UNIVERSITE DES SCIENCES ET TECHNOLOGIES DE LILLE

Année : 1998

THESE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE

Discipline : PRODUCTIQUE,
AUTOMATIQUE et INFORMATIQUE INDUSTRIELLE

présentée et soutenue publiquement

le 09 juillet 1998

par

Ouajdi KORBAA
Ingénieur Ecole Centrale de Lille

COMMANDE CYCLIQUE DES SYSTEMES FLEXIBLES DE PRODUCTION MANUFACTURIERE A L'AIDE DES RESEAUX DE PETRI : DE LA PLANIFICATION A L'ORDONNANCEMENT DES REGIMES TRANSTOIRES

directeur de thèse :

M. J.-C. GENTINA Professeur à l'Ecole Centrale de Lille

JURY

M. J. P. CASSAR	Examinateur	Professeur à l'Université de Lille I, Président
M. Y. DALLERY	Examinateur	Directeur de Recherche au CNRS - HDR
M. J. ERSCHLER	Rapporteur	Professeur à l'INSA de Toulouse
M. H. OHL	Invité	Docteur, Ingénieur à Andersen Consulting
M. X. XIE	Rapporteur	Chargé de Recherche à l'INRIA Lorraine-HDR
M. P. YIM	Examinateur	Maître de conférences à l'Ecole Centrale Lille

Thèse préparée au Laboratoire d'Automatique et Informatique industrielle de Lille
L.A.I.L. URA CNRS D1440 - Ecole Centrale de Lille.

" Dieu bénit l'homme, non pour avoir trouvé, mais pour avoir cherché "

Victor Hugo

A mes parents

Remerciements

Le travail présenté dans ce mémoire a été effectué au Laboratoire d'Automatique et d'Informatique Industrielle de Lille (LAIL) dans le pôle Production Flexible Manufacturière (PFM) à l'Ecole Centrale de Lille (EC Lille) sous la direction scientifique de Monsieur le Professeur **J.-C. Gentina**, Directeur de l'EC Lille. Je tiens à le remercier très vivement pour l'accueil, l'encadrement et les précieux conseils dont j'ai bénéficié tout au long de ce travail.

Je suis très reconnaissant à :

Monsieur **J. Erschler**, Professeur à l'Institut National des Sciences Appliquées de Toulouse,

Monsieur **X. Xie**, Chargé de Recherches, H.d.R., INRIA Lorraine,

pour l'honneur qu'ils me font en acceptant d'examiner ce travail et d'être les rapporteurs de cette thèse.

Je tiens également à remercier :

Monsieur **J.P. Cassar**, Professeur à l'Université de Lille I,

Monsieur **Y. Dallery**, Directeur de Recherche au CNRS, H.d.R.,

Monsieur **H. Ohl**, Docteur, Ingénieur à Andersen Consulting

Monsieur **P. Yim**, Maître de conférences à l'EC Lille,

pour l'honneur qu'ils me font en examinant ce travail et en acceptant de participer à mon jury de thèse.

J'adresse également une pensée particulière à tous les membres du LAIL, qui, par leur disponibilité et leurs conseils m'ont grandement aidé dans le développement de mes recherches. Je veux remercier tout particulièrement tous les thésards du LAIL : (**Hervé**, **Nathalie**, **Philippe**, ...) pour leur aide spontanée et désintéressée, leur sympathie et leur bonne humeur.

Je tiens également à exprimer toute ma gratitude et ma reconnaissance à mes amis pour leur soutien tout au long de ma thèse et plus particulièrement pour m'avoir supporté durant ma rédaction.

Enfin, je remercie très sincèrement Monsieur **M. Vangreveninge** pour la reprographie de ce mémoire ainsi que Mme **E. Vérin** pour sa gentillesse et sa disponibilité.

Sommaire

Annexe III 223

Annexe IV 239

Introduction

Chapitre I

Chapitre II

Chapitre III

Chapitre IV

Introduction

Conduite de flux des Systèmes Flexibles de Production Manufacturière à l'aide d'une commande prévisionnelle

Il est un fait que la mondialisation de l'économie est en train de bouleverser considérablement et profondément les bases mêmes de notre économie et principalement du secteur de la production industrielle. En effet, les exigences de la clientèle sont de plus en plus difficiles à satisfaire en terme d'innovation, de qualité et de prix. Productivité, réactivité et réduction de coût se doivent donc de faire bon ménage.

Ainsi, les **S**ystèmes **F**lexibles de **P**roduction **M**anufacturière (SFPM) sont progressivement apparus en raison d'une part des progrès technologiques et d'autre part de la nécessité d'adapter la production à une demande de plus en plus variable. Un système de production flexible est un système automatisé utilisant des ressources paramétrables et possédant des degrés de liberté, sous forme d'indéterminismes de fonctionnement, lui permettant d'ajuster de manière fine (optimale) la production à l'évolution de la commande. En effet, les demandes de productions tendent vers des **moyennes voire petites séries**. L'un des avantages majeurs de ce type de système est d'une part de pouvoir produire simultanément un certain nombre de pièces de types différents et d'autre part de modifier la production et les flux de manière réactive.

Néanmoins, le coût assez élevé des investissements engendrés par les systèmes flexibles nécessite de bien spécifier les besoins en production et de maîtriser le fonctionnement de l'atelier de production. Il faut donc être en mesure de dimensionner l'atelier, d'optimiser le flux de production (augmenter la productivité), gérer les différents types de pièces cohabitant dans le système (varier la production) et pouvoir guider l'évolution du système en faisant face aux aléas de la production. Une telle démarche d'ingénierie comporte nécessairement une phase de **conception** comprenant la **modélisation**, l'**analyse** et l'**évaluation des performances** et la **détermination de la commande des SFPM**. Plusieurs travaux ont été menés dans ce sens au sein de la communauté de recherche avec

notamment le développement d'une méthodologie de Conception Assistée depuis la spécification jusqu'à l'implantation sur site des Systèmes de Production Automatisés en Industrie Manufacturière (CASPAIM) au sein du groupe Production Flexible Manufacturière (PFM) du Laboratoire d'Automatique et d'Informatique industrielle de Lille (L.A.I.L.) : cf. [GEN 88], [KAP 88], [BOI 91], [CRU 91], [HAM 91], [TOG 92], [BOU 93a], [ELK 93], [AMA 94], [AUS 94], [CRA 94], [HUV 94], [OHL 95a], [TAW 95], [CAS 96], [KER 96] et [CAM 97].

Ce mémoire s'intéresse essentiellement à l'élaboration d'une commande des SFPM depuis la phase conception (Analyse des Performances) jusqu'à la phase d'exploitation (Ordonnancement). Nous avons opté pour une étude **hors-ligne**, sous l'hypothèse de pannes rares, en vue de la détermination d'une **commande prévisionnelle** à flux de production maximal qui sera utilisée comme référence de production. Cette approche est à opposer aux commandes en-ligne (commandes temps réel , cf. [ERS 76], [BIL 93], [ROU 94] et [BIL 95], et commandes réactives, cf. [MAN 96]). Ces types de commande utilisent généralement des règles locales de décision et présentent l'avantage de pouvoir s'adapter en cas de défaillances ou de dégradation de fonctionnement.

Nous nous basons essentiellement sur les travaux de H. Ohl et H. Camus, cf. [OHL 95a] et [CAM 97], qui ont abouti à une méthodologie structurée pour l'évaluation des performances et l'ordonnancement des SFPM. Ce problème consiste à résoudre progressivement les flexibilités qui caractérisent ces systèmes de production afin d'optimiser la production et construire une **commande cyclique** qui permet d'atteindre les performances optimales du système. L'implantation de cette commande nécessite un module appelé *pilotage*, cf. [TAW 95], et un module de *surveillance*, cf. [ELK 93], afin de comparer la production réelle et le comportement déterministe optimal élaboré au niveau de l'ordonnancement. Nous tolérons cependant un type de perturbation qui consiste en l'introduction d'une commande prioritaire ou urgente. Pour cela nous allons utiliser le caractère répétitif de la commande afin de rechercher le meilleur instant d'introduction de la commande tout en minimisant la perte de temps, cf. Figure 1. Dans le cas de la détection de réelles dégradations([1]) des performances de fonctionnement, le module surveillance se charge de faire un diagnostic de la situation en « temps réel » pour décider soit de continuer

[1] Ce phénomène se manifeste par une augmentation des temps opératoires due à l'usure des outils par exemple.

la production, si la défaillance est bénigne, soit de lancer une nouvelle évaluation de la commande prévisionnelle pour prendre en compte ces nouvelles données.

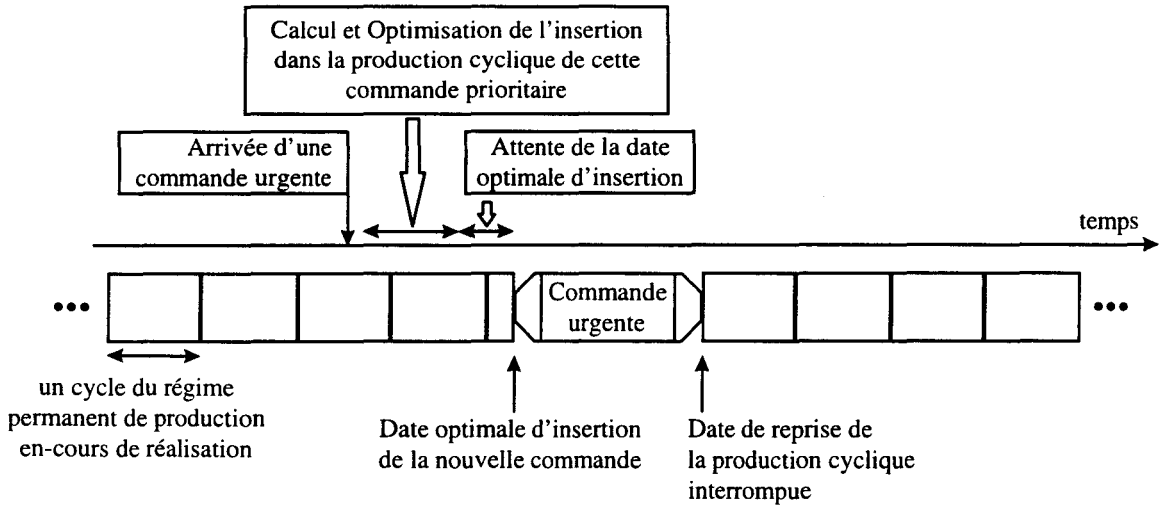


Figure 1 : Réactivité par rapport à l'arrivée d'une commande urgente, cf. [CAM 97]

La panne franche de certaines ressources est plus contraignante. Elle risque de modifier certaines caractéristiques du système (des ressources voire même des parties entières peuvent être hors circuit ou inaccessibles). Le module *recouvrement*, cf. [BER 96], se charge alors de détecter la nouvelle configuration du système qui sera utilisée dans le calcul d'une nouvelle commande prévisionnelle. Ainsi, le fonctionnement de l'atelier, tel qu'il a été prévu dans le projet CASPAIM, cf. Figure 2, est le suivant : une commande initiale est planifiée sous forme de programme de production et transférée à la *planification fine*. Ce module se charge de décomposer chaque production en un ensemble de productions cycliques qui seront ordonnancées afin de déterminer les commandes totales (régimes permanents cycliques et régimes transitoires).

Ces commandes prévisionnelles et déterministes sont ensuite transmises au module *pilotage*, pour être mise en oeuvre, et au module *surveillance*, afin de suivre l'évolution de la production et détecter les éventuels écarts par rapport à la commande optimisée. Dans le cas d'une commande urgente et prioritaire, le module d'*ordonnancement* se charge de trouver le meilleur instant de son lancement et le communique au pilotage et à la surveillance.

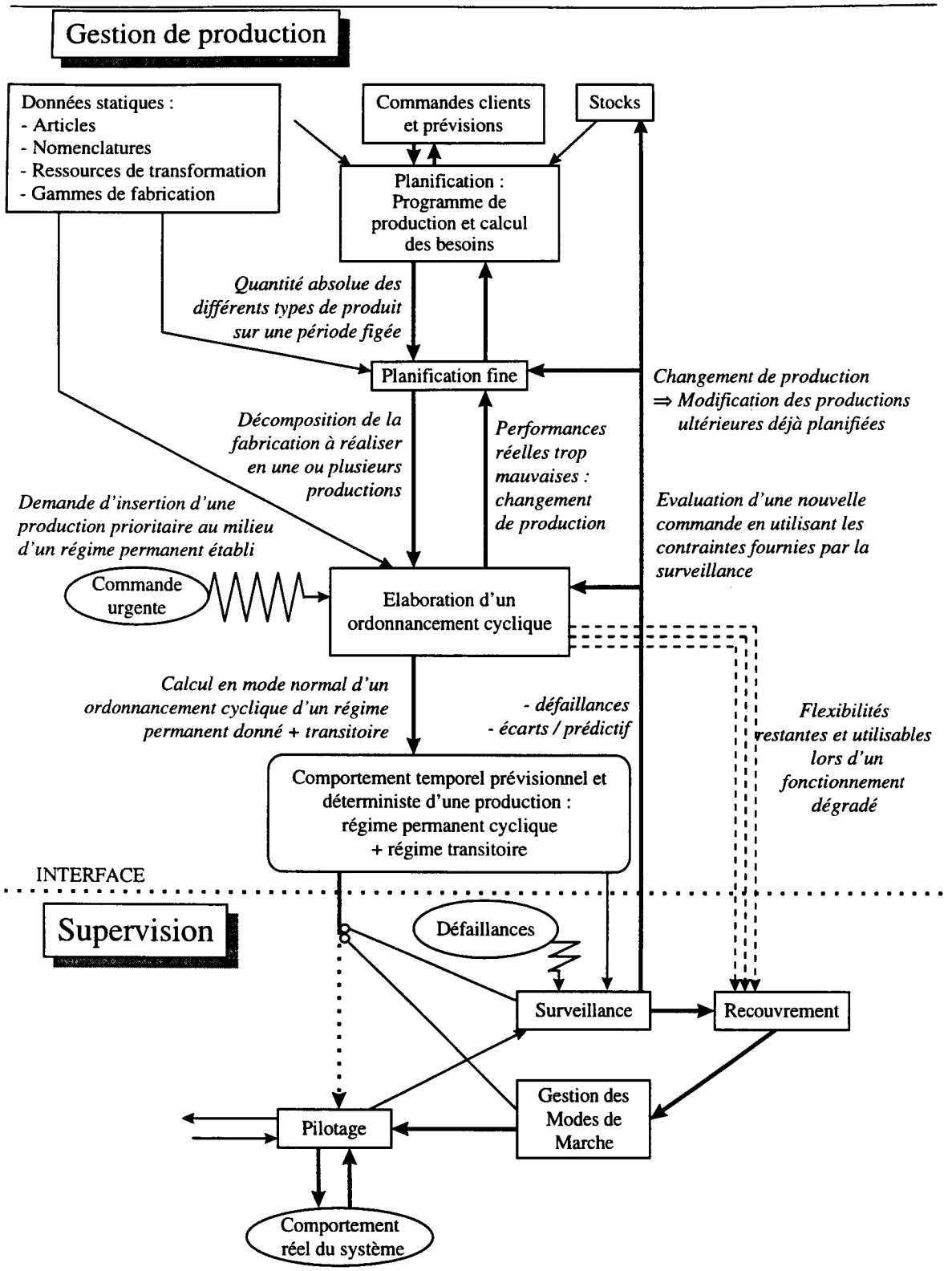


Figure 2 : Interface Gestion de production - Supervision

Le module *pilotage* conserve la maîtrise de la situation même dans le cas d'une légère dégradation auquel cas il a pour mission de continuer la production en essayant d'approcher au mieux de la commande optimale. Cependant, si une défaillance grave est constatée, par la *surveillance*, le module *recouvrement* se charge de déterminer les parties

encore en état de fonctionnement pour les comparer aux flexibilités restantes que le module *ordonnancement* lui communique. Nous pouvons, alors, avoir deux cas de figures : soit le système peut encore fonctionner, avec plus ou moins d'efficacité, soit ceci est impossible. Dans le premier cas, la *gestion des modes* prend la relève pour assurer la fin de la production. Quant au second, il nécessite l'ordonnancement d'une nouvelle commande, avec les nouvelles données du système, voire même une nouvelle planification si il s'avère impossible de produire un ou plusieurs types de pièces.

L'ingénierie des SFPM met en œuvre plusieurs modules différents, il apparaît donc indispensable d'utiliser un outil fédérateur pour la modélisation et l'interfaçage de notre approche avec ces différents modules, cf. Figure 2. C'est ainsi que nous avons opté pour les **Réseaux de Petri** (RdP), outil générique de modélisation des Systèmes à Evénements Discrets, cf. [SIL 96a et b], comme outil de modélisation et d'intégration de nos travaux au sein du projet CASPAIM. En effet, les compétences de l'équipe PFM vont de la conception à l'implantation. Il est donc intéressant de pouvoir utiliser un outil de modélisation cohérent : ce qui est tout à fait possible avec les RdP. Les modèles RdP que nous utilisons ont été développés et défini dans le cadre de CASPAIM par [CRU 91] et [AMA 94].

Parmi les centres d'intérêts du projet CASPAIM, nous nous intéressons plus particulièrement, au sein de l'équipe **Evaluation de Performances**, aux problèmes de planification fine, d'optimisation des performances et de détermination de la commande. Les travaux menés jusque là, cf. [OHL 95a] et [CAM 97], ont abouti à la détermination d'une méthodologie basée sur une approche structurée de résolution du problème.

Dans cette démarche, les indéterminismes du système sont progressivement levés pour aboutir à un Graphe d'Evénements représentant l'ordonnancement cyclique déterministe. Nous pouvons distinguer trois grands axes dans cette démarche : la planification fine, la résolution des flexibilités et l'ordonnancement. Le premier a été formalisé et codé, cf. [CAM 97].

La résolution des indéterminismes a été développée, cf. [OHL 95a] et partiellement étudiée d'un point de vue combinatoire, cf. [CAM 97]. Quant à l'ordonnancement, l'équipe évaluation de performances a étudié le problème et présenté une première application, cf. [OHL 95a], améliorée par la suite, cf. [CAM 97]. Les résultats que nous présentons dans ce mémoire s'attachent à prolonger ces travaux en approfondissant l'étude dans l'objectif

d'une informatisation totale de la méthode d'élaboration d'un ordonnancement complet d'une demande de production.

Plan de la thèse

Ce mémoire se décompose en quatre chapitres et traite les différentes phases de la résolution dans l'ordre logique requis, cf. Figure 3.

Dans le **Chapitre I** nous présentons, dans un premier temps, le contexte, les définitions et les hypothèses de l'étude. Puis, nous étudions les différents degrés de liberté et les différents moyens de les résoudre, à travers une revue de l'état de l'art, pour aboutir à une justification, a posteriori, de la démarche proposée par [OHL 95a] et [CAM 97]. Après avoir choisi la méthode de résolution, nous présentons et discutons les critères de performances utilisés et l'outil de modélisation adopté. Puis, nous présentons la première étape (planification fine) qui décompose la demande en plusieurs productions cycliques.

Le **Chapitre II** concerne l'analyse des différentes phases précédant l'ordonnancement. Chaque étape est formalisée et étudiée d'un point de vue combinatoire. Cette étude mathématique vise à déterminer un algorithme de résolution, le nombre de solutions attendues ou au moins une estimation de ce dernier. Cette démarche vise à contribuer à l'informatisation complète de l'ensemble de la méthodologie. En fin de ce chapitre, nous obtenons un graphe ordonnançable.

Le **Chapitre III** donne, tout d'abord, le principe de l'algorithme d'ordonnancement élaboré dans [KOR 97a]. Puis, nous abordons les difficultés dues au codage de cet algorithme qui aboutit à l'ordonnancement d'un régime permanent cyclique modélisé par un Graphe d'Evénement représenté également par un Diagramme de Gantt. Cet algorithme ne prenant pas en compte les opérations de transfert, nous présentons ensuite une approche préliminaire de contrôle du système de transport. Nous étudions alors l'influence des opérations de transfert sur l'en-cours du système ainsi que les problèmes de blocage par saturation. Puis, nous donnons le principe d'une heuristique d'ordonnancement avec opérations de transfert et l'appliquons à un exemple d'illustration.

Le **Chapitre IV** est consacré à l'analyse et l'optimisation des régimes transitoires. En fait, dans l'hypothèse de grandes séries il est assez naturel de négliger le régime transitoire, cf. [MUN 91]. Néanmoins, si nous nous intéressons à des petites ou moyennes séries, les régimes transitoires ne sont donc plus à négliger. C'est dans cet esprit que nous étudions

les régimes transitoires en répertoriant les différents types de transitoires et les difficultés inhérentes à chacun d'entre eux. Nous effectuerons ensuite la résolution théorique et analytique des transitoires de lancement et d'arrêt de la production.

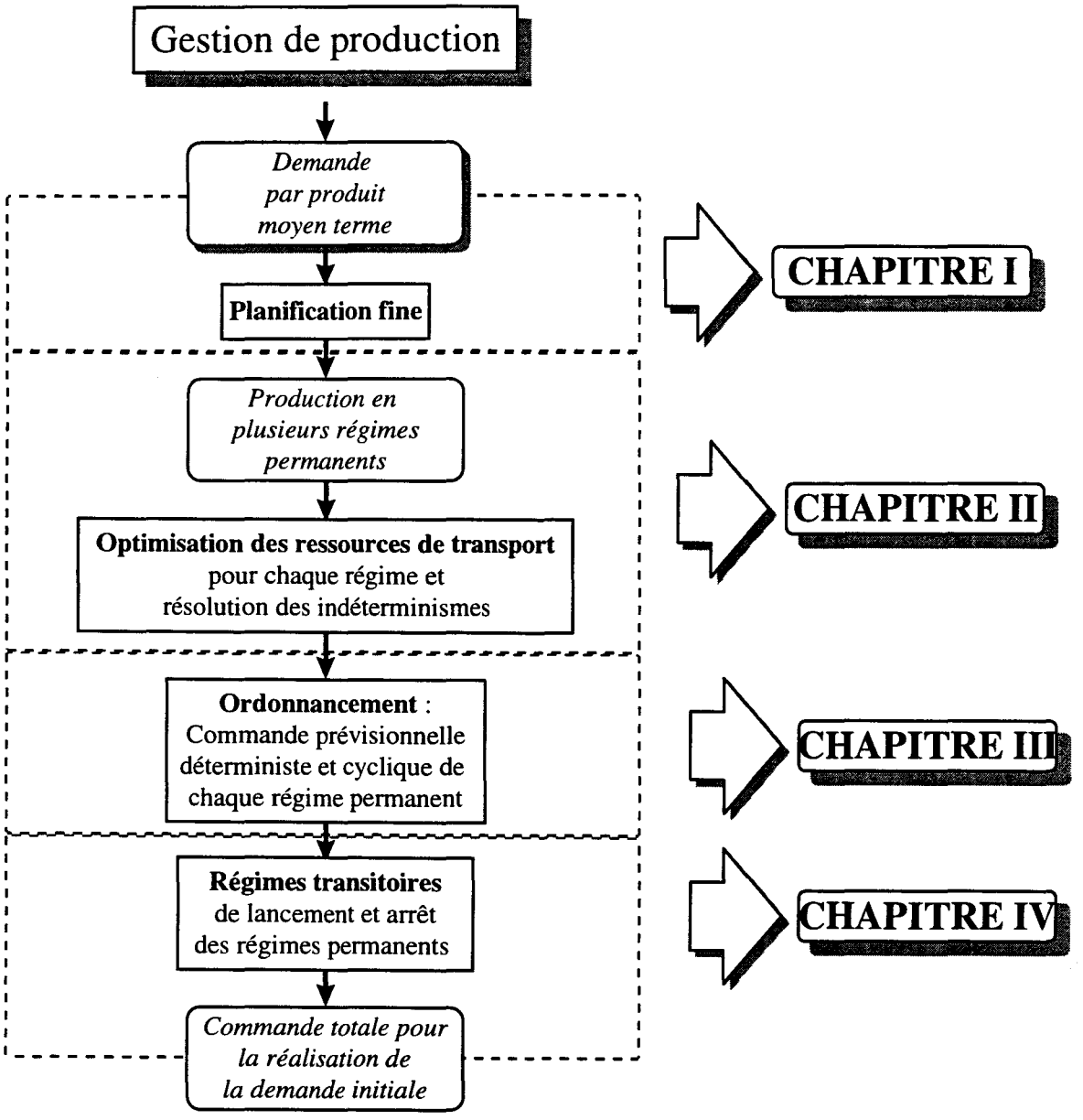


Figure 3 : Plan du mémoire

En conclusion générale, nous dressons un bilan des différents résultats proposés dans ce mémoire. Puis, nous présentons quelques perspectives à court terme et moyen terme. Dans le souci de ne pas alourdir la présentation de ce mémoire, nous avons préféré reporter en annexes les démonstrations mathématiques.

Chapitre I

I. Commande Prévisionnelle des SFPM : Position et Formulation du Problème

I.1 Position du Problème

Le problème abordé dans ce mémoire concerne la recherche d'une commande prévisionnelle totale dans un environnement de Production Flexible Manufacturière. La commande est déterminée par la nature ou les types des pièces à produire ainsi que par le nombre d'exemplaires de chaque type issus du carnet de commande ou du service de gestion de production. L'optimisation de la commande, dans notre étude, aura pour principal objectif la minimisation du temps total de production (couramment appelé Makespan) ainsi que des objectifs complémentaires d'ordre quantitatif (coût, en-cours utilisé, palettes engagées dans le système, ...) et qualitatif (complexité de calcul et d'implémentation, ...).

Une telle problématique est justifiée par la rigueur de la compétitivité et l'on pourrait s'interroger sur les raisons apparemment paradoxales d'une bibliographie relativement peu abondante sur ce sujet. En effet, dans la littérature on trouve assez souvent des études partielles de cette problématique générale. En réalité, la combinatoire en est d'une très grande complexité ce qui dissuade les chercheurs de se lancer dans la détermination d'une approche d'optimisation globale. Cependant, nous avons pu montrer au sein de l'équipe PFM qu'il était possible de découpler certains problèmes pour les étudier séparément. L'avantage d'une telle approche étant évidemment de décomposer un problème d'une très grande complexité en plusieurs problèmes « abordables ». Néanmoins, l'inconvénient majeur réside dans la possibilité d'accroissement du nombre des solutions intermédiaires à considérer et donc à l'amplification des délais de calcul. La difficulté se situe donc principalement dans la recherche d'une technique de décomposition qui ne pénalise pas les objectifs d'optimisation précités.

Au cours de ce chapitre, nous allons, dans un premier temps, présenter le contexte de l'étude. Nous proposerons ensuite de formuler le problème abordé et de rappeler les

résultats émanant d'études antérieures portant sur le domaine. Après avoir positionné le problème et rappelé les hypothèses et les principales définitions, nous présenterons la méthode de résolution choisie, méthode initialement élaborée par Harald Ohl et Hervé Camus dans leurs travaux de thèse successifs ([OHL 95a] et [CAM 97]) au sein du LAIL. Cette approche est progressive, elle tient compte successivement des différents degrés de liberté que possède le système considéré. A la suite de cette présentation, nous discuterons la linéarisation proposée pour en justifier a posteriori les principales motivations. Rappelons que l'objectif ultime de l'Equipe Production Flexible Manufacturière concerne la mise au point d'une approche intégrée et automatisée de conception des Systèmes Flexibles de Production Manufacturière et consiste en particulier à s'intéresser à l'Evaluation des Performances et à l'Ordonnancement des Ateliers Flexibles.

La plupart des définitions et hypothèses rappelées dans ce chapitre sont issues des travaux de l'équipe Evaluation de Performances. Nous avons jugé nécessaire de les présenter de façon succincte afin de ne pas alourdir inutilement ce mémoire. Nous nous efforcerons d'en faire la synthèse tout en recommandant au lecteur de ce mémoire de consulter ces références pour de plus amples informations.

I.2 Contexte de Production

I.2.1 Systèmes Flexibles de Production Manufacturière

Les Systèmes Flexibles de Production Manufacturière (SFPM) représentent une large classe des systèmes de production. Ils sont actuellement en évolution sensible. Le contenu exact du terme « flexibilité » est sujet à différentes interprétations. Pour être plus précis, les SFPM correspondent à des systèmes de production de petites à moyennes séries sous un certain nombre de degrés de liberté. Ils sont à mi-chemin entre les systèmes de production de masse (production automobile par exemple) où la production est linéaire (sans flexibilité opératoire) et les ateliers « totalement flexibles » où la production est unitaire, cf. Figure I-1. Les systèmes considérés dans ce mémoire sont à opposer aux systèmes informatiques qui banalisent les en-cours et les routages pour s'intéresser essentiellement aux opérations à effectuer.

Le SFPM est donc « ***un atelier constitué d'un ensemble d'éléments capables de fabriquer un ensemble de pièces par programmation*** », cf. [KER 96]. Il n'existe guère de SFPM type ou standard et on peut classer dans cet ensemble un type particulier appelé *Atelier flexible*. Les ateliers flexibles sont dédiés à la production de **petites et moyennes séries**, cf. Figure I-1. « ***Ils forment une solution et un outil pour résoudre le problème de fabriquer le bon produit au bon moment en quantités nécessaires et suffisantes et au moindre coût*** », cf. [BAR 92]. Nous nous intéressons ici à des ateliers flexibles où les opérations de transformation réalisées sur les pièces sont de type **usinage** (fraisage, perçage, tournage, ...).

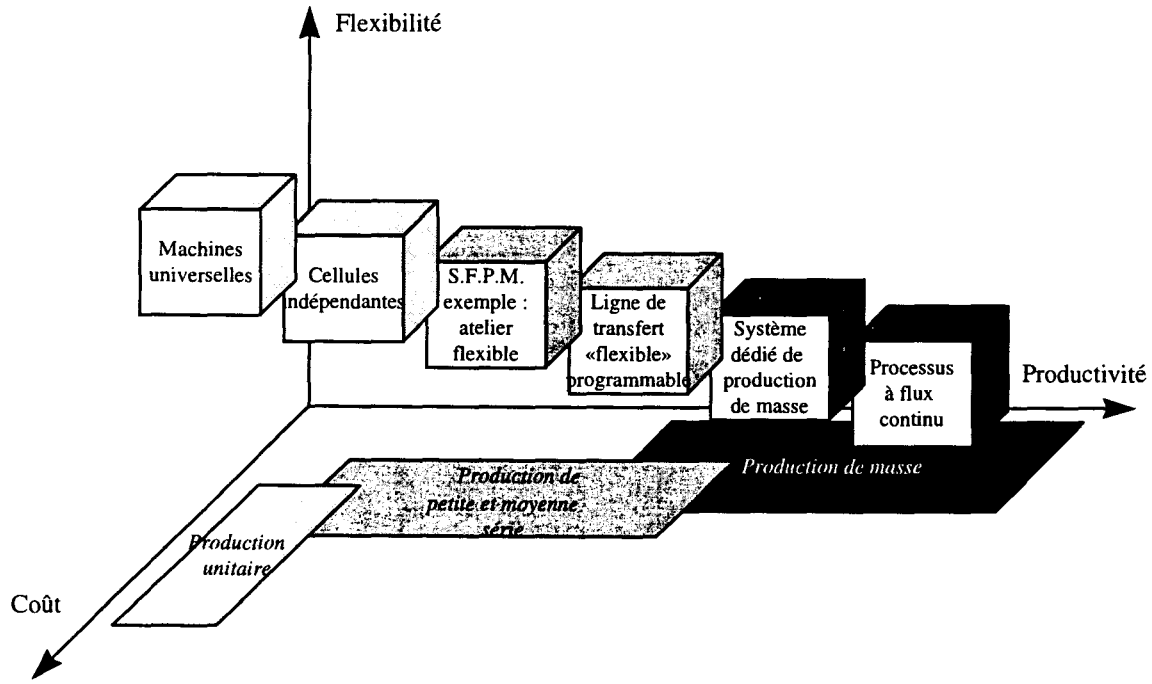


Figure I-1 : Productivité, Flexibilité et Coût [KER 96]

Considérons une production simultanée de différents types de pièces dans un atelier flexible. Le procédé de fabrication de chacun de ces types de pièces peut ne pas être unique. C'est pourquoi nous envisageons certains degrés de liberté au niveau des gammes de fabrication de chaque type de pièces, cf. § I.2.1, pour distinguer les différentes classes de flexibilités de production.

I.2.2 Hypothèses de Production

Nous supposons qu'il n'y a pas de préemption des tâches lors du fonctionnement normal du système, c'est-à-dire qu'une opération de transformation sur une pièce conserve la ressource machine associée jusqu'à la fin de l'usinage de cette pièce. De plus, nous approximons les durées opératoires par des valeurs déterministes. Certes cette hypothèse peut paraître restrictive et inappropriée face à des opérations dont la durée peut varier autour d'une valeur moyenne (intervention humaine par exemple). Cependant les autres techniques d'approximation de ces durées (lois aléatoires exponentielles par exemple) sont elles aussi associées à des hypothèses dont la validité n'est que relative. Ajoutons que notre choix semble judicieux dans le cadre de systèmes de production où toutes les opérations sont a priori automatisées. Nous supposerons donc qu'il n'existe pas d'indéterminisme sur les temps opératoires. Cette hypothèse nous permettra de justifier l'existence d'une **commande déterministe** de notre système pour lequel tous les paramètres seront a priori évaluables (hypothèse de causalité déterministe).

Nous sommes néanmoins conscients du risque de dégradation du fonctionnement de l'atelier et donc de la possibilité d'existence d'écart entre la commande prévisionnelle, que nous cherchons à obtenir, et le déroulement réel de la production. Cette hypothèse aura de ce point de vue l'avantage de nous permettre d'estimer le makespan de la production et d'atteindre la vitesse optimale de production. De ce fait toute variation de la performance réelle en regard de la performance théorique permettra de détecter la défaillance partielle de l'atelier. Nous postulerons donc que la commande prévisionnelle, que nous voulons établir, pourra servir de référence optimale pour la phase de pilotage du système. Il s'agira dans la réalité d'approcher au mieux cette commande.

I.2.3 Les pannes de fonctionnement

Nous supposons donc que **les pannes des machines sont assez rares** et par conséquent qu'il existe des intervalles de temps suffisamment longs pendant lesquels la commande prévisionnelle pourra être pleinement exploitée sans interruption. Il découle de cette hypothèse la possibilité d'atteindre les performances optimales prévues et de les maintenir sur un temps suffisamment long. Il est donc impératif de bien connaître les fréquences de pannes afin de pouvoir valider cette hypothèse. Les échelles temporelles des événements

pris en compte, énoncées dans [KIM 83] et connues sous le nom d'**hypothèses de Gershwin**, seront donc essentielles pour justifier la mise en œuvre de l'approche déterministe. L'hypothèse de fonctionnement déterministe peut sembler contraignante, mais elle permet dans une large mesure d'optimiser les flux de production et de maîtriser la production à tout moment par une connaissance précise de l'état du système.

De plus, il sera possible d'identifier une dégradation de la performance lors de l'apparition de défaillances du système. Le module de supervision devra alors gérer les modes de marche dégradés. Dans le pire des cas, lorsqu'il sera impossible d'appliquer la commande prévisionnelle, nous proposerons de recalculer une nouvelle commande déterministe. C'est dans ce contexte que nous appliquerons un ordonnancement hors-ligne, cependant suffisamment réactif([1]) pour pouvoir faire face rapidement à des défaillances graves (panne d'une des machines critiques, par exemple). En phase d'exploitation, nous devrons donc prendre en charge des contraintes temporelles strictes de calcul d'ordonnancement.

I.2.4 Les ressources du système

Les ressources du système considéré sont des ressources de transformation (machines d'usinage, de fraisage ...). Elles peuvent être **simples** (unitaires) ou **multiples** (composées de machines identiques([1]) capables d'effectuer les mêmes opérations dans le même temps opératoire). Néanmoins, ces ressources sont, dans tous les cas, limitées en nombre et caractérisent la vitesse de production. Nous rappelons que les opérations sont non préemptives. Nous considérerons également deux autres types de ressources : d'une part le système de transport et d'autre part l'en-cours que nous chercherons à optimiser afin de minimiser les immobilisations matérielles.

Pour l'instant nous n'avons pas envisagé d'étudier le cas des ressources imbriquées en raison des blocages structurels qu'elles peuvent entraîner. Néanmoins, nous pouvons être confrontés à des problèmes de blocage dus à la saturation du système physique de transport par les en-cours (saturation de la capacité finie du système de transport), cf. [AUS 94] et [GEN 97].

[1] Au sens du temps de calcul ou plus tôt de recalcul de la commande.

En fait, la situation de blocage par saturation ne devrait pas intervenir dans le cas de recherche du minimum d'en-cours respectant le débit maximal. Dans le cas défavorable d'une saturation du système avec en-cours minimal, le contexte d'étude sera différent et il s'agirait de trouver le meilleur débit à en-cours fixé. Ce problème est aussi à envisager dans le cas où les ressources de transport sont limitées.

Concernant le système de transport, nous ne traiterons pas le problème posé par sa conception (notamment le choix de l'architecture de transport). Nous considérerons donc que le système de transport ne constitue pas une ressource critique (ressource ayant une influence sur le débit ou la vitesse du système) pour le système de production. Nous ne prendrons donc pas en compte des opérations de transfert lors de l'évaluation préliminaire des performances du système et notamment dans la recherche de la machine critique.

Nous verrons en particulier qu'il est tout à fait possible d'intégrer les opérations de transfert dans la phase d'ordonnancement. Nous étudierons notamment la conséquence de l'hypothèse souvent implicite de la réalisation des opérations de transfert en temps masqué. Concernant le transport, nous supposons que le modèle graphique qui lui est associé est fortement connexe, ceci pour assurer le transfert de toute pièce d'un point quelconque à un autre quelconque de l'atelier flexible.

Le système de transport connecte donc toutes les ressources de transformation et détermine ainsi les relations d'accessibilité. Dans le cas d'une défaillance de cette relation, durant la production, le module *recouvrement* se chargera de réévaluer le graphe d'accessibilité pour analyser le risque de secteurs inaccessibles du système (cf. travaux de P. Berruet [BER 96]). Ajoutons que la possibilité de disposer de plusieurs chemins pour aller d'une machine à une autre permet de pallier certaines défaillances du système de transport et donc d'éviter l'arrêt total de la production.

Nous supposerons de plus que le système de transport considéré autorise la présence de buffers d'entrée (voire de sortie) pour chaque machine. L'attente potentielle en amont d'une machine en assure l'éventuelle saturation.

---

[1] Nous n'émettons aucune hypothèse quant à l'emplacement réel de ces ressources identiques : elles peuvent être géographiquement éloignées ou rapprochées.

En effet, la ressource devient immédiatement utilisable dès sa libération par la pièce précédente. Les ressources de transport sont supposées être de type **palettes** (physiques ou virtuelles), de plus les pièces (brutes ou semi-finies) **restent affectées définitivement à leurs ressources de transport** de l'entrée dans le système jusqu'à la fin de fabrication des pièces. Par le terme « *affectées* » nous entendons que même si la pièce est dissociée de sa palette durant un usinage, la palette restera en attente au pied de la machine pour être rechargée par la même pièce en fin d'opération. De plus, aucune palette ne sera injectée « vide » dans le système de transport pour éviter d'encombrer inutilement le système de transport.

Les palettes peuvent être **universelles** (pouvant transporter tout type de pièces) ou([1]) **dédiées** à une ou plusieurs familles de pièces. Dans le second cas, nous distinguerons les différents types de palettes.

Notons H le nombre de ces classes et $h \in \{1..H\}$ l'un des types de palettes. S_h représente alors l'ensemble des produits que les ressources de transport de type h peuvent supporter.

Nous supposerons de plus que deux types différents de palettes ne peuvent transporter le même type de produits : $\forall h,\ \forall h' \neq h,\ S_h \cap S_{h'} = \varnothing$. Ainsi la considération des ressources de transport associées à chaque type de produits permet ici d'obtenir une partition mathématique de l'ensemble des types de pièces à fabriquer dans l'atelier flexible. Ces hypothèses vont permettre d'étudier indifféremment l'optimisation du maximum de l'en-cours et l'optimisation des ressources de transport : elles sont, dans ce cas, confondues.

I.3 Evaluation des Performances et Ordonnancement

I.3.1 Degrés de liberté du problème

Notre ultime objectif, dans le cadre des hypothèses avancées précédemment, consiste à prendre en charge le problème d'élaboration de la commande depuis le niveau planification fine (objectifs de production avec les dates de livraison associées) jusqu'à la détermination de l'ordonnancement.

[1] Ou exclusif : Les palettes sont soit dédiées soit universelles.

Afin de réduire la complexité de ce problème, plusieurs fois Non Polynomiale difficile (NP-difficile[1]), l'équipe Production Flexible Manufacturière a proposé de découper en classes de décisions les différentes flexibilités potentielles. Ces classes de décision représentent chacune des phases élémentaires, cf. Annexe I, d'optimisation selon les critères de performance qui seront introduits par la suite, cf. § I.3.3. Elles sont reprises ci-après :

- **Classe D_1 :** *Choix des produits à fabriquer simultanément (Planification fine)*

Ce paramètre déterminera en grande partie la performance du système, cependant, il peut être influencé voire limité par des contraintes extérieures ou de faisabilité ou techniques ainsi que par le carnet de commande.

- **Classe D_2 :** *Choix des ratios de production*

C'est le nombre de pièces à fabriquer de chaque type il peut être exprimé soit sur les entiers (nombre exact de pièces) soit sur les rationnels strictement positifs (pourcentage de pièces par rapport au nombre total).

- **Classe D_3 :** *Résolution des flexibilités de gamme*

Nous appelons flexibilité l'opportunité de différentes procédures utilisées pour fabriquer un type de pièces. Les cas que nous traitons dans notre étude sont :

1. sous-classe D_{3a} : *flexibilité de permutation* : elle exprime la possibilité d'exécuter certaines opérations sans « ou presque pas » de contraintes de précédence strictes. Cette flexibilité n'a aucune incidence sur la charge des machines concernées puisque quelque soit le chemin qu'on choisira on utilisera les mêmes machines avec les même durées opératoires.

2. sous-classe D_{3b} : *flexibilité d'affectation* : elle traduit le fait qu'une même opération peut être effectuée par des ressources de types différents avec des durées opératoires différentes.

[1] Contrairement aux problèmes NP-complets, nous ne pouvons trouver, pour les problèmes NP-difficiles, d'isomorphismes avec des problèmes « de référence » tel que : SAT : Problème de satisfaction de contraintes dans une équation booléenne d'ordre zéro.

3. sous-classe D_{3c} : *flexibilité de procédé* : elle représente la possibilité de substituer un une suite d'opérations par une autre suite d'opérations, éventuellement sur des ressources de types différents.

4. sous-classe D_{3d} : *flexibilités complexes* : cette sous-classe, qui ne figure pas dans les degrés de liberté présentés dans la thèse d'Harald Ohl et est désormais prise en compte grâce à l'étape de planification fine élaborée par Hervé Camus, cf. § I.3.1. Elle comporte les flexibilités imbriquées et enchaînées.

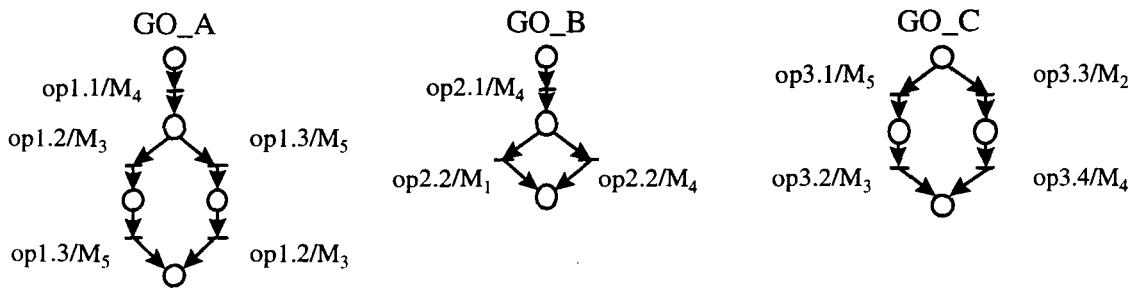


Figure I-2 : Modélisation des trois premiers types de flexibilités de gammes opératoires

La flexibilité enchaînée consiste en la succession de deux (ou plus) des flexibilités précédentes. Dans l'exemple de la Figure I-3, nous avons représenté une flexibilité enchaînée comportant deux flexibilités d'affectation successives.

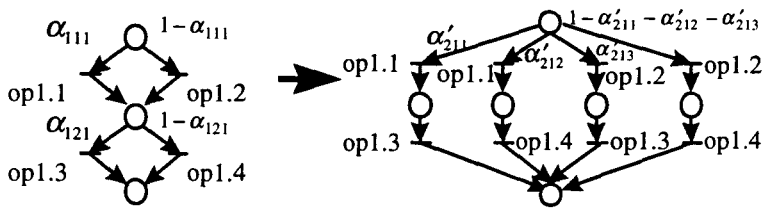
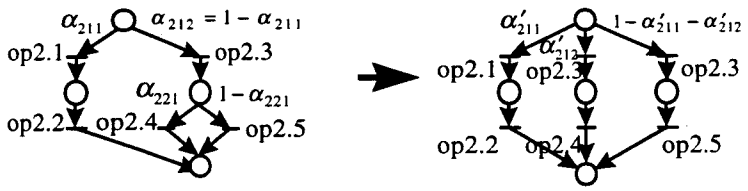


Figure I-3 : Exemple de flexibilité enchaînée

Quant à la flexibilité imbriquée, elle consiste à faire intervenir une nouvelle flexibilité dans une branche déjà ouverte. Dans l'exemple Figure I-4, la flexibilité imbriquée est formée d'une flexibilité de procédé qui contient dans une de ces deux branches (branche α_{212}) une flexibilité d'affectation.



flexibilité imbriquée

Dépliage de la flexibilité imbriquée

Figure I-4 : Exemple de flexibilité imbriquée

Ces deux flexibilités compliquent considérablement les calculs par rapport aux trois précédentes. En effet, si les trois premières sous-classes engendrent des équations linéaires pour le calcul de flux, la flexibilité imbriquée engendre obligatoirement des équations bi- (voire multi-) linéaires, cf. [OHL 95a].

En ce qui concerne la flexibilité enchaînée, le dépliage des différentes branches, pour l'obtention d'un modèle linéaire, conduit à une importante augmentation du nombre de variables en jeu, cf. Figure I-3.

- **Classe D_4 :** *Affectation d'une opération à une machine particulière*

Dans le cas des ressources multiples, après affectation d'une opération à une ressource, chaque opération doit être associée à un exemplaire unique de la ressource multiple.

- **Classe D_5 :** *Ordonnancement*

Cette classe représente la problématique visant à déterminer le séquencement et les instants d'affectation des produits sur les machines.

- **Classe D_6 :** *En-cours*

Cette classe concerne globalement les ressources de transport (palettes), elle peut être scindée en quatre sous-classes :

1. sous-classe D_{6a} : détermine le niveau (nombre) d'en-cours dans le système.

2. sous-classe D_{6b} : décide de la répartition de cet en-cours. Ce problème consiste à marquer le graphe d'événements (modèle final de l'ordonnancement recherché) pour garantir la vivacité tout en minimisant l'en-cours nécessaire.

3. sous-classe D_{6c} : concerne le choix des pièces qui utiliseront les même palettes

4. sous-classe D_{6d} : introduit les contraintes de précédence entre les différentes pièces utilisant les mêmes ressources de transport.

A cette liste déjà définie, nous ajoutons un dernier degré de liberté, non pris en compte jusqu'alors, qui concerne l'élaboration des transitoires de début/fin et changement de régimes de production :

- **Classe D_7 :** *Transitoire*

Cette classe représente la problématique visant à lancer la production à partir d'un système vide pour atteindre le régime permanent ou encore à vider le système en fin de production ou à changer de régime (permanent) de production dans les meilleures des conditions. Pour la résolution de l'ensemble de ces flexibilités, H. Ohl et H. Camus ont procédé à la linéarisation de ces critères en introduisant une hiérarchie entre les différentes classes. Cette approche séquentielle n'est que l'une des multiples possibilités de résolution de ce problème. Nous allons donc mettre en évidence le parallélisme entre les différents indéterminismes afin de justifier a posteriori l'approche suivie. Sur la Figure I-5, nous représentons les différentes classes de décision en modélisant uniquement les contraintes de précédence implicites qui les relient.

Pour commencer la résolution (niveau 1), nous devons déterminer les pièces à produire simultanément ainsi que les nombres d'exemplaires demandés (classes D_1 et D_2). Cette opération décomposera la demande initiale en une voire plusieurs productions. Ensuite, nous pouvons étudier :

- l'extraction des chemins (classes D_{3b}, D_{3c} et D_{3d}) : cette opération vise à lever l'indéterminisme sur les flexibilités de gamme avec des règles basées sur l'optimisation du flux du système de production qui prend en compte l'aspect discret de la commande.

- l'affectation d'une opération à une machine particulière (classe D_4) puis l'ordonnancement des opérations (classe D_5). La première étape gère l'indéterminisme lié aux ressources multiples. En effet, étant donné que certaines ressources existent en plusieurs exemplaires, il convient de distribuer « équitablement » les opérations à effectuer sur les différentes machines identiques tout en veillant à ne pas dégrader les performances

du système en terme de flux de matière. Quant à la seconde, elle s'occupe de donner un ordre de passage aux opérations affectées à une machine ainsi que les dates de passage.

⇨ La contrainte de précédence entre ces deux classes est stricte, c'est ce que nous avons exprimé par la flèche orientée entre les deux classes.

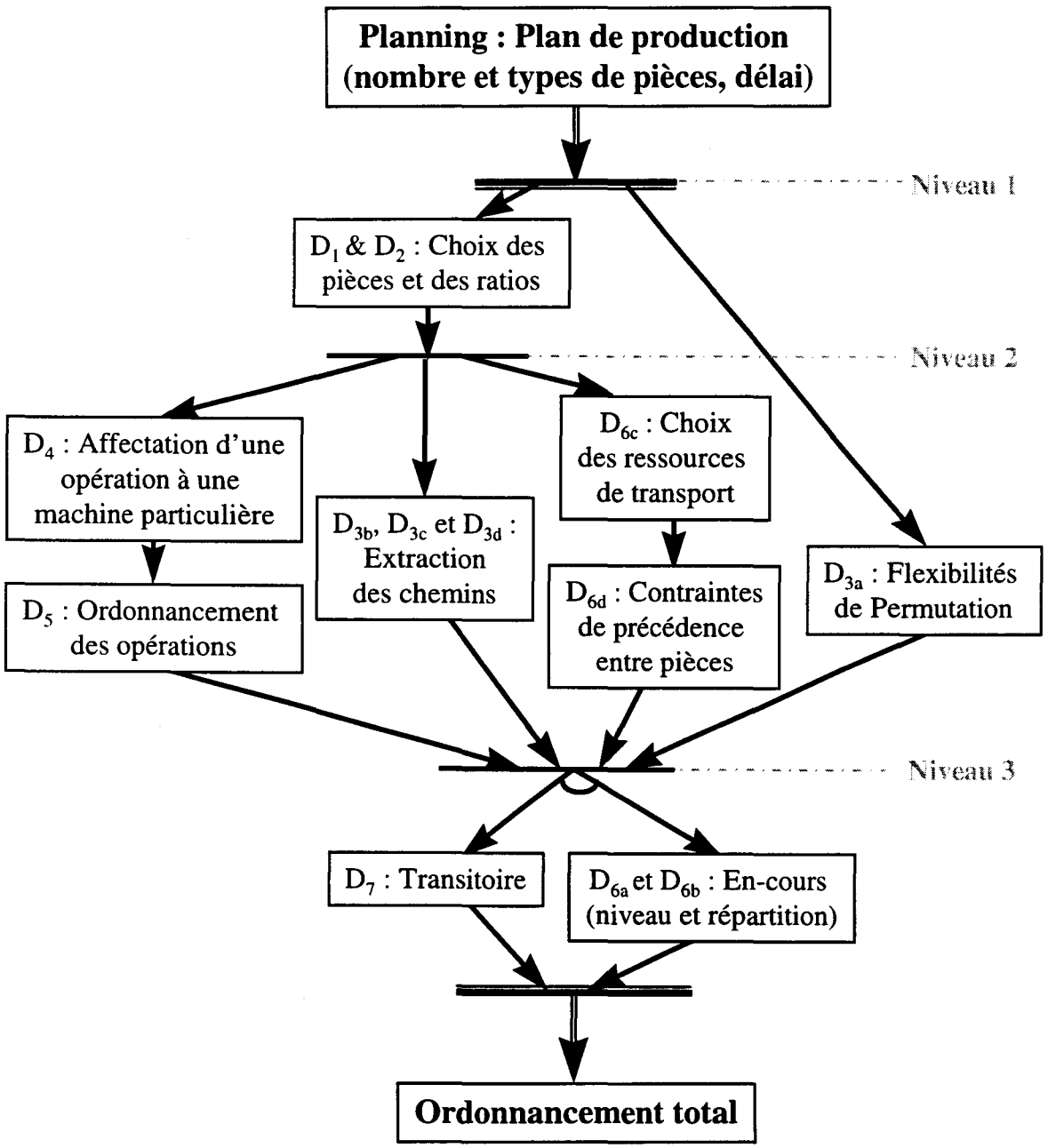


Figure I-5 : Contraintes de précédence implicites entre les différentes classes de décision

•le choix des ressources de transport (classe D_{6c}) suivi du choix de contraintes de précédence entre pièces (classes D_{6d}). L'ordre entre ces deux classes est strict et la deuxième opération ne peut être réalisée avant la première.

Ces trois points peuvent être menés en parallèle, cependant ils doivent être tous achevés (niveau 3) avant de pouvoir prendre en compte l'en-cours (classes D_{6a} et D_{6b}) et le transitoire (classe D_7). Ces deux derniers problèmes peuvent également être étudiés parallèlement sachant que s'ils sont hiérarchisés, le résultat de l'un influence considérablement le comportement du second. Notons que la résolution des flexibilités de permutation peut être effectuée indépendamment des autres entre le niveau 1 et le niveau 3.

I.3.2 Choix d'une démarche de résolution

I.3.2.1 Objectifs et Compromis

Nous cherchons donc à étudier, analyser et optimiser, selon des critères qui seront donnés dans § I.3.3, la commande prévisionnelle d'un Atelier Flexible régie par une demande stricte en nombre et type de pièces avec un impératif de production sous forme de délai de livraison. L'ampleur de la complexité du problème est telle qu'il nous est impossible d'envisager une résolution simultanée de toutes les classes de décision.

Si l'on prend par exemple le cas des classes de décision D_5, D_{6a} et D_{6b}, le problème (communément appelé problème d'ordonnancement) de leur résolution est, dans le cas général, NP-difficile, et il en est de même pour la plupart des autres degrés de liberté. La décomposition de la résolution du problème en plusieurs phases ou étapes successives s'impose nécessairement pour tenter d'optimiser réellement les différents choix potentiels qui résultent de la flexibilité du système. En fait, nous sommes confrontés à deux difficultés contradictoires pour la hiérarchisation de la résolution. La première étant de décomposer le moins possible la résolution afin d'éviter l'utilisation répétée d'heuristiques. La seconde étant, à l'inverse, de décomposer le plus possible afin d'avoir des sous-problèmes « faciles » à résoudre. Il s'agit donc de trouver un bon compromis entre ces deux démarches.

L'avantage d'une linéarisation de la résolution réside dans le fait qu'elle permet une étude complète de chaque phase : complexité, nombre de solutions admissibles ... De plus, chaque fois que cela s'imposera, nous pourrons faire des « concessions » en limitant le

nombre des solutions étudiées ou en utilisant des heuristiques mais à condition de pouvoir se situer par rapport à l'optimum. C'est pour cette raison que nous procéderons à des calculs de bornes en fin de chaque phase qui permettront d'évaluer les solutions trouvées.

I.3.2.2 Problème d'Ordonnancement

Le sous-problème d'ordonnancement (classes D_5, D_{6a} et D_{6b} réunies) ne peut être optimisé d'une façon générale([1]). En effet, dans la littérature, seuls des cas particuliers, tels que le cas du *flow-shop à deux machines* [JOH 54] et [HAP 95], ont été étudiés d'une manière optimale et ceci en raison de la combinatoire et de la complexité du problème. De plus, puisque nous cherchons à établir une commande évoluant à débit maximal, la solution qui consisterait à évaluer un ordonnancement global s'adaptant aux variations de la production ou aux défaillances, ne peut en aucun cas être envisagée.

