franchissements des autres transitions de sortie. Par conséquent, les sorties d'un système représenté par un réseau de Petri à sorties contrôlables peuvent être contrôlées avec une grande flexibilité. Cela correspond à la réalité des systèmes de production: on peut décider de fabriquer un type de produits sans pour autant être obligé de fabriquer d'autres types de produits simultanément.

La théorème suivant montre que les réseaux de Petri à sorties contrôlables possèdent des propriétés importantes pour un système de production[3].

Théorème 3.1 *Un réseau de Petri à sorties contrôlables G est consistant, vivant, réversible mais non borné quels que soient les nombres de jetons situés initialement dans les places de process, i.e., dans P.*

La démonstration du théorème 3.1 est fournie dans l'annexe A.

La figure 3.1 donne trois réseaux de Petri[4]. Les conditions $H1 - H4$ sont vérifiées pour ces trois réseaux de Petri. Le RdP (a) n'est pas à sorties contrôlables

[3] Pour plus de détails, voir section A.1.2

[4] Un autre exemple se trouve page 69

car son seul T-invariant contient $t2$ et $t3$. Le RdP (b) n'est pas à sorties contrôlables non plus car il n'est pas consistant. Le RdP (c) est à sorties contrôlables car il existe deux T-invariants $\{2*t1,\ t2,\ t3\ ,t5\ ,t6\ \}$ et $\{t1,\ 2*t2,\ t4,\ t5,\ t7\}$ vérifiant la condition $H5$.

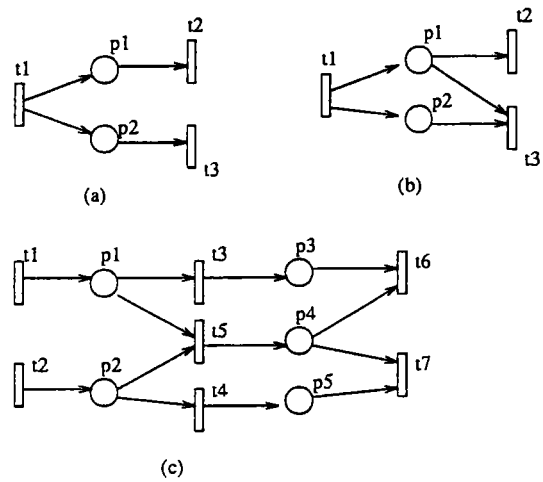


FIG. 3.1 - *Trois réseaux de Petri*

Dans la suite de ce paragraphe, nous montrons comment identifier des réseaux de Petri à sorties contrôlables. Pour un réseau de Petri donné, les conditions $H1 - H4$ ne sont pas difficiles à vérifier. Les conditions $H1$ et $H2$ peuvent être vérifiées place après place. Si le réseau de Petri obtenu par suppression des places de ressource ne contient ni place source, ni place puits, la condition $H4$ est vérifiée. La condition $H3$ peut être vérifiée à l'aide d'une approche classique utilisée pour déterminer les circuits dans les graphes. Il nous reste à vérifier la condition $H5$.

Afin de vérifier la condition $H5$, nous déterminons, pour chaque transition de sortie $t \in T_{out}$, un T-invariant y_t dont le support ne contient pas d'autre transition de sortie que t et dont le support est maximal. Si l'union des supports de ces T-invariants couvre le réseau de Petri G, alors la condition $H5$ est vérifiée. Sinon, il n'est pas à sorties contrôlables.

Un tel T-invariant y_t s'obtient en résolvant le problème de programmation linéaire suivant:

$$Max \sum_{\tau \in T} z_\tau$$

sous les contraintes:

$$C \times y_t = 0$$
$$y_t[s] = 0, \quad \forall s \in T_{out} - \{t\}$$
$$z_\tau \leq y_t[\tau], \quad \forall \tau \in T$$
$$0 \leq z_\tau \leq 1 \ et \ y_t[\tau] \geq 0, \quad \forall \tau \in T$$

Finalement, les théorèmes suivants donnent deux classes de conditions pour qu'un RdP soit à sorties contrôlables. Le RdP (c) de la figure 3.1 appartient aux deux classes.

Théorème 3.2 *Un réseau de Petri consistant vérifiant les conditions $H1 - H4$ est à sorties contrôlables si chaque T-invariant à support minimal concerne une seule transition de sortie.*

Théorème 3.3 *Un réseau de Petri consistant vérifiant les conditions $H1 - H4$ est à sorties contrôlables si chaque transition du sous-réseau G', obtenu en supprimant les places de ressource, possède au plus une place de sortie.*

Les démonstrations des théorèmes 3.2 et 3.3 sont fournies dans l'annexe A.

3.2.2 Intégration des modules

Dans ce paragraphe, nous supposons que les modules sont modélisés à l'aide de réseaux de Petri à sorties contrôlables. L'objectif est de présenter un mécanisme d'intégration de modules qui préserve les propriétés qualitatives des modules, de façon à ce que les modèles RdP finaux possèdent les propriétés souhaitées pour un système de production.

Le mécanisme d'intégration fait appel à un ensemble de places, que nous appelons places d'interface, qui permettent de relier les modèles des modules.

Considérons un ensemble de modules $G_1, \cdots, G_n$, où G_i est un réseau de Petri à sorties contrôlables. Un système intégré G est un réseau de Petri obtenu par intégration des modules $G_1, \cdots, G_n$ à l'aide d'un ensemble de places d'interface Q et d'arcs inter-modules Γ de la manière suivante:

- l'ensemble des places de G est l'union de l'ensemble de places de $G_1, \cdots, G_n$ et Q;
- l'ensemble de transitions de G est l'union de l'ensemble de transitions de $G_1, \cdots, G_n$;
- l'ensemble des arcs de G est l'union de l'ensemble des arcs de $G_1, \cdots, G_n$ et Γ.

Nous faisons les hypothèses suivantes au niveau de l'intégration. Elles sont notées $H6$ et $H7$ et analysées dans l'annexe A:

H6: Chaque place d'interface n'est ni place de source ni place de puits;

H7: Chaque transition de sortie des modules est reliée à au plus une place d'interface.

Pour un système acyclique, nous avons:

Théorème 3.4 *Un système acyclique G, obtenu par intégration des réseaux de Petri à sorties contrôlables sous les conditions $H6$ et $H7$, est vivant, consistant, réversible mais non borné.*

Cette propriété montre que l'intégration des modules des réseaux de Petri à sorties contrôlables préserve toutes les propriétés des modèles des modules. Cela est naturel compte tenu du résultat suivant:

Théorème 3.5 *Le système intégré G est aussi un réseau de Petri à sorties contrôlables.*

Les démonstrations des théorèmes ci-dessus se trouvent dans l'annexe A.

3.3 Simplification des réseaux de Petri

Le modèle RdP d'un système complexe ayant été construit selon l'approche modulaire décrite précédemment. nous nous intéressons dans ce paragraphe à la simplification des modèles des modules afin d'obtenir des modèles permettant l'évaluation du comportement du système intégré à l'aide d'une approche hiérarchique. Le modèle simplifié proposé ci-dessous préserve la relation d'entrée-sortie d'un module du point de vue des flux de jetons. La simplification est basée sur les **T-invariants réduits**. Les propriétés des T-invariants réduits, leur calcul et leur utilisation pour la simplification d'un module sont également présentés ci-dessous.

3.3.1 Les T-invariants réduits et leurs propriétés

Définition 3.2 (T-invariants réduits) *Soit $G = (P, T, F, W, M_0)$ le modèle RdP d'un module, soit C sa matrice d'incidence, $S = T_{in} \cup T_{out}$ l'ensemble des transitions d'entrée/sortie. Un vecteur Z de dimension $|S|$ est dit T-invariant réduit si il existe un vecteur $Y \in \mathbb{R}^{|T|}$ avec $Y \geq 0$ tel que $CY = 0$ et $Y[t] = Z[t]$, $\forall t \in S$.*

Bien que les T-invariants réduits soient des vecteurs à composantes réelles et que les T-invariants soient des vecteurs à composantes entières, nous avons le résultat suivant:

Théorème 3.6 *Pour tout T-invariant réduit Z, il existe un T-invariant Y de G et un nombre réel α tel que $Y[t] = \alpha \times Z[t]$, $\forall t \in S$.*

De plus,

Théorème 3.7 *L'ensemble des T-invariants réduits forme un cône convexe.*

Les démonstrations des théorèmes 3.6 et 3.7 sont fournies dans l'annexe B.

Selon les propriétés des cônes convexes, nous pouvons définir les familles génératrices et les familles génératrices minimales des T-invariants réduits.

Définition 3.3 (famille génératrice) *Un ensemble de T-invariants réduits* $\{Z_1, Z_2, \cdots, Z_k\}$ *est dit famille génératrice si chaque T-invariant réduit* Z *peut être écrit comme une combinaison linéaire non-négative de* $Z_1, Z_2, \cdots, Z_k$, *i.e., si il existe des nombres non-négatifs* $a_1, a_2, \cdots, a_k$ *tels que* $Z = a_1 \times Z_1 + a_2 \times Z_2 + \cdots + a_k \times Z_k$.

Définition 3.4 (famille génératrice minimale) *Une famille génératrice* $\{Z_1, Z_2, \cdots, Z_k\}$ *est dite minimale si aucun des T-invariants réduits* $Z_1, Z_2, \cdots, Z_k$ *ne peut être exprimé en combinaison linéaire non-négative des autres.*

La famille génératrice minimale est unique selon la propriété suivante:

Théorème 3.8 *Soient* $\{Z_1, Z_2, \cdots, Z_k\}$ *et* $\{W_1, W_2, \cdots, W_l\}$ *deux familles génératrices minimales des T-invariants réduits. Alors,* $k = l$ *et pour chaque* Z_i, *il existe un nombre positif* x_i *et un* $W_{I(i)}$ *tels que* $Z_i = x_i \times W_{I(i)}$.

La démonstration du théorème 3.8 se trouve dans l'annexe B.

3.3.2 La simplification des modules

Étant donné le modèle réseau de Petri d'un module $G = (P, T, F, W, M_0)$ avec transitions d'entrée/sortie $S = T_{in} \cup T_{out}$, la simplification commence par la détermination de la famille génératrice minimale $\{Z_1, Z_2, \cdots, Z_k\}$ des T-invariants réduits. Le modèle simplifié ou agrégé peut alors être défini. C'est un réseau de Petri généralisé $G^* = \{P_{in} \cup P_{out}, T_{in} \cup T_{out} \cup Q, F^*, W^*\}$ où $|P_{in}| = |T_{in}|, |P_{out}| = |T_{out}|, |Q| = k$, F^* est l'ensemble des arcs reliant les places aux transitions et les transitions aux places, et W^* est une fonction qui associe à chaque arc un poids. La définition de G^* consiste à:

- créer une place $p \in P_{in}$ pour chaque transition d'entrée $t \in T_{in}$ et relier t à p à l'aide d'un arc de poids 1;
- créer une place $p \in P_{out}$ pour chaque transition de sortie $t \in T_{out}$ et relier p à t à l'aide d'un arc de poids 1;
- créer une transition t_i, pour chaque T-invariant réduit Z_i;
- relier toute place $p \in P_{in}$ telle que $Z_i[{}^\bullet p] > 0$ à t_i à l'aide d'un arc de poids $Z_i[{}^\bullet p]$;

- relier t_i à toute place $p \in P_{out}$ telle que $Z_i[p^\bullet] > 0$ à l'aide d'un arc de poids $Z_i[p^\bullet]$.

Il est aisé de démontrer que la famille génératrice des T-invariants réduits du modèle simplifié est identique à celle du modèle RdP du module. De plus, dans le modèle simplifié, chaque T-invariant à support minimal contient une seule transition intermédiaire, et l'ensemble de ces T-invariants correspond à la famille génératrice minimale des T-invariants réduits.

3.3.3 Détermination de la famille génératrice des T-invariants réduits

Dans ce paragraphe, nous supposons que le modèle réseau de Petri G à considérer est acyclique. Le calcul est similaire pour les réseaux de Petri généraux.

Supposons que $S = \{t_1, t_2, \cdots, t_r\}$ et $T = \{t_1, t_2, \cdots, t_r, t_{r+1}, \cdots, t_n\}$. Bien entendu, $n \geq r$. Considérons la matrice $D = [d_{ij}]_{r \times n}$ avec $d_{ii} = 1, \forall 1 \leq i \leq r$ et $d_{ij} = 0, \forall i \neq j$. Elle permet de déterminer le T-invariant réduit Z d'un T-invariant Y comme suit:

$$Z = D \times Y$$

Il a été démontré dans [88] que l'ensemble des T-invariants à support minimal forme une famille génératrice minimale de l'ensemble des vecteurs $Y \in \mathbb{R}^{|T|}$ avec $Y \geq 0$ tel que $C \times Y = 0$. Nous avons le résultat suivant:

Propriété 3.1 *Soit $\Phi = \{Y_1, Y_2, \cdots, Y_q\}$ l'ensemble des T-invariants à support minimal de G. Alors, $\{D \times Y_1, D \times Y_2, \cdots, D \times Y_q\}$ est une famille génératrice des T-invariants réduits.*

La démonstration est fournie dans l'annexe B.

Ce résultat suggère une méthode appelée méthode I de détermination de la famille génératrice. Elle consiste à:

(i) déterminer l'ensemble des T-invariants à support minimal;

(ii) dériver une famille génératrice en appliquant le résultat 3.1;

(iii) supprimer les T-invariants réduits qui peuvent être exprimés comme une combinaison linéaire non négative des autres T-invariants réduits.

La faiblesse de la méthode I provient du nombre important des T-invariants à support minimal. On a montré que ce nombre croît de manière exponentielle lorsque le nombre de transitions augmente. Cependant, pour les modules réalistes, la famille génératrice est composée d'un nombre faible de T-invariants réduits. Dans la suite, nous développons une méthode appelée méthode II de détermination de la famille génératrice sans passer par les T-invariants à support minimal de G.

La méthode II se déroule en quatre étapes (un exemple est proposé dans l'annexe B pour illustrer ces étapes):

- La première étape consiste à regrouper les sommets par niveaux de sommets de telle sorte qu'il n'existe pas de chemin reliant un sommet du niveau i à un sommet du niveau j lorsque $j > i$. Pour cela, il suffit d'associer à chaque sommet x le niveau $l(x)$, où $l(x)$ est le nombre de sommets sur le plus long chemin reliant x à une transition de sortie. Puisqu'un réseau de Petri est un graphe biparti, $l(x)$ est un nombre impair si x est une transition, et $l(x)$ est un nombre pair si x est une place. Par convention, $l(x) = 1$ pour toute transition de sortie $x \in T_{out}$ et $l(x) = 2K + 1$ pour toute transition d'entrée $x \in T_{in}$ où $2K + 1$ est le nombre total de niveau.

- La deuxième étape consiste à remplacer chaque arc, reliant un sommet x du niveau i à un sommet y du niveau j avec $i > j+1$, par un chemin élémentaire allant de x à y qui contient $(i - j - 1)/2$ transitions supplémentaires et $(i - j - 1)/2$ places supplémentaires. Cette transformation est illustrée par la figure B.6 de l'annexe B (page 95). Elle préserve les T-invariants. De plus, dans le réseau de Petri obtenu, chaque arc relie un sommet de niveau donné à un sommet de niveau immédiatement inférieur.

- La troisième étape consiste à: (i) extraire le sous-réseau de Petri $G(3)$ composé des sommets des niveaux $i \leq 3$, (ii) déterminer la famille génératrice des T-invariants réduits de $G(3)$, et (iii) construire le modèle simplifié $G^*(3)$ de $G(3)$.

- La quatrième étape est un processus itératif. Elle consiste, pour $I = 5, 7, \cdots, 2K + 1$, à: (i) extraire le sous-réseau de Petri $G(I)$ composé des sommets de niveaux $i \leq I$, (ii) dériver un autre réseau de Petri $G'(I)$ en remplaçant le sous-réseau $G(I - 2)$ par son modèle simplifié $G^*(I - 2)$, (iii) déterminer la famille génératrice minimale des T-invariants réduits de $G'(I)$, et (iv) construire le modèle simplifié $G^*(I)$ de $G'(I)$ qui est également, selon le théorème 3.9, le modèle simplifié de $G(I)$.

Théorème 3.9 *Considérons un sous-réseau $G(I)$, avec $I \in \{5, 7, , \cdots, 2K + 1\}$, composé des sommets de niveaux $i \leq I$ et le réseau de Petri $G'(I)$ dérivé de $G(I)$ en remplaçant le sous-réseau $G(I - 2)$ par son modèle simplifié $G^*(I - 2)$. Un vecteur à composantes réelles est un T-invariant réduit de $G(I)$ si et seulement si il est un T-invariant réduit de $G'(I)$.*

La démonstration se trouve dans l'annexe B.

Selon ce théorème, la famille génératrice minimale des T-invariants réduits de $G(2K + 1)$ est celle du réseau de Petri initial G, et le modèle simplifié $G^*(2K + 1)$ est le modèle simplifié de G.

Plus de détails sur la simplification se trouvent dans l'annexe B.

Chapitre 4

Modélisation modulaire et gestion hiérarchisée

Nous nous intéressons à la modélisation et à l'évaluation des systèmes non cycliques (Pour plus de détails, voir annexe C). Les systèmes de ce genre se caractérisent par le fait qu'ils sont appelés à répondre à des demandes exogènes qui évoluent dans le temps. On les appelle encore *on-line* pour souligner le fait qu'ils évoluent en contact permanent avec le monde extérieur à l'entreprise [105] [101].

Nous nous contenterons de modéliser les gammes de fabrication des produits à l'aide des réseaux de Petri élémentaires, puis d'utiliser les propriétés de ce modèle pour proposer une approche modulaire qui permet de traiter des systèmes de grande taille. Cette approche modulaire conduit naturellement à une gestion hiérarchisée de **la planification à court terme** et de **l'ordonnancement.**

L'objectif est de décider de la quantité de chaque type de produit à fabriquer durant chaque période élémentaire en tenant compte de la capacité globale du système, puis de décider, à l'intérieur de la première période élémentaire, des ressources qui vont exécuter les différentes opérations et des instants de début de ces opérations.

La première phase, qui s'intéresse à établir les quantités de produits à fabriquer durant chacune des périodes élémentaires, est **la planification à court terme.** Son calcul se basé sur les capacités globales des ressources et ne tient pas compte des ordres de passage des produits sur les machines. C'est la raison pour laquelle on fait intervenir dans ces calculs une durée de travail qui est inférieure à la durée d'une période élémentaire: ainsi, les ressources seront en sous-charge durant les périodes élémentaires, ce qui donnera la souplesse nécessaire pour que la seconde phase aboutisse.

La seconde phase, **l'ordonnancement**, qui s'intéresse à la première période élémentaire, consiste à affecter les opérations aux ressources et à décider du début de chaque opération en tenant compte du fait qu'une ressource exécute au plus une opération à chaque instant. L'objectif est de trouver un ordonnancement qui réalise les fabrications spécifiées par la planification à court terme pour la première

période élémentaire. Il est possible qu'un tel ordonnancement admissible ne puisse être trouvé, soit parce qu'il n'existe pas, soit parce que l'algorithme heuristique utilisé n'a pas permis de l'obtenir. La seule solution est alors de réduire la durée de travail et de relancer la planification à court terme. Bien entendu, cela se fait au détriment de la qualité de la solution et peut conduire à la sous-utilisation des ressources.

La planification à court terme a pour objectif d'optimiser un critère. Il y a plusieurs critères habituellement considérés [105]. Nous utilisons la minimisation de la somme des coûts de stockage et des coûts de rupture.

Pour la planification à court terme à l'horizon H, on utilise l'approche par *horizon glissant*: au début de la seconde période élémentaire, on recommence la planification sur l'horizon H en tenant compte des nouvelles données qui sont intervenues depuis le calcul précédent et des produits déjà ordonnancés sur cet intervalle, on calcule l'ordonnancement sur la première période élémentaire, et ainsi de suite. La Figure 4.1 schématise cette situation [105].

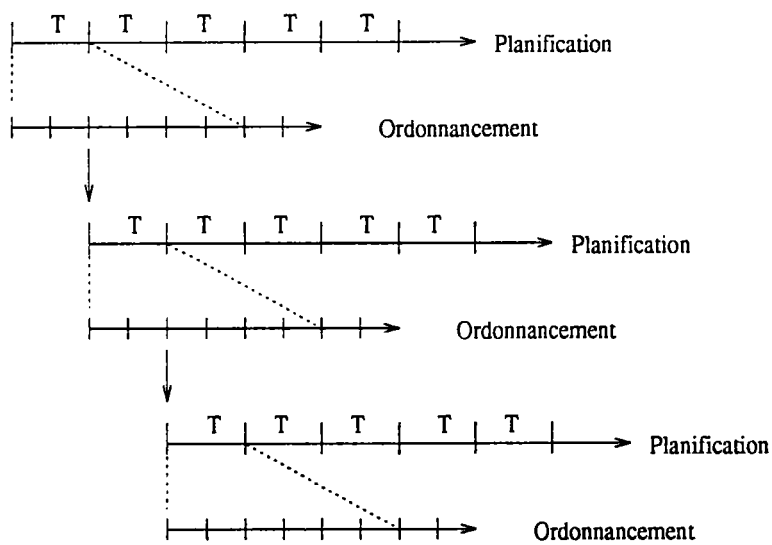


FIG. 4.1 – *Schématisation du processus "horizon glissant"*

4.1 Modélisation à l'aide des RdP

Considérons un système de fabrication capable de fournir plusieurs types de produits. Les quantités commandées par les clients pour la fin de chacune des périodes élémentaires sont connues. Nous supposons qu'une période élémentaire a une durée très supérieure au temps nécessaire pour produire une unité de n'importe quel type. C'est une hypothèse non restrictive dans la pratique.

La fabrication d'une unité de produit se caractérise par les opérations à effectuer, les ressources disponibles pour effectuer ces opérations, et les

temps opératoires. Une opération peut être une *opération de transformation* ou une *opération d'assemblage*. Nous ne considérons pas ici les opérations de désassemblage.

Le modèle utilisé pour la planification est l'ensemble des modèles des gammes de fabrication des produits. La gamme de fabrication d'un produit, du point de vue de la gestion de la production, est la succession des opérations à effectuer avec, pour chacune d'elles, la liste des machines sur lesquelles elle peut être effectuée et le temps nécessaire pour exécuter cette opération sur chacune des ressources.

Prenons un exemple. Considérons un atelier comportant cinq machines ou stations de travail notées M_1, M_2, M_3, M_4, M_5. Supposons que l'atelier fabrique deux types de produits, P_1 et P_2. Supposons encore qu'une unité de produit P_1 ait à subir successivement deux opérations: la première opération peut être exécutée sur M_1 en quatre unités de temps ou sur M_2 en deux unités de temps; la deuxième opération peut être exécutée sur M_3 en trois unités de temps ou sur M_4 en quatre unités de temps. Ce processus est représenté en haut de la Figure 4.2. La gamme de fabrication de produit P_2 est un processus d'assemblage qui est représenté en bas de la Figure 4.2.

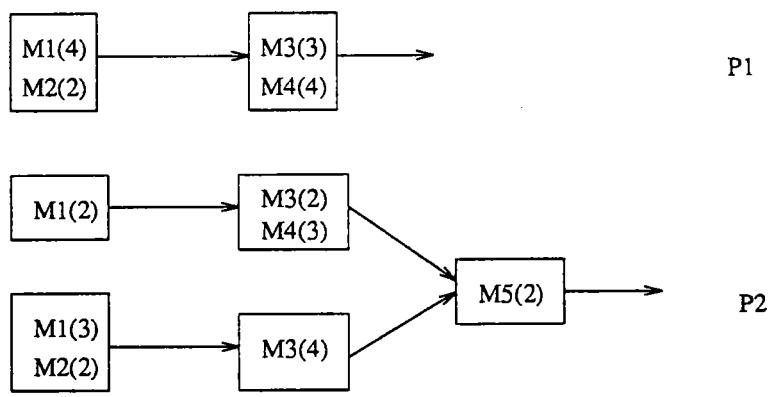


FIG. 4.2 – *Deux gammes de fabrication*

Le RdP correspondant à la gamme de fabrication d'un produit comporte des transitions sources temporisés à zéro. Les franchissements de ces transitions représentent les lancements en fabrication des nouvelles unités de produits. Le RdP correspondant à la gamme de fabrication d'un produit comporte aussi une transition puits dont le franchissement représente la fin de fabrication d'une unité de ce produit.

Le RdP correspondant aux gammes de fabrication représentées dans la Figure 4.2 est indiqué dans la Figure 4.3.

Le modèle de planification est constitué de composantes connexes disjointes, chacune de ces composante étant un réseau de Petri à sortie contrôlable sans place ressource. Chaque composante connexe est le modèle d'une gamme de fabrication et ne comporte qu'une seule transition puits.

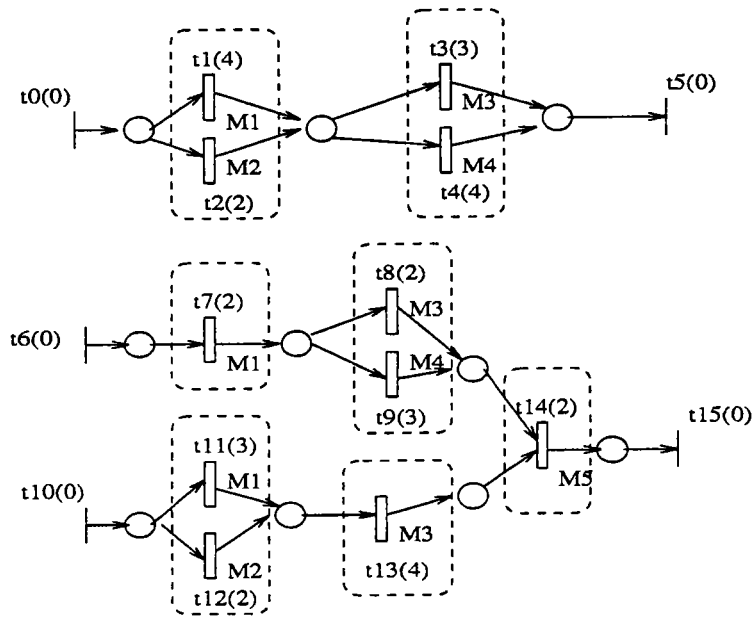


FIG. 4.3 – *Le RdP correspondant aux deux gammes de fabrication*

Dans le modèle utilisé pour la planification, aucun mécanisme n'empêche deux transitions correspondant à la même machine d'être franchies simultanément: on ne s'intéresse qu'à la charge des machines et non à l'ordre d'exécution des opérations.

Le modèle d'ordonnancement s'obtient en complétant le modèle pour la planification de la manière suivante:

- pour chaque machine, on introduit une place contenant exactement un jeton. Ces places sont les places ressources;
- on introduit les arcs qui font de chaque place ressource la place d'entrée et de sortie de toutes les transitions correspondant à cette machine.

Ce faisant, on empêche plusieurs transitions correspondant à la même machine d'être franchies simultanément.

Le modèle d'ordonnancement correspondant à l'exemple ci-dessus est représenté dans la Figure 4.4.

Comme on ne considère pas les opérations de désassemblage, on peut démontrer que les modèles d'ordonnancement sont des réseaux de Petri à sorties contrôlables (la démonstration se trouve en page 81: c'est la preuve de propriété A.11).

4.2 Modélisation modulaire

On sait que l'utilisation des réseaux de Petri élémentaires conduit à des modèles de taille trop importante pour être acceptables. D'où l'idée d'utiliser une approche

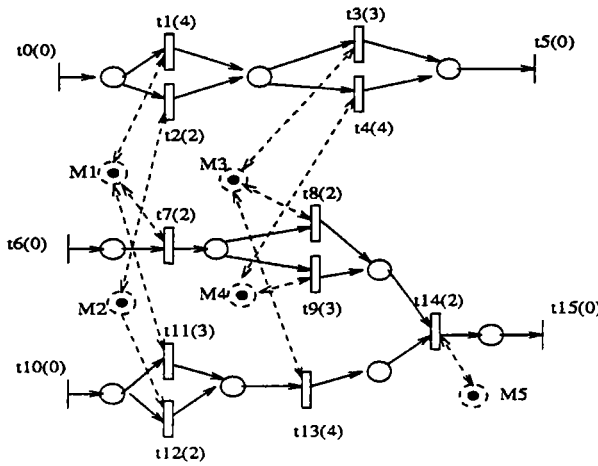


FIG. 4.4 – *Le modèle d'ordonnancement*

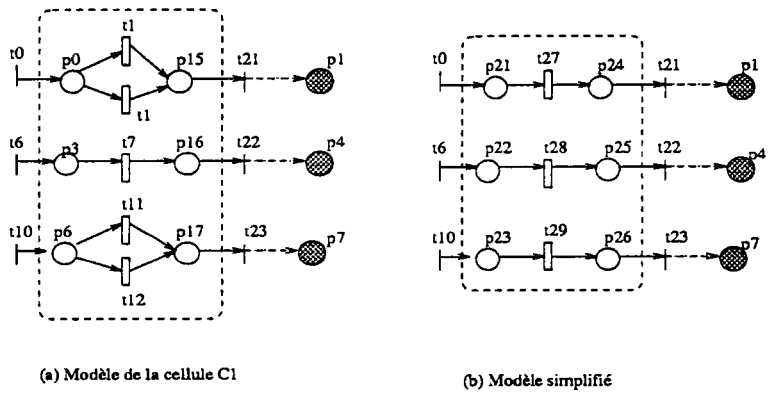


FIG. 4.5 – *Modèle RdP de la cellule C_1 et sa simplification*

modulaire (voir aussi section 1.3) que l'on peut résumer comme suit:

- décomposer le système de fabrication en modules. Le découpage d'un système de fabrication en modules dépend du système considéré. Dans le cas de notre exemple, si l'on groupe les machines M_1 et M_2 dans la cellule C_1, les machines M_3 et M_4 dans la cellule C_2, et si la machine M_5 constitue à elle seule la cellule C_3, on obtient trois modules.

- modéliser ces modules. La modélisation des cellules est faite à l'aide des réseaux de Petri et des places d'interface. La modélisation des cellules de notre exemple est représentée dans les Figures 4.5(a), 4.6(a), et 4.7(a).

- vérifier que les propriétés qualitatives de chacun des modules sont les propriétés souhaitées. Dans cet exemple, il n'est pas difficile de verifier les propriétés qualitatives telles que vivacité, réversibilité, consistance, bornitude etc. de ces trois modules, car ils sont tous des RdP à sorties contrôlables (voir l'annexe A pour plus de détails).

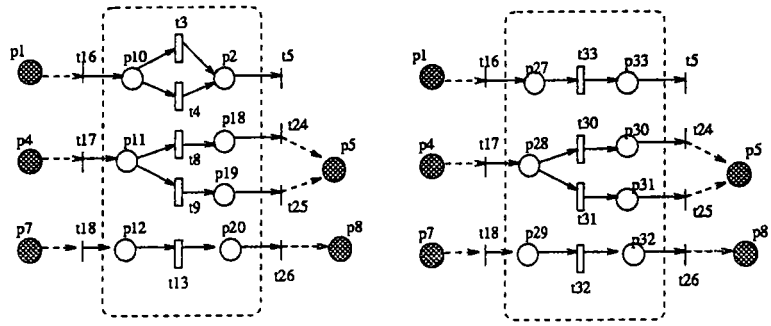


FIG. 4.6 – *Modèle RdP de la cellule C_2 et sa simplification*

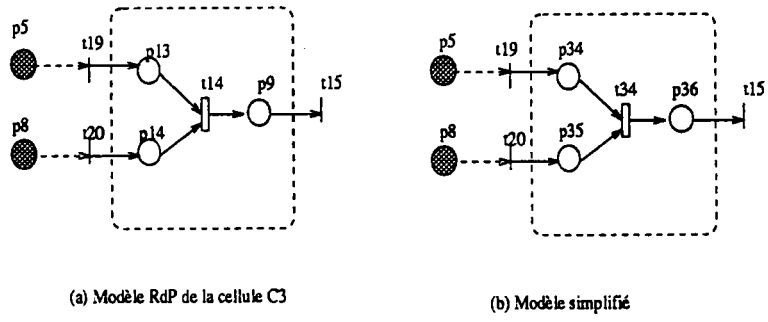


FIG. 4.7 – *Modèle RdP de la cellule C_3 et sa simplification*

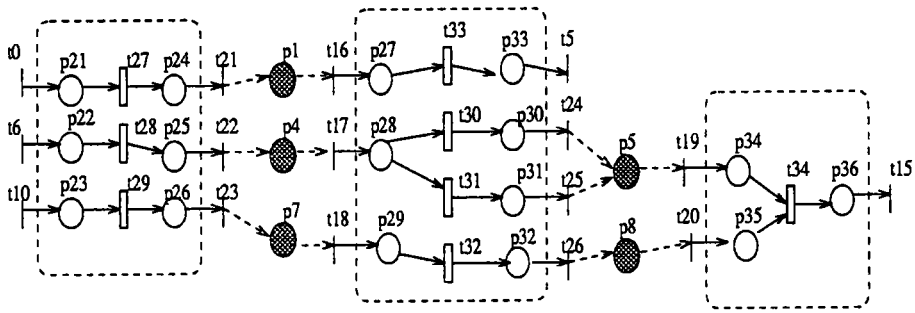


FIG. 4.8 – *Intégration des modèles simplifiés*

- simplifier chacun de ces modules d'une manière qui préserve les propriétés qualitatives. Dans notre exemple, on peut facilement simplifier les modules comme indiqué dans les Figures 4.5(b), 4.6(b), et 4.7(b). Pour les modules plus complexes, nous avons développé un algorithme général (pas seulement applicable aux RdP acycliques, mais aussi capable de traiter des RdP cycliques) et efficace pour leurs simplifications (voir l'annexe B pour plus de détails).
- intégrer les modules simplifiés d'une manière qui préserve également les propriétés qualitatives. L'intégration se fait par la fusion des places d'interfaces. Le RdP intégré de l'exemple est schematisé dans la Figure 4.8.

4.3 Gestion hiérarchisée

Le modèle intégré, composé de modules simplifiés, servira à la planification à moyen terme et fournira, en sortie, les quantités à fabriquer par chaque module durant chaque période élémentaire. On pourra effectuer la planification à court terme et l'ordonnancement au niveau de chaque module, éventuellement sur des processeurs différents fonctionnant en parallèle.

Nous savons que les propriétés qualitatives désirées sont vérifiées et nous avons montré, dans la section précédente, en quoi consiste la simplification des modules et l'intégration des modules simplifiés. Nous y reviendrons en détails dans l'annexe B et dans l'annexe C.

Dans l'annexe C, nous montrerons aussi comment calculer la planification à moyen terme et la planification à court terme lorsque les demandes sont connues à la fin de chaque période élémentaire.

La planification à moyen terme est basée sur le modèle simplifié. L'objectif est de trouver, pour chaque module, le nombre de frachissements de chaque transition de sortie durant chaque période élémentaire. Pour plus de détails sur la formulation générale de ce problème, voir section C.7 de l'annexe C.

La planification à court terme est basée sur le modèle de chaque module. L'objectif est de maximiser le débit des modules tout en tenant compte de la capacité des modules et en respectant les décisions prises au niveau haut de la hiérarchie (i.e., la planification à moyen terme). Pour plus de détails sur la formulation de ce problème, voir section C.8 de l'annexe C.

Pour ordonnancer des modules, nous utilisons deux méthodes: le recuit simulé et la procédure par séparation et évaluation (Voir, e.g., [11] [32] [41]). Nous observons que la procédure par séparation et évaluation est particulièrement efficace lorsque la taille du problème est réduite. Lorsque la taille du problème est importante, la méthode du recuit simulé devient de plus en plus intéressante [29].

4.4 Présentation du logiciel HMPS

Logiciel HMPS a été développé au sein de l'équipe SAGEP de l'INRIA, dans le cadre du projet SIREP (Spécification et Intégration à l'aide des Réseaux de Petri des fonctions d'un système de production)[1]. Le projet SIREP s'intéresse à la modélisation et à l'évaluation fonctionnelle des systèmes de production discrets.

Logiciel HMPS est realisé en language C et language Tcl/Tk. Sa mise en œvre a été faite sur un ordinateur IBM RS6000. Le choix de cette machines s'impose par le fait que nous faisons appel à la bibliothèque d'optimisation IBM OSL, qui tourne sur cette machine. Logiciel prototype HMPS permet de gérer un système discret ouvert de manière hiérarchique. HMPS a été appliqué à des donnés de l'industrie (Péchiney) avec l'aide des ingénieurs du groupe.

Le menu principal du logiciel HMPS comporte les options suivantes:

- définition du modèle;
- définition des cellules;
- construction du réseaux de Petri simplifié;
- planification du réseaux de Petri simplifié;
- planification des cellules;
- lancement de l'ordonnancement;

Initialement, seule l'option " définition du modèle" est activée. Une fois qu'un modèle a été défini, l'option "définition des cellules" est activée. Aussi, une fois que les cellules ont été définies, l'option "construction du réseaux de Petri simplifié" est activée. De la même façon, une fois que le réseaux de Petri simplifié a été construit,

[1] Plus de détails se trouvent dans le rapport final du projet SIREP: **SIREP - Rapport final, compte rendu de fin d'opération d'une recherche financée par le Ministère de la Recherche et de l'Espace** et annexe D de ce mémoire

l'option "planification du réseaux de Petri simplifié" peut être activée. Une fois que la planification du réseaux de Petri simplifié a été faite, l'option "planification des cellules" peut être activée. L'option "lancement de l'ordonnancement" ne sera activée qu'après la planification des cellules.

Pour plus de détails sur le fonctionnement du logiciel, voir l'annexe D.

Chapitre 5

Evaluation de l'approche

Nous avons proposé une approche modulaire et hiérarchique. Cette approche est basée sur la théorie des réseaux de Petri. Les résultats développés ont été intégrés dans le logiciel HMPS (voir annexe D).

L'objectif de ce chapitre est d'analyser les performances de cette approche.

Nous étudions d'abord les performances de notre approche à l'aide de trois jeux d'exemples proposés dans Savi [115]. Nous comparons les résultats que nous avons obtenus avec ses résultats.

Nous analysons ensuite un exemple provenant de l'industrie et étudié dans le cadre du projet SIREP. Nous verrons que notre approche est efficace pour les problèmes que nous avons rencontrés dans la pratique.

5.1 Exemples illustratifs

Quinze structures de gammes de fabrications sont définis pour évaluer l'application de l'approche proposée dans [115]. Elles peuvent représenter le processus de fabrication d'un type de produit donné.

Pour obtenir un exemple de test, on:

- choisit le nombre de types de produits ainsi que le nombre de machines utilisées,
- affecte au hasard une de ces quinze structures de gammes à chaque type de produits,
- affecte au hasard une machine et une durée opératoire à chaque opération de la gamme.

La loi de distribution uniforme, $U[a, b]$, est appliquée au choix des durées opératoires, des demandes de produits, des coûts de stockage et des coûts de rupture.

Deux groupes d'exemples ont été définis (voir Table 5.1):

	Groupe I					Groupe II				
Exemple	1	2	3	4	5	6	7	8	9	10
# de produits	6	7	7	8	9	20	30	40	50	100
# de machines	6	5	7	5	5	10	10	20	20	20

TAB. 5.1 – *Deux groupes d'exemple*

- le premier groupe (exemple 1 à exemple 5) est composé d'exemples de petite taille. Les exemples de ce groupe utilisent les quinze gammes de fabrication.
- le deuxième groupe (exemple 6 à exemple 10) n'utilise qu'une seule gamme de fabrication qui consiste à un seul routage par produit.

Basés sur ces deux groupes d'exemples, trois groupes de tests ont été faites dans [115].

Pour tester les performances de notre approche, nous avons, pour chaque exemple d'un groupe de test,

- groupé les machines en cellules (même si ce groupement dépend du besoin de l'utilisateur, il y a quelques règles qui sont analysés dans l'annexe C),
- affecté au hasard (selon la loi de distribution uniforme) un coût de stockage à chaque place d'interface,
- affecté au hasard (également selon une loi de distribution uniforme) un marquage initial à chaque place d'interface.

Les résultats obtenus pour les trois groupes de tests sont présentés respectivement dans les Tables 5.2, 5.3, et 5.4, où:

- EX est le numero d'exemple,
- DP est la durée de la période,
- SE est la solution exacte (avec l'approche monolithique),
- SS est la solution heuristiquue proposée dans Savi [115],
- SAMH est la solution obtenue à l'aide de l'approche modulaire et hiérarchique,
- VC est la valeur du critère,
- TC est le temps de calcul, en unités du temps de CPU sur IBM RS6000,

EX	DP	VC			TC		
		SE	SS	SAMH	SE	SS	SAMH
1	390	58	64	85	122,21	0,93	0,79
2	570	8	14	19	10,87	1,67	0,12
3	480	9	9	11	9,92	1,89	0,11
4	1000	72	72	94	600,52	2,54	4,17
5	800	29	30	40	236,98	2,89	2,01
6	700	89	90	118	46,57	1,97	0,50
7	1300	56	60	80	10,82	2,78	0,10
8	850	70	80	85	202,15	3,52	1,15
9	1000	85	88	115	122,35	5,88	1,52
10	1750	77	77	110	336.87	13,79	2,87

TAB. 5.2 – *Premier test*

EX	DP	VC			TC		
		SE	SS	SAMH	SE	SS	SAMH
1	380	6	11	15	2,18	0,95	0,05
2	560	3	10	14	168,20	1,18	1.02
3	400	4	4	6	90,57	0,91	0,57
4	750	10	17	23	3,02	1,04	0,05
5	580	0	7	10	4,60	1,27	0,06
6	820	33	43	56	36,64	1,26	0,35
7	1140	13	14	19	181,72	1,68	1,29
8	1050	68	68	83	48,12	2.04	0,53
9	1060	30	35	46	12,08	2,85	0,15
10	2200	22	23	30	14,79	8,97	0,20

TAB. 5.3 – *Deuxième test*

EX	DP	VC			TC		
		SE	SS	SAMH	SE	SS	SAMH
1	430	2	9	12	27,69	1,12	0,50
2	850	3	3	5	36,52	0,79	0,54
3	500	5	10	13	112,97	1,48	1,01
4	700	9	12	14	14,07	1,20	0,22
5	880	4	6	8	22,84	0,90	0,40
6	900	28	29	38	5,27	1,05	0,10
7	1200	67	68	89	33,50	1,89	0,52
8	900	12	12	16	8,40	2,13	0,33
9	1000	35	36	47	18,10	3,13	0,64
10	1900	47	47	62	19,85	8,15	0,57

TAB. 5.4 – *Troisième test*

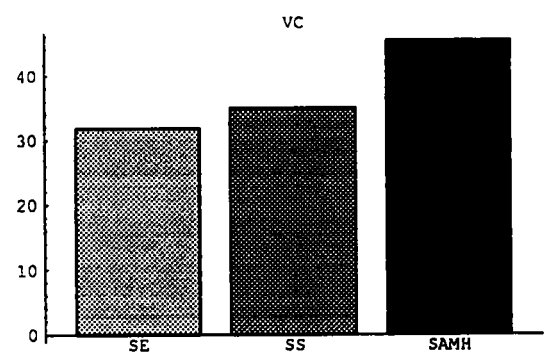


FIG. 5.1 – *Valeurs critères des tests SE, SS, et SAMH*

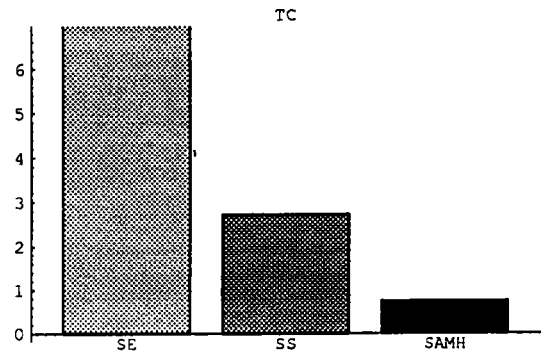


FIG. 5.2 – *Temps de calcul pour tests SE, SS, et SAMH*

Les valeurs moyennes des résultats de ces trois tests sont donnés dans les Figures 5.1 et 5.2.

Dans notre cas, la valeur du critère est obtenue comme la somme des coûts de stockage et des coûts de rupture au niveau haut et au niveau bas. La résolution des problèmes de planification au niveau haut et au niveau bas faite applel à la programmation linéaire dans logiciel OSL. Comme nous pouvons voir dans les Tables 5.2, 5.3, et 5.4, pour chacun de ces trente exemple, la valeur du critère (VC) obtenue par l'approche modulaire et hiérarchique (SAMH) est plus grande que celles par l'approche exacte (SE) et l'approche proposée dans [115] (SS). Cependant, temps de calcul avec l'approche SAMH est beaucoup plus courte que temps de calcul avec SE et SS. Ceci justifie d'une part l'utilisation de l'approche modulaire et hiérarchique. D'autre part, avec l'approche modulaire et hiérarchique, on peut traiter des problèmes de tailles importantes qui ne peuvent être pas résolu par l'approche SE et SS.

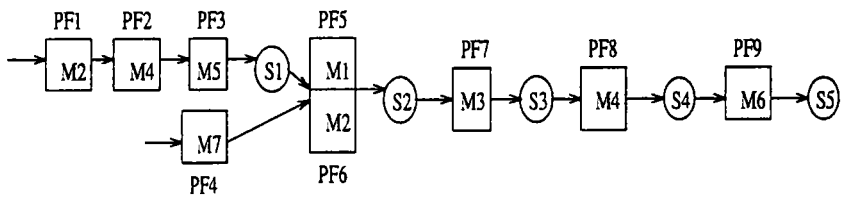
5.2 Application industrielle

Dans cette section, nous étudions un exemple provenant de l'industrie[1].

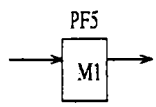
Nous disposons de 28 types de produits (P_1 à P_{28}) à fabriquer sur 7 machines (M_1 à M_7). P_1 à P_{20} suivent la gamme de fabrication indiquée dans la Figure 5.3(a), P_{21} à P_{28} suivent la gamme de fabrication indiquée dans la Figure 5.3(b). Dans cette figure, il y a 9 Points de Fabrication (PF): PF1 à PF9. Chaque PF est associé à une machine. Il est clair que certaines machines sont associées à plusieurs PFs.

[1] Cet exemple est fourni par Monsieur J.-M. BEAUVILLE de la Société Aluminium Péchiney pour le projet SIREP.

Par exemple, M_2 est associée aux PF1 et PF6, M_4 est associée aux PF2 et PF8. P_{21} à P_{28} suivent une gamme qui consiste à un seul PF (PF5) auquel est associé la machine M_1. Les points de fabrication PF5 et PF6 constituent partiellement le même point de fabrication. Il contient deux machines (M_1 et M_2) à fonction identique[2] pour les opérations à PF5 et PF6. Il faut noter que les deux machines peuvent être utilisées pour d'autres opérations. Par exemple, M_2 est aussi utilisée pour exécuter l'opération à PF1, M_1 est aussi utilisée pour exécuter l'opération de transformation des produits P_{21} à P_{28} comme indiqué dans la Figure 5.3(b).



(a) Gamme pour P1 ..... P20



(b) Gamme pour P21 ..... P28

FIG. 5.3 – *Gamme pour P_1 à P_{28}*

Dans la Figure 5.3, nous avons aussi 5 stocks tampons (S1 à S5) qui servent à stocker les produits intermédiaires (dans S1, S2, S3, et S4) et les produits finis (dans S5).

Les temps élémentaires (en minutes) sont présentés dans la Table 5.5 pour les produits P_1 à P_{20} et dans la Table 5.6 pour les produits P_{21} à P_{28}.

Les demandes des produits P_1 à P_{28} durant les 7 premiers mois sont données comme indiqué dans la Table 5.7.

Dans tous les stocks tampons (S1 à S5), la rupture des produits est interdite. Les ccoûts de rupture sont $\beta_1 = \cdots = \beta_5 = 200$. Dans le stock tampon $S2$, le stockage des produits est interdit. Le coût de stockage est $\alpha_2 = 200$. Les coûts de stockage dans les stocks tampons sont: $\alpha_1 = 0,6$, $\alpha_3 = 3$, $\alpha_4 = 5$ $\alpha_5 = 7$. Ces valeurs figurent dans la Table 5.8.

La période élémentaire est 1 poste de 8 heures. Il y a 15 postes de 8 heures travaillés par semaine.

La sous-période est le mois. Donc chaque sous-période consiste à 60 périodes élémentaires.

[2] Les temps de fabrication sur les machines sont différents

	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9
P_1	0,1	0,5	0,2	0,5	2	1	1	2	4
P_2	0,1	0,5	0,2	0,5	2,5	1	1,5	2,5	5
P_3	0,1	0,5	0,2	0,5	2	1,5	0,5	2,5	4
P_4	0,1	1	0,3	0,5	2	0,5	1	1,5	5
P_5	0,1	1	0,3	0,5	2	0,5	1	1,5	4
P_6	0,1	1	0,3	0,5	1,5	1	1	1,5	4
P_7	0,1	1	0,3	0,5	2,5	0,5	1	2	5
P_8	0,1	1	0,3	0,5	1,5	1,5	1	2	4,5
P_9	0,2	0,5	0,2	0,5	2	1	1	2	4
P_{10}	0,2	0,5	0,3	0,5	2,5	1	1,5	2,5	5
P_{11}	0,2	0,5	0,3	0,5	2	1,5	0,5	2,5	4
P_{12}	0,2	0,5	0,3	0,5	2	0,5	1	1,5	5
P_{13}	0,2	0,5	0,3	0,5	1,5	1	1	1,5	4
P_{14}	0,2	0,5	0,3	0,5	2,5	0,5	1	2	5
P_{15}	0,2	0,5	0,3	0,5	1,5	1,5	1	2	4,5
P_{16}	0,2	0,5	0,4	0,5	1,5	1,5	1	2	4,5
P_{17}	0,2	0,5	0,4	0,5	2,5	0,5	1	2	5
P_{18}	0,2	0,5	0,4	0,5	1,5	1	1	1,5	4
P_{19}	0,2	0,5	0,4	0,5	2	0,5	1	1,5	5
P_{20}	0,2	0,5	0,4	0,5	2	1,5	0,5	2,5	4

TAB. 5.5 – *Temps élémentaires pour P_1 à P_{20}*

P_{21}	P_{22}	P_{23}	P_{24}	P_{25}	P_{26}	P_{27}	P_{28}
1	1,5	1	1,5	1	1,25	1,25	1

TAB. 5.6 – *Temps élémentaires pour P_{21} à P_{28}*

	Mois						
	1	2	3	4	5	6	7
P_1	0	0	50	0	50	0	0
P_2	150	50	0	400	150	150	0
P_3	50	300	300	150	0	300	50
P_4	300	400	400	150	0	0	0
P_5	0	0	50	150	200	0	0
P_6	0	50	0	400	400	400	0
P_7	50	300	400	400	400	400	400
P_8	150	150	300	0	0	50	50
P_9	300	400	400	0	0	50	200
P_{10}	400	300	200	150	150	150	150
P_{11}	0	0	300	50	300	150	150
P_{12}	0	50	0	300	300	300	300
P_{13}	300	0	50	50	300	300	400
P_{14}	200	0	0	150	50	200	200
P_{15}	50	400	50	400	50	0	0
P_{16}	150	400	50	200	150	200	200
P_{17}	300	400	200	200	0	200	150
P_{18}	400	400	0	0	0	50	50
P_{19}	200	50	0	0	0	200	400
P_{20}	300	50	200	0	0	0	400
P_{21}	50	300	50	200	300	0	300
P_{22}	300	0	200	50	400	150	300
P_{23}	200	50	150	300	300	150	0
P_{24}	50	0	50	300	150	200	150
P_{25}	0	0	0	0	50	0	50
P_{26}	0	150	150	50	50	50	150
P_{27}	0	400	400	150	50	150	0
P_{28}	400	200	0	0	50	0	50

TAB. 5.7 – *Demandes des produits (mensuelles)*

	S1	S2	S3	S4	S5
coût de stockage	0,6	200	3	5	7
coût de rupture	200	200	200	200	200

TAB. 5.8 – *Coûts de stockage et coûts de rupture*

L'objectif est de minimiser les coûts de stockage et les coûts de rupture à l'année.

L'approche modulaire et hiérarchique est utilisée pour résoudre ce problème. Une planification avec la valeur du critère égale à 145,61 est obtenue avant 28,47 minutes.

Il faut noter que les approches SE et SS sont incapables de traiter ce problème.

5.3 Conclusions

Selons les résultats que nous avons obtenus pour résoudre les problèmes précédents, nous pouvons constater que l'approche que nous avons proposée dans ce mémoire, i.e., l'approche modulaire et hiérarchique, est efficace pour les problèmes que nous avons rencontrés dans la pratique.

Cependant, l'amélioration de ce logiciel est toujours possible, afin qu'une grande variété de gammes de fabrication de produits puisse être traitée. Quelques améliorations possibles sont mentionnés dans le chapitre 6 et l'annexe D.

Chapitre 6

Conclusions générales et perspectives

6.1 Conclusions générales

Dans ce mémoire, nous nous sommes fixés comme objectif de montrer que les approches modulaires permettent de modéliser des systèmes de production complexes, et que les approches hiérarchiques permettent de gérer les tels systèmes. Les nombreuses propriétés importantes que nous avons présentées dans les chapitres précédents et dont certaines sont antérieures à ce travail (e.g., [35] [37] [92] [105] [120]), montrent que les réseaux de Petri sont un outil complet qui permet de réaliser la modélisation, l'analyse et l'évaluation des systèmes à événements discrets et, notamment, des systèmes de production discrets.

Dans un monde de compétition intense, la capacité de s'adapter aux changements de plus en plus rapide des demandes est devenue indispensable. Les efforts que nous faisons sur la modélisation, l'analyse et l'évaluation des systèmes de production non cycliques consistent précisément à apporter à l'industrie un outil pour les aider dans la conception préliminaire d'un nouveau système ou dans la ré-organisation des systèmes existants. Le développement d'un tel outil nécessite de trouver des solutions efficaces aux nombreux problèmes tels que:

- la définition de nouveaux réseaux de Petri pour modéliser les systèmes rencontrés dans l'industrie;
- la définition de conditions d'intégration aussi peu restrictives que possible et qui garantissent la préservation des propriétés des modules;
- l'évaluation quantitative des systèmes intégrés en fonction des scénarios fournis par l'utilisateur. Cela nécessite la résolution de problèmes de planification et d'ordonnancement de la production. Nous pouvons profiter des propriétés des modèles de réseaux de Petri pour proposer des solutions efficaces à nos problèmes de planification et d'ordonnancement.

Motivé par les résultats des projets industriels, nous développons actuellement une méthodologie de modélisation des systèmes de production complexes. Cette méthodologie consiste à:

- décomposer le système de production en sous-systèmes de production appelé modules;
- modéliser les modules à l'aide des réseaux de Petri contrôlés;
- vérifier les propriétés qualitatives et évaluer les performances des modèles des modules. Les propriétés qualitatives considérées sont la *vivacité*, la *bornitude*, la *réversibilité*, etc.
- intégrer les modèles des modules.

Nous nous sommes concentrés sur l'intégration de modèles représentant des modules (ou sous-systèmes) de fabrication. Une méthodologie modulaire pour l'intégration étant adoptée, nous avons défini une classe de réseaux de Petri applicable à la modélisation d'un large ensemble de systèmes de fabrication. Les réseaux de Petri utilisés sont vivants, consistants et réversibles. Nous avons montré que l'intégration de ces modules conserve ces propriétés qualitatives sous des conditions peu restrictives. De plus, le modèle global appartient également à la classe de réseaux de Petri considérée. Les résultats obtenus permettent d'éclairer les problèmes de conception, de gestion et de contrôle des systèmes de production de grande taille.

Afin d'utiliser les modèles obtenus à l'aide de l'approche modulaire, nous avons proposé le concept de *T-invariant réduit*. Nous utilisons les T-invariants réduits pour caractériser les relations entre les transitions d'entrée d'un module et ses transitions de sortie. Un algorithme efficace a été développé pour déterminer les T-invariants réduits.

Les T-invariants réduits permettent d'obtenir les modèles agrégés des modules. Ces modèles agrégés possèdent les propriétés qualitatives requises et sont indispensables pour l'évaluation du comportement du système intégré à l'aide d'une approche hiérarchique.

Un système de gestion de production hiérarchisée se compose d'un certain nombre de niveaux de prise de décisions. Les niveaux sont reliés entre eux par le flux des décisions qui descendent dans la hiérarchie et par les retours d'information sur l'état du système qui remontent dans la hiérarchie (voir figure 1.3).

Nous avons étudié en particulier la gestion hiérarchisée à deux niveaux. Nous avons vu qu'il est possible de gérer, à l'aide des RdP élémentaires, des systèmes non-cycliques de taille importante. Il suffit de les décomposer en modules dont les modèles sont contrôlables, et de simplifier ces modèles. La gestion se déroule de la manière suivante:

- Planification du modèle obtenu par intégration des modèles simplifiés sur

un horizon H, c'est-à-dire, sur H période élémentaires. Cette planification conduit à deux résultats importants, à savoir:

1. l'évolution des stocks inter-modules sur l'horizon H; ces stocks inter-modules sont représentés par les place d'interface,
2. le nombre de franchissements de chaque transition de sortie des modules durant chaque période élémentaire.

   Ces calculs constituent **le niveau haut de la hiérarchie.**

- Planification et ordonnancement des modules. Ces calculs, qui constituent **le niveau bas de la hiérarchie**, partent du nombre de franchissements de chaque transition de sortie des modules durant chaque période élémentaire.

Nous voyons que la hiérarchie se constitue de manière naturelle lorsqu'on utilise les RdP pour modéliser le système. Les RdP sont donc un outil d'intégration au niveau de la gestion des systèmes de production.

Enfin, nous avons établi une démarche descendante qui consiste à résoudre le modèle global simplifié, puis à se servir de la solution obtenue comme contrainte à satisfaire par les modules détaillés du niveau bas. Un logiciel prototype HMPS a été développé pour faciliter cette démarche.

6.2 Perspectives

Les recherches futures exploiteront les potentiels des outils modernes pour la modélisation, l'analyse et l'évaluation des systèmes à événements discrets.

L'approche hiérarchique de la gestion des systèmes de production est toujours confrontée par une question fondamentale: combien de niveaux hiérarchiques sont nécessaires pour représenter fidèlement la réalité? Bien que nous ayons adopté une structure à deux niveaux, une méthode qui facilite la conception du système hierarchisé dans différentes situations pratiques reste à trouver [44].

Dans ce travail, nous nous sommes concentrés sur les gammes de fabrication qui contiennent deux types d'opérations: *transformation* et *assemblage*. Bien que ces deux types d'opérations intéressent une large famille de gammes, on peut trouver, dans la pratique, des types de gammes incluant d'autres types d'opérations comme par exemple l'opération de désassemblage. Il faut donc envisager un système capable de prendre en compte un ensemble plus large de types de gammes.

En ce qui concerne le logiciel HMPS, on a observé qu'il y a plusieurs points à améliorer:

- de meilleures méthodes d'ordonnancement basées sur les RdP restent à trouver. Rapellons que l'on ne cherche pas une ordonnancement optimal, mais une ordonnancement admissible.

- des logiciels de programmation linéaire nécessaires, pour la planification au niveau haut et au niveau bas est envisagée. Le présent logiciel fait appel à un package commercial OSL pour calculer un plan optimal, ce qui est un grand obstacle dans la commercialisation du logiciel, car OSL n'est pas disponible dans la plupart d'entreprises moyennes et petites.

- une interface homme-machine plus conviviable reste à réaliser. La conception d'une telle interface basée sur d'autre langage que Tcl/Tk, si possible, est à envisager.

- Les situations comme l'agrégation des types de produits en familles et des temps opératoires stochastiques restent à réaliser.

Annexe A

Controllable-Output Petri nets

In this part [1], we define a class of Petri nets, called CO nets (short for Controllable-Output nets), explore their properties, study the integration issue concerned with the combination of CO net modules to form large systems, and develop algorithms for identifying CO nets. CO nets form a basic building block for the modular and hierarchical approach to system analysis proposed in our work.

A.1 Basic notions and properties of CO nets

In this section, we first introduce the notion of CO net (or CO net module), then we explore some properties of CO nets.

A.1.1 Definition of CO nets

Definition A.1 (CO net module – Controllable-Output net module) *A Petri net $G = (P \cup R, T, F, M_0)$, where P and R are two disjoint sets of process places and resource places, is called a controllable-output net module (CO net module, for short) if the following conditions hold:*

H1. $(t,r) \in F \Longleftrightarrow (r,t) \in F, \forall t \in T, \forall r \in R$

H2. $M_0(r) \geq 1, \forall r \in R$

H3. The subnet $G' = (P, T, F', M'_0)$, where F' is the restriction of F on $(P \times T) \cup (T \times P)$, is an acyclic graph without isolated nodes. M'_0 is the restriction of M_0 on P.