C'est pour cette raison que nous avons choisi a priori l'ordonnancement cyclique qui consiste à décomposer la production en plusieurs petites productions ce qui permet d'ordonnancer un nombre limité d'opérations dans une fenêtre temporelle réduite et donc de réduire considérablement la combinatoire. Ce choix peut s'avérer contraignant quant à la possibilité d'obtenir la solution optimale([2]), si elle est atteignable([3]). Nous avons résumé Figure I-6 les principaux résultats([4]) que nous pouvons trouver dans la littérature concernant le problème de la détermination de la commande des SFPM.

La commande en-ligne se distingue par sa robustesse face à d'éventuelles perturbations. Cependant, il est difficile de parler d'optimalité du point de vue flux de matière puisque celui-ci n'est connu qu'une fois la production achevée. De plus, ce genre de commande utilise généralement des règles de décision locales « myopes » par rapport au reste du système ce qui peut affecter les performances du système en terme de flux.

---

[1] Par ce terme nous entendons une résolution sans aucune contrainte en dehors des critères de performances

[2] Même si nous optimisons l'ordonnancement dans la fenêtre temporelle, la somme de ces fenêtres n'est pas optimale : « *la somme de sous chemins optimaux n'est pas forcément un chemin optimal* » d'après le théorème de Bellmann, cf. [BEL 65]

[3] Etant donné le caractère NP-difficile du problème, l'optimum peut nécessiter un temps infini de calcul, il est donc dit non atteignable.

[4] Cette liste n'est pas exhaustive.

Concernant la commande hors ligne, nous pouvons distinguer deux courants de pensées distincts par leurs hypothèses, leurs critères de performances et les typologies des systèmes concernés. La première possibilité pour la détermination des commandes hors-ligne est l'ordonnancement acyclique. Elle s'intéresse particulièrement aux typologies extrêmes tels que le flow-shop et les Graphes d'Evénements (pas de flexibilités opératoires : classe de décision D_3) ou encore l'open-shop (flexibilité opératoire totale) avec les algorithmes génétiques. Ce genre de commande est bien adapté aux petites demandes où on peut considérer la totalité des tâches à réaliser pour effectuer une résolution globale en prenant en compte la totalité des opérations.

I.3.2.3 Ordonnancement Cyclique

La deuxième possibilité réside dans l'ordonnancement cyclique adapté aux moyennes demandes, où le nombre de tâches est assez grand pour éviter une résolution globale, au profit d'une décomposition en petites quantités identiques. La commande ainsi optimisée est répétée autant de fois que nécessaire pour réaliser la demande totale. Cependant, nous montrerons que ce type d'ordonnancement est également adapté aux petites demandes, cf. Chapitre IV.

Parmi les commandes cycliques nous distinguons celles qui sont K-périodiques, cf. [CHR 85], [MUN 91] et [HAN 94]. Ces travaux se sont particulièrement intéressés aux systèmes d'information avec préemption d'opérations ce qui n'est pas le cas des systèmes de production flexible en général. La K-périodicité nécessite généralement un nombre de pièces à produire par période (appelé *horizon de production*) assez élevé ce qui augmente la complexité du problème. De plus, le facteur K est initialement non maîtrisé puisqu'on ne connaît qu'un majorant sous forme de p.p.c.m. des différents nombres d'exemplaires des ressources de transformation. Nous nous intéressons donc aux commandes hors-ligne du type 1-périodique par rapport aux machines : c'est-à-dire qu'à chaque cycle, chaque machine effectuera la même séquence d'opérations.

Nous avons représenté Figure I-6 les principaux résultats concernant la commande des SFPM : nous avons différencié les travaux menés au sein du LAIL (rectangles arrondis) ainsi que les orientations que nous avons choisies pour notre étude (en trais épais et en **Gras**). Notons que la partie relevant de l'étude des régimes transitoires sera introduite dans ce mémoire au Chapitre IV.

Ainsi, parmi les ordonnancements 1-cycliques les plus connus nous pouvons citer :

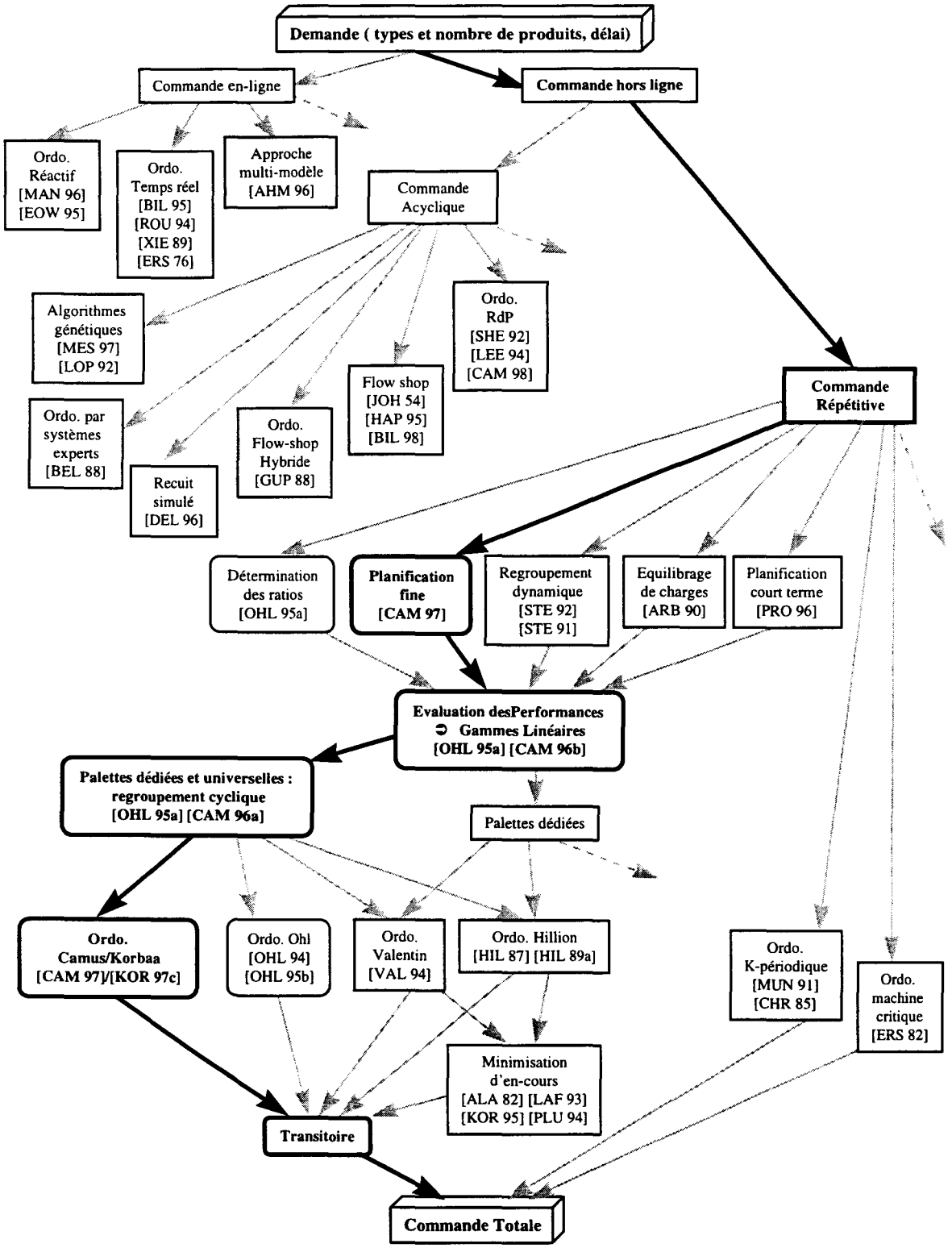


Figure I-6 : Principaux résultats bibliographiques sur la Commande des SFPM

•Ordonnancement de la machine critique [ERS 82] : cette méthode se base sur la minimisation des tailles des buffers des machines comme critère principal et ne traite pas la minimisation de l'en-cours.

•Ordonnancement 1-périodique [HIL 87] : cette méthode dissocie les deux classes de décision D_5 et D_6 en opérant, souvent, en deux phases. Dans un premier temps, l'algorithme de Hillion cherche à résoudre le problème complet de l'ordonnancement des opérations sur les machines et à déterminer le nombre d'en-cours nécessaire ainsi que sa position. Le placement des opérations est progressif (une opération à la fois).

Cependant, il arrive que cet ordonnancement ne respecte pas le temps de cycle optimal préfixé. Dans ce cas, ce premier algorithme continue d'ordonnancer sans respect de la vitesse maximale. Ensuite, on passe le relais à une deuxième routine qui se chargera, à partir du Graphe d'Evénement ainsi déterminé, d'éliminer le marquage du graphe pour ne garder que le séquencement des opérations sur les machines et d'en calculer un nouveau qui respecte le temps de cycle optimal.

•Ordonnancement 1-périodique [VAL 94] : cette méthode utilise le même principe que celle de Hillion tout en améliorant les performances. Cette amélioration provient de l'introduction de la possibilité de récupération de la marge de machines non consommée ultérieurement.

•Ordonnancement par cycles disjoints [OHL 95b] : cette méthode rompt avec les deux précédentes en proposant un algorithme de **calcul d'ordonnancement admissible en une exécution du programme**. Cette méthode place toutes les opérations dans une fenêtre temporelle figée et n'autorise pas à une opération à commencer dans un cycle et à terminer dans le suivant.

•Ordonnancement par chevauchement de cycles [KOR 97a] : cette méthode part du même principe que la précédente : placement progressif des opérations, recherche par faisceaux avec profondeur réglable et ordonnancement admissible en une exécution de l'algorithme. L'apport principal réside dans le fait que, désormais, une opération peut démarrer dans un cycle et se terminer au cycle suivant (on parle alors de chevauchement de cycles). C'est la méthode que retiendrons dans ce mémoire, cf. chapitre III.

I.3.2.4 Solution Adoptée : Hiérarchisation de la résolution

Nous avons donc opté pour l'approche progressive qui consiste en une hiérarchisation suivie d'une résolution linéaire en sept phases des différentes étapes, cf. Figure I-7et Figure I-8. Durant chacune de ces phases, on cherchera à optimiser un ou plusieurs degrés de liberté pour construire l'ensemble des solutions admissibles qui formeront les données de l'étape suivante. Les choix potentiels ultérieurs seront préservés systématiquement afin de maintenir la possibilité d'une optimisation effective.

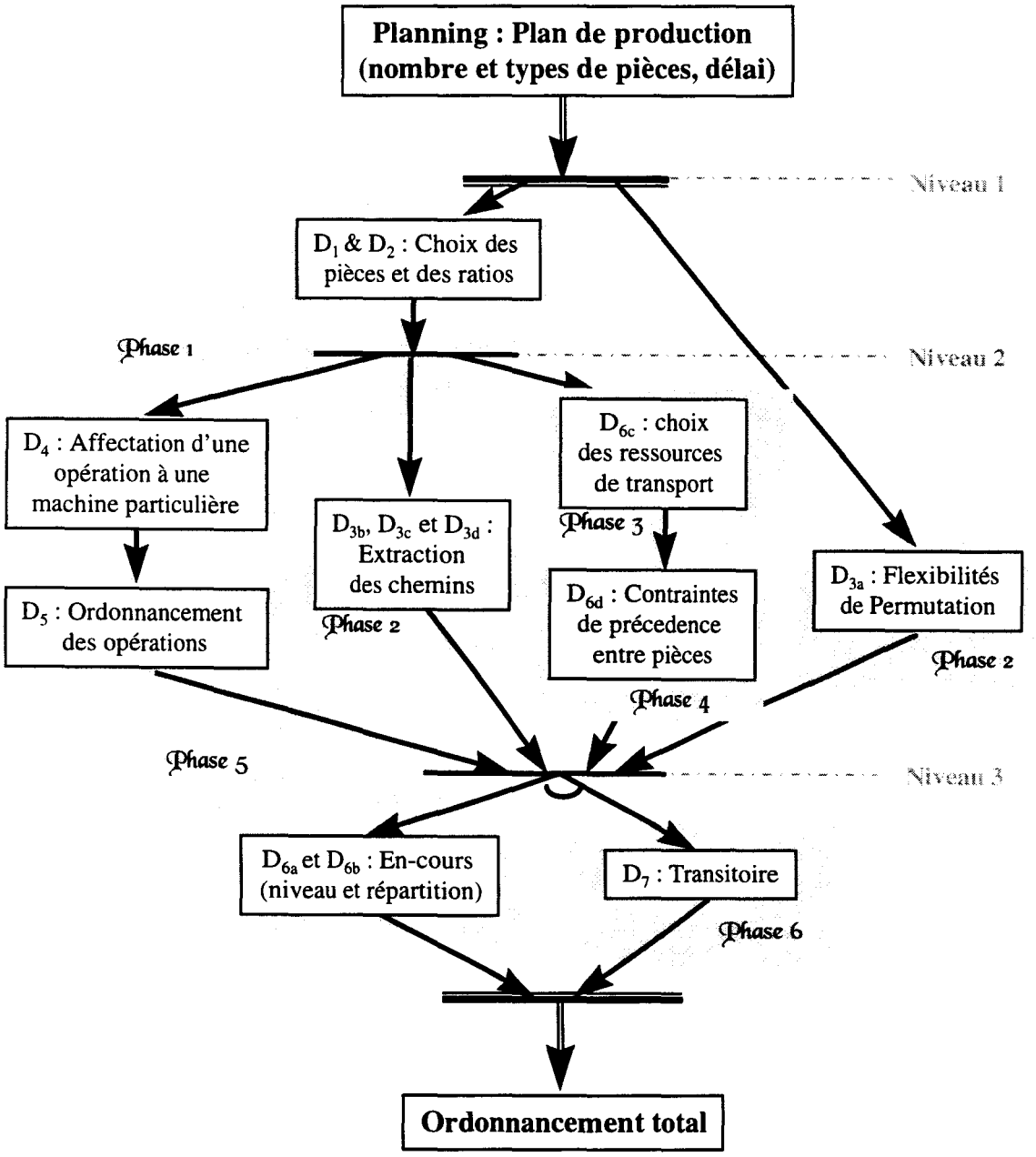


Figure I-7 : Décomposition en phases des degrés de flexibilités

Ainsi, on résout une à une les flexibilités pour construire progressivement le graphe de commande. Nous présentons, Figure I-8, les différentes phases de résolution avec les classes de décision correspondantes, les flexibilités étudiées, les caractéristiques et le modèle associé. C'est sur cette base que nous formulerons les améliorations complémentaires ad hoc.

La première phase, appelée planification fine, cf. § I.4.1, concerne les classes de décision D_1, D_2, et D_4. Elle s'intéresse à la détermination des pièces qui seront produites simultanément ainsi que les nombres d'exemplaires demandés (il s'agit là de découper la production globale en plusieurs régimes permanents cycliques).

De plus, elle introduit directement le caractère discret du problème en calculant directement des ratios de production entiers et en affectant par la même occasion les opérations aux différentes machines. Ensuite, pour chaque régime permanent, nous appliquerons les six phases restantes. Un régime permanent donné, résultant de la phase de planification fine, est donc caractérisé par les données suivantes : les types de pièces à réaliser simultanément, le nombre de pièces à produire par cycle (horizon de production), les ratios de routage entiers et les machines distinctes ([1]).

La phase 2 se charge de lever les indéterminismes dus aux flexibilités opératoires (classe de décision D_3) par le dépliage des flexibilités complexes et le choix des branches empruntées pour la production des différentes pièces. Les phases 3 et 4 introduisent les ressources de transport.

Nous connaissons déjà les types de palettes que nous allons utiliser (universelles ou dédiées), on distingue alors les ensembles de palettes selon les pièces qu'elles auront à transporter (partitions de gammes). Les pièces auront un ordre préfixé de passage sur les palettes, cet ordre est déterminé par la phase 4.

Durant la phase 5, nous nous intéresserons au problème d'ordonnancement proprement dit pour établir la commande du régime permanent respectant le débit maximal avec le minimum de palettes.

[1] Comme nous avons déjà affecté les opérations aux différentes machines, nous pouvons les distinguer individuellement

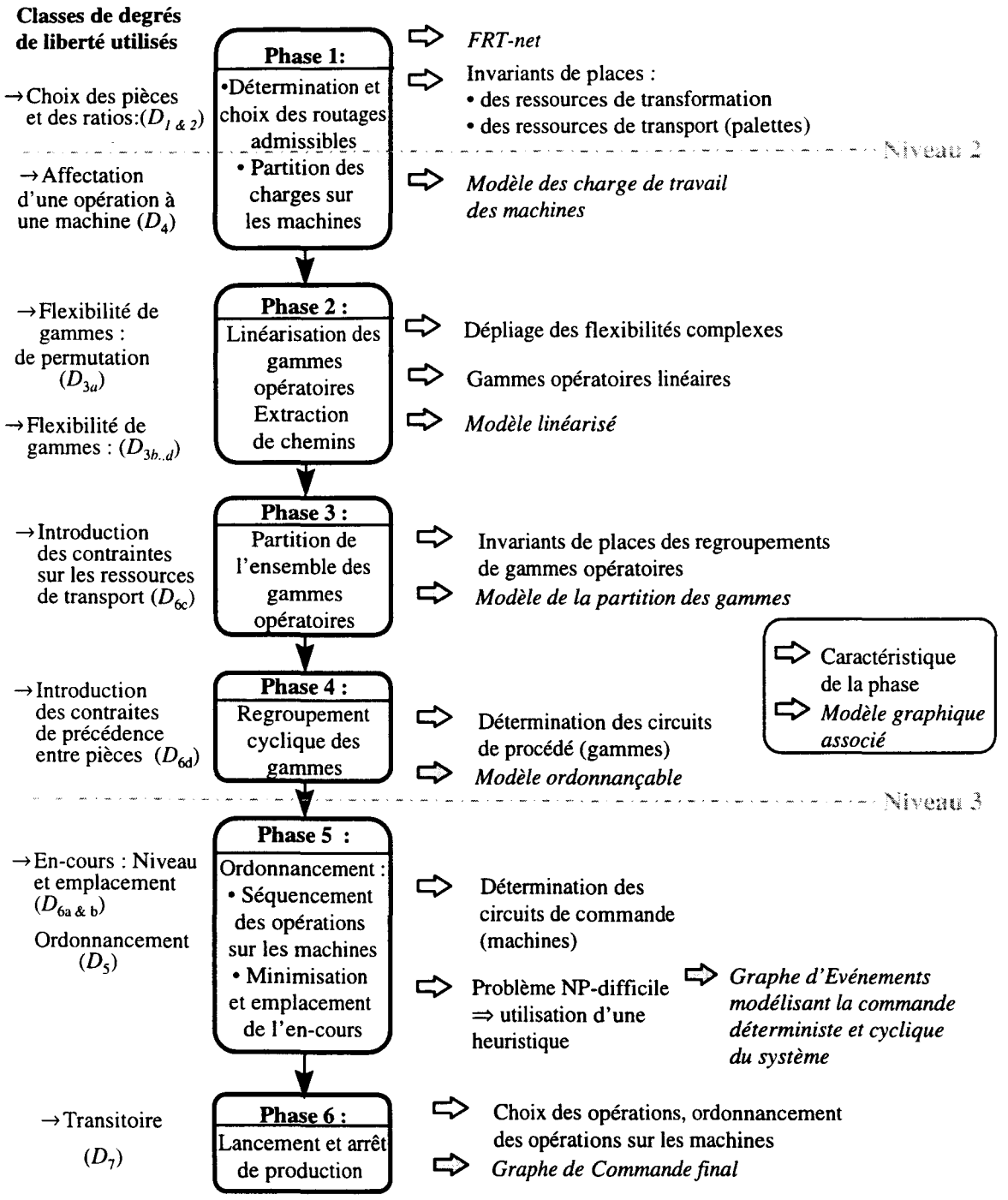


Figure I-8 : Détails, phase par phase, de l'approche structurée adoptée

Enfin, la phase 6 terminera l'étude en déterminant les régimes transitoires nécessaires pour le lancement et l'arrêt de la production ainsi que pour la transition entre deux régimes permanents successifs. Cette phase aura pour objectif de considérer les régimes permanents établis comme données et de minimiser les transitions entre deux régimes de fonctionnement.

I.3.3 Critères de Performance et Outils

La résolution de la commande se base sur des critères de performance que nous proposons de caractériser par leur nature « quantitative ou qualitative ».

I.3.3.1 Critères quantitatifs :

- domaine de la productivité : **minimisation du temps total de production** (appelé également **makespan**), ou la maximisation du flux de production,
- domaine mixte économique et productif : **minimisation de l'en-cours du système** (appelé Work In Process),
- point de vue économique : il existe deux types de coûts liés à l'en-cours :
  - coût de conception et de fabrication des supports de bridage des pièces appartenant aux différents types de palettes. Ce coût est lié directement à la **minimisation de l'en-cours maximal du système**.
  - coût théorique d'immobilisation d'un produit brut, semi-fini et fini. Ce coût est dû au stockage des produits. Nous ne prenons en compte ici que celui dû aux produits semi-fini (stockage entre sous systèmes de la production globale ou immobilisation dans les buffers de notre atelier). Ce coût est donc fonction de la **minimisation de l'en-cours moyen du système**.

I.3.3.2 critères qualitatifs :

- **simplicité de la détermination de la commande**. Il faut à la fois que le temps de calcul de la commande prévisionnelle soit **rapide** (contraintes strictes en phase d'exploitation) et que la commande soit **facile à déterminer** et à implanter.
- **robustesse de la commande** vis-à-vis des écarts possibles de temps opératoires entre les durées déterministes retenues par la commande prévisionnelle et le fonctionnement réel du système (utilisation des marges de gammes).
- **réactivité de la commande** face à des perturbations propres au système de production, telles que des défaillances des ressources de l'atelier et des

perturbations au niveau de la production, en considérant, par exemple, l'arrivée de commandes non planifiées jugées prioritaires. Il faut que la production puisse réagir rapidement face à de tels événements.

- facilité d'implantation de la commande (élaboration d'une modélisation de commande directement implantable, par exemple).

- minimisation du coût d'implantation hardware (capteurs, calculateurs), software (programmes), personnel (besoin de personnes qualifiées, coût de la formation) : installation, mise en œuvre, maintenance. Pour ce facteur économique, il est difficile de maîtriser et d'évaluer de manière correcte le coût de l'implantation. C'est pour cela que nous avons décidé de placer cet objectif dans les critères qualitatifs. Pour l'instant, cette question n'a pas encore été abordée à notre niveau.

- compatibilité de la production au niveau de l'atelier flexible avec la gestion des stocks et les types de gestion respectifs en amont et en aval de la production considérée ici.

Les trois premiers critères qualitatifs nous confortent dans le choix d'un régime permanent cyclique pour deux raisons principales :

1. la combinatoire résultant de l'ensemble des choix de décisions possibles dues à la flexibilité opératoire reste limitée et dénombrable,

2. la possibilité d'arrêter la production fréquemment, à chaque fin de cycle, pour introduire des commandes urgentes ou pour contrôler le bon fonctionnement de la production, cf. Chapitre III.

Certains de ces critères sont contradictoires (par exemple : la recherche de l'optimum du makespan et la simplicité de la commande, ou encore la minimisation de l'en-cours et du temps de calcul ...). La solution sera donc un compromis difficile à quantifier.

I.3.4 Modélisation

En ce qui concerne la modélisation des différents problèmes abordés, nous avons opté pour les Réseaux de Petri (RdP) plutôt que pour les Réseaux Files d'Attente (RFA), fréquemment utilisés à travers la littérature, dans l'hypothèse stochastique. Ce choix est justifié par maintes raisons telles que la simplicité et l'efficacité de cet outil graphique, la puissance du bagage mathématique associé, l'existence d'interfaces avec d'autres outils de modélisation et enfin l'intégration au sein du projet CASPAIM.

Le projet CASPAIM s'intéresse aussi bien à la conception, qu'à l'analyse des performances ou à l'implémentation. Il parait donc nécessaire de disposer d'une référence de modélisation pour ces différentes parties. Les Réseaux de Petri, outil générique de modélisation des Systèmes à Evénements Discrets (SED), cf. [SIL 96a et b], sont très bien adaptés aux problématiques qui nous concernent et permettent d'avoir des points de vue, plus ou moins détaillés, sur le système selon les besoins.

En effet, les RdP permettent de modéliser le comportement statique du système (vue macroscopique : conception) aussi bien que le comportement dynamique (microscopique : possibilité de distinguer l'évolution de la plus petite composante du système) des produits et des ressources. Grâce à cet outil, certaines notions, telles que les ressources de transfert, de transformation et de stockage, deviennent implicites et facilement représentables. Il en est de même pour la notion de dualité entre les produits (pièces) et les ressources (machines), développée dans CASPAIM II. En ce qui concerne les modèles que nous utilisons, nous nous basons sur les modèles et les terminologies de [CRU 91] et [AMA 94] principalement pour la modélisation progressive des gammes.

Ajoutons que tous les calculs d'analyse de flux réalisables avec les RFA, cf. [FDI 89], [BOU 93b], [BOU 96] et [DIM 93], sont tout aussi possibles avec les RdP Stochastiques, cf. [FLO 85], [MAR 82] et [DEA 93]. De plus, les RdP permettent de représenter, d'une manière claire et assez simple, avec le même formalisme, tout aussi bien les propriétés mathématiques que l'évolution des différentes composantes du système et par conséquent, l'état du système de production en régime dynamique, cf. [RAM 80], [SIF 80], [DIC 93], [GOL 86], [SIL 87], [ALA 89], [SIL 89], [DUB 90], [DAV 92], [PRO 95], [KHA 96] et [GIU 96].

Les Réseaux de Petri permettent également de prendre en compte les problèmes de synchronisation([1]), d'asynchronisme, de communication ... Ils permettent enfin d'en analyser les propriétés fondamentales (vivacité, finitude, réversibilité ...) et aussi d'évaluer leur comportement temporel. En fait, les Réseaux de Petri sont pour les systèmes à événements discrets ce que sont les équations différentielles pour les systèmes continus : *un outil simple et efficace pour modéliser le comportement d'un système disposant notamment d'un énorme potentiel de résolution grâce à l'outil mathématique associé.*

Pour l'intégration de toutes ses composantes, le projet CASPAIM a mis en place la composante *Référentiel*, qui est une structure d'accueil pour les modèles des systèmes utilisés ou générés par chacune des composantes du projet. Le but est d'assurer, pendant la conceptions des modèles, la cohérence des versions et de gérer les liens de dépendance entre les modèles([2]).

Ce référentiel utilise, comme langage de modélisation, le *Unified Modeling Language* (UML) qui constitue un formalisme neutre et connu de chaque intervenant, cf. [BEN 93], [GAM 94], [IVA 93], [MAU 95] et [ZAY 95]. Ce langage « s'interface » facilement avec les Réseaux de Petri. Nous utiliserons également cet outil pour modéliser le Macro-fonctionnement de notre méthode c'est-à-dire les données, les paramètres et les liens entres les deux, cf. [NDI 97] et [BIG 97].

I.3.5 Notations

Nous utilisons, dans ce mémoire, les RdP ordinaires, étendus aux temporisations déterministes des transitions pour modéliser le comportement dynamique d'un atelier flexible. Nous avons développé une approche de modélisation qui transforme progressivement un Réseau de Petri, modélisant le système de production avec toutes ses flexibilités, en un modèle, entièrement déterministe, de la commande élaborée. Nous supposons que le lecteur est familier avec les principales propriétés des RdP, cf. [RAM 80], [MUR 89] et [DIC 93]. Nous introduisons les notations suivantes : Soit N un Réseau de Petri. Il est caractérisé par $N =< P, T, W^-, W^+ >$, où :

[1] L'assemblage ou les ressources imbriquées par exemple

[2] Appelée aussi Méta-Modélisation

$P = \{P_1, P_2, ..., P_n\}$ représente l'ensemble des places,

$T = \{t_1, t_2, ..., t_m\}$ représente l'ensemble des transitions du graphe,

$W = W^+ - W^-$ est la matrice d'incidence du réseau.

$M(P_i)$ indique le marquage de la place P_i et $M_0(P_i)$ est le marquage initial de la place P_i.

Soit σ une séquence de tirs de transitions. $\bar{\sigma}$ est le vecteur caractéristique (de Parikh) de σ, dont la i-ème composante correspond au nombre d'occurrences de la transition t_i dans la séquence σ.

$R(N, M_0)$ est l'ensemble des marquages accessibles des places du RdP N à partir du marquage initial M_0 et $L(N, M_0)$ est l'ensemble des séquences de transitions tirables à partir de M_0.

Soit $\sigma_n \in L(N, M_0)$ et $M_{\sigma_n} \in R(N, M_O)$ le marquage obtenu après le tir de la séquence σ_n. Nous avons alors $M_{\sigma_n} = M_0 + W.\bar{\sigma}_n$, équation d'évolution du réseau appelée parfois équation d'état et noté $M_0[\sigma_n > M_{\sigma_n}$.

Y est appelé P-semiflot du RdP N, si et seulement si (ssi) $Y^T.W = 0$, $Y \neq 0$ et $Y \in IN^n$, c'est-à-dire que Y exprime la conservation de la somme (éventuellement pondérée) des marques présentes dans ce P-semiflot :

$$\forall M \in R(N, M_0),\ Y^T.M = Y^T.M_0 + Y^T.W.\bar{\sigma} = Y^T.M_0.$$

De manière semblable, X est appelé T-semiflot du RdP N, si et seulement si $W.X = 0$, $X \neq 0$ et $X \in IN^m$, c'est-à-dire que X est une composante consistante du RdP : $M_X = M_0 + W.X = M_0$.

Finalement, nous rappelons que nous utilisons les Réseaux de Petri, à la fois comme outil de modélisation mais aussi pour aider à la compréhension et à la résolution des différents problèmes.

I.4 Planification fine

Comme nous l'avons vu précédemment, cf. § I.3.2.4, nous avons opté pour une hiérarchisation de la méthode et un découpage en six phases de résolution. Nous allons maintenant présenter la première étape, appelée planification fine, qui a pour rôle de découper une demande initiale en plusieurs régimes permanents afin d'optimiser les critères qui seront énoncés.

Cette méthode a été mise en œuvre pour résoudre les problèmes de planification, cf. [CAM 97]. En effet, ce dernier propose d'optimiser le flux statique du système (considérer la demande en pourcentage de pièce et calculer le flux maximal théorique qu'on peut atteindre) pour trouver ensuite l'horizon de production (nombre de pièces à produire par cycle) discret qui s'approche le plus de cette performance.

Le problème majeur d'une telle méthode réside dans le fait que, très souvent, l'horizon nécessaire, pour approcher les performances théoriques, est grand. Ceci nous oblige à avoir des temps de cycles conséquents, ce qui conduit à diminuer la réactivité et la souplesse de la commande. En effet, la réactivité de la commande peut être interprétée par le fait de pouvoir détecter et réagir rapidement à une défaillance ou une dégradation de fonctionnement. Cette information est disponible une fois tous les temps de cycles puisqu'on est capable d'observer la sortie ou non d'une pièce finie aux dates imparties. D'autre part, la réactivité de la commande peut également être traduite par le fait de pouvoir arrêter la production et vider le système facilement pour répondre à une demande prioritaire, on peut alors parler de souplesse de la commande.

Nous verrons dans le Chapitre IV que ceci est facilement réalisable au début de chaque période. Dans ces deux cas, une augmentation sensible du temps de cycle augmente les délais d'attente pour observer la sortie des pièces ou pour pouvoir arrêter la production et par conséquent diminue la réactivité du système. Pour ces raisons, la méthode élaborée propose de découper la demande en plusieurs productions de petites tailles (horizon de production) dont on connaît la performance réelle puisqu'on étudiera le système directement dans son aspect discret.

I.4.1 Formulation du problème de Planification fine

Problème : Trouver le meilleur moyen de réaliser une production, en un ou plusieurs régimes permanents, afin de respecter les contraintes imposées et optimiser les critères de performance considérés.

Classes de Décision associées : D_1 et D_2 et D_4.

Nous supposons disposer d'un programme prévisionnel de production pour lequel une demande figée, représentée par des quantités absolues de pièces, est à réaliser dans un certain délai. En fait, les quantités sont calculées par le niveau *gestion de production*, hiérarchiquement supérieur, grâce à MRP II (Manufacturing Resource Planning) par exemple. De plus, les délais sont supposés suffisants pour la réalisation de la production.

Plusieurs stratégies de production peuvent ici être envisagées selon qu'on veuille ou non utiliser les flexibilités du système de production en dynamique ou de façon statique par exemple en figeant la configuration de l'atelier pour chacun des produits traités séparément. Les possibilités de production dont nous disposons sont donc les suivantes :

1. Produire tous les types de produits simultanément (en respectant les ratios de production initiaux),
2. Produire les types de pièces en séquence mais de façon unitaire,
3. Réaliser la commande en plusieurs fois avec, à chaque régime, une production différente en types et/ou en nombre de pièces.

C'est pourquoi nous devons mettre en place une étape de planification fine pour l'optimisation de ces productions selon les critères que nous allons fixer. En effet, étant donné la complexité des critères de performances énoncés dans § I.3.3, nous ne pouvons envisager la prise en compte de tous les critères simultanément. Ainsi, le critère de minimisation de l'en-cours est différé à la fin d'élaboration de la commande pour ne pas gêner l'optimisation simultanée des autres critères. Pour cette étape, nous supposerons donc disposer d'un en-cours suffisant et de ressources de transport associées infiniment rapides et nous reporterons à une phase ultérieure l'optimisation de l'en-cours et des décisions de routage.

Les notations et variables nécessaires à la formulation du problème sont définies ci-après :

$T(I)$ quantité de pièces de type i à réaliser, donnée fournie par la gestion de production, $i \in \{1, ..., I_{max}\}$.

MP^* temps nécessaire pour réaliser la production demandée (makespan).

RP_{max} **nombre de régimes permanents** à considérer pour réaliser la production.

I_{max} nombre de types différents de pièces à fabriquer.

RP_p p-ème régime permanent, $p \in \{1, ..., RP_{max}\}$. Il est caractérisé par :

$E(RP_p)$ horizon cyclique discret de travail ou ensemble de produits à réaliser au cours d'un cycle de fabrication. Il est lui même caractérisé par :

$I(p)$ nombre de pièces de type i à réaliser, $i \in \{1, ..., I_{max}\}$.

N_{max} nombre maximum de pièces à réaliser en un cycle de fabrication

N_{I-max} ou $N_{I-max}(p)$ nombre maximum de pièces de type i à produire sur n'importe quel régime permanent ou sur le p-ème régime.

$E(RP_p)$ peut donc se mettre sous la forme : $E(RP_p) = \{I(p).I, \forall i \in \{1, ..., I_{max}\}\}$

$CT(RP_p)$ Temps de cycle optimal associé à cet horizon $E(RP_p)$.

$X(p)$ nombre de répétitions de l'horizon $E(RP_p)$ pendant le régime permanent RP_p

NC_{min} nombre minimum de répétitions de cycles pour considérer un fonctionnement réel cyclique de n'importe quel régime permanent

$I_{jk}(p)$ ratio de routage entier associé à la k-ème branche du j-ème conflit du i-ème type de pièces du p-ème régime permanent.

$O_{t-r}(p)$ nombre d'opérations de durée t affectées à la ressource multiple r, $\forall p \in \{1, ..., RP_{max}\}$, $\forall r \in \{1, ..., R\}$, $\forall t$.

$O_{t-r.m}(p)$ nombre d'opérations de durée t affectées à la m-ème machine de la r-ème ressource, $m \in \{1, ..., M_r\}$

$Z_{r-m}(p)$ charge de travail affectée à la m-ème machine de la r-ème ressource du p-ème régime permanent. Nous pouvons la définir ainsi : $Z_{r-m}(p) = \sum_{\forall t} t.O_{t-r.m}(p)$.

C'est la machine la plus chargée, pour un régime permanent donné, qui sera la machine menante. Elle fixera la vitesse de production du système et le temps de cycle optimal. A flux maximum de production, cette machine fonctionnera à saturation.

C nombre d'outils

e_{cr} nombre d'emplacements pour un outil de type c sur une machine de type r

e_r nombre d'emplacements d'un magasin d'outils d'une machine de type r

v_{ijkcr} = 1, si l'usinage d'une opération de la k-ème branche du j-ème conflit des gammes de produits de type i requiert sur une machine de la ressource r un outil de type c,

= 0 sinon,

$b_{ijk}(p)$ = 1, si $I_{jk}(p) \neq 0$, $b_{ijk}(p)$ indique pour le régime permanent considéré RP_p si au moins une séquence d'opérations de la branche associée à ce ratio entier de routage $I_{jk}(p)$ a été retenue

= 0, sinon

$y_{cr}(p)$ = 1, si l'outil de type c est chargé sur les machines de la ressource r,

= 0, sinon

Les critères que nous utilisons dans cette optimisation sont les suivants :

- **minimisation de la durée totale de la production** à réaliser à court terme **(makespan)** : ce sera notre critère principal permettant d'assurer le respect du délai de production, conforter l'hypothèse de pannes rares, se réserver une marge de temps qui peut servir pour faire face à des perturbations non prévues, assurer une opération de maintenance, réaliser des commandes prioritaires, ...

Le makespan d'une production est la somme des durées des régimes permanents et des régimes transitoires associés. Notons que nous ne considérerons la détermination du régime transitoire qu'après avoir résolu l'optimisation des régimes permanents périodiques. Dans ces conditions, afin de ne pas alourdir la formulation, nous n'allons pas considérer les bornes exprimant la durée du régime transitoire. Le critère s'écrit alors sous la forme suivante :

$$\text{minimiser } MP^* \text{ avec}$$

$$MP^* = \sum_{p=1}^{RP_{max}} X(p).CT(RP_p) + \begin{array}{l}\text{estimation d'une borne supérieure du temps} \\ \text{de production des pièces restant à réaliser} \\ \text{en régimes transitoires}\end{array} \qquad \text{C I - 1.1}$$

Equation I-1

le deuxième terme est une estimation du temps nécessaire à la production de pièces hors régimes permanents et leurs transitoires associés([1]), cf. Equation I-4.

- **minimisation du nombre de régimes permanents** à mettre en œuvre pour limiter d'une part la durée de tous les régimes de configuration et d'autre part le nombre des régimes transitoires pendant lesquels le système ne fonctionne pas à flux optimal.

  Ceci permettra de valider l'hypothèse de prépondérance du régime permanent. Ce nombre est contraint par le nombre max. de pièces à produire :

$$\text{minimiser } RP_{max} \text{ tel que}$$

[1] Les régimes permanents, ainsi que leurs transitoires associés, peuvent ne pas couvrir la totalité de la production demandée, on prévoit alors de les réaliser durant un transitoire, dit hors fonctionnement cyclique ou également transitoire isolé. Nous chercherons bien entendu à limiter ce régime transitoire particulier.

$$RP_{\max} \in IN \text{ et } RP_{\max} \leq I_{\max}$$

Equation I-2

On cherche à limiter le nombre de régimes permanents étant donné qu'il est difficile de maîtriser les transitoires. Par ailleurs, on connaît une solution évidente pour $RP_{\max} = I_{\max}$ (c'est la production successive de chaque produit seul dans l'atelier). Il n'est donc pas nécessaire de chercher des productions de plus de $I_{\max}$ régimes permanents.

- **minimisation de la taille des horizons cycliques** de production : cet objectif conduit à simplifier le calcul et l'implémentation de la commande cyclique pour augmenter la réactivité temporelle de la commande et à minimiser la taille des stocks de sécurité et ainsi garantir le fonctionnement cyclique des régimes permanents.

$$\forall p \in \{1, ..., RP_{\max}\}, \text{ minimiser } card\left(E\left(RP_p\right)\right) = \sum_{i=1}^{I_{\max}} I(p) \text{ tel que}$$

$$\begin{cases} \forall p \in \{1, ..., RP_{\max}\}, \forall i \in \{1, ..., I_{\max}\}, I(p) \in IN & \text{C I-3.1} \\ \text{et } \sum_{i=1}^{I_{\max}} I(p) \leq N_{\max} (= 10) & \text{C I-3.2} \\ \text{et } I(p) \leq N_{I-\max}(p) \text{ ou } N_{I-\max} & \text{C I-3.3} \end{cases}$$

Equation I-3

la seconde contrainte C I-3.2 exprime une condition extrême de limitation de l'horizon (l'ensemble de pièces à produire par cycle). La troisième, C I-3.3, contraint le nombre de pièces de chaque type produits par cycle à être inférieur à une limite déjà fixée.

- **maximisation de la production de pièces pendant les régimes permanents et leurs transitoires associés** : il n'est pas toujours possible de réaliser toutes les pièces en régimes permanents (et transitoires associés). Nous gardons alors la possibilité de réaliser un nombre de pièces, à minimiser bien entendu, durant un

transitoire isolé([1]). Cette hypothèse vise à ne pas transformer notre problème en une méthode générale d'ordonnancement dont on a déjà évoqué la complexité.

$$\text{maximiser} \sum_{p=1}^{RP_{max}} X(p).\left(\sum_{i=1}^{I_{max}} I(p)\right)$$

$$\text{avec} \begin{cases} \forall p \in \{1, ..., RP_{max}\},\ X(p) \in IN & \text{C I-4.1} \\ \sum_{p=1}^{RP_{max}} X(p).\left(\sum_{i=1}^{I_{max}} I(p)\right) \geq \left\lceil \frac{99,5}{100}.\left(\sum_{i=1}^{I_{max}} T(I)\right)\right\rceil & \text{C I-4.2} \\ \sum_{p=1}^{RP_{max}} X(p).I(p) \leq T(I),\ \forall i \in \{1, ..., I_{max}\} & \text{C I-4.3} \end{cases}$$

Equation I-4

Pour les applications numériques nous considérerons admissible la production de moins de 0,5% de pièces durant ces transitoires isolés, d'où le terme : $\frac{99,5}{100}$ dans la contrainte C I-4.2.

- **maximisation du nombre de répétitions de chacune des commandes cycliques** au cours d'un régime permanent : ce critère garantit le fonctionnement cyclique des régimes permanents et permet un contrôle des flux de production périodique et systématique.

$$\forall p \in \{1, ..., RP_{max}\},\ \text{maximiser}\ X(p) \quad \text{avec}$$

$$X(p) \neq 0 \text{ et } X(p) \geq NC_{min} (= 10)$$

Equation I-5

Les critères précédemment énoncés concernent exclusivement les régimes permanents (durées, nombre, tailles ...). Ceux que nous introduisons maintenant concernent des problèmes plus spécifiques tels que la minimisation des charges des machines, la prise en compte du caractère discret du problème et la gestion des outils.

[1] Nous appelons ce régime transitoire « isolé » parce qu'il ne correspond à aucun régime permanent.

- **minimisation du temps de cycle** pour chaque régime permanent : ce critère est en fait dépendant de la contrainte C I.1.1 et traduit la volonté de travailler à flux maximal et donc à imposer la saturation des machines dites critiques([1]).

$$\forall p \in \{1, ..., RP_{max}\}, \text{ minimiser } CT(RP_p) \quad \text{avec}$$

$$\forall p \in \{1, ..., RP_{max}\}, \forall r \in \{1, ..., R\}, \forall m \in \{1, ..., M_r\} \; CT(RP_p) \geq Z_{r-m}(p)$$

Equation I-6

- **Respect du caractère discret du problème :** la formulation des précédents critères n'a introduit que des variables entières. Nous allons donc continuer dans cette direction en imposant que les charges des machines soient admissibles. Ce critère s'écrit sous la forme suivante :

$$\forall p \in \{1, ..., RP_{max}\}, \forall r \in \{1, ..., R\}, \forall t, \sum_{m=1}^{M_r} O_{t-r.m}(p) = O_{t-r}(p)$$

Equation I-7

dans cette relation $O_{t-r}(p)$ et $O_{t-r.m}(p)$ représentent le nombre d'opérations de durée t affectées respectivement à la *r*-ème ressource et à la *m*-ème machine de la ressource *r*, pour un cycle du régime permanent RP_p.

- Nous envisageons, pour terminer cette analyse, des critères concernant la **gestion du parc des outils**, cf. [STE 91], **et les capacités finies des magasins d'outils des machines**.

  Ce point correspond plus à une contrainte qu'à un problème d'optimisation car nous sommes limités par les contraintes d'espace et d'encombrement sur les machines.

[1] C'est la machine la plus lente du système c'est à dire celle dont la somme des temps opératoires ,des opérations qu'elle effectue (appelée également charge de la machine), est la plus grande parmi toutes les machines.

$$\forall p \in \{1, ..., RP_{\max}\},\ \forall r \in \{1, ..., R\},\ \forall c \in \{1, ..., C\},$$
$$y_{cr}(p) = \max_{\forall i, \forall j, \forall k} \left(v_{ijkcr} . b_{ijk}(p) \right)$$

Equation I-8

$$\forall p \in \{1, ..., RP_{\max}\},\ \forall r \in \{1, ..., R\},$$
$$e_r \geq \sum_{c=1}^{C} e_{cr} . y_{cr}(p)$$

Equation I-9

Si l'on considère l'ensemble de tous les critères (Equation I-1 à Equation I-9), il apparaît que le problème est **Multicritère multilinéaire à variables entières**. La complexité est telle qu'il n'est pas envisageable de trouver une solution optimale globale. Nous proposons donc de mettre en œuvre une heuristique de résolution.

I.4.2 Résolution

Nous allons chercher à déterminer une borne inférieure de référence permettant de comparer la qualité des différentes solutions. Pour calculer cette référence nous allons nous baser sur l'approche analyse de flux proposée par H. Ohl. Cette méthode met en place un seul régime permanent pour produire la totalité de la demande. Ce régime permanent respecte donc les ratios de production initiaux ($r_{produit\ I} = \frac{T(I)}{\sum_{\forall J} T(J)}$ [1]) et fonctionne à flux maximal.

Il s'agit donc de résoudre le problème en raisonnant en terme de flux continu, cf. [FRE 88], [CAM 93] et [PLU 94], et par conséquent sans prendre en compte le caractère discret du problème. La solution obtenue représente ainsi la vitesse maximale que le système pourrait atteindre pour réaliser la production demandée : c'est la borne inférieure théorique du makespan([2]). Cette borne inférieure peut être éventuellement atteinte dans les

[1] Le ratio associé à un produit *I* est égal au nombre de pièces *I* demandé sur le nombre total de pièces.

[2] Cette borne inférieure est également la borne inférieure pour le problème général d'ordonnancement.

cas où les machines sont simples et le nombre de pièces demandées ont un diviseur commun assez grand([1]). Cependant, elle n'est pas toujours atteignable dans le cas général car elle ne tient pas compte du caractère discret de la commande. Nous allons donc utiliser cette borne à titre de référence de comparaison des différentes solutions que nous allons élaborer.

La solution la plus facile à mettre en œuvre est celle qui consiste à produire les différents types de pièces d'une manière séquentielle (c'est la solution la moins coûteuse en temps de calcul). Les atouts de cette solution sont bien entendu la simplicité de la commande et les contraintes faibles voire inexistantes quant à la gestion des outils des machines. Cependant elle pose quelques problèmes relativement à la gestion des stocks amont et aval. En effet, elle oblige l'atelier amont à fournir toutes les pièces d'un même type avant de passer au type suivant, il en est de même concernant l'atelier aval. Il faut donc soit disposer d'un stock amont/aval suffisant, soit imposer le même comportement aux ateliers amont et aval.

Cette solution peut donc être considérée comme une borne max. du makespan et nous ne retiendrons que les solutions dont le temps total de production est inférieur ou égal à cette valeur. La flexibilité d'un système de production peut donc conduire à une amélioration très sensible de la productivité au prix d'une gestion performante de la complexité du système.

Plusieurs solutions sont possibles pour prendre en compte tous les critères. Nous pouvons utiliser une somme pondérée de chaque critère ou encore hiérarchiser ces critères. Il semble que la meilleure approche consiste à privilégier la minimisation du makespan (cf. Equation I-1) comme critère principal pour ne considérer les autres que comme des contraintes à satisfaire.

Il en résulte alors un problème de programmation par contraintes. Pour ces raisons H. Camus a opté pour un codage du problème en Prolog. Cette heuristique a donné jusqu'alors des résultats intéressants, nous prévoyons cependant de l'implémenter sur le logiciel ILOG Solver plus adapté à la résolution de ce genre de problèmes.

[1] Par exemple si la demande est (200 A, 300 B et 200 C) elle peut être écrite aussi sous la forme : 100*(2 A, 3 B et 2 C).

Ce problème n'est pas entièrement résolu puisqu'il reste à préciser l'ordonnancement des régimes permanents entre eux. En effet, la résolution de ce problème multilinéaire, multicritère à variables entières ne spécifie aucunement l'ordre de passage des différents régimes permanents. Cet ordre est non négligeable dans la détermination des durées des régimes transitoires. Le séquencement des productions cycliques entre elles sera donc envisagé en même temps que l'étude des régimes transitoires, cf. Chapitre IV.

I.5 Conclusion

Dans ce premier chapitre, nous avons, dans un premier temps, présenté la problématique, le contexte et les hypothèses du problème de la commande prédictive des Systèmes Flexibles de Production Manufacturière. La plus importante hypothèse que nous avons avancé concerne les pannes. En effet, l'hypothèse de pannes à occurrences rares est indispensable pour une analyse prévisionnelle du comportement du système et permet également de considérer l'existence de régimes permanents stables ayant des durées suffisamment « longues » pour envisager une méthode d'analyse et surtout d'optimisation des performances hors-ligne. Dans le cas contraire, de pannes fréquentes, une telle étude ne peut servir que de référence d'optimalité. En effet dans ce cas les méthodes de détermination de commandes en-ligne (ordonnancement temps réel, ordonnancement réactif) sont mieux adaptées.

Cependant, étant donné qu'un atelier de production n'est pas totalement déterministe, nous sommes conscients des limites potentielles d'une commande optimisée déterministe. En effet elle nécessite un niveau de **pilotage**, cf. [TAW 95], qui se chargera de veiller au bon déroulement de la production pour détecter les éventuelles pannes ou dégradations de fonctionnement. Si la défaillance est bénigne la production ordonnancée peut être maintenue avec perte de performance([1]). Dans le cas contraire il faudra procéder à la réévaluation d'une nouvelle commande tenant compte du nouveau contexte de travail déterminé par le niveau **surveillance**([2]). Dans un souci d'intégration de nos travaux avec

[1] elle se manifeste en général par une augmentation du temps de cycle et par conséquent par une diminution du flux de production.

[2] Ce niveau se charge de déterminer les machines disponibles et accessibles. C'est à partir de ces informations qu'on effectue un nouvel ordonnancement.

les différentes composantes du projet CASPAIM, nous avons veillé à utiliser un outil graphique générique et intégrant toutes les phases la conception. Les réseaux de Petri constituent dans ce sens un outil graphique convivial facteur d'intégration. Les réseaux de Petri sont ainsi adaptés à la modélisation des différentes phases de conception tenant compte des différentes flexibilités et facilitant leur résolution.

Nous avons établi, dans ce chapitre, les classes de décision et degrés de liberté associés, puis nous avons défini les principaux critères de performance et présenté la méthode de résolution adoptée. Cette approche, proposée par H. Ohl puis par H. Camus est ici précisée pour permettre la prise en compte des flexibilités imbriquées et surtout des régimes transitoires. Le caractère progressif et linéaire de l'approche est confirmé. En réalité la linéarisation de la résolution a eu pour objet initial la simplification de la complexité de résolution. Dans ce chapitre nous avons montré qu'il existe des contraintes de précédence implicites entre certaines classes de décision. Si l'on prend en considération la combinatoire qui résulte des différents degrés de liberté, il n'est pas possible de considérer raisonnablement le problème d'une manière globale c'est à dire par la résolution simultanée de toutes les flexibilités.

Pour terminer ce chapitre nous avons présenté l'étape de planification fine nécessaire pour la décomposition de la production en plusieurs régimes permanents nécessitant de ce fait la gestion de différents régimes transitoires.

Chapitre II

II. Méthode Générale d'Evaluation et Optimisation des Performances : Présentation et Formulation des Différentes Etapes

II.1 Introduction

Dans ce chapitre, nous allons présenter la démarche structurée et progressive de l'évaluation de performances et de l'ordonnancement proposée par le groupe Evaluation de Performance au sein de l'équipe PFM du Laboratoire d'Automatique et Informatique industrielle de Lille. Comme nous l'avons déjà signalé, cette étude a fait l'objet de deux thèses de doctorat successivement présentées par Harald Ohl et Hervé Camus.

Cependant, dans l'optique d'une informatisation totale de l'approche nous devons dépasser le stade d'étude du problème, antérieurement proposée, pour la résolution de chaque étape afin d'affiner la formalisation du problème en vue du calcul du nombre de solutions (ou de bornes les cas échéants). Pour être plus précis, nous cherchons à évaluer le nombre des solution possibles et donc à caractériser finement la combinatoire du problème. Nous allons donc nous baser sur les travaux précédents en ce qui concerne les caractéristiques des phases, les flexibilités concernées ... et nous intéresser essentiellement à la formulation mathématique du problème pour tenter de dégager une méthodologie de calcul et une aide à la résolution.

L'exemple qui sera présenté à partir de § II.2 est celui défini dans [CAM 97]. Cet exemple a été choisi afin de permettre au lecteur de naviguer, éventuellement, entre une description du problème, basée sur le graphisme du RdP pour la présentation du système, et la formulation mathématique associée que nous présentons au cours de ce chapitre. Nous appliquerons l'étape de planification fine sur un exemple suffisamment complexe pour illustrer valablement la démarche. Pour **chaque phase** de la démarche progressive d'Evaluation et Optimisation de Performances, nous présenterons le **principe**, les **classes de décision** concernées, une brève description de la **manière de procéder** ainsi qu'une

formalisation mathématique pour l'évaluation du **nombre de solutions** potentielles ou éventuellement d'une borne de ce nombre. En effet, « *à chaque fois qu'elle est possible, l'analyse mathématique nous aide énormément à réduire l'explosion combinatoire* »[1].

A la suite de l'étape précédente, nous avons effectué la planification fine de la demande initiale. Nous disposons donc d'une production décomposée en plusieurs régimes permanents. Chaque régime permanent q est caractérisé par : l'**horizon de production** E(q), le **temps de cycle optimal** associé CT(q), le **nombre** X(q) **de répétitions de l'horizon**, les distributions des opérations sur les différentes machines et les différents ratios de routages entiers pour les flexibilités opératoires. Nous allons donc nous intéresser à la détermination de la commande de chaque régime permanent en respectant le temps de cycle optimal et en minimisant l'en-cours.

Ainsi, nous allons analyser, dans ce chapitre, trois phases de l'approche de résolution (phases 2, 3 et 4). La première étape est la linéarisation du modèle (classe de décision D_3), elle dénombre les différents chemins possibles pour chaque type de produits. Ces chemin sont dus aux flexibilités opératoires dans les gammes de production. La deuxième étape concerne la résolution des indéterminismes dus aux ressources de transport (classes de décision D_{6c}) en affectant, à chaque palette, une ou plusieurs pièces. Quant à la troisième phase elle se charge de donner un ordre de passage de ces pièces sur les palettes correspondantes.

II.2 Première analyse

Rappelons que nous cherchons à déterminer une commande respectant une demande précise en minimisant le makespan. L'analyse préliminaire a montré qu'il n'est pas raisonnable d'envisager une résolution avec la méthode générale d'ordonnancement, en raison de la très grande complexité du problème. La borne inférieure théorique évaluée en continu n'est pas atteignable dans le cas général. Par conséquent, elle est uniquement utilisée à titre de référence d'optimalité. Ce minimum devient donc, à ce stade de l'analyse, sans relation avec la réalité de la solution cyclique envisagée.

---

[1] « *Mathematical analysis, whenever possible, helps us reduce the computational drain tremendously* », cf. [BEL 82].

Soit une demande initiale : $\{T(I),\ I \in \{1, ..., I_{max}\}\}$. Après la phase de planification fine, cette demande est décomposée en un ensemble de productions cycliques : $\{RP_p,\ p \in \{1, ..., RP_{max}\}\}$. Considérons RP_q un régime permanent fixé de cet ensemble caractérisé par son horizon de production cyclique $E(RP_q) = \{I(q).I,\ \forall i \in \{1, ..., I_{max}\}\}$, le temps de cycle optimal associé $CT(RP_q)$ et le nombre de répétitions de cet horizon $X(q)$. Bien entendu, suite à la phase de planification fine, nous connaissons la répartition discrète effective des charges sur les machines. Par conséquent, les machines critiques et le temps de cycle optimal qu'elles imposent ont été déterminés.