[1] The material in this part comes mainly from the paper:

Jean-Marie Proth, Liming Wang, Xiaolan Xie, "A class of Petri net for manufacturing system integration", *INRIA Research Report No. 2055*, 1993 (A revised version of this paper is submitted for publication in *IEEE Trans. on Robotics and Automation*)

H4. The extremity nodes of G are transitions. Let T_{in} and T_{out} be respectively the set of input transitions and the set of output transitions, i.e.,

$$T_{in} = \{t \mid {}^\bullet t = \emptyset,\ t \in T\}, \qquad T_{out} = \{t \mid t^\bullet = \emptyset,\ t \in T\}$$

H5. The net G' is covered by a set of T-invariants each of which is related to a unique output transition, i.e., $\forall t \in T_{out}$, $\exists y_t \in \mathbf{N}^{|T|}$, such that

$$C\ y_t = 0,\ \| y_t \| \cap T_{out} = \{t\},\ and \bigcup_{t \in T_{out}} \| y_t \| = T.$$

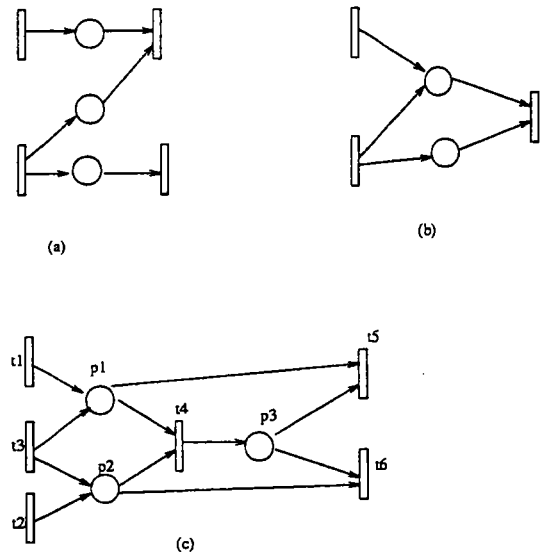


FIG. A.1 – *CO nets and non-CO nets*

Some remarks are due to clarify the definition of CO net.

Remark A.1 H1 and H2 together imply that resource places $r \in R$ are implicit places that are, informally speaking, those places whose markings are always sufficient for the firing of their output transitions. According to [23][92], we know that removing the implicit places does not change the behaviour of the net. As a result, we need only to consider the net G'.

Remark A.2 Hypotheses H3 and H4 imply that any place is an internal node, i.e., it is neither a sink place nor a source place. Furthermore the set of input transitions T_{in} and the set of output transitions T_{out} are disjoint, i.e., $T_{in} \cap T_{out} = \emptyset$.

Remark A.3 Hypothesis H5 implies that the net G' is consistent. But a consistent net does not lead to hypothesis H5 as shown in Figure A.1 . It is the requirement of H5 that motivates us to term this class of Petri net modules as

controllable-output net modules, since H5 requires that each output transition of the module is related to at least one T-invariant whose support does not contain any other output transitions of the module. That is to say, the firing of each output transition can be totally independent of the firings of other output transitions. Thus the output of the systems represented by CO nets can be *controlled* with great flexibility.

Figure A.1 gives three Petri nets in which (c) is a CO net, while (a) and (b) are not since the net (b) is not consistent, and (a) has only one T-invariant whose support contains both output transitions of the net, thus H5 is violated. Note that for simplicity resource places and related connecting arcs are not shown in these Petri net graphs. The reason that (c) is a CO net is that there exist two T-invariants $y_{t_5} = [1\ 0\ 1\ 1\ 1\ 0]$ and $y_{t_6} = [0\ 1\ 1\ 1\ 0\ 1]$, each corresponding exactly to one output transition and the union of their supports covers all the transitions of the net.

A.1.2 Properties of CO nets

In this subsection we investigate some structural and behavioural properties of CO nets. Specifically we are interested in consistency, liveness, boundedness, and reversibility, which are the properties required by Petri net models of real manufacturing systems.

Property A.1 (Consistency) *A CO net is consistent.*

Proof: It is obvious from assumption H5.

Property A.2 (Liveness) *A CO net is live. no matter what the initial marking is.*

Proof: Since G is an acyclic graph, we can relabel the transitions as $t_1,\ t_2,\ \cdots,\ t_n$ (assume that n is the number of transitions in G) such that $\forall t_i \in T$, no path exists from t_{i+k} to t_i $(k = 1,\ 2,\ \cdots,\ n-1)$, and the first transition(s), say $t_1,\ t_2,\ \cdots,\ t_r$ $(1 \leq r < n)$ are input transition(s).

In the following, we show by induction that

$$\forall t_i \in T, \exists \sigma_i \in \{t_1, t_2, \cdots, t_{i-1}\}^*, \text{such that } M[\sigma_i \circ t_i >, \forall M \in \mathbf{N}^{|P|} \qquad \text{(A.1)}$$

where A^* is the set of all finite sequences (including empty sequence which is denoted by λ) formed by the elements in the finite set A, and $\sigma \circ t$ represents the concatenation of a sequence σ with an element t.

- First, induction assumption holds for any input transition $t_i \in \{t_1,\ t_2,\ \cdots, t_r\}$, since $M[\sigma_i \circ t_i >, \forall M \in \mathbf{N}^{|P|}$, where $\sigma_i = \lambda$.

- We then assume that claim A.1 holds for any t_i such that $i \leq k < |T|$. Consider transition t_{k+1}, all the transitions which immediately precede t_{k+1}, i.e., the transitions belonging to ${}^\bullet({}^\bullet t_{k+1})$, belong to $\{t_1,\ t_2,\ \cdots,\ t_k\}$. This is the consequence of the way transitions have been re-labelled.

- Thus if all the transitions of ${}^\bullet({}^\bullet t_{k+1})$ have been fired, it is possible to fire t_{k+1}. Furthermore, according to the induction assumption, $\forall t_j \in {}^\bullet ({}^\bullet t_{k+1}), \exists \sigma_j \in \{t_1,\ t_2,\ \cdots, t_{j-1}\}^*$ such that $M[\sigma_j \circ t_j >, \forall M \in \mathbf{N}^{|P|}$. As a consequence, $M[\sigma_{k+1} \circ t_{k+1} >, \forall M \in \mathbf{N}^{|P|}$ holds, where σ_{k+1} consists in firing once the sequence $\sigma_j \circ t_j,\ \forall t_j \in {}^\bullet ({}^\bullet t_{k+1})$.

This completes the proof.

For this property, we notice that there is an alternative proof. That is, since the net G does not contain any syphon, it is live for any initial marking M_0 according to [23][92]. As can be noticed, this property holds for any acyclic Petri net without source place(s).

Property A.3 (Unboundedness) *A CO net G is not bounded.*

Proof: This property is true due to the existence of input transitions. Since input transitions are always enabled, and thus can always be fired. As a consequence, the number of tokens in the output places of the input transitions can increase to infinity.

The facts that CO net is consistent and that the subnet structure of a CO net is acyclic lead to the following:

Property A.4 (Reversibility) *A CO net G is reversible, whatever the initial marking is.*

Proof: For any $M \in R(M_0)$, the proof proceeds as follows:

(a) $\exists \sigma \in T^*$ such that $M_0[\sigma > M$.

(b) Since G is consistent, then there exists positive $|T| \times 1$ vector x, such that $Cx = 0$.

(c) From (a) and (b), we have that there exists a positive integer k, such that $kx - \bar{\sigma} \geq 0$, and $kx - \bar{\sigma} \neq 0$ where $\bar{\sigma}$ is the firing count vector of σ.

which yields

(d) $M_0 = M + C(kx - \bar{\sigma})$

(e) From Theorem 16 in [92], (d) and (c) imply that M_0 is reachable from M. This completes the proof.

A.2 System integration

In the last section, we defined CO nets and studied some properties of CO nets, and obtained the result that a CO net is consistent, live, and reversible, though not bounded. Usually several manufacturing modules are linked together to accomplish a certain manufacturing function. Knowing that each module has the qualitative properties as introduced above, the problem is to select an integration process which preserves these properties, i.e., which guarantees that the integrated system still has these qualitative properties. We consider the integration process in which all the CO net modules are linked together through a set Q of places, called interface places.

Definition A.2 (Integrated system) *Consider a set of modules $G_1, G_2, \cdots, G_n$, where $G_i = (P_i \cup R_i, T_i, F_i, M_{0,i})$ with input transitions T^i_{in} and output transitions T^i_{out}. An integrated system G is a Petri net resulting from the integration of modules $G_1, G_2, \cdots, G_n$ via a set Q of interface places and inter-module arcs Γ,*

$$G = (\bigcup_{i=1}^{n} P_i \cup R_i \cup Q, \bigcup_{i=1}^{n} T_i, \bigcup_{i=1}^{n} F_i \cup \Gamma)$$

where

$$\Gamma = (\bigcup_{i=1}^{n} T^i_{out} \times Q) \cup (Q \times \bigcup_{i=1}^{n} T^i_{in})$$

Furthermore the module connection through interface places satisfy the following conditions:

H6. Each interface place is neither a source place nor a sink place.

H7. Each output transition of any module is connected to at most one interface place, i.e.,

$$|t^\bullet| \leq 1 \qquad \forall t \in \bigcup_{i=1}^{n} T^i_{out}$$

Let us now give an example which is a typical Petri net system satisfying the above two conditions H6 and H7. Its Petri net graph is shown in Figure A.2

Definition A.3 (System contraction) *System contraction is the process of transforming an integrated system into a directed graph (called contracted graph) by contracting each module of the system as a node, viewing each place $q \in Q$ and each transition $t \in \{s \mid s \in T^i_{in} \cup T^i_{out}$, for all modules G_i in the system$\}$ as a node, and preserving the inter-module arcs.*

The contracted graph of the system shown in Figure A.2 is drawn in Figure A.3.

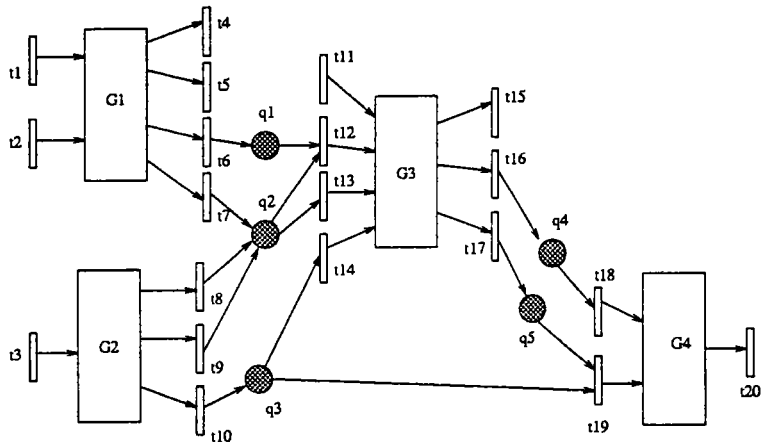


FIG. A.2 – *A typical structure of system integration*

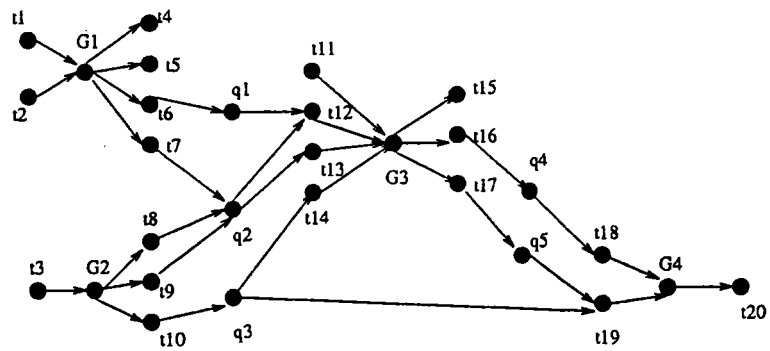


FIG. A.3 – *A contracted graph*

Definition A.4 (Acyclic system) *An acyclic system formed by modules* $G_1, G_2, \cdots, G_n$ *via the set* Q *of interface places and inter-module arcs* Γ *is a system*

$$G = (\bigcup_{i=1}^{n} P_i \cup R_i \cup Q, \bigcup_{i=1}^{n} T_i, \bigcup_{i=1}^{n} F_i \cup \Gamma)$$

whose contracted graph is acyclic.

Property A.5 (consistency of acyclic systems) *An acyclic system* G *satisfying H6 and H7 is consistent.*

Proof: From the definition of acyclic system, we have that it is always possible to re-label the modules as $G_1, G_2, \cdots, G_n$, such that there are no directed paths from G_j to G_i if $j > i$, $i, j = 1, 2, \cdots, n$.

For each module G_i, let $\{y_t \mid \forall t \in T^i_{out}\}$ be a set of T-invariants satisfying assumption H5, i.e.,

$$\bigcup_{t \in T^i_{out}} \|y_t\| = T_i \tag{A.2}$$

and

$$y_t[t] > 0 \text{ and } y_t[s] = 0 \qquad \forall s \in T^i_{out} - \{t\} \tag{A.3}$$

The central idea of the proof lies in constructing a T-invariant $Y > 0$ of the integrated Petri net system model. Clearly Y can be written in the following form $[Y_1^T\ Y_2^T\ \cdots\ Y_n^T]^T$ where Y_i^T is a vector whose components concern the transitions of G_i.

Two conditions are to be considered. First, $Y_i > 0$ is also a T-invariant of G_i. Second, the T-invariant Y should satisfy the balance equations for all interface places Q, i.e.,

$$\sum_{s \in {}^\bullet q} Y[s] = \sum_{s \in q^\bullet} Y[s] \qquad \forall q \in Q \tag{A.4}$$

In the following we construct Y in a backward fashion. Computing firstly Y_n, we then construct $Y_{n-1}, Y_{n-2}, \cdots, Y_1$ consecutively.

When computing Y_i of module G_i, we first compute the components corresponding to its output transitions. For this purpose, consider a positive number x_t for all $t \in T^i_{out}$, $(i = 1, 2, \cdots, n)$ defined as follows:

$$x_t = \begin{cases} 1 & \text{if } t^\bullet = \emptyset \\ \frac{\sum_{s \in q^\bullet} Y[s]}{|{}^\bullet q|} & \text{if } t^\bullet = q \end{cases}$$

where q is the unique output place of t.

From the definitions for G_i, each transition $s \in q^\bullet$ is an input transition of some of the modules $G_{i+1}, \cdots, G_n$, which implies that $Y[s]$ is known. Thus x_t is well defined.

The vector Y_i is thus defined as follows:

$$Y_i = \sum_{t \in T^i_{out}} \frac{x_t}{y_t[t]} \times y_t$$

It is readily verified that $Y_i[t] = x_t$ for all $t \in T^i_{out}$ due to relation A.3.

Since all x_t's are positive real numbers, relation A.2 implies that Y_i is also a T-invariant of G_i and $Y_i > 0$.

We still need to prove that the vector Y satisfies the flow balance equation for every interface place $q \in Q$, i.e., we need to prove that the vector Y satisfies equation A.4. This should be clear from the following induction:

$$\sum_{s \in {}^\bullet q} Y[s] = \sum_{s \in {}^\bullet q} \{\frac{1}{|{}^\bullet q|} \sum_{t \in q^\bullet} Y[t]\} = \sum_{t \in q^\bullet} Y[t]$$

where the first equality is true since q is the unique output place of all $t \in {}^\bullet q$. This completes the proof.

Furthermore, for acyclic systems, we also have the following

Property A.6 (other properties of acyclic systems) *An acyclic system G satisfying H6 and H7 is*

- *live*
- *reversible*
- *unbounded*

Proof: Since G is an acyclic net without source and sink places and since it is consistent, this property follows from Remarks A.1 and A.2.

From the last two properties about acyclic systems, we notice that the integration preserves both structural and behavioural properties of modules. This is not a surprise as we can prove that the integrated model is itself a controllable-output net module, i.e., a CO net.

Property A.7 *The integrated system G satisfies assumptions H1-H5.*

Proof: First, we explicitly write G in the following form:

$$G = (P \cup R,\ T,\ F,\ M_0)$$

with

$$P = \bigcup_{i=1}^{n} P_i \cup Q, \quad R = \bigcup_{i=1}^{n} R_i, \quad T = \bigcup_{i=1}^{n} T_i, \quad F = \bigcup_{i=1}^{n} F_i \cup \Gamma$$

Clearly, G satisfies assumptions H1-H4. We now prove that assumption H5 also holds.

For this purpose, let T_{in} and T_{out} be respectively the set of input transitions and the set of output transitions of G. Obviously, we have

$$T_{in} \subseteq \bigcup_{i=1}^{n} T_{in}^{i}, \quad T_{out} \subseteq \bigcup_{i=1}^{n} T_{out}^{i}$$

For each output transition $\tau \in T_{out}$, let us construct a T-invariant Y_τ in a similar way as the one used for constructing Y in the proof of property A.5. We construct

$$Y_{\tau,i} = \sum_{t \in T_{out}^{i}} \frac{x_{\tau,t}}{y_t[t]} \times y_t$$

with

$$x_{\tau,t} = \begin{cases} 1 & \text{if } t = \tau \\ 0 & \text{if } t \in T_{out} - \{\tau\} \\ \frac{1}{|^\bullet q|} \times \sum_{s \in q^\bullet} Y_\tau[s] & \text{if } t^\bullet = q \end{cases}$$

where $Y_{\tau,i}, x_{\tau,t}, y_t$ are respectively similar notations as Y_i, x_t, y_t defined in the proof of property A.5, and q is the unique output place of t.

As in the proof of property A.5, it can be shown that Y_τ is a T-invariant. Now let us prove by induction that the following relation holds:

$$Y = \sum_{\tau \in T_{out}} Y_\tau \tag{A.5}$$

or equivalently:

$$Y_i = \sum_{\tau \in T_{out}} Y_{\tau,i} \qquad \text{for } i = 1, ..., n \tag{A.6}$$

where Y is the T-invariant of G used in the proof of property A.5.

It is easy to show that equation A.6 holds for $i = n$. Assume that it holds for all $i = k+1,\ k+2,\ ...,\ n$. Consider then the module G_k. From the definition of $x_{\tau,t}$ and x_t, we have

$$\sum_{\tau \in T_{out}} x_{\tau,t} = x_t = 1, \qquad \forall t \in T_{out} \cap T_{out}^{k}$$

Furthermore, by induction assumption, for all $t \in T_{out}^{k}$, such that $t^\bullet = q$,

$$\sum_{\tau \in T_{out}} x_{\tau,t} = \sum_{\tau \in T_{out}} \frac{\sum_{s \in q^\bullet} Y_\tau[s]}{|^\bullet q|} = \frac{\sum_{s \in q^\bullet} Y[s]}{|^\bullet q|} = x_t$$

since all transitions $s \in q^\bullet$ belong to modules G_{k+1}, G_{k+2}, ..., G_n as a result of acyclicity. The above two relations imply that

$$\sum_{\tau \in T_{out}} x_{\tau,t} = x_t, \qquad \forall t \in T_{out}^{k}$$

which yields

$$\sum_{\tau \in T_{out}} Y_{\tau,k} = \sum_{\tau \in T_{out}} \sum_{t \in T_{out}^k} \frac{x_{\tau,t}}{y_t[t]} \times y_t = \sum_{t \in T_{out}^k} \frac{x_t}{y_t[t]} \times y_t = Y_k$$

Equation A.5 is then proved. Since Y is a T-invariant and $Y > 0$, we conclude that Y_τ ($\forall \tau \in T_{out}$) are T-invariants, that $\|Y_\tau\| \cap T_{out} = \{\tau\}$, and that $\bigcup_{\tau \in T_{out}} \|Y_\tau\| = T$. This completes the proof.

A.3 Identification of CO net

From the discussions in the previous sections we know that a CO net G has the properties of liveness, consistency, and reversibility. Thus it would be nice if we could find out whether a given Petri net is a CO net or not. This leads us to the issue to CO net identification.

It is not difficult to check whether a given Petri net satisfies conditions H1-H4 in the definition of CO net. Conditions H1 and H2 can be checked place by place. We then remove the resource places. If the resulting Petri net contains neither sink nor source place and if it does not contain any isolated node, condition H4 is satisfied. Condition H3, i.e., acyclicity, can thus be checked by using any graph theory based approach [48].

In order to settle the problem of identifying H5, let us re-examine this condition. H5 means that for each output transition t of a given Petri net G satisfying H1-H4, there exists a T-invariant y_t of the acyclic Petri net G' obtained by removing the resource places, such that the the support of y_t contains, besides other non-output transitions, one and only one output transition, that is t. Moreover the support of such T-invariants for all the output transitions of G' covers all the transitions in G'. From this re-examination, we conclude that the following set of relations must hold if H5 is satisfied:

$$\left.\begin{array}{ll} \sum_{t \in T_{out}} y_t[s] \geq 1 & \forall s \in T \\ C \times y_t = 0 & \forall t \in T_{out} \\ y_t[s] = 0 & \forall s \in T_{out} - \{t\} \\ y_t[s] \geq 0 & \forall s \in T \end{array}\right\} \quad \text{(A.7)}$$

In the following we first investigate some features of H5, then we propose two algorithms to check the satisfiability of H5 for a given Petri net, i.e., to identify if the given Petri net is a CO net.

A.3.1 Some conditions for H5

We denote by (t_1, t_2), when no confusion may arise, a non-oriented path joining t_1 with t_2. Such a path is said to be simple if each place on this path has exactly one input and one output transition.

Let us now give an example to illustrate the notion of *simple non-oriented path.* Figure A.4 shows a general situation in which a simple non-oriented path occurs. The simple non-oriented path (t_1, t_2) is $t_1 p_1 t_3 p_2 t_5 p_4 t_2$. Directed dotted lines represent the possible interaction of (t_1, t_2) with the rest of the net.

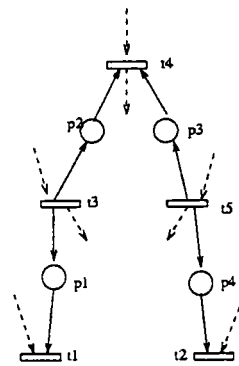


FIG. A.4 – *A non-oriented path*

Property A.8 (a necessary condition) *If a given acyclic Petri net G' satisfies H5 then there does not exist simple non-oriented path linking two output transitions.*

Proof: Consider a place p with a unique input transition t and a unique output transition t'. It is clear that $y[t] = y[t']$, for any T-invariant y. As a result, for any two output transitions t_1, t_2 connected through a simple non-oriented path, $y[t_1] = y[t_2]$, for any T-invariant y, which implies that any T-invariant support containing one of the two transitions contains also the other one. This shows that H5 cannot be satisfied if there exists a simple non-oriented path connecting two output transitions.

Remark A.4 *:* The condition given above is not a sufficient one. This can be shown by the Petri net G' in Figure A.5, which has no simple non-oriented path in it, and which does not fulfill H5 either.

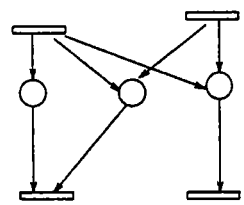


FIG. A.5 – *The insufficiency of "no non-oriented path" condition*

We notice that the Petri net in Figure A.5 is not consistent. Requiring that G' be consistent, we get the following sufficient condition.

Property A.9 (a sufficient condition) *A given acyclic Petri net G' satisfies H5 if G' is consistent, and if each minimal T-invariant support contains one and only one output transition.*

Proof: For each output transition t, let $y_t^{(1)}$, $y_t^{(2)}$, ..., $y_t^{(r_t)}$ be the set of minimal T-invariants related to t. Consider the following T-invariant

$$y_t = \sum_{i=1}^{r_t} y_t^{(i)}$$

Since each minimal T-invariant support contains one and only one output transition, we have

$$y_t[s] = 0, \ \forall s \in T_{out} - \{t\}$$

Thus

$$\|y_t\| \cap T_{out} = \{t\}$$

Furthermore

$$\bigcup_{t \in T_{out}} \|y_t\|$$

is equivalent to the union of all minimal T-invariant supports. Since G' is consistent, it is equal to T, i.e.,

$$\bigcup_{t \in T_{out}} \|y_t\| = T$$

Remark A.5 : The fact that the above condition is not a necessary one is shown in Figure A.1(c). The incidence matrix C of the Petri net in Figure A.1(c) is

$$C = \begin{bmatrix} 1 & 0 & 1 & -1 & -1 & 0 \\ 0 & 1 & 1 & -1 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 & -1 \end{bmatrix}$$

Obviously this net is consistent. There are two minimal T-invariants whose supports contain t_5. They are

$$y_{(5)}^1 = [2\ 1\ 0\ 1\ 1\ 0], \qquad \text{and} \qquad y_{(5)}^2 = [1\ 0\ 1\ 1\ 1\ 0]$$

The two minimal T-invariants whose supports contain t_6 are

$$y_{(6)}^1 = [1\ 2\ 0\ 1\ 0\ 1], \qquad \text{and} \qquad y_{(6)}^2 = [0\ 1\ 1\ 1\ 0\ 1]$$

while there also exists another minimal T-invariant

$$y_{5,6} = [0\ 0\ 3\ 2\ 1\ 1]$$

the support of which contains both t_5 and t_6.

The sufficient characterisation of H5 given in property A.9 is not obvious to recognise. A sufficient characterisation of H5 from graph theoretical point of view can be obtained as follow.

Property A.10 (a graph theory based sufficient condition) *If a given acyclic Petri net G' is consistent and the following property holds:*

$$H8: \qquad \forall t \in T,\ |t^\bullet| \leq 1$$

Then G' satisfies H5.

Proof: Due to property A.9 it is necessary to prove that each minimal T-invariant support contains one output transition. To prove this, it is enough to show that any T-invariant support containing at least two output transitions is not a minimal support.

For this purpose, consider a T-invariant y such that $y[t_1] > 0,\ y[t_2] > 0$, for $t_1,\ t_2 \in T_{out}$. Since G' is acyclic, it is possible to relabel the places $p_1,\ p_2,\ \ldots,\ p_m$ with $m = |P|$ such that there is no directed path from p_i to p_j, if $i > j$.

Let us introduce a subset of transitions $T^* = \{t \in T \mid y[t] > 0\}$. We construct a T-invariant y^* whose support is contained in T^*. This will conclude the proof.

Let

$$y^*[t_1] = 1, \text{ and } y^*[t] = 0 \qquad \forall t \in T_{out} - \{t_1\} \tag{A.8}$$

Starting from p_m, then $p_{m-1},\ \ldots,\ p_1$, we consider their input transitions and determine the value $y^*[t]$ as follows:

$$y^*[t] = \begin{cases} 0, & \text{if } t \notin T^* \\ \frac{1}{|^\bullet p_i \cap T^*|} \sum_{s \in p_i^\bullet} y^*[s], & \text{if } t \in T^* \end{cases} \tag{A.9}$$

with $p_i = t^\bullet$ which can be uniquely determined as a result of assumption H8.

Clearly for any place p_i without output transitions belonging to T^*, we have

$$y^*[t] = 0 \qquad \forall t \in^\bullet p_i \cup p_i^\bullet$$

which implies

$$\sum_{t \in^\bullet p_i} y^*[t] = \sum_{t \in p_i^\bullet} y^*[t] = 0 \tag{A.10}$$

For any place p_i such that at least one output transition belongs to T^*, there exists at least one input transition of p_i belonging to T^*, i.e., $|^\bullet p_i \cap T^*| > 0$. From relation A.9, we have

$$\sum_{t \in^\bullet p_i \cap T^\bullet} y^*[t] = \sum_{s \in p_i^\bullet} y^*[s]$$

which implies that

$$\sum_{t \in^\bullet p_i} y^*[t] = \sum_{s \in p_i^\bullet} y^*[s] \tag{A.11}$$

as $y^*[t] = 0, \quad \forall t \notin T^*$.

Relations A.10 and A.11 imply that y^* is also a T-invariant. Furthermore we have $y^*[t_2] = 0$, and $y^*[t] = 0, \quad \forall t \notin T^*$. Thus, $\|y^*\| \subseteq T^* - \{t_2\}$.

In fact, the conditions in property A.10 can be loosened as in the following:

Property A.11 (a loosened sufficient condition) *If a given acyclic Petri net G' does not contain source and sink places and H8 holds, then G' satisfies H5.*

Proof: Relabel the places p_1, p_2, ..., p_m as in the proof of property A.10. Construct iteratively a T-invariant y_t for each output transition $t \in T_{out}$. First,

$$y_t[t] = 1, \text{ and } y_t[s] = 0, \quad \forall s \in T_{out} - \{t\} \tag{A.12}$$

Starting from p_m, then p_{m-1}, $\cdots$, p_1, the other components of y_t are defined as follows:

$$y_t[s] = \frac{1}{|^\bullet p_i|} \sum_{\tau \in p_i^\bullet} y_t[\tau], \quad \forall s \in^\bullet p_i \tag{A.13}$$

Since each transition s has at most one output place, i.e., $|s^\bullet| \leq 1$, the related component $y_t[s]$ is uniquely defined either by A.12 or A.13. The vector y_t is obviously a T-invariant, since A.13 implies that

$$\sum_{s \in^\bullet p_i} y_t[s] = \sum_{s \in^\bullet p_i} \left(\frac{1}{|^\bullet p_i|} \sum_{\tau \in^\bullet p_i} y_t[\tau]\right) = \sum_{\tau \in p_i^\bullet} y_t[\tau], \quad \forall 1 \leq i \leq n$$

which is a necessary and sufficient condition for y_t to be a T-invariant.

H5 holds if the net G' is covered by T-invariants y_t, i.e., any transition s is related to at least one T-invariant y_{t^*} ($y_{t^*}[s] > 0$). To prove the latter statement, consider an elementary path from s to an output transition t^*. Since G' is acyclic and contains no source and sink places, such a path exists. Let ($s = t_0, q_1, t_1, q_2, \cdots, t_k, q_k, t_{k+1} = t^*$) denote this path with $s, t_1, \cdots, t_k, t^* \in T$, and $q_1, q_2, \cdots, q_k \in P$. From the definition of the T-invariant y_{t^*}, we have:

$$y_{t^*}[t_i] = \frac{1}{|^\bullet q_{i+1}|} \sum_{\tau \in q_{i+1}^\bullet} y_{t^*}[\tau] \geq \frac{1}{|^\bullet q_{i+1}|} y_{t^*}[t_{i+1}], \quad \forall 0 \leq i \leq k$$

which leads to:

$$y_{t^*}[s] = \frac{1}{|^\bullet q_1|} y_{t^*}[t_1] \geq \frac{1}{|^\bullet q_1||^\bullet q_2|} y_{t^*}[t_2] \geq \cdots \geq \prod_{i=1}^{k} \frac{1}{|^\bullet q_i|} y_{t^*}[t^*] > 0$$

This complets the proof.

The practical meaning of this result is that if the manufacturing system does not contain disassembly operations, then the model of the system is a CO net. This result is particularly used in our modular modelling approach (see the claim on page 115).

A.3.2 Identification algorithm

When checking the satisfiability of condition H5 for a given acyclic Petri net G', we can first make use of property A.8 to see if there exists a simple non-oriented path linking two output transitions. If not, properties A.9 and A.10 can be further utilised to check H5. In this subsection, we propose two algorithms to tackle the identification problem. We notice that a natural way to verify A.7 is to solve a linear programming problem as done in the following algorithm.

Algorithm A.1 ==========

INPUT: Incidence matrix C of a given Petri net.

OUTPUT: Satisfiability of condition H5.