Pour produire toutes les pièces demandées durant RP_q, toutes les opérations nécessaires doivent être réalisées et en particulier celles qui appartiennent aux machines critiques. Donc, pour réaliser la demande sur RP_q, chaque machine critique doit assurer la réalisation de sa charge cyclique (égale au temps de cycle $CT(RP_q)$) $X(q)$ fois. Le temps d'occupation total de la machine critique, sur le régime q, est égal à :

$$\begin{matrix}\text{Temps d'occupation} \\ \text{de la machine critique}\end{matrix} = X(q) * CT(RP_q) \text{ unités de temps.}$$

Cette valeur correspond à la borne inférieure du makespan de RP_q. Notons qu'elle ne constitue pas une borne inférieure pour la durée du régime permanent cyclique de RP_q car la machine critique peut également opérer pendant le transitoire de début ou de fin de cette production. Ainsi, la borne inférieure de la production totale est donnée par la somme de toutes les bornes inférieures des makespan des différentes productions cycliques planifiées :

$$\text{Borne Inf. du Makespan total} = \sum_{p=1}^{RP_{max}} X(p).CT(RP_p)$$

Notre problème n'est donc plus de trouver une commande minimisant le makespan théorique mais plutôt **minimisant le makespan « pour un comportement cyclique fixé »**. Pour illustrer les propositions et démonstrations des phases suivantes, nous introduisons l'exemple suivant : soit un atelier flexible composé de six ressources de transformation (R_1 à R_6) dont deux multiples (R_1 existe en trois exemplaires : $M_{1.1}$ à $M_{1.3}$, et R_4 en deux : $M_{4.1}$ et $M_{4.2}$). Considérons une demande fixée de 9 pièces d'un type A, 6 pièces de B et 6 pièces

de C. Les pièces du type A seront transportées par des palettes dédiées P_{rt1}, tandis que les deux autres types de pièces partageront les mêmes ressources de transport P_{rt2}. Pour la modélisation du problème, nous allons procéder selon la méthode proposée par le projet CASPAIM en déterminant à partir d'un modèle initial prégraphe un modèle développé et structuré, cf. [CRU 91], [AMA 92] et [CAM 96c] pour les définitions et les terminologies.

Cette méthode permet de mettre en évidence les parallélismes du système ainsi que le partage des ressources en présence. De plus, elle présente l'avantage de pouvoir contrôler l'évolution de la modélisation afin de veiller à la conservation de certaines propriétés dans le modèle telles que la vivacité et la finitude([1]). La première phase consiste à décrire les types de pièces en langage naturel, c'est-à-dire les fonctions à réaliser pour la production de chaque classe de pièces. On met ensuite en place les *Gammes Logiques* (GL) qui spécifient les opérations et l'ordre dans lequel celles-ci doivent être exécutées.

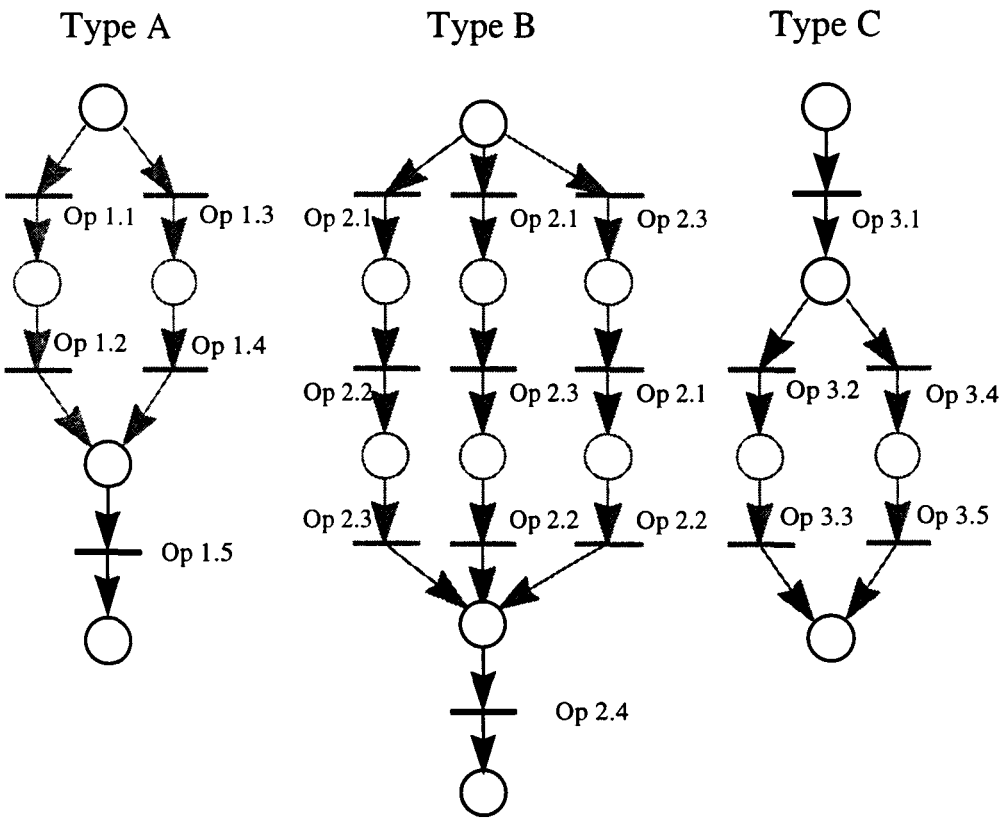


Figure II-1 : Description des gammes logiques de l'exemple illustratif

[1] C'est la propriété q'un marquage borné du Réseau de Petri, cf. [DAV 92] page 43.

La partie procédé n'est pas prise en compte et les GL ne spécifient donc ni les transferts, ni les ressources qui effectuent les opérations. Une flexibilité au niveau des GL modélise soit la possibilité d'une permutation de l'ordre des opérations (classe de décision D3a) comme c'est le cas pour le type de pièce B, cf. Figure II-1, soit un choix entre des procédés de fabrication différents (classe de décision D3c) comme c'est le cas des types A et C.

L'étape suivante consiste à ajouter aux GL les informations relatives au procédé afin de spécifier sur quelle ressource une opération est effectuée. Le résultat graphique est caractérisé par une *Gamme Opératoire Restreinte* (GOR) dans laquelle toute opération est associée à un type de ressource. Dans la Figure II-2, nous pouvons remarquer les différentes flexibilités du système :

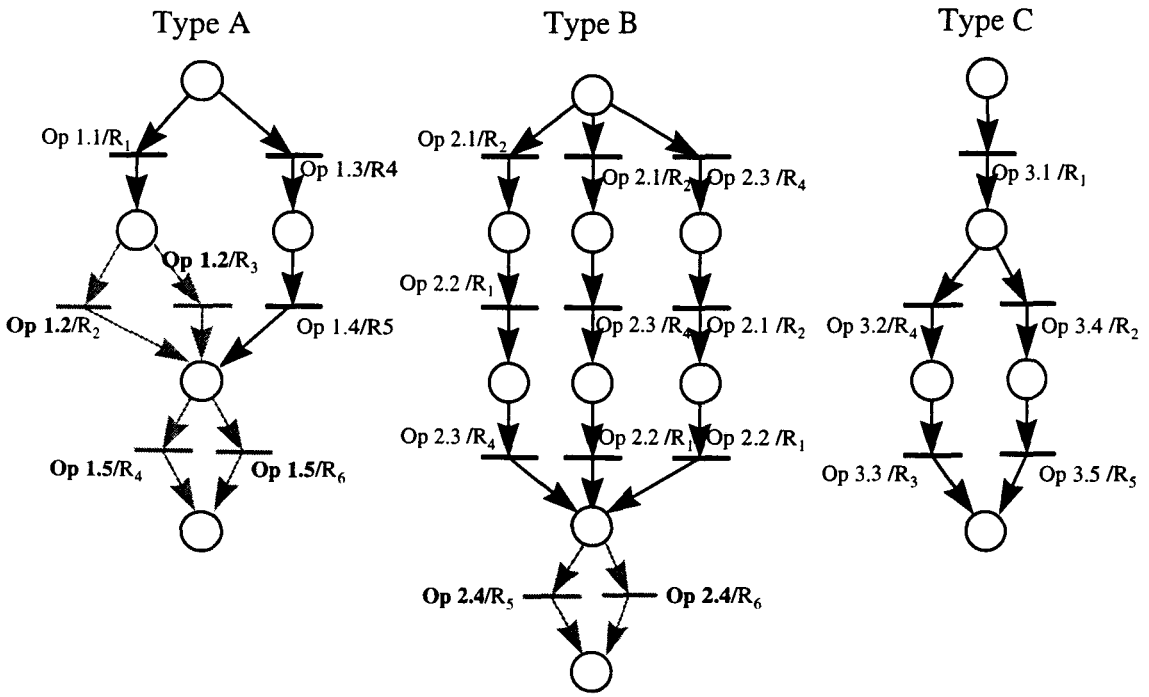


Figure II-2 : Modélisation des Gammes Opératoires restreintes

- flexibilité de procédé dans la gamme du type C (classe de décision D_{3c})
- flexibilité de permutation de la gamme du type B suivie par une flexibilité d'affectation (l'opération Op 2.4 peut être réalisée sur la ressource R_5 ou la ressource R_6) $\Rightarrow$ l'ensemble est une flexibilité enchaînée (classe de décision D_{3d}).
- une première flexibilité d'affectation (classe de décision D_{3b}) dans la première branche de la flexibilité de procédé (classe de décision D_{3c}) de la

gamme du type A (l'opération Op 1.2 peut être réalisée sur la ressource R_2 ou la ressource R_3) ce qui en fait une flexibilité imbriquée. Cette flexibilité est suivie par une autre flexibilité d'affectation (l'opération Op 1.5 peut être réalisée sur la ressource R_4 ou la ressource R_6) : l'ensemble est donc une flexibilité enchaînée.

A ce niveau de description, les fonctions nécessaires pour la réalisation des différentes pièces sont connues et nous pouvons nous intéresser à la modélisation du problème total : prise en compte des horizons de production, modélisation des ressources de transport, ...

II.3 Modélisation et développement de l'exemple illustratif

But : Pour un régime cyclique, déterminer le modèle Réseau de Petri représentant les différents objets, contraintes, ressources ...

Classe de Décision associée : aucune.

Pour construire le modèle de départ, représentant les caractéristiques du système, ses contraintes, ses ressources ..., cf. Figure II-3, il faut effectuer, dans l'ordre, les étapes suivantes :

1. Spécification, en langage naturel, des fonctions à réaliser pour chaque type de pièces,

2. Traduction de cette spécification en gammes logiques,

3. Détermination des gammes opératoires restreintes en précisant les ressources effectuant les différentes pièces,

4. Ajout des durées opératoires des différentes opérations (entre parenthèses) ainsi que les ressources partagées (places R_1 à R_6)[1],

[1] Nous avons préféré ne pas représenter les arcs reliant les ressources partagées pour ne pas alourdir le schéma.

5. Ajout des ressources de transport (places P_{rt1} et P_{rt2}) qui forment ainsi les boucles intérieures,

6. Ajout de la boucle extérieure (avec la transition t_f) modélisant le comportement cyclique du système ainsi que les ratios de production (r_A, r_B et r_C).

On obtient alors un modèle Réseau de Petri appelé dans la littérature « RdP à T-semiflots librement reliés »[1]. Par construction, ce modèle vérifie les « bonnes » propriétés de vivacité de finitude, ... dont nous avons besoin pour la suite de l'étude, cf. [CAM 90].

La borne max. du makespan (hors régimes transitoires) nécessaire pour la réalisation de la demande est donnée par la production des pièces du type A suivies du types B puis C., cf. § I.4.2. Elle est donnée par la formule suivante, cf. Annexe II :

$$\text{Borne max. du permanent} = \underbrace{40}_{\substack{\text{production} \\ \text{du type A}}} + \underbrace{24}_{\substack{\text{production} \\ \text{du type B}}} + \underbrace{27}_{\substack{\text{production} \\ \text{du type C}}} = 91 \text{ u.t.}$$

Equation II-1 : Borne max. du makespan optimal hors régimes transitoires

Cette borne max. est la seule qu'on puisse calculer pour l'instant. Elle donne un majorant de la durée du régime permanent correspondant au makespan optimal que nous cherchons à déterminer.

Concernant la borne inf. théorique, elle est déterminée par le calcul de flux, cf. [FRE 88], d'une production de 21 pièces dont $\frac{3}{7}$ de type A, $\frac{2}{7}$ de type B et autant de type C avec une prise en compte du caractère discret, cf. Annexe II :

$$\text{Borne inf. théorique du makespan} = 69 \text{ u.t.}$$

Etant donné les contraintes sur la taille sur l'horizon cyclique (inférieur à 10), l'application de l'étape de planification fine donne un horizon cyclique E={3 A, 2 B et 2 C}, un temps de cycle optimal associé CT=24 unités de temps et un nombre de répétitions de cet horizon donné par X=3. D'où la nouvelle borne min. du makespan :

$$\text{Borne min. du makespan} = 3 * 24 = 72 \text{ u.t.}$$

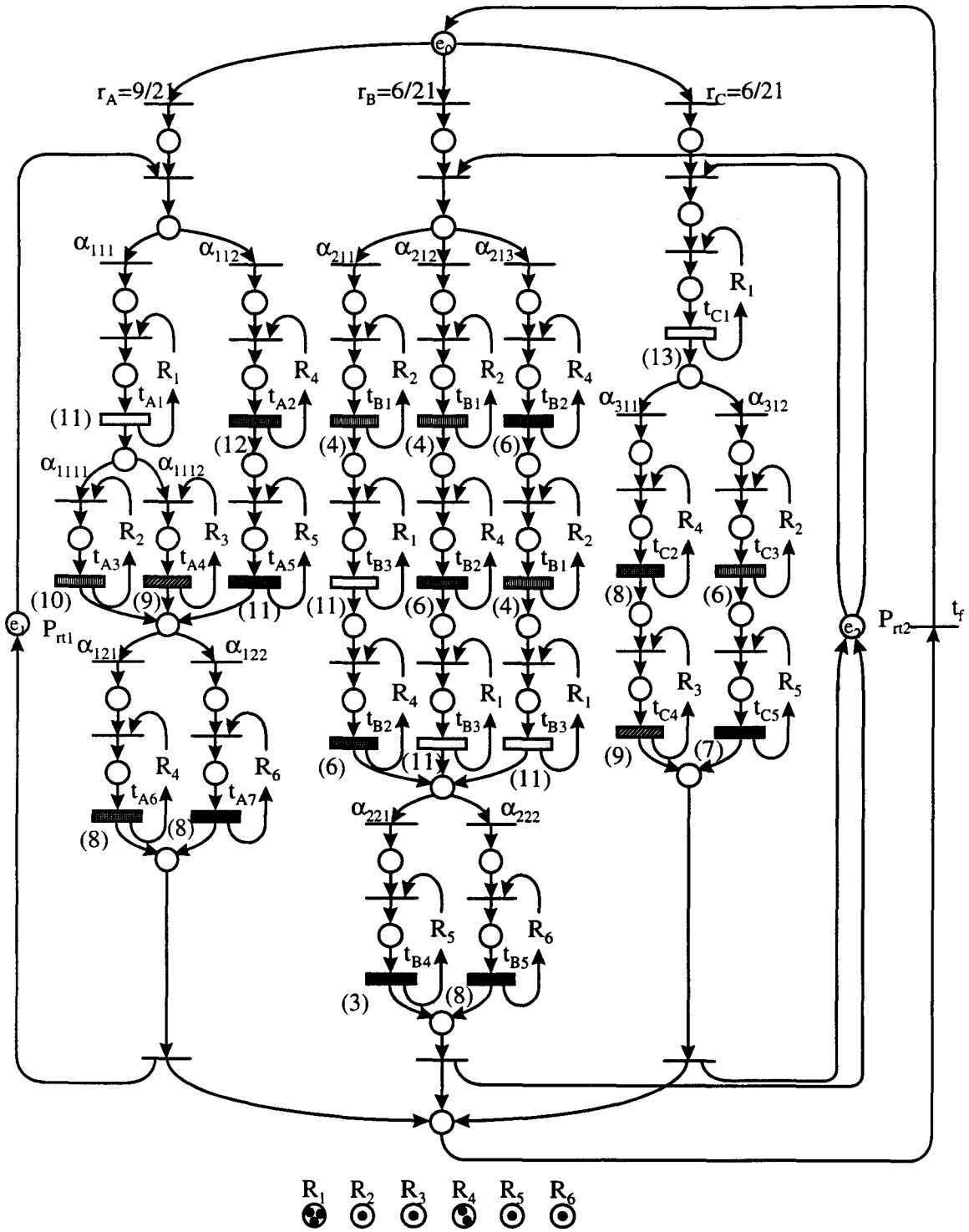


Figure II-3 : Réseau de Petri modélisant le problème initial

Nous allons donc, dans la suite de ce mémoire, étudier ce cas de figure en lui appliquant la démarche progressive d'évaluation de performances l'algorithme d'ordonnancement (cf.

[1] Free Related T-semiflows nets (FRT nets) en anglais : ceci veut dire que le tir d'un T-semiflot (réalisation d'une pièce)

Chapitre III), et le calcul du régime transitoire (cf. Chapitre IV). Ainsi, nous déterminerons la commande associée à cet horizon de production E. Comme nous pouvons le constater, le modèle initial est basé sur une représentation du flux continu. Pour représenter l'aspect discret de la commande nous allons utiliser le modèle mis en place par H. Ohl, cf. Figure II-4. Cette représentation « force » le réseau de Petri à respecter l'horizon de production E par l'introduction de la transition t_x qui régule les tirs des T-semiflots par cycle de fonctionnement. De plus, l'étape de planification fine détermine la répartition des charges discrètes sur les différentes machines, cf. Figure II-5. Donc, désormais on ne parle plus de ressources mais plutôt de machines qui sont toutes différenciées.

La gamme représentant le type A (dont on fabriquera I_A=3 pièces par cycle) utilisera, à chaque période, la machine $M_{1.1}$ pour réaliser la première opération d'une pièce, la machine $M_{1.2}$ pour réaliser la première opération d'une autre pièce et la machine $M_{4.2}$ pour celle de la troisième pièce. Ensuite, pour les deux premières pièces elle sollicitera une fois sur deux la machine M_2 et une autre fois la machine M_3. Cette flexibilité imbriquée n'est plus un indéterminisme étant donné que nous connaissons parfaitement la distribution des pièces lui appartenant. En ce qui concerne la dernière opération à réaliser sur les 3 différentes pièces, on aura le choix entre les machines $M_{4.1}$ et M_6 sachant qu'à chaque période la première doit être utilisée une fois pour deux utilisations de la deuxième.

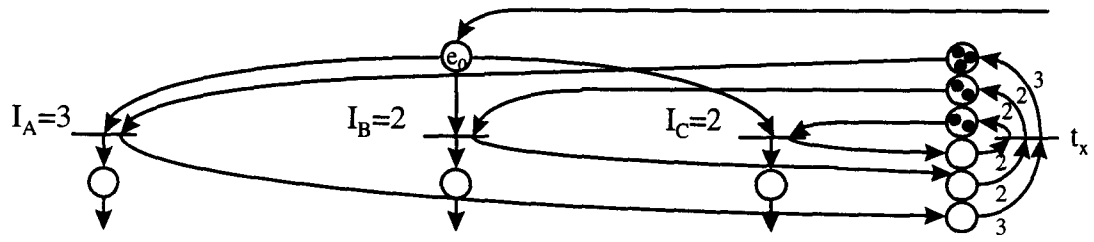


Figure II-4 : Modélisation des ratios discrets de production, [OHL 95a].

Concernant les deux pièces du type B, à fabriquer par cycle, nous avons encore le choix entre les trois flexibilités de permutation (puisqu'elles sont équivalentes du point de vue charge de travail), mais pour la dernière opération, les deux pièces utiliseront la machine M_5 (le ratio de routage associé à la machine M_6 est nul). Par conséquent, cette flexibilité est résolue.

dépend uniquement des ratios de production et non de la disponibilité des ressources (de transport par exemple).

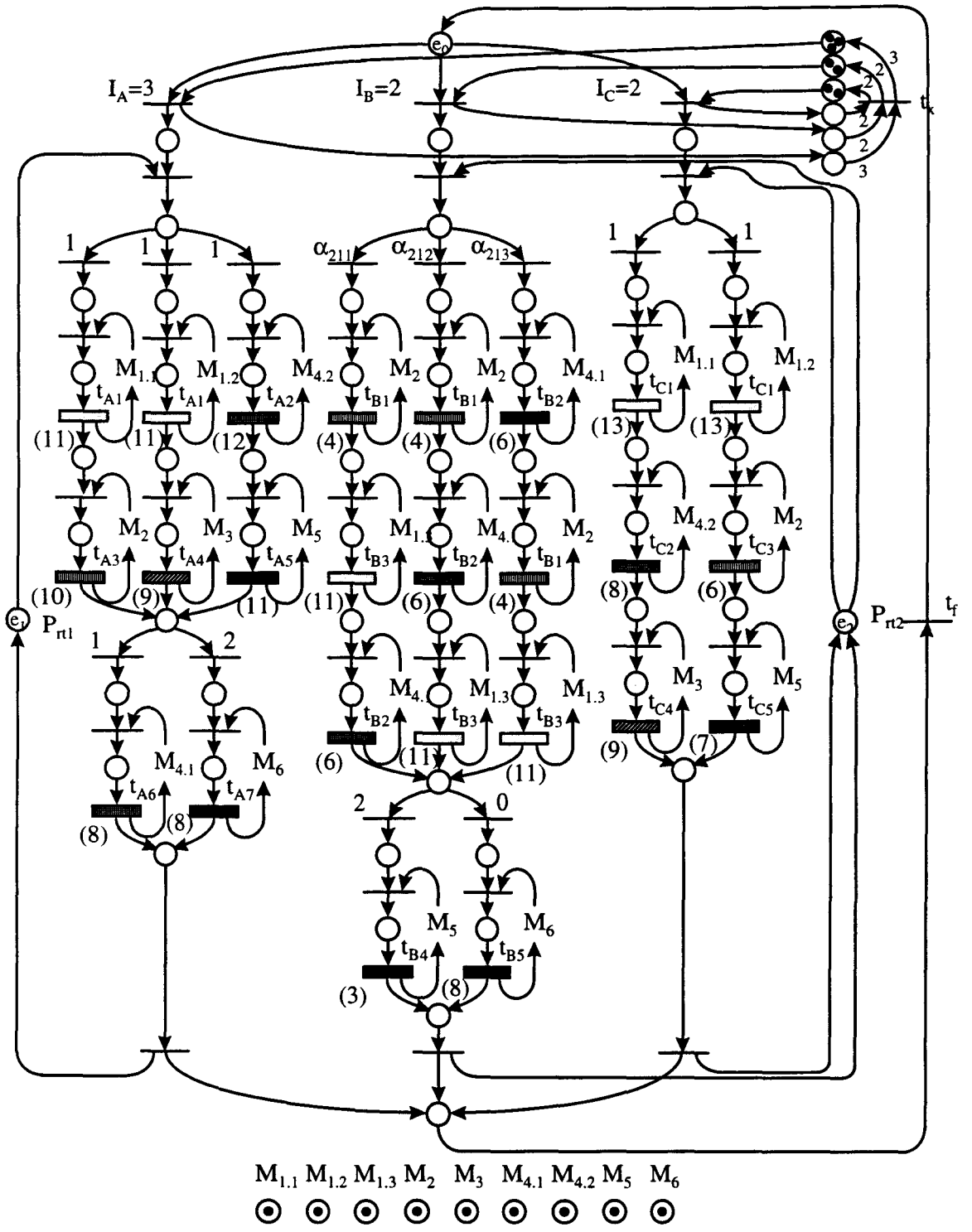


Figure II-5 : Modèle initial du régime permanent obtenu par la planification fine

Finalement, les deux pièces du type C, à réaliser par période, utiliseront comme prévu la ressource R_1 pour la première fois mais, en plus de cette information que nous avions déjà lors de la mise en place des gammes opératoires restreintes, nous savons maintenant que cette opération se fera une fois sur deux sur la machine $M_{1.1}$ et l'autre fois sur la machine

$M_{1.2}$. Pour terminer la fabrication des deux pièces, l'une des deux passera par la première branche ($M_{4.2}$ puis M_3) tandis que l'autre passera par la deuxième (M_2 suivie de M_5). En connaissant les distributions discrètes des opérations sur les différentes machines, nous pouvons déterminer les charges des machines par cycle de fonctionnement :

$$\left|\begin{array}{rcl} M_{1.1} & : & Z = \mathbf{24}\ u.t. \\ M_{1.2} & : & Z = \mathbf{24}\ u.t. \\ M_{1.3} & : & Z = 22\ u.t. \\ M_2 & : & Z = \mathbf{24}\ u.t. \\ M_3 & : & Z = 19\ u.t. \\ M_{4.1} & : & Z = 20\ u.t. \\ M_{4.2} & : & Z = 20\ u.t. \\ M_5 & : & Z = \mathbf{24}\ u.t. \\ M_6 & : & Z = 16\ u.t. \end{array}\right.$$

Equation II-2 : Charges des différentes machines

Par conséquent, nous connaissons également les machines critiques, qui fonctionneront à plein régime et sans attente, ce sont les machines $M_{1.1}$, $M_{1.2}$, M_2 et M_5. Leurs charges de travail sont égales à 24 u.t. par cycle de fonctionnement. De plus, nous pouvons remarquer que la machine la moins sollicitée (M_6) fonctionne à 67% du temps de cycle ce qui signifie que les charges sont bien réparties entre les machines qui fonctionnent presque toutes à plein régime. Les indéterminismes restants sont décrits ci-après, dans l'ordre selon lequel ils seront traités :

1. choix des chemins empruntés pour la flexibilité de permutation (type B)
2. linéarisation des gammes (flexibilités enchaînées)
3. détermination des pièces qui utiliseront les mêmes ressources de transport (partition des gammes)
4. introduction des contraintes de précédence entre ces pièces (regroupements cycliques)
5. ordonnancement des opérations sur les machines et détermination de l'en-cours nécessaire et suffisant pour respecter le temps de cycle optimal.

II.4 Linéarisation du modèle

Problème : Extraire les chemins admissibles en fonction des ratios de routages entiers associés et choisir les branches à utiliser dans les flexibilités de permutation.

Classe de Décision associée : D_3

Cette étape se charge de lever tous les indéterminismes dus aux flexibilités de gammes encore existantes (flexibilités de permutation et flexibilités enchaînées). En effet, la détermination des différents ratios de routage ne concerne pas les flexibilités de permutations (puisqu'elles n'influencent pas les charges des différentes machines). De même, la détermination des différents ratios de routage dans une flexibilité enchaînée ne donne pas d'indications quant aux vrais chemins à parcourir, cf. Figure II-6 pour un exemple. Ainsi, nous cherchons dans cette phase à déterminer, pour chaque type de pièces, les différents itinéraires possibles, respectant les contraintes de routages, pour la fabrications des différents exemplaires.

II.4.1 Flexibilités de permutation

Nous nous intéressons, dans un premier temps, à la détermination des chemins de la flexibilité de permutation que nous allons réellement emprunter. Pour notre exemple, ce problème consiste à choisir, parmi les p=3 branches de cette flexibilité, les I_B=2 branches dont nous avons besoin, sachant que le même chemin peut être emprunté plusieurs fois. C'est donc un problème purement combinatoire, identique au problème de tir de I_B boules dans une urne contenant un total de p boules avec remise de la boule tirée à chaque fois. Le nombre de solutions est donc dans notre cas de $C_4^2 = 6$ pour notre exemple, cf. [KAU 68]. Plus généralement :

$$\text{Nombre de combinaisons dans une flexibilité de permutation} = C_{p+I_B-1}^{I_B}$$

Les 6 solutions sont donc à envisager, elles peuvent conduire à des résultats différents. De plus si nous avions plusieurs flexibilités de permutation, nous aurions à les considérer toutes (i.e. le produit des différents nombres de solutions pour chaque flexibilité). Cette opération est certes coûteuse en combinatoire (nombre élevé de combinaisons à considérer)

mais va dans le sens de l'obtention de la meilleure solution. Pour la suite nous traitons une combinaison consistant à utiliser les deux premières branches de la flexibilité de permutation. Ne subsistent alors que les flexibilités enchaînées (gamme A)[1] que nous allons étudier dans la suite de ce paragraphe.

II.4.2 Flexibilités enchaînées

Le problème est d'extraire tous les chemins admissibles d'une flexibilité enchaînée. Par exemple dans la gamme de pièces A, cf. Figure II-6, nous avons trois chemins différents (que nous allons appeler A1.1, A1.2 et A1.3) chacun à associer à l'un des trois suivants : A2.1, A2.2 et A2.2 (c'est la branche contenant la machine M_6).

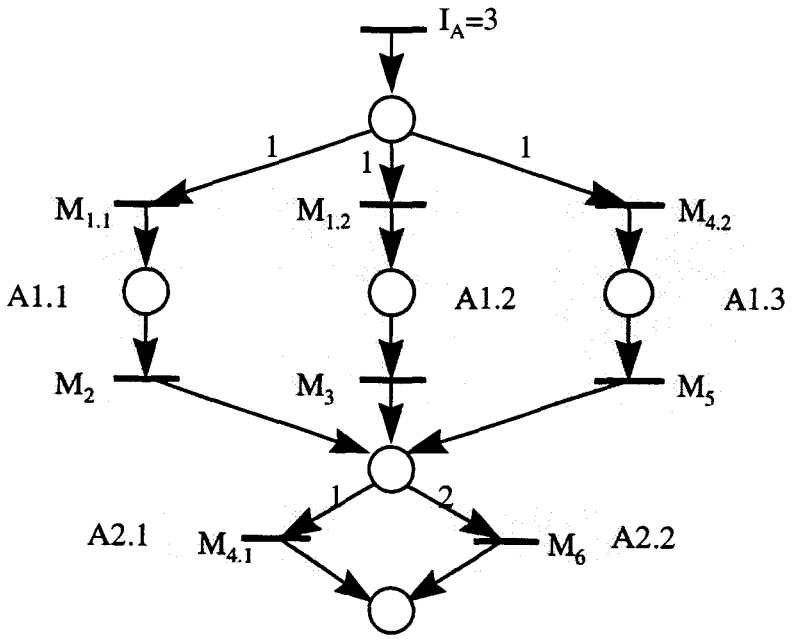


Figure II-6 : Flexibilité enchaînée du type de pièces A

Les différentes combinaisons possibles sont au nombre de trois :

$\mathcal{M}_1$={(A1.1, A2.1), (A1.2, A2.2),(A1.3, A2.2)}([2]),

$\mathcal{M}_2$={(A1.1, A2.2), (A1.2, A2.1), (A1.3, A2.2)} et

$\mathcal{M}_3$={(A1.1, A2.2), (A1.2, A2.2), (A1.3, A2.1)}.

[1] Comme nous avons remarqué sur le résultat de la planification fine, la gamme B ne contient plus de flexibilité enchaînée étant donné qu'une des deux branches (Machine M_6) possède un ratio de routage nul.

[2] C'est la combinaison représentée à la Figure II-7

II.4.2.1 Etude du cas général

Dans le cas général, soit un ensemble de chemins $\mathcal{A}1=\{\overbrace{A1.1,...,A1.1}^{\lambda_1 \text{ fois}},\overbrace{A1.2,...,A1.2}^{\lambda_2 \text{ fois}},...,\overbrace{A1.n,...,A1.n}^{\lambda_n \text{ fois}}\}$ à associer à $\mathcal{A}2=\{\overbrace{A2.1,...,A2.1}^{\beta_1 \text{ fois}},\overbrace{A2.2,...,A2.2}^{\beta_2 \text{ fois}},...,\overbrace{A2.m,...,A2.m}^{\beta_m \text{ fois}}\}$ tel que $\sum_{i=1}^{n}\lambda_i = \sum_{j=1}^{m}\beta_j = I_A$ où I_A est le ratio de routage entier associé à la gamme A.

Dans le cas extrême où tous les chemins de $\mathcal{A}1$ sont différents ainsi que ceux de $\mathcal{A}2$ ($\forall i,j\ \lambda_i = 1 = \beta_j$), il est évident qu'il y a I_A !([1]) combinaisons différentes([2]).

Par conséquent, I_A ! est un majorant du nombre de combinaisons que nous recherchons. Nous allons établir un majorant plus précis. Soit la matrice n lignes m colonnes à éléments entiers : $\mathcal{M}(m_{ij} \in IN,\ i\in\{1,...,n\}$ et $j\in\{1,...,m\})$ qui, à un élément A1.i et un élément A2.j, associe l'entier m_{ij} $\begin{vmatrix} m_{ij} \le \lambda_i \\ m_{ij} \le \beta_j \end{vmatrix}$ égal au nombre de branches A1.i couplées avec des branches A2.j.

Pour l'exemple illustratif, les trois combinaisons s'écrivent :

$$\mathcal{M}1=\begin{array}{c} \\ A1.1 \\ A1.2 \\ A1.3 \\ \\ \end{array}\begin{array}{c} \begin{array}{cc} A2.1 & A2.2 \end{array} \\ \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \\ \begin{array}{cc} \beta_1 = 1 & \beta_2 = 2 \end{array} \end{array}\quad \begin{array}{l} \lambda_1 = 1 \\ \lambda_2 = 1, \\ \lambda_3 = 1 \end{array}$$

$$\mathcal{M}2=\begin{array}{c} \\ A1.1 \\ A1.2 \\ A1.3 \\ \\ \end{array}\begin{array}{c} \begin{array}{cc} A2.1 & A2.2 \end{array} \\ \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \\ \begin{array}{cc} \beta_1 = 1 & \beta_2 = 2 \end{array} \end{array}\quad \begin{array}{l} \lambda_1 = 1 \\ \lambda_2 = 1 \\ \lambda_3 = 1 \end{array} \text{ et } \mathcal{M}3=\begin{array}{c} \\ A1.1 \\ A1.2 \\ A1.3 \\ \\ \end{array}\begin{array}{c} \begin{array}{cc} A2.1 & A2.2 \end{array} \\ \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \\ \begin{array}{cc} \beta_1 = 1 & \beta_2 = 2 \end{array} \end{array}\quad \begin{array}{l} \lambda_1 = 1 \\ \lambda_2 = 1. \\ \lambda_3 = 1 \end{array}$$

[1] I_A ! est le produit factoriel de I_A.

[2] Pour le premier élément il y a I_A possibilités, pour le second I_A-1 ainsi de suite d'où le résultat, cf. [KAU 64].

Ces matrices sont construites de façon à avoir $\forall j \ \sum_{i=1}^{n} m_{ij} = \beta_j$ et $\forall i \ \sum_{j=1}^{m} m_{ij} = \lambda_i$. La recherche du nombre T de combinaisons possibles revient donc à la résolution d'un système à $(m+n)$ équations linéaires à variables entières :

$$\begin{cases} \forall i, j \quad m_{ij} \in IN \\ \sum_{i=1}^{n} m_{ij} = \beta_j \\ \sum_{j=1}^{m} m_{ij} = \lambda_i \end{cases}$$

Equation II-3 : Formalisation du problème d'extraction des chemins

Lemme II.1 :

Le nombre de solutions du système : $\begin{cases} \forall i \quad x_i \in IN \\ \sum_{i=1}^{m} x_i = p \end{cases}$ *, où m et p sont deux entiers naturels non nuls, est égal à :* $T(m,p) = C_{m+p-1}^{m-1} = C_{m+p-1}^{p} = \frac{(m+p-1)!}{(p)!*(m-1)!}$.

Démonstration : cf. Annexe II([1]).

L'application de ce lemme au problème qui nous préoccupe est immédiate : pour tout j ∈ {1,...,m}, le nombre de solutions de l'équation $\sum_{i=1}^{n} m_{ij} = \beta_j$ isolée est donné par :

$$T1(n, \beta_j) = C_{\beta_j+n-1}^{n-1}.$$

De même pour tout i ∈ {1,...,n}, le nombre de solutions de l'équation $\sum_{j=1}^{m} m_{ij} = \lambda_i$ prise seule est donné par :

$$T2(m, \lambda_i) = C_{\lambda_j+m-1}^{m-1}.$$

[1] D'autres démonstrations, différentes de celle que nous présentons, sont possibles, cf. [COM 74] et [MOU 97].

Soit le sous-système S1 : $\forall j \ \sum_{i=1}^{n} m_{ij} = \beta_j$ contenant m équations. Il est évident qu'il suffit de résoudre m-1 équations de ce système et de vérifier les contraintes données par le sous-système S2 : $\forall i \ \sum_{j=1}^{m} m_{ij} = \lambda_i$ pour déterminer toutes les variables de S1.

Alors, le nombre de solutions de ce sous-système est donné par :

$$Nbre\_sol.(S1) \leq \frac{\prod_{j=1}^{m} T1(n,\beta_j)}{\max_j \left(T1(n,\beta_j)\right)}.$$

Le même raisonnement appliqué au sous-système S2 nous donne :

$$Nbre\_sol.(S2) \leq \frac{\prod_{i=1}^{n} T2(m,\lambda_i)}{\max_i \left(T2(m,\lambda_i)\right)}$$

Donc, en utilisant les deux derniers résultats, nous obtenons la proposition suivante :

Proposition II.1 :

la borne supérieure du *nombre total de solutions T* est donnée par :

$$T \leq \min\left(\frac{\prod_{j=1}^{m} T1(n,\beta_j)}{\max_j \left(T1(n,\beta_j)\right)}, \frac{\prod_{i=1}^{n} T2(m,\lambda_i)}{\max_i \left(T2(m,\lambda_i)\right)} \right)$$

Equation II-4 : Borne supérieure du nombre de combinaisons dans une flexibilité enchaînée

II.4.2.2 Application à l'exemple

• Système S1 : $\begin{cases} m_{1.1} + m_{1.2} = 1 & \text{nombre de solutions : } T1(2,1) = 2 \\ m_{2.1} + m_{2.2} = 1 & \text{nombre de solutions : } T1(2,1) = 2 \\ m_{3.1} + m_{3.2} = 1 & \text{nombre de solutions : } T1(2,1) = 2 \end{cases}$

$$Nbre\_sol.(S1) \leq \frac{\prod_{i=1}^{3} T1(2,\lambda_i)}{\max_i\left(T1(2,\lambda_i)\right)} = \frac{2*2*2}{2} = 4 \text{ solutions}$$

• Système S2 : $\begin{cases} m1.1 + m2.1 + m3.1 = 1 & \text{nombre de solutions : } T2(3,1) = 3 \\ m1.2 + m2.2 + m3.2 = 2 & \text{nombre de solutions : } T2(3,2) = 6 \end{cases}$

$$Nbre\_sol.(S2) \leq \frac{\prod_{j=1}^{2} T2(3,\beta_j)}{\max_j\left(T2(3,\beta_j)\right)} = \frac{6*3}{6} = 3 \text{ solutions}$$

d'où, en utilisant l'Equation II-4 :

$$T \leq \min.\left(\frac{\prod_{j=1}^{m} T1(n,\beta_j)}{\max_j\left(T1(n,\beta_j)\right)}, \frac{\prod_{i=1}^{n} T2(m,\lambda_i)}{\max_i\left(T2(m,\lambda_i)\right)} \right) = \min.(4,3) = 3 \text{ solutions}$$

En fait, pour cet exemple la borne maximale (égale à 3 solutions) est égale au nombre réel de combinaisons et on peut remarquer que nous avons bel et bien amélioré la première borne calculée : I_A ! = 6. Rappelons que cette méthode n'est possible que pour des flexibilités enchaînant 2 flexibilités simples. Dans les cas de plusieurs « étages » de flexibilités, la seule méthode que nous proposons, serait d'étudier ces flexibilités enchaînées deux par deux, de calculer à chaque fois une borne max. et de considérer cette borne comme un nombre effectif de branches à associer aux branches suivantes, cf. [MOU 97].

Parmi les trois combinaisons possibles de notre exemple, nous représentons (cf. Figure II-7) celle qui correspondent à $\mathcal{M}_1$ que nous allons conserver pour la suite de la réalisation. Rappelons auparavant qu'il faut considérer les trois cas pour pouvoir garantir la solution optimale.

Le modèle RdP résultant de cette étape est un graphe linéarisé où seuls les indéterminismes dus aux ressources de transport et aux machines simples partagées persistent, cf. Figure II-7. Chaque branche linéaire (appelée gamme linéaire) de ce graphe correspond à une pièce à réaliser durant un cycle du régime permanent.

Par conséquent, nous aurons à réaliser, par période, trois pièces distinctes par leur procédé de fabrication, du type A (notées A1, A2 et A3) qui utiliserons le même type de palettes P_{rt1} ainsi que deux pièces « distinctes » de type B (notées B1 et B2) et deux pièces « différentes » de type C (notées C1 et C2). Ces quatre dernières pièces utiliseront le même type de palette P_{rt2}. Durant la phase suivante, nous allons nous intéresser à ces ressources de transport en attribuant à chaque ensemble de pièces un ensemble de palettes qui les achemineront réellement durant le processus de fabrication.

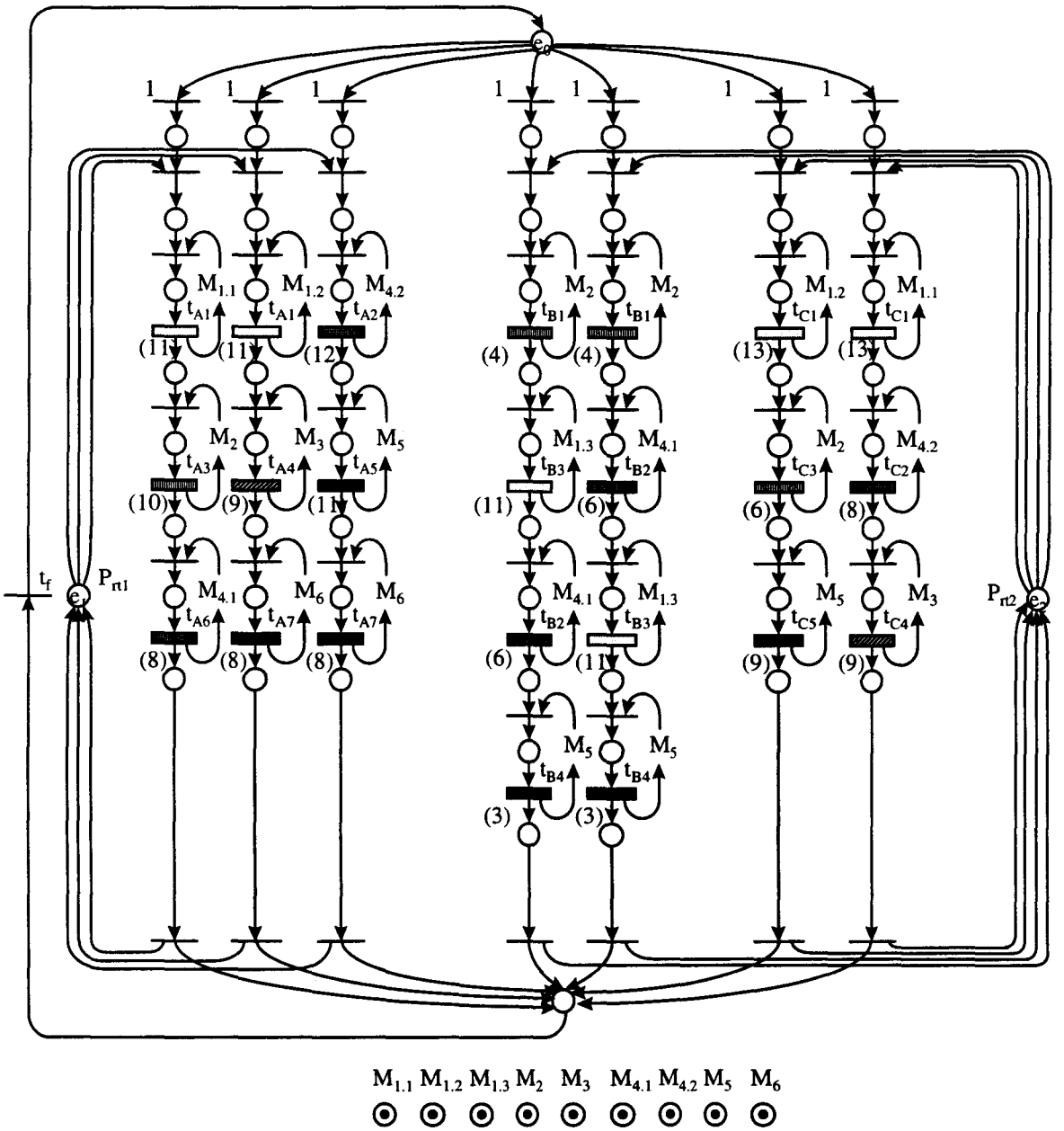


Figure II-7 : Modèle linéarisé

II.5 Partition des gammes

Problème : trouver tous les regroupements possibles entre les pièces utilisant les mêmes palettes afin de minimiser ultérieurement l'en-cours.

Classes de Décision associées : D_{6c}

Dans la suite logique de la recherche d'optimisation des performances du système, nous nous intéressons maintenant à l'optimisation du nombre de ressources de transport utilisées. H. Ohl a introduit la possibilité de dédier une palette (ou un ensemble de palettes) à plusieurs pièces. Nous allons évaluer le nombre de combinaisons possibles concernant le regroupement de pièces pour l'utilisation des mêmes palettes. Soit H le nombre de types de palettes. Pour tout $h \in \{1,...,H\}$, nous notons S_h l'ensemble de gammes linéaires pouvant être transportées par le type de palettes h, cf. § I.2.4.

Nous allons étudier les différentes décompositions possibles d'un ensemble S_h en plusieurs sous-ensembles disjoints (également appelés partitions). Chaque partition contiendra les pièces qui seront réellement transportées, à tour de rôle, par les mêmes palettes physiques. Considérons le cas de l'exemple illustratif. Les trois gammes linéaires réalisant des pièces de type A (appelées A1, A2, et A3) utilisent la même ressource de transport P_{rtl} (elles forment donc l'ensemble S_{Prtl}={A1, A2, A3}). Les partitions possibles de S_{Prtl} sont au nombre de 5 :

Q^1={{A1, A2, A3}}([1]),

Q^2={{A1, A2}, {A3}},

Q^3={{A1, A3}, {A2}},

Q^4={{A2, A3}, {A1}} et

Q^5={{A1}, {A2}, {A3}}([2]).

[1] Cette configuration est appelée séquentialisation totale puisqu'elle regroupe toutes les gammes.

[2] Cette configuration est appelée parallélisation totale, elle consiste à affecter chaque palette à une seule pièce.

II.5.1 Calcul des bornes

Notons $S_k = \left\{ \overbrace{G1,...G1}^{\alpha_1 \text{ éléments}}, \overbrace{G2,...G2}^{\alpha_2 \text{ éléments}}, ..., \overbrace{Gn,...,Gn}^{\alpha_n \text{ éléments}} \right\}$ un ensemble de gammes linéaires nécessitant le type de palettes k avec G1, G2, ..., Gn : n gammes différentes dans le sens où elles correspondent à des séquences d'opérations différentes([1]) et non pas à des types de pièces différents([2]). Soit également $N = \sum_i \alpha_i$ le cardinal de l'ensemble S_k.

Si tous les α_i sont égaux à 1 (chaque gamme linéaire est en exemplaire unique), le nombre de combinaisons possibles est donné par **le nombre de Bell B_n**, cf. [COM 74] pour les définitions et [MOU 97] pour une démonstration. Ce nombre est défini par l'équation récurrente suivante :

$$\begin{cases} B_0 = 1 \\ B_n = \sum_{i=1}^{n} \left(B_i * C_n^i \right) \end{cases}$$

Par conséquent, le nombre de Bell est un **majorant du nombre de partitions dans le cas général**. Nous allons maintenant définir une borne inférieure du nombre de partitions pour en améliorer l'évaluation. Soit $\mathbb{Q}$ une partition donnée comportant q ensembles $\mathbb{Q}_i$, i $\in \{1,...,q\}$ telle que :

$$\begin{cases} \bigcup_{i=1}^{q} \mathbb{Q}_i = S_k \\ \forall i, j \quad \mathbb{Q}_i \cap \mathbb{Q}_j = \varnothing \\ \forall i \quad \text{card}(\mathbb{Q}_i) = l(i) \neq 0 \text{ avec} \\ \qquad\qquad \sum_{i=1}^{q} l(i) = N \end{cases}$$

Equation II-5

[1] Les séquences d'opérations sont différenciées soit par l'utilisation de ressources de transformation différentes, soit par un ordre de passage différent sur les machines soit par des temps opératoires différents : ne sont considérées comme identiques que les gammes strictement identiques au vu des critères précédemment cités.

[2] Deux gammes linéaires peuvent être différentes tout en réalisant le même type de pièces : dans l'exemple Figure II-7, il y a 3 gammes linéaires distinctes {A1, A2 et A3}produisant le même type de pièces A.

Nous pouvons, sans crainte de modifier le problème, choisir les Q_i de façon à avoir une suite $l(i)$ monotone. En effet, étant donné que nous construisons des ensembles non ordonnés, permuter Q_i et Q_j ne change aucunement la solution.

Supposons que S_k ne contienne que des éléments identiques (n=1 et α_1=N).

Chercher le nombre de partitions possibles revient à déterminer toutes les décompositions possibles de S_k en q sous-ensembles disjoints ($q \in \{1,...,N\}$). En utilisant l'Equation II-5, nous déduisons que ce problème est équivalent à la décomposition de l'entier N en une suite monotone d'entiers positifs $l(i)$.

Soit a(i), $i \in IN^*$, une **suite décroissante** d'entiers positifs (cette suite est donc stationnaire et convergente[1]) telle que $\sum_{i>0} a(i) = N$. Il est évident que $\forall\ i > N \quad a(i)=0$.

Nous avons donc $\sum_{i=1}^{N} a(i) = N$. Le nombre de possibilités de construction de suites a(i) respectant cette condition est appelé p(*somme* = N, *nombre d'éléments* = N) = p(N,N).

Lemme II.2 :

Le nombre de partitions possibles dans un ensemble de N éléments identiques est égal à p(N,N). Ce nombre est une ***borne inférieure*** *du cas général de partitions dans un ensemble de N éléments quelconques.*

Démonstration : cf. Annexe II.

Calcul des valeurs de p(N,N) :

Soit a(i) une **suite croissante** finie de q entiers naturels vérifiant : $\sum_{i=1}^{q} a(i) = N$. Nous allons calculer le nombre p(N,q) de solutions possibles de construction de a(i). On distingue **deux cas disjoints** :

- a(1) = 0 : alors le nombre de solutions est égal à p(N,q-1).

[1] Démonstration triviale en utilisant le critère de convergence sur les suites et le théorème de Bolzano-Weierstrass

- $a(1) \neq 0$: alors tous les a(i) sont différents de 0 (puisque a(i) est croissante). On peut donc soustraire une unité à chaque élément de la somme ce qui nous donne $\sum_{i=1}^{q}(a(i)-1)=\sum_{i=1}^{q}a'(i)=N-q$ et le nombre de solutions est de p(N-q,q).

Les deux cas étant disjoints, le nombre total de partitions d'un entier naturel N en q entiers positifs est donc égal à la somme des deux soit :

Lemme II.3 :

avec la définition précédente de p(N,N), on obtient :

$$p(1,q)=1,\ p(N,1)=1$$
et
$$p(N,q)=p(N,q-1)+p(N-q,q)$$

Equation II-6 : définition récurrente de p(N,q)

Nous donnons, ci-dessous, un tableau de valeurs de p(N,q) sachant que pour l'instant nous nous intéressons uniquement à la diagonale (en gras sur fond gris) qui représente les valeurs de p(N,N) pour $N \in \{1, ..., 9\}$, cf. [MOU 97].

p(N,q)	1	2	3	4	5	6	7	8	9
1	**1**	1	1	1	1	1	1	1	1
2	1	**2**	2	2	2	2	2	2	2
3	1	2	**3**	3	3	3	3	3	3
4	1	3	4	**5**	5	5	5	5	5
5	1	3	5	6	**7**	7	7	7	7
6	1	4	7	9	10	**11**	11	11	11
7	1	4	8	11	13	14	**15**	15	15
8	1	5	10	15	18	20	21	**22**	22
9	1	5	12	18	23	26	28	29	**30**

Tableau I-1 : Nombre de décompositions de N en q entiers positifs

Nous pouvons, cependant, remarquer que p(N,q) reste constant pour q > N. En effet, décomposer N en q entiers positifs avec q>N revient à fixer au moins q-N entiers à zéro et à chercher les valeurs des N entiers restant donc p(N,q>N) = p(N,N). On obtient donc dans le cas général :

$$p(N,N) \leq \text{Nombre de partitions possibles de } S_k \leq B_N$$

Equation II-7 : bornes du nombre de partitions de S_k

II.5.2 Etude du problème de répartition

Après avoir établi des bornes du nombre de solutions recherchées, nous allons décomposer le problème en sous-problèmes afin de mieux l'étudier, cf. Figure II-10. En effet, parmi toutes les solutions nous distinguons celles qui sont composées du même nombre q de sous-ensembles. Ces partitions formées exactement de q composantes sont ensuite distinguées selon les structures de leurs composantes. Enfin, on dénombre les combinaisons possibles et on remonte en faisant la somme à chaque niveau.

Afin d'illustrer cette décomposition, nous allons l'appliquer à S_{prt1}={A1, A2, A3} de type A pour l'exemple illustratif, cf. Figure II-8. Nous retrouverons les 5 combinaisons suivantes :

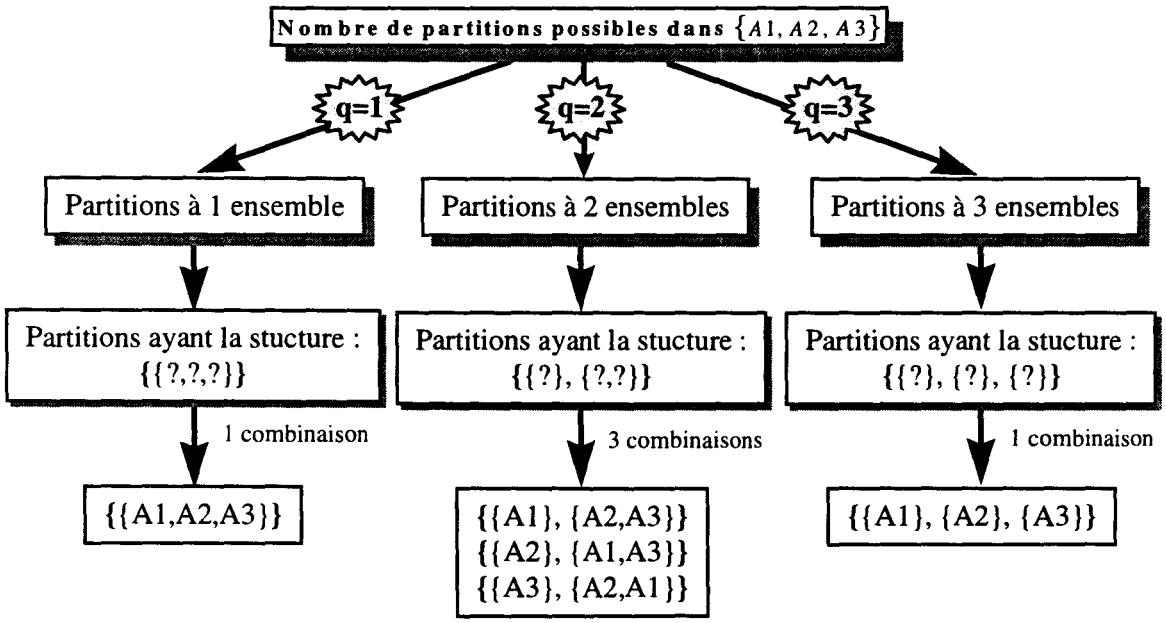


Figure II-8 : Partitions possibles de S_{Prt1}

De même, pour l'ensemble de gammes S_{prt2}={B1, B2, C1, C2} de l'exemple illustratif, cf. Figure II-9 : le nombre de combinaisons possibles est égale à 15. Nous représentons Figure II-9 la décomposition de ce cas particulier de recherche du nombre de partitions dans un ensemble de 4 éléments distincts :

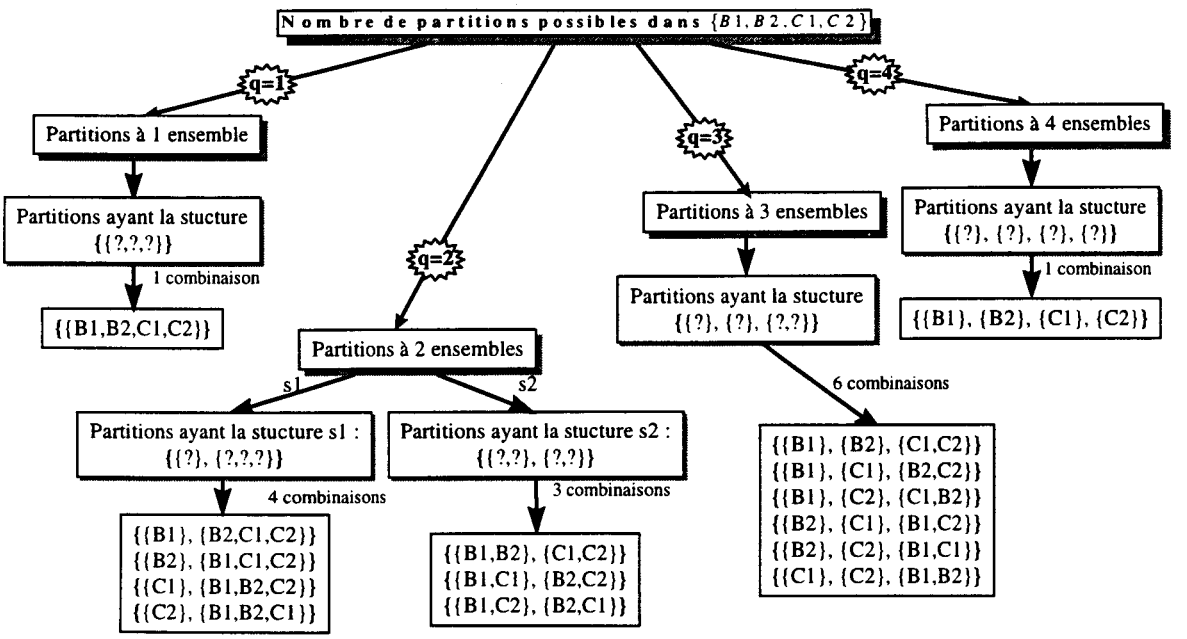


Figure II-9 : Partitions possibles de S_{Prt2}

- Pour q = 1 : il y a une seule partition possible de S_{prt2} est formée d'un seul ensemble,

- pour q = 2 : nous distinguons deux sous-classes différentes s1 et s2 : la première étant formée de deux sous-ensembles (un singleton et un ensemble de 3 éléments) et la seconde formée de deux sous-ensembles chacun de cardinal égal à 2,

- pour le cas q = 3 : une seule classe est possible : elle est constituée de deux singletons et d'un ensemble à deux éléments,

- finalement, le dernier cas est formé d'une seule combinaison contenant 4 singletons.

Pour le cas général, nous considérons l'ensemble S_k formé de N éléments. Pour tout $q \in \{1,...,N\}$, nous allons chercher le nombre C(N,q) de structures différentes dans une partition de N éléments en q sous-ensembles.

Deux partitions Q et Q' sont dites à structures identiques si et seulement si pour chaque ensemble de Q, il existe un ensemble dans Q' ayant le même nombre d'éléments (autrement dit $\forall i \quad l(i) = l'(i)$).

Dans les cas où q = 1 et q = N, il est clair qu'il n'y a qu'une seule structure possible : dans le premier cas c'est un seul ensemble regroupant tous les éléments de S_k et dans le second c'est un ensemble de singletons.

Proposition II.2 :

Le nombre C(N,q) de structures différentes dans une partition de N éléments en q ensembles est donné par la forme suivante :

$$C(q,q)=1\ ,\ C(N,1)=1$$
et
$$C(N,q)=C(N-q,q)+p(N-q,q-1)$$

Equation II-8 : définition récurrente de C(N,q)

Démonstration : cf. Annexe II.

En utilisant le Tableau I-1 pour les valeurs de p(N-q,q-1), on détermine le tableau de valeurs de C(N,q) suivant :

C(N,q)	1	2	3	4	5	6	7	8	9
1	1								
2	1	1							
3	1	1	1						
4	1	2	1	1					
5	1	2	2	1	1				
6	1	3	3	2	1	1			
7	1	3	4	3	2	1	1		
8	1	4	5	5	3	2	1	1	
9	1	4	7	6	5	3	2	1	1

Tableau I-2 : Valeurs de C(N,q)

Nous pouvons alors noter qu'il y a des cases vides. Ces cases correspondent à des valeurs indéterminées ou nulles. En effet, décomposer un entier N en exactement q entiers strictement positifs avec q>N est impossible (car $\forall$ i, a(i) $\geq 1 \Rightarrow \sum_{i=1}^{q} a(i) \geq q > N$ c.q.f.d.), cf. [MOU 97]. Nous pouvons également constater que les éléments de la diagonale sont tous égaux à 1 puisque C(N,N) = 1. Il en est de même pour la première colonne qui correspond à C(N,1) = 1. Donc, en synthèse de cette étape, nous obtenons la procédure suivante :

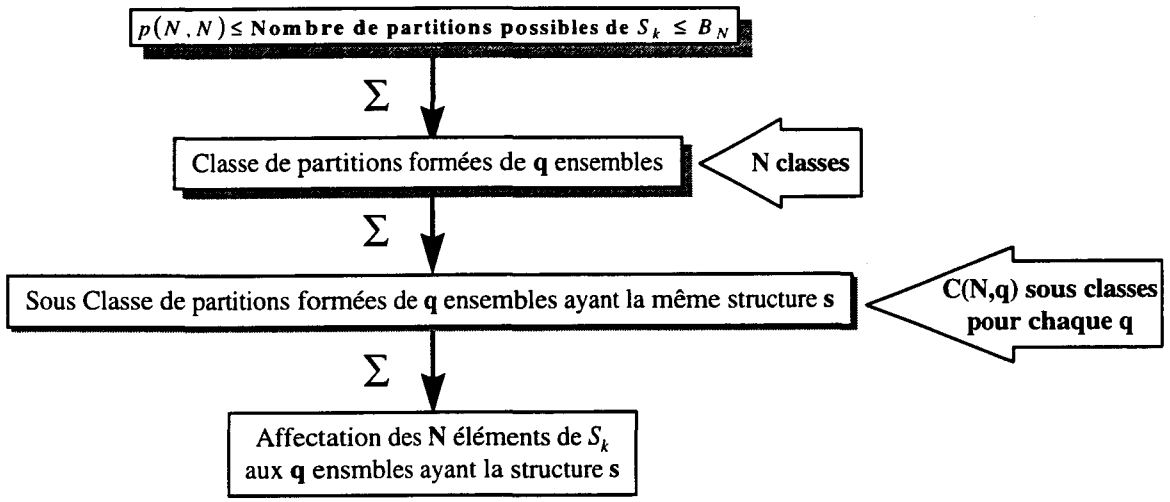


Figure II-10 : Décomposition du problème de partitions de S_k.

Il reste alors à compter le nombre de solutions dans chaque sous-classes pour en faire la somme. Parmi toutes les combinaisons, nous n'en retenons qu'une seule pour la suite du calcul en rappelant que nous devons toutes les considérer pour conserver l'optimalité de la recherche. Dans le cas de l'exemple illustratif, le nombre de solutions provenant du modèle linéarisé choisi à la fin de l'étape précédente, cf. Figure II-7, est égal à 5 partitions possibles de S_{Prt1} et 15 pour S_{Prt2}, soit 75 solutions intermédiaires associées au modèle linéaire donné.

Les combinaisons retenues pour l'étude du regroupement cyclique sont les suivantes : Q' : {{A1, A2, A3}} et Q'' : {{B1, B2, C1, C2}} soit une séquentialisation totale de chaque ensemble. Une fois que toutes les partitions, dans les différents ensembles S_h, sont fixées, il reste à donner un ordre cyclique à chaque composante de chaque ensemble. En effet, les éléments de chaque ensemble d'une partition, vont être acheminés par les mêmes palettes : il s'agit alors de fixer **l'ordre déterministe et unique** du passage de chaque pièce sur l'ensemble des palettes utilisées.

II.6 Regroupement cyclique

Problème : Donner un ordre « cyclique » aux éléments de chaque ensemble provenant d'une partition fixée.

Classe de Décision associée : D_{6d}

Soit une partition $\mathcal{Q}$ donnée et soit un ensemble $\mathfrak{E}$ non vide de cette partition : $\mathfrak{E} = \left\{ \overbrace{G1,...G1}^{\delta_1 \text{ éléments}}, \overbrace{G2,...G2}^{\delta_2 \text{ éléments}}, ..., \overbrace{Gn,...,Gn}^{\delta_n \text{ éléments}} \right\}$. Nous savons que les gammes appartenant à cet ensemble seront effectivement transportées par les même palettes dans un ordre à déterminer. Nous nous proposons d'étudier les différentes combinaisons possibles (appelées également regroupements cycliques ou macro-gammes[1]) concernant les ordres de passage sur les palettes.

Bien entendu, comme nous cherchons à déterminer des solutions différentes nous définissons la classe d'équivalence : Deux macro-gammes sont équivalentes (appartiennent à la même classe) si elles sont **équivalentes** au sens de la **permutation cyclique**.

II.6.1 Détermination du nombre de solutions

A partir de $\mathfrak{E}$ plusieurs macro-gammes (notées M_j) peuvent être déduites. Soit ξ ce nombre de macro-gammes. Nous allons maintenant évaluer ce nombre et présenter un algorithme pour le calculer. Pour le calcul de ξ nous allons procéder par étape en prenant en premier lieu le cas le plus facile à étudier (cas où tous les nombres δ_i, caractérisant le nombre d'exemplaires de la gamme linéaire de type G_i, sont premiers entre eux) puis en généralisant le résultat.

Lemme II.4 :

$$Si\ \text{pgcd}(\delta_1, ..., \delta_n) = 1,\ alors\ \xi = \frac{\left(\sum_{i=1}^{n} \delta_i\right)!}{\prod_{i=1}^{n} (\delta_i)!} * \frac{1}{\left(\sum_{i=1}^{n} \delta_i\right)}.$$

Démonstration : cf. Annexe II

La deuxième étape consiste à considérer le cas où tous les δ_i admettent un diviseur commun premier égal à δ.

[1] Parce qu'elles forment un ensemble de plusieurs gammes linéaires

Lemme II.5 :

Si $\text{pgcd}(\delta_1, ..., \delta_n) = \delta$ *avec* δ ***nombre premier***, *alors* $\xi = \xi_\delta + \xi_1$, *où* ξ_δ *représente le nombre de classes d'équivalences des permutations cycliques d'ordre* $\frac{\sum_{i=1}^{n} \delta_i}{\delta}$ *exactement et* ξ_1 *celui des permutations cycliques d'ordre* $\left(\sum_{i=1}^{n} \delta_i\right)$ *exactement.*

$$\xi_\delta = \frac{\left(\sum_{i=1}^{n} \frac{\delta_i}{\delta}\right)!}{\prod_{i=1}^{n}\left(\frac{\delta_i}{\delta}\right)!} * \frac{1}{\sum_{i=1}^{n} \frac{\delta_i}{\delta}} \quad \textit{et} \quad \xi_1 = \left(\frac{\left(\sum_{i=1}^{n} \delta_i\right)!}{\prod_{i=1}^{n}(\delta_i)!} - \frac{\left(\sum_{i=1}^{n} \frac{\delta_i}{\delta}\right)!}{\prod_{i=1}^{n}\left(\frac{\delta_i}{\delta}\right)!} \right) * \frac{1}{\sum_{i=1}^{n} \delta_i}.$$

Démonstration : cf. Annexe II.

Proposition II.3 :

Si $\text{pgcd}(\delta_1, ..., \delta_n) = \delta$ *et si l'ensemble des diviseurs de* δ *est* $\{d_{\delta 1} = \delta, ..., d_{\delta k} = 1\}$, *alors le nombre total de macro-gammes obtenus à partir de* $\mathfrak{E}$ *est donné par* $\xi = \sum_{i=1}^{k} \xi_{d_{\delta i}}$, *où* $\xi_{d_{\delta i}}$ *représente le nombre de classes d'équivalences d'ordre* $\frac{\sum_{k=1}^{n} \delta_k}{d_{\delta i}}$ *et vaut :*

$$\xi_{d_{\delta i}} = \left(\begin{array}{l} \text{nombre de permutations ordonnées pour } \mathfrak{E}\big|_{d_{\delta i}} \\ - \text{ nombre de permutations ordonnées pour } \mathfrak{E}\big|_{d_{\delta j}} \text{ pour tous les } d_{\delta j} \text{ multiples de } d_{\delta i} \end{array} \right) * \frac{1}{\sum_{k=1}^{n} \frac{\delta_k}{d_{\delta i}}}$$

Démonstration : cf. Annexe II.

Nous avons vu que, pour un ensemble de gammes linéaires regroupées dans une partition, la détermination des classes d'équivalence des permutations cycliques passe

obligatoirement par la connaissance de $\delta = \text{pgcd}(\delta_1, ..., \delta_n)$ et de ses diviseurs $\{d_{\delta 1} = \delta, ..., d_{\delta k} = 1\}$.

Nous avons alors élaboré un algorithme de détermination du nombre de solutions représenté par le graphe, cf. Figure II-11, qui relie tous les diviseurs entre eux. Nous calculons, à chaque niveau (pour chaque diviseur $d_{\delta i}$) du graphe, le nombre de classes différentes d'équivalences : $\xi_{d_{\delta i}}$. Les liens orientés entre deux diviseurs $d_{\delta i}$ et $d_{\delta j}$ signifient que $d_{\delta i}$ est un multiple de $d_{\delta j}$.

Il est donc nécessaire de retirer tous les éléments des classes d'équivalence liés à $d_{\delta i}$ parmi ceux dénombrés avec $d_{\delta j}$ avant de déterminer le nombre exact de classes différentes d'équivalences liées à $d_{\delta j}$. L'algorithme de calcul du nombre de regroupements cycliques obtenus à partir d'un regroupement donné de gammes linéaires est donc le suivant :

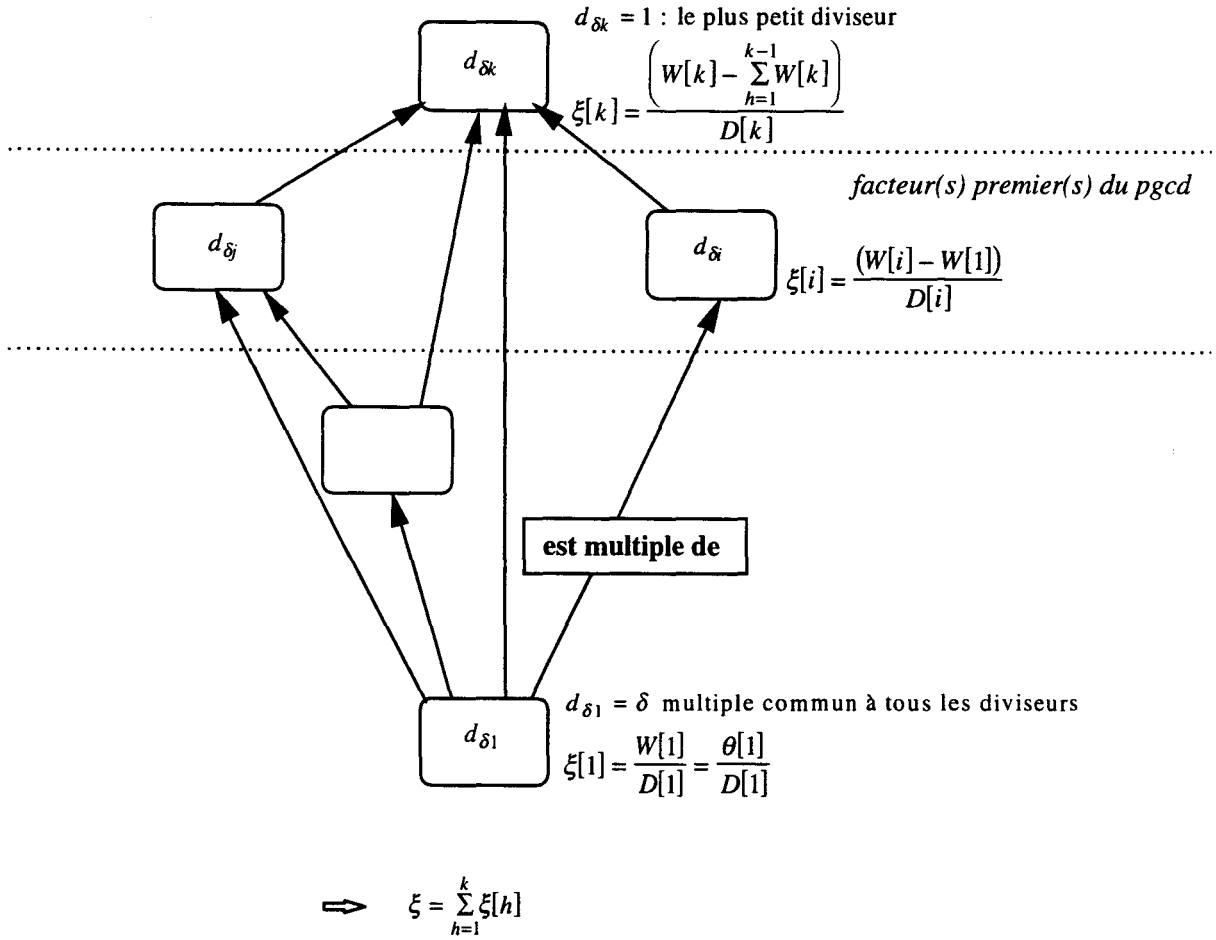


Figure II-11 : Schéma d'analyse du dénombrement des regroupements cycliques

/* Calcul intermédiaire */

- Déterminer $\delta = \text{pgcd}(\delta_1, ..., \delta_n)$.
- Déterminer les diviseurs de λ et les classer dans l'ordre : $d_{\delta 1} > d_{\delta 2} > ... > d_{\delta k}$.

/* Construction du graphe */

- **Pour (1)** i = 1 à k, Faire

  | $D[i] = \sum_{j=1}^{n} \frac{\delta_j}{d_{\delta i}}$

  | $P[i] = \prod_{j=1}^{n} \left(\frac{\delta_j}{d_{\delta i}} \right)!$

  | $\theta[i] = \frac{(D[i])!}{P[i]}$

  | $W[i] = \theta[i]$

  | **Pour (2)** j = 1 à (i-1) Faire

  | Si $d_{\delta j}$ est un multiple de $d_{\delta i}$ alors $W[i] = W[i] - W[j]$

  | **Fin Pour (2)**

  | $\xi[i] = \frac{W[i]}{D[i]}$

- **Fin Pour (1)**
- $\xi = 0$
- Pour i = 1 à k, Faire $\xi = \xi + \xi[i]$.

II.6.2 Algorithme d'extraction des solutions

Après avoir rappelé la méthode de dénombrement des classes d'équivalence des regroupements cycliques, nous allons déterminer un élément de chaque classe. Soit

$\mathcal{R}_1 = \left(\overbrace{G1-...-G1}^{\delta_1} - \overbrace{G2-...-G2}^{\delta_2} - ... - \overbrace{Gn-...-Gn}^{\delta_n} \right)$ un regroupement cyclique donné où Gi représente une gamme linéaire i et δi le nombre d'exemplaires de Gi dans $\mathcal{R}_1$. Nous allons représenter cette classe par la **matrice solution** $\mathcal{X}_1$ (N lignes n colonnes) suivante :

$$\mathcal{X}_1 = \begin{array}{c} \begin{matrix} G1 & G2 & \cdots & \cdots & Gn \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & & & \vdots \\ 1 & 0 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & 0 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & & & \vdots \\ 0 & 1 & 0 & \cdots & \cdots & 0 \\ \vdots & & & & & \\ 0 & \cdots & \cdots & \cdots & 0 & 1 \\ \vdots & \vdots & & & & \\ 0 & \cdots & \cdots & \cdots & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} \} \delta_1 \\ \\ \} \delta_2 \\ \\ \\ \} \delta_n \end{matrix}$$

Dans le cas général, toute combinaison $\mathcal{R}$ possible est représentée par une matrice $\mathcal{X}$ (x_{ij}) N lignes et n colonnes telle que $\begin{vmatrix} x_{ij}=1 \text{ si le } i^{\text{ème}} \text{ élément de } \mathcal{R} = \text{Gj} \\ x_{ij} = 0 \text{ sinon} \end{vmatrix}$. Dans la suite de cette partie nous allons confondre un élément $\mathcal{R}$ et sa représentation matricielle $\mathcal{X}$. Soit la matrice $\mathcal{F}$ de permutation cyclique (N lignes N colonnes) suivante :

$$\mathcal{F} = \begin{bmatrix} 0 & 1 & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & 0 & 1 & 0 & \cdots & \cdots & 0 \\ \vdots & \vdots & 0 & \ddots & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \ddots & 0 & \vdots \\ \vdots & \vdots & & & 0 & 1 & 0 \\ 0 & 0 & \cdots & \cdots & \cdots & 0 & 1 \\ 1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \end{bmatrix}.$$

En effectuant une multiplication à droite de $\mathcal{F}$ par $\mathcal{X}$ on obtient $\mathcal{X}'$ ($\mathcal{F}.\mathcal{X}=\mathcal{X}'$)[1] tel que $\mathcal{X}'$ est le résultat d'une permutation cyclique d'ordre 1([2]) à droite de $\mathcal{X}$. Le résultat d'une multiplication à gauche est une permutation cyclique d'ordre 1 à gauche. Par construction, il est clair que $\mathcal{F}$ est idempotente : $\mathcal{F}^N = I_N$ où I_N est la matrice identité d'ordre N (matrice N*N où tous les éléments sont nuls sauf la diagonale qui est composée de 1).

[1] $\mathcal{F}.\mathcal{X}=\mathcal{X}'$ est le produit matriciel de $\mathcal{M}^{N*N} * \mathcal{M}^{N*n} \rightarrow \mathcal{M}^{N*n}$ qui à un élément $\mathcal{X}$ associe $\mathcal{X}'$ son image par l'endomorphisme $\mathcal{F}$

[2] On décale tous les éléments de $\mathcal{R}$ d'une case à droite sauf le dernier élément qui est envoyé au début.

De plus, il est facilement démontrable que les puissances de la matrice $\mathcal{F}$ forment l'ensemble des permutations cycliques. Nous noterons la relation d'équivalence (permutation cyclique) $\mathcal{R}$.

Lemme II.6 :

Soient $\mathcal{X}$ et $\mathcal{Y}$ deux regroupements cycliques, ils sont équivalents pour la relation d'équivalence $\mathcal{R}$ définie si et seulement si :

il existe un entier naturel $i \in \{0,...,N-1\}$ tel que $\mathcal{Y}$ est l'image de $\mathcal{X}$ par $\mathcal{F}^i$:

$$\mathcal{X} \mathcal{R} \mathcal{Y} \Leftrightarrow \exists\, i \in \{0,...,N-1\} \text{ tel que } \mathcal{F}^i.\mathcal{X}=\mathcal{Y}$$

Démonstration : cf. Annexe II.

Par conséquent la famille de regroupements équivalents à $\mathcal{X}$ (appelée classe d'équivalence de $\mathcal{X}$) est obtenue par : $\{\mathcal{X}, \mathcal{F}.\mathcal{X}, \mathcal{F}^2.\mathcal{X}, ..., \mathcal{F}^{N-1}.\mathcal{X}\}$. Notons que cette famille peut avoir des éléments identiques si $\mathcal{X}$ contient des composantes identiques. Dans ce cas on obtient : $\exists i < N$ tel que $\mathcal{F}^i.\mathcal{X} = \mathcal{X}$.

Proposition II.4 :

$\mathcal{X}$ et $\mathcal{Y}$ étant deux regroupements cycliques.

$$\mathcal{X} \text{ et } \mathcal{Y} \text{ sont non équivalents} \Leftrightarrow \forall\, i \in \{0,...,N-1\} \quad \mathcal{F}^i.\mathcal{X} \neq \mathcal{Y}$$

Démonstration : cf. Annexe II.

II.6.2.1 Algorithme

De ces deux dernières propositions, nous avons déduit un algorithme de détermination d'un seul représentant de chaque classe d'équivalence. A partir d'un élément de départ $\mathcal{X}_0$ fixé, l'algorithme procède par permutation quelconque des composantes de cet élément de départ et conserve le résultat s'il n'est pas équivalent à tous les éléments déjà stockés. En effet, nous savons que tout élément $\mathcal{X}$ contient une seule composante non nulle (égale à 1) par ligne. Pour trouver tous les éléments il suffit de permuter tous les 1 entre eux puis d'éliminer les solutions indésirables.

Enoncé :

$\mathfrak{L}$: liste vide, ξ : nombre de classes d'équivalence.

construire un élément χ_0

| générer les permutations quelconques χ de d'éléments de χ_0

| | générer l'ensemble $\{\chi_i\}$ des éléments de la classe d'équivalence de χ

| | si un élément χ_i de la classe d'équivalence de χ est dans la liste $\mathfrak{L}$

| | | alors quitter la boucle

| | | sinon stocker χ dans $\mathfrak{L}$

| recommencer jusqu'à ce que $\mathfrak{L}$ contienne ξ éléments.

II.6.2.2 Codage

Le codage de cet algorithme a été réalisé en Prolog III. Le codage proposé par la matrice de permutation a permis de manipuler des vecteurs au lieu de matrices ce qui entraîne un coût nettement plus faible du point de vue mémoire et temps de calcul, cf. [MOU 97]. En effet, le fait d'avoir un seul élément non nul dans chaque ligne de la matrice χ nous a donné la possibilité de représenter χ par un vecteur υ à N composantes contenant à la $i^{ème}$ composante la position du « 1 » (l'élément non nul) de la $i^{ème}$ ligne de χ. Par exemple la matrice χ_1 devient $\upsilon_1 = \left[\overbrace{1,\ldots,1}^{\delta_1},\overbrace{2,\ldots,2}^{\delta_1},\ldots,\overbrace{n,\ldots,n}^{\delta_n}\right]$.

Donc, multiplier $\mathfrak{F}$ par χ revient à faire une permutation cyclique sur υ ce qui est beaucoup moins coûteux que de faire N*N*n multiplications et additions : Par exemple $\mathfrak{F}.\chi_1=\chi_1'$ est représenté par υ_1' avec $\upsilon_1' = \left[n,\overbrace{1,\ldots,1}^{\delta_1},\overbrace{2,\ldots,2}^{\delta_1},\ldots,\overbrace{n,\ldots,n}^{\delta_n-1}\right]$.

II.6.3 Application à l'exemple

Le résultat de cette étape est un réseau de Petri, appelé graphe ordonnançable, avec comme indéterminismes restant le séquencement des opérations sur les machines ainsi que

le nombre et la répartition de l'en-cours. Pour notre exemple illustratif, nous avons vu à la fin de l'étape précédente que nous avions fixé les partitions de la façon suivante :

$$Q_1 : \{\{A1, A2, A3\}\} \text{ et } Q'_1 : \{\{B1, B2, C1, C2\}\}.$$

Concernant le premier ensemble nous avons N=n=3 et $\delta_1=\delta_2=\delta_3=1$.

D'où :

$$\text{pgcd}(\delta_1, \delta_2, \delta_3) = 1 \text{ et}$$

$$\xi = \frac{\left(\sum_{i=1}^{3}\delta_i\right)!}{\prod_{i=1}^{3}(\delta_i)!} * \frac{1}{\left(\sum_{i=1}^{3}\delta_i\right)} = \frac{\left(\sum_{i=1}^{3}1\right)!}{\prod_{i=1}^{3}(1)!} * \frac{1}{\left(\sum_{i=1}^{3}1\right)} = \frac{(3)!}{3} = 2 \text{ combinaisons.}$$

Ces deux macro-gammes sont :

$$\mathcal{R}_1=(A1\text{-}A2\text{-}A3) \Leftrightarrow \mathcal{X}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Leftrightarrow \mathcal{V}_1=[1,2,3]$$

$$\text{et } \mathcal{R}_2=(A1\text{-}A3\text{-}A2) \Leftrightarrow \mathcal{X}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \Leftrightarrow \mathcal{V}_2=[1,3,2].$$

Quant au second, nous avons N=n=4 et $\delta_1=\delta_2=\delta_3=\delta_4=1$. D'où

$$\text{pgcd}(\delta_1, \delta_2, \delta_3, \delta_4) = 1 \text{ et}$$

$$\xi = \frac{\left(\sum_{i=1}^{4}\delta_i\right)!}{\prod_{i=1}^{4}(\delta_i)!} * \frac{1}{\left(\sum_{i=1}^{4}\delta_i\right)} = \frac{\left(\sum_{i=1}^{4}1\right)!}{\prod_{i=1}^{4}(1)!} * \frac{1}{\left(\sum_{i=1}^{4}1\right)} = \frac{(4)!}{4} = 6 \text{ combinaisons différentes.}$$

Ces six macro-gammes différentes sont les suivantes :

$$\mathcal{R}_1'=(B1\text{-}B2\text{-}C1\text{-}C2) \Leftrightarrow \Upsilon = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \Leftrightarrow \mathcal{V}_1'=[1,2,3,4],$$

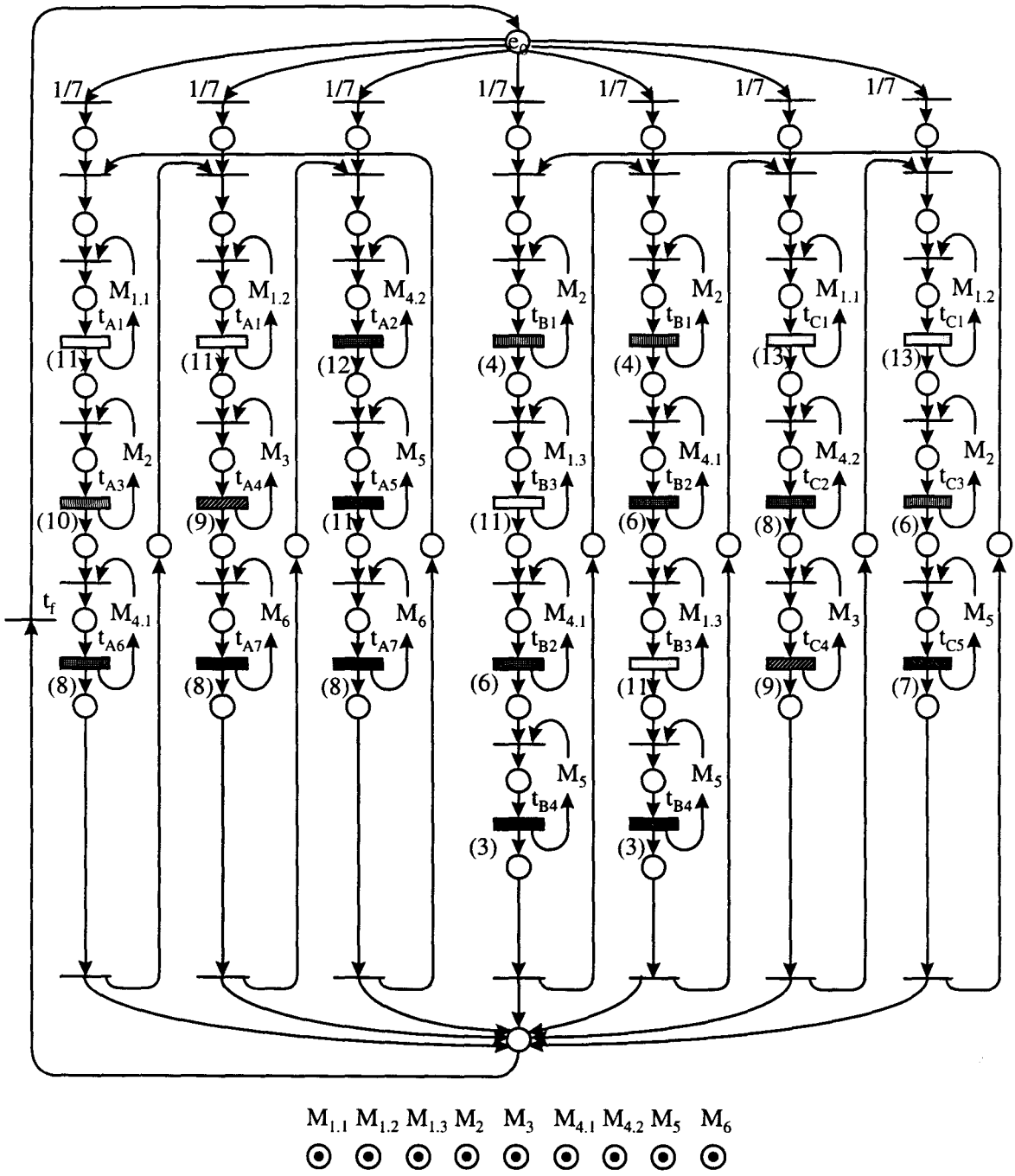


Figure II-12 : Modèle ordonnançable

$$\mathcal{R}_2\text{'}=(B1\text{-}B2\text{-}C2\text{-}C1) \Leftrightarrow \mathcal{Y}_2=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix} \Leftrightarrow \mathcal{O}_2\text{'}=[1,2,4,3],$$

$$\mathcal{R}_3\text{'=(B1-C1-B2-C2)} \Leftrightarrow \Upsilon_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \Leftrightarrow \mathcal{V}_3\text{'=[1,3,2,4]},$$

$$\mathcal{R}_4\text{'=(B2-B1-C1-C2)} \Leftrightarrow \Upsilon_4 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \Leftrightarrow \mathcal{V}_4\text{'=[2,1,3,4]},$$

$$\mathcal{R}_5\text{'=(B2-B1-C2-C1)} \Leftrightarrow \Upsilon_5 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \Leftrightarrow \mathcal{V}_5\text{'=[2,1,4,3]}$$

$$\text{et } \mathcal{R}_6\text{'=(B2-C1-B1-C2)} \Leftrightarrow \Upsilon_6 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \Leftrightarrow \mathcal{V}_6\text{'=[2,3,1,4]}.$$

Il y a donc 6*2=12 graphes ordonnançables différents résultant de la solution choisie à l'étape précédente. Nous avons représenté Figure II-12 celui correspondant à $\mathcal{R}_1$ (qu'on notera macro-gamme M_1^1) et $\mathcal{R}_1$' (qu'on notera macro-gamme M_1^2). Ainsi se termine la démarche d'évaluation de performances. Il nous reste alors à utiliser un algorithme d'ordonnancement pour obtenir la commande déterministe qui définit le régime permanent.

II.7 Synthèse de la démarche et Réduction de la Complexité

Avant d'aborder la phase d'ordonnancement proprement dite, nous allons terminer ce chapitre en synthétisant l'état de la démarche d'évaluation des performances à ce stade de notre étude. Nous avons vu, au cours de l'étape de planification fine, que nous n'avions, pour l'exemple illustratif choisi, qu'un seul régime cyclique à étudier et à optimiser.

Durant la deuxième étape de linéarisation du graphe, nous avons pu montrer sur notre exemple qu'il reste 6 combinaisons possibles pour la flexibilité de permutation (type B) et 3 pour la flexibilité enchaînée (type A). Les deux problèmes étant indépendants, le nombre de solutions intermédiaires à la fin de cette étape est de 18. Nous avons alors poursuivi la

résolution avec une solution (parmi les 18) pour résoudre le problème de partitions de gammes où nous avons alors dégagé 75 solutions intermédiaires !

Enfin, une seule de ces 75 combinaisons donne 12 possibilités de modèles ordonnançables. Nous remarquons donc clairement que la combinatoire est énorme. C'est pour cette raison que nous proposons un ordre de résolution visant à éviter la détermination exhaustive de tous les ordonnancements. Cette méthode consiste à introduire un indicateur de performances pour classer les solutions intermédiaires et ainsi tenter d'éviter une recherche exhaustive, cf. Figure II-13. Cet indicateur est une borne inférieure du nombre de palettes (en-cours) à utiliser, il peut également être calculé directement à partir du Réseau de Petri, cf. [SIF 80].

A partir d'un régime permanent fixé, nous appliquons la phase de linéarisation des gammes. Pour chaque graphe de gammes linéaires nous calculons la première borne minimale B d'en-cours :

$$B(\text{graphe linéarisé}) = \sum_{h=1}^{H} \left\lceil \frac{\sum \text{Durées Opératoires}_{\text{gammes linéaires} \in S_h}}{CT(RP_q)} \right\rceil$$

Les différentes combinaisons de modèles linéarisés sont classées dans l'ordre croissant de leurs bornes min. d'en-cours et seront traitées également dans cet ordre, cf. Figure II-13.

Ensuite, nous analysons le partitionnement des ensembles S_h. Pour chaque partition, nous calculons la nouvelle borne d'en-cours. Cette nouvelle borne est plus significative puisqu'elle prend en compte le regroupement réel des gammes linéaires qui utiliseront les mêmes palettes dont nous déterminerons l'en-cours nécessaire.

$$B(\text{gammes partitionnées}) = \sum_{h=1}^{H} \left(\sum_{\text{partitions de } S_h} \left\lceil \frac{\sum \text{Durées Opératoires}_{\text{gammes linéaires} \in \text{partition } S_h}}{CT(RP_q)} \right\rceil \right)$$

Nous pouvons remarquer que les en-cours nécessaires pour un ensemble de partitions provenant du même graphe linéaire est supérieur ou égal à la borne minimale d'en-cours associée à ce graphe linéaire, cf. Figure II-13. Les regroupements cycliques obtenus à partir de ces partitions possèdent les mêmes bornes minimales d'en-cours que leurs partitions associées étant donné qu'ils possèdent les mêmes distributions de charges, les mêmes

gammes linéaires et les mêmes partitions de gammes. Ces bornes de l'en-cours ont été introduites en raison du fait qu'il n'est pas possible de maîtriser le nombre d'en-cours avant détermination complète du régime permanent (après l'heuristique d'ordonnancement).

L'ordonnancement des graphes ordonnançables établit alors le nombre d'en-cours nécessaire et suffisant pour satisfaire la contrainte du temps de cycle. En conséquence, pour trouver le plus petit nombre d'en-cours suffisant il faut procéder à l'ordonnancement des différents graphes ordonnançables, dans l'ordre croissant de leurs bornes minimales d'en-cours. La recherche est alors arrêtée dans l'un des deux cas suivants :

- l'en-cours trouvé est égal à la borne associée à l'exemple : on sait alors que tous les exemples restant ont des bornes minimales supérieures ou égales à cet en-cours.

- la borne d'en-cours du graphe suivant est supérieure ou égale au plus petit en-cours déjà trouvé (ce qui est le cas pour l'exemple Figure II-13).

Cette classification des solutions intermédiaires est utilisée pour réduire l'espace de recherche sachant qu'il existe des exemples pour lesquels il faudra parcourir la totalité de l'espace des solutions pour trouver la meilleure solution. En pratique, ce gain de complexité dans les exemples les plus courants est en général très significatif.

Nous pouvons également limiter l'ordonnancement au cas qui correspond à la plus petite borne minimale d'en-cours. Ce travail théorique permet alors d'évaluer les performances théoriques de référence. De ce fait, il devient possible d'apprécier le gain en temps de calcul par rapport à la perte d'optimalité quantifiable au sens d'une marge.

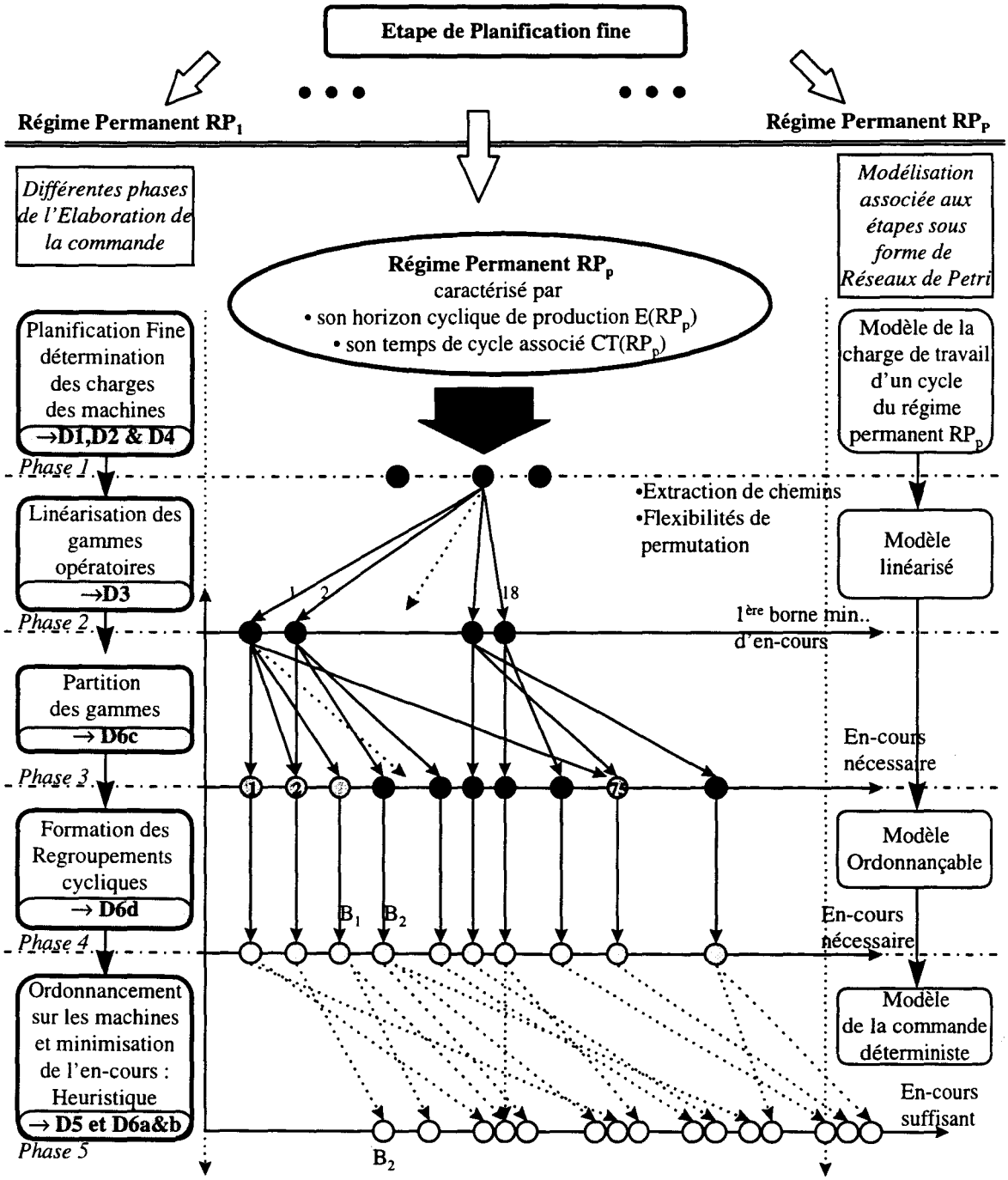


Figure II-13 : Construction de l'espace de recherche et sens du parcours des solutions

II.8 Conclusion

Ce chapitre a été consacré à l'étude de l'approche progressive de l'optimisation des performances. Comme nous l'avions proposé au chapitre précédent, la demande initiale a été décomposée en plusieurs régimes permanents cycliques.

La méthode que nous avons présentée dans ce chapitre consiste à évaluer les performances de chaque régime permanent (en termes de flux de production et d'en-cours nécessaire) puis à transformer progressivement le modèle initial en un graphe d'événements sur lequel il ne restera plus qu'à ordonnancer les ressources de production. Outre la présentation de la résolution, des classes de décision et des flexibilités concernées, une étude mathématique approfondie originale de chaque étape a été proposée. Cette étude a essentiellement permis de déterminer, pour chaque étape, une évaluation du nombre des solutions ou une borne significative de ce nombre. Ainsi, pour la phase de linéarisation des gammes, nous avons déterminé le nombre exact des solutions provenant des flexibilités de permutation ainsi qu'une borne du nombre de solutions des flexibilités enchaînées.

Nous avons présenté ensuite une décomposition du problème de partitionnement des gammes et exprimé le nombre des différentes classes d'équivalence. Puis, nous avons déterminé le nombre de regroupements cycliques dans une partition quelconque et présenté un algorithme pour le calculer. Ensuite, nous avons mis en place un algorithme d'extraction d'un seul élément de chaque classe d'équivalence.

Nous avons donc présenté dans ce chapitre pour chaque phase une nouvelle manière d'appréhender chaque sous-problème ce qui nous a conduit à mieux évaluer l'ampleur de la combinatoire. Cette étude, qui n'a pas réduit la complexité intrinsèque des différents problèmes, a permis d'envisager de nouvelles méthodes de résolution et surtout de réduire le nombre prévisionnel des solutions (c'est-à-dire de diminuer les bornes du nombre de solutions à évaluer à chaque phase), cf. Figure II-14. Sur la Figure II-14, nous présentons la démarche linéaire et progressive d'évaluation et optimisation des performances dans sa nouvelle forme, cf. Chapitre I, en comparaison aux résultats de H. Camus, cf. [CAM 97].

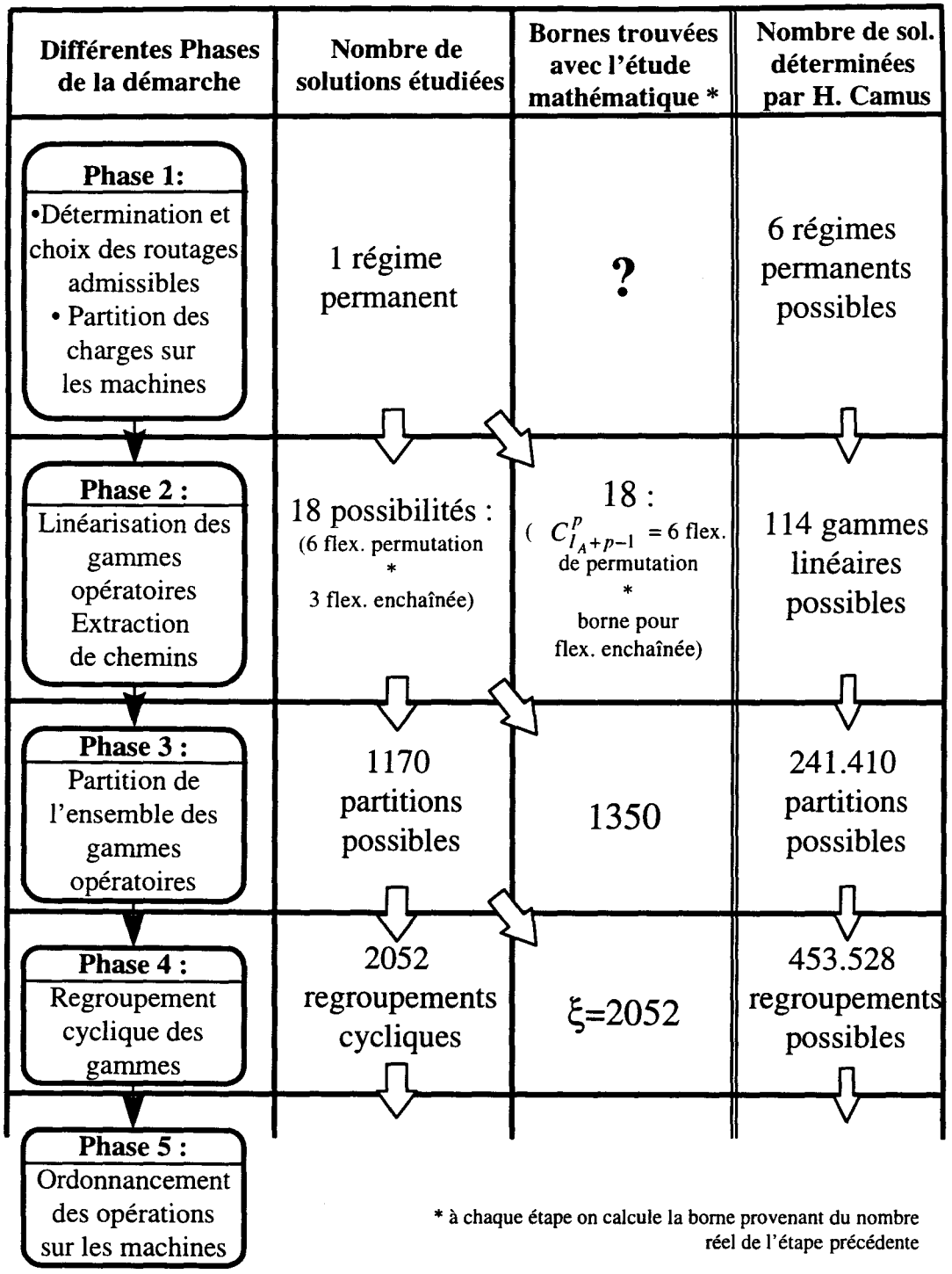


Figure II-14 : Apport de l'étude mathématique de la démarche, cf. Annexe II

Chapitre III

III. Ordonnancement Cyclique

III.1 Introduction

A l'issue de l'approche présentée au chapitre précédent, nous proposons d'étudier le problème d'ordonnancement proprement dit afin de déterminer le séquencement des opérations sur les machines, de calculer les dates de tirs et les en-cours associés à chaque opération. Nous rappelons que notre problème d'ordonnancement a pour principal objectif de respecter le temps de cycle optimal (contrainte stricte) tout en minimisant l'en-cours utilisé (critère secondaire).

III.1.1 Rappels

Les problèmes d'ordonnancement sont connus pour être complexes et hautement combinatoires, cf. [BEL 82]. Il a été prouvé que les problèmes de planification de projets sont de combinatoire *polynomiales* et que l'ordonnancement cyclique est *Non Polynomial complet*, cf. [CAR 88], [SER 89]. L'introduction des ressources de transformation (associées aux différentes opérations) rend le premier problème *Non Polynomial difficile* dans la majorité des cas et laisse le second dans la classe des problèmes *Non Polynomiaux*, cf. [SER 89].

Ainsi, toutes les méthodes de résolution de l'ordonnancement([1]), en général et du cas cyclique en particulier, sont basées sur des méthodes heuristiques ou énumératives. Parmi les principaux résultats dans le domaine de l'ordonnancement cyclique, nous pouvons citer : ordonnancement de la machine critique avec minimisation des tailles des buffers [ERS 82], étude mathématique du problème d'ordonnancement pour la mise en place d'un modèle générique [SER 89], ordonnancement 1-cyclique avec respect du temps de cycle optimal et minimisation de l'en-cours [HIL 89b], ordonnancement k-cyclique pour les systèmes d'information [MUN 91] et [CHR 97], ordonnancement 1-cyclique avec respect

[1] Mis à part l'ordonnancement de flow-shop à deux ou trois machines par exemple.

du temps de cycle optimal et minimisation de l'en-cours [VAL 94], ordonnancement 1-cyclique avec respect du temps de cycle optimal et minimisation de l'en-cours [OHL 95a]).

Nous pouvons également citer le problème du HSP([1]) qui, quoique cyclique, est fondamentalement différent du problème auquel nous nous intéressons, cf. [VAR 96]. En effet, nous considérons que seules les machines peuvent être critiques et que les opérations de transfert se déroulent en temps masqués, alors que pour les problèmes de trempage, seuls les robot de manutention sont critiques. De plus, les opérations sont préemptives et les durées non déterministes mais plutôt contraintes par des intervalles.

Dans le cas qui nous concerne, le passage par une heuristique est inéluctable étant donné la combinatoire énorme du problème. La nécessité de mettre en place un nouvel algorithme est motivée par la volonté de présenter une formulation, du problème de l'ordonnancement cyclique respectant le temps de cycle optimal et minimisant l'en-cours, sans ajouter de contraintes artificielles qui n'auraient de signification dans le système. Dans tous les cas, nous sommes amenés à limiter la combinatoire de la résolution. Cependant, dans le cas d'une bonne formulation du problème, ces limitations seront maîtrisables dans le sens où elles seront paramétrables et connues.

C'est donc ainsi que nous avons mis en place un algorithme d'ordonnancement, cf. [KOR 97a]. Cette heuristique est basée sur des travaux antérieurs (méthodes de Hillion, Valentin et Ohl) afin de mettre en place une formulation qui soit la plus proche possible de celle du problème d'ordonnancement cyclique respectant le temps de cycle et minimisant l'en-cours utilisé.

Néanmoins, cette volonté de minimisation de l'en-cours n'est pas propre à cette phase d'ordonnancement, mais doit être associée aux phases[1] : partition de gammes, regroupement cyclique et ordonnancement de notre méthode de résolution. En effet, cette démarche a été initiée par l'équipe Evaluation de Performances et vise à minimiser l'en-cours. Elle consiste à vouloir profiter pleinement des flexibilités des ressources de transport et à tenter d'associer des pièces de même type aux mêmes palettes. Donc, même si nous allons proposer ici un nouvel algorithme, nous considérons que cette phase reste indissociable de la méthode générale d'évaluation des performances et d'ordonnancement.

---

[1] Hoist Scheduling Problem : problèmes de galvanoplastie.

L'ordonnancement cyclique recherché est 1-cyclique par rapport aux machines, c'est-à-dire que chaque machine reprend exactement le même état et le même fonctionnement à chaque cycle. L'algorithme, que nous chercherons à mettre en place, aura également pour objectif de déterminer l'ordonnancement cyclique du régime permanent en un tour du programme. Ceci veut dire qu'il est tenu d'obtenir un **ordonnancement admissible** à la fin de son exécution. Cet ordonnancement est possible, cf. [COM 71], puisqu'il suffit de marquer toutes les places des différentes gammes du graphe ordonnançable pour saturer les machines critiques et fonctionner à flux maximal.

Nous rappelons que nous disposons, pour chaque modèle ordonnançable, d'une borne d'en-cours, cf. Chapitre II. Le problème est que cette borne est une condition nécessaire mais pas suffisante. **Nous ne connaissons donc le nombre d'en-cours réel à utiliser qu'après la phase d'ordonnancement**.

III.1.2 Notations et définitions

Afin de faciliter la compréhension de l'algorithme d'ordonnancement, présenté dans ce chapitre, nous allons introduire les notions indispensables pour le placement des opérations. La plupart des ces définitions ne sont utiles que pour **cette phase**.

III.1.2.1 Modélisation de l'ordonnancement

Nous allons modéliser notre ordonnancement cyclique déterministe par une sous-classe des Réseaux de Petri : le Graphe d'Evénements, cf. [COM 71], [HIL 89b] et [LAF 91] pour la modélisation et [LAF 92] pour les propriétés. De plus, nous utiliserons tout au long de la phase d'ordonnancement, le Diagramme de Gantt Dual pour représenter l'ordonnancement cyclique, cf. [OHL 94] et [OHL 95b].

Ce diagramme de Gantt Dual est de largeur minimale égale au temps de cycle optimal afin de représenter au moins une période de fonctionnement. Nous retrouvons les deux points de vues privilégiés de la représentation adoptés dans le projet CASPAIM : le point de vue machines et le point de vue en-cours ou palettes.

Les machines ont un comportement 1-cyclique, cf. Figure III-1, ce qui n'est pas nécessairement le cas des en-cours. En effet, nous avions avancé, dès le premier chapitre,

[1] Ce sont les phases 3, 4 et 5 de notre méthode structurées de résolution, cf. figure I-7.

que nous supposions les ressources de transport, de type palettes, associées individuellement aux pièces pendant toute la durée de leur production. Cette hypothèse nous a permis de confondre la minimisation de l'en-cours (pièces brutes ou semi-finies dans le système) avec la minimisation des ressources de transport (palettes).

Ainsi, pour une pièce donnée, cf. Figure III-1, il est possible d'avoir p en-cours associés dans le système pour voir une pièce finie sortir, par cycle (pour cet exemple $p = 4$). Ce nombre p représente alors le nombre de palettes utilisées pour produire A.