STEP 1: Solve the following linear programming problem (PL1)

$$min \sum_{s\in T} \sum_{t\in T_{out}} y_t[s] \qquad (PL1-1)$$

$$s.t. \quad \sum_{t\in T_{out}} y_t[s] \geq 1, \quad \forall s \in T \qquad (PL1-2)$$

$$C \times y_t = 0, \quad \forall t \in T_{out} \qquad (PL1-3)$$

$$Y_t[\tau] = 0, \quad \forall \tau \in T_{out} - \{t\} \qquad (PL1-4)$$

STEP 2: If PL1 has a solution, then *H5 is satisfied*, other *H5 is not satisfied.*
========== end of algorithm ==========

Remark A.6 : In the above algorithm, the constraints of the linear programming problem PL1 are just the requirements of A.7. The correctness of the algorithm is obvious.

Remark A.7 : We can see that by means of the above algorithm it is straightforward to arrive at a conclusion for the CO identification problem. However the disadvantage that goes along with this straightforwardness is that the whole computational load is centred on one step, i.e., STEP 1.

In order to circumvent the drawback of the algorithm, we propose the following algorithm, in which we make use of "divide and conquer" method and approach the whole problem recursively. At each iteration, we try to find out the *maximal*

support T-invariant[2] concerning only one given output transition. When all such T-invariants for all the output transitions are found, we sum them up. Then it is sufficient to reach a conclusion as for the satisfiability of H5. The trick of searching the *maximal support T-invariant* at each iteration accelerates undoubtedly the speed of calculation.

Algorithm A.2 ==========

INPUT: Incidence matrix C of the given Petri net

OUTPUT: Satisfiability of H5

DECLARATION: $|T| \times 1$ matrices y, y_t

STEP 1: For each $t \in T_{out}$, solve the following problem (PL2)

$$max \sum_{\tau \in T} z_\tau \qquad (PL2-1)$$

$$s.t. \quad z_\tau \leq y_t[\tau], \quad \forall \tau \in T \qquad (PL2-2)$$

$$z_\tau \leq 1, \quad \forall \tau \in T \qquad (PL2-3)$$

$$C \times y_t = 0 \qquad (PL2-4)$$

$$y_t[s] = 0, \quad \forall s \in T_{out} - \{t\} \qquad (PL2-5)$$

$$z_\tau \geq 0, \; y_t[\tau] \geq 0, \quad \forall \tau \in T \qquad (PL2-6)$$

STEP 2: set $y := \sum_{t \in T_{out}} y_t$.

STEP 3: If $y > 0$, then *H5 is satisfied*, otherwise *H5 is not satisfied.*
========== end of algorithm ==========

Remark A.8 : In the above algorithm, the functions of constraints PL2-5 and PL2-4 are respectively the same as constraints PL1-4 and PL1-3 in Algorithm 4.1. The only difference is in constraints PL2-3 and PL2-2 which manifest the fact that we try to find out, at each iteration, the *maximal support T-invariant.* These two

[2] The notion *maximal support T-invariant* is not a standard term in Petri net parlance. However it is coined here to best express our intention to find out a T-invariant concerning a given output transition and having as many members as possible in its support.

constraints together with the objective function PL2-1 insure that for each output transition t the corresponding T-invariant obtained by solving PL2 is a *maximal support T-invariant.*

A.4 Concluding remarks

We have defined a new class of Petri net modules, CO nets, which were proved to be live, consistent, and reversible. Though it is unbounded, it can be made bounded if adequately controlled. The most interesting result is that the integration of CO net modules preserves all these properties. As a matter of fact, it was also proved that the integrated net is itself a CO net. These results provide a new way to model complex manufacturing systems and to verify that the required qualitative properties hold for the whole model by only checking the modules. We thus can cope with the complexity of real-life systems.

Annexe B

Simplification/Reduction of module models

In this part[1], we study in depth some algorithms pertaining to the simplification or reduction of Petri net modules. Such reduction or simplification procedures are useful in getting *simplified models* of the modules when a modular approach is adopted to deal with the complexity of real-life systems.

B.1 Computation of minimal support T-invariant

Many algorithms have been proposed for computing the set of minimal support T-invariants. In this section, we first introduce a property about the set of minimal support T-invariants, and then summarise one of the algorithms, the one proposed by Martinez and Silva [88]. Both the property and the algorithm will be used later. Especially the algorithm acts as a foundation to an efficient algorithm of reduced T-invariant computation proposed later in this chapter.

Property B.1 (set of minimal support T-invariants) *Let G be a general ordinary Petri net. Let $\Phi = \{Y_1, Y_2, ..., Y_q\}$ be the set of minimal support T-invariants and let $Y \in \mathbb{R}^{|T|}$ with $Y \geq 0$ be a non-negative real number vector such*

[1] The material in this part comes mainly from the following two articles:

1. Jean-Marie Proth, Liming Wang, Xiaolan Xie, "A hierarchical and modular approach to production management based on Petri nets – module simplification", in *Proc. Int. Conf. Emerging Technologies and Factory Automation*, October, Paris, 1995.

2. Liming Wang, Xiaolan Xie, "Reduced T-invariants of Petri nets", submitted for publication in *Journal of Discrete Event Dynamic Systems: Theory and Applications.*

that $C \times Y = 0$. *Then there exist non-negative reals* α_1, α_2, ..., α_q *such that*

$$Y = \sum_{i=1}^{q} \alpha_i \times Y_i$$

This property implies that the set of minimal support T-invariants is not only a *generating family* of the set of T-invariants but also a generating family of the convex cone $\{C \times Y = 0,\ Y \in \mathbb{R}^{|T|},\ Y \geq 0\}$.

Algorithm B.1 (Computation of minimal support T-invariants) [88]
Let C be the incidence matrix of a given Petri net model, n be the number of transitions, m be the number of places and I_n be an identity matrix of dimension n.

STEP 1: Set $A := C^T$, $\quad D := I_n$.

STEP 2: For $i = 1, 2, ..., m$, determine the minimal support T-invariants of the net composed of only $p_1, p_2, ..., p_i$ as follows:

2.1 Add to the matrix $[D \mid A]$ all the rows which are linear combinations of pairs of rows of $[D \mid A]$ and which annul the i-th column of A, where $[D \mid A]$ is a standard notion of juxtaposing two matrices.

2.2 Eliminate in matrix $[D \mid A]$ the rows in which the i-th column of A is non-null.

2.3 Eliminate the rows of D which cannot be minimal support T-invariants.

STEP 3: The rows of D corresponds to the minimal support T-invariants.

========== end of algorithm ==========

Let us give an example to illustrate the above algorithm. Consider a Petri net as shown in Figure B.1. The number associated with arc (p_2, t_7) corresponds to the related weight. When applying algorithm B.1 to the Petri net, we get a set of minimal support T-invariants. It comprises of the following eight T-invariants.

$Y_1 = [3\ 1\ 1\ 0\ 1\ 0\ 1\ 0]$ $\quad$ $Y_2 = [3\ 1\ 0\ 1\ 1\ 0\ 1\ 0]$
$Y_3 = [3\ 1\ 1\ 0\ 0\ 1\ 1\ 0]$ $\quad$ $Y_4 = [3\ 1\ 0\ 1\ 0\ 1\ 1\ 0]$
$Y_5 = [0\ 1\ 1\ 0\ 1\ 0\ 0\ 1]$ $\quad$ $Y_6 = [0\ 1\ 0\ 1\ 1\ 0\ 0\ 1]$
$Y_7 = [0\ 1\ 1\ 0\ 0\ 1\ 0\ 1]$ $\quad$ $Y_8 = [0\ 1\ 0\ 1\ 0\ 1\ 0\ 1]$

B.2 Reduced T-invariant: definition and properties

In this section we introduce the notion of reduced T-invariant and explore some properties relevant to reduced T-invariant. First let us introduce the notion of reduced T-invariant.

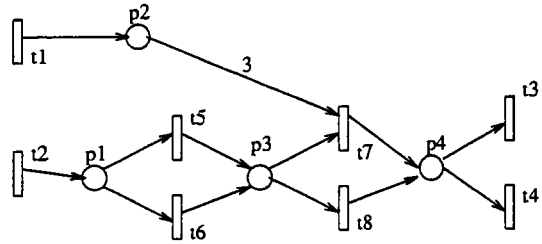


FIG. B.1 – *An illustrative Petri net*

B.2.1 Definition of reduced T-invariant

Definition B.1 (Reduced T-invariant) *Let $G = (P, T, F, W, M_0)$ be a Petri net model, and let S be a subset of transitions called* interface transitions, *i.e., $S \subseteq T$. A $|S| \times 1$ non-negative real number vector Z is called reduced T-invariant with respect to S or reduced T-invariant for short if and only if there exists $Y \in \mathbb{R}^{|T|}$ with $Y \geq 0$ such that $C \times Y = 0$ and $Y[t] = Z[t]$ for all $t \in S$.*

Clearly, to any T-invariant Y of a Petri net model is associated a unique reduced T-invariant Z with $Z[t] = Y[t]$ for all $t \in S$. The reverse is not true in general. Typically, to the same reduced T-invariant correspond several T-invariants. For example, the eight T-invariants of the Petri net in Figure B.1 correspond to only four reduced T-invariants, they are:

$$Z_1 = Z_3 = [3\ 1\ 1\ 0] \qquad Z_2 = Z_4 = [3\ 1\ 0\ 1]$$
$$Z_5 = Z_7 = [0\ 1\ 1\ 0] \qquad Z_6 = Z_8 = [0\ 1\ 0\ 1]$$

B.2.2 Properties of reduced T-invariant

We now explore some properties of reduced T-invariant. Without loss of generality, let us assume that $S = \{t_1, t_2, ..., t_r\}$ and $T = \{t_1, t_2, ..., t_r, t_{r+1}, ..., t_n\}$. Of course $n \geq r$. Let us introduce a matrix $D = [d_{ij}]_{r \times n}$ defined as follows:

$$d_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Clearly, the reduced T-invariant Z of any T-invariant Y can be determined as follows:

$$Z = D \times Y \tag{B.1}$$

Let Ψ be the set of non-negative real number vectors $Y = \mathbb{R}^{|T|}$ such that $C \times Y = 0$ with $Y \geq 0$. Let Ψ^r be the set of reduced T-invariants. From the definition of reduced T-invariants (see definition B.1), a $|S| \times 1$ real number vector Z belongs to Ψ^r if an only if there exists a vector $Y = \Psi$ such that $D \times Y = Z$.

This implies that the set of reduced T-invariants can be obtained from Ψ by means of a linear mapping D. Since Ψ forms a convex cone, we have:

Property B.2 *The set of reduced T-invariants Ψ^r forms a convex cone.*

From the properties of convex cones, property B.2 implies that Ψ^r can be generated by non-negative linear combination of a finite set of reduced T-invariants. This gives rise to the notions of generating family and minimal generating family.

Definition B.2 (generating family) *A set of reduced T-invariants $\{Z_1, Z_2, ..., Z_k\}$ of a module model G is called generating family if any reduced T-invariant Z can be expressed as a non-negative linear combination of $Z_1, Z_2, ..., Z_k$, i.e., there exist non-negative reals $a_1, a_2, ..., a_k$ such that $Z = a_1 \times Z_1 + a_2 \times Z_2 + ... + a_k \times Z_k$.*

Definition B.3 (minimal generating family) *A generating family of reduced T-invariants $\{Z_1, Z_2, ..., Z_k\}$ is called minimal if none of them can be obtained by non-negative linear combination of the others.*

In terms of convex analysis, the set of generating family corresponds to the set of extreme directions of the convex cone. Intuitively, the set of extreme directions should be unique. This conforms to the following property.

Property B.3 (uniqueness of the minimal generating family) *Let $\{Z_1, Z_2, ..., Z_k\}$ and $\{W_1, W_2, ..., W_l\}$ be two distinct minimal generating families of reduced T-invariants. Then $k = l$, and for each Z_i there exist a permutation I of $\{1, ..., l\}$ such that $Z_i = x_i \times W_{I(i)}$ for all $i \in \{1, ..., k\}$ and some positive real x_i.*

Proof: Since $\{Z_1, Z_2, ..., Z_k\}$ and $\{W_1, W_2, ..., W_l\}$ are generating families, there exist non-negative reals a_{ij} and b_{ij} for $1 \leq i \leq k$ and $1 \leq j \leq l$ such that

$$W_j = a_{1j} \times Z_1 + ... + a_{kj} \times Z_k, \quad \text{for } 1 \leq j \leq l \tag{B.2}$$

$$Z_i = b_{i1} \times W_1 + ... + b_{il} \times W_l, \quad \text{for } 1 \leq i \leq k \tag{B.3}$$

Replacing relations B.2 in B.3, we have:

$$Z_i = \sum_{j=1}^{l} (b_{ij} \times \sum_{\iota=1}^{k} a_{\iota j} Z_\iota), \quad \forall 1 \leq i \leq k$$

or equivalently,

$$Z_i = \sum_{\iota=1}^{k} (\sum_{j=1}^{l} b_{ij} \times a_{\iota j}) \times Z_\iota, \quad \forall 1 \leq i \leq k \tag{B.4}$$

From the definition of minimal generating families (see definition B.3), Z_i cannot be obtained by the non-negative linear combination of the other Z_ι with $\iota \neq i$. Thus relation B.4 implies that:

$$\sum_{j=1}^{l} b_{ij} \times a_{ij} = 1 \tag{B.5}$$

and

$$\sum_{j=1}^{l} b_{ij} \times a_{\iota j} = 0, \quad \forall \iota \neq i$$

The latter relation implies that

$$b_{ij} \times a_{\iota j} = 0, \quad \forall \iota \neq i \tag{B.6}$$

Similarly, by replacing relations B.3 in B.2, we obtain:

$$\sum_{i=1}^{k} a_{ij} \times b_{ij} = 1 \tag{B.7}$$

and

$$a_{ij} \times b_{i\phi} = 0. \quad \forall \phi \neq j \tag{B.8}$$

From relations B.3, for any $1 \leq u \leq k$, at least one of the non-negative reals b_{u1}, ..., b_{ul} is non-null. Let us assume that $b_{uI(u)} > 0$. From relation B.6, we have:

$$a_{\iota I(u)} = 0, \quad \forall \iota \neq u \tag{B.9}$$

Combining relation B.7 with $j = I(u)$, we have

$$b_{uI(u)} \times a_{uI(u)} = 1 \tag{B.10}$$

which implies that $a_{uI(u)} > 0$. From relation B.8 with $j = I(u)$,

$$b_{u\phi} = 0, \quad \forall \phi \neq I(u) \tag{B.11}$$

Combining relations B.3 and B.11

$$Z_u = b_{uI(u)} \times W_{I(u)}, \quad \forall 1 \leq u \leq k \tag{B.12}$$

Since Z_u cannot be obtained by the non-negative linear combination of the other Z_ι with $\iota \neq u$, we have $I(u) = I(i)$ for all $i \neq u$. This implies that

$$k \leq l \tag{B.13}$$

Similarly we can prove that for each $1 \leq v \leq l$, there exists a $J(v)$ with $1 \leq J(v) \leq k$ such that

$$W_u = a_{J(v)v} \times Z_{J(v)}, \quad \forall 1 \leq v \leq l$$

Since W_v cannot be obtained by the non-negative linear combination of the other W_j with $j \neq v$, we have $J(v) = J(j)$ for all $j \neq v$. This implies that $l \leq k$. Combining with relation B.13, $k = l$. This completes the proof.

Property B.4 *Let $\Phi = \{Y_1, Y_2, ..., Y_q\}$ be the set of minimal support T-invariants. The set of reduced T-invariants $\{DY_1, DY_2, ..., DY_q\}$ is a generating family of reduced T-invariants.*

Proof: Obvious as Φ is a generating family of the convex cone $\Psi = \{Y \in \mathbb{R}^{|T|}, C \times Y = 0, Y \geq 0\}$ and the set of the reduced T-invariants is a linear mapping of Ψ, i.e., $\Psi^r = D \times \Psi$.

This property provides a way to obtain the minimal generating family of the reduced T-invariants. It proceeds in three steps as follows. The first step computes the set of minimal support T-invariants. The second step derives a generating family of the reduced T-invariants as shown in property B.4. The third step computes the minimal generating family by removing the reduced T-invariants that can be obtained by non-negative linear combination of the others.

The major drawback with regard to this method is that the number of minimal support T-invariants may be very large while the number of reduced T-invariants in the generating family is small in most cases of interest. The next section is devoted to the development of an efficient algorithm which derives the minimal generating family without computing the minimal support T-invariants.

B.3 Computation of reduced T-invariant

Before getting on the issue of reduced T-invariant computation, we first address the simplification/reduction issue of a Petri net. More precisely, we introduce two Petri nets having the same set of interface transitions and the same generating family of reduced T-invariants as the original net. Two schemes of aggregation and a preliminary result are firstly introduced.

B.3.1 Two schemes of aggregation

Definition B.4 (aggregation I) *The aggregated model of a Petri net module $G = (P, T, F, W)$ with the set of interface transitions S and a minimal generating family of reduced T-invariants $\{Z_1, Z_2, ..., Z_r\}$ is a general Petri net $G^* = (P_S, S \cup Q, F^*, W^*)$ where $|P_S| = |S|$, $|Q| = r$, F^* is the set of directed arcs and W^* is the weighting function. It is defined as follows:*

(i) each interface transition $t \in S$ has a unique output place $p \in P_S$ and the weight of the arc (t, p) is equal to 1;

(ii) a new transition t_i is associated with each reduced T-invariant Z_i;

(iii) for any $p \in P_S$ such that $Z_i[^\bullet p] > 0$, an arc (p, t_i) of weight $Z_i[^\bullet p]$ is introduced.

If the set of interface transitions can be decomposed into a set of input transitions T_{in} and a set of output transitions T_{out}, an alternative aggregation scheme can be obtained as follows.

Definition B.5 (aggregation II) *The aggregated model of a module $G = (P, T, F, W)$ with $S = T_{in} \cup T_{out}$ and with a minimal generating family of reduced T-invariants $\{Z_1, Z_2, ..., Z_r\}$ is a general Petri net $G^* = (P_{in} \cup P_{out}, T_{in} \cup T_{out} \cup Q, F^*, W^*)$ where $|P_{in}| = |T_{in}|$, $|P_{out}| = |T_{out}|$, $|Q| = r$, F^* is the set of directed arcs and W^* is the weighting function. It is defined as follows:*

(i) each input transition $t \in T_{in}$ has a unique output place $p \in P_{in}$ and the weight of the arc (t, p) is equal to 1;

(ii) each output transition $t \in T_{out}$ has a unique input place $p \in P_{out}$ and the weight of the arc (p, t) is equal to 1;

(iii) a new transition t_i is associated with each reduced T-invariant Z_i;

(iv) for any $p \in P_{in}$ such that $Z_i[^\bullet p] > 0$, an arc (p, t_i) of weight $Z_i[^\bullet p]$ is introduced;

(v) for any $p \in P_{out}$ such that $Z_i[p^\bullet] > 0$, an arc (t_i, p) of weight $Z_i[p^\bullet]$ is introduced;

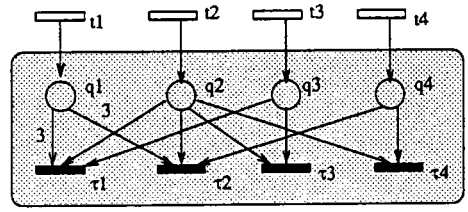
Let us re-examine the Petri net of figure B.1. The four reduced T-invariants given above form a minimal generating family. The aggregated models are given in figure B.2.

Let us notice that both aggregated models have the same minimal generating family of reduced T-invariants as the original net. Furthermore, in any aggregated model, each minimal T-invariant contains exactly one non-interface transition and each reduced T-invariant in the minimal generating family corresponds to exactly one minimal support T-invariant.

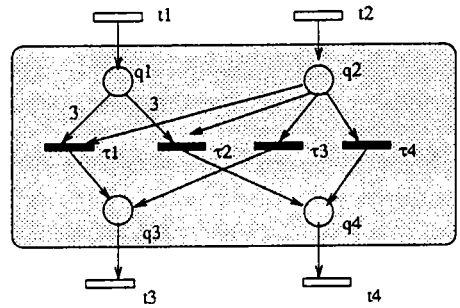
B.3.2 A preliminary result

A preliminary result which is to be used extensively for the computation of reduced T-invariants is introduced in this subsection. Algorithms are then proposed for both acyclic Petri nets and general ones.

Consider a Petri net G obtained from two different Petri nets G_a and G_b by merging some transitions as shown in figure B.3. Without loss of generality, let $G_a = (P_a, T_a \cup V, F_a, W_a)$, $G_b = (P_b, T_b \cup V, F_b, W_b)$ and $G = (P, T, F, W)$ where $P_a \cap P_b = \emptyset$, $P = P_a \cup P_b$, $T = T_a \cup V \cup T_b$, $F = F_a \cup F_b$ and $W = W_a \cup W_b$.



(a) Aggregation I



(b) Aggregation II

FIG. B.2 – *An illustration of aggregation schemes*

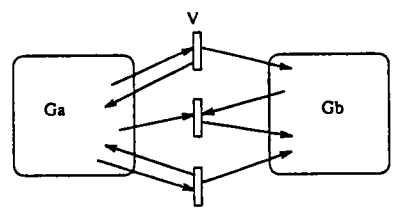


FIG. B.3 – *Integration of two Petri nets by transition merging*

Property B.5 *A vector* $Y = [Y_a,\ Y_V,\ Y_b]$ *is a T-invariant of* G *iff* $[Y_a,\ Y_V]$ *is a T-invariant of* G_a *and* $[Y_V,\ Y_b]$ *is a T-invariant of* G_b.

Let us replace G_b with another Petri net $G_c = (P_c,\ V \cup T_c,\ F_c,\ W_c)$ and let G' be the Petri net obtained by integrating G_a and G_c.

Property B.6 *If* G_b *and* G_c *have the same generating family of reduced T-invariants with respect to the set of transitions* V, $Y = [Y_a,\ Y_V,\ Y_b]$ *is a T-invariant of* G *iff there exists a T-invariant* Y' *of* G' *such that* $Y' = [Y_a,\ Y_V,\ Y_c]$. *Furthermore* G *and* G' *have the same generating family of reduced T-invariants with respect to any set of transitions* S *such that* $S \subseteq T_a \cup V$.

Proof: Since $Y = [Y_a,\ Y_V,\ Y_b]$ is a T-invariant of G, according to property B.5, $[Y_V, Y_b]$ is a T-invariant of G_b which implies that Y_V is a reduced T-invariant of G_b with respect to V. As a result, it is also a reduced T-invariant of G_c. From the property of reduced T-invariant, there exists a T-invariant of G_c of form $[Y_V, Y_c]$. Since $[Y_a, Y_V]$ is a T-invariant of G_a, then $Y' = [Y_a,\ Y_V,\ Y_c]$ is a T-invariant of G'.

The above reasoning implies that to each T-invariant $Y = [Y_a,\ Y_V,\ Y_b]$ of G corresponds at least one T-invariant of G' of form $Y' = [Y_a,\ Y_V,\ Y_c]$ and vice versa. Thus, G and G' have the same generating family of reduced T-invariant with respect to any set of transitions S such that $S \subseteq T_a \cup V$.

This property provides an efficient way to compute the minimal generating family of reduced T-invariants of a Petri net. For the net G of Figure B.3, we can first compute the minimal generating family of G_b with respect to V and then replace G_b by one of its aggregated models to obtain a simplified model G'. Property B.6 guarantees that G and G' have the same generating family of reduced T-invariants with respect to any set of transitions S such that $S \subseteq T_a \cup V$.

B.3.3 Acyclic Petri nets with input and output transitions

In this subsection we consider the case of acyclic Petri net with input transitions and output transitions. The Petri net in Figure B.1 is such a Petri net in which $t1$ and $t2$ are input transitions and $t3$ and $t4$ are output transitions. This section presents an algorithm for computing the minimal generating family of reduced T-invariants with respect to the set of input and output transitions[2], i.e., $S = T_{in} \cup T_{out}$. The algorithm proceeds in **three steps.**

The **first step** of the algorithm consists in representing the net by layers of nodes such that there is no path from any node of layer i to other nodes of layers

[2] Although a backward tracking procedure which starts from the output transitions and ends at the input transitions is very straightward in getting one T-invariant for a given acyclic Petri net, it is not advisable in obtaining a minimal generating family of the net since it lacks systematic mechanism for doing this.

i, $i+1$, ... This is possible by assigning to a node x to the layer $l(x)$ defined as follows:

$l(x)$:=number of nodes on the longest path leading from x to an output transition

with the convention of $l(x) = 1$ for all output transitions $x \in T_{out}$.

By this method, the Petri net of Figure B.1 can be decomposed into seven layers as shown in Figure B.4.

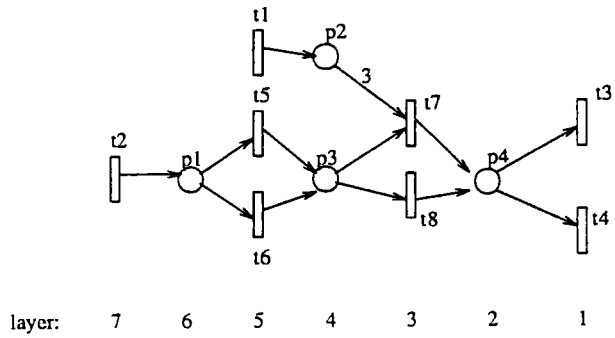


FIG. B.4 – *Decomposition of an acyclic Petri net into layers*

Since a Petri net is a bipartite graph and that $l(x)$ is the number of nodes on the longest path from x to an output transition, thus $l(x)$ is an odd number if x corresponds to a transition and $l(x)$ is an even number if x is a place. As a result, the layers 1, 3, 5, ... contain only transitions and layers 2, 4, 6, ... contain only places.

In general, after this first treatment, a transition of layer $2k+1$ may still be connected directly to a place of layer $2l$ with $l < k$. Similarly a place of layer $2l$ may still be connected directly to a transition of layer $2k-1$ with $k < l$. An example is given in Figure B.5.

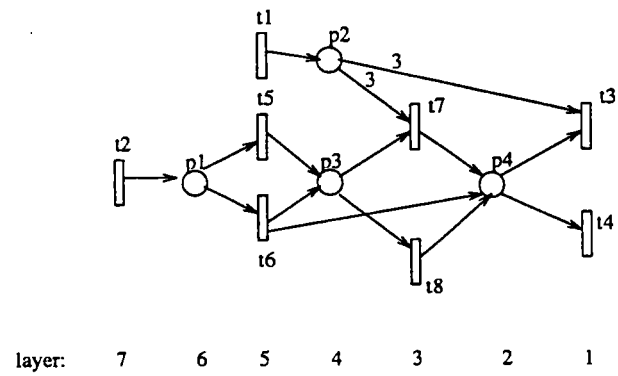
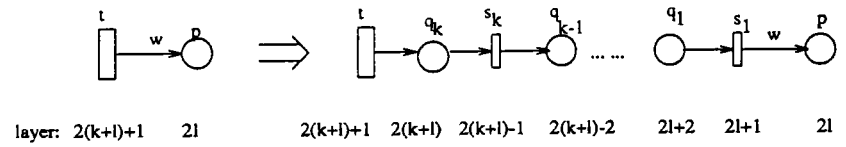


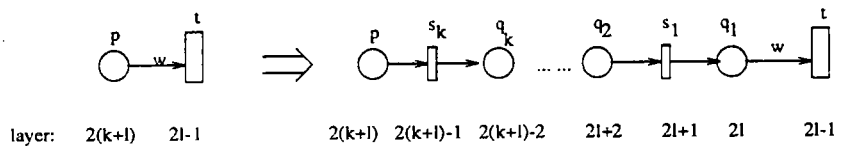
FIG. B.5 – *Cross-layer connections*

The **second step** of the algorithm consists in removing these cross-layer connections such that nodes of layer i are only connected to nodes of layer $i-1$ and $i+1$. This is done by introducing new places and new transitions as follows.

For any connection from a transition of layer $2(k+l)+1$ to a place of layer $2l$ with $k>0$ and $l>0$, k new transitions and k new places are introduced as shown in B.6(a). For any connection from a place of layer $2(k+l)$ to a transition of layer $2l-1$ with $k>0$ and $l>0$, k new transitions and k new places are introduced as shown in Figure B.6(b). It is obvious that these transformations do not change the T-invariant relationship.



(a) Connections from transitions to places



(b) Connections from places to transitions

FIG. B.6 – *Elimination of cross-layer connections*

For the Petri net of Figure B.5, the introduction of two new places $p5$ and $p6$ and two new transitions $t9$ and $t10$ eliminates all cross-layer connections and leads to the Petri net model of Figure B.7.

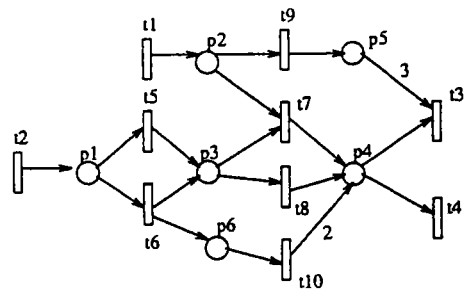


FIG. B.7 – *Removing cross-layer connections*

The **third step** of the algorithm is an iterative process. At the beginning , only nodes of layers 1,2, and 3 are considered. Let $G(2)$ be the related Petri net. Clearly layer 3 nodes are input transitions and layer 1 nodes are output transitions. The algorithm B.1 is applied to obtain the set of minimal T-invariants of $G(2)$.

We then derive the aggregated model of $G(2)$ according to aggregation II (see definition B.5). Let us denote it as $G^*(2)$. Figure B.8 shows the Petri nets $G(2)$ and $G^*(2)$ of the Petri net of Figure B.4.

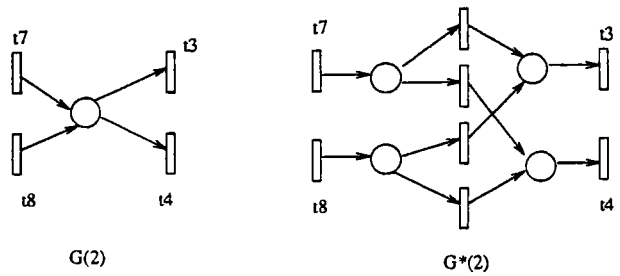


FIG. B.8 – $G(2)$ *and* $G^*(2)$

At the second iteration, the Petri net $G(3)$ composed of nodes of layers 1,2,3,4, and 5 is considered. The goal is to compute its aggregated model $G^*(3)$. For this purpose, we replace its subnet $G(2)$ by $G^*(2)$ to obtain a new Petri net $G'(3)$. Again the algorithm B.1 is applied to obtain the set of minimal T-invariants of $G'(3)$. Its aggregation model can then be derived. Thanks to property B.6, this aggregation model is also the aggregated model of $G(3)$. i.e., $G^*(3)$. The algorithm continues in a similar way for the other layers. After having considered all the layers at iteration k, the resulting aggregated model $G^*(k)$ is also the aggregated model of the original net G.

Figure B.9 shows the Petri nets $G'(3)$ and $G^*(3)$ of the Petri net of Figure B.4. Figure B.10 gives the net $G'(4)$ while the final aggregated model $G^*(4)$ is identical to the one of Figure B.2(b).