En conséquence, bien que nous ayons effectivement une sortie de pièce A finie par cycle et une entrée d'une pièce A brute par cycle, la durée réelle de fabrication de la pièce A est de p_A cycles. Le fonctionnement du système est alors d'ordre de cyclicité égal à $P = p.p.c.m.(p_A, p_B, ...)$ cyclique par rapport à l'en-cours (c'est-à-dire que nous retrouvons effectivement strictement le même état des palettes tous les P cycles). La production reste cependant 1-cyclique par rapport aux machines.

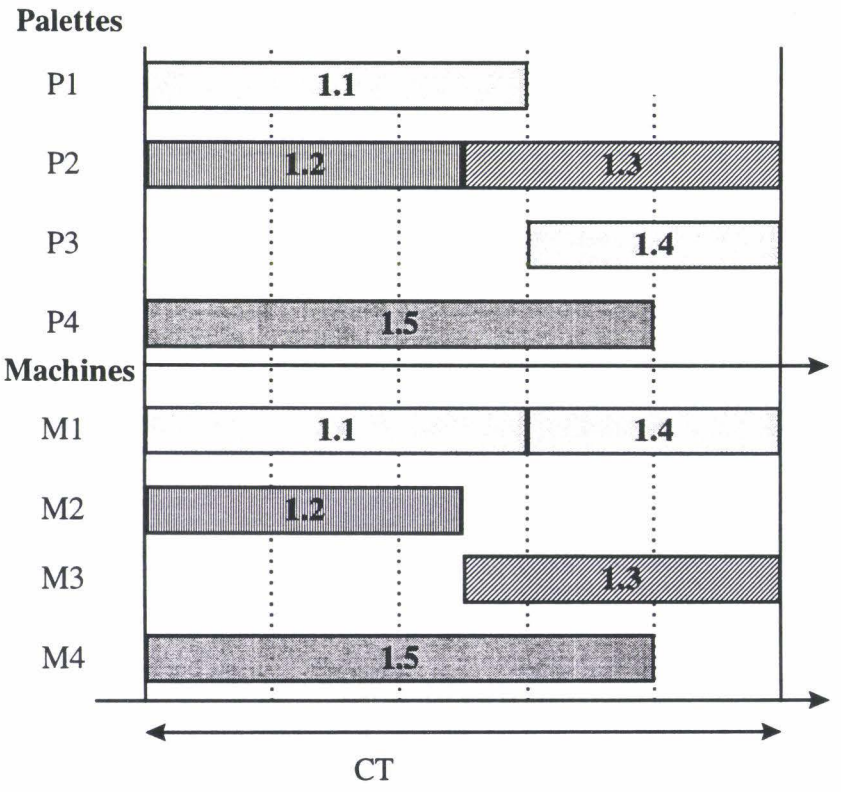


Figure III-1 : Diagramme de Gantt Dual

Ce phénomène est dû au fait que, souvent, la longueur temporelle d'une gamme (composée des durées opératoires et des temps d'attente sur les machines) est supérieure au

temps de cycle. Il faut alors utiliser « assez d'en-cours » pour respecter ce temps de cycle optimal et saturer les machines menantes. En fait tout se passe comme si, durant un cycle, chaque partie de la pièce était virtuellement et simultanément produite sur plusieurs palettes différentes. En réalité, la pièce est bel et bien produite sur la même palette, néanmoins, durant le cycle *i* nous voyons sortir la pièce lancée durant le cycle *i-n+1*, cf. Figure III-1.

III.1.2.2 Date de disponibilité

La date de disponibilité d'une pièce est la date à laquelle la pièce et sa palette associée quittent une machine et deviennent donc disponibles pour subir la prochaine transformation. Cette date correspond donc à la date de fin de la dernière opération engagée. Cette notion est indispensable pour savoir à quel moment la pièce est disponible et prête à passer à la prochaine machine.

III.1.2.3 Marge de gamme

Nous avons vu que, pour chaque macro-gamme, nous disposons d'une borne d'en-cours nécessaire pour respecter le débit maximal([1]). Ainsi, au début de l'ordonnancement nous disposons d'une certaine marge égale à :

nombre d'en-cours évalué[2] * CT - somme des temps opératoires.

Ensuite, durant l'ordonnancement, nous pouvons être amené à attendre qu'une machine ait terminé une autre opération, avant d'engager celle qui nous intéresse. Ce temps d'inactivité, non récupérable par la suite, est perdu, il est donc à soustraire de la marge initiale. Par conséquent, si cette marge devient négative, il faut augmenter l'en-cours nécessaire pour maintenir le temps de cycle. Dès que nous aurons dépassé ou atteint l'en-cours prévu (la borne inférieure d'en-cours), la marge de gamme devient :

marge de gamme = CT - date de disponibilité de la pièce

[1] Cette borne est obtenue par la somme des temps opératoires des opérations, de la macro-gamme, divisée par le temps de cycle optimal et arrondie à l'entier supérieur ou égal.

[2] A cet instant il est égal à la borne inf. de l'en-cours de la macro-gamme.

III.1.2.4 Chevauchement de cycles

Dans les approches antérieures d'ordonnancement cyclique (Hillion, Valentin, Ohl), les auteurs proposaient d'ajouter une palette dès lors que le temps restant sur un en-cours est inférieur au temps opératoire de l'opération suivante. Sur la Figure III-1 nous avons représenté un exemple d'ordonnancement de ce type (appelé également sans chevauchement de cycles) d'une gamme utilisant 4 palettes. Nous pouvons remarquer que l'opération 1.2 nécessite un nouvel en-cours car le temps restant sur le premier est inférieur à sa durée.

Afin de minimiser l'en-cours nécessaire, et de profiter pleinement du caractère cyclique de la commande, nous avons introduit la notion de chevauchement de cycles, cf. [KOR 97c]. Cette notion permet de pouvoir lancer une opération durant un cycle pour la terminer le cycle suivant. Sur l'exemple Figure III-2 nous avons représenté l'ordonnancement de la même gamme de la Figure III-1 mais avec chevauchement de cycles. Nous pouvons ici observer que l'opération 1.2 est placée à la suite de 1.1, bien qu'*empiétant* sur le cycle suivant, l'idée de chevauchement des cycles a permis de diminuer l'encours utilisé.

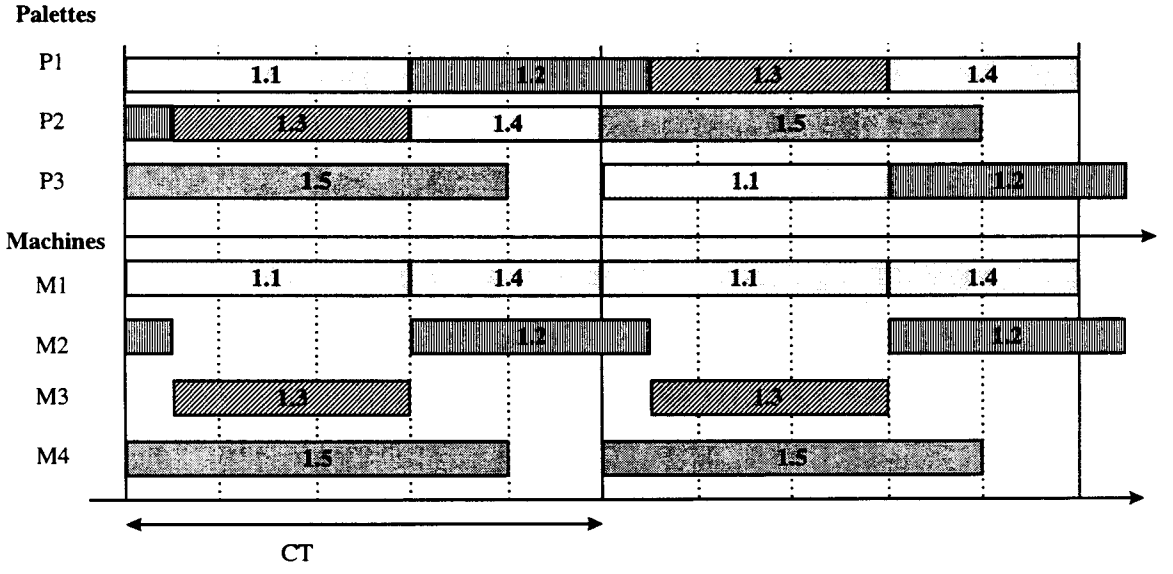


Figure III-2 : Intérêt du chevauchement des cycles

III.1.2.5 Intervalle de disponibilité

Le principe de chevauchement de cycles nous oblige donc à ne pas considérer une fenêtre figée de largeur CT pour y ordonnancer les différentes opérations. En effet, nous

risquons à chaque instant de « sortir » de cette fenêtre pour terminer une opération démarrant dans la fenêtre en question et terminant dans le cycle suivant.

Ce problème se répercute inéluctablement sur les disponibilités des machines. Par ce terme nous entendons les intervalles de temps pendant lesquelles les machines ne sont pas occupées. Il est clair que, durant l'ordonnancement, ces disponibilités sont indispensables pour le placement des opérations restantes. En conséquence, nous définissons les intervalles de disponibilité d'une machine comme l'ensemble des intervalles de temps où la machine est inactive, cf. Figure III-3. Dans cet ensemble nous ne considérons que les intervalles de durée supérieure ou égale à la plus petite opération restante c'est-à-dire que nous ne gardons que les intervalles susceptibles de contenir au moins une des opérations restant à ordonnancer. Nous rappelons que les opérations ne sont pas préemptives, qu'elles ne sont pas divisibles et qu'elles doivent être réalisées en une seule fois.

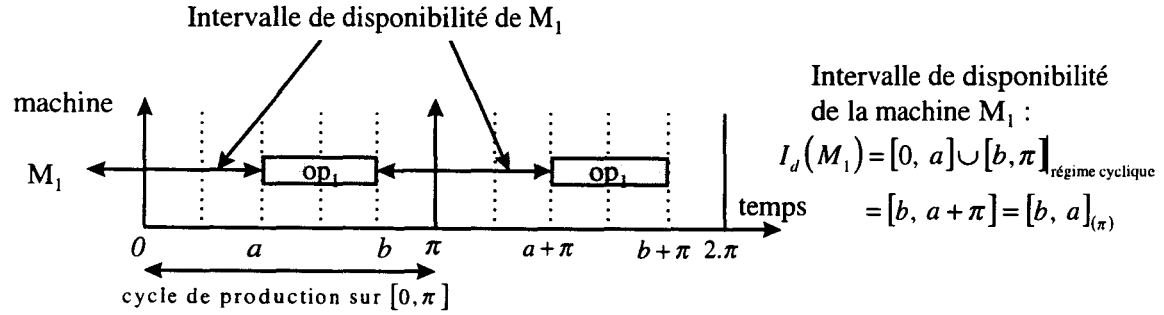


Figure III-3 : définition d'un intervalle de disponibilité

III.1.2.6 Marge de machine

Ainsi, que nous l'avons illustré dans le paragraphe précédent, il est possible d'éliminer un intervalle trop petit pour contenir l'une des opérations restantes sur les machines. La durée de cet intervalle éliminé est alors perdue pour la suite de l'ordonnancement. Dès l'étape de planification fine, nous connaissons parfaitement la charge de chaque machine (par cycle) qui est bien entendu inférieure ou égale au temps de cycle.

Ainsi, chaque machine admet, initialement, une marge « de manœuvre » positive (nulle pour les machines critiques). Cette marge est égale au temps de cycle diminué de la charge cyclique de la machine. Nous rappelons que nous cherchons à obtenir un ordonnancement en une itération du programme, nous définissons donc la marge « courante » de chaque machine :

marge de machine = marge initiale - durée des intervalles perdus

Cette marge ne doit en aucun cas devenir négative. En effet, si cette marge devient négative ceci veut dire que la somme des intervalles de disponibilité restant ne peut contenir la somme des opérations à réaliser sur la machine et par conséquent que l'ordonnancement courant n'est pas admissible.

III.2 Algorithme d'ordonnancement

Dans ce mémoire nous n'allons pas nous focaliser sur le principe de l'algorithme, cf. [KOR 97c]. Nous insisterons plutôt sur notre contribution à la résolution de ce problème et notamment sa programmation. Nous indiquerons quelles sont les solutions proposées pour la mise en œuvre de l'application ainsi que les performances du programme. Nous rappelons que les données de l'algorithme sont initialement réduites aux séquences linéaires d'opérations (macro-gammes) et que le temps de cycle et les bornes inférieures de l'en-cours sont calculées par l'application.

III.2.1 Principe

L'heuristique d'ordonnancement est basée sur quatre notions fondamentales que sont le placement progressif des opérations (selon des stratégies de placement), l'exploration en profondeur des solutions, les fonctions de coût et la validité d'un ordonnancement courant. Nous allons passer en revue ces notions, en expliquer le fonctionnement et présenter les modules chargés d'effectuer ces opérations.

III.2.1.1 Politiques de placement

Afin d'être sûr d'obtenir l'ordonnancement cyclique optimal, qui ne relève pas nécessairement d'une stratégie de placement au plus tôt, nous devons essayer, pour chaque opération, toutes les dates de placement possibles. En effet, un placement qui nous fait perdre le maximum de marge pour une gamme donnée peut s'avérer payant pour la suite et donner le meilleur en-cours global. Un tel choix est bien évidemment impossible à adopter compte tenu de l'énorme combinatoire qu'il engendre.

Nous devons donc proposer *un compromis raisonnable* selon un nombre limité de méthodes de placement, pour réduire la combinatoire, sans réduire, si possible, les

opportunités d'une bonne optimisation. Afin d'expliquer les stratégies de placement, nous introduisons l'exemple de la Figure III-4. Dans cet exemple, deux opérations sont déjà placées et trois restent à ordonnancer. La prochaine opération ($op_{1,3}$) est à ordonnancer sur la machine M_1 qui possède un intervalle de disponibilité $I_{d1}(M_1)$.

- Placement **au plus tôt dans la gamme** : cette politique de placement est utilisée par la plupart des heuristiques d'ordonnancement cyclique. En effet, il peut sembler naturel d'utiliser cette approche dont on connaît l'optimalité pour les Graphes d'Evénements. Ce n'est hélas pas le cas pour les RdP plus généraux.

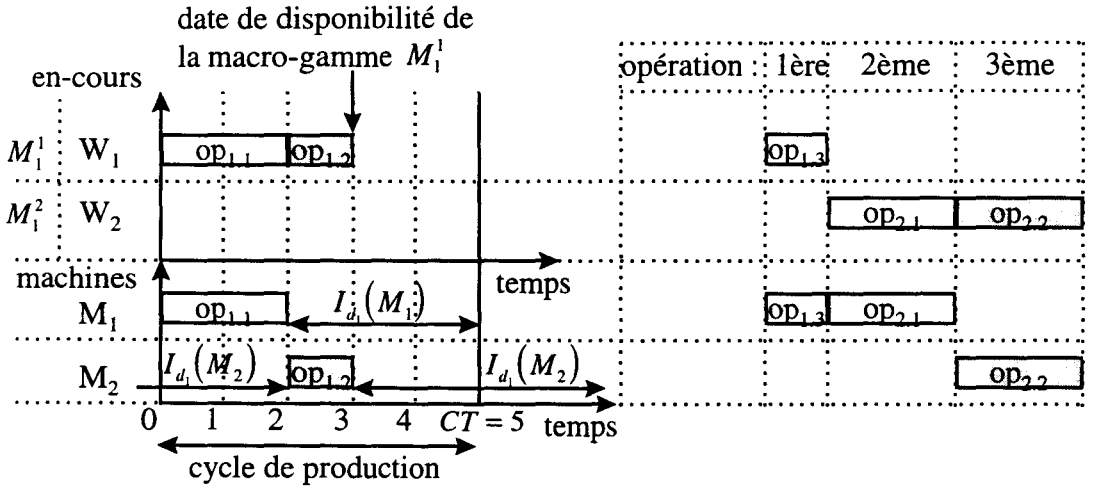


Figure III-4 : Politiques de placement

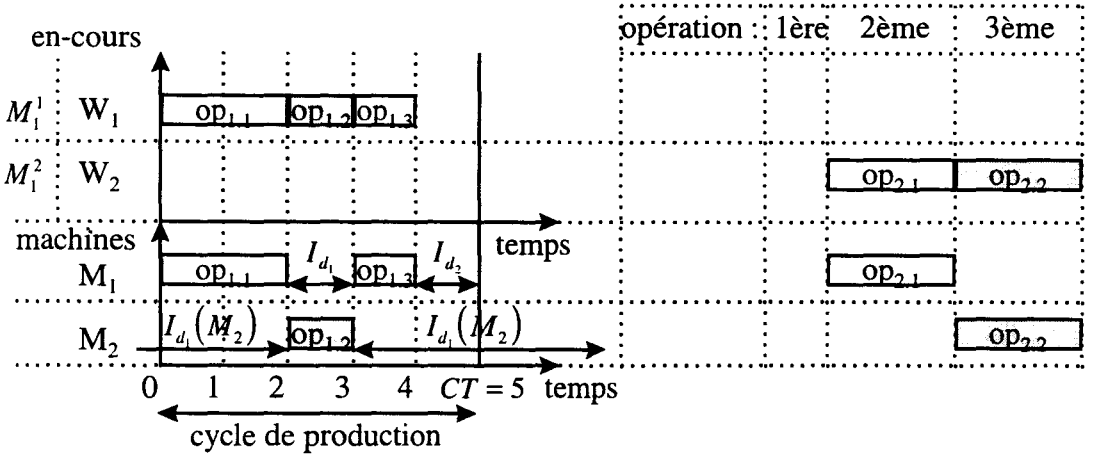


Figure III-5 : Placement au plus tôt dans la gamme

Cette stratégie permet de perdre le minimum possible de marge de gamme et donc de consommer le minimum d'en-cours. Il arrive, cependant, dans le cas général, que le placement au plus tôt dans la gamme se situe au milieu d'un intervalle de disponibilité

machine et qu'il génère en conséquence deux intervalles. Par exemple le placement au plus tôt de l'opération 1.3 remplace l'intervalle de disponibilité initial de M_1, cf. Figure III-4, par deux intervalles I_{d1} et I_{d2}, cf. Figure III-5.

Ces intervalles sont tous les deux plus petits que l'opération $op_{2.1}$ restante sur la machine M_1. L'ordonnancement n'est donc pas admissible (la marge de machine devient négative). Cette politique de placement n'est donc pas totalement satisfaisante. Il nous faut donc proposer d'autres méthodes capables de garantir la faisabilité de l'ordonnancement en une exécution de programme.

- Placement **au plus tôt dans un intervalle** de disponibilité : cette stratégie peut permettre de ne pas perdre de marge de machine en plaçant l'opération au plus tôt dans un intervalle de disponibilité. Cependant si l'intervalle choisi est à peine plus grand que l'opération en question, le nouvel intervalle de disponibilité peut être plus petit que les opérations restantes et la faisabilité de l'ordonnancement peut se trouver altérée.

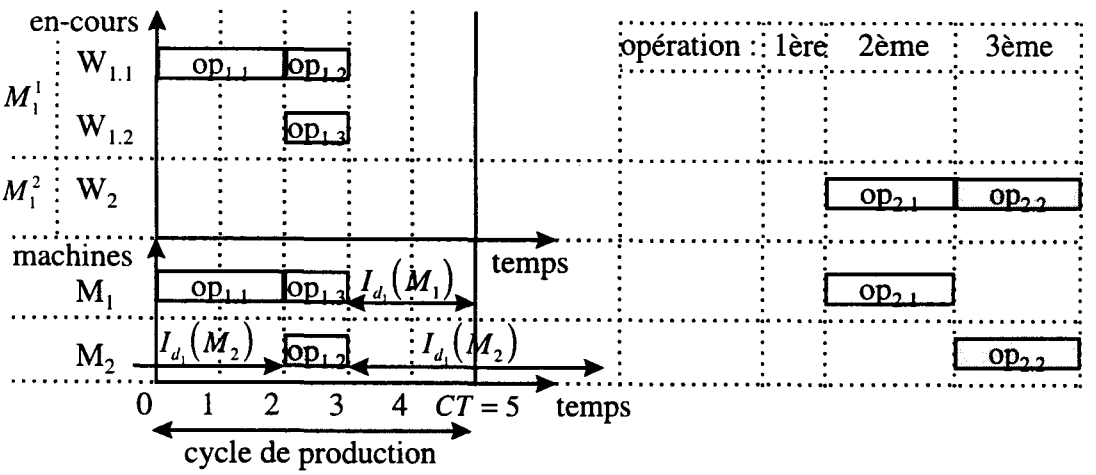


Figure III-6 : Placement au plus tôt dans un intervalle de disponibilité

Cette procédure devra donc être appliquée à tous les intervalles de disponibilité de la machine. Sur l'exemple de la Figure III-6, où l'en-cours a été augmenté, nous remarquons que le placement au plus tôt de l'intervalle de disponibilité initial donne lieu à un seul nouvel intervalle assez grand pour contenir l'opération restante : l'ordonnancement est donc admissible à ce niveau d'itération.

Nous pouvons également remarquer que ce placement a engendré l'ajout d'un en-cours supplémentaire. Nous introduirons donc une troisième et dernière stratégie de placement.

- Placement **au plus tard dans l'intervalle** : cette stratégie permet de placer les opérations à la fin des intervalles de disponibilité, cf. Figure III-7.

Nous pouvons remarquer sur notre exemple que ce placement a permis de ne pas consommer de la marge de gamme et en même temps de ne pas ajouter d'en-cours. Bien entendu cette méthode de placement n'est pas non plus systématiquement la meilleure des trois procédures proposées. Il faut alors tester les trois politiques de placement pour chaque opération. Il faudra donc appliquer cette procédure à tous les intervalles de disponibilité de la machine considérée.

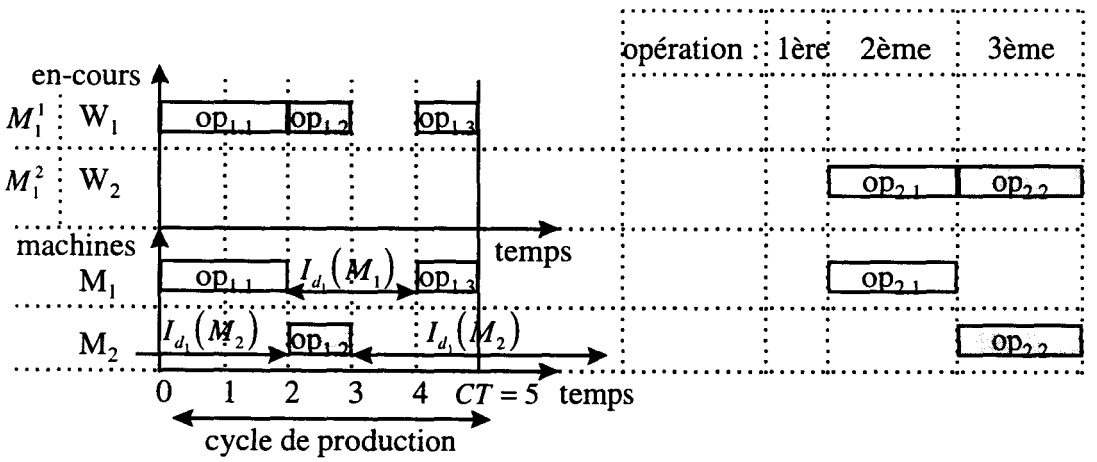


Figure III-7 : Placement au plus tard dans un intervalle de disponibilité

Le placement de chaque opération selon les différentes opérations est réalisé par un module que nous appelons *Tir*. Ce module est également chargé d'initialiser les intervalles de disponibilité. En effet, tant que la machine n'a pas encore été utilisée elle ne possède pas d'intervalle de disponibilité (ceci revient à l'initialiser à $]-\infty,+\infty[$).

Dès qu'une première opération se présente pour être ordonnancée sur cette machine, il n'y a qu'une seule possibilité à savoir le placement au plus tôt de la gamme([1]). C'est ainsi que le premier intervalle de disponibilité s'initialise à [date de fin de l'opération, date de début + CT].

[1] Vu que la machine n'a pas encore été utilisée il n'y a donc pas d'intervalle de disponibilité à considérer.

III.2.1.2 Existence d'un ordonnancement admissible

Le compromis entre la « limitation de la combinatoire » et « l'optimalité de la solution » que nous venons d'évoquer ne doit pas nous faire oublier un des points les plus importants de notre cahier de charge. En effet, après avoir testé les trois politiques de placement nous devons vérifier que l'algorithme est toujours capable de trouver une solution en une exécution du programme. Pour chaque opération à placer, les trois méthodes seront donc testées. Ainsi, il existe, dans l'arbre de recherche, une branche formée uniquement de placements au plus tôt dans l'intervalle de disponibilité. Nous avons également prouvé que cette stratégie de placement n'augmente pas le nombre d'intervalles de disponibilité des machines (elle remplace l'intervalle initial par un autre strictement inclus dans le premier).

Par conséquent, dans cette branche chaque machine aura au plus un intervalle de disponibilité et toutes les opérations seront placées les unes contre les autres sans perte de marge de machine. Cette branche ne donnera vraisemblablement pas l'ordonnancement ayant le meilleur en-cours. Elle fournira néanmoins un ordonnancement **admissible** respectant le temps de cycle optimal. Par la suite, la richesse de composition des politiques de placement combinées permettra de minimiser l'en-cours.

Pour une itération donnée de l'algorithme, la validité du placement de l'opération suivante est vérifiée par un module spécifique. Le module que nous appelons *Possible* tente un placement rapide et brutal des opérations restantes dans les intervalles de disponibilité de la machine considérée. Si cette contrainte est respectée c'est que l'ordonnancement est encore potentiellement réalisable sinon le placement de l'opération courante n'est pas validé et l'on teste la branche suivante.

III.2.1.3 Recherche par faisceaux

Pour la détermination de la « meilleure » opération à ordonnancer à chaque étape nous effectuons une recherche par faisceaux. En effet, l'utilisateur commence par choisir une profondeur de recherche π. Ensuite, l'algorithme se charge de trouver, parmi les opérations restantes, toutes les séquences d'opérations possibles de longueur π. Ces séquences doivent respecter les contraintes de précédence des macro-gammes.

Cette recherche par faisceaux est destinée à limiter la combinatoire de la recherche en considérant un nombre limité d'opérations sachant que si la profondeur de recherche

initiale est supérieure au nombre total d'opérations alors l'algorithme traite toutes les opérations d'un coup et le résultat obtenu est « optimal » par rapport à la réduction de complexité choisie. En somme, nous avons la possibilité de limiter la combinatoire ou à l'inverse demander la profondeur maximale. Cependant, déterminer, à une itération donnée, toutes les séquences de profondeur π ne revient pas à considérer (Nombre de macro-gammes)$^{\pi}$ cas différents, cf. [OHL 95a]. En effet, parmi ces solutions certaines séquences sont équivalentes et donnent lieu au même ordonnancement. Deux séquences sont équivalentes si elles présentent les deux conditions suivantes :

1. elles sont formées des mêmes opérations
2. elles correspondent à une permutation d'un certain nombre d'opérations appartenant à des gammes différentes et à des machines différentes.

En fait, la permutation de deux opérations, de deux macro-gammes différentes, nécessitant deux machines différentes ne modifie aucunement l'ordonnancement car les deux opérations sont totalement indépendantes. Le module Séquence se charge de former toutes les séquences de longueur π et de garder un représentant de chaque ensemble de séquences équivalentes. Chaque séquence est évaluée et l'on ne conserve que la meilleure solution en plaçant effectivement la première opération([1]) pour relancer ensuite la recherche.

III.2.1.4 Fonctions de coût

Afin d'évaluer les séquences et de choisir la « meilleure » d'entre elles, nous proposons de mettre en place une fonction de coût prenant en compte l'ensemble des contraintes et paramètres du problème. Ces paramètres sont l'en-cours à minimiser, les marges de gammes à ne pas « gaspiller », et les marges de machines à ne pas rendre négatives. La fonction de coût est donc une somme pondérée des ces trois fonctions :

[1] Dans le cas où la profondeur initiale est supérieure ou égale au nombre d'opérations nous plaçons toute la séquence trouvée car nous savons qu'elle est optimale (vu qu'on a considéré l'ensemble des opérations).

$$\text{Coût d'une séquence} = \begin{array}{l} \text{Cwip} * \text{Coût en-cours} + \\ \text{Cmacro} * \text{Coût macro-gammes} + \\ \text{Cmachine} * \text{Coût machines} \end{array}$$

Equation III-1 : Fonction de coût

Notons aussi que le caractère discret du problème conduit à des évolutions non monotones selon l'influence des différents paramètres et pondérations. Nous avons donc laissé le choix de ces pondérations à l'initiative de l'utilisateur. De même, les fonctions de coût ne sont ni uniques ni universelles, nous en avons testé quelques unes avant de retenir les fonctions suivantes qui donnent de bons résultats avec les exemples que nous avons traités, cf. [KRO 97] :

- en-cours : la minimisation de l'en-cours est l'objectif principal de l'heuristique. Nous avons vu dans le chapitre précédent que nous sommes en mesure de déterminer un en-cours minimal pour le respect du flux optimal (borne inférieure de l'en-cours). Ainsi, nous pénaliserons l'ajout d'un en-cours, tant qu'il reste en deçà de cette borne. Les en-cours supplémentaires contribueront à accroître d'une manière conséquente le coût :

$$\text{Coût en-cours} = \begin{cases} \dfrac{1}{\text{borne inf.}} & \text{si en-cours courant} \le \text{borne inf.} \\ 1 & \text{sinon} \end{cases}$$

Equation III-2 : Coût de l'en-cours.

- Marges de gammes : toujours dans le souci d'utiliser un minimum d'en-cours nous voulons céder le moins possible de marge de gamme. En effet, la perte de marge de gamme peut entraîner, par la suite, l'ajout d'un en-cours. Donc, plus nous perdons de la marge de gamme plus nous pénaliserons le coût « macro » :

$$\text{Coût macro(Macro)} = \frac{\sum \text{opérations placées(Macro)} + \sum \text{marges perdues(Marco)}}{\sum \text{opérations placées(Macro)}}$$

Equation III-3 : Coût de la marge de gammes

- Marges de machines : comme pour les deux fonctions précédentes nous cherchons à pénaliser la perte de marge de machines([1]) :

$$\text{Cout machine(M)} = \frac{CT - \sum \text{Intervalles dispo.(M)} + \sum \text{opérations restantes(M)}}{CT}$$

Equation III-4 : Coût de la marge de machines

Cette fonction est majorée par 1 (c'est d'ailleurs sa valeur pour les machines menantes). Ainsi, plus sa valeur tend vers 1 plus le placement devient contraint (la marge de manoeuvre est réduite).

Ainsi, pour chaque séquence et pour chaque combinaison de placements de ses opérations, le module *coût* détermine la valeur de la fonction coût qui sera utilisée pour évaluer la qualité de la solution.

III.2.2 Mise en oeuvre

Après avoir présenté notre approche, nous sommes en mesure de présenter l'heuristique d'ordonnancement selon la procédure générale suivante :

Tant que (1) {opérations à ordonnancer} ≠ ∅

Déterminer {séquences non équivalentes}

Tant que (2) {séquences non équivalentes } ≠ ∅

Déterminer toutes les possibilités de placement pour chaque opération de la séquence

Tant que (3) il reste des placements différents

placer « virtuellement » les opérations selon les politiques associées

Si ordonnancement admissible alors

déterminer le coût du placement de la séquence

mémoriser la séquence et son placement si elle est meilleure

[1] C'est le module *Possible* qui veille à ce que cette marge soit toujours positive, ici nous nous occupons uniquement à pénaliser la perte de la marge de machines.

Fin Si

Fin Tant que (3)

{séquences non équivalentes } = {séquences non équivalentes} \ {séquence testée}

Fin Tant que (2)

Placer la première opération de la meilleure séquence

{opérations à ordonnancer} = {opérations à ordonnancer} \ {opération placée}

Fin tant que (1).

Les modules auxquels l'application fait appel sont les modules d'*initialisation*, d'*affichage*, de *saisie*, de *sauvegarde* ainsi que les modules suivants :

Séquence : à partir de l'ensemble des opérations restantes et de la profondeur π on détermine toutes les séquences d'opérations de longueur π non équivalentes.

Tir : à partir d'une séquence donnée par le module précédent on détermine toutes les combinaisons de placements différents pour chaque opération.

Place : pour une séquence donnée et une combinaison de placements on détermine toutes les dates de tir des opérations ainsi que les en-cours associés.

Possible : détermine si le placement de la séquence est admissible ou non (test de la faisabilité de l'ordonnancement).

Coût : détermine le coût d'un ordonnancement admissible d'une séquence.

Calcul : procédure principale de l'heuristique, place la première opération de la meilleure séquence et recommence le calcul.

Le codage de l'application a été réalisé en langage C (environnement BORLAND) sous DOS. Les premiers tests ont révélé une combinatoire trop importante. Pour cette raison nous avons décidé de limiter la combinatoire avec une règle simple de bon sens. En effet, nous avons affirmé dans la théorie que nous devions tester les politiques de placement au plus tôt et au plus tard dans les intervalles de disponibilité pour tous les intervalles disponibles ce qui conduit à une combinatoire de l'ordre de (1+2*nombre d'intervalles disponibles) pour chaque opération.

Nous proposons de limiter ce placement à l'intervalle de disponibilité le plus proche afin de réduire la complexité (à 3 placements par opération) tout en respectant également la minimisation de perte de la marge de gamme. Bien entendu nous pouvons construire de toutes pièces des exemples pour lesquels ce choix placement nous prive d'atteindre l'optimum. Cependant, cette limitation permet de réduire une combinatoire énorme et ne nous a pas empêchée d'atteindre l'en-cours optimal pour les nombreux exemple traités.

Les autres possibilités que nous avions de limiter la combinatoire concernent la limitation du nombre d'intervalles et de la profondeur. La première possibilité est beaucoup plus gênante car elle peut réellement être la cause d'une perte d'optimalité importante ; elle n'a donc pas été prise en compte. La limitation du nombre d'intervalles de disponibilité contraint, en effet, l'heuristique à abandonner le placement au plus tôt à partir d'un certain moment et ne contribue pas à la minimisation d'en-cours.

Concernant la limitation de la profondeur, nous avons également laissé ce paramètre à l'initiative de l'utilisateur. Il en est de même pour les paramètres de pondération des fonctions de coûts qui dépendent de l'intérêt que porte l'utilisateur à ces différentes composantes.

III.2.3 Application

Nous avons appliqué cette heuristique à l'exemple illustratif utilisé tout au long de ce mémoire. Nous disposons d'un graphe ordonnançable, cf. Figure II-12, contenant deux macro-gammes. D'après le chapitre précédent nous savons que le temps de cycle optimal est égal à 24 u.t. Les machines critiques sont au nombre de 4 : ce sont les machines $M_{1.1}$, $M_{1.2}$, M_2 et M_5. Les bornes inférieures d'en-cours sont calculées de la façon suivante :

$$\text{Borne}(\text{macro}-\text{gamme } 1) = \left\lceil \frac{29+28+31}{24} \right\rceil = \lceil 3.66 \rceil = 4 \text{ palettes}$$

$$\text{Borne}(\text{macro}-\text{gamme } 2) = \left\lceil \frac{24+24+30+26}{24} \right\rceil = \lceil 4.33 \rceil = 5 \text{ palettes}$$

$$\text{Borne inf. de l'en}-\text{cours du système} = 4+5 = 9 \text{ palettes}$$

Nous savons également que cette borne d'en-cours est atteignable puisque un ordonnancement calculé « à la main », cf. Figure III-8, a été déjà déterminé pour cet

exemple, cf. [CAM 97]. Cet ordonnancement est représenté par un diagramme de Gantt dual, cf. Figure III-8, et modélisé par un Graphe d'Evénements, cf. Figure III-9. Cette double représentation nous permet d'une part de garder la cohérence de la modélisation avec les Réseaux de Petri, adoptés depuis le début de l'étude et d'autre part d'avoir des points de vue de représentation complémentaires pour une meilleure compréhension de l'ordonnancement.

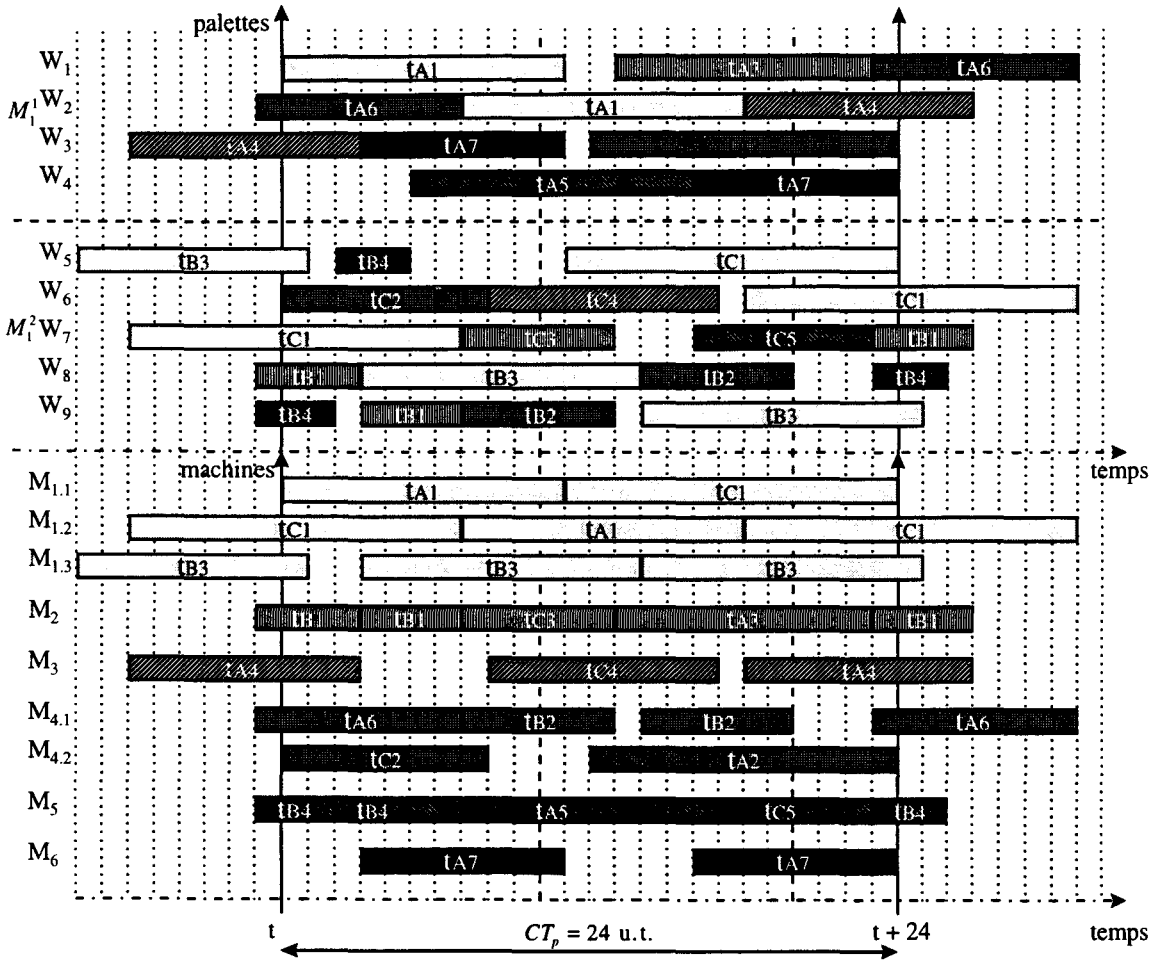


Figure III-8 : Régime permanent calculé à la main

Pour tester l'algorithme sur cet exemple, nous avons commencé par utiliser des profondeurs de recherche réduits étant donné la taille du problème. Les paramètres de pondération des fonctions de coûts ont été fixés à 100 pour la pondération du coût de l'en-cours, 1 pour la marge de gammes et 8 pour la marge de machines. Le langage de programmation choisi est le langage C. Quant à la machine, nous avons utilisé un PC doté d'un processeur Intel Pentium 150 Mhz.

C'est ainsi qu'avec un temps de calcul faible nous avons obtenu un résultat sous-optimal pour 10 en-cours, cf. Figure III-11. Ce résultat a été obtenu avec une profondeur de recherche égale à 2 et **un temps de calcul de moins d'une seconde** et ce malgré le nombre assez élevé de **séquences traitées (425)** et de **branches terminales obtenues (1917)**. Notons tout de même qu'il n'existe pas de *benchmark* pour les problèmes d'ordonnancement cyclique. Par conséquent, il n'est pas possible de comparer ces résultats avec une référence théorique. Nous nous contenterons donc de commenter nos propres résultats.

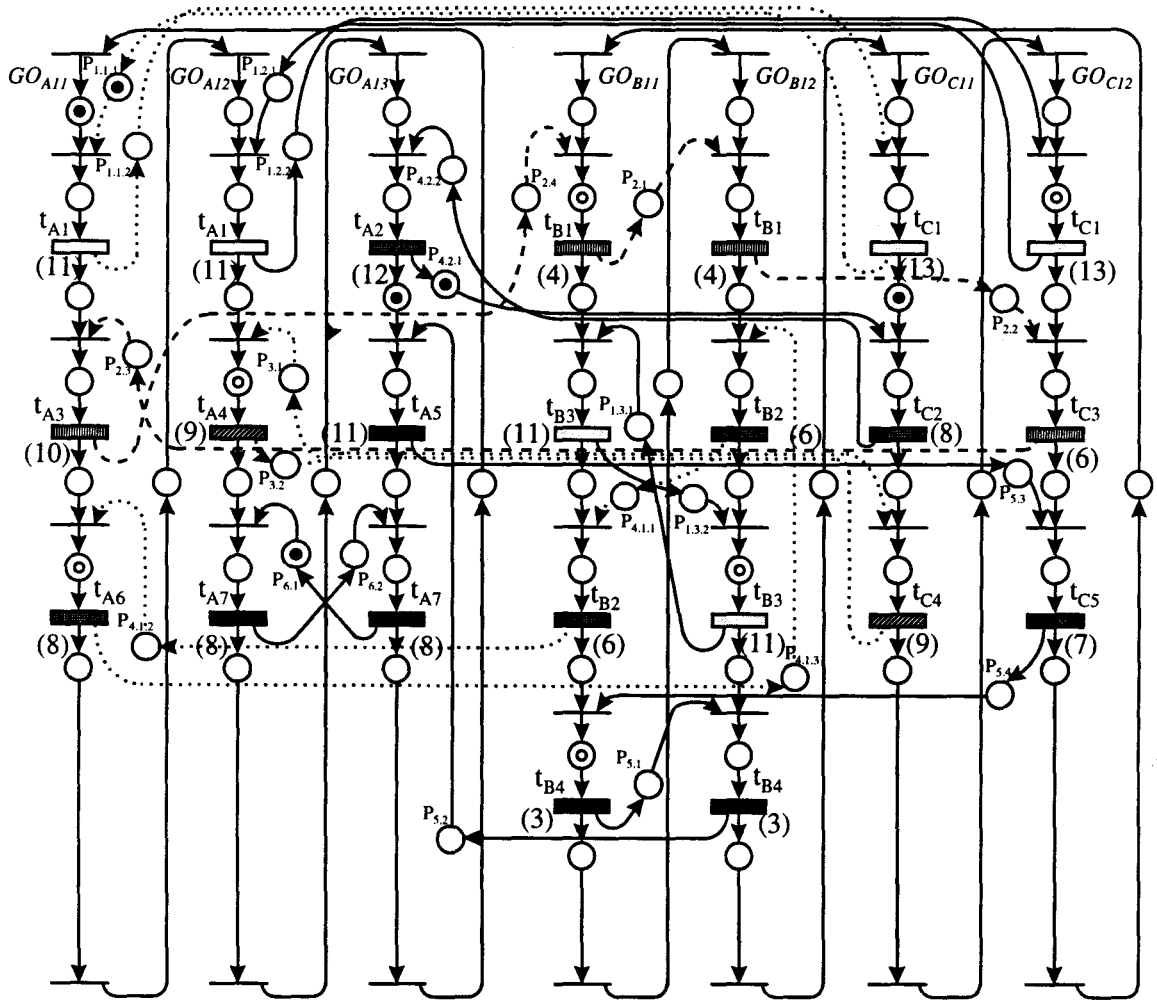


Figure III-9 : Graphe d'Evénement

En réalité, la combinatoire de cette profondeur maximale est impressionnante puisque le nombre de branches terminales obtenues est égal à **317.621.137 branches** pour **8511 séquences non équivalentes** étudiées. L'ordonnancement obtenu est très proche de celui que nous avons réalisé à la main.

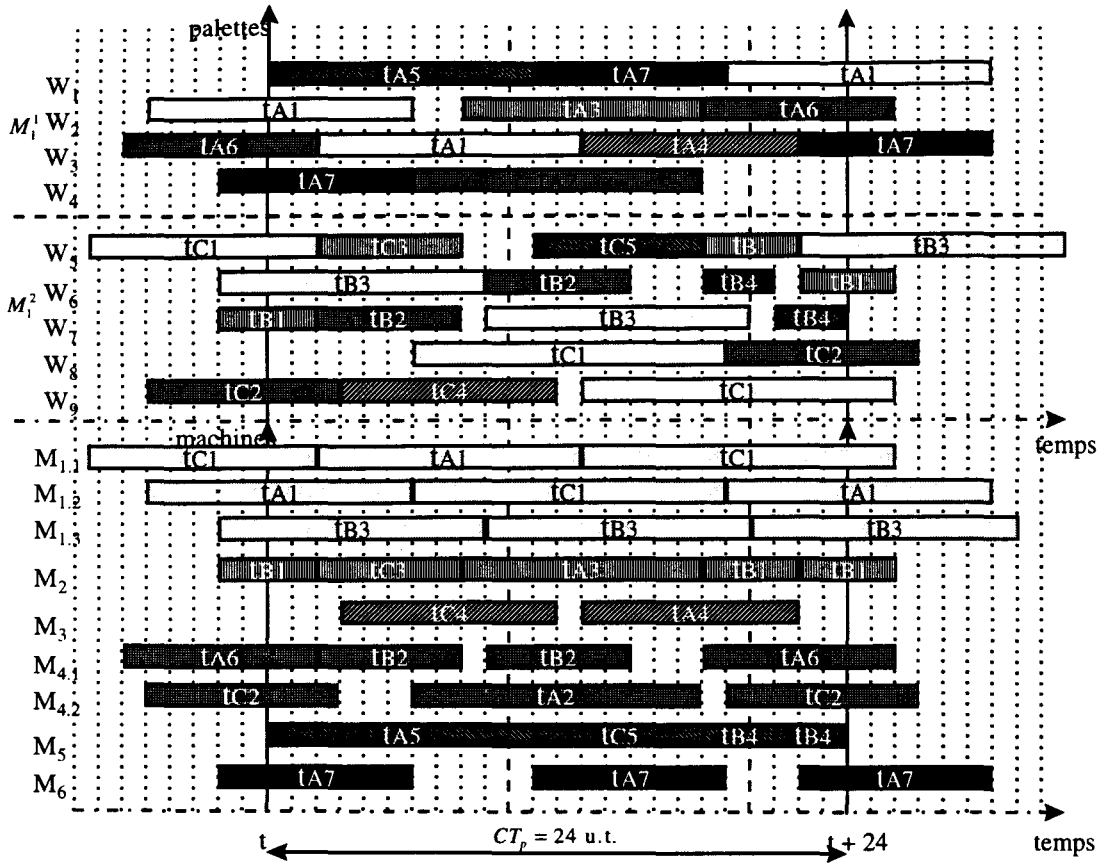


Figure III-10 : Résultat Optimal de l'heuristique

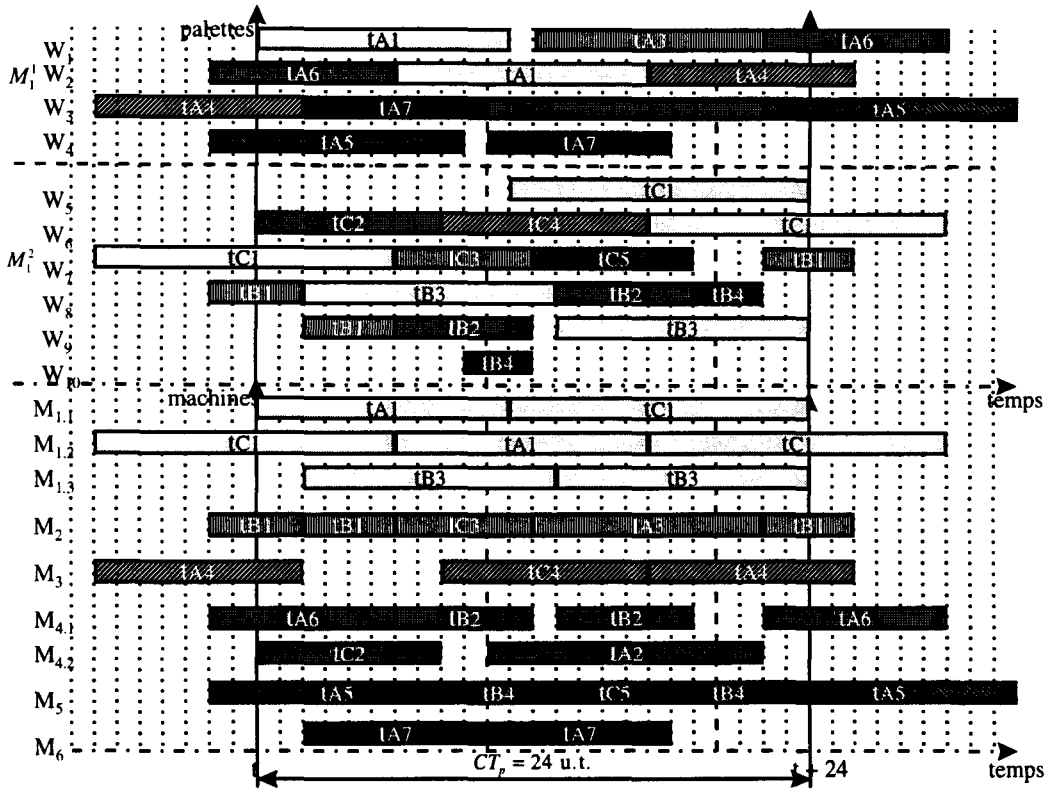


Figure III-11 : Résultat sous optimal de l'heuristique

Nous avons ensuite testé la profondeur de recherche maximale sur cet exemple. L'ordonnancement obtenu est optimal du point de vue de l'en-cours : il respecte le temps de cycle optimal avec 9 palettes (borne inférieure d'en-cours), cf. Figure III-10. Ce résultat est atteint au bout d'**une journée de calcul**.

III.2.4 Performances

Nous avons vu, au cours de la présentation des différents modules et des paramètres en jeu, d'une part que la combinatoire de l'application est très élevée (surtout pour des profondeurs de recherche importantes) et que d'autre part le nombre de branches étudiées est loin d'être calculable à l'avance étant donné l'occurrence d'ordonnancements non admissibles. Ainsi, même si nous pouvions déterminer à une itération donnée de l'application le nombre de séquences non équivalentes en fonction du nombre de macro-gammes, du nombre d'opérations les composant, du nombre de machines et bien sûr de la profondeur, cela n'est plus envisageable pour les placements admissibles des opérations formant ces séquences. Il n'est donc plus possible d'évaluer la complexité de l'heuristique même pour des productions spécifiques.

Cependant, afin de caractériser les performances de l'algorithme, nous considérons l'exemple illustratif de ce mémoire ainsi que des exemples bibliographiques de référence étudiés et publiés. Dans un premier temps, considérons l'exemple illustratif du mémoire, cf. Figure III-8. Nous étudierons dans ce cas l'influence de la profondeur de recherche[1] π sur les performances en terme de temps de calcul, d'en-cours déterminé, et de combinatoire résultante. Tout d'abord, notre observation concerne l'en-cours qui n'est pas une fonction monotone de la profondeur de recherche, cf. Figure III-12.

Ceci est principalement dû au caractère discret du problème ainsi qu'à la définition même de la profondeur de recherche. En effet, l'aspect discret de la commande fait en sorte que la moindre variation dans la résolution du problème (paramètres, variables, ...) peut générer de grands écarts entre les résultats. De plus, l'utilisation de la profondeur de recherche π rend l'algorithme, à chaque instant, totalement indifférent aux opérations hors de sa portée.

[1] Les différents tableaux de valeurs sont en Annexe III.

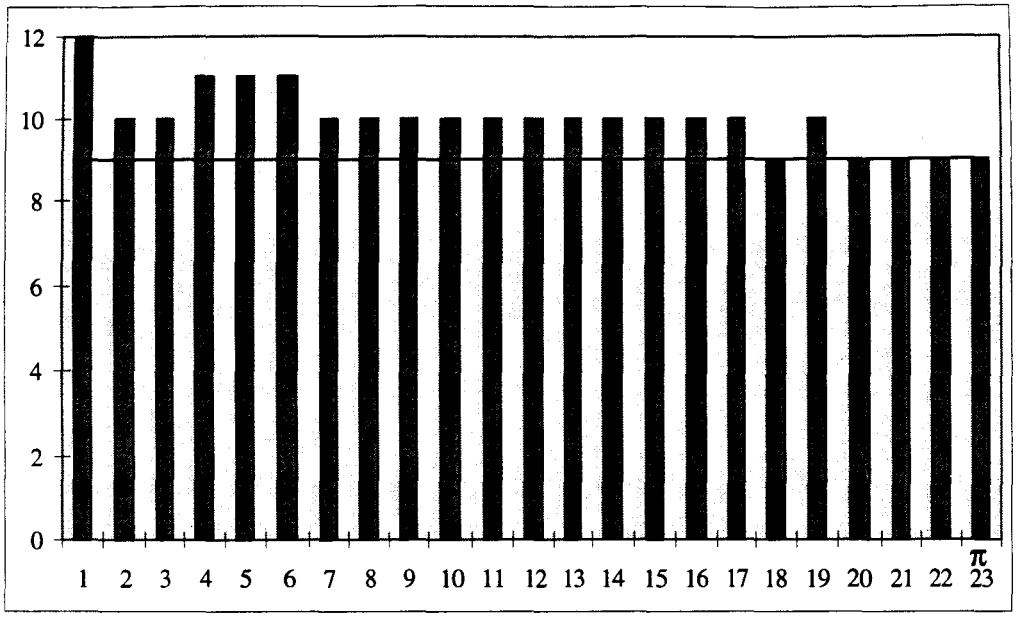


Figure III-12 : Performances de l'heuristique

Ainsi, si les variations de l'en-cours pour $\pi \in \{1, 2, 3\}$ peuvent sembler logiques (en augmentant la profondeur nous pouvons augmenter la sensibilité de la recherche), le passage à $\pi = 4$ peut paraître déroutant. En réalité, l'explication est simple, il est vrai que les séquences de longueur 4 opérations en contiennent déjà 3 et donc devraient faire au moins aussi bien.

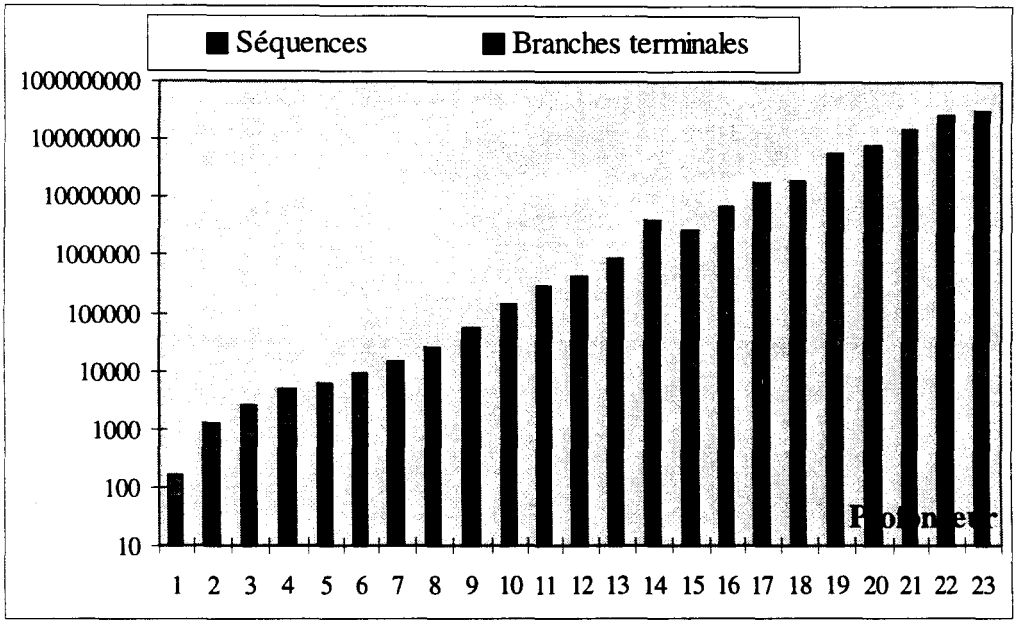


Figure III-13 : Combinatoire de la résolution

Néanmoins, l'existence d'une quatrième opération peut obliger l'algorithme à se lancer dans une branche, réalisant le meilleur coût, choix qui peut s'avérer ultérieurement fatal

pour la minimisation de l'en-cours. En somme, la limitation de la profondeur oblige l'algorithme à juger au fur et à mesure sur des séquences de longueur limitées, sans pour autant savoir si la branche choisie est la « meilleure ».

Concernant la combinatoire du problème, nous avons relevé le nombre de séquences non équivalentes traitées ainsi que le nombre de branches terminales en fonction de la profondeur de la recherche, cf. Figure III-13. Si ces deux fonctions sont, dans l'ensemble, croissantes nous avons tout de même observé un phénomène identique à celui de l'en-cours à savoir une baisse inattendue du nombre de séquences pour une augmentation de la profondeur (π = 19 et 20). cf. Annexe III pour les explications.

Pour terminer, nous évoquerons la durée de calcul nécessaire pour obtenir les différentes solutions. En effet, nous avons vu que l'application a déterminé un ordonnancement sous-optimal en moins d'une seconde et un ordonnancement optimal après une journée de calcul ! Ainsi, nous avons enregistré le logarithme du temps de calcul([1]) en fonction de la profondeur de recherche. Cette représentation révèle un comportement quasi linéaire.

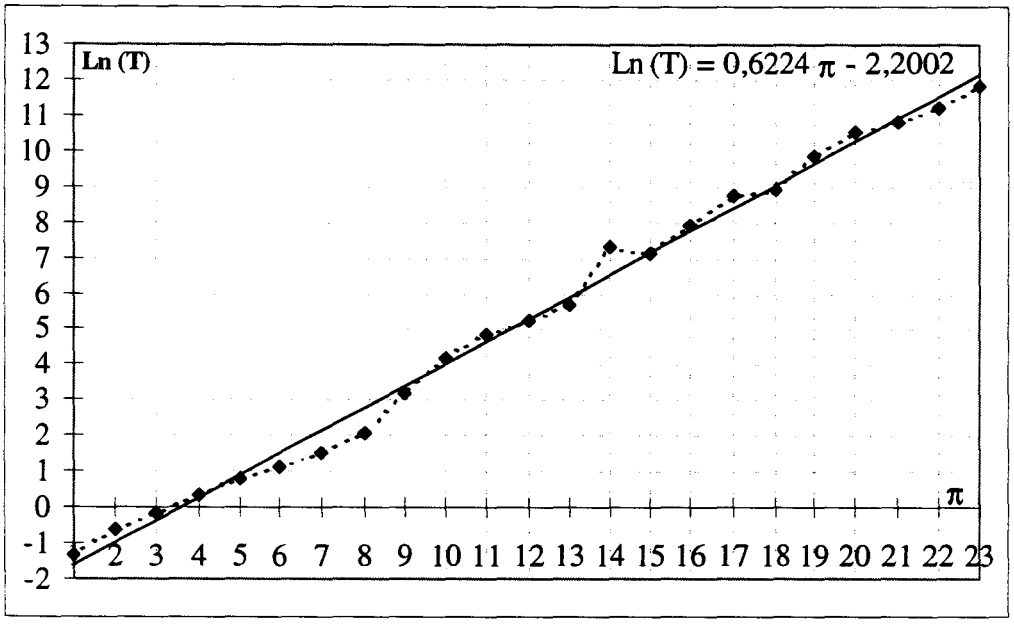


Figure III-14 : Durées du calcul

[1] Ces temps de calcul (ne sont pas des temps CPU) sont à considérer avec un minimum d'incertitude due au fait que toutes les prises de valeurs n'aient pas été effectuées dans exactement les mêmes conditions (Swap disque, écran de veille, Anti-virus automatique, autres applications tournant en parallèle ...)

C'est ainsi que nous avons pu déterminer expérimentalement, pour notre exemple, l'évolution de la complexité traitée par l'heuristique. Cette complexité est exponentielle et de la forme :

$$\text{Durée de calcul}(\pi) \approx e^{0.62*\pi-2.2} = 11.1*10^{-2}*1.86^{\pi}$$

Equation III-5 : Durée du calcul en fonction de profondeur de recherche.

Nous allons maintenant comparer les résultats de notre heuristique à ceux d'exemples tirés de la bibliographie, cf. [HIL 87], [HIL 88], [VAL 94] et [OHL 95a]. Chaque exemple de référence, cf. Annexe III, est étudié avec notre heuristique, cf. Tableau III-1. Pour chaque exemple nous donnons la borne inférieure d'en-cours, le résultat bibliographique, le résultat optimal[1] de notre heuristique, ainsi que sa durée, et éventuellement le résultat d'une profondeur limitée pour les exemples dont la durée et jugée excessive. La comparaison se fera uniquement sur l'en-cours puisque nous ne disposons pas dans les références bibliographiques des durées de calcul des exemples traités. Nous nous contenterons donc de juger la durée de calcul de notre heuristique.

La première constatation, que nous pouvons formuler, concerne l'efficacité de l'heuristique. En effet, nous constatons que nous obtenons des résultats au moins aussi bien que les résultats trouvés par les auteurs. La deuxième remarque concerne le temps de calcul qui peut aller de 1 seconde à 3000 secondes).

Exemple	[HIL 87]	[HIL 88]	[VAL 94]	[OHL 95]
Borne inf. d'en-cours	**5** palettes	**5** palettes	**5** palettes	**4** palettes*
En-cours donné par les auteurs	**5** palettes	**6** palettes	**5** palettes	**5** palettes
En-cours trouvé par l'heuristique	**5** palettes	**5** palettes	**5** palettes	**5** palettes
Durée de calcul de l'heuristique	**1h 40 mn**	**1 s**	**10 mn**	**1 s**
Solution Rapide (en-cours)	**6** palettes	**5** palettes	**6** palettes	**5** palettes
Durée de calcul associée	**1,4 s**	**1 s**	**3 s**	**1 s**

Tableau III-1 : Comparaison avec d'autres algorithmes.

[1] Nous avons tenu à comparer le comparable : pour chaque exemple nous appliquons exactement le même exemple sans chercher à trouver les meilleures macro-gammes pour minimiser l'en-cours au maximum. Ainsi nous cherchons uniquement à comparer le résultat de l'heuristique seule aux autres algorithmes.

* Borne Inf. non atteignable.

Ceci s'explique par le fait que nous considérons trois politiques de placement. En effet, les quatre exemples bibliographiques utilisent le placement au plus tôt dans l'en-cours contre trois placements possibles pour notre heuristique. Ainsi, quand le seul placement au plus tôt suffit pour trouver l'en-cours optimal, la combinaison des trois politiques de placement sont également capables de l'atteindre. Néanmoins, dans le premier cas il faudra t temps de calcul contre $t*\pi^3$ pour notre heuristique (il faudra essayer les trois possibilités à chaque fois). L'avantage de ces placements est qu'ils sont capables de trouver le meilleur en-cours là où le placement au plus tôt seul peut échouer, cf. [HIL 88].

III.2.5 Conclusion

Dans la première partie de ce chapitre, nous avons présenté, formulé et résolu le problème de l'ordonnancement cyclique avec respect du temps de cycle optimal et minimisation de l'en-cours. Cette étude s'est concrétisée par la mise en place d'une heuristique d'ordonnancement informatisée.

Malgré la combinatoire très importante, inhérente au problème, nous avons la possibilité de limiter l'arbre de recherche et obtenir très rapidement un résultat, moyennant une perte relative d'optimalité (cf. première colonne sur le Tableau III-1). Cependant, nous pouvons également rechercher le meilleur résultat atteignable par l'heuristique, moyennant cette fois-ci un temps de calcul assez long (cf. première colonne sur le Tableau III-1).

La recherche d'une solution passe donc par un compromis temps de calcul - en-cours calculé. Rappelons que cette phase d'ordonnancement est une étape qui s'inscrit dans l'approche globale de résolution. Par conséquent, nous ne devons pas oublier les phases de partition de gammes et de regroupement cyclique qui sont primordiales pour la minimisation de l'en-cours. Pour illustrer cette remarque nous allons reprendre l'exemple de [VAL 94], cf. Annexe III. Dans cet exemple, il s'agit de réaliser, par cycle, cinq pièces (dont 3 d'un même type P1 et 2 d'un même type P2) sur 3 machines (M1, M2 et U1) avec :

P1 : U1 (2 u.t.), M1 (3 u.t.) et M2 (2 u.t.).

P2 : M1 (1 u.t.) et U1 (2 u.t.).

Dans cette configuration, il est clair que le temps de cycle optimal est de 11 u.t. et que la machine menante est M1. L'application de la méthode de Hillion nous donne 6 palettes

pour respecter le temps de cycle tandis que celle de Valentin donne 5 palettes, cf. [VAL 94]. Si nous appliquons notre heuristique dans les mêmes hypothèses que les deux précédentes (c'est-à-dire en affectant chaque palette à une seule pièce) nous obtenons 5 palettes en 10 mn. de calcul, cf. Tableau III-1. Dans tous les cas, nous ne pouvons pas obtenir mieux étant donné que la borne inf. de l'en-cours est égale à $3*\left\lceil\frac{2+3+2}{11}\right\rceil+2*\left\lceil\frac{1+2}{11}\right\rceil=5$.

Cependant, en utilisant pleinement les flexibilités des ressources de transport et en appliquant les deux précédentes phases de partitions de gammes et de regroupement cyclique, nous pouvons améliorer sensiblement ce résultat. En effet, si nous utilisons une séquentialisation totale (c'est-à-dire que nous regroupons toutes les pièces d'un même type dans les mêmes partitions), nous obtenons 2 macro gammes :

R1 : U1 (2 u.t.), M1 (3 u.t.), M2 (2 u.t.), U1 (2 u.t.), M1 (3 u.t.), M2 (2 u.t.), U1 (2 u.t.), M1 (3 u.t.) et M2 (2 u.t.).

R2 : M1 (1 u.t.), U1 (2 u.t.), M1 (1 u.t.) et U1 (2 u.t.).

La borne d'en-cours associée à ce regroupement est la suivante :

$$\text{Borne associée au regroupement} = \left\lceil\frac{3*7}{11}\right\rceil+\left\lceil\frac{2*3}{11}\right\rceil = 3 \text{ palettes}$$

L'utilisation de l'heuristique d'ordonnancement sur cette nouvelle configuration, cf. [KOR 97a], nous donne, en moins d'une seconde de calcul[1], 3 palettes pour respecter le temps de cycle optimal ce qui correspond à la borne inf. de l'en-cours. En conséquence, nous optimisons d'avantage l'en-cours utilisé par le système en associant les phases de partition de gammes et de regroupement cyclique à l'étape d'ordonnancement proprement dit.

Néanmoins, l'ordonnancement du régime permanent cyclique obtenu ne prend pas en compte les opérations de transfert entre les différentes machines (ces opérations sont jusque là supposées se dérouler en temps négligeable ou nul). Pour faire correspondre notre

[1] Malgré la conservation du nombre d'opérations et celui des machines, la modification du nombre de gammes a fait passer le temps de calcul de 10 mn. (cf. Tableau III-1) à moins d'une seconde.

ordonnancement au fonctionnement réel de l'atelier, nous devons donc prendre en compte ces opérations de transfert. C'est l'objet de la deuxième partie de ce chapitre.

III.3 Introduction au système de transport

Après avoir présenté l'heuristique d'ordonnancement, nous allons maintenant nous intéresser au système de transport pour prendre en compte les opérations de transfert, jusque là supposées se dérouler en temps masqué.

Nous allons, plus particulièrement, étudier l'impact du système de transport sur l'en-cours utilisé ainsi que la meilleure manière d'introduire les opérations de transport d'une machine à une autre sans remettre en cause nos résultats ni augmenter la combinatoire de recherche. En effet, nous voulons maintenir une évolution à temps de cycle minimal. L'ajout d'opérations de durées non nulles nous conduit naturellement à augmenter l'en-cours afin de rendre ces opérations transparentes par rapport au flux de la production.

Dans tous les cas, nous éviterons de recourir à l'ordonnancement sans opérations de transfert, cf. première partie de ce chapitre, puis à l'introduction des opérations de transport car cette approche est fastidieuse : elle décompose le problème en deux phases ce qui engendre une augmentation de la combinatoire et probablement une perte des performances voire même de l'optimalité. Au contraire, nous chercherons, au mieux, à réutiliser l'heuristique déjà établie, moyennant quelques modifications bien entendu.

III.3.1 Prise en compte des opérations de transfert

L'introduction des opérations de transfert (de durées non nulles) ne remet pas en question notre hypothèse de départ sur la **non criticité des ressources de transport**. Ainsi, les ressources de transport ne seront pas prises en compte lors de l'optimisation du flux de la production. Par conséquent, du fait de cette hypothèse de non criticité du système de transport, le transport n'est pas pris en compte durant l'étape de planification fine. Il en est de même pour la phase de linéarisation du graphe et la détermination des partitions des gammes.

Cependant, la formation des macro-gammes introduit la notion de contrainte de précédence entre deux pièces et impose, de ce fait, à la palette un transfert de la machine, utilisée pour réaliser la dernière pièce, vers celle marquant le démarrage d'une nouvelle

pièce. Nous allons donc réintroduire les opérations de transfert au niveau de la création des regroupements cycliques.

En conséquence, nous obtiendrons des macro-gammes différentes de celles trouvées précédemment puisqu'elles doivent tenir compte des opérations de transfert entre deux machines successives. La combinatoire de la détermination des regroupements cycliques est donc de même ordre dans les deux cas (avec ou sans opérations de transport).

De plus, étant donné que la phase partitionnement des gammes coïncide avec le calcul de la borne minimale d'en-cours nécessaire, nous proposons maintenant d'introduire une nouvelle borne min. d'en-cours, à la fin de la phase de regroupement des gammes, qui tient compte des opérations de transfert. Il est cependant possible de disposer d'un système de transport « flexible » où plusieurs chemins sont possibles entre deux machines. Dans ce cas nous utiliserons la durée minimale de transfert pour le calcul de la borne minimale. Pour illustrer nos propos nous représentons Figure III-15 un atelier de production formé des éléments suivants :

- un convoyeur (anneau de transport principal) de capacité égale à 34 cellules unitaires([1]).
- 6 machines M1, ..., M6 placées, dans cet ordre autour de l'anneau principal :
  - ✓ la machine M5 est placée directement sur l'anneau
  - ✓ le robot de chargement/déchargement M1 ainsi que les autres machines sont placées en dérivation avec des buffers d'entrée de capacité fixée à 3 unités et des buffers sortie unitaires (tout deux gérés en FIFO).
- un raccourci AB, parallèle à M2 et M3, de capacité égale à 5 unités
- un raccourci BC, parallèle à M5 et M6, de capacité égale à 5 unités

La capacité globale du système est égale à 72 palettes (34 dans l'anneau principal, 10 dans les branches en pointillés AB et BC, (3+1)*5=20 dans les buffers des machines en dérivation, 2 dans l'entrée et sortie de M5 et 6 dans les machines). Le système de transport

[1] Une cellule unitaire peut contenir une et une seule palette.

est supposé unidirectionnel (tournant dans le sens des aiguilles d'une montre) à vitesse constante indépendante de son encombrement. Nous fixons à titre d'exemple le temps nécessaire à la traversée d'une cellule à une unité de temps.

Ainsi pour rallier M5 à partir de M2 en utilisant le raccourci AB il faut 11 unités de temps : (buffer de sortie de M2 : 1 u.t.) + (M2→A : 2 u.t.) + (A→B : 5 u.t.) + (B→M5 : 2 u.t.) + (buffer entrée M5 : 1 u.t.). C'est cette durée que nous utiliserons pour le calcul d'une nouvelle borne d'en-cours nécessaire.

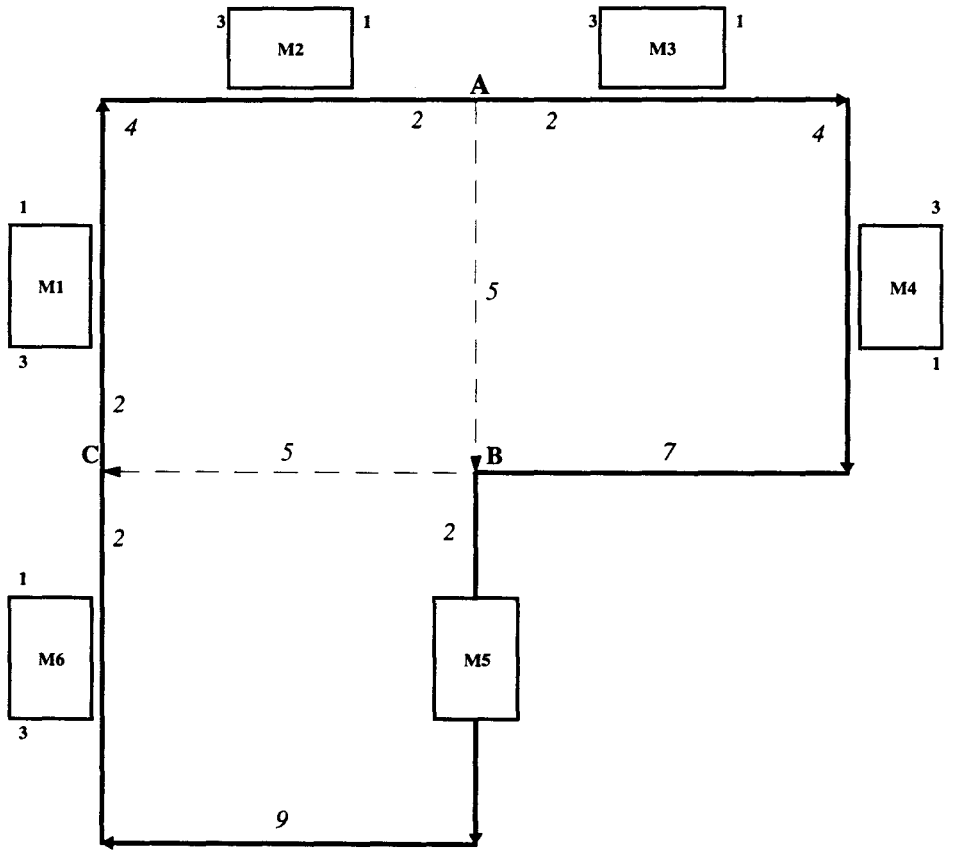


Figure III-15 : Atelier de production, cf. [MAN 96]

III.3.1.1 Exemple illustratif

Pour la suite de l'étude du système de transport, cf. Figure III-15, nous considérerons la production suivante formée de trois types de produits G1, G2 et G3, cf. Figure III-16, où :

- G1 : M1 (10 u.t.), M2 (50 u.t.), M3 (40 u.t.), M4 (30 u.t.), M5 (20 u.t.) et M1 (10 u.t.) ;
- G2 : M1 (10 u.t.), M4 (30 u.t.), M2 (50 u.t.), M5 (20 u.t.) et M1 (10 u.t.) ;

- G3 : M1 (10 u.t.), M5 (20 u.t.), M4 (30 u.t.) et M1 (10 u.t.) ;
- Horizon de production E = { G1, G2, G3 }, Temps de cycle optimal associé CT = 100 u.t. et Machine critique M2 ;
- Borne inf. d'en-cours $B_0 = \left\lceil \frac{160+120+70}{100} \right\rceil = \lceil 3{,}4 \rceil = 4$ palettes (sans opérations de transfert).

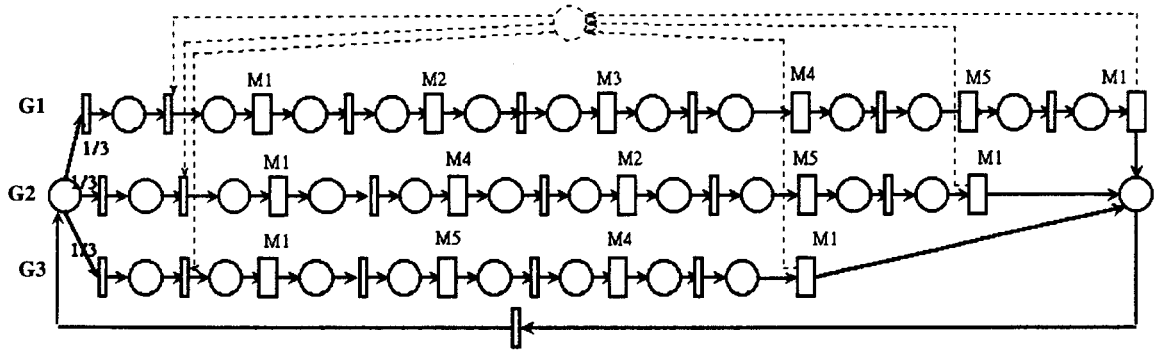


Figure III-16 : Graphe linéarisé

L'utilisation de l'heuristique d'ordonnancement conduit à l'ordonnancement cyclique de la Figure III-17, où le minimum théorique d'en-cours est atteint avec l'utilisation de deux macro-gammes {G1} et {G2, G3}. Ce résultat a été obtenu **en moins d'une seconde de calcul**. Nous allons maintenant étudier l'impact du système de transport, cf. Figure III-15, sur ce résultat.

III.3.1.2 Influence du système de transport sur les performances de la production

L'introduction du système de transport pose une nouvelle question, ou plutôt une contrainte, à savoir la capacité du système en terme de nombre limite d'en-cours à transporter. Si, pour une production donnée, nous calculons analytiquement le débit (Φ) de d'une cellule de production en fonction de l'en-cours (N) utilisé, cf. Figure III-18. Nous distinguons trois régimes de fonctionnement :

- Régime linéaire : caractérisé par un en-cours trop faible pour pouvoir saturer l'une des ressources de transformation et le débit est de la forme $\Phi = k * N$
- Régime saturé : caractérisé par un débit constant ($\Phi = \Phi_{max}$) où les machines menantes sont saturées. L'ajout d'un en-cours sert alors à augmenter la marge de gamme.

- Régime sursaturé : caractérisé par une chute brusque du débit de production de sa valeur maximale à la valeur zéro ($\Phi = 0$). Le système est alors bloqué. Ce régime a été étudié par [AUS 94].

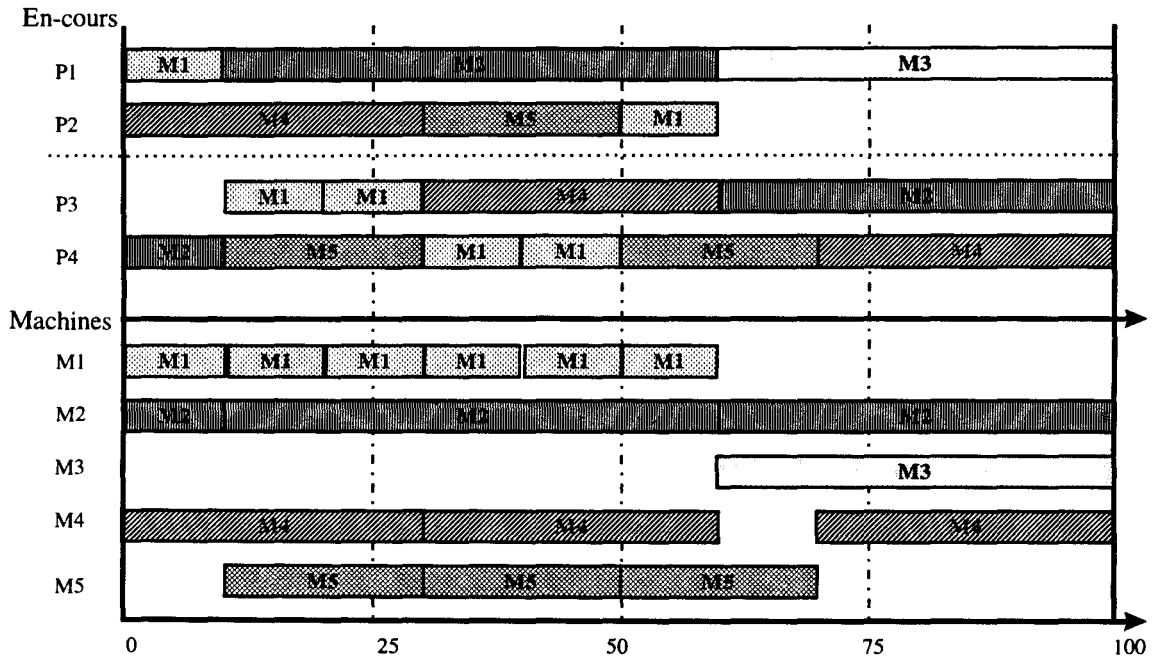


Figure III-17 : Ordonnancement sans opérations de transfert

Etant donné que nous cherchons à évoluer à débit maximal et à en-cours minimal nous sommes ici quasiment certain de ne pas rencontrer ce problème de saturation du système de transport.

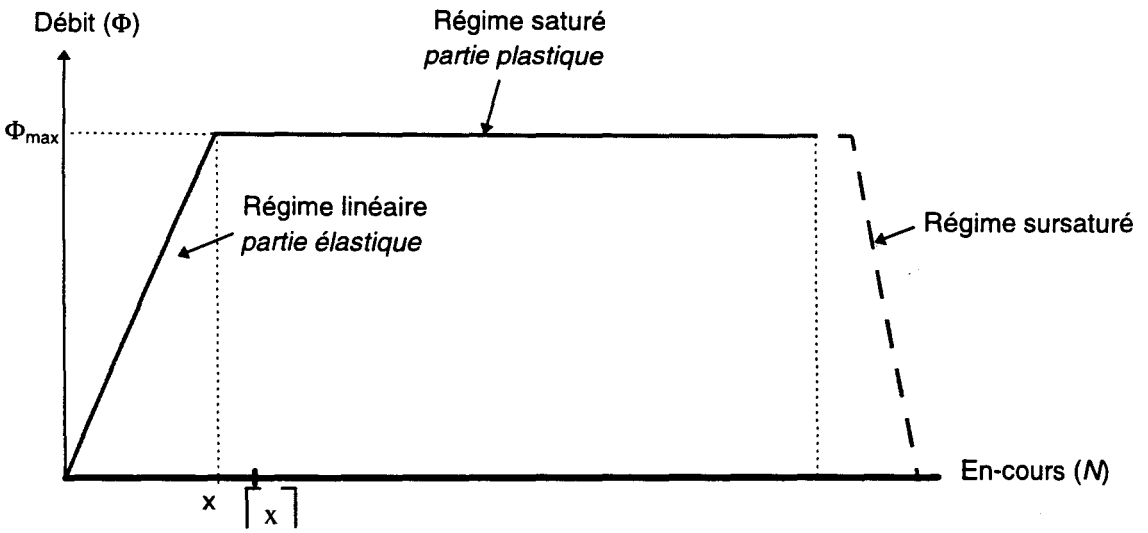


Figure III-18 : Débit du système en fonction de l'en-cours

Nous nous situons donc, pour notre démarche globale, au niveau du régime saturé (débit maximal). Nous cherchons alors à atteindre le point d'intersection x avec le régime linéaire (ou plutôt $\lceil x \rceil$) par valeurs positives, cf. Figure III-18.

III.3.1.3 Modélisation et simulation

Après avoir évoqué le problème d'encombrement et blocage du système nous allons nous intéresser à sa modélisation. En effet, il est impératif d'avoir un modèle, à la fois, homogène, avec le modèle de la production, et représentatif du système réel c'est-à-dire sans ajout de contraintes artificielles. Nous allons donc utiliser les Réseaux de Petri pour la modélisation des opérations de transfert.

Cependant, et dans un souci de clarté et de simplicité du modèle, nous considérerons le système de transport par tronçon. Un tronçon est une composante du convoyeur, de taille maximale, ayant une seule entrée et une seule sortie. Ainsi un buffer d'entrée ou de sortie est un tronçon tout comme une partie du convoyeur comprise entre deux noeuds successifs. En conséquence, les gammes linéaires G1, G2 et G3 sont modifiés par l'introduction des opérations de transfert selon les ajouts suivant ci-dessous en gras :

- G1 : M1 (10 u.t.), **M1→M2** (1+4+3=8 u.t.)[1], M2 (50 u.t.), **M2→M3** (8 u.t.), M3 (40 u.t.), **M3→M4** (8 u.t.), M4 (30 u.t.), **M4→M5** (11 u.t.), M5 (20 u.t.), **M5→M1** (17 u.t.) et M1 (10 u.t.) ;
- G2 : M1 (10 u.t.), **M1→M4** (1+4+2+2+4+3=16 u.t.), M4 (30 u.t.), **M4→M2** (22 u.t.), M2 (50 u.t.), **M2→M5** (11 u.t.), M5 (20 u.t.), **M5→M1** (17 u.t.) et M1 (10 u.t.) ;
- G3 : M1 (10 u.t.), **M1→M5** (15 u.t.), M5 (20 u.t.), **M5→M4** (29 u.t.), M4 (30 u.t.), **M4→M1** (18 u.t.) et M1 (10 u.t.) ;

Bien entendu les machines critiques ainsi que le temps de cycle optimal sont les mêmes et le seul changement concerne la borne inférieure de l'encours :

$$B'_0 = \left\lceil \frac{212+186+132}{100} \right\rceil = \lceil 5{,}3 \rceil = 6 \text{ palettes.}$$

[1] Pour aller de M2 à M1, il faut parcourir la sortie de M1 (1 u.t.) puis le M1→M2 (4 u.t.) et l'entrée de M2 (3 u.t.)

Afin de connaître la capacité réelle du système, en terme d'en-cours supporté, nous avons procédé à une série de simulations. Ces simulations ont consisté à introduire des pièces de types G1, G2 et G3 avec des ratios 1/3, 1/3 et 1/3 (pour respecter l'horizon de production) et à étudier le flux de la production en fonction de l'en-cours utilisé et du nombre de pièces produites, cf. Figure III-19.

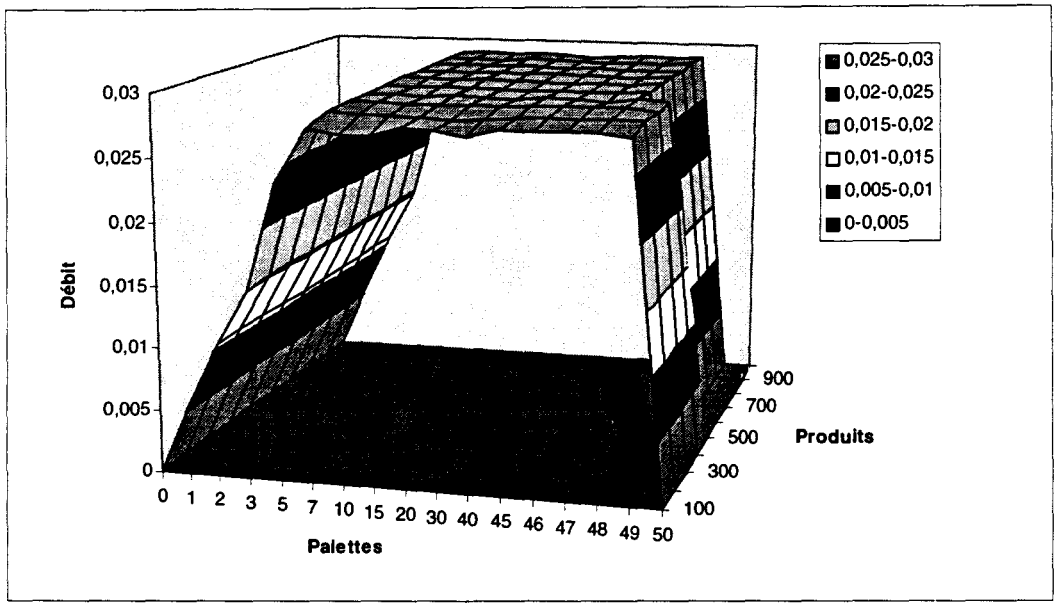


Figure III-19 : Simulation de la capacité du système de transport

Les simulations ont été réalisées à l'aide d'un logiciel de simulation de Réseaux de Petri : *VISUAL SIMNET*, cf. [GEN 97] et Annexe III. On observe que le modèle obtenu est sursaturé à partir de 49 ou 50 palettes (en fonction du contexte et du nombre de produits demandés). Ceci veut dire que l'atelier peut se bloquer avant la borne théorique de saturation. Notons tout de même que ces valeurs sont celles d'une simulation sur un modèle qui n'est pas totalement fidèle au système physique. Nous pouvons également remarquer que le débit est presque linéaire pour un nombre de palettes compris entre 1 et 7 et qu'à partir de ce nombre on atteint le régime sursaturé. En conséquence, en minimisant l'en-cours du système (autours de 6 à 7 palettes) nous ne devrions pas avoir de blocage dû à une saturation du système de transport.

III.3.2 Heuristique d'ordonnancement avec transport

Nous allons maintenant nous intéresser à l'ordonnancement en présence d'opérations de transfert. Cette heuristique sera basée sur l'algorithme d'ordonnancement développé dans

la première partie de ce chapitre. En effet, nous allons chercher à réutiliser les mêmes modules et le même fonctionnement global.

III.3.2.1 Principe

Pour retrouver un fonctionnement analogue à celui de l'algorithme d'ordonnancement sans ressources de transfert nous devons considérer ces dernières comme des ressources à part entière c'est-à-dire les traiter au même niveau que les machines de transformation. Cependant la difficulté d'un tel fonctionnement réside dans le fait que si nous considérons des ressources simples (cellules) nous risquons de voir exploser la combinatoire de la résolution ainsi que celle de la représentation par Réseaux de Petri. Il en est de même si nous considérons des ressources multiples (tronçons) car elles sont cumulatives ce qui nous oblige à mettre en place un nouveau module de résolution des conflits d'accès à ce type particulier de ressources.

Néanmoins, en observant de près les caractéristiques du système de transport nous pouvons facilement déduire que les ressources le composant sont loin d'être aussi sollicitées et contraintes que les ressources de transformation. Ceci est principalement dû au grand nombre de ressources unitaires de transfert (cellules du système de transport) par rapport au nombre de machines et à notre souci de minimiser le nombre d'en-cours dans le système. Ainsi, il n'est pas nécessaire de disposer d'une modélisation trop fine l'échelle des cellules de transfert.

Afin de ne pas modifier considérablement le principe de l'algorithme, nous ne considérerons pas ces tronçons comme des ressources à part entière car elles ont des particularités difficiles à prendre en compte. En réalité, ces tronçons peuvent être classés en deux catégories différentes selon qu'ils servent à transférer uniquement ou qu'ils servent à transférer et éventuellement à stocker les pièces.

Prenons par exemple le système de transport, cf. Figure III-15, nous pouvons distinguer un trajet principal (en trait épais). Les palettes qui circulent sur cette partie ne doivent en aucun cas être bloquées car ceci pourrait impliquer le blocage de plusieurs autres pièces et perturber le fonctionnement du système entier. Le reste du système de transport (AB, BC et les buffers d'entrées et sorties) est considéré en plus comme système de stockage c'est-à-dire qu'il est possible d'y bloquer un ou plusieurs en-cours en attendant la libération de la

sortie. Ces deux catégories devraient néanmoins être considérées différemment pour prendre en compte leurs particularités ce qui complique encore la tache de l'algorithme.

III.3.2.2 Résolution

Comme nous venons de voir, les ressources de transport sont très peu sollicitées et ne risquent pas de poser des problèmes d'accès ou d'encombrement. C'est pour cela que nous avons décidé de prendre en compte uniquement les durées de transfert. Ceci signifie que nous ne traiterons pas les ressources de transfert comme des ressources à part entière mais plutôt comme durée de temps à insérer entre deux opérations de transformation. Cette méthode de résolution nous permet de reprendre exactement le même algorithme avec un ajout systématique des durées de transfert après le placement de chaque opération. La date de disponibilité de l'en-cours devient alors :

date de disponibilité = date de fin de la dernière opération placée + durée de transfert vers la prochaine machine.

Ainsi, le calcul des marges de gammes est strictement le même :

$$\text{marge de gamme initiale} = \begin{array}{r} \text{Borne inf. d'en}-\text{cours (avec transport)} \\ -\sum \text{durées opératoires} \\ -\sum \text{temps de transfert} \end{array}$$

Durant le calcul, si la machine n'est pas disponible à la date de disponibilité de l'en-cours alors nous perdons de la marge de gammes exactement comme dans le cas d'ordonnancement sans les opérations de transfert, cf. Figure III-20.

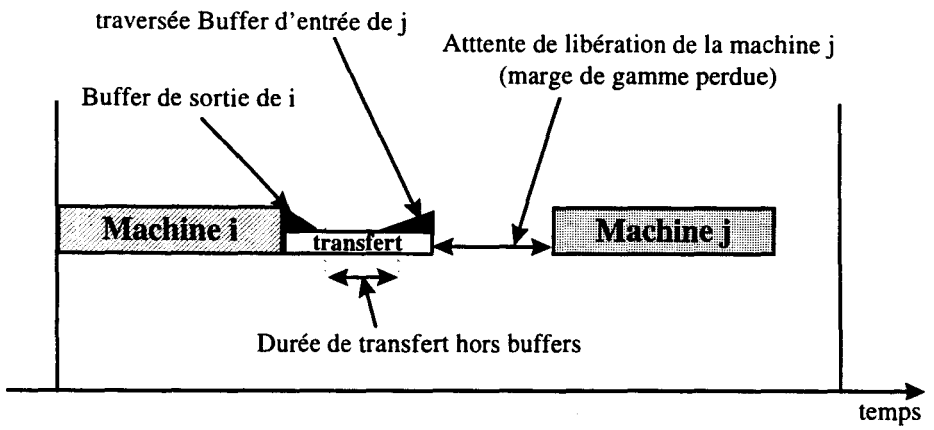


Figure III-20 : Première représentation d'une opération de transfert

Une fois l'ordonnancement terminé, il reste deux problèmes à résoudre. Les conflits d'accès à ces ressources de transport et l'affectation des marges de gammes à des lieux de stockage. En effet, bien que les conflits d'accès soient peu nombreux, il n'en reste pas moins qu'il peut y en avoir quelques uns et qu'il faudra les résoudre.

De plus, la prise en compte du système de transport oblige à affecter chaque en-cours dans le système à une ressource pendant son séjour dans le système. En réalité, même si l'ordonnancement donne des « blancs » sous forme de marge de gamme perdue, durant ces intervalles l'en-cours se trouve quelque part dans le système et il occupe une ressource (de transport). Il faut donc affecter les palettes à des ressources de transport (ou plutôt de stockage) pendant ces marges perdues de gamme.

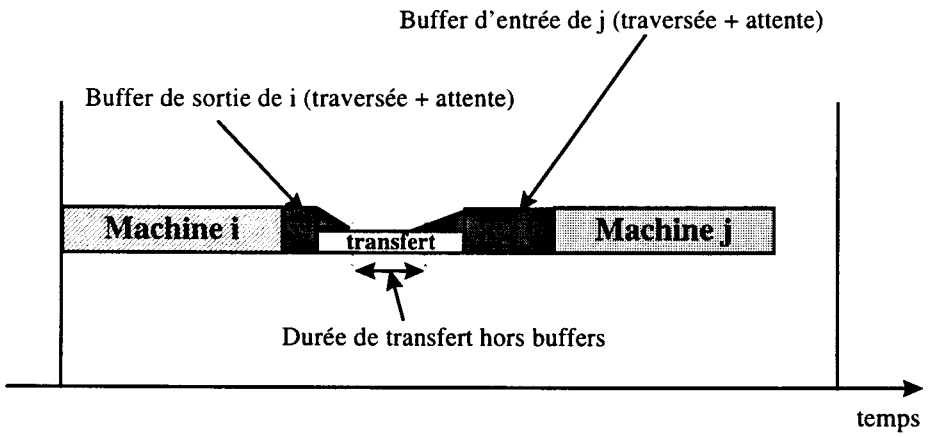


Figure III-21 : Représentation réelle de la position de l'en-cours dans le système

En pratique ces lieux sont les buffers d'entrées ou de sorties des machines ou les tronçons dits secondaires de transport (AB et AC pour notre cas). Dans ce genre de cas nous préférons occuper le buffer de sortie de la machine pour deux raisons essentielles : la première étant que la durée de transformation d'une pièce est bien plus importante que les durées de traversée d'un tronçon. Quant à la seconde elle consiste à éviter d'occuper l'entrée d'une machine trop tôt car nous risquerions de bloquer l'entrée d'autres pièces qui doivent chronologiquement être produites avant.

Nous donnons à titre d'exemple une solution possible, cf. Figure III-21, à l'exemple Figure III-20. Nous pouvons constater que la durée de transfert hors buffers est toujours la même (il ne faut pas bloquer le système de transport principal) et que le séjour dans les buffers a été allongé pour résoudre le problème de stockage.

Nous allons maintenant appliquer ce principe d'heuristique d'ordonnancement avec les opérations de transfert à l'exemple Figure III-16. Nous avons à ordonnancer les macro-gammes avec les opérations de transport, cf. III.3.1.3. nous rappelons également que la borne inférieure d'en-cours est égale à 6 palettes.

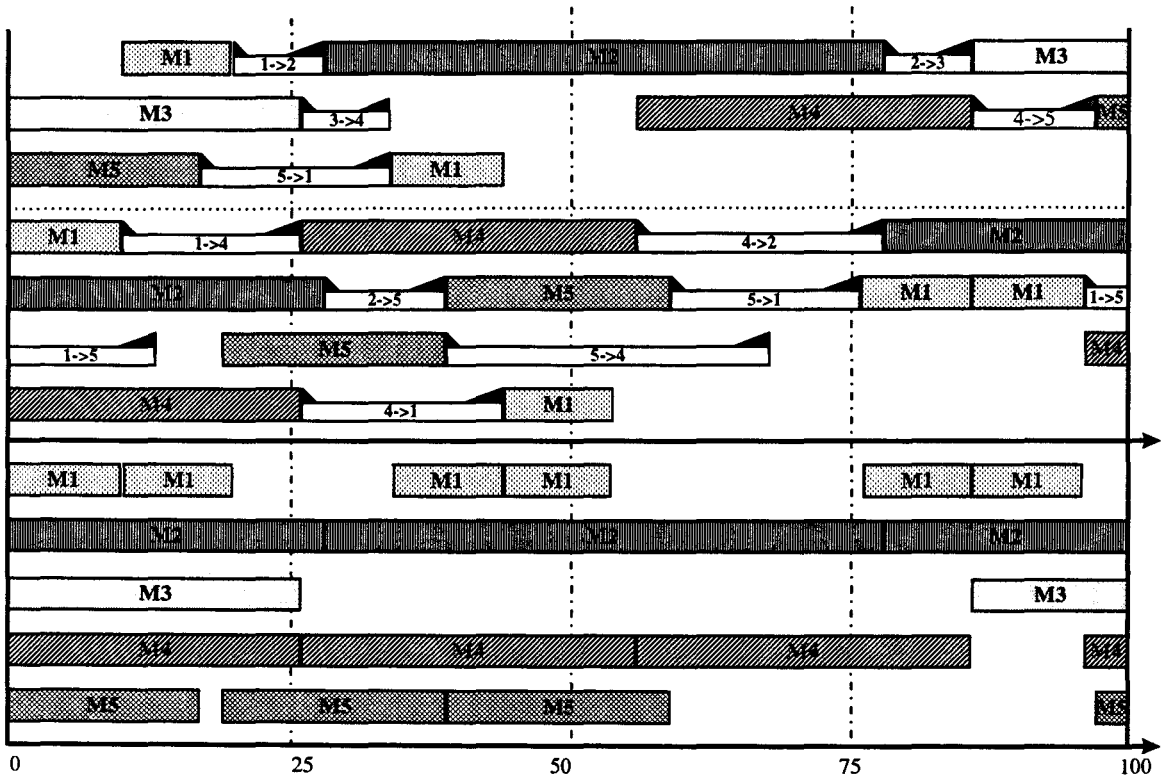


Figure III-22 : Application de l'ordonnancement avec transport

L'ordonnancement de cette nouvelle configuration se fait donc selon les mêmes principes que précédemment avec comme seul changement le placement des opérations de transfert (à durées fixées d'avance) sans vérification ni résolution des conflits d'accès aux ressources de transfert. Le résultat, cf. Figure III-22, est sensiblement identique à celui sans transfert. Le nombre d'en-cours déterminé est de 7 palettes (un en-cours de plus que la borne inférieure).

Il ne reste donc qu'à résoudre les éventuels conflits et à « remplir » les intervalles vides entre les opérations en affectant les palettes à des ressources de stockage en attendant leur transformation. Du point de vue conflit, nous n'en avons décelé aucun. Quant au stockage, il nous a suffi d'affecter les palettes aux buffers d'entrées des machines pour qu'elles puissent attendre la libération de la machine, cf. Figure III-23.

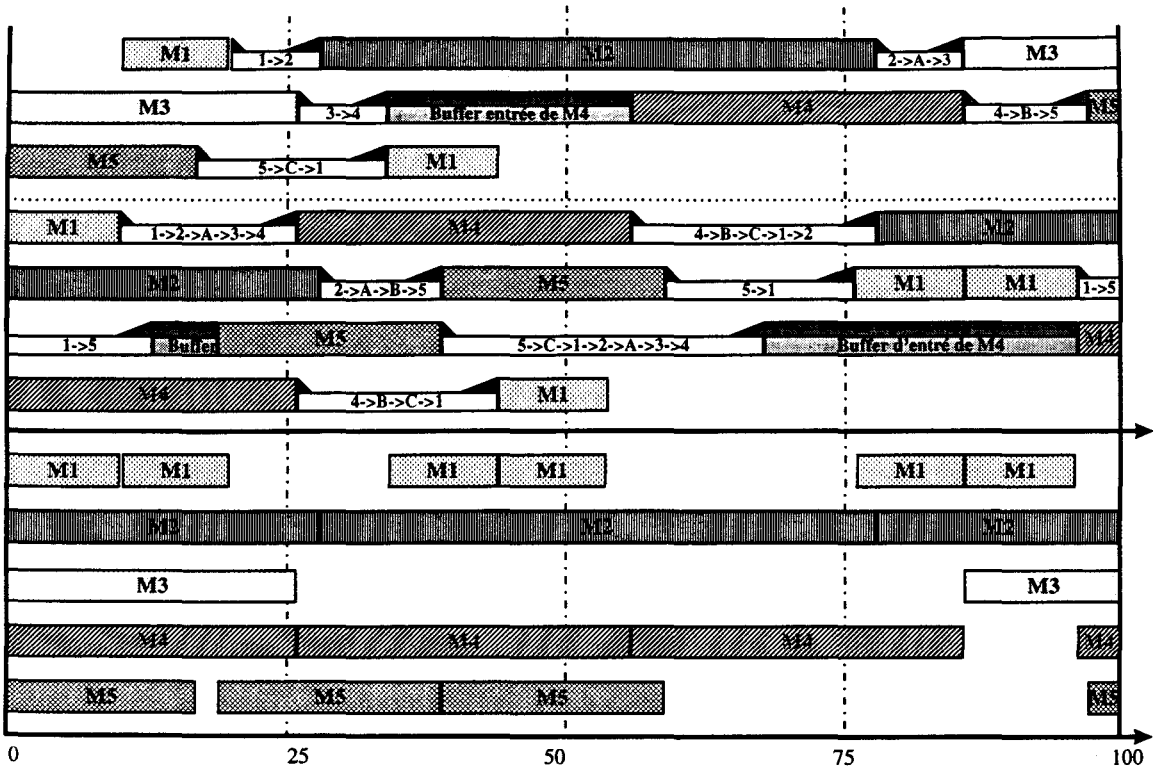


Figure III-23 : Résolution des conflits : Ordonnancement cyclique final

Le résultat obtenu est donc un ordonnancement cyclique déterministe du régime permanent. Ce cycle est à répéter *théoriquement* X fois (X est le nombre de répétitions du régime permanent donné par la phase de planification fine).

III.4 Conclusion

Dans ce chapitre, nous avons, dans un premier temps, présenté et résolu le problème de l'ordonnancement cyclique respectant le temps de cycle optimal tout en minimisant l'en-cours du système. La résolution a été effectuée par une heuristique étant donné la complexité, au moins NP-difficile, du problème. Dans un second temps, nous avons introduit les opérations de transfert entre les différentes ressources du système. Nous avons alors étudié l'influence de ces opérations sur les performances de l'atelier aussi bien du point de vue de l'en-cours que du point de vue blocage par saturation du système. Ce travail a abouti à la mise en place d'une heuristique d'ordonnancement avec opérations de transfert. Nous rappelons, une fois de plus, que cette phase d'ordonnancement est à considérer dans le contexte de la démarche globale d'évaluation de performances et ordonnancement.

Le régime permanent optimisé ne peut être atteint en temps nul puisqu'il faut, en général([1]), une période de transition pour que le flux atteigne sa valeur maximale et une autre période de transition pour faire passer l'atelier de sa vitesse maximale à l'arrêt ou à une nouvelle production : le régime transitoire sera l'objet du chapitre suivant.

Cependant, pour ne pas alourdir les notations et les calculs du chapitre suivant nous allons considérer des régimes permanents sans opérations de transfert en précisant que les résultats obtenus sont tout à fait applicables aux ordonnancements cycliques avec opérations de transfert.

[1] sauf dans le cas où l'ordonnancement obtenu utilise une palette par pièce et sans chevauchement de cycles.

Chapitre IV

IV. Etude et Optimisation des Régimes Transitoires

IV.1 Position du Problème

Ce chapitre concerne l'ordonnancement des régimes transitoires. Dans ce but, nous supposons disposer de régimes permanents optimisés, et nous cherchons l'optimisation des transitoires de lancement, d'arrêt et d'enchaînement d'un régime permanent à la suite d'un autre afin de minimiser la durée totale de production. Les régimes permanents initialement déterminés ne sont pas remis en question. En effet, nous avons analysé, durant les chapitres précédents, la combinatoire importante et donc la complexité du problème d'optimisation des performances. Il ne serait donc pas raisonnable d'augmenter la complexité de l'étude en renégociant le séquencement des opérations sur les machines durant le régime global (transitoire + permanent).

En fait, le choix de la conduite prévisionnelle déterministe doit conduire à la maîtrise de l'état du système. L'étude et l'optimisation des régimes transitoires nous permettront de réagir rapidement aux aléas de la production moyennant, bien entendu, une détection rapide des anomalies. En effet, le module de surveillance pourra comparer, au moins à chaque cycle, l'état du système par rapport à la commande prévisionnelle (dates de sorties des pièces).

Il est donc possible de réagir rapidement à ces aléas, en fonction du temps de calcul (de l'ordre de quelques minutes) des régimes transitoires et de leur implantation dans le système de contrôle de l'atelier flexible. Il faut toutefois rappeler, la nécessité de considérer un arrêt total de la production (en cas de panne) pour justifier la nécessité de conserver une connaissance parfaite de l'état du système et des conditions de redémarrage de l'atelier à partir desquels on va pouvoir lancer la nouvelle commande calculée. Durant cet arrêt, nous pouvons à la fois effectuer les opérations de maintenance nécessaires et relancer le calcul de la commande.

IV.1.1 Intérêt des régimes transitoires

Nous supposons donc, à ce niveau de notre étude, que seuls les régimes permanents ont été ordonnancés et optimisés. Il subsiste donc le problème d'**implantation de la commande** notamment la mise en régime de la production (lancement de la production) et de fin de production.

Dans la littérature, peu de travaux ont été réalisés sur ce sujet. A. Munier a étudié le problème de mise en régime d'ordonnancement cyclique k-périodique, appliqué aux systèmes d'information, cf. [MUN 91] et [HAN 94]. Dans ce contexte, la phase transitoire n'a pas posé beaucoup de problèmes puisqu'elle a consisté à conserver la même résolution temporelle que celle du régime permanent : duplication des opérations relevant du régime permanent durant la phase transitoire puis prise en compte des contraintes de précédence dans les gammes opératoires en éliminant les opérations indésirables. Si, dans la gamme opératoire une opération OP1 précède une autre OP2, alors la première occurrence de OP2 est effectuée après la première occurrence d'OP1. Ceci doit être également vérifié pour les autres occurrences des différentes opérations. Ce transitoire est donc assez simplement déterminé sans recherche d'optimalité. De même, J. Erschler a étudié le régime permanent pour le cas du job-shop ainsi que le régime transitoire associé à une commande cyclique d'un flow-shop, cf. [ERS 82].

Le transitoire permettant d'enchaîner deux régimes permanents, dans las cas de *mono-production* (un seul produit à la fois), a été étudié par C. Varnier, cf. [VAR 96], pour le *Hoist Scheduling Problem*. La résolution a été effectuée par la PLC (programmation logique sous contraintes).

La deuxième raison nous incitant à considérer les régimes transitoires est due à la nature même des problème que nous étudions c'est-à-dire des **régimes permanents finis** pour lesquels la durée du transitoire est loin d'être négligeable. En effet, les ordonnancements cycliques que nous pouvons trouver dans la littérature concernent exclusivement les régimes permanents infinis, cf. [HIL 87], [VAL 94], ... Il est donc compréhensible que le transitoire soit négligé par rapport au permanent. Nous allons donc chercher à optimiser les régimes transitoires en tenant compte du fait qu'ayant déjà optimisé le contenu d'un cycle du régime permanent (en-cours et durée) nous pourrions alors globalement **optimiser le makespan** d'une production effective. En effet, pour le makespan effectif de la production,

on ne possède, pour l'instant, qu'une borne inférieure résultant de la somme totale théorique des durées des différents régimes permanents, cf. § II.2 :

$$\text{Borne inf. du makespan total} = \sum_{p=1}^{RP_{\max}} X(p).CT\left(RP_p\right)$$

Cette borne est obtenue à la fin de l'étape de planification fine, cf. Figure IV-21. Quant au transitoire, nous ne pouvons pas déterminer la moindre borne significative avant d'avoir déterminé les régimes permanents. Concernant le makespan, nous n'avons pas pour l'instant de borne supérieure. Nous ne maîtrisons donc pas sa durée. Rappelons que l'une des principales contraintes concerne le **respect des délais** ; **Nous ne pouvons, pour l'instant, le garantir**.

On sait qu'en raison des régimes transitoires, il sera impossible de répéter effectivement le nombre de cycles requis pour réaliser le nombre exact de pièces pendant le régime permanent. Les régimes transitoires ne pourront être optimisés de la même façon que les régimes permanents car ils ne sont pas soumis aux mêmes contraintes de production.

Le régime transitoire servira également comme tampon pour terminer certaines productions n'appartenant à aucun régime permanent (**transitoires isolés**). Ces derniers sont introduits pour régler les problèmes d'arrondi résultant d'une décomposition de la production en plusieurs régimes permanents cycliques de petites tailles.

De plus, la détermination des régimes transitoires permettra, si cela s'avère nécessaire, d'**introduire de nouvelles commandes « prioritaires »** avec un minimum de perte de performances. Il en sera de même pour les **opérations urgentes de maintenance**. La maîtrise des transitoires s'avère donc fondamentale dans la mise en oeuvre et la justification d'une bonne utilisation de la flexibilité d'un système de production.

Pour toutes ces raisons, il est nécessaire d'étudier ces régimes transitoires et de réduire leurs durées au bénéfice des régimes permanents qui ont été optimisés pour la production recherchée. Il est donc nécessaire de limiter l'importance et la durée des régimes transitoires nécessaires à l'établissement rapide des régimes permanents optimisés.

En effet, si on suppose que le nombre minimal de répétitions d'une commande se situe aux alentours d'une dizaine de fois et que l'on a réussi à gagner 10% sur le temps de cycle, on aura finalement réussi à réduire le makespan d'une période de fonctionnement.

L'optimisation du transitoire n'est donc pas à négliger dans ce cas puisqu'à une durée d'un cycle près nous pourrions perdre ce qui a été gagné durant l'optimisation du régime permanent.

IV.1.2 Nouvelle borne inférieure pour le Makespan

Nous disposons a priori d'une borne inférieure grossière du makespan (compte tenu du grand nombre de flexibilités). La levée des indéterminismes permet, dans un premier temps, d'affiner cette borne.