Formally speaking, the algorithm proceeds as follows:

Algorithm B.2 (For acyclic Petri nets) ===

STEP 1: Determine the layer $l(x)$ of any node x and let $2K-1$ be the number of layers.

STEP 2: Eliminate the cross-layer connections by introducing new places and new transitions.

STEP 3: For $k = 2$ to K,

3.1 Extract the net $G(k)$ composed of nodes of layers $1, 2, ..., 2k-1$;

3.2 Replace the subnet $G(k-1)$ by its aggregated model $G^*(k-1)$ and construct a new net $G'(k)$;

3.3 Apply algorithm B.1 to obtain the set of minimal support T-invariants of $G'(k)$;

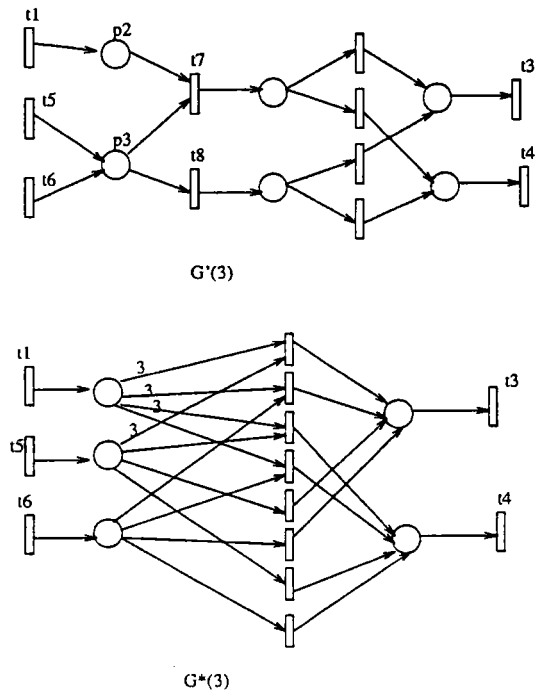


FIG. B.9 – $G'(3)$ *and* $G^*(3)$

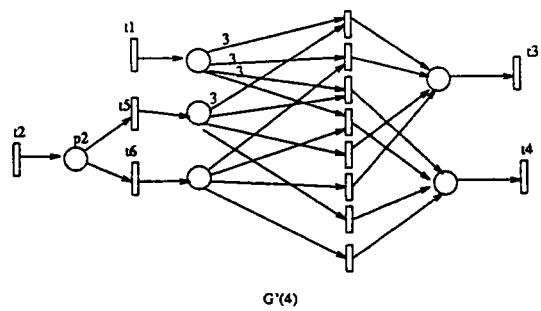


FIG. B.10 – $G'(4)$

3.4 Derive the generating family of reduced T-invariants of $G'(k)$ with respect to its input transitions and its output transitions.

3.5 Construct the aggregated model of $G'(k)$, i.e., $G^*(k)$, which is also the aggregated model of $G(k)$ thanks to property B.6.

========== end of algorithm =========

It should be pointed out that when computing the minimal support T-invariants of $G'(k)$ by algorithm B.1 in step 3.3, we need not to start with $D := I_n$ and $A := C^T$. Since the minimal support T-invariants of the subnet $G^*(k-1)$ are already known, the minimal support T-invariants of the subnet of $G'(k)$ composed of only places belonging to $G^*(k-1)$ are known. As a result of this, we can start from this point by letting D denote this set of minimal support T-invariants and by setting $A[i,j] = D[i,\bullet] \times C[j,\bullet]^T$ where $D[i,\bullet]$ corresponds to the i-th minimal support T-invariant and $C[j,\bullet]$ is the row of the incidence matrix corresponding to the j-th place of layer $2k-2$, i.e., a place belonging to $G'(k)$ but not $G^*(k-1)$.

Property B.7 *The algorithm B.2 provides the minimal generating family of a given Petri net G at step K, i.e., the aggregated model $G^*(K)$ is an aggregated model of G.*

Proof: Let us show by induction that the subnet $G(k)$ has the same generating family of reduced T-invariants, with respect to its input and output transitions $T_{2k-1} \cup T_1$, where T_{2k-1} denotes transitions of layer $2k-1$, as the net $G'(k)$ for all k. This is trivially true at step $k = 2$ as $G'(2)$ is identical to $G(2)$ and thus $G^*(2)$ is the aggregated model of $G(2)$.

Let us assume that $G(k)$ and $G'(k)$ have the same generating family of reduced T-invariants with respect to their input and output transitions $T_{2k-1} \cup T_1$.

We denote by G_a the Petri net composed of transitions belonging to $T_{2k+1} \cup T_{2k-1} \cup T_1$, places of layer $2k$ and arcs connecting these places to transitions belonging to $T_{2k+1} \cup T_{2k-1}$. It is worth noticing that the transitions belonging to T_1 are isolated nodes in G_a. Clearly $G(k+1)$ is obtained from G_a and $G(k)$ by merging transitions belonging to $T_{2k-1} \cup T_1$ and $G'(k+1)$ is obtained from G_a and $G'(k)$ by merging transitions belonging to $T_{2k-1} \cup T_1$. According to property B.6, $G(k+1)$ and $G'(k+1)$ have the same generating family of reduced T-invariants with respect to their input and output transitions $T_{2k+1} \cup T_1$.

B.3.4 General Petri nets

Two approaches are possible in the general case. The **first approach** consists in first transforming the initial Petri net model into an acyclic Petri net having the same T-invariant relationship and then deriving the minimal generating family of the reduced T-invariants by using algorithm B.2. To obtain an acyclic Petri net,

we replace an arbitrary transition on each circuit by two new transitions and a new place as shown in Figure B.11. Clearly this transformation does not change the T-invariants of the original net. Furthermore the transition t_e becomes an output transition while t_b becomes an input transition. Using this transformation, any Petri net can be transformed into an acyclic Petri net having the same minimal support T-invariants.

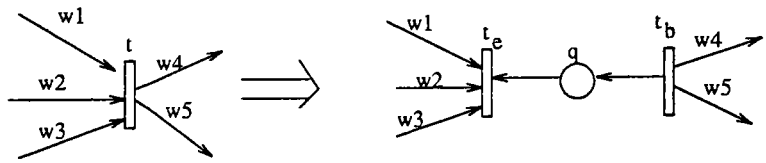


FIG. B.11 – *An elementary transformation to obtain acyclic Petri net*

The major drawback of this approach is the introduction of extra-places and extra-transitions. In the following, we propose a second approach which directly computes the generating family of reduced T-invariants of a general Petri net with respect to any set of interface transitions S. The **second approach** proceeds in three steps and in a similar way as algorithm B.2.

The **first step** consists in assigning to each node x a number $l(x)$ called layer such that $l(p) = 2k$ with $k = 1, 2, ..., K-1$ for all place $p \in P$ and $l(t) = 2k-1$ with $k = 1, ,2, ,..., K$ for all transition $t \in T$. Figure B.12 gives an example in which the nodes are assigned to 7 layers.

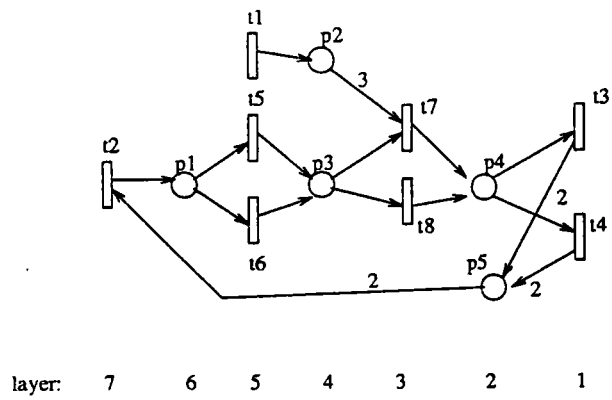


FIG. B.12 – *A Petri net with seven layers with* $S = \{t_1, t_2\}$

The **second step** of the algorithm consists in removing these cross-layer connections such that nodes of layer i are only connected to nodes of layers $i-1$ and $i+1$. As in the case of acyclic Petri nets, this is done by introducing new places and new transitions as follows.

For any connection from a node of layer $2k+l$ to a node of layer l, k new places and k new transitions are introduced as in the case of acyclic Petri net. However

in the case of general Petri net, there may exist connections from nodes of layer 1 to nodes of layer $2k+l$. For any connection from a place of layer $2l$ to a transition of layer $2(k+l)+1$ with $k>0$ and $l>0$, k new transitions and k new places are introduced as shown in Figure B.13(a). For any connection from a transition of layer $2l-1$ to a place of layer $2(k+l)$ with $k>0$ and $l>0$, k new transitions and k new places are introduced as shown in Figure B.13(b). It is obvious that these transformations do not change the T-invariant relationship.

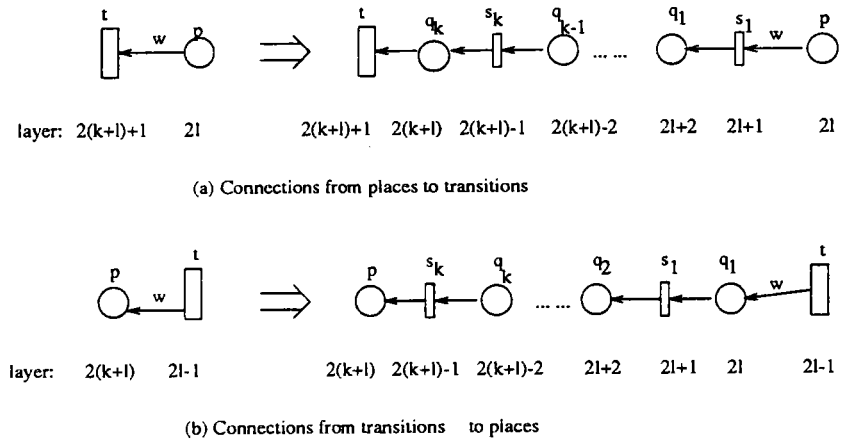


FIG. B.13 – *Elimination of cross-layer connections from low layer nodes to high layer nodes*

By the end of the second step, the net of Figure B.12 becomes the net of Figure B.14.

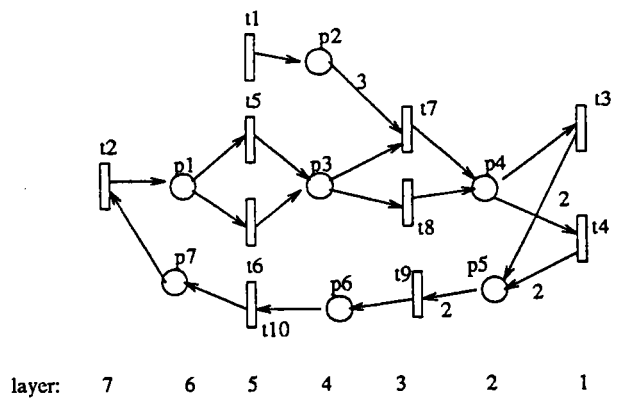


FIG. B.14 – *Model obtained by removing cross-layer connections*

The **third step** is an iterative process similar to the third step of algorithm B.2. At the beginning, only nodes of layers 1, 2 and 3 are considered. Let $G(2)$ be the related Petri net. The algorithm B.1 is applied to obtain the set of minimal

T-invariants of $G(2)$. We then derive the generating family of reduced T-invariants with respect to $S_2 = (S \cap T_1) \cup T_3$ where T_{2k-1} denotes the set of transitions of layer $2k-1$. The aggregated model of $G(1)$ according to definition B.4, i.e., aggregation I, is then obtained. Let us denote it as $G^*(2)$. Figure B.15 shows the Petri nets $G(2)$ and $G^*(2)$ of the Petri net of Figure B.12 (see also Figure B.14).

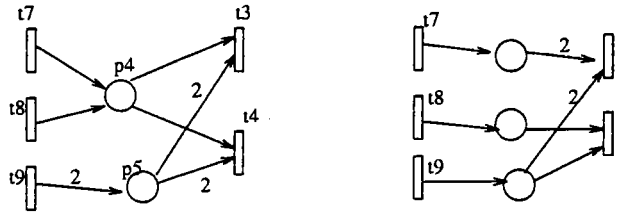


FIG. B.15 - $G'(2)$ *and* $G^*(2)$ *with* $S_2 = \{t_7, t_8, t_9\}$

At the second iteration, the Petri net $G(3)$ composed of nodes of layers 1,2,3,4, and 5 is considered. The goal is to compute its aggregated model $G^*(3)$ with respect to the set of interface places $S_3 = (S \cap S_2) \cup T_5$. For this purpose, we replace its sub-net $G(2)$ by $G^*(2)$ to obtain a new Petri net $G'(3)$. Again the algorithm B.1 is applied to obtain the set of minimal T-invariants of $G'(3)$. Its aggregated model can then be derived. Thanks to property B.6, this aggregated model is also the aggregated model of $G(3)$, i.e., $G^*(3)$. The algorithm continues in a similar way for the other layers. After having considered all the layers at iteration k, the resulting aggregated model $G^*(k)$ is also the aggregated model of the original net G.

Figure B.16 shows the Petri nets $G'(3)$ and $G^*(3)$ of the Petri net of Figure B.12. Figure B.17 gives the nets $G'(4)$ and $G^*(4)$. There is a unique reduced T-invariant $\{3t_1, t_2\}$.

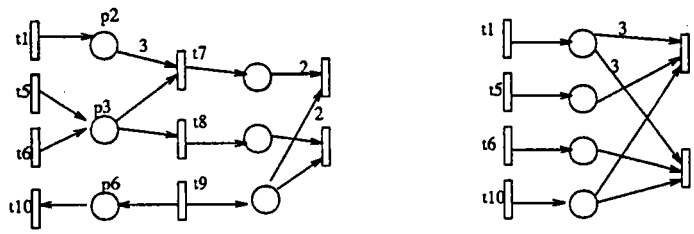


FIG. B.16 - $G'(3)$ *and* $G^*(3)$ *with* $S_3 = \{t_1, t_5, t_6, t_{10}\}$

Formally speaking, the algorithm proceeds as follows:

Algorithm B.3 (For general Petri nets) ==========

STEP 1: Assign each place p to a layer $l(p) = 2l$ and each transitions t to a layer $l(t) = 2k - 1$ under constraints $1 \leq l \leq K - 1$ and $1 \leq k \leq K$.

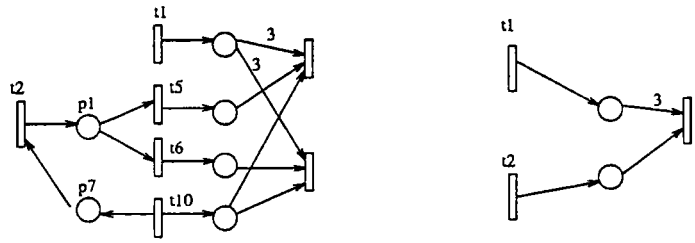


FIG. B.17 - $G'(4)$ *and* $G^*(4)$ *with* $S_4 = \{t_1, t_2\}$

STEP 2: Eliminate the cross-layer connections by introducing new places and new transitions.

STEP 3: For $k = 2$ to K,

3.1 Extract the net $G(k)$ composed of nodes of layers 1, 2, ,..., $2k-1$.

3.2 Replace the sub-net $G(k-1)$ by its aggregated model $G^*(k-1)$ and construct a new net $G'(k)$;

3.3 Apply algorithm B.1 to obtain the set of minimal support T-invariants of $G'(k)$;

3.4 Derive the generating family of reduced T-invariants of $G'(k)$ with respect to $S_k = (S \cap S_{k-1}) \cup T_{2k-1}$ where $S_1 = T_1$;

3.5 Construct the aggregated model of $G'(k)$ which is also the aggregated model of $G(k)$ thanks to property B.6.

========== end of algorithm ==========

Property B.8 *The algorithm B.3 provides the minimal generating family of the net G at step K, i.e., the aggregated model $G^*(K)$ is an aggregated model of G.*

B.3.5 Efficiency of the algorithms

We have integrated the algorithms proposed previously into the prototype software HMPS (see annex D). Our extensive computational exercises show that these algorithms can handle Petri nets with hundreds of transitions and places in less than 1 CPU second on a SUN SPARC 1+ station. They never cause obstacles to the function of the software.

B.4 Concluding remarks

We have introduced a new notion of Petri nets, i.e., reduced T-invariants, which can be used to characterize the flow relationship of a Petri net model viewed from some interface places. Efficient algorithms have been developped to determine the generating family of the reduced T-invariants. The notion of reduced T-invariants captures the input-output relationship of the model. It is particularly useful when complex systems are modelled in a modular way.

Annexe C

Modular modelling and hierarchical management

Although there exist a number of frameworks (see chapter 1) for the study of discrete event systems, specifically discrete manufacturing systems, we have noted in Table 1.1 that Petri nets are a *complete* tool that allows one to model, analyse and evaluate performances of manufacturing systems. Petri nets are appropriate in manufacturing environments because:

- they offer a graphical and precise formalism which facilitates the communication among different working teams,
- they handle explicitly states and events at the same time,
- they capture interactions of concurrent and sequential events,
- they can formally represent parallelism and synchronisation which are common phenomena in manufacturing systems,
- they allow progressive modelling by using stepwise refinement or modular composition,
- they also allow easy integration of deterministic and/or stochastic timing constraints,
- they offer a well founded theory for the qualitative verification of net model properties (liveness, boundedness, consistency, reversibility, etc.),
- they provide a family of tools for specification, modelling, qualitative validation, performance evaluation and real-time execution,
- they also provide a family of tools for the study of various functions at various levels (global coordination, local control) of manufacturing systems, thus form an important aid for integrating the whole system.

We also notice that Petri nets are not *perfect*. As observed in practice, modern manufacturing systems are often very complex. When complex systems are dealt with by means of Petri nets, the so called "combinatorial explosion" of states is inevitable, as in all other *state-oriented* approaches. However, through modularisation and hierarchisation, complexity can be managed properly.

In this chapter, we focus on the modular modelling and hierarchical management of complex manufacturing systems within the framework of Petri net theory.

C.1 System representation

We are interested in the management aspect of manufacturing systems. By management we mean planning and scheduling the relevant activities involved in the system running. Thus we regard BOMs (Bills of Material) as the basis of our modelling work. A manufacturing system is characterised by the BOMs of the products handled by the system. The BOM of a product is a precedence constraint relation among all the operations required to manufacture the product. Operations are the results of interactions between products and resources. In order to simplify modelling and focus on the interesting activities, we only consider machines as resources, though there may well be other kinds of resources, e.g., AGVs, fixtures, pallets, etc..

We identify several types of operations:

- *transformation*: A basic transformation operation has an input buffer and an output buffer as shown in Figure C.1(a). The function of a transformation operation is to *transform* the products in its input buffer into the products in its output buffer by a certain processing procedure.

  The procurement of the products to be processed in the input buffer may come from different sources, and the processed products in the output buffers may be used for different purposes. This is shown in Figure C.1(b).

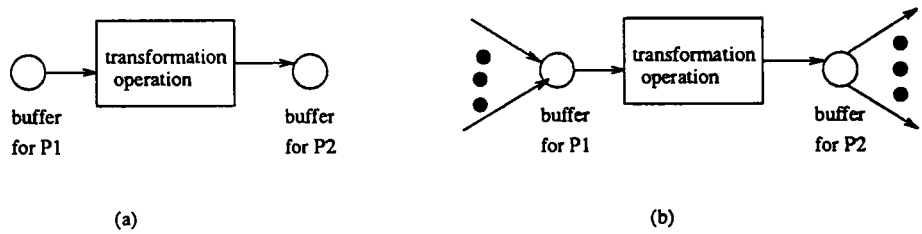


FIG. C.1 - *Transformation operations*

Basic transformation operations can be cancatenated to form a series of transformation operations, as shown in Figure C.2.

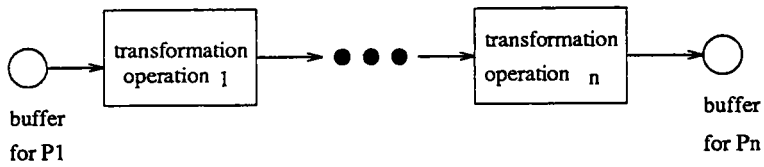


FIG. C.2 – *Transformation operations in series*

- *assembly*: An assembly operation is one which has several input buffers and one output buffer. It takes the components stored in the input buffers and *assembles* them into a new product stored in the output buffer. An assembly operaiton is shown in Figure C.3.

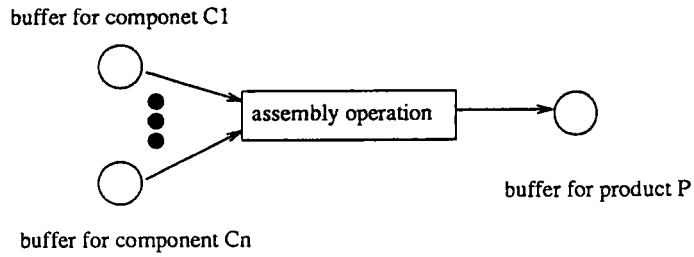


FIG. C.3 – *An assembly operation*

- *disassembly*: A disassembly operation is *dual* to an assembly one in the sense that it takes one product from the input buffer and separates out the components and deposits them in the corresponding output buffers. Such an operation is shown in Figure C.4.

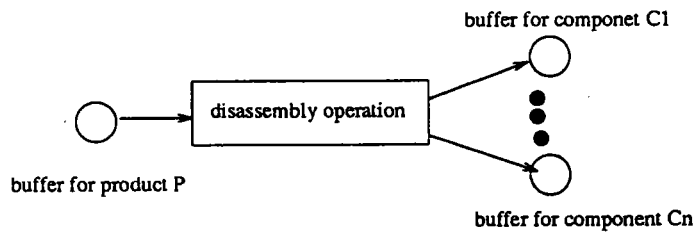


FIG. C.4 – *A disassembly operation*

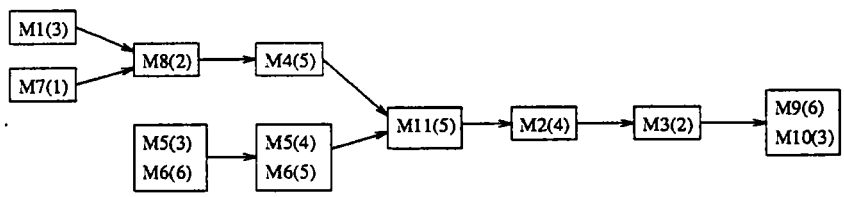
Remark C.1 *Each operation discussed above, either transformation or assembly or disassembly, can be executed by more than one resource (machine). The machines that can carry out a common operation are said to be functionally identical machines for this operation. It is easy to depict the multiple-machine situations for each of the operations discussed above.*

Remark C.2 *The operations discussed above constitute the most widely encountered situations in real-world manufacturing systems. From our experience, disassembly operations rarely happens in practice, though they do appear in certain circumstances. We exclude disassembly operations from our later discussions without restricting ourselves too much in practical applications of our results.*

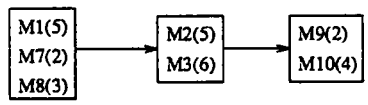
Let us present an example to illustrate what a system we are interested in looks like.

Example C.1

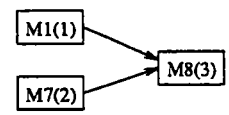
Suppose that we have a manufacturing plant, and that we are asked to produce 3 products, P_1, P_2, and P_3 on 10 machines, M_1 to M_{10}. Note that there may exist more than 10 machines in the plant, and there may have more than 3 products that the plant can process. All the data about machines and products for the plant may be stored in a product database. However we are interested in the 3 products that are to be produced on the given 10 machines in a certain period of time. The BOMs of the 3 products are shown in Figure C.5.[1] *The numbers in the parentheses after the machine labels are durations for corresponding processing. Each square represents an operation. If there are more than one machine in a square, it means that these machines are functionally identical machines for the operation. Note also that, for simplicity, the input and output buffers for all operations are not explicitly drawn in the figure.*



(a) BOM for P1



(b) BOM for P2



(c) BOM for P3

FIG. C.5 – *BOMs for P_1, P_2, P_3*

1. It should be emphasized that this is a very simple *picture* of BOMs. In fact, BOMs have many attributes.

C.2 System modelling by means of Petri nets

As seen from the previous discussions, Petri nets are one of the satisfactory tools that support the study of modelling, analysis, and performance evaluation of manufacturing systems. In this section, we investigate system modelling issue.

Even in the framework of Petri net theory, there exist different modelling practices, depending on points of views taken by different practitioners in studying different aspects of complex systems [37] [68] [105] [120].

We adopt the following convention to model manufacturing processes as described in the last section:

- An operation carried out by a resource is represented by a transition associated with a place representing the resource. Such a place is called resource place.
- A buffer is represented by a place called process place.
- Precedence constraints between operations are respresented by arcs connecting transitions and places.

Such a simple mapping between manufacturing processes and their Petri net models makes Petri nets readily acceptable by practitioners. It is thus straightforward to represent a system by means of Petri net.

For example, the Petri net in Figure C.6(a) can be interpreted as a model for a manufacturing process consisting of one transformation operation represented by transition t_1. This operation is carried out by a machine represented by resource place r. The other transitions t_0 and t_2 represent the procurement of raw materials and the distribution of finished products. Process places p_1 and p_2 represent respectively the input and the output buffer for the operation. In the same spirit, the Petri net in Figure C.6(b) can be interpreted as a manufacturing process consisting of two cancatenated transformation operations carried out serially by machines respresented by resource places r_1 and r_2. Similarly, the Petri net in Figure C.6(c) models a manufacturing process consisting of one transformation operation that can be carried out alternatively by two functionally identical machines represented by resource places r_1 and r_2.

In Figure C.7(a) is depicted the Petri net model of an assembly operation carried out by a machine represented by resource place r. Transitions t_0 and t_1 represent the procurement of components C_1 and C_2 the buffers of which are respectively represented by process places p_1 and p_2. The assembled product is stored in output buffer represented by process place p_3. Finally, transition t_3 represents the distribution of the product. Figure C.7(b) shows a situation in which there exist two functionally identical machines for an assembly operation.

As we have noticed, transitions drawn by vertical bars represent the input or output activities, and those drawn by rectangles represent the operational activities.

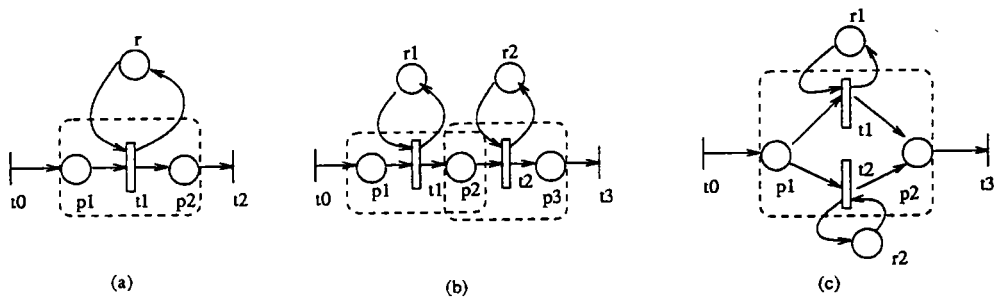


FIG. C.6 – *PN models for transformation operations*

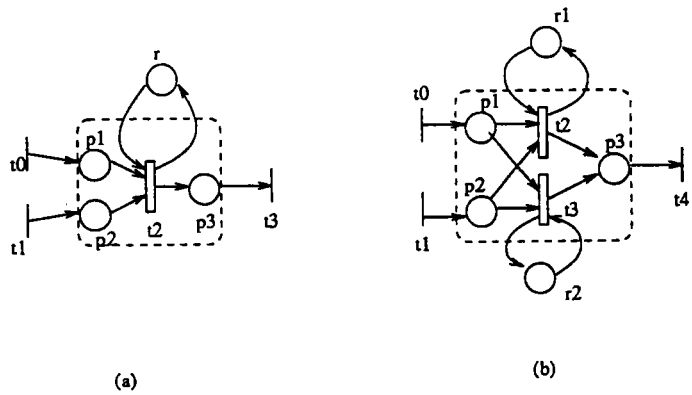


FIG. C.7 – *PN model for assembly operations*

It is thus not difficult to figure out the Petri net models of the BOMs of P_1, P_2, P_3 described in Example C.1.

Example C.2

The Petri net model for the system of Example C.1 is shown in Figure C.8. Note that for brevity, the processing times associated with transitions are not present. For the same reason, only the resource place r2 corresponding to machine M_2 is present. A token in a resource place represents the fact that the resource is available.

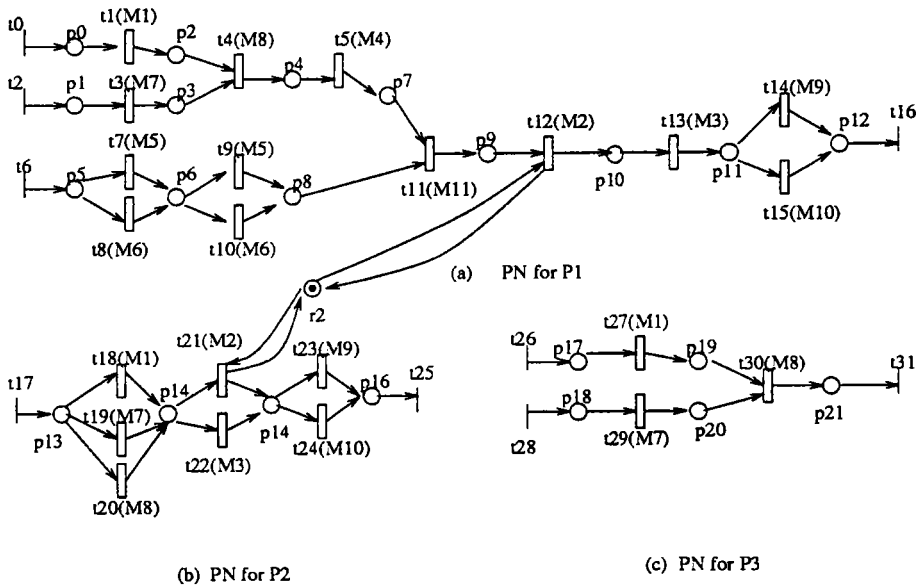


FIG. C.8 – *Petri net models for P_1, P_2, P_3*

The following algorithm presents a general procedure for transforming BOMs into Petri nets.

Algorithm C.1 (Build Petri net model of BOMs) ===

STEP 1: Represent each buffer for semi-finished product as a place, called process place. Represent each machine as a place, called resource place.

STEP 2: If a transformation operation can be carried out by $m(\geq 1)$ machines, then represent the operation by as many rectangle transitions (say, $t_1, \cdots, t_m$) with a common input process place (say, p_1) and a common output process place (say, p_2). Add arcs (p_1, t_i) and (t_i, p_2), $\forall i = 1, \cdots, m$. Also add m other resouce places (say, $r_1, \cdots, r_m$), and add arcs (t_i, r_i) and (r_i, t_i), $\forall i = 1, \cdots, m$.