Soit une demande initiale : $\{T(I),\ I \in \{1, ..., I_{max}\}\}$. Après la phase de planification fine, cette demande est décomposée en un ensemble de régimes permanents : $\{RP_p,\ p \in \{1, ..., RP_{max}\}\}$.

Considérons RP_q un régime permanent fixé caractérisé par son horizon de production cyclique $E(RP_q) = \{I(q).I,\ \forall i \in \{1, ..., I_{max}\}\}$, le temps de cycle associé $CT(RP_q)$ et le nombre de répétitions de cet horizon $X(q)$.

En fin de la phase 2 (linéarisation des gammes), de l'approche de détermination de la commande d'un régime permanent, nous avons déterminé les machines critiques, la répartition discrète des charges sur les machines et associé une gamme linéaire opératoire à chaque pièce qui va être produite durant un cycle : $\{GL_q(k),\ k \in \{1, ..., \text{Card}(E(RP_q))\}\}$.

Soit $GL_q(l)$ une gamme linéaire de cet ensemble et m une machine critique. Nous définissons :

- $NO(GL_q(l))$: le nombre d'opérations dans cette gamme,
- $prem(GL_q(l),m)$: le rang de la première opération de la gamme appartenant à la machine critique m,
- $dern(GL_q(l),m)$: le rang de la dernière opération de la gamme appartenant à la machine critique m

(si aucune opération de la gamme $GL_q(l)$ n'appartient à la machine critique m alors nous prenons comme notation $prem(GL_q(l),m) = 1$ et $dern(GL_q(l),m) = NO(GL_q(l))$).

Il vient alors :

$$\forall l \in \left\{1, ..., \text{Card}\left(E\left(RP_q\right)\right)\right\} \qquad 1 \le prem\left(GL_q(l),m\right) \le dern\left(GL_q(l),m\right) \le NO\left(GL_q(l)\right)$$

Equation IV-1

Pour produire toutes les pièces, il est clair que toutes les machines doivent réaliser la totalité de leur charge. Donc chaque machine critique effectuera sa charge cyclique ($X(q)$ cycles de durées $CT(RP_q)$), soit un temps d'occupation total de $X(q)*CT\left(RP_q\right)$ unités de temps.

Pour atteindre une opération appartenant à une machine critique m, nous devons tout d'abord lancer les $prem(GL_q(l),m)$-1 opérations non critiques de la gamme associée. De même que pour arrêter la production, il faudra achever les $NO(GL_q(l))$-$dern(GL_q(l),m)$ opérations suivantes.

En notant $Z(l,n)$ le temps opératoire de la $n^{ème}$ opération de la gamme linéaire l, on obtient une borne inférieure du makespan pour un régime permanent et son transitoire associé :

$$\text{Borne inf. makespan optimal} = \underbrace{X(q)*CT\left(RP_q\right)}_{\substack{\text{charge d'une} \\ \text{machine critique}}} + \max_{\substack{\forall \text{ machine} \\ \text{critique m}}} \left(f(m)+g(m)\right)$$

$$f(m) = \min_{l \in \left\{1,...,\text{Card}\left(E\left(RP_q\right)\right)\right\}} \left(\sum_{n=1}^{prem\left(GL_q(l),m\right)-1} Z(l,n) \right) : \text{min. des durées opératoires nécessaires pour atteindre la machine critique m}$$

$$g(m) = \min_{l \in \left\{1,...,\text{Card}\left(E\left(RP_q\right)\right)\right\}} \left(\sum_{n=dern\left(GL_q(l),m\right)+1}^{NO\left(GL_q(l)\right)} Z(l,n) \right) : \text{min. des durées opératoires nécessaires pour terminer la pièce après m}$$

Equation IV-2 : Borne inférieure du makespan pour une production cyclique à la fin de l'étape de linéarisation des gammes

Cette borne inférieure est également **valable pour le problème général d'ordonnancement** puisqu'elle est basée uniquement sur l'optimisation du flux de production. En ce qui concerne l'ensemble de la production nous ne pouvons pas additionner toutes les bornes inférieures associées aux différentes productions cycliques car

nous prévoyons d'étudier les transitoires inter-productions, cf. § IV.6, pendant lesquels un régime cyclique se termine alors qu'un autre est lancé : les transitoires de début de l'un et de fin de l'autre s'entrelacent.

IV.1.3 Notations et Hypothèses

IV.1.3.1 Fonctions des régimes transitoires

Nous distinguons trois types différents de régimes transitoires :

1. le régime transitoire de lancement de la production à partir d'un système initialement à l'arrêt et vide appelé **pré-production**,

2. le régime transitoire de fin de production qui se charge de vider le système et d'arrêter la production appelé **post-production**.

3. le régime transitoire se situant entre deux régimes permanents : il gère la fin de la première production et, en même temps, le début de suivante. Il est appelé **inter-productions** (entrelacement de deux régimes transitoires de fin et lancement).

Le premier transitoire aura pour mission de lancer la production et d'atteindre, le plus vite possible (cf. critères de performance que nous définirons au § IV.1.3.2) un point donné à partir duquel le fonctionnement cyclique optimisé démarre. On aura donc un ensemble d'opérations à ordonnancer au plus tard (par rapport à la date de démarrage du régime permanent). Cependant, ce problème diffère du problème d'ordonnancement au plus tard habituel puisque la date due est différente selon les opérations. En effet, comme nous l'avons vu au chapitre précédent, il n'existe pas forcément de date fixe de synchronisation à laquelle toutes les machines ont terminé d'opérer (ordonnancement à chevauchement de cycles). Ainsi, certaines opérations démarreront pendant le régime transitoire et se termineront durant le régime permanent.

Ces opérations ne doivent en aucun cas être déplacées car elles conduiraient à une modification de la date du début du régime permanent et entraîneraient une modification de l'ensemble des opérations à considérer pour le transitoire. De même, pour le régime transitoire de post-production, il sera chargé de vider le système, le plus vite possible, en terminant la dernière production. Il consistera donc à ordonnancer un certain ensemble

d'opérations au plus tôt. Comme dans le cas précédent, certaines opérations chevaucheront le dernier cycle du régime permanent et le régime de post-production. Le cas du régime transitoire inter-productions est à considérer spécifiquement car il diffère des deux précédents. Ce régime introduit de nouveaux problèmes et de nouvelles variables inexistants dans le pré et post-production.

Les deux premiers régimes transitoires seront étudiés et résolus séparément dans la suite de ce chapitre. Quant au régime transitoire inter-productions, nous proposons une introduction aux nouveaux paramètres et hypothèses, qu'il nécessite par rapport aux deux premiers, ainsi qu'une formulation de problème. La résolution effective de ce problème demeure une perspective de nos travaux.

IV.1.3.2 Critères de performance

Comme nous l'avons vu depuis le début de l'étude, la minimisation du temps total de production est le critère pertinent que nous prendrons en compte. Dans un premier temps, nous avons cherché à minimiser la durée des régimes permanents([1]). Nous considérons donc par hypothèse, avoir déjà déterminé et optimisé le régime permanent, nous allons donc reconsidérer le critère originel de minimisation du temps total de production (makespan).

Pour cette raison, le critère de **minimisation du makespan** constitue le **critère de performance principal** pour l'étude des régimes transitoires. En effet, pour le niveau « gestion de production » seul le temps total de production est considéré pour juger de la qualité de la solution proposée (sous réserve que le temps d'évaluation d'une solution ne soit pas prohibitif : problème de complexité).

Dans l'éventualité de plusieurs solutions admissibles (admettant le même makespan minimal), nous proposons de choisir **la solution maximisant la durée du régime permanent**. En effet, pour deux solutions ayant le même makespan nous chercherons à produire le plus longtemps possible selon une commande périodique. Cette solution revient donc à **minimiser le régime transitoire** puisque :

$$\text{durée transitoires} + \text{durée permanents} = \text{Makespan}.$$

[1] Ce critère s'est traduit par la détermination du temps de cycle minimal et la considération de cette valeur comme contrainte forte à respecter.

Si, malgré tout, nous obtenions plusieurs solutions admissibles nous choisirions alors les solutions qui minimisent la durée du régime de pré-production ou celle du régime de post-production et ceci sans incidence sur les performances globales du système([1]).

IV.1.3.3 Paramètres restants

Nous supposons donc disposer d'un ordonnancement 1-cyclique, par rapport aux machines, représenté par un graphe d'événements et/ou un diagramme de Gantt sous forme d'une fenêtre temporelle de largeur égale au temps de cycle $CT(RP_q)$. Il est évident que cette fenêtre peut être représentée différemment([2]) puisqu'elle peut être déplacée arbitrairement sur l'axe temporel. Notons t_0 l'origine de cette fenêtre. *Nous avons donc le choix dans cette date* et nous pouvons disposer d'autant de représentations du régime permanent que nous voulons moyennant une translation temporelle de paramètre ε : $t'_0 = t_0+\varepsilon$ ($\varepsilon \in [0, CT(RP_q)[$).

Afin de ne pas alourdir les notations et les calculs, nous supposons que $t_0=0$. Nous disposons donc d'une fenêtre temporelle $[0, CT(RP_q)]$ de largeur $CT(q)$ et ayant pour origine la date zéro. De plus, nous allons supposer, sans perte de généralité, que toutes les dates caractérisant la fenêtre temporelle (durées et dates de début des gammes) sont entières.

Ainsi, le régime permanent est représenté par une fenêtre temporelle de date d'origine zéro. Ce régime ne possède donc pas a priori de date de début imposée. Cette **Date de Début du Régime Permanent** (notée **DDRP** et appartenant à l'intervalle $[0, CT(RP_q)[$([3])) est donc un paramètre libre dans l'optimisation du makespan. Nous verrons ultérieurement que nous pourrons également profiter des flexibilités initiales des gammes au niveau des transitoires pour optimiser le temps total de production, cf. IV.6.3.

IV.2 Etude d'un cas de production isolée

Nous nous intéressons, dans un premier temps, au cas d'une production isolée, caractérisée par un temps de cycle CT et un nombre X de répétitions de l'horizon de

[1] Toutes les solutions restantes ont la même durée du makespan, du régime permanent et du transitoire.

[2] modulo $CT(q)$

[3] Etant donné le caractère cyclique de la commande **DDRP +n*CT ≡ DDRP**.

production *E*. Nous étudierons donc uniquement les régimes transitoires de pré et post-production. A titre d'illustration et également pour simplifier les expressions et les notations que nous allons introduire, nous considérerons un exemple simple, cf. Figure IV-3 et Figure IV-4. Nous appliquerons ultérieurement les résultats trouvés à l'exemple d'illustration présenté dans le chapitre précédent, cf. Figure III-8.

IV.2.1 Hypothèse

L'hypothèse principale que nous allons utiliser consiste à supposer que, **durant le transitoire de pré-production, aucune pièce n'est entièrement terminée**. Le régime transitoire de pré-production ne contient donc aucune réalisation d'aucune gamme.

Cette hypothèse permet tout d'abord de considérer que la Date de Début du Régime Permanent est effectivement comprise entre zéro et *CT* pour éviter de transformer l'étude du transitoire en un problème général d'ordonnancement. En effet, si nous tolérons la possibilité de produire, une ou plusieurs pièces, pendant le régime de pré-production nous pourrions alors à la limite envisager de réaliser toute la production durant le régime de pré-production et par conséquent transformer le problème de minimisation du makespan en un problème de réordonnancement de toutes les gammes linéaires.

En conclusion, si, en changeant le date de début du régime permanent nous observons qu'une pièce de l'une des gammes se trouve entièrement réalisée pendant le régime de pré-production, cette gamme est alors différée en fin de production.

IV.2.2 Etude préliminaire

Nous allons maintenant étudier les deux types de régime transitoire que sont les régimes pré-production et post-production.

IV.2.2.1 Définitions

Après avoir précédemment calculé une borne inf. du makespan durant la détermination de la commande cyclique du régime permanent, cf. Equation IV-2, nous allons maintenant en déterminer une borne supérieure. Considérons un régime permanent fixé représenté par un diagramme de Gantt (une fenêtre temporelle de largeur CT), cf. Figure IV-1.

L'exemple contient deux gammes respectivement notées 1 et 2, chaque gamme étant caractérisée par sa date de début et de fin. Nous ne nous intéressons pas ici aux opérations appartenant à chaque gamme ni aux marges de gammes. Cette fenêtre dépend de la date t_0 considérée comme origine du diagramme donc toutes les valeurs, que nous pouvons déterminer, sont fonction de cette date.

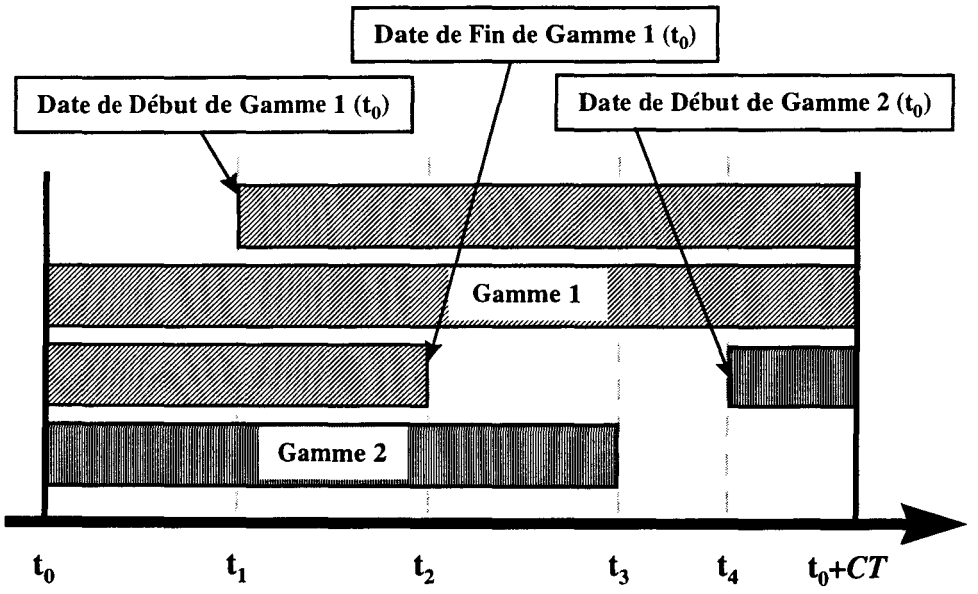


Figure IV-1 : Définition du nombre de palettes utilisées par une gamme

Notons $n_p(i,t)$ le nombre de palettes utilisées par la gamme i à la date t([1]). Pour l'exemple Figure IV-1 :

$$n_p(1,t)=\begin{cases}3 & \text{pour } t\in]t_1,t_2[\\ 2 & \text{pour } t\in [0,CT]\setminus]t_1,t_2[=[0,t_1]\cup[t_2,CT]\\ & \qquad\qquad =[t_2,t_1+CT]\end{cases}$$

$$n_p(2,t)=\begin{cases}0 & \text{pour } t\in [t_3,t_4]\\ 1 & \text{pour } t\in [0,CT]\setminus[t_3,t_4]=[0,t_3[\cup]t_4,CT]\\ & \qquad\qquad =]t_4,t_3+CT[\end{cases}$$

Nous remarquons que la fonction $n_p(i,t)$ est constante par morceaux, et qu'elle possède deux points de discontinuité que sont ses dates de début et de fin et qu'enfin elle ne peut prendre que deux valeurs dont la différence est égale à 1. Notons également :

[1] Par hypothèse nous considérons qu'à la date de début de la gamme la pièce n'a pas encore pris la palette et qu'à la date de fin de la gamme la palette est rendue. D'où les points de discontinuité.

$n_{t\max}(i) = \max_t\left(n_p(i,t)\right)$ le nombre maximal de palettes utilisées par la gamme i

$n_{p\max}(t) = \max_i\left(n_p(i,t)\right)$ le nombre max. de palettes utilisées par toutes les gammes à la date t.

En différenciant les gammes linéaires les unes des autres, nous savons que sur chaque période nous terminons une pièce d'une gamme linéaire. En conséquence, pour produire *X* pièces, il faut nécessairement *X* cycles auxquels il convient d'ajouter le temps nécessaire au lancement de la première pièce pour chacune des gammes.

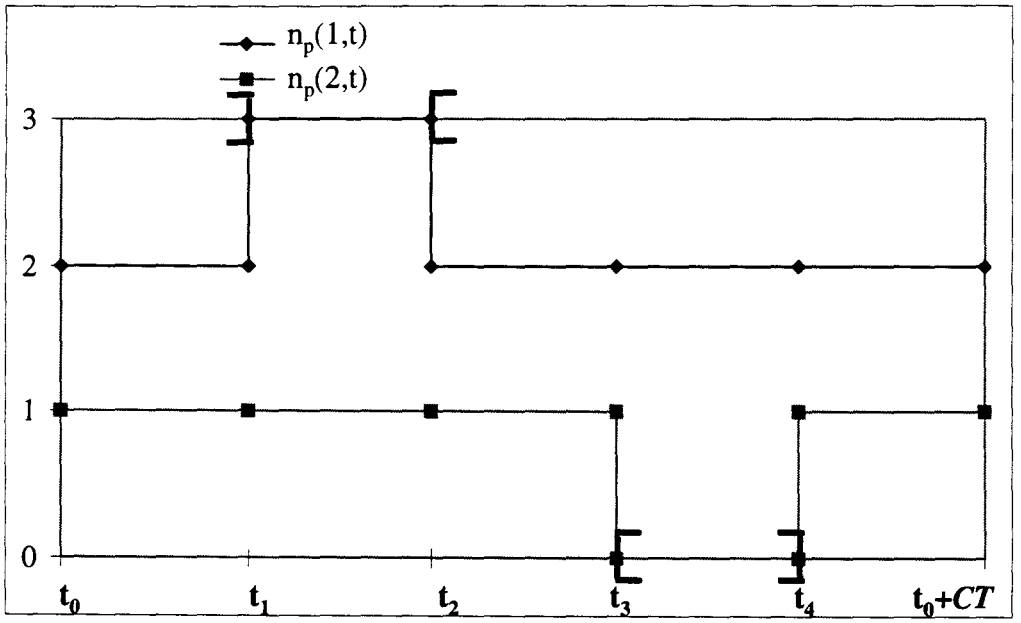


Figure IV-2 : Nombre de palettes utilisées par les deux gammes

Nous en déduisons une première borne maximale du temps total de production :

$$\text{Borne max. du makespan} = \left(X + \max_i\left\{n_{t\max}(i)\right\}\right) * CT$$

Equation IV-3 : Borne maximale du makespan

De même si l'on considère à une date t donnée les palettes utilisées par toutes les gammes nous pouvons en déduire une borne supérieure non plus du makespan mais du **makespan optimal** :

$$\text{Borne sup. du makespan Optimal} = \left(X + \min_{t} \left\{ n_{p\max}(t) \right\} \right) * CT$$

Equation IV-4 : Borne sup. du makespan optimal

En effet nous savons qu'à une date t, une gamme utilise au plus $n_{p\max}(t)$ palettes. Fixons la date t_0, correspondant à $n_{p\max}(t_0) = \min_{t} \left\{ n_{p\max}(t) \right\}$, comme Date de Début du Régime Permanent. Pour cette date, la borne sup. du makespan est égale à X*CT (pour voir sortir les X pièces finies de chaque type) + $n_{p\max}(t_0) * CT$ pour lancer les $n_{p\max}(t_0)$ palettes initialement utilisées en supposant que l'on reprend le même canevas au niveau des transitoires que celui du régime permanent. Ces deux bornes sont calculées à partir de solutions réelles du problème d'ordonnancement du transitoire.

IV.2.2.2 Présentation de l'exemple

Nous proposons d'introduire un exemple illustratif simple pour une meilleure compréhension de l'étude. Soit une production formée d'un seul régime permanent durant lequel nous avons à produire 12 pièces de 3 types différents G_1, G_2 et G_3 selon les mêmes ratios. A chaque cycle un exemplaire de chaque type est produit. Pour cela, nous disposons de trois machines M_1, M_2 et M_3. Cette production est caractérisée par les paramètres suivants :

- horizon de production E={G_1, G_2, G_3},
- nombre de répétitions de E : X=4,
- gammes opératoires :

  G_1 : Opération 1.1 : M_1 (3 u.t.) ; Opération 1.2 : M_2 (4 u.t.)

  G_2 : Opération 2.1 : M_3 (3 u.t.) ; Opération 2.2 : M_1 (3 u.t.) ; Opération 2.3 : M_2 (2 u.t.)

  G_3 : Opération 3.1 : M_1 (4 u.t.) ; Opération 3.2 : M_3 (4 u.t.)

- Temps de cycle optimal : CT=10 u.t.,
- Machine critique : M_1,

- En-cours minimal : 3 palettes.

La phase d'ordonnancement nous montre qu'il est possible d'atteindre les performances optimales avec le minimum d'en-cours en regroupant les gammes G_2 et G_3 en une macro-gamme G, cf. Figure IV-3. Cette macro-gamme utilise deux palettes de même type P_1 et P_2 alors que G_1 utilise une palette P_3. Bien que disposant d'un Graphe d'Evénements déterministe, nous pouvons remarquer qu'il existe encore une **flexibilité concernant le tir de la transition** 3.2. En effet, cette opération admet une marge de 3 unités de temps ce qui la rend dépendante de la **stratégie de tir que nous fixons** (au plus tôt, au plus tard, stratégie contrôlée par un ordonnanceur). Pour la suite, nous choisissons la stratégie de tir au plus tard de l'opération 3.2 dans son intervalle de disponibilité, cf. Figure IV-4. Nous montrerons qu'un tel choix peut finalement affecter le makespan final, cf. IV.4.

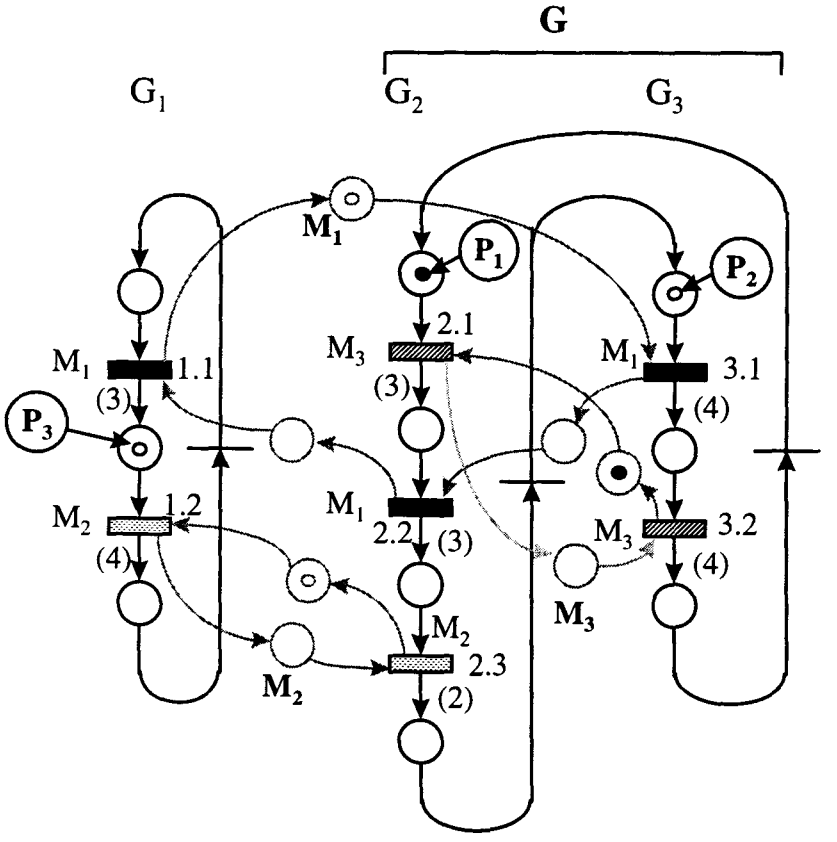


Figure IV-3 : Graphe d'Evénements

Considérons la fenêtre temporelle [0,CT]. Nous l'utiliserons comme référentiel pour la suite de l'étude (toutes les dates seront déterminées par rapport à l'origine t_0=0). Comme nous l'avons indiqué au début de ce chapitre, les bornes du makespan, avant le démarrage

de l'étude du régime transitoire, sont les suivantes, cf. Equation IV-2, Equation IV-3 et Equation IV-4 :

$$\text{Borne inf.} = X * CT + f(M_1) + g(M_1) = 4*10+2 = 42 \ \ u.t.$$

$$f(M_1) = \min_{l\in\{1,2,3\}} \left(\sum_{n=1}^{prem(G_l,M_1)-1} Z(l,n) \right) = 0 \ \ u.t.$$

$$g(M_1) = \min_{l\in\{1,2,3\}} \left(\sum_{n=dern(G_l,M_1)+1}^{NO(G_l)} Z(l,n) \right) = 2 \ \ u.t.$$

$$\text{Borne max. du makespan} = \left(X + \max\{1,1,2\}\right) * CT = 6*10 = 60 \ \ u.t.^*$$

$$\text{Borne sup. du makespan optimal} = \left(X + \min\{1,1,1\}\right) * CT = 5*10 = 50 \ \ u.t.^*$$

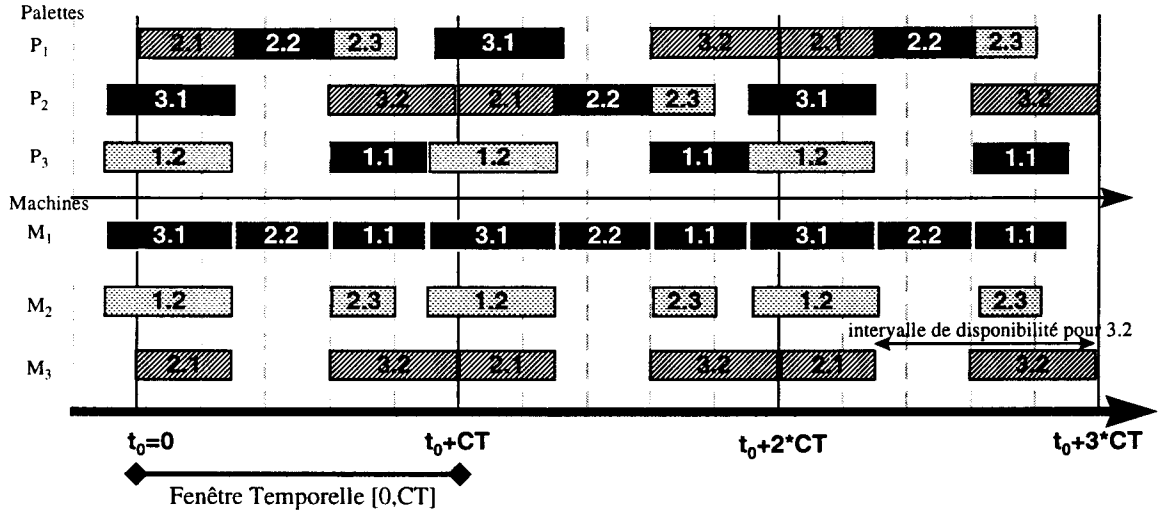


Figure IV-4 : Régime Permanent

Ainsi, si l'on considère l'encadrement du makespan résultant (42 u.t.≤ Makespan ≤50 u.t.) nous avons une marge d'optimisation de l'ordre de 20% de la durée totale de la production. Pour cet exemple, l'optimisation du régime transitoire est donc significative notamment si la borne inférieure est susceptible d'être atteinte.

IV.2.2.3 Etude du Régime Pré-production

Rappelons que, durant le régime transitoire de pré-production (démarrage de la production) aucune pièce ne doit être terminée. Nous allons étudier la durée de ce régime

* Nous rappelons que les différentes bornes dépendent des en-cours utilisés par les gammes et non des macro-gammes

en fonction de la date de début du régime permanent (également date de fin du pré-production).

Pour estimer une borne supérieure du régime de pré-production, nous utilisons la méthode de Munier, cf. [MUN 91]. Cette méthode duplique les opérations du régime permanent en amont, dans le régime transitoire, en éliminant les opérations inutiles, cf. Figure IV-5. Par conséquent, la borne supérieure du régime de pré-production en fonction de la Date de Début du Régime Permanent (DDRP) est obtenue par la formule suivante :

$$\text{Borne sup. Pré-production}(DDRP) = \max_i \left\{ CT * n_p(i, DDRP) - DDGL(i, DDRP) \right\}$$

où : n_p(i,DDRP) est le nombre de palettes utilisées par la gamme i à la date DDRP

DDGL(i,DDRP) ∈ [0, CT[est la **Date de Début de la Gamme Linéaire** i dans une fenêtre temporelle ayant pour origine la date DDRP, cf. Figure IV-1.

Nous notons également DFGL(i,DDRP) ∈]0, CT], la **Date de Fin de la Gamme Linéaire** i dans une fenêtre temporelle ayant pour origine la date DDRP. En fait, pour avoir les valeurs de la fonction DDGL pour une gamme i donnée, il suffit de prélever sa valeur DDGL(i,0) sur la fenêtre [0,CT] de la Figure IV-4 et :

$$DDGL(i,t) = DDGL(i,0) - t \quad \{+\text{CT} \quad \text{si} < 0\}$$ [1]

Equation IV-5 : Définition de la Date de Début d'une Gamme Linéaire

En ce qui concerne DFGL la même formule est applicable pour obtenir :

$$DFGL(i,t) = DFGL(i,0) - t \quad \{+\text{CT} \quad \text{si} \leq 0\}$$

Equation IV-6 : Définition de la Date de Fin d'une Gamme Linéaire

En appliquant les formules ainsi établies, nous obtenons pour la Date de Début du Régime Permanent : DDRP = 0 :

G_1 : $CT * n_p(1,0) - DDGL(1,0) = 10 * 1 - 6 = 4$

[1] Ceci correspond au changement d'origine du repère

G_2 : $CT * n_p(2,0) - DDGL(2,0) = 10 * 0 - 0 = 0$

G_3 : $CT * n_p(3,0) - DDGL(3,0) = 10 - 9 = 1$

Soit pour DDRP=0, la borne supérieure du régime de pré-production est égale à $\max\{0,1,4\} = 4$ *u.t.* Sur la Figure IV-5, nous avons représenté le cas DDRP=0. La gamme G_2 commence à la date 0. Donc toutes les opérations de cette gamme se trouvant en régime transitoire sont supprimées (on suppose ne pas produire de pièces entières durant le pré-production).

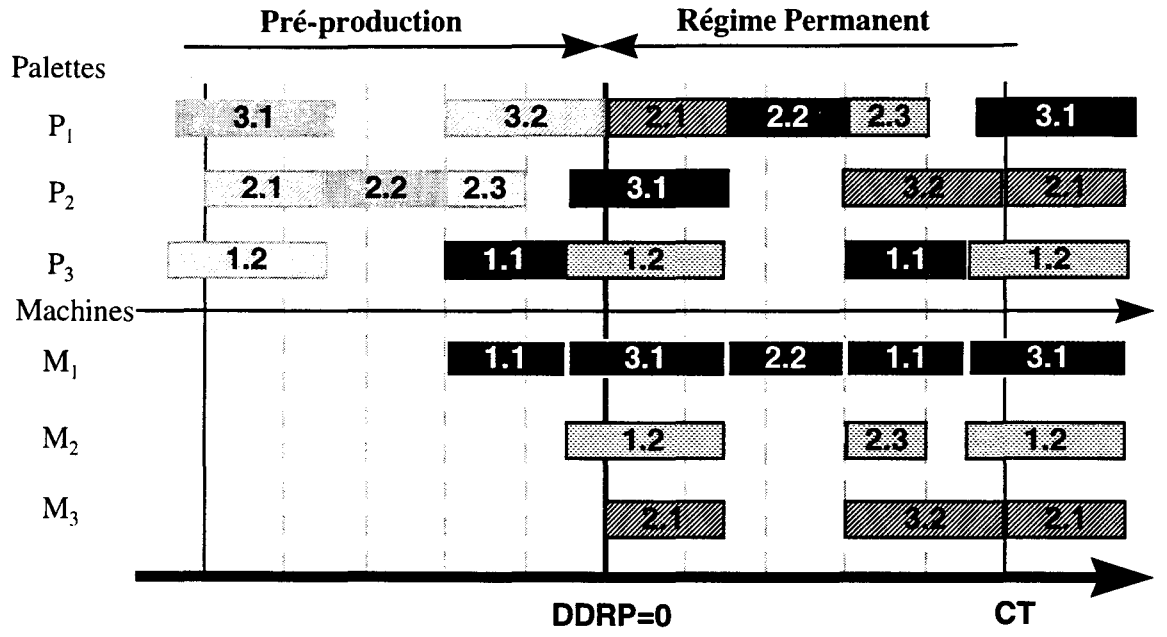


Figure IV-5 : Construction de la borne supérieure du pré-production

Pour la gamme G_3, la date DDRP traverse la première opération, dont toutes les opérations précédentes sont supprimées. Finalement, pour la gamme G_1 nous gardons, durant le régime de pré-production, les opérations nécessaires pour atteindre l'opération 1.2 et nous supprimons le reste. Les opérations à supprimer sont représentées en gris. Cette procédure est à réaliser pour toutes les dates possibles de DDRP c'est-à-dire pour les CT (= 10) dates, puisque nous avons supposé que les dates sont toutes multiples de l'unité de temps.

Pour la borne inférieure du régime de pré-production nous dupliquons, pour chaque palette P_j, les opérations $O(P_j)_{DDRP}$ à réaliser durant le transitoire sans tenir compte des

contraintes de ressources partagées. Ensuite, nous relevons pour chaque palette j la somme des charges de $O(P_j)_{DDRP}$ notée $Z(O(P_j)_{DDRP})$. Il vient alors :

$$\text{Borne inf. Pré} - \text{production}(DDRP) = \max_j \left\{ Z\left(O\left(P_j\right)\right)_{DDRP} \right\}$$

Dans l'exemple Figure IV-6, nous représentons le cas DDRP=4 :

Palette P_1 : $Z\left(O\left(P_1\right)\right)_4 = 4 \ u.t.$

Palette P_2 : $Z\left(O\left(P_2\right)\right)_4 = 4 \ u.t.$

Palette P_3 : $Z\left(O\left(P_3\right)\right)_4 = 0 \ u.t.$

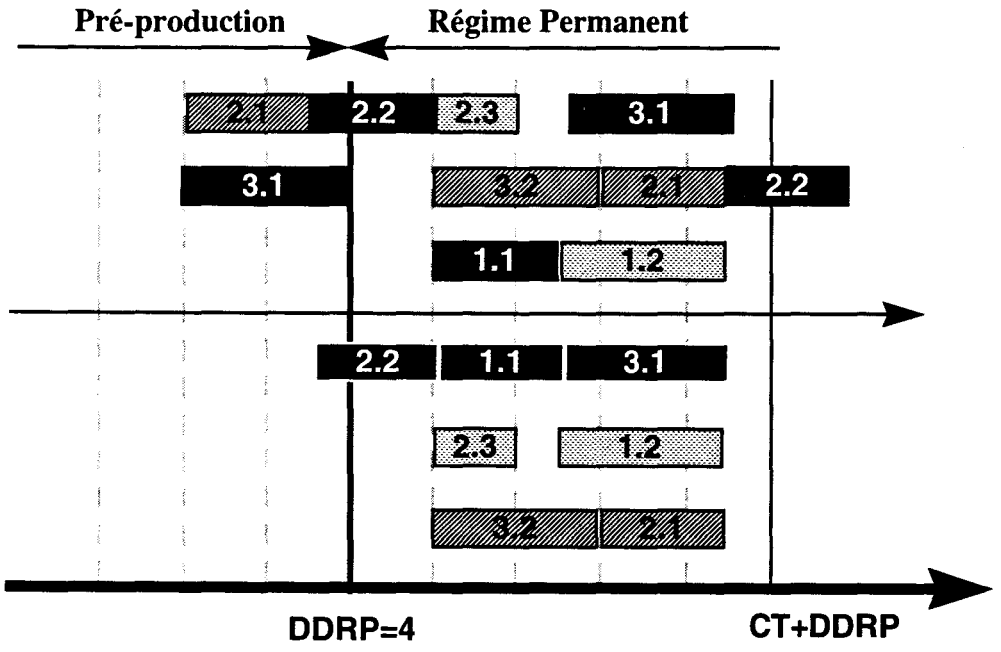


Figure IV-6 : Calcul de la borne inférieure du Pré-production

D'où $\text{Borne inf. Pré} - \text{production}(4) = \max\{0,4,4\} = 4 \ u.t.$ Bien entendu cette procédure est à mettre en oeuvre pour toutes les dates possibles de DDRP c'est-à-dire pour tous les éléments de {0,1,...,CT-1}, cf. Figure IV-7. Sur cette figure nous avons représenté l'évolution des bornes inf. et sup. ainsi que les durées optimales du régime pré-production en fonction de la date de début du régime permanent.

Ces courbes sont définies sur *IR* bien que, pour l'étude, nous ne nous intéressons qu'aux valeurs entières. Comme il en est de même pour les durées opératoires, aucun événement

de type début ou fin d'opération ne peut avoir lieu entre deux valeurs entières successives ce qui préserve la continuité et la monotonie des fonctions dans ces intervalles.

Nous pouvons alors remarquer que sur l'intervalle [0,3[([1]) les bornes sup. et inf. sont égales. Il n'est donc pas nécessaire d'optimiser ici le régime pré-production. Cependant pour les autres dates nous avons à ordonnancer les opérations afin d'en minimiser la durée. Nous pouvons constater l'existence de points de discontinuité pour les trois courbes.

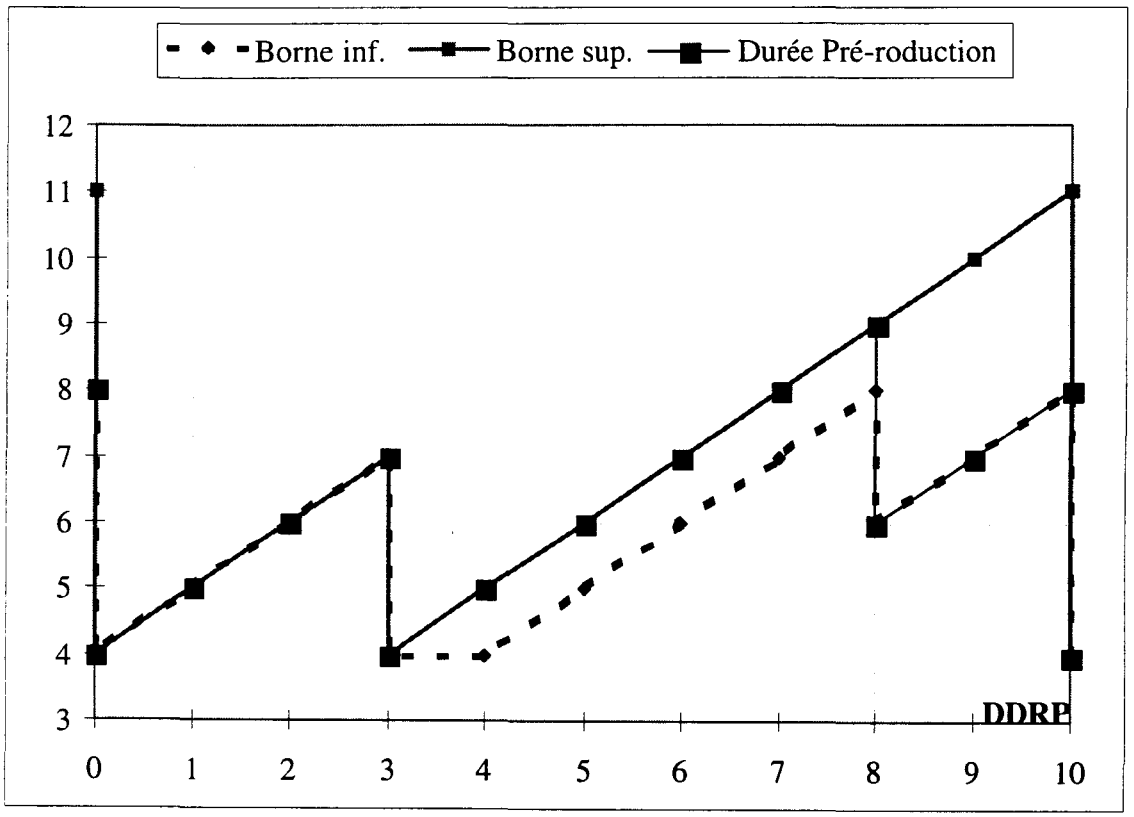


Figure IV-7 : Bornes et durée optimale du régime Pré-production

Lemme IV.1

Les bornes inf. et sup. ainsi que ***la durée du régime pré-production*** *sont* ***croissantes par morceaux.***

Démonstration : cf. Annexe IV.

En fait, ces points de discontinuité sont dus à l'hypothèse de non production durant le régime de pré-production. Ainsi, supposons que nous venons de déterminer ces bornes

[1] C'est à dire pour les valeurs entières 0,1,2.

pour DDRP = t et que, pour cette date, il existe i tel que DFGL(i,t) = 1([1]). Il est évident qu'en effectuant l'opération t = t+1, pour calculer ces mêmes valeurs pour la valeur suivante de début de régime permanent, on obtiendrait DFGL(i,t+1) = 0 correspondant à pièce entièrement produite en pré-production. Cette pièce est différée vers la fin de la production et DFGL(i,t+1) = CT ce qui est conforme à l'intervalle de définition de cette variable.

Par conséquent, si les bornes du régime de pré-production étaient contraintes par cette pièce, leurs valeurs subiraient une variation brusque. De ce fait, les fonctions borne sup. et inf. du pré-production sont continues à droite aux points de discontinuité. Signalons également que la borne min. peut être constante entre deux dates successives (cf. Dates 3 et 4 Figure IV-7) auquel cas elles sont également croissantes.

Concernant les valeurs optimales du régime de pré-production, cette fonction est également croissante par morceaux cependant les **points de discontinuité** ne sont pas uniquement des dates de fin de gammes mais aussi des **dates de fin d'opérations**. Dans ce cas, le fait de traverser une date de fin d'opération (et donc de l'inclure dans le transitoire) implique un changement dans le séquencement des opérations sur une machine, ce qui entraîne une diminution de la durée optimale du régime de pré-production. Entre deux dates successives de discontinuité la durée du régime de pré-production peut être constante et/ou croissante de coefficient directeur égal à 1.

IV.2.2.4 Etude du Régime Permanent

Ainsi que nous l'avons vu, à chaque période, nous lançons en production une nouvelle pièce brute de chaque gamme linéaire. Nous savons que X pièces de chaque gamme linéaire sont à fabriquer. Par conséquent, pour chaque gamme linéaire du système nous devons veiller à ne pas lancer le $X+1^{ème}$ exemplaire. Le régime permanent s'arrête à la première date de lancement virtuel du $X+1^{ème}$ exemplaire d'une gamme.

En réalité, si nous cherchons à minimiser le makespan du problème général d'ordonnancement, cette date ne représentera en fait qu'une borne sup. de la date de fin du régime permanent comme on le verra pour l'exemple Figure IV-10.

[1] DFGL(i,t)=1 signifie que la date t+1 est une date de fin de la gamme

Cependant, cette remise en cause peut logiquement aboutir à la suppression totale du régime permanent ce qui nous obligerait à réordonnancer complètement la production. Nous rappelons que nous chercherons un compromis entre le critère de minimisation du makespan et le temps de calcul.

C'est pourquoi nous nous sommes fixé quelques contraintes : utilisation des régimes permanents cycliques, aucune production pendant le régime pré-production et fin de régime permanent au lancement virtuel de la X+1ème pièce afin de limiter les temps de calcul.

Nous avons également vu qu'à la date de démarrage du régime permanent DDRP une gamme linéaire i utilise $n_p(i, DDRP)$ palettes et ainsi que $n_p(i, DDRP)$ pièces semi-finies de la gamme i circulent dans l'atelier. Ces $n_p(i, DDRP)$ pièces sont lancées durant le régime de pré-production. Alors, le 1er cycle du régime permanent introduit la $n_p(i,DDRP)$+1ème pièce et la X+1ème pièce est introduite durant le X-$n_p(i,DDRP)$+1ème cycle exactement à la date de début de la gamme linéaire i. D'où :

$$\text{Durée Régime Permanent } (DDRP) = \min_i \left\{ \left(X - n_p(i, DDRP) \right) * CT + DDGL(i, DDRP) \right\}$$

Equation IV-7 : Durée du Régime Permanent

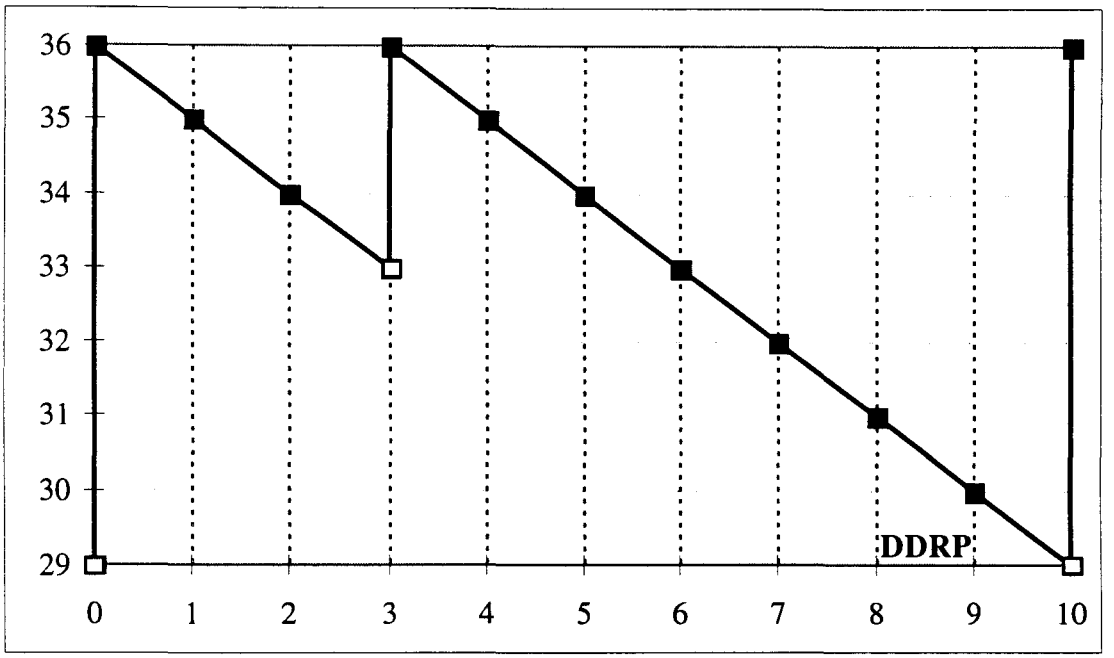


Figure IV-8 : Durée du régime permanent en fonction de la date de début du régime permanent

Lemme IV.2

*La **durée du régime permanent est strictement décroissante par morceaux**. Le coefficient directeur est égal à -1 et les points de discontinuité correspondent à des Dates de Fin de Gammes (DFGL).*

Démonstration : cf. Annexe IV.

En fait le régime permanent est délimité par les deux transitoires et ne dépend que de la date DDRP. Ces points de discontinuité sont donc inclus dans l'ensemble des dates de fin de gammes. La durée du régime permanent est continu à droite de ces dates de fin de gammes.

IV.2.2.5 Etude du Régime Post-production

Le régime de post-production démarre à la date de fin du régime permanent, c'est-à-dire à la date DDRP + Durée Régime permanent([1]). Il est chargé de terminer la production en vidant le système. Si nous utilisons le même principe que pour le régime de pré-production, les bornes de ce régime restent difficiles à déterminer car sa date de début, vu les origines temporelles choisies, est une fonction complexe des différents. Pour déterminer la borne supérieure nous allons garder la même résolution des conflits que pour le régime permanent. Nous voulons produire X pièces de chaque gamme linéaire i, alors la sortie de la $X^{ème}$ pièce de i se déroule pendant le $X^{ème}$ cycle du régime permanent à la date(repère d'origine DDRP) :

$$(X-1)*CT + DFGL(i, DDRP).$$

La borne supérieure du régime post-production pour la date DDRP s'exprime sous l forme suivante :

$$\text{borne sup. post - production } (DDRP) = \begin{array}{c} \max_i \{(X-1)*CT + DFGL(i, DDRP)\} \\ -\min_i \{(X - n_p(i, DDRP))*CT + DDGL(i, DDRP)\} \end{array}$$

étant donné que X ne dépend pas de i, l'équation se simplifie comme suit :

[1] Cette date est calculée par rapport au repère Figure IV-4, cependant pour des raisons de commodité nous utiliserons DDRP comme origine d'un nouveau où tout le pré-production se déroulera dans la partie "négative". La date de début du post-production devient donc égale à durée du Régime Permanent.

$$(X-1)*CT+\max_i\{DFGL(i,DDRP)\}-\left(X*CT+\min_i\{-n_p(i,DDRP)*CT+DDGL(i,DDRP)\}\right)$$

d'où l'expression finale de la borne supérieure du régime post-production :

$$\text{borne sup post production } (DDRP) = \begin{matrix}\max_i\{DFGL(i,DDRP)\}-CT \\ +\min_i\{n_p(i,DDRP)*CT-DDGL(i,DDRP)\}\end{matrix}$$

Equation IV-8 : Borne supérieure du régime post-production

Quant à la borne min., elle est plus délicate à mettre en œuvre. En effet, pour chaque palette P_j, nous devons déterminer l'ensemble $O(P_j)_{DDRP}$ d'opérations restant à réaliser. Ensuite nous calculons la charge totale $Z(O(P_j)_{DDRP})$ de chacun de ces ensembles et la borne s'écrit :

$$\begin{matrix}\text{Borne inf. de durée du} \\ \text{post}-\text{production } (DDRP)\end{matrix} = \max_j.\left\{Z\left(O(P_j)_{DDRP}\right)\right\}$$

Equation IV-9 : Borne Inférieure du post-production

Lemme IV.3

Les bornes inf. et sup. et ***la durée optimale du régime post-production*** *sont* ***constantes par intervalles.*** *Les points de discontinuité sont des Dates de Fin de Gammes Linéaires (DFGL).*

Démonstration : cf. Annexe IV.

Les points de discontinuité sont dus, comme pour le régime pré-production, à l'hypothèse de non production durant le transitoire de lancement. En effet, le fait de différer la production d'une pièce entière durant le transitoire post-production augmente le nombre d'opérations à traiter durant ce régime et peut causer une augmentation des bornes relatives.

Concernant la durée optimale du post-production, elle est obtenue en réordonnançant les opérations se trouvant en post-production afin d'en minimiser la durée. Cette fonction est constante par intervalles. Ses points de discontinuités sont inclus dans l'ensemble des dates de fin de gammes linéaires.

IV.2.3 Optimisation du makespan

Nous cherchons en premier lieu à minimiser le makespan. Nous allons donc calculer séparément les trois régimes de production et en calculer les durées optimales en fonction de la date de début du régime permanent (DDRP). La procédure donnant le résultat optimal du makespan passe impérativement par une recherche exhaustive de toutes les valeurs de sa fonction, cf. [KOR 97b] :

DDRP = 0, DDRPop=0 /* Initialisation */
Makespan = borne sup. du makespan optimal
Tant que (Makespan > borne inf. makespan et DDRP < CT)
    /* *Pré-production* */
    Conserver les opérations traversées par la date DDRP telles quelles
    Ordonnancer au plus tard les opérations restantes du pré-production (min. durée)
    Déterminer durée pré-production
    /* *Permanent* */
    Calculer durée permanent (cf. Equation IV-7)
    /* *Post-production* */
    Début post production = DDRP + durée permanent
    Conserver les opérations traversées par début du post-production telles quelles
    Ordonnancer au plus tôt les opérations restantes du pré-production (min. durée)
    Déterminer durée post-production
    /* *Makespan* */
    aux = durée pré-production + durée permanent + durée post-production
    Si aux ≤ Makespan alors Makespan = aux et DDRPop = DDRP
    DDRP = DDRP+1
Fin tant que

Cette procédure donne le résultat décrit Figure IV-9. Nous y trouvons les durées optimales du régime transitoire (pré + post), du régime permanent et du makespan. Nous avons également représenté les bornes inf. et sup. du makespan pour mieux apprécier la qualité des résultats.

Nous avons représenté en "symboles vides" les limites à gauche non atteintes et en symbole plein la valeur de ces fonctions continues à droite en ces mêmes points. Ces points de discontinuité ne correspondent pas nécessairement tous à des dates de fin de gammes.

Lemme IV.4

La durée optimale du régime transitoire est ***croissante par intervalles.***

Démonstration : cf. Annexe IV.

Nous avons démontré que des dates de fin d'opérations peuvent causer un changement conséquent dans le séquencement des opérations sur les machines durant le transitoire de pré-production et par conséquent provoquer à cette date une discontinuité dans la durée du makespan optimal, cf. [KOR 98] et § IV.4.1.

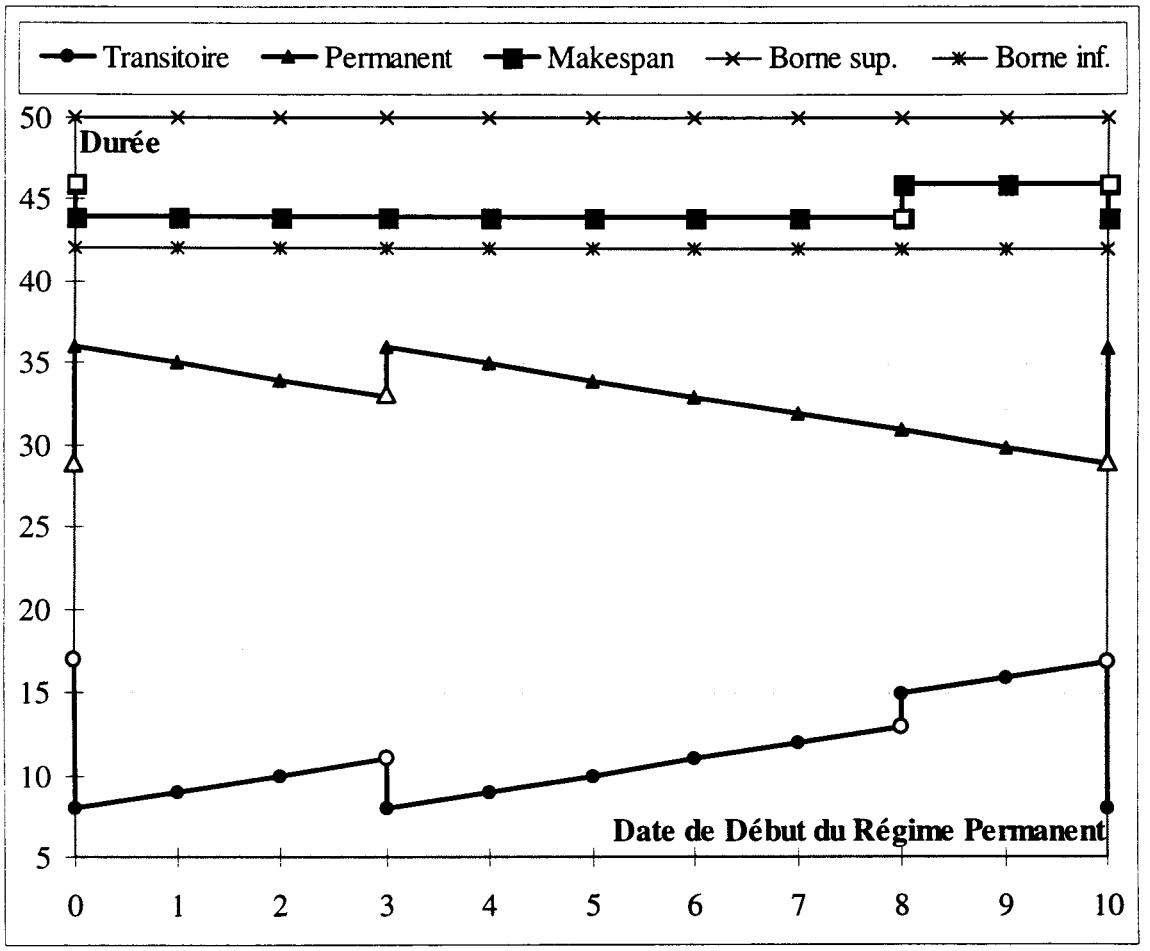


Figure IV-9 : Durées du Makespan et des régimes Transitoire et Permanent

Lemme IV.5

Entre deux dates de fin de gamme successives le makespan est décroissant.

Démonstration : cf. Annexe IV.