STEP 3: If an assembly operation can be carried out by $m(\geq 1)$ machines, then represent the operation by as many rectangle transitions (say, $t_1, \cdots, t_m$) with as many input process places (say, $p_1, \cdots, p_m$) and one output process place

(say, p_{m+1}). Add arcs (p_i, t_j) and (t_j, p_{m+1}), $\forall i,\ j = 1, \cdots, m$. Also add m other resource places (say, $r_1, \cdots, r_m$), and add arcs (t_i, r_i) and (r_i, t_i), $\forall i = 1, \cdots, m$.

STEP 4: Represent the procurement operation of each raw material by a vertical bar transition (say, t_1) and represent the buffer for the raw material by a process place (say, p_1). Add an arc (t_1, p_1). Also represent the distribution operation of each finished product by a vertical bar transition (say, t_2) and represent the buffer for the finished product by a process place (say, p_2). Add an arc (p_2, t_2).

STEP 5: Merge the process places representing a common buffer into one process place representing the buffer. Merge the resource places representing a common machine into one resource place representing the machine.

C.3 Modular modelling

In the last section, we have shown how to model with Petri nets a manufacturing system represented by the BOMs of the products it handles. In practice, machines are grouped into certain cells. some products may traverse several cells in order to be processed on some machines in the cells it traverse. Therefore the information of machines in different cells need to be taken into account in the models.

Basically there are two ways of modelling a complex manufacturing system in a modular manner:

1. - Model the modules individually,
   - Find the connections among the modules, and
   - Compose them into the whole system model.

2. - Inspect the whole system from a global viewpoint,
   - *Cut* or decompose it into connected modules according to the knowledge about the formation of the system,
   - Handle the modules and the whole system in a harmonious way.

We take the second way at least for the following reasons:

- In the first way, once a system is *a priori* treated individually, the connections among the modules may be quite arbitrary and often difficult to find. Thus this way is of a *myopic* nature.

- In the second way, the modules may be identified from some *a priori* information about the composition of the system, moreover the connections among the modules are kept quite naturally, and the modules and the whole system are thus dealt with in a *natural* and *harmonious* way.

Although, in practice, how to *cut* or *decompose* the whole system into modules may vary according to different purposes, one has to adopt certain common practices:

1. No two modules are allowed to contain common machines. That is, each machine can only be contained in one module.

2. The size[2] of the modules should be reasonable. Two extreme cases should be avoided:

   - each module contains only one machine,
   - all machines are grouped into one module.

   The former is undesirable since it does not help in dealing with complexity if numerous machines are present. The latter is useless since it does not take advantage of modular approach.

3. More often, grouping machines into cells may be a complicated task if some performance measures (e.g., workload balance, inter-module traffic etc.) are to be optimized while system topology serves as a constraint.

When machines are grouped into cells, the Petri net model is also *broken down* into modules. Three cases have to be considered:

Case 1: If $m(\geq 2)$ machines performing respectively m cancatenated transformation operations (see Figure C.6(b) where $m = 2$) are to be grouped into m different modules, the m transitions associated with the m machines are further separated by adding new places and transitions. This is shown in Figure C.9 where $m = 2$.

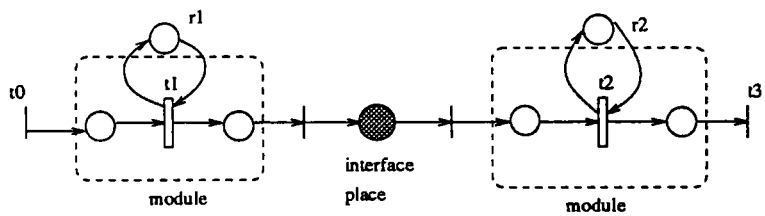


FIG. C.9 – *Decomposing cancatenated transformation operations*

Case 2: If $m(\geq 2)$ functionally identical machines for a transformation operation (see Figure C.6(c) where $m = 2$) are to be grouped into m different modules, each transition associated with a machine is *expanded* as a subnet with an input transition and an output transition belonging to corresponding module. This is shown in Figure C.10 where $m = 2$.

2. Here the size of a module is defined to be the number of machines contained in the module.

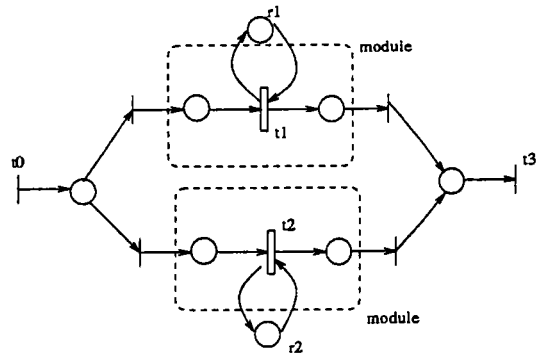


FIG. C.10 – *Decomposing a transformation operation*

Case 3: If $m(\geq 2)$ functionally identical machines for an assembly operation (see Figure C.7(b) where $m = 2$) are to be grouped into m different modules, each transition associated with a machine is *expanded* as a subnet with m input transitions and an output transition belonging to corresponding module. This is shown in Figure C.11 where $m = 2$.

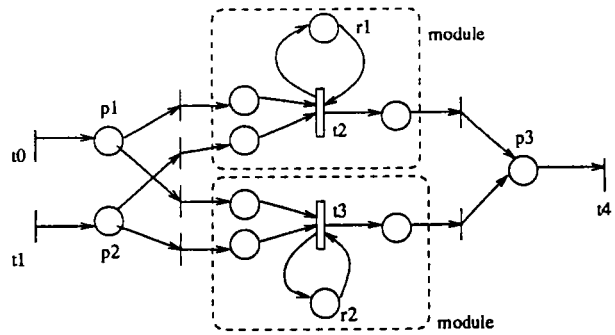


FIG. C.11 – *Decomposing an assembly operation*

Consider the system of Example C.1. Its Petri net model is given in Figure C.8.

Example C.3

Suppose that we are informed by the manager of the plant that the machines are grouped into 3 cells:

- *in cell 1 there are* M_1, M_7, M_8, M_4,
- *in cell 2 there are* M_5, M_6,
- *in cell 3 there are* M_2, M_3, M_9, M_{10}, M_{11}

Based on such information about cell formation, and the decomposition procedure previously described, the Petri net models of the cells can be readily obtained according to the decomposition procedure previously described. Figure C.12 shows the Petri net model of cell 1, while the Petri net models of cells 2 and 3 are shown in Figure C.13. The shaded places represent interface places.

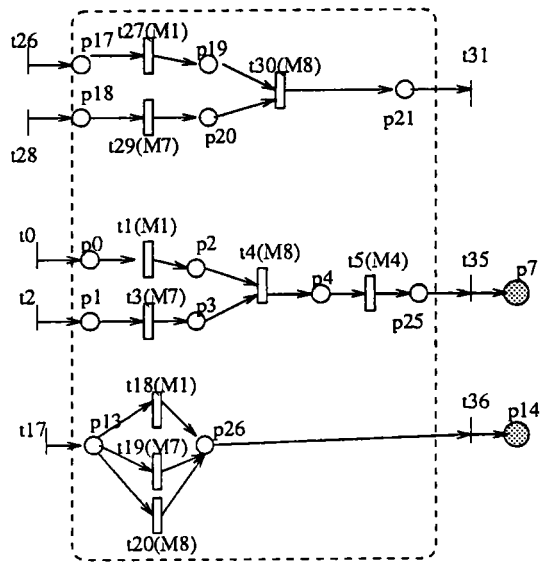
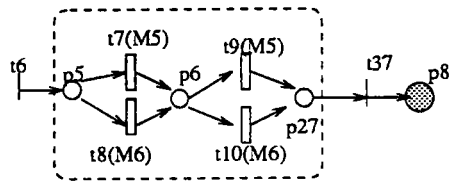


FIG. C.12 – *Petri net model for cell 1*

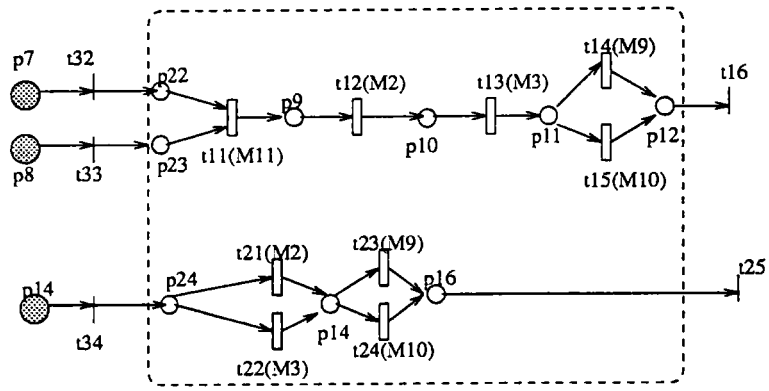
C.4 System features

Manufacturing systems with the following features are considered:

- Machines are grouped into cells *a priori*,
- Times needed to perform operations in the system are deterministic and are given *a priori*,
- The cost of holding one intermediate product and the cost for holding one end product of all product types are known *a priori*,
- The cost of backlogging one end product is known *a priori*,
- No backlogging is allowed for intermediate products,
- Disassembly operations are not present in BOMs,



(a) Petri net model for cell 2



(b) Petri net model for cell 3

FIG. C.13 – *Petri net models for cells 2 and 3*

Since the systems considered do not contain disassembly operations, according to property A.11 on page 81, we **claim** that the Petri net models of our systems are CO nets.

Consequently, the whole system can be modelled as a CO net. Each cell is also modelled by a CO net module, the connection among the CO net modules satisfies H6 and H7 as specified on page 72.

The modular approach we take to model such kinds of systems can be summarized in Figure C.14. Based on the information about BOMs of the products, we model the whole system by means of Petri net. As demonstrated previously, the Petri net model is a CO net. According to the decomposition procedure presented in the last section, the CO net model is further decomposed into interconnected CO net modules. Such a decomposition is crucial to the implementation of a hierarchical management scheme proposed hereafter.

C.5 Hierarchical management scheme

Once the Petri net model of the whole system is known and the decomposition of the whole system into CO net modules is done, we propose to use a three-level

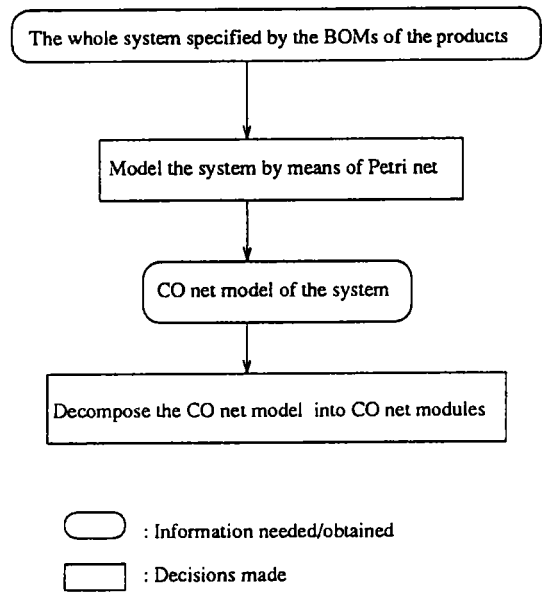


FIG. C.14 – *Modular modelling*

hierarchical management scheme:

- Simplified whole system planning: The goal of simplified whole system planning is to specify module outputs and inputs in the sub-periods.
- Cell planning: Cell planning is to give production quantity for each operation in each module in the elementary periods of the first sub-period. Each sub-period usually contains several elementary periods.
- Scheduling: Scheduling is to specify the start time for each firing of each transition in the modules.

More precisely, the hierarchical management scheme is implemented as follows (see Figure C.15):

1. First we simplify each module by extracting from it a minimal generating family of reduced T-invariants (see annex B), which leads to a simplified whole system. For example, the simplified whole system of the system presented in Example C.1 (see also Example C.2 and Example C.3) is drawn in Figure C.16. The solid black rectangles represent fictitious transitions that *lump* the simplification information in terms of reduced T-invariants.
2. Based on the information about the simplified whole system, medium term planning which is done on the simplified whole system can be started. In order to determine module inputs and outputs in the sub-periods, we take

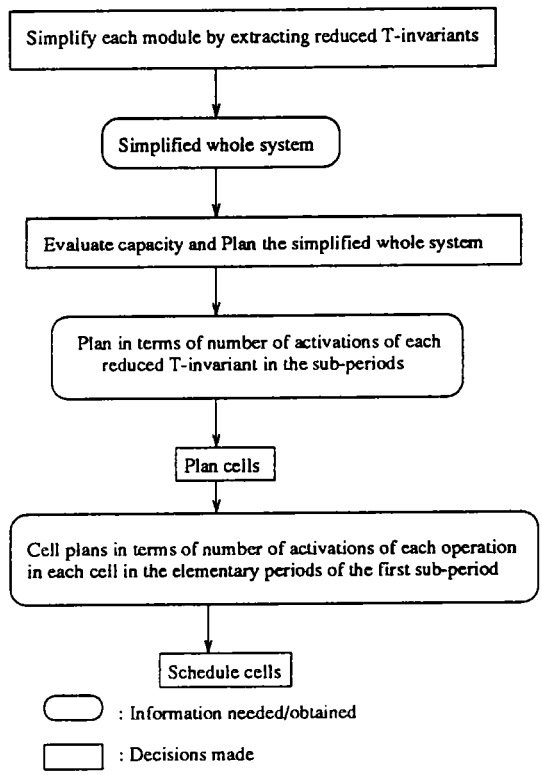


FIG. C.15 – *Hierarchical planning*

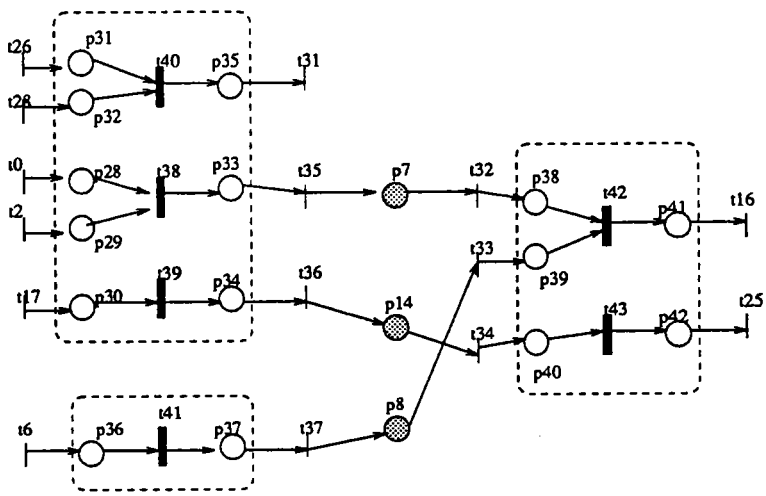


FIG. C.16 – *Petri net model for the simplified whole system*

as decision variables a set of scalars that correspond to the number of activations of each reduced T-invariant in the sub-periods. We then solve a minimization problem which takes the sum of inventory and backlogging costs as the objective function. This problem is subject to capacity and flow constraints. The result is the number of activations of each reduced T-invariant in the sub-periods. From this result, it is straightforward to get the information about module inputs and outputs in the sub-periods (Details are presented in section C.7).

It is worth noting that capacity evaluation is a crucial issue for all kinds of hierarchical approaches to production management (see e.g. [44]). As such, we take a careful study of this issue before getting on solving the simplified whole system planning problem. We propose a simple procedure which approximate *pessimistically* the system capacity, and make use of it in planning the simplified whole system.

3. Once the information about the module inputs and outputs is given, cells can be managed concurrently under the premise that sufficiently large quantities of intermediate products are stored in the inter-cell buffers such that cells are *decoupled* in that each cell can function independently of the others. In planning each cell, we also solve a minimization problem which takes the inventory costs and backlogging costs as objective function and is subject to the flow and capacity constraints of the cell. Another constraint is that the medium term plan concerning the inputs and outputs of the cell must be respected. The solution to the minimization problem is the number of firings of each transition of the module in the elementary periods of the first sub-period. (Details are presented in section C.8.)

4. With the results of cell planning, cell scheduling can now be activated. A meta-heuristic, simulated annealing, and an implicit enumeration approach, branch-and-bound, are used to deal with the scheduling task. (see also section C.8.)

C.6 Capacity evaluation

C.6.1 Relevant notations

Let $\bar{K}$ be the number of sub-periods in a planning horizon. Let H and $\bar{H}$ be respectively the length of each elementary period[3] and the length of each sub-period. Thus the number of elementary periods in a sub-period is $K = \frac{\bar{H}}{H}$.

The demand for the final product related to output transition $t \in T_{out}$ in elementary period k is denoted by $d_{t,k}$. Thus the demand for the final product

3. We follow the convention that all elementary periods are of equal length

related to output transition $t \in T_{out}$ in each sub-period κ is: $D_{t,\kappa} = \sum_{k=1}^{K} d_{t,k}$. Let α_t, β_t, $\forall t \in T_{out}$ be respectively the unit inventory cost and the unit backlogging cost for the final product related to output transition $t \in T_{out}$.

At the level of simplified system planning, we are interested in the number of activations of the reduced T-invariants of the modules. Thus it is natural for us to take some variable associated with reduced T-invariants as decision variable. Before doing so, let us introduce some useful notations.

Let $G^j = (P^j \cup R^j, T^j, F^j, M_0^j)$, $j \in \{1, \cdots, N\}$ be the j-th CO net module, C^j be the incidence matrix of G^j, T_{in}^j and T_{out}^j be respectively the set of input transitions and that of output transitons of G^j. Denote T_r^j to be the set of transitions of G^j such that all transitions in the set are associated with the same resource represented by $r \in R^j$, i.e., $T_r^j = \{t \in T^j | (t,r) \in F^j \text{ and } (r,t) \in F^j\}$. Assume that in the simplified whole system there are totally b reduced T-invariants: Z_1, $\cdots$, Z_b. For each Z_i, $i \in \{1, \cdots, b\}$, there must exist a module G^j, $j \in \{1, \cdots, N\}$ of which Z_i is a reduced T-invariant. Let $\mu : \{1, \cdots, b\} \mapsto \{1, \cdots, N\}$ be such a mapping, then μ is uniquely defined by $j = \mu(i) \in \{1, \cdots, N\}$, $\forall i \in \{1, \cdots, b\}$. Let also $\Upsilon_j = \{i | \mu(i) = j, 1 \leq i \leq b\}$ be the index set of reduced T-invariants of G^j.

We then associate with each reduced T-invariant Z_i a scalar $z_{i,\kappa}$ interpreted as the number of activations of Z_i in sub-period κ.

C.6.2 Evaluating system capacity

In order to plan the simplified whole system, we need to get knowledge about material flow and system capacity.

While the information about material flow is easy to get from the simplified whole system model, the system capacity information is not so obvious. Let Ω represent the region in which the capacity of each module G^j is not violated if $z_{i,\kappa}$'s $(\forall \kappa, \forall i \in \Upsilon_j)$ are used as the numbers of activations of the reduced T-invariants of G^j. Since the capacity information comes from each cell, Ω must be determined via the flow of each G^j and the workload of resources in G^j resulting from such a flow. Thus an ideal representation of Ω can be formulated as follows:

For each module G^j, $j = 1, \cdots, N$, for each κ, $\exists Y_{j,\kappa} \geq 0$ such that

$$C^j \times Y_{j,\kappa} = 0 \tag{C.1}$$

$$Y_{j,\kappa}[t] = \sum_{i \in \Upsilon_j} z_{i,\kappa} \times Z_i[t], \quad \forall t \in T_{in}^j \cup T_{out}^j \tag{C.2}$$

$$\sum_{t \in T_r^j} Y_{j,\kappa}[t] \times \theta_t \leq \bar{H}, \quad \forall r \in R^j \tag{C.3}$$

C.1 is the flow relation.

C.2 demonstrates that the (semi-)flow[4] $Y_{j,\kappa}$ must conform to the specification of $z_{i,\kappa}$ for input and output transitions.

C.3 is the workload condition resulting from the flow $Y_{j,\kappa}$. θ_t is the firing time of transition t.

C.6.3 An approximation of system capacity

The system capacity as defined via C.1, C.2, and C.3 represents an ideal case, i.e., the information about system capacity used at the high level planning comes from the detailed knowledge about each module. However this does not provide any help in reducing complexity, since it takes a strong flavour of monolithic approach.

In order to overcome this difficulty, we propose the following algorithm to get a simple evaluation of system capacity in each sub-period κ.

Algorithm C.2 (Capacity evaluation) *=====*

STEP 1: For each Z_i, $i \in \Upsilon_j$, solve the following problem which is to find a flow that minimises the maximum workload of resources:

$$Min\ W \tag{C.4}$$

subject to:

$$C^j \times y = 0 \tag{C.5}$$

$$y[t] = Z_i[t], \quad \forall t \in T^j_{in} \cup T^j_{out} \tag{C.6}$$

$$\sum_{t \in T^j_r} y[t] \times \theta_t \leq W, \quad \forall r \in R^j \tag{C.7}$$

Let y_i be the optimal solution to the problem corresponding to Z_i.

STEP 2: Set

$$\Theta_{i,r} = \begin{cases} \sum_{t \in T^j_r} y_i[t] \times \theta_t, & if\ r \in R^j \\ 0, & if\ r \notin R^j \end{cases}$$

STEP 3: $\Omega' = \{(z_1, \cdots, z_b) | \sum_{i \in \Upsilon_j} z_i \times \Theta_{i,r} \leq \bar{H}, \forall r \in R^j,\ \forall j = 1, \cdots, N\}$ is a simple pessimistic evaluation of system capacity.

In STEP 1 of algorithm C.2, we intend to find out, for each reduced T-invariant Z_i beloging to a module G^j, a flow that maximises the throughput of the module. In [128], it is shown that this is equivalent to the minimization of maximum workload, which is the minimization problem formulated in STEP 1.

In STEP 2, we calculate $\Theta_{i,r}$ which represents the workload of resource r if such a flow corresponding to Z_i is followed.

4. If there is no confusion, the term "flow" is also used. In fact, they are used interchangeably with the term *T-invariant*.

Finally in STEP 3, we try to find out a set of scalars $(z_i, \cdots, z_b)$, where z_i is interpreted as the number of times we take the flow corresponding to Z_i. Obviously, if $(z_i, \cdots, z_b)$ is to be feasible, the workload of each resource in each sub-period is not allowed to exceed the length of the sub-period. It can be proved (see Theorem C.1) that Ω' is a simple pessimistic evaluation of system capacity.

Note that this approach is based, in some way, on an "ideal" production requirement. But we know that the capacity of an aggregated system depends on the production mix. This means that the capacity computed above is an approximation of the real capacity. Furthermore, it is impossible to evaluate how "far" this approximation is from the real capacity.

However, we can at least prove that this evaluation is pessimistic, that is, using this evaluation, it is possible to find a feasible solution at the next lower level. This is the concern of the following theorem.

Theorem C.1 *Ω' in STEP 3 of algorithm C.2 is a pessimistic evaluation of system capacity.*

Proof: System capacity is represented by C.1, C.2, and C.3. Assume that $(z_i, \cdots, z_b) \in \Omega'$, i.e.,

$$\sum_{i \in \Upsilon_j} z_i \times \Theta_{i,r} \leq \bar{H}, \quad \forall r \in R^j, \ \forall j = 1, \cdots, N, \ \forall \kappa$$

then

$$\sum_{i \in \Upsilon_j} z_i \sum_{t \in T_r^j} y_i[t] \times \theta_t \leq \bar{H}$$

or equivalently

$$\sum_{t \in T_r^j} \theta_t \sum_{i \in \Upsilon_j} z_i \times y_i[t] \leq \bar{H}$$

Set $Y_{j,\kappa} = \sum_{i \in \Upsilon_j} z_i \times y_i$, then C.3 holds.

Also, since $C^j \times y_i = 0$, we have

$$C^j \times Y_{j,\kappa} = \sum_{i \in \Upsilon_j} z_i \times C^j \times y_i = 0$$

thus C.1 holds.

Finally, since C.6 holds, we have:

$$Y_{j,\kappa}[t] = \sum_{i \in \Upsilon_j} z_i \times y_i[t] = \sum_{i \in \Upsilon_j} z_i \times Z_i[t], \quad \forall t \in T_{in}^j \cup T_{out}^j$$

Thus C.2 holds. This completes the proof.

C.7 Simplified whole system planning

The purpose of the simplified whole system planning is to obtain, based on the information about system capacity and product demands, a plan for the inputs and outputs of the cells such that some objective function is optimised. We take the sum of inventory costs and backlogging costs as the objective function. The goal is to minimise such an objective function over a planning horizon.

In terms of Petri nets, we are looking for the numbers of firings of the input and output transitions of all the cells in the sub-periods. As explained previously, we take as decision variables the numbers $z_{i,\kappa}$ of activations of the reduced T-invariants in the sub-periods.

The problem of planning the simplified whole system can be reduced to the determination of $z_{i,\kappa}$'s. Once the $z_{i,\kappa}$'s are known for a given sub-period κ, we then know how many times each reduced T-invariant should be activated in κ. Since the reduced T-invariants concern the input and output transitions of the modules, we thus get a complete knowledge about how many times input transitions and output transitions of the modules should fire in sub-period κ.

In order to determine $z_{i,\kappa}$'s, we solve a minimization problem which takes the sum of inventory and backlogging costs as the objective function, and is subject to flow and capacity constraints.

Formally, the problem of calculating $z_{i,\kappa}$'s can be formulated as follows:

$$Min \sum_{\kappa=1}^{\bar{K}} (\sum_{q \in Q} M_\kappa(q) \times \gamma_q + \sum_{t \in T_{out}} (\alpha_t \times S^+_{t,\kappa} + \beta_t \times S^-_{t,\kappa})) \tag{C.8}$$

subject to:

$$M_{\kappa+1}(q) = M_\kappa(q) + \sum_{i=1}^{b} (\sum_{t \in {}^\bullet q} Z_i[t] \times z_{i,\kappa} - \sum_{t \in q^\bullet} Z_i[t] \times z_{i,\kappa}) \qquad \forall q \in Q, \forall \kappa \tag{C.9}$$

$$S_{t,\kappa+1} = S_{t,\kappa} + \sum_{i=1}^{b} Z_i[t] \times z_{i,\kappa} - D_{t,\kappa} \qquad \forall t \in T_{out}, \ \forall \kappa \tag{C.10}$$

$$(z_{i,\kappa}, \ \forall 1 \leq i \leq b) \in \Omega', \quad \forall \kappa \tag{C.11}$$

C.8 is the objective function which has two items that represent different costs.

The first item is the cost of token inventory in interface places over the planning horizon, where $M_\kappa(q)$ is the quantity of tokens (intermediate products) in interface place $q \in Q$ at the beginning of period κ, and γ_q is the unit inventory cost of intermediate products in place $q \in Q$.

The second item is inventory cost and backlogging cost of the final products over the horizon. $S_{t,\kappa}$ is the inventory level of the buffer for the final product related to output transition $t \in T_{out}$ at the beginning of sub-period κ. $S^+_{t,\kappa} = max\{0, S_{t,\kappa}\}$ is the surplus, and $S^-_{t,\kappa} = max\{0, -S_{t,\kappa}\}$ is the backlog.

C.9 is a constraint on the evolution of token quantities in interface places in each sub-period.

C.10 is a constraint on the evolution of quantities of final products in each sub-period.

C.11 specifies the capacity constraint that must be satisfied at the medium term planning level. Note that the simple though pessimistic approximation Ω' as proposed previously is used to specify the capacity constraint. This procedure is illustrated in section C.9.

C.8 Cell planning and cell scheduling

The aim of cell planning is to obtain a plan for each cell, based on the information about cell capacity and the simplified whole system plan. A plan for a cell specifies the actual number of activations of each operation in the cell in an elementary period.

In terms of Petri nets, what is to be sought in cell planning is, for each Petri net module, the number of firings of transitions of the module in an elementary period. T-invariants are used to achieve this goal.

The problem of planning module G^j in the elementary periods of sub-period κ can be formulated as follows:

$$Min \sum_{\forall k \ in \ \kappa} \sum_{t \in T^j_{out}} (\alpha_t \times V^+_{t,k} + \beta_t \times V^-_{t,k}) \tag{C.12}$$

subject to:

$$M^j_{k+1} = M^j_k + C^j \times y^k, \quad \forall k \tag{C.13}$$

$$V_{t,k+1} = V_{t,k} + y^k[t] - d_{t,k}, \quad \forall t \in T^j_{out} \tag{C.14}$$

$$\sum_{t \in T^j_r} y^j[t] \times \theta_t \leq H, \qquad \forall r \in R^j \tag{C.15}$$

$$\sum_{\forall k \ in \ \kappa} y^k[t] = z_{i,\kappa} \times Z_i[t], \quad \forall t \in T^j_{in} \cup T^j_{out}, \ \forall i \in \Upsilon_j \tag{C.16}$$

The objective function C.12 reflects the fact that the sum of inventory and backlogging costs is to be minimised. $V_{t,k}$ is the inventory level of the buffer for the (intermediate) product related to transition $t \in T^j_{out}$ at the beginning of elementry period k.

C.13 is a token flow constraint. M^j_k is the marking of module G^j in period k. y^k is a T-invariant of G^j in period k. C.14 means that a feasible flow should meet demand requirement.

C.15 is the capacity constraint.

C.16 demonstrates the fact that the medium term plan must be respected.

Cell planning gives the number of firings of transitions of each cell in an elementary period. The task of cell scheduling is to determine *when* each firing should

occur such that the required number of firings of transitions are satisfied in the elementary period.

In our work, two algorithms are used to tackle this problem. One is *brach and bound (BAB)* [78], which is an implicit enumeration algorithm. The other is *simulated annealing (SA)* [71], which is a meta-heuristic with analogy to the annealing process in thermodynamics. It is noted that BAB is suitable for small problems, while for larger ones, SA should be preferred though it may, in some cases, take a long time to get a feasible result, i.e., a schedule which enables completion of the short-term planning tasks within an elementary period.

C.9 Computational experience

In this section, we report our computational experience in applying the previously proposed modular and hierarchical approach to system management. We first illustrate the procedure of system capacity approximation by studying an example, and report some comparative results between monolithic approach and our modular and hierarhical approach. We then present a complete solution to the management problem concerning the example.

C.9.1 Capacity evaluation

Consider the system in Example C.1. From Figure C.16, we see that there are 6 reduced T-invariants $Z_1, \cdots, Z_6$ (corresponding respectively to each solid black rectangle).

Let us assume that the correspondence between Z_i, $i \in \Upsilon_j$, $j = 1, 2, 3$ and $t \in T_{in}^j \cup T_{out}^j$ is shown in Table C.1.

Reduced T-invariants	Transitions concerned
Z_1	t_{31}, t_{26}, t_{28}
Z_2	t_{35}, t_0, t_2
Z_3	t_{36}, t_{17}
Z_4	t_{37}, t_6
Z_5	t_{16}, t_{32}, t_{33}
Z_6	t_{25}, t_{34}

TAB. C.1 – *Correspondence between reduced T-inv. and transitions*

then, $\mu : \{1, \cdots, 6\} \mapsto \{1, 2, 3\}$ is uniquely defined as shown in Table C.2.

Thus, in this specific case, the problem of planning simplified whole system is (refer to relations C.8 to C.11)

i	1	2	3	4	5	6
$\mu(i)$	1	1	1	2	3	3

TAB. C.2 - *Definition of* μ

$$Min\ \sum_{\kappa=1}^{\bar{K}}(M_\kappa(p_7)\gamma_{p_7} + M_\kappa(p_8)\gamma_{p_8} + M_\kappa(p_{14})\gamma_{p_{14}}$$
$$+\alpha_{t_{16}}S^+_{t_{16},\kappa} + \beta_{t_{16}}S^-_{t_{16},\kappa} + \alpha_{t_{25}}S^+_{t_{25},\kappa} + \beta_{t_{25}}S^-_{t_{25},\kappa} + \alpha_{t_{31}}S^+_{t_{31},\kappa} + \beta_{t_{31}}S^-_{t_{31},\kappa})$$

subject to:

$$M_{\kappa+1}(p_7) = M_\kappa(p_7) + Z_2[t_{35}]z_{2,\kappa} - Z_5[t_{32}]z_{5,\kappa}$$
$$M_{\kappa+1}(p_8) = M_\kappa(p_8) + Z_4[t_{37}]z_{4,\kappa} - Z_5[t_{33}]z_{5,\kappa}$$
$$M_{\kappa+1}(p_{14}) = M_\kappa(p_{14}) + Z_3[t_{36}]z_{3,\kappa} - Z_6[t_{34}]z_{6,\kappa}$$
$$S_{t_{31},\kappa+1} = S_{t_{31},\kappa} + Z_1[t_{31}]z_{1,\kappa} - D_{t_{31},\kappa}$$
$$S_{t_{16},\kappa+1} = S_{t_{16},\kappa} + Z_5[t_{16}]z_{5,\kappa} - D_{t_{16},\kappa}$$
$$S_{t_{25},\kappa+1} = S_{t_{25},\kappa} + Z_6[t_{25}]z_{6,\kappa} - D_{t_{25},\kappa}$$

Now if we take C.11 as capacity constraint, we have to find flows $Y_{j,\kappa}$, $\forall\kappa$, $j = 1,2,3$ such that C.1, C.2, and C.3 hold. Take G^2 (cell 2) for example (see Figure C.13(a) and Figure C.16), we have

$$C^2 \times Y_{2,\kappa} = 0$$
$$Y_{2,\kappa}[t_{37}] = Z_4[t_{37}]z_{4,\kappa}$$
$$Y_{2,\kappa}[t_6] = Z_4[t_6]z_{4,\kappa}$$
$$Y_{2,\kappa}[t_7]\theta_{t_7} + Y_{2,\kappa}[t_9]\theta_{t_9} \leq \bar{H}$$
$$Y_{2,\kappa}[t_8]\theta_{t_8} + Y_{2,\kappa}[t_{10}]\theta_{t_{10}} \leq \bar{H}$$

where C^2 is the incidence matrix of G^2.

However, using Algorithm C.2, we can get a very simple formultion of system capacity. First , by solving

$$Min\ W$$

subject to:

$$C^2 \times y = 0$$
$$y[t_{37}] = Z_4[t_{37}]$$
$$y[t_6] = Z_4[t_6]$$
$$y[t_7]\theta_{t_7} + y[t_9]\theta_{t_9} \leq W$$
$$y[t_8]\theta_{t_8} + y[t_{10}]\theta_{t_{10}} \leq W$$

we get an optimal flow: $y_4 = (1 \times t_6, 1 \times t_7, 1 \times t_9, 1 \times t_{37})$, thus we have

$$\Theta_{4,r_5} = y_4[t_7]\theta_{t_7} + y_4[t_9]\theta_{t_9} = 1 \times 3 + 1 \times 4 = 7$$

therefore the capacity constraint corresponding to G^2 is simply:

$$7 \times z_4 \leq \bar{H}$$

The above reasoning applies also to G^1 and G^3. Moreover according to Theorem C.1, we know that in doing so we get a pessimistic yet simpler approximation of system capacity.

Our computational experience shows that the capacity evaluation algorithm proposed above (Algorithm C.2) gives an approximation of about 4% of the real capacity. On the other hand, using Algorithm C.2 reduces, on the average, 75% of computation time. This, on the whole, shows that though Algorithm C.2 provides a pessimistic approximation of real capacity, it is well acceptable due to the reduction of computation it brings about. Moreover a pessimistic approximation of system capacity guarantees the feasibility of the high level plan in the sense that such a plan is implementable at the low level.

C.9.2 Solving a concrete example

In this section, we present a complete solution to the management problem of the system described in Example C.1.

First we specify the relevant parameters:

- Number of sub-periods: $\bar{K} = 2$,
- Number of elementary periods in a sub-period: $K = 2$,
- Length of an elementary period: $H = 50$.

Table C.3 gives information about product demands.

	sub-period $\kappa = 1$		sub-period $\kappa = 2$	
	$k = 1$	$k = 2$	$k = 3$	$k = 4$
Product 1	4	2	4	4
Product 2	2	3	3	5
Product 3	3	4	6	3

TAB. C.3 - *Demands information*

The initial inventory information is given in Table C.4.

Table C.5 presents data about inventory costs and backlogging costs.

	Initial inventory
Product 1	1
Product 2	1
Product 3	1

TAB. C.4 – *Initial inventory*

	Inventory cost	Backlogging cost
Product 1	10	50
Product 2	20	40
Product 3	30	30

TAB. C.5 – *Inventory costs and backlogging costs*

	Initial inventory	Inventory cost
Interface place P_7	6	5
Interface place P_8	6	10
Interface place P_{14}	5	15

TAB. C.6 – *Interface information*

Cell	Transition	# of firing
1	t_{31}	6
1	t_{35}	7
1	t_{36}	7
2	t_{37}	7
3	t_{16}	6
3	t_{25}	4

TAB. C.7 – *Simplified whole system planning results*

We also need information about initial inventory in interface places and initial cost in interface places. This is given in Table C.6.

The results of the simplified whole system planning in the first elementary period is given in Table C.7 (cf. Figure C.16).

The results for the planning of cells 1, 2, and 3 are respectively shown in Tables C.8, C.9, and C.10 (cf. Figures C.13 and C.12).

t_1	t_3	t_4	t_5	t_{18}	t_{19}	t_{20}	t_{27}	t_{29}
7	7	7	7	0	7	0	6	6
t_{30}	t_{35}	t_{36}	t_0	t_2	t_{17}	t_{26}	t_{28}	t_{31}
6	7	7	7	7	7	6	6	6

TAB. C.8 - *Plan for cell 1*

t_7	t_8	t_9	t_{10}	t_{37}	t_6
7	0	0	7	7	7

TAB. C.9 - *Plan for cell 2*

t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{21}	t_{22}
6	6	6	6	0	0	4
t_{23}	t_{24}	t_{32}	t_{33}	t_{34}	t_{16}	t_{25}
4	0	6	6	4	6	4

TAB. C.10 - *Plan for cell 3*

Scheduling cell 1 (cf. Figure C.12) gives the following results which are presented respectively in Table C.11 (by means of SA) and Table C.12 (by means of BAB). The starting times for each firing of the transitions associated with each machine are given. The number in the parenthesis after each transition stands for the time needed for each firing of the transition.

The scheduling of cell 2 (cf. Figure C.13) becomes trivial since only one transition is associated with each of the two machines M_5 and M_6 in cell 2: t_7 with M_5, and t_{10} with M_6. This fact comes from the cell 2 planning (see Table C.9). This phenomenon has an interesting meaning: the potential structural conflict between transitions t_7 and t_8 and that between transitions t_9 and t_{10} are resolved at the cell planning level, which alleviates, to some extent, the burden of scheduling level.

When scheduling cell 3 (cf. Figure C.13), we are warned that the elementary period is not long enough to carry out a schedule that satisfies the planned requirement for the firing of each transition in cell 3 and that at the same time keeps

the capacity of the cell within its limit. In other words, the plan for cell 3 is not feasible. This gives us a message that we need to take some actions so that the scheduling of cell 3 can be done. Either we must enlarge the capacity of the cell (e.g., by adding more machines) or we must re-start the planning by considering a shorter planning period such that less tasks are generated for the cells and thus less burden is put on the cell scheduling level.

Machine	Transition	Starting times
M_1	$t_1(3)$:	0 4 8 11 14 18 21
	$t_{27}(1)$:	3 7 17 24 25 26
M_7	$t_3(1)$:	0 1 4 5 6 11 14
	$t_{19}(2)$:	19 21 23 25 27 29 31
	$t_{29}(2)$:	2 7 9 12 15 17
M_8	$t_4(2)$:	3 8 13 15 17 22 24
	$t_{30}(3)$:	5 10 19 26 29 32
M_4	$t_5(5)$:	5 10 15 20 25 30 35

TAB. C.11 – *Schedule for cell 1 (by means of SA)*

Machine	Transition	Starting times
M_1	$t_1(3)$:	0 3 6 9 12 15 21
	$t_{27}(1)$:	18 19 20 24 25 26
M_7	$t_3(1)$:	0 1 2 5 8 9 10
	$t_{19}(2)$:	3 6 11 13 15 19 23
	$t_{29}(2)$:	17 21 25 27 29 31
M_8	$t_4(2)$:	3 6 9 12 15 18 26
	$t_{30}(3)$:	20 23 28 31 34 37
M_4	$t_5(5)$:	5 10 15 20 25 30 35

TAB. C.12 – *Schedule for cell 1 (by means of BAB)*

Annexe D

Package HMPS (*Hierarchical and Modular approach to production Planning and Scheduling*)

D.1 Introduction to language Tcl/Tk

In this section, we give a very brief introduction to the language Tcl/Tk, and explain why we choose Tcl/Tk for our GUI development.

Tcl stands for *Tool Command Language*. It is a scripting language[1] and an interpreter for that language. Tcl and its associated X window toolkit, Tk, were designed and crafted by John Ousterhout of University of California Berkeley [98].

As a scripting language, Tcl is similar to other UNIX shell languages (e.g., Bourne Shell, C Shell, Korn Shell etc.). But it is the ability to easily add a Tcl

[1] There is no universally accepted definition of what a scripting language is. However it typically has certain features:

- Interpreted execution, so a compile-link cycle is not required.
- Simple syntax. For example statements are usually terminated by newlines, rather than by semicolons or other punctuation.
- Untyped variables that act as strings or numbers depending on the operations that is being performed on them.
- Variables created when referenced, rather than through explicit declarations.
- High-level string manipulation features such as concatenation, substring operations, and searching/sorting primitives built into the language.
- No pointers or memory allocation. Garbage collection of unused memory is done automatically.

Another useful distinction is that scripting languages are not the same as macro languages which typically lack the control flow and procedure-building capabilities that are required in a general-purpose programming language.

interpreter to the application that sets it apart from other shells.

There are many Tcl extensions available. Tk is one of the most notable extensions. Tk has been designed for the X window system, although ports to other window systems are expected to appear eventually. Tk provides a set of Tcl commands that create and manipulate *widgets*. A widget is a window in a graphical user interface that has a particular appearance and behaviour. The term *widget* and *window* are often used interchangeably.

Although there exist several other interactive graphical user interface development tools, such as the Desktop Kornshell and MetaCard, the reasons that we choose Tcl/Tk for our GUI development come mainly from the following facts:

1. Since Tcl/Tk is a scripting language,
   - there is a rapid turnaround, i.e., there is no waiting for long compilations,
   - the Tcl commands provide a higher-level interface to X than most standard C library toolkits,
   - the graphical user interface is clearly factored out from the rest of the application.
2. The large number of Tcl/Tk users is a strong indicator that the use of these tools will continue to increase and their quality will continue to improve.
3. The very fact that Tcl/Tk is free also makes it an attractive package for us[2].

D.2 The package HMPS

In this section, we give an overview of the architecture of the prototype package HMPS developed at project SAGEP of INRIA-Lorraine[3]. Emphasis is put on the graphical user interface, i.e., human-machine interaction.

[2] Tcl/Tk is evolving quite rapidly. The distributions are available by anonymous FTP from *ftp.cs.berkeley.edu* in */tcl/*. They are also available from *harbor.ecn.purdue.edu* in */pub/tcl/sprite-mirror/*

[3] More details are reported in:

> J.-M. Beauville, R.Cabirol, F. Chu, J. Grandmougin, J.-M. Proth, N. Sauer, L.M. Wang, **SIREP - Rapport final, Spécification et Intégration à l'aide des Réseaux de Petri des fonctions d'un système de production**, *Compte rendu de fin d'opération d'une recherche financée par le Ministère de la Recherche et de l'Espace*, Septembre, 1994.

D.2.1 HMPS versus MASP

HMPS has been developed on the basis of the prototype MASP [115]. MASP has three parts: human-machine interface, model definition, model evaluation.

However the structure and contents of HMPS are different from those of MASP. HMPS has enriched all the three parts of MASP. Moreover HMPS has developed new parts which are indispensable to the implementation of the modular and hierarchical management scheme proposed previously. Model definition is enriched to take into account the fact that the whole system is composed of interconnected subsystems. Thus a new part of defining subsystems (modules, or cells) is added. Furthermore, simplification of the modules requires that another new part of cell simplification be added.

As for the evaluation of the system performance, more functions are added to meet the requirements of hierarchical management. MASP was implemented under the assumption that the system is treated as an entire entity, no concept of subsystem is involved. Since HMPS was developed under a different management scheme, i.e., hierarchical management of modularly modelled systems, from the one that governed the implementation of MASP, HMPS requires that production planning be carried out in at least two layers.

Thus two new parts have been built in HMPS: the simplified whole system planning and the cell planning. Moreover another new part that implements the connection between the two layers has also been constructed.

Finally, the graphical user interface has been re-designed and enriched according to the requirements of the modular modelling and hierarchical management framework proposed in this thesis.

D.2.2 Architecture of HMPS

Figure D.1 exhibits the title screen of the prototype package. Please note that all the widgets appearing in this section are provided by the *twm* window manager. If a different window manager is used, the decorations may be different.

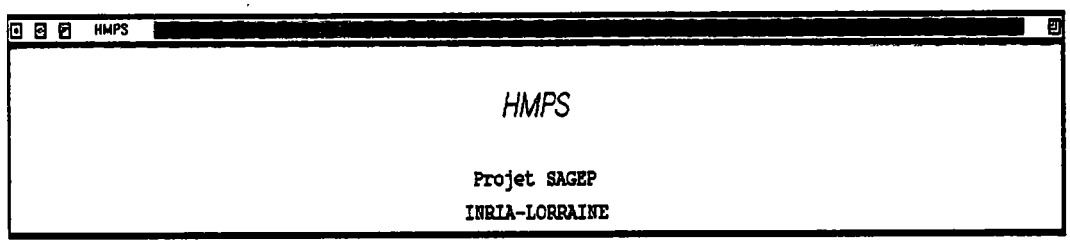


FIG. D.1 – *Title of the package*

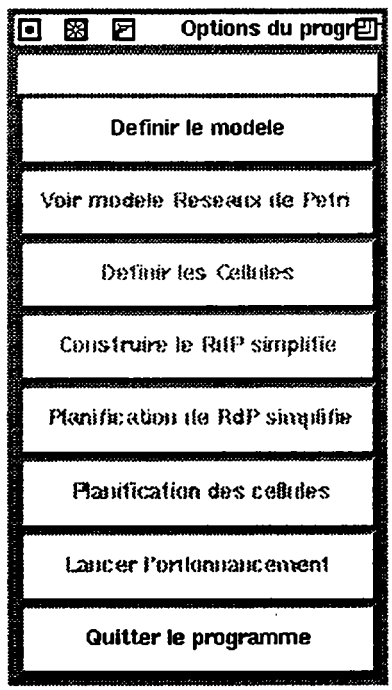
The main menu as shown in figure D.2 specifies the first functional hierarchy of the package. At this hierarchy, there are only two buttons that are active, i.e., actions can be taken on these buttons while nothing can be done on the buttons

that are not active. Thus, at this hierarchy, we can only do two things: either *define a model* or *quit the program* as shown in the figure.

From a user's point of view, the structure of the prototype consists of the following steps:

- Defining a model
- Building an *executive product base*
- Defining cells
- Specifying demands and getting results for planning
- Scheduling the cells

In the rest of the section, we introduce these steps one by one.

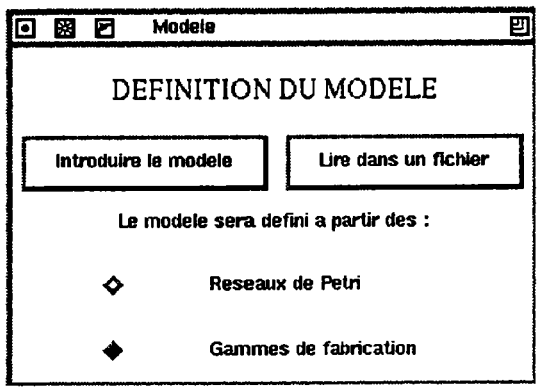


Bright-font buttons: active;
Dim-font buttons: inactive.

FIG. D.2 – *The main menu*

D.2.3 Defining a model

Defining a model means that a computer recognised model of some manufacturing processes is to be built. In HMPS, two ways of defining a model are given, as shown in figure D.3: treating the input manufacturing processes as Petri nets, and treating them as BOM (Bill of Material). Also the manufacturing processes can



A computer recognised model can be provided by either inputting with a keyboard or reading from an existing file that contains information of the model.

FIG. D.3 – *Defining a model*

be input with a keyboard or via reading an existing file that contains information of the model.

To the best of our knowledge, it is a common practice to define a model with a keyboard and by treating the manufacturing processes as a BOM. In this case, the model is defined by the following quantities:

- number of products;
- number of machines;
- number of operations involved for each product in the BOM;
- the *characteristic*[4] of each operation involved;

Figure D.4 gives a snapshot of how to input a model by specifying the above mentioned quantities.

After the above mentioned quantities are input for all the products, there remains an important thing to do: to specify the precedence constraints between the operations. Figure D.5 shows how to do it in HMPS. It is worth noting that if there are more operations preceding the present operation, then the present operation either *is* an assembly operation or *is not* depending on the precedence constraint given in the BOM. For the snapshot in figure D.5, the present operation is the one labelled 2, and there are three operations preceding it. These three preceding operations provide semi-products *asynchronously* to the present one. The information about stocks are to be provided.

The process of building a model ends at the point that all the manufacturing processes are input by the way just shown, and the model is saved as a file that contains all the information about the BOM of the products to be manufactured. Such a file will later be called a *product base.*

[4] By characteristic of an operation, we mean whether the operation is a *simple* operation that requires only one machine, or it is a *complex* operation that requires several machines. If it requires several machines, these machines may be used in serial or in parallel.

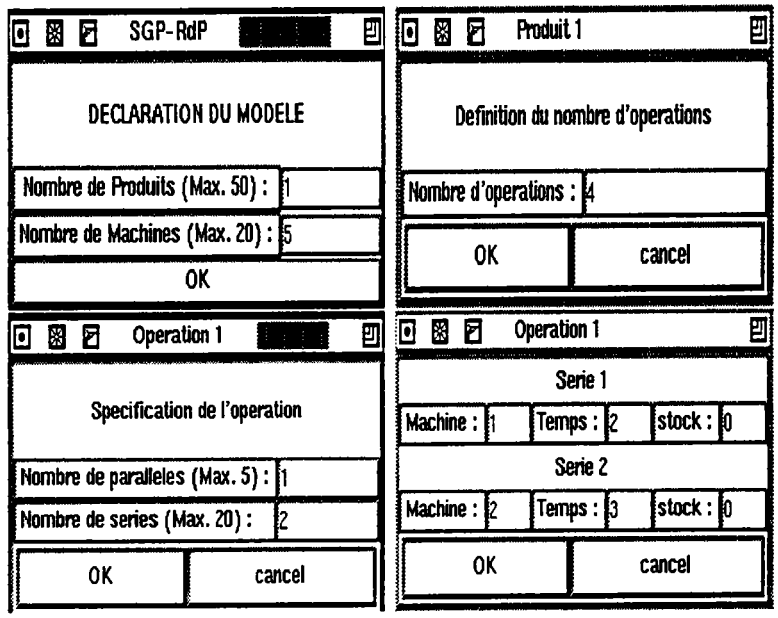


FIG. D.4 – *Input a model as BOM*

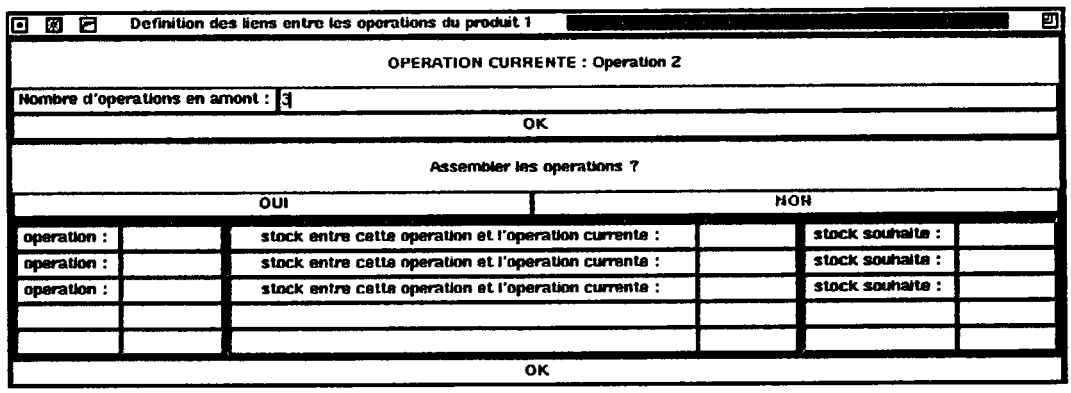
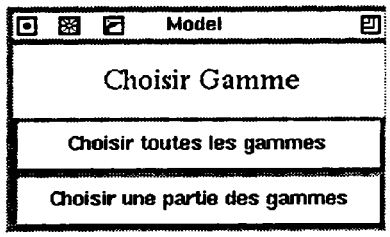


FIG. D.5 – *Precedence constraint specification*

D.2.4 Building an executive product base

Once a product base is built, the information about all the products that can be manufactured by a manufacturing system in a long period of time is known. However in a short period of time, not all the products need to be manufactured. Thus it is required that one choose from the product base a certain number of products that are to be manufactured in a short period of time. This is exactly the objective of building an executive product base.

In figure D.6, it is shown that if a file is given in which contains the information about the manufacturing processes, then depending on the *nature*[5] of the file, we can choose the whole file to execute, or we can choose some of the processes in the file to proceed.



Two possibilities: if the file itself is an executive product base file, then choose all; if the file is a product base file, then choose part of it.

FIG. D.6 – *Choosing an executive product base from a product base file*

If we choose part of the file, then we have to specify:

- how many manufacturing processes are to be chosen;
- which manufacturing processes are to be chosen.

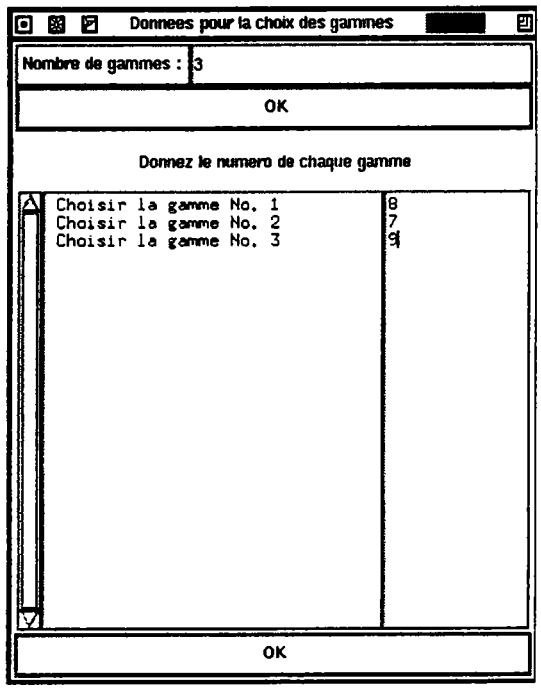
This situation is shown in figure D.7 in which three manufacturing processes are to be chosen, and they are the processes labelled 8,7,9.

D.2.5 Defining cells

It is up to the users to *define* cells. By defining cells, we mean the specification of how many machine groups are to be clustered, and in each machine group, how many and which machines are to be included. These machine groups are called cells. Figure D.8 gives a snapshot in which it is shown that the user decides to groups the machines into three cells, and in the first cell, there are four machines, they are machines 1,7,8,4.

The information about cell definition can be saved in a file, as does the information about the product base or executive product base. After cell definition is input and saved, the main menu (see figure D.2) looks as shown in figure D.9.

[5] Here, by "the nature of a file", we mean whether it is a product base file or an executive product base file.



Three processes are to be chosen, they are processes labelled 8,7,9.

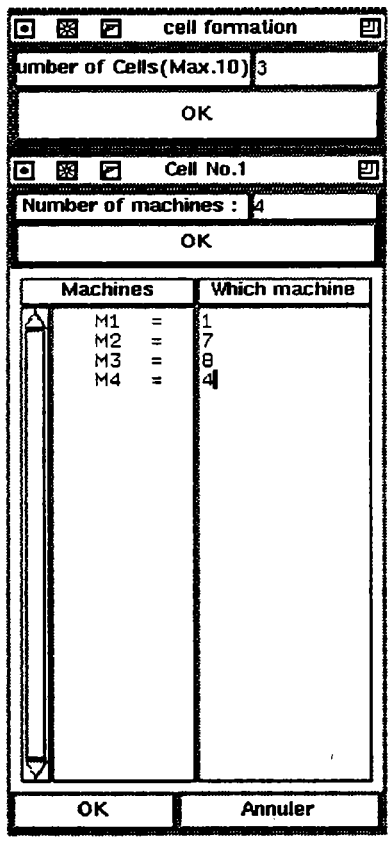
FIG. D.7 – *How many and which processes to choose*

Now we are ready to construct a simplified Petri net model of the manufacturing processes in the executive product base chosen previously. This is done by clicking on the button "Construire le RdP simplifie" (see figure D.9), which leads to the activation of the button "Planification de RdP simplifie".

D.2.6 Specifying demands and getting results for planning

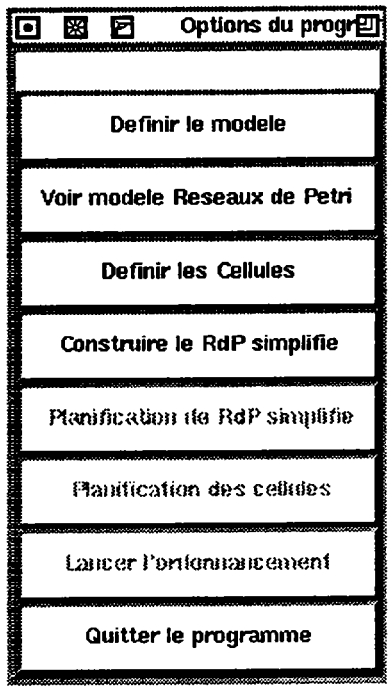
Planning is concerned with two aspects: planning the simplified system, and planning the cells. The simplified system described by a simplified Petri net is done first. Before doing this, necessary parameters have to be specified and input. These parameters are:

- objective function: we confine ourselves to the minimisation of the stock costs and backlogging costs, in both simplified system planning and cell planning;
- number of elementary periods;
- duration of each elementary period;
- quantity of demand for each product in each elementary period;
- initial stock of each product at the beginning of first period;
- unit cost of stock of each product;



There are 3 cells. In cell 1, there are 4 machines: machines 1,7,8,4.

FIG. D.8 – *Cell definition*



After cell definition, more buttons are becoming active, cf. figure D.2

FIG. D.9 – *New look of main menu after cell definition*

- unit cost of backlogging of each product.

For planning of the simplified system, we also need the information about the interface between the cells. This is specified by:

- initial stock of products in the interface places;

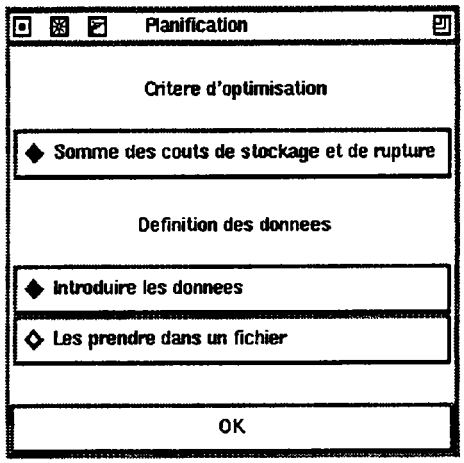
- unit cost of stock for products in the interface places.

Figure D.10 shows that the objective function we are interested in is the sum of stocking costs and backlogging costs.

Shown in figure D.11 are ways of inputting some other parameters. Note that the parameter *real duration of each elementary period* (not longer than the presently used length of period) is specified in case that if the plan under the present duration of elementary period is not feasible, we can reduce the length of period and see if the plan under the reduced duration of period is feasible. If the present length of period is reduced to the real duration, and the plan is still infeasible, then we say that we can not find a feasible plan under the real length of elementary period. Otherwise we can always find a feasible plan under the real duration of elementary period.

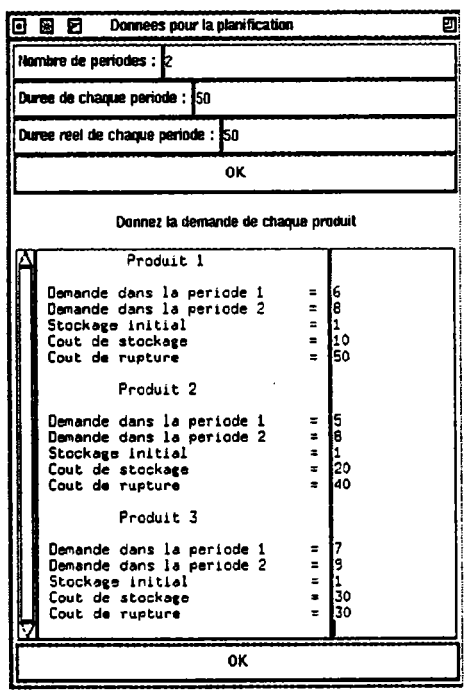
Ways of inputting the interface information is shown in figure D.12.

The results of planning the simplified system and of planning cells are shown in figure D.13. The result of planning the simplified system gives the specification



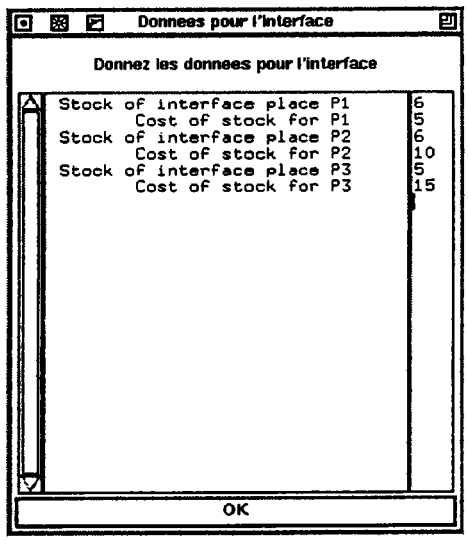
Planning data can be input via keyboard, or via reading a file, if it exists, which contains the data.

FIG. D.10 – *Objective function*



2 elementary periods;
length of each period is 50;
real duration of each elementary period (as a reference to the presently used duration) is 50;

FIG. D.11 – *Planning parameters*



Initial stock of product in interface places and unit cost of product in interface places are specified.

FIG. D.12 – *Interface information*

of the number of firing times for each output transition of all the cells during the given period of planning time. And the outcome of planning each cell specifies the number of firing times of every transition in the cell during the planning period.

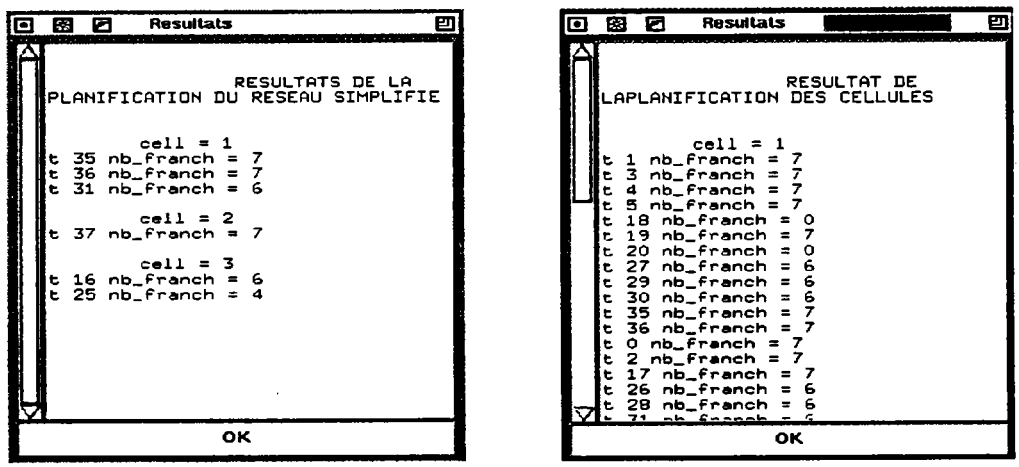
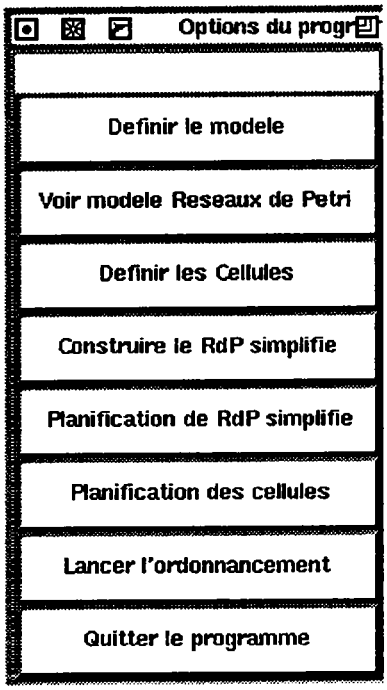


FIG. D.13 – *Results for simplified system planning and cells planning*

After the planning, only one button "Lancer l'ordonnancement" is left to-be-activated in the main menu (see figure D.14). The activation of this button leads to scheduling of the cells.

D.2.7 Scheduling the cells

Although planning a cell (as Petri net model) gives the required number of firing times of each transition in the cell so that the cost of stock and the stock

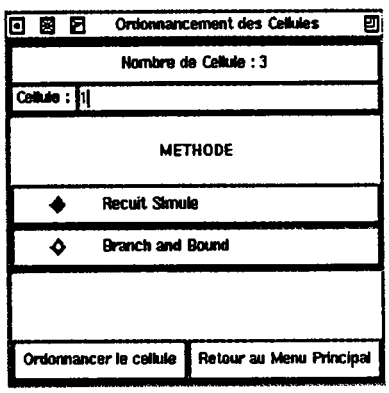


After planning, only one button "Lancer l'ordonnancement" is left to-be-activated, cf. figures D.2 and D.9

FIG. D.14 – *New look of main menu after planning*

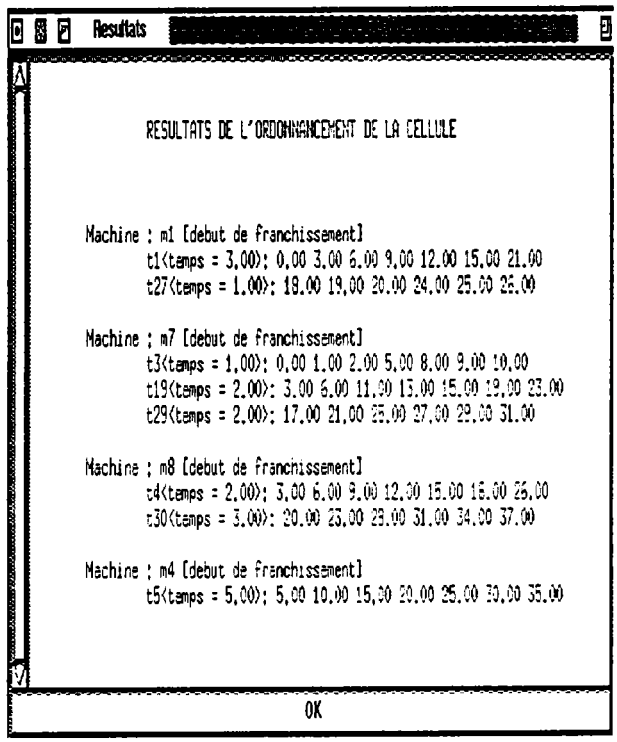
of backlogging are minimised, nothing is said about at which time the required number of firings of each transition should occur. This is a task to be undertaken by the scheduling procedures.

In HMPS, we have taken advantage of two methods for the task of scheduling: simulated annealing and branch-and-bound. This is shown in figure D.15, in which the two radio buttons allows to choose either of the two methods, and one can also choose which cell to work on. No details are meant to be presented here, interested readers may find useful these references: [26] [29] [79] [117] [118].



In HMPS, two methods are provided for scheduling: SA and BAB.

FIG. D.15 – *Scheduling menu*



Scheduling a cell gives the starting instant of each firing of the required number of firings for each transition in the cell.

FIG. D.16 – *Scheduling result*

D.3 Conclusion

A prototype package HMPS is presented. It is designed on the basis of modular modelling and hierarchical management of manufacturing systems. The modular modelling work is done in the framework of Petri net theory. We proposed a new type of Petri net, CO net, to specifically tailor the requirement of modular modelling and system integration. We also proposed an efficient algorithm for simplifying a given Petri net model in the sense that the net structure is simplified while the input-output flow relation of the original Petri net is preserved. The algorithm applies not only to CO nets but also to general Petri nets. The use of the package is facilitated by a graphical user interface designed in language Tcl/Tk.

Although the preliminary use of the prototype package shows that it is capable of solving some problems quite satisfactorily, there still remains a long way to go to get it competitive with existing commercial packages, like PROVISA (AT&T), WITNESS (AT&T), AutoSched (AutoSimulations), QUEST (Deneb Robotics Inc.), MPX (Network Dynamics Inc.), FACTOR (Pritsker Corp.), ManSim/X (TYECIN Systems Inc.), to name a few.

Some perspectives we envisage are as follows:

- There is room to the improvement of the graphical user interface. As seen

from the previous introduction to HMPS, the whole GUI is principally based on a *button activation* scheme, which has been shown to be clumsy. New versions of Tcl/Tk have provided enhancement on *pull-down* menus, which enables production of more friendly, more agreeable human-machine interface.

- Present ways of inputting product base lack flexibility and robustness. New options, like stochastic processing times and new BOM structures, are difficult to get incorporated into the present package because of the rigid structure of the package. More fault-tolerant mechanisms need to be incorporated so that error messages not only are error indicators but also are informative in finding out where the errors are. To sum up briefly, a critical re-design from the point of view of software engineering needs to be examined.

- As for the method of planning adopted in HMPS, namely an open-loop top-down planning practice, plan feasibility problem is a prominent issue. Although there are some discussions about this issue in the literature, see e.g., [36], [76], [34] [39], a simple closed-loop practice does not constitute the only way out, theoretically sound and practically efficient approaches are always in urgent need.

- Though scheduling problems are generally quite notorious [11] [32] [41], and simulated annealing and branch-and-bound methods are two frequently used methods to deal with them, new attacks are always attempted as can be seen through the enormous articles on this issue. It might be fruitful to incorporate various approaches to HMPS and adapt them to different practical situations.

- As pointed out previously, a new look at the package HMPS from the software engineering point of view is certainly necessary if the aim of HMPS is to be competitive with existing commercial packages. Various tricks in dealing with the efficient use of memory space and reducing time consumption are, however, also indispensable to get it *compact* enough to solve larger practical problems.

Bibliographie

[1] K. Aardal and T. Larsson. A benders decomposition based heuristic for the hierarchical production planning problem. *European Journal of Operational Research*, 45:4–14, 1990.

[2] T. Agerwala and Y. Choed-Amphai. A synthesis rule for concurrent systems. In *Proc. 15th Design Automation Conference*, pages 305–331, Las Vegas, June 1978.

[3] R.Y. Al-Jaar and A.A. Desrochers. Petri nets in automation and manufacturing. In *Advances in Automation and Robotics*, volume 2. JAI Press, 1989.

[4] R.N. Anthony. *Planning and control systems: a framework for analysis.* Harvard University Press, Graduate School of Business Administration, Cambridge, Massachussettes, 1965.

[5] Ch. Ausfelder, E. Castelain, and J.-C. Gentina. A method for hierarchical modelling of the command of flexible manufacturing systems. *IEEE Trans. Systems, Man and Cybernetics*, 24(4):564–573, 1994.

[6] S. Axsater. On the design of the aggregate model in a hierarchical production planning system. *Engineering and Process Economics*, 4:89–97, 1979.

[7] S. Axsater. Aggregation of product data for hierarchical production planning. *Operations Research*, 29(4):744–756, 1981.

[8] S. Axsater. On the feasibility of aggregate production plans. *Operations Research*, 24(5):796–800, 1986.

[9] S. Axsater and H. Jonson. Aggregation and disaggregatiuon in hierarchical production planning. *European Journal of Operational Research*, 17:338–350, 1984.

[10] S. Axsater, H. Jonsson, and A. Thorstenson. Approximate aggregation of product data. *Engineering Cost and Production Economics*, 7:119–126, 1983.

[11] K.R. Baker. *Introduction to sequencing and scheduling.* Wiley, New York, 1974.

[12] J. Benassy. *La gestion de production.* Hermes, 1987.

[13] A. Bensoussan, M. Crouchy, and J.M. Proth. *Mathematical Theory of Production Planning.* Elsevier Science Publishers B.V., 1983.

[14] G. Berthlot. Checking properties of nets using transformation. In G. Rozenberg, editor, *Advances in Petri Nets 1985*, Lecture Notes in Computer Science, pages 19–40. Springer-Verlag, 1985.

[15] G.M. Birtwistle. *DEMOS: A system for discrete event modelling on SIMULA.* MacMillan, 1979.

[16] G.R. Bitran, E.A. Haas, and A.C. Hax. Hierarchical production planning: a single stage system. *Operations Research*, 29(4):717–743, 1981.

[17] G.R. Bitran, E.A. Haas, and A.C. Hax. Hierarchical production planning: a two stage system. *Operations research*, 30(2):232–251, 1982.

[18] G.R. Bitran, E.A. Hass, and H. Matsuo. Production planning of style goods with setup costs and forecast revisions. *Operations Research*, 32(2):226–236, 1986.

[19] G.R. Bitran and A.C. Hax. On the design of hierarchical production planning systems. *Decision Sciences*, 8:28–55, 1977.

[20] G.R. Bitran and A.C. Hax. Disaggregation and resource allocation using convex knapsack problems with bounded variables. *Management Science*, 27(4):431–441, 1981.

[21] G.R. Bitran and D. Tirupati. Hierarchical production planning. Technical Report 3017-89-MS, MIT, Sloan School of Management, Cambridge, 1989.

[22] M.R. Bowers and J.P. Jarvis. A hierarchical production planning and scheduling model. *Decision Sciences*, 23(1):144–159, 1992.

[23] G.W. Brams. *Reseaux de Petri: Theorie et Pratique.* MASSON, 1983.

[24] J.A. Buzacott and J.G. Shanthikumar. Design of manufacturing systems using queueing models. *Queueing Systems*, 12:135–214, 1992.

[25] J.A. Buzacott and J.G. Shanthikumar. *Stochastic models of manufacturing systems.* Prentice-Hall, Englewood Cliffs, NJ, 1992.

[26] J. Carlier and P. Chretienne. Timed petri net schedules. In G. Rozenberg, editor, *Advances in Petri nets*, volume 340 of *Lecture Notes in Computer Science*, pages 62–84. Springer-Verlag, 1988.

[27] C. G. Cassandras and P. J. Ramadge. Toward a control theory for discrete event systems. *IEEE Control Systems Maganizine*, pages 66–111, June 1990.

[28] C.B. Chu, F. Chu, J.-M. Proth, and X.L. Xie. Verification incrementale de la consistance d'un reseaux de petri. to appear in Technique et Science Informatique, 1995.

[29] F. Chu, J. M. Proth, and V.M. Savi. Ordonnancement basé sur les réseaux de petri. Technical Report 1960, INRIA, 1993.

[30] G. Cohen, D. Dubois, J.P. Quadrat, and M. Viot. A linear system theoretic view of discrete event processes and its use for performance evaluation in manufacturing. *IEEE Trans. Automatic Control*, 30(3):210–220, March 1985.

[31] G. Cohen, P. Moller, J.P. Quadrat, and M. Voit. Algebraic tools for the performance evaluation of discrete event systems. In *Proceedings of the IEEE* [61], pages 39–58.

[32] R.W. Conway, W.L. Maxwell, and L.W. Miller. *Theory of scheduling.* Addison Wesley, Massachussetts, 1967.

[33] R.A. Cuninghame-Green. *Minimax Algebra*, volume 166 of *Lecture Notes in Economic and Mathematical Systems.* Springer-Verlag, 1979.

[34] S. Dauzere-Peres. *Planification et ordonnancement de la production: une approche integree coherente.* PhD thesis, Universite Paul Sabastier de Toulouse, March 1992.

[35] R. David and H. Alla. *Du grafcet aux réseaux de Petri.* Edition Hermès, 2e édition, Paris, 1992.

[36] M.A.H. Dempster, M.L. Fisher, B.J. Lagweg, L. Jansen, J.K. Lenstra, and A.H.G. Rinnooy Kan. Analytical evaluation fo hierarchical planning systems. *Operations Research*, 29(4):707–716, 1981.

[37] F. DiCesare, G. Harhalakis, J.M. Proth, M. Silva, and F. Vernadat. *Practice of Petri nets in manufacturing.* Chapman and Hall, 1993.

[38] S.E. Elmaghraby. The economic lot scheduling problem: review and extension. *Management Science*, 24(6):587–598, February 1978.

[39] J. Erscheler, G. Fontan, and C. Merce. Consistency of the disaggregation process in hierarchical planning. *Operations Research*, 34(3):464–469, 1986.

[40] C.A. Fernandes and F.A.C. Gomide. A heuristic procedure for production planning of modern manufacturing systems. *Computers and Industrial Engineering*, 20(4):475–485, 1991.

[41] S. French. *Sequencing and scheduling.* John Wiley and Sons, 1982.

[42] L.F. Gelders and L.N. van Wassenhove. Production planning: a review. *European Journal of Operational Research*, 7:101–110, 1981.

[43] L.F. Gelders and L.N. van Wassenhove. Hierarchical integration in production planning: theory and practice. *Journal of Operations Management*, 3(1):27–35, 1982.

[44] S.B. Gershwin. Hierarchical flow control: a framework for scheduling and planning discrete events in manufacturing systems. In *IEEE Proceedings, Special Issue on DES.* [61].

[45] V. Giard. *Gestion de la production.* Economica, 2nd edition, 1988.

[46] P. Glasserman and D.D. Yao. Generalized semi-markov processes: antimatroid structure and second-order properties. *Mathematics of Operations Research*, 17(2):444–469, May 1992.

[47] P.W. Glynn. A gsmp formalism for discrete event systems. In *Proceedings of IEEE* [61], pages 14–23.

[48] M. Gondran and M. Minoux. *Graphes et algorithmes.* Eyrolles, Paris, 1979.

[49] G. Gordon. *The application of GPSS V to discrete system simulation.* Prentice-Hall, Englewood Cliffs, NJ, 1975.

[50] S.C. Graves. Unsing langrangean techniques to solve hierarchical production planning problems. *Management Sciences*, 28(3), 1982.

[51] H.M. Hanisch. Analysis of timed petri nets by means of dynamic graphs. *Petri Net Newsletter*, 36:24–31, 1990.

[52] H.M. Hanisch. Analysis of timed place/transition nets using minimal state graphs. In G. Cohen and J.P. Quadrat, editors, *11th International Conference on Analysis and Optimisation of Systems*, volume 199 of *Lecture Notes in Control and Information Sciences*, pages 205–212. Springer-Verlag, June 1994.

[53] G. Harhalakis and J.M. Proth. Some open problems in the design and use of modern production systems. *Applied Stochastic Models and Data Analysis*, 7:33–45, 1991.

[54] G. Harhalakis, J.M. Proth, and X.L. Xie. A comprehensive approach of planning and scheduling based on petri nets. *Applied Stochastic Models and Data Analysis*, 9(3):187–213, 1993.

[55] C.J. Harris. Out of control into systems engineering research. In *11th Computing and Control Lecture*, Savoy Place, London, February 1994. IEE.

[56] I. Hatano, K. Yamagata, and H. Tamura. Modelling and on-line scheduling of flexible manufacturing systems using stochastic petri nets. *IEEE Trans. Software Engineering*, 17(2):126–132, February 1991.

[57] A. C. Hax and H.C. Meal. Hierarchical integration of production planning and scheduling. In M.A. Geisler, editor, *Studies in Management Sciences*, volume I, pages 53–69. North Holland - American Elsevier, 1975.

[58] J.W. Hermann, A. Mehra, I. Minis, and J.M. Proth. Hierarchical production planning with part, spatial and time aggregation. In *Proc. 4th Int. Conf. on CIM and Automation Technology*, pages 430–435, Reselaer, New York, October 1994.

[59] H. P. Hillion and J. M. Proth. Performance evaluation of job-shop systems using timed event-graphs. *IEEE Trans. Automatic Control*, 34(1):3–9, 1989.

[60] M.H. Hillion. *Modelisation et analyse des systemes de productoin discrets par les reseaux de Petri temporises.* PhD thesis, Universite Pierre et Marie Curie, Paris VI, January 1989.

[61] Y.C. Ho, editor. *Special issue on dynamics of discerete event systems*, volume 77 of *IEEE Proceedings*, 1989.

[62] Y.C. Ho and X.R. Cao. *Pertuabation analysis of discrete event dynamic systems.* Kluwer Academic Publishers, 1991.

[63] C.A.R. Hoare. *Communicating Sequential Processes.* International Series in Computer Science. Prentice-Hall, 1985.

[64] B.W. Hollocs. Improving manufacturing operations with witness computer simulation. *AT&T Technology*, 6(1):16–21, 1991.

[65] L. E. Holloway and B. H. Krogh. Synthesis of feedback control logic for a class of controlled petri nets. *IEEE Trans. Automatic Control*, 35(5):514–523, 1990.

[66] K. Inan and P. Varaiya. Finitely recursive process models for discrete event systems. *IEEE Trans. on Automatic Control*, 33(7):626–639, July 1988.

[67] R.R. Inman and P.C. Jones. Decomposition for scheduling flexible manufacturing systems. *Operations Research*, 41(3):608–617, 1993.

[68] K. Jensen. *Coloured Petri nets: Basic concepts, analysis methods and practical use, Vol.1.* EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1992.

[69] U.S. Karmarkar and L. Schrage. The deterministic dynamic product cyclingproblem. *Operations Research*, 33(2), 1985.

[70] J. Kimemia and S. B. Gershwin. An algorithm for the computer control of a flexible manufacturing system. *IIE Transations*, 15(4):353–362, 1983.

[71] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[72] I. Koh and F. DiCesare. Modular transformation methods using deviation bounds in automated manufacturing systems. *IEEE Trans. Systems, Man and Cybernetics*, 21(6):1512–1522, 1991.

[73] A. B Kurzhanski and P. Varaiya, editors. *Discrete event systems: Models and applications*, volume 106 of *Lecture Notes on Information Sciences*, 1988.

[74] S. Laftit, J.M. Proth, and X.L. Xie. Optimisation of invariant criteria for event graphs. *IEEE Trans. on Automatic Control*, 37(5):547–555, 1992.

[75] L.S. Lasdon and R.c. Terjung. An efficient algorithm for multi-item scheduling. *Operations Research*, 19, 1971.

[76] J.B. Lasserre. An integrated model for job-shop planning and scheduling. *Management Science*, 38(8):1201–1211, August 1992.

[77] J.B. Lasserre, J.P. Martin, and F. Roubellat. Aggregate model and decomposition method for mid-term production planning. *International Journale of Production Research*, 21(6):835–843, 1983.

[78] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnoy Kan, and D.B. Shimoys. Sequencing and scheduling: algorithms and complexity. Technical Report NF1 11.89/03, Eindohoven University of Technology, November 1989.

[79] D.Y. Lee and F. DiCesare. Fms scheduling using petri nets and heuristic search. In *Proc. 1992 IEEE Int. Conf. on Robotics and Automation*, pages 1057–1062, Nice, France, May 1992.

[80] K.H. Lee and J. Favrel. Hierarchical reduction method for analysis and decomposition of petri nets. *IEEE Trans. Ssystems, Man, and Cybernetics*, 15(2):272–281, 1985.

[81] K.H. Lee, J. Favrel, and P. Baptiste. Generalized petri net reduction method. *IEEE Trans. on Systems, Man, and Cybernetics*, 17(2):297–303, 1987.

[82] G.K. Leong, M.D. Oliff, and R.E. Markland. Improved hierarchical production planning. *Journal of Operations Management*, 8(2):90–114, 1989.

[83] C. Libosvar. *Hierarchical production management: the flow-control layer.* PhD thesis, Université de Metz, 1988.

[84] C.W. Lin and C.L. Moodie. Hierarchical production planning for a modern steel manufacturing system. *International Journal of Production Research*, 27(4):613–627, 1989.

[85] J.G. Long. *Sur la conduite hierarchisee des systemes flexibles de production.* PhD thesis, L'Institut National Polytechnique de Grenoble, June 1993.

[86] A.G.J. MacFarlane. Information, knowledge and control. In H.L. Trentelman and J.C. Willems, editors, *Essays on Control: Perspectives in the Theory and its Applications*, volume 14 of *Progress in Systems and Control Theory*, pages 1–28. Birkhäuser Boston, 1993.

[87] M.S. Madan and K.C. Gilbert. An exact solution algorithm for a class fo production planning and scheduling problems. *Operations Research*, 43(10):961–970, 1992.

[88] J. Martinez and M. Silva. A simple and fast algorithm to obtain all invariants of a generalised petri net. In *2nd European Workshop on Application and Theory of Petri nets*, pages 411–422, 1981.

[89] A. Mehra. *Design and analysis of hierarchical production planning systems for parallel computing environments.* PhD thesis, University of Maryland, 1995.

[90] K. Meier. *Commande hiérarchisée d'un système de production.* PhD thesis, Université de Metz, January 1989.

[91] J.G. Miller and A.V. Roth. A taxonomy of manufacuring strategies. *Management Science*, 40(3):285–304, March 1994.

[92] T. Murata. Petri nets: properties, analysis and applications. *Proceedings of IEEE*, 77(4):541–580, 1989.

[93] R. Nagi. *Design and operation of hierarchical production management systems.* PhD thesis, University of Maryland, 1992.

[94] S.J. Nam and R. Logendran. Aggregate production planning - a survey of models and methodologies. *European journal of Operational Research*, 61:255–272, 1992.

[95] Y. Narahari and N. Viswanadham. A petri net approach to the modelling and analysis of flixible manufacturing systems. *Annals of Operations Research*, pages 449–472, 1985.

[96] J.J. O'Reilly and N.K. Ryan. Introduction to slam ii and slamsystem. In *Proc. of 1992 Winter Simulation Conference*, pages 352–358, Arlington, VA, USA, December 1992.

[97] J.S. Ostroff and W.M. Woham. A framework for real time discrete event control. *IEEE Trans. on Automatic Control*, 35(4):386–387, April 1990.

[98] J. Ousterhout. *Tcl and the Tk toolkit.* Addison-Wesley Publishing Company Inc., 1994.

[99] C.D. Petden, R.E. Shannon, and R.P. Sadowski. *Introduction to simulation with SIMAN.* McGraw-Gill, NJ, 1990.

[100] J.L. Peterson. *Petri net theory and the modeling of systems.* Prentice Hall, Inc., Englewood Cliffs, N.J., 1981.

[101] J.M. Proth and I. Minis. Production management in a petri net environment. *Recherche opérationnelle*, 29(3):321–352, 1995.

[102] J.M. Proth, N. Sauer, Y. Wardi, and X.L. Xie. Marking optimisation of stochastic timed event graphs using ipa. In *32nd IEEE Conf. on Decision and Control*, pages 686–691, San Antonio, Texas, December 1993.

[103] J.M. Proth, N. Sauer, and X.L. Xie. Stochastic timed event graphs: bounds, cycle time reachability and marking oprimisation. Technical Report 1763, INRIA, France, 1992.

[104] J.M. Proth and X.L. Xie. Cycle time of stochastic event graphs: evaluation and marking optimisation. *IEEE Trans. on Automnatic Control*, 39(7):1482–1486, July 1994.

[105] J.M. Proth and X.L. Xie. *Les reseaux de Petri pour la conception des systemes de production.* MASSON, 1995.

[106] P. J. Ramadge and W. M. Woham. The supervisory control of a class of discrete event systems. *SIAM J. Control and Optimization*, 25(1):206–230, January 1987.

[107] P. J. Ramadge and W. M. Woham. The control of discrete event systems. In *Proceedings of IEEE* [61], pages 81–98.

[108] C. Ramchandani. Analysis of asynchronous concurrent systems using petri nets. Technical Report 120, MIT, Cambridge, MA, 1974.

[109] W. Reisig. *Petri nets. An Introduction*, volume 4 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1985.

[110] M. Rose. Production planning at texas instruments improves service and reduce costs. *Industrial Engineering*, 23(1):33–36, 1991.

[111] W.G.M.M. Rutten. Hierarchical mathematical programming fo roperational planning in a process industry. *European Journal of Operational Research*, 64:363–369, 1993.

[112] G.H. Saad. Hierarchical production planning systems: extensions and modifications. *Operations Research*, 41(7):609–624, 1990.

[113] M. Sajkowski. Protocol verification using discrete event models. In Kurzhanski and Varaiya [73], pages 100–114.

[114] N. Sauer. *Les graphes d'evenements stochastiques et leur utilisation pour l'evaluation des systemes de production.* PhD thesis, Universite de Metz, September 1994.

[115] V. Savi. *Conception préliminaire des systèmes de production à l'aide des réseaux de Petri: évaluation des performances.* PhD thesis, Université de Metz, 1994.

[116] V.M. Savi and X.L. Xie. Liveness and boundedness analysis for petri nets with event graphs modules. In K Jensen, editor, *Application and Theory of Petri Nets 1992*, Lecture Notes in Computer Science, pages 328–347. Springer-Verlag, 1992.

[117] L. Shen, Q. Chen, and J.Y. Luh. Truncation of petri net models for simplifying computation of optimum scheduling problems. *Computersin Industry*, 20(1):25–43, 1992.

[118] H.M. Shih and T. Sekiguchi. A timed petri net and beam search based online fms scheduling system with routing flexibility. In *Proc. 1991 IEEE Int. Conf. on Robotics and Automation*, pages 2548–2553, Sacramento, California, April 1991.

[119] J. Sifakis. Use of petri nets for performance evaluation. *Acta Cybernet*, 4(2):185–202, 1978.

[120] M. Silva and R. Valette. Petri nets and flexible manufacturing. In G. Rozenberg, editor, *Advances in Petri nets*, volume 424 of *Lecture Notes in Computer Science*, pages 374–416. Springer-Verlag, 1989.

[121] P.H. Stark. Arc timed net analyser. *Petri Net Newsletter*, 37:27–33, 1990.

[122] I. Suzuki and T. Murata. A method for stepwise refinement and abstraction of petri nets. *Journal of Computer and Systems Sciences*, 27:51–76, 1983.

[123] S.D. Thompson and W.J. Davis. An integrated approach for modelling uncertainty in aggregate production planning. *IEEE Transactions on SMC*, 20(5):1000–1012, 1990.

[124] S.D. Thompson, D.T. Watanabe, and W.J. Davis. A comparative study of aggregate production planning strategies under conditions of uncertainty and cyclic product demands. *International Journal of Production Research*, 31(8):1957–1979, 1993.

[125] H. Tsubone, H. Matsura, and T. Tsutsu. Hierarchical production planning system for a two-stage process. *International Journal of Production Research*, 29(4):769–785, 1991.

[126] R. Valette. Analysis of petri nets by stepwise refinements. *Journal of Computer ans System Sciences*, 18:35–46, 1979.

[127] L.N. Wassenhove and L.F. Gelders. Hierarchical integration in production planning: theory and practice. *Journal of Operations Management*, 3(1):27–35, 1982.

[128] X.L. Xie. *Controle hierarchique d'un systeme de production soumis a perturbations.* PhD thesis, Universite de Nancy, June 1989.

[129] X.L Xie. Superposition properties and performance bounds of stochastic timed event graphs. *IEEE Trans. on Automatic Control*, 39(7):1376–1386, July 1994.

[130] X.L. Xie. Analyse, conception et contrôle des systemes de production. Habilitation à diriger des recherches en sciences, Univeristé de Metz, May 1995.

[131] G. Zapfel and H. Missbauer. New concepts for production planning and control. *European Journal of Operational Research*, 67(3):297–320, 1993.

[132] M. C. Zhou, F. DiCesare, and A.A. Descrocher. A hybrid methodology for synthesis of petri net models for manufacturing systems. *IEEE Trans. Robotics and Automation*, 8(3):350–361, 1992.

[133] M. C. Zhou, F. DiCesare, and D. L. Rudolph. Design and implementation of a petri net based models for manufacturing system. *Automatica*, 28(6):1199–1208, 1992.

[134] P.H. Zipkin. Computing optimal lot sizes in the economic lot esheduling problem. *Operations Research*, 39:56–63, 1991.

[135] K. Zoller. Optinal disaggregation of aggregate production plans. *Management Science*, 17(8):533–579, 1971.

Index