Nous obtenons alors que le makespan minimal pour notre exemple est égal à 44 unités de temps. Cette valeur est atteinte pour DDRP $\in$ [0, 8[. Dans cet intervalle, nous conservons les dates qui maximisent la durée du permanent c'est-à-dire qui correspondent à une durée du régime permanent de 36 u.t. soit DDRP $\in$ {0,3}.

Ces dates sont équivalentes du point de vue performances. Elles donnent les mêmes valeurs pour les régimes de pré-production et de post-production.

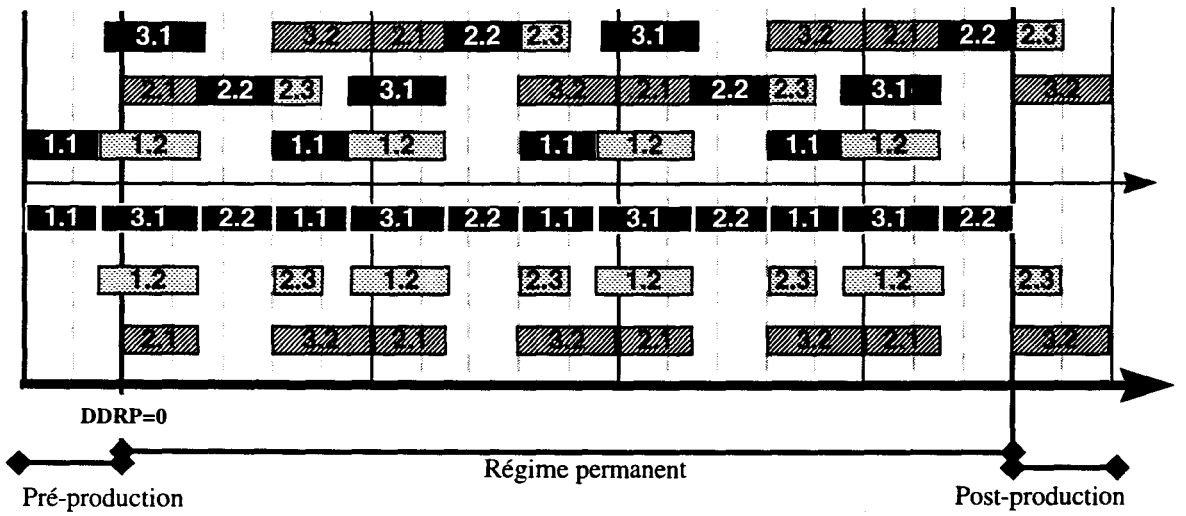


Figure IV-10 : Ordonnancement final

Nous voyons très bien sur cette figure qu'il est tout à fait possible d'optimiser encore le makespan (il suffit d'avancer l'opération 3.2 du post-production de 3 u.t.). Ceci réduirait le makespan de 44 à 42 conduisant ainsi dans ce cas particulier à la solution optimale minimisant réellement le makespan puisque makespan réel = borne inf. du makespan.

Cependant, ceci n'est pas généralisable pour plusieurs raisons :

- cet exemple est un cas particulier où la borne inf. est atteignable ce qui n'est pas généralement le cas. Nous ne connaissons pas a priori la valeur optimale sans avoir résolu de manière exhaustive le problème d'ordonnancement.

- l'approche consistant à essayer de réduire le régime permanent pour minimiser le makespan ne respecte pas les hypothèses fixées par notre méthode (début du post-production au lancement virtuel de la X+1ème pièce) et peut conduire au réordonnancement complet de la production ce qui est ici exclu par hypothèse.

Nous avons représenté, cf. Figure IV-10, l'ordonnancement total correspondant à la Date de Début du Régime Permanent égale à zéro. Nous pouvons distinguer le régime de pré-production (de durée 4 u.t.), le régime permanent (de durée 36 u.t.) et le régime de post-production (de durée 4 u.t.) soit un makespan égal à 44 u.t.

Nous pouvons remarquer que le régime transitoire ainsi calculé est indépendant du nombre de répétitions X de l'horizon de production. En effet, $\forall\ X \geq 1$ l'ordonnancement du régime transitoire que nous venons de déterminer est applicable et optimal. Le rapport transitoire/makespan dépend donc de X (nombre de répétitions de l'horizon de production). Cependant, ce qu'il faut surtout noter c'est que le transitoire associé à cette production est indépendant de X.

Pour cet exemple, nous obtenons :

$$\frac{Transitoire}{Makespan}(X) = \frac{8 \text{ u.t.}}{X * CT + 4 \text{ u.t.}}^{1}, \quad \frac{Transitoire}{Makespan}(X) = \frac{6 \text{ u.t.}}{X * CT + 4 \text{ u.t.}}^{2}$$

Ajoutons que, si nous voulons introduire une production urgente ou une opération de maintenance éminente, nous avons la possibilité de le faire « proprement » une fois par cycle et sans calcul supplémentaire.

Pour évaluer l'importance du régime transitoire pour cette production (X = 4), nous traçons la courbe du rapport entre le transitoire et le makespan en fonction de DDRP, cf. Figure IV-11. Nous constatons alors que ce rapport est compris entre 0,18 et 0,37 pour des valeurs presque constantes du makespan d'où l'intérêt de l'étude exhaustive du Makespan en fonction des différentes valeurs de DDRP et de la minimisation du transitoire.

[1] Avant décalage de l'opération 3.2

[2] Après décalage de 3.2

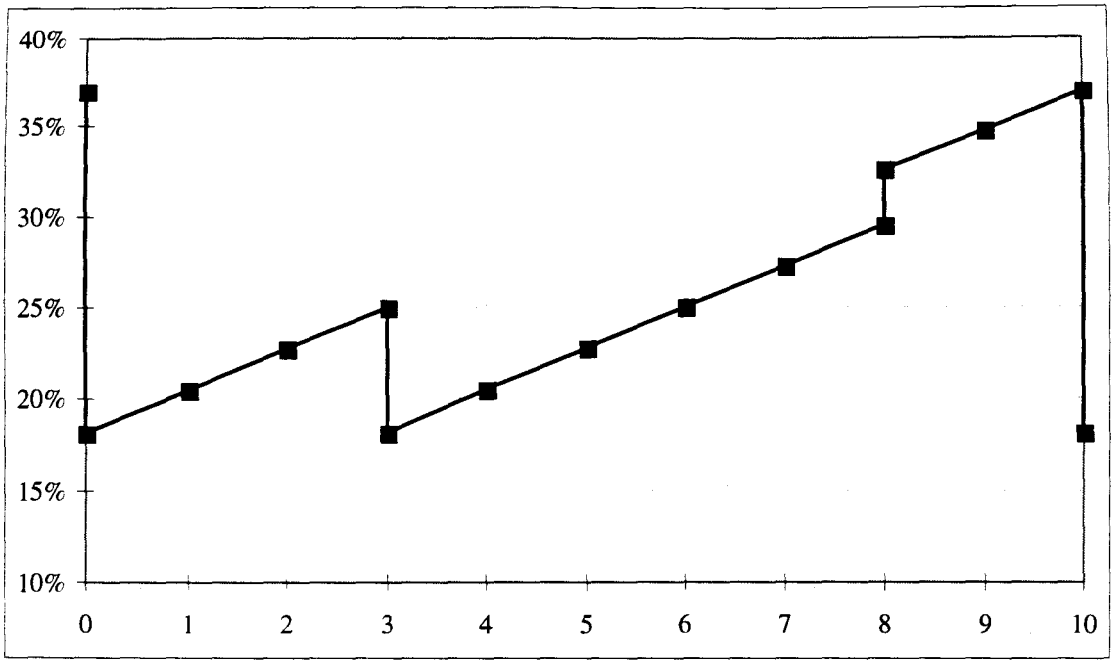


Figure IV-11 : Rapport transitoire / makespan

IV.2.4 Heuristique de détermination de la commande

Il est clair que le problème est d'une combinatoire assez importante puisqu'il faut énumérer CT dates possibles ce qui contribue à justifier l'idée déjà avancée de réduction de la valeur de la période CT. Il faut ensuite, pour chaque date, ordonnancer au plus tard les opérations du régime de pré-production, calculer la date de début du post production et ordonnancer les opérations du post-production au plus tôt.

De ce fait nous avons mis en place une heuristique qui a pour objectif essentiel la minimisation du makespan en réduisant l'étude aux seules Dates de Début du Régime Permanent pertinentes.

Proposition IV.1

*$\exists$ une gamme i / **le makespan optimal est atteint** pour DDRP = DFGL(i,0)-1*

Démonstration : Par l'absurde (cf. Annexe IV.)

A partir de ce lemme, nous déduisons qu'en étudiant l'ensemble {DFGL(i,0)-1 / i $\in$ [1, card(E)]} nous sommes certains d'atteindre le makespan optimal (correspondant aux hypothèses de l'étude bien entendu).

IV.2.4.1 Principe de l'heuristique

La procédure principale se charge de parcourir l'ensemble {DFGL(i,0)-1 / i ∈ [1, card(E)]} et, pour chacun de ses éléments, de lancer les modules suivants : élimination des dates inutiles dans le régime de post-production puis dans le régime de pré-production et de minimiser le makespan. A chaque itération, le makespan résultant est comparé à ceux déjà obtenus. Il est conservé s'il est meilleur que les makespan déjà établis. Ainsi, pour une date d ∈ {DFGL(i,0)-1 / i ∈ [1, card(E)]} nous définissons une représentation du régime permanent ayant pour origine d : [d, d+CT]. Nous dupliquons cette fenêtre temporelle $X + n_{pmax}(d)$([1]) fois de façon à avoir un ordonnancement admissible contenant toutes les opérations nécessaires pour la production à réaliser.

Par exemple, nous appliquons cette procédure pour DDRP=DFGL(1,0) - 1 = 2, cf. Figure IV-12,. Nous avons $n_{pmax}(2)=1$ par conséquent nous représentons un ordonnancement contenant $X+n_{pmax}(2)=4+1=5$ périodes. Pour faciliter la compréhension de l'étude nous mettons l'origine de cette représentation de façon à ce que la production commence à la date $-n_{pmax}(d)*CT$ et qu'elle se termine à la date X*CT. Ceci permet de caractériser le régime de pré-production durant les dates "négatives" et le début du régime permanent à partir de la date 0.

Cependant dans cet ordonnancement apparaissent plus d'opérations que nécessaire. C'est pour cela que nous allons éliminer les opérations inutiles. Nous commençons par les opérations tronquées, c'est-à-dire celles qui ne rentrent pas entièrement dans la représentation de la commande totale. Ces opérations sont représentées en « gris clair » sur la Figure IV-12. Ensuite nous nous intéressons aux opérations du post-production. Nous savons que la sortie du $X^{ème}$ exemplaire de chaque gamme à lieu durant l'intervalle [(X-1)*CT, X*CT]([2]). A partir de la date X*CT, pour chaque palette p, nous « remontons le temps » en éliminant les opérations jusqu'à rencontrer la dernière opération d'une gamme, cf. Figure IV-12.

Pendant le régime de pré-production, nous avons émis l'hypothèse de non production totale d'une pièce. Donc, à partir de la date 0 et jusqu'à la date $-n_{pmax}(d)$, pour chaque palette, nous conservons toutes les opérations rencontrées jusqu'à la première opération

[1] $(X + n_{pmax}(d))*CT$ est une borne sup. du makespan pour la date d

[2] à partir du début du régime permanent nous terminons une pièce de chaque type par période

d'une gamme (incluse) ou la dernière opération d'une gamme (non incluse). Puis, nous éliminons les opérations restantes.

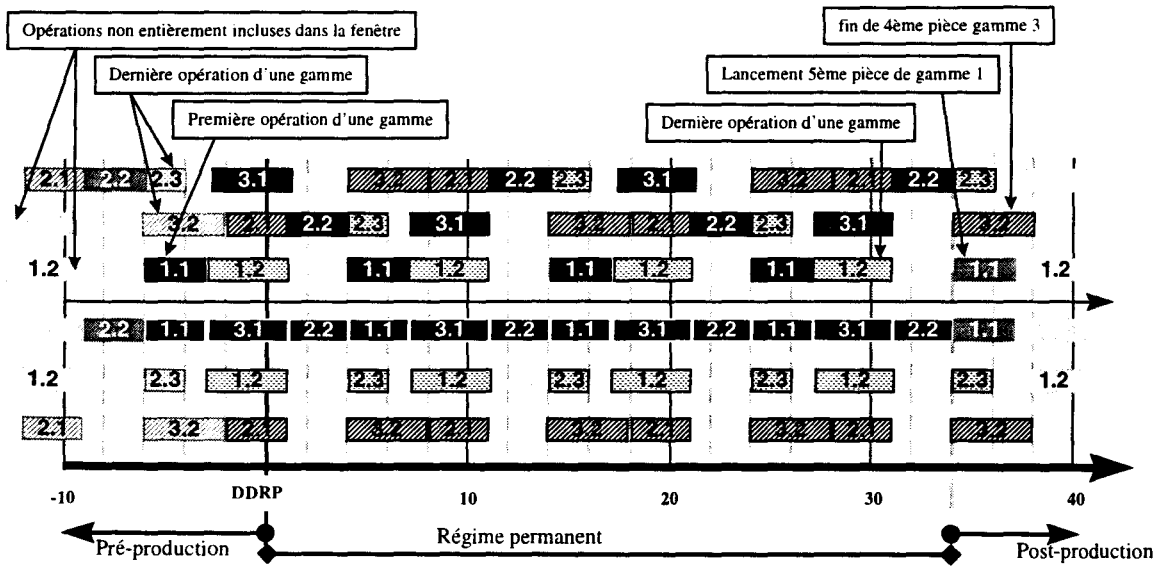


Figure IV-12 : Elimination des opérations inutiles

IV.2.4.2 Enoncé

Les données dont nous avons besoin pour cette heuristique sont les suivantes : $n_{pmax}(t)$, X, CT, nb_palettes (nombre de palettes utilisées par le système), Gamme (i) et NO(Gamme(i)) (nombre d'opérations de la gamme i). De même, pour toute opération Op, nous définissons :

g : la gamme associée

Dd : la date de début de l'opération dans la fenêtre considérée

d : la durée de l'opération

N : le numéro de l'opération dans la gamme

p : palette associée à l'opération Op

L'énoncé de la procédure principale ainsi proposée est donc le suivant :

Procédure Principale
M = borne sup. du makespan optimal Pour d ∈ {DFGL(i,0)-1 / ∀ i ∈ {1, Card(E)}} Définir une fenêtre F=[d, d+CT] du régime permanent Dupliquer F (X + n_{pmax}(d)) fois *Procédure éliminer op en post*

```
    Procédure éliminer op en pré
    Procédure optimiser Makespan(M_aux)
    Si M_aux < M
    Alors
        Mémoriser ordonnancement
        M = M_aux
    Fin Si
Fin pour
```

Les procédures de suppression des opérations inutiles au régime transitoire, comme l'illustre la Figure IV-13, sont les suivantes :

```
Procédure éliminer op en post
Dpost = X*CT                                    /*Durée post-production*/
t = X*CT
Pour p=1 jusqu'à nb_palettes
    Tant que t > 0
        Si ∃ opération Op
        Alors
            Si Op.Dd + Op.d > (X + n_pmax)*CT ou Op.N <> NO(Op.G)
            Alors
                Eliminer Op
                t = Op.Dd + 1
                If Op.Dd < Dpost Alors Dpost = Op.Dd
            Fin Si
        Fin Si
    t = t – 1
    Fin Tant que
Fin pour
```

```
Procédure éliminer op en pré
Dpré = n_pmax(d) * CT                           /* Durée pré-production */
t = n_pmax(d) * CT
Pour p=1 jusqu'à nb_palettes
    Tant que t > - n_pmax(d) * CT
        Si ∃ opération Op
        Alors
            Si Op.N = 1
            Alors
                Si Op.Dd < Dpré Alors Dpré = Op.Dd
                Pour aux = Op.Dd jusqu'à - n_pmax(d) * CT Faire
                    Eliminer toutes les opérations
                t = - n_pmax(d) * CT
            Sinon
                Si Op.N = NO(Op.G)
                Alors
                    Pour aux = 0 jusqu'à - n_pmax(d) * CT Faire
```

```
                                        Eliminer toutes les opérations
                                    t = - npmax(d) * CT
                                Fin Si
                            Fin Si
                        Fin Si
                        t = t – 1
                Fin Tant que
        Fin pour
```

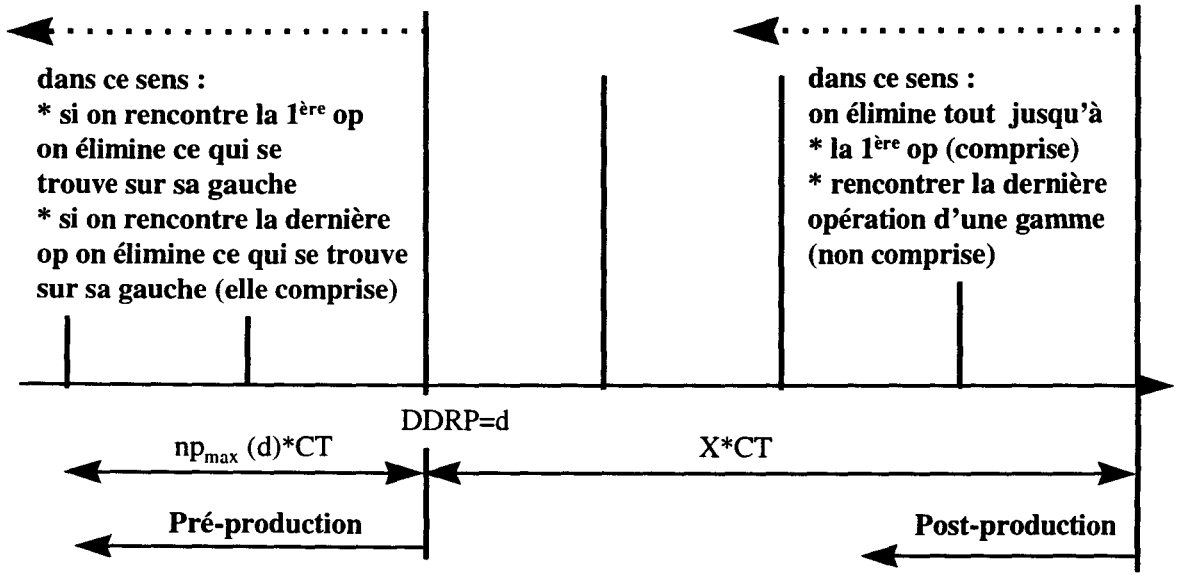


Figure IV-13 : Schéma récapitulatif des procédures suppression des opérations

La dernière étape à réaliser concerne l'optimisation du makespan. Nous avons opté pour l'ordonnancement au plus tard des opérations du régime de pré-production et au plus tôt des opérations du régime post-production, cf. [GOT 93] et [CAR 96]. Nous verrons § IV.8.1. que d'autres méthodes sont également possibles. Ces ordonnancements au plus tôt et au plus tard doivent pouvoir remettre en question les contraintes de précédence, établies pour le régime permanent, des opérations sur les différentes machines. De plus, les distributions des charges sur les machines doivent pouvoir être réétudiées pour une meilleure optimisation de la durée du makespan.

IV.2.4.3 Optimalité des résultats de l'heuristique

La minimisation du makespan pour l'exemple de la Figure IV-14 donne le résultat suivant : nous pouvons constater que makespan est de 44 u.t. (le minimum possible avec cet exemple). Cependant, le résultat ne respecte pas le critère de minimisation de la durée du régime transitoire (ici sa durée est de 10 u.t. contre 8 u.t. pour le minimum). Ce décalage est dû au fait que nous avons préféré trouver le makespan optimal en un minimum

de calcul soit en étudiant uniquement l'ensemble {DFGL(i)-1 / $i \in [1, Card(E)]$}. En conséquence, nous sommes assurés de trouver le minimum du makespan mais sans satisfaire l'optimisation du deuxième critère de performance.

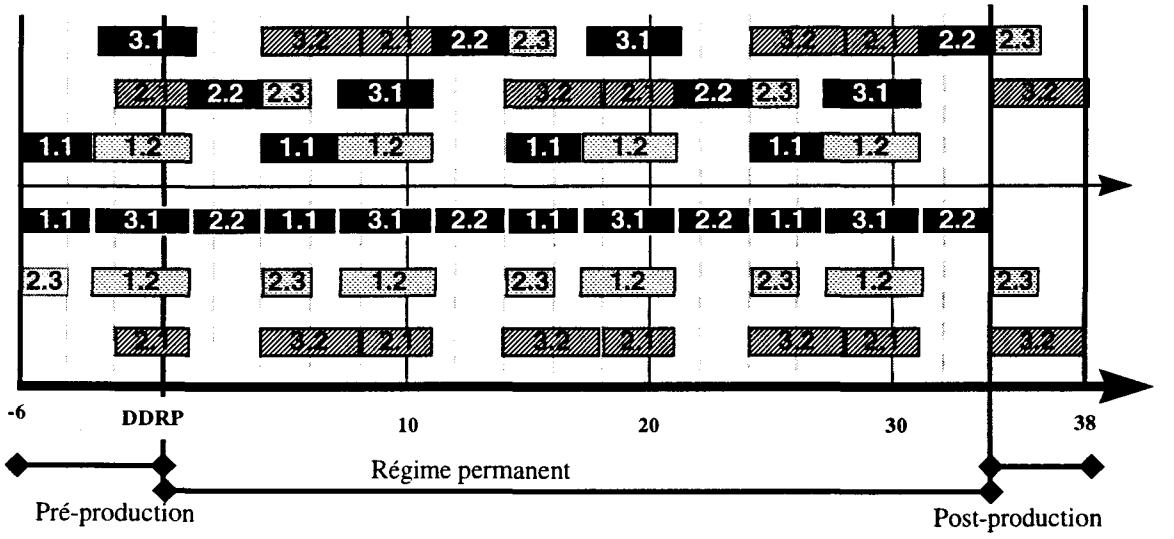


Figure IV-14 : Résultat de l'Heuristique

IV.3 Application à l'exemple illustratif du chapitre III

Nous allons à présent appliquer les résultats précédents sur l'exemple plus conséquent, présenté au chapitre III, afin de déterminer la commande prévisionnelle totale dans un contexte plus significatif et plus réaliste. Nous rappelons que le régime permanent déterminé durant le chapitre précédent, cf. Figure III-8, est caractérisé par l'horizon de production E = {9 A, 6 B, 9 C} à répéter 3 fois (X=3). A la fin de l'ordonnancement du régime permanent, nous obtenons les bornes suivantes du makespan, cf. Equation IV-4, Equation IV-2 et Annexe IV :

$$\text{Borne inf. du makespan} = \underbrace{X(q)*CT(RP_q)}_{\substack{\text{charge d'une} \\ \text{machine critique}}} + \max_{\substack{\text{machine} \\ \text{critique m}}} \left(f(m)+g(m)\right) = 3*24+17 = 89 \text{ u.t.}$$

$$\text{avec} \begin{cases} f(m) = \min_{l \in \{1,\ldots,card(E)\}} \left(\sum_{n=1}^{prem(GL_q(l),m)-1} Z(l,n) \right) \\ g(m) = \min_{l \in \{1,\ldots,card(E)\}} \left(\sum_{n=dern(GL_q(l),m)+1}^{NO(GL_q(l))} Z(l,n) \right) \end{cases}$$

$$\text{Borne sup. du makespan Optimal} = \left(X + \min_{t}\left\{ n_{p\max}(t) \right\} \right) * CT$$
$$= (3+2) * 24 = 120 \text{ u.t.}$$

Equation IV-10 : Application du calcul des bornes à l'exemple illustratif.

La marge d'optimisation est donc de l'ordre de 30%. L'application de l'heuristique nous amène à étudier 7 dates : {DFGL(i,0)-1 / i ∈ [1, card(E)]}={0-1, 2-1, 5-1, 7-1, 11-1, 17-1, 23-1}={1, 4, 6, 10, 16, 22, 23}, cf. Annexe IV pour les différents calculs.

Le minimum du makespan est obtenu pour la date DDRP = 1 et est égal à 95 u.t. (49 u.t. pour le permanent et 46 u.t. pour le transitoire), cf. Figure IV-15.

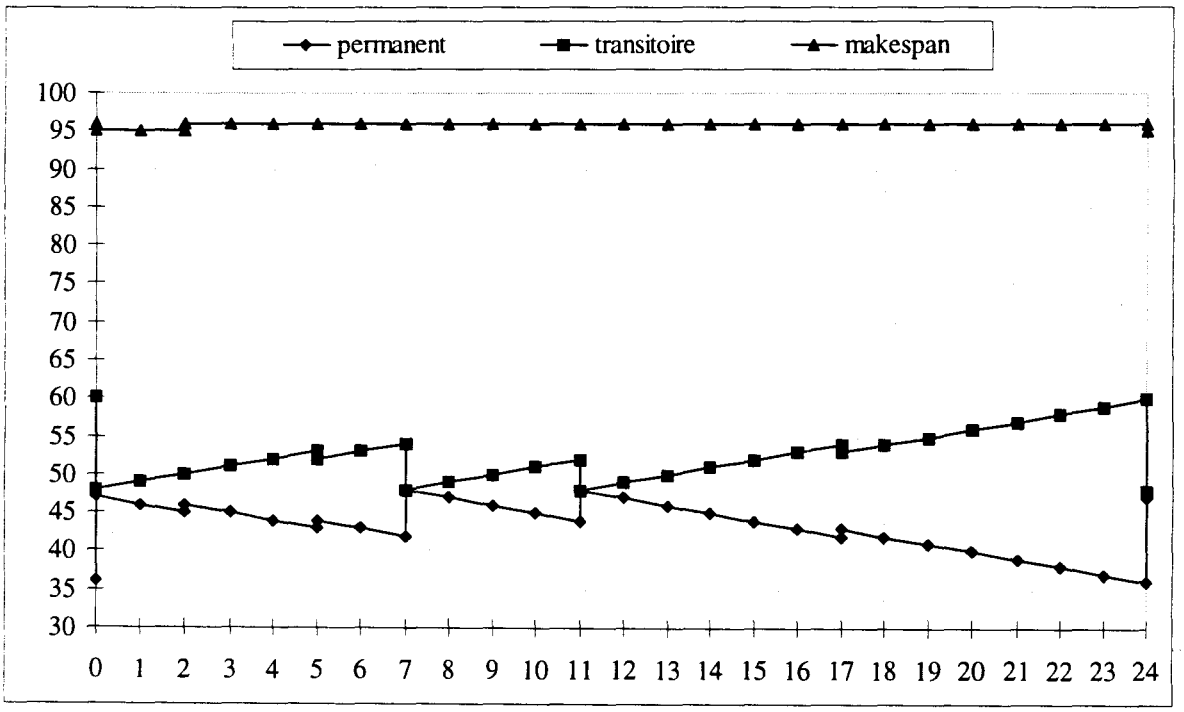


Figure IV-15 : Durées du régime permanent, transitoire et du makespan

IV.4 Analyse des résultats

Avant de continuer l'étude des régimes transitoires nous récapitulons dans le tableau ci-dessous les principaux résultats. Ce tableau se base sur les différents lemmes et résultats intermédiaires que nous avons mis en place :

	Pré-production	Permanent	Post-production	Makespan
discontinuité en dehors des DFGL	possible	non	non	possible
monotonie entre deux points de discontinuité	croissant	décroissant	constant	décroissant

Il apparaît bien en conclusion une règle simple d'évaluation précise des valeurs optimales des durées des différents régimes :

- **régime permanent** : il est strictement décroissant entre deux dates de fin de gamme avec un coefficient directeur égal à -1, il suffit donc d'étudier les Dates de Fin de Gammes Linéaires et de les extrapoler avec des fonctions linéaires de coefficient -1.
- **régime de post-production** : comme il est constant entre deux dates de fin de gamme, il suffit d'étudier les Dates de Fin de Gammes Linéaires et de les extrapoler avec des fonctions constantes.

En ce qui concerne les valeurs du **régime de pré-production** nous avons à parcourir l'ensemble des dates de fin d'opérations (ce qui est confondu, en général, avec l'ensemble des CT dates possibles pour DDRP).

IV.5 Influence des paramètres

Après avoir étudié l'influence de la date de début du régime permanent, nous allons nous intéresser aux autres paramètres pouvant influencer le régime transitoire voire le makespan. Ces différents paramètres ne sont pas indépendants les uns des autres, c'est pour cela que nous essaierons dans la mesure du possible d'étudier l'influence de chaque paramètre pris séparément.

IV.5.1 Paramètres liés à l'ordonnancement

IV.5.1.1 Flexibilité de tir des transitions du Graphe d'Evénements

La première flexibilité, dont nous disposons, est liée au tir des transitions du Graphe d'Evénements, cf. IV.2.2.2. Nous avions opté pour une stratégie de tir au plus tard de l'opération 3.2 pour l'exemple Figure IV-3, cf. Figure IV-4. Nous considérons maintenant un tir au plus tôt de cette opération (l'opération 3.2 s'effectue durant l'intervalle [3, 7] sur

la fenêtre temporelle [0, CT]) sans modifier les autres opérations. En étudiant les différentes valeurs de DDRP, nous obtenons aussi le résultat Figure IV-16. Afin de mieux voir les différences engendrées par le déplacement de l'opération 3.2, nous avons ajouté, en pointillé, la courbe de valeurs du makespan(DDRP) associé au placement au plus tard de 3.2, cf. Figure IV-9.

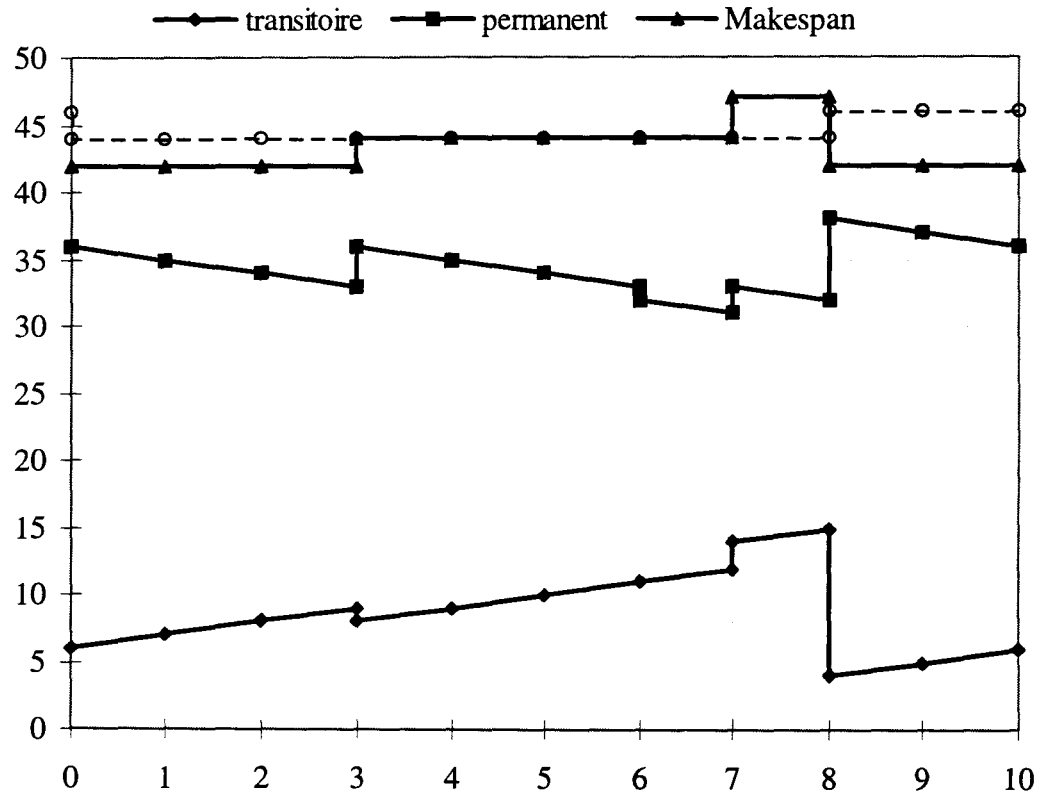


Figure IV-16 : Durées du régime permanent, transitoire et du Makespan

La première constatation concerne le makespan optimal. En effet, avec un placement au plus tôt de la transition 3.2, le makespan optimal passe de 44 u.t. à 42 u.t. Cette valeur correspond à la borne inf. du makespan optimal. Nous sommes alors certains de l'optimalité de la solution. Le makespan minimal (42 u.t.) correspondant à DDRP = 0. Nous présentons la commande totale correspondant à cette valeur, cf. Figure IV-17.

Le maximum du makespan est également affecté par le déplacement de l'opération 3.2 : il passe de 46 u.t. à 47 u.t. Par conséquent, l'interprétation du Graphe d'Evénements, c'est-à-dire la politique de tir des transitions, conditionne en partie le résultat final. Au contraire, le tir de l'opération 1.2 au plus tard n'engendre aucune conséquence sur le résultat final. L'effet de déplacement d'opérations est alors totalement imprévisible.

Nous ne pouvons donc nous permettre d'étudier toutes les possibilités dues aux flexibilités de tir des opérations du Graphe d'Evénements sans augmenter de manière conséquente la complexité de notre approche. Nous utiliserons toujours la représentation par diagramme de Gantt obtenue par l'heuristique d'ordonnancement sauf si le makespan obtenu n'est pas satisfaisant, auquel cas, nous pouvons chercher à tester toutes les possibilités.

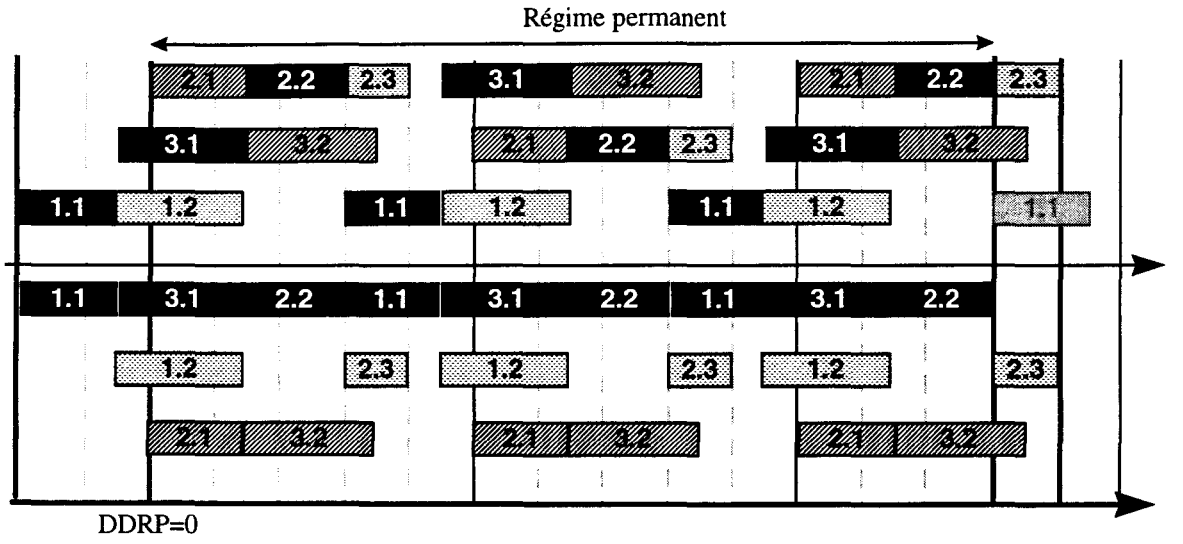


Figure IV-17 : Commande optimisant le makespan

IV.5.1.2 Macro-gammes et en-cours

Nous avons remarqué que les différents calculs effectués jusqu'alors dépendent du nombre de palettes utilisées par chaque gamme et non pas par les macro-gammes. En effet, comme nous cherchons à produire les différentes pièces, nous devons différencier les gammes y compris celles appartenant à des macro-gammes.

L'exemple de la Figure IV-18 présente deux ordonnancements différents (utilisant les mêmes dates de déclenchement des transitions) de l'exemple Figure IV-3. Le premier ordonnancement procède à une parallélisation totale des gammes et le second regroupe G_2 et G_3 en une macro-gamme. Dans ce cas, nous obtenons exactement les mêmes régimes transitoires et le même makespan. Nous rappelons que l'utilisation de ces macro-gammes nous a permis de minimiser l'en-cours utilisé par le système.

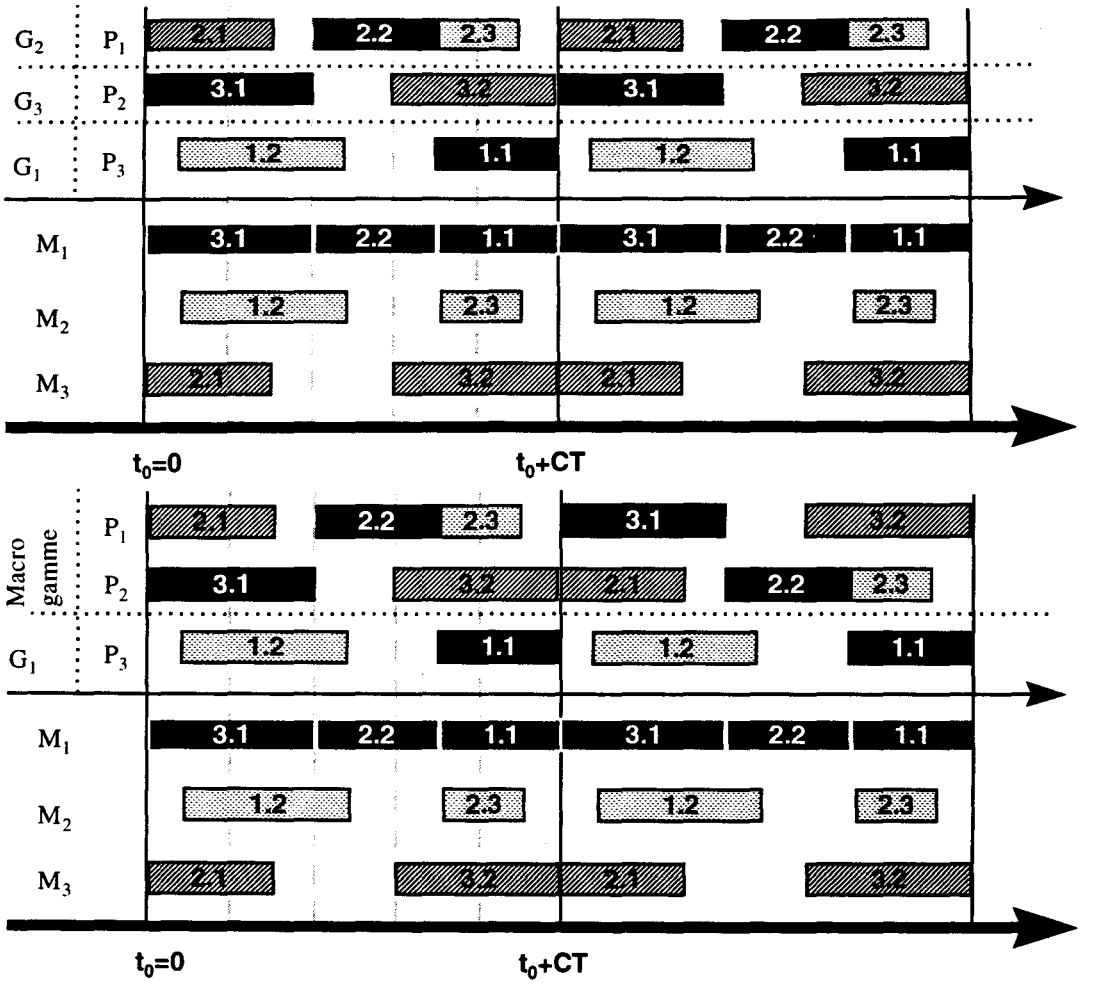


Figure IV-18 : Influence des macro-gammes

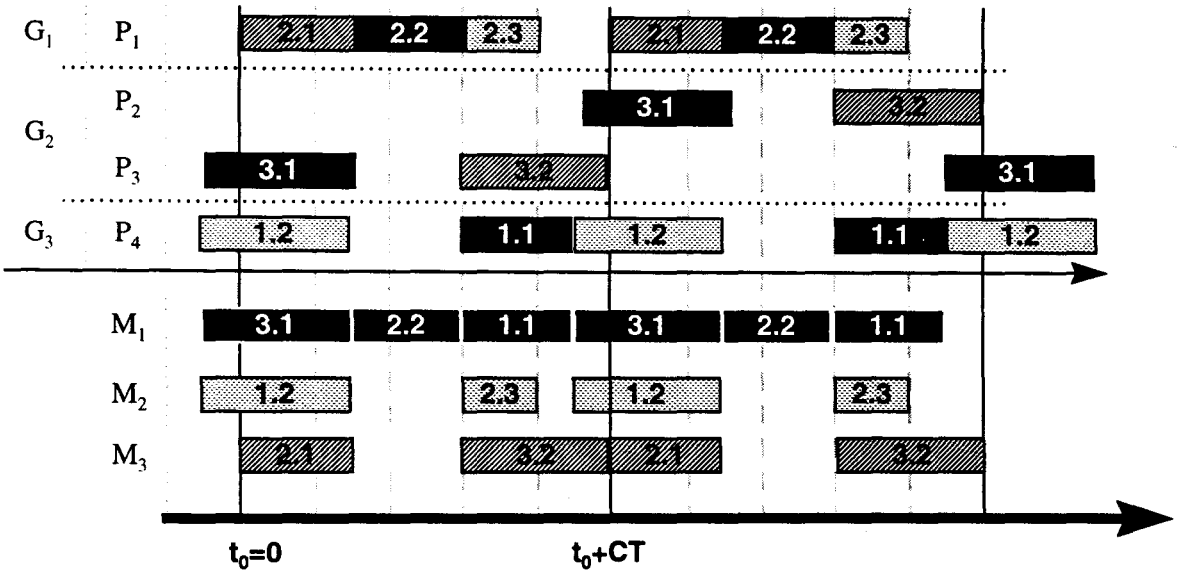


Figure IV-19 : Influence de la minimisation de l'en-cours du système

En gardant les mêmes dates de franchissement des transitions, que dans l'exemple Figure IV-4, et en dissociant la macro-gamme en deux gammes linéaires G_2 et G_3, nous obtenons le résultat Figure IV-19. Dans ce cas, même si l'en-cours total utilisé par le système a augmenté nous obtenons tout de même le même transitoire et le même makespan. Ceci est dû au fait que l'en-cours utilisé par chaque gamme n'a pas changé. Cependant, si nous augmentons le nombre de palettes utilisées par une gamme, le résultat final risque d'être modifié. En effet, la durée du régime permanent est modifiée puisqu'elle est fonction du nombre de palettes.

IV.5.1.3 Marges de gammes

Il est difficile d'isoler l'effet d'une modification sur les marges de gamme. En effet l'ajout d'une marge de gamme peut :

- modifier uniquement la date de tir d'une opération (sans influencer les autres), auquel cas nous retrouvons un problème d'utilisation de flexibilité du Graphe d'Evénements, cf. IV.5.1.1 ;
- modifier la marge entre deux gammes (déplacement du lancement de la gamme uniquement) sans modifier les dates de tir des autres opérations : nous avons alors le même cas que précédemment ;
- modifier les dates de tir d'un certain nombre d'opérations de la même gamme, dans ce cas nous changeons les dates de fin d'opérations (dates potentielles de discontinuité dans la durée du régime transitoire) ;
- modifier le nombre de palettes utilisées par la gamme linéaire, auquel cas nous retrouvons l'influence de l'en-cours sur le makespan.

IV.5.2 Influence des phases précédant l'ordonnancement

Après avoir passé en revue les facteurs d'influence dus à l'ordonnancement, nous allons analyser ceux qui se situent en amont de l'heuristique. Ces paramètres sont plus difficiles à étudier car ils se situent très loin de l'optimisation du transitoire (puisqu'il y a l'algorithme d'ordonnancement dont on ne peut prédire le résultat).

Parmi ces facteurs, nous trouvons l'optimisation du flux de production durant le régime permanent qui se traduit par la minimisation du temps de cycle. Il est évident que le régime

permanent correspondant à une production à flux non optimal est plus long que celui d'un flux optimal. Par conséquent, avec un séquencement identique sur les machines, nous pouvons espérer avoir les mêmes durées pour les régimes transitoires. Le makespan est alors logiquement plus grand.

L'influence de l'horizon de production est plus subtile. En effet, l'horizon de travail contraint tous les autres facteurs (temps de cycle, en-cours du système, en-cours de chaque gamme). Ceci peut être expliqué par le fait que plus nous avons de gammes linéaires à ordonnancer simultanément plus nous contraignons le système et moins on a d'espoir d'atteindre la borne minimale d'en-cours.

IV.6 Existence d'un régime permanent

L'étude de l'exemple principal de ce mémoire, cf. §.IV.3, a abouti à une commande dont la durée du régime transitoire est du même ordre que celle du régime permanent. De plus, la durée du régime permanent est de 47 u.t. soit moins de deux cycles. Nous pouvons alors nous interroger sur l'existence effective d'un régime permanent dans cette commande voire même sur l'utilité de l'étude d'une commande cyclique si elle ne donne pas réellement lieu à une commande cyclique ?

En fait, pour répondre à cette question nous devons rappeler l'énoncé initial du problème à savoir une commande de 24 pièces (dont 9 A, 6 B et 9C) à optimiser. L'étude préliminaire nous a révélé qu'une production successive des trois types les uns à la suite des autres nécessite 100 u.t. à laquelle il faut ajouter la durée de quatre régimes transitoires (un pré-production, deux inter-productions et un post-production).

L'amélioration sensible de la performance justifie donc la production simultanée. De plus, comme nous l'avons rappelé au début de ce mémoire il n'existe pas à notre connaissance de méthode d'ordonnancement acyclique optimisant le makespan et l'en-cours et utilisant les flexibilités de gammes et de ressources. Enfin, la commande cyclique obtenue pour cet exemple est optimale pour le flux de production (temps de cycle minimisé) et pour l'en-cours utilisé.

En outre, nous avons mis en place, tout au long de la résolution, des bornes qui ont permis d'améliorer les performances de la solution finale (malgré le petit nombre de pièces

à produire) sans oublier la simplification considérable de la complexité du problème initial grâce à l'utilisation de l'hypothèse de répétitivité.

En tout état de cause, même si nous n'avons pas obtenu deux périodes de régime permanent, nous pouvons apprécier les apports de la commande cyclique d'une part du point de vue des performances obtenues et d'autre part du point de vue de la réduction de la complexité.

Nous allons maintenant établir une condition suffisante pour garantir une durée minimale du régime permanent. Soit d_0 cette durée minimale. En utilisant l'Equation IV-7 nous obtenons :

$$\text{Durée Permanent } (DDRP) = \min_i \left\{ \left(X - n_p(i, DDRP) \right) * CT + DDGL(i, DDRP) \right\} \geq d_0$$

$$\Leftrightarrow \quad \forall i \quad \left(X - n_p(i, DDRP) \right) * CT + DDGL(i, DDRP) \geq d_0$$

Comme DDGL peut prendre, dans le cas général, toutes les valeurs de [0, CT] alors :

$$\Rightarrow \quad \forall i \quad \left(X - n_p(i, DDRP) \right) * CT \geq d_0$$

$$\Rightarrow \quad \forall i \quad X - \frac{d_0}{CT} \geq n_p(i, DDRP)$$

$$\Rightarrow \quad X - \frac{d_0}{CT} \geq \max_i \left\{ n_p(i, DDRP) \right\}$$

$$\Rightarrow \quad X \geq n_{p\max}(DDRP) + \frac{d_0}{CT}$$

Nous allons fixer d_0 à 2*CT. En effet nous allons considérer qu'il existe un régime permanent cyclique dès que le cycle de fonctionnement se répète au minimum une fois. D'où la relation entre le nombre de répétitions de l'horizon de production et le nombre de palettes utilisées par le système :

$$X \geq n_{p\max}(DDRP) + 2$$

Equation IV-11 : Condition suffisante d'existence d'un régime cyclique

IV.7 Généralisation : Cas de plusieurs productions

Après avoir étudié le cas d'une seule production isolée, nous allons évoquer le problème d'enchaînement des productions. Nous allons donc étudier le problème des régimes transitoires inter-productions afin de dégager la problématique et d'identifier les nouveaux paramètres par rapport à une production isolée dans une perspective de résolution définitive de ce régime.

IV.7.1 Formulation

Le régime transitoire inter-productions est traité de façon spécifique car il se différencie des deux précédents par les problèmes qu'il pose, les variables en jeu et la méthode de résolution. En effet, ce régime contient des opérations appartenant à deux productions successives. De plus, contrairement aux deux précédents transitoires étudiés, ses points de départ et d'arrivée ne sont pas fixés a priori.

Parmi les nouveaux problèmes que le régime inter-productions fait apparaître, nous trouvons celui lié à la non production durant le pré-production. Cette hypothèse nous a permis de déterminer un certain nombre de bornes utiles pour optimiser le makespan ainsi que pour estimer un bon compromis entre l'optimisation du makespan et le temps de calcul. Il serait donc intéressant de pouvoir la conserver pour des productions non isolées. L'hypothèse devient donc la suivante ; elle exprime le fait qu'**aucune pièce appartenant à la nouvelle production ne puisse apparaître avant le début du nouveau régime permanent**.

De plus, le chevauchement de deux productions peut entraîner plusieurs cas de figures. En effet nous pouvons décider de différentes stratégies : finir entièrement une production avant d'en lancer une autre, lancer les pièces de la nouvelle production au fur et à mesure de la sortie des pièces de l'ancienne production, tout remettre à plat et de réordonnancer l'ensemble des opérations appartenant au transitoire inter-productions.

S'ajoutent à ces nouveaux problèmes de nouveaux paramètres et variables. En effet les cas de figures cités précédemment sont plus ou moins simples à étudier dans le cas où les deux productions successives utilisent le même nombre de palettes. Cependant si le nouveau régime utilise plus de palettes que le premier nous devons nous interroger sur le choix des dates d'introduction de ces palettes supplémentaires.

Dans le cas contraire, la même question s'impose : à quel moment retirer les palettes inutiles pour la nouvelle production ?. De plus, il est possible que les deux régimes successifs n'utilisent pas du tout le même type de palettes. Nous nous sommes donc limité, à ce niveau de notre étude, à constater la très grande complexité de l'optimisation des régimes transitoires inter-productions cycliques.

IV.7.2 Ordonnancement des régimes permanents entre eux

L'étude des régimes transitoires inter-productions nous renvoie vers une question, jusque là restée en suspend, concernant l'ordonnancement des régimes permanents entre eux, cf. I.4.2. Il est clair, en effet, que le choix d'un programme de production (ou encore d'une suite ordonnée de régimes permanents) contraint fortement la durée du régime transitoire puisqu'il conditionne les régimes inter-productions.

Le but de l'ordonnancement des régimes permanents entre eux est donc de minimiser le makespan en optimisant les durées des transitoires entre deux productions successives. Le problème réside dans le fait qu'on ne peut maîtriser cette durée qu'au moment où l'on dispose de la détermination des différents régimes permanents et des bornes du régime transitoire.

De plus, l'ordonnancement des productions entre elles est influencé par la méthode de résolution adoptée. En effet, si nous choisissons par exemple une stratégie d'étude du régime de post-production et de pré-production pour les juxtaposer (si la capacité du système de transport le permet) alors l'ordonnancement choisi doit minimiser la différence des durées du pré et post-production. De même, si nous choisissons d'approcher le pré et le post-production le plus possible alors le meilleur ordonnancement des productions entre elles est celui qui minimise la somme des durées du pré et du post production. Etc....

IV.7.3 Modélisation par Réseaux de Petri

Nous avons utilisé les Réseaux de Petri comme support de cette étude, nous proposons alors de modéliser l'ordonnancement final avec cet outil. Nous obtiendrons ainsi un modèle de référence de la commande. Il est clair que le système possède trois régimes de fonctionnement distincts (correspondant aux trois régimes pré-production, permanent et post-production). Par conséquent, nous pouvons considérer trois parties distinctes de RdP associées respectivement à ces régimes. De plus, nous devons éviter une représentation de

chaque opération à effectuer car un tel modèle serait trop grand et de plus ne serait pas fidèle au caractère cyclique du régime permanent. Ainsi, pour mieux comprendre le modèle que nous allons construire, nous allons recourir à la représentation par séquences de tir en rappelant qu'elle n'est pas unique. Nous représentons donc les trois régimes de la façon suivante :

$$\underbrace{t_{i_1}\text{-...-}t_{i_p}}_{\text{en pré-production}} \underbrace{-\left(t_{i_1}\text{-...-}t_{i_q}\right)^{\left(X-n_{pmax}(DDRP)\right)}\text{-}t_{i_1}\text{-...-}t_{i_r}}_{\text{régime permanent}} - \underbrace{t_{i_1}\text{-...-}t_{i_s}}_{\text{post-production}}$$

Equation IV-12 : Forme générale de la commande finale

Notons que le régime permanent est formé de deux parties : la première est cyclique se répétant $(X\text{-}n_{pmax}(DDRP))$ fois, cf. IV.2.2.4., quant à la deuxième, elle représente une fraction d'une période([1]) terminant le régime permanent. Cette représentation a l'avantage d'être indépendante du nombre X de répétitions de l'horizon de production néanmoins, elle reste tout de même fonction de la Date de Début du Régime Permanent.

En appliquant cette procédure à l'ordonnancement final de la Figure IV-10 (avec DDRP=0), nous obtenons le résultat suivant :

$$\underbrace{1.1\text{-}3.1\text{-}1.2}_{\text{en pré-production}} \underbrace{-(2.1\text{-}2.2\text{-}2.3\text{-}1.1\text{-}3.2\text{-}3.1\text{-}1.2)^{(X-1)}\text{-}2.1\text{-}2.2}_{\text{régime permanent}} - \underbrace{2.3\text{-}3.2}_{\text{post-production}}$$

Equation IV-13 : Séquence de tirs modélisant la commande totale en fonction de X

La séquence du régime pré-production contient les opérations à lancer durant ce régime. En conséquence, puisque les opérations 3.1 et 1.2 se terminent durant le régime permanent, nous considérons qu'elles sont lancées pour la première fois durant le régime permanent et à la fin de celui-ci.

Nous considérons ensuite la partie cyclique (représentée par la séquence de transitions sous la puissance $(X\text{-}n_{pmax}(0) = X\text{-}1)$ suivie de la partie non cyclique du régime permanent (opérations 2.1 et 2.2)). Le régime post-production est formé de deux opérations (2.3 et 3.2).

[1] Nous rappelons que la durée du régime permanent cyclique n'est pas obligatoirement multiple d'une période.

La traduction de la séquence de tir en un modèle Réseau de Petri, cf. Figure IV-20, se fait d'une manière naturelle :

- Pré-production : pour chaque gamme nous construisons une séquence linéaire d'opérations qui fournit un en-cours pour la gamme associé dans le graphe d'Evénements. Notons que pour la gamme G2 aucune opération n'est lancée en pré-production. Le séquencement sur les opérations des machines M1 et M2 est représentée en gris foncé pour lever l'indéterminisme en pré-production.

- Permanent cyclique : il est représenté par un Graphe d'Evénements (en gris clair), cf. Figure IV-3. Une place contenant un nombre de jetons égal à l'exposant de la séquence est synchronisée avec le début de chaque gamme pour que le nombre de répétitions soit égal au nombre nécessaire.

- Permanent acyclique : À chaque cycle, chaque gamme produit un jeton (une sorte d'historique) afin de lancer la suite du permanent cyclique une fois que nous avons répété X-1 fois la période. Nous enchaînons alors avec une séquence linéaire (vide pour G1).

- Post-production : c'est la suite et fin des séquences précédentes pour terminer la production. Comme en pré-production nous indiquons les contraintes de précédence sur les machines en gris foncé.

Le résultat final est un Réseau de Petri particulier : c'est un **RdP libre de choix** (*Choice-Free Petri net*)([1]), cf. [TER 97]. Ce graphe ne comporte plus d'indéterminisme. Il a été démontré que le déclenchement au plus tôt des transitions d'un Choice-Free est optimal par rapport au makespan, cf. [SIL 96a] et [CAM 98]. Il est donc directement transformable en une commande d'automates, chose possible avec un logiciel tel que *PETRI PARTNER*, cf. [BOU 91].

[1] Cette sous-classe de Réseaux de Petri est définie par $\forall p \in P \quad |p^{\bullet}| \leq 1$: c'est la notion la plus générale d'un graphe sans choix.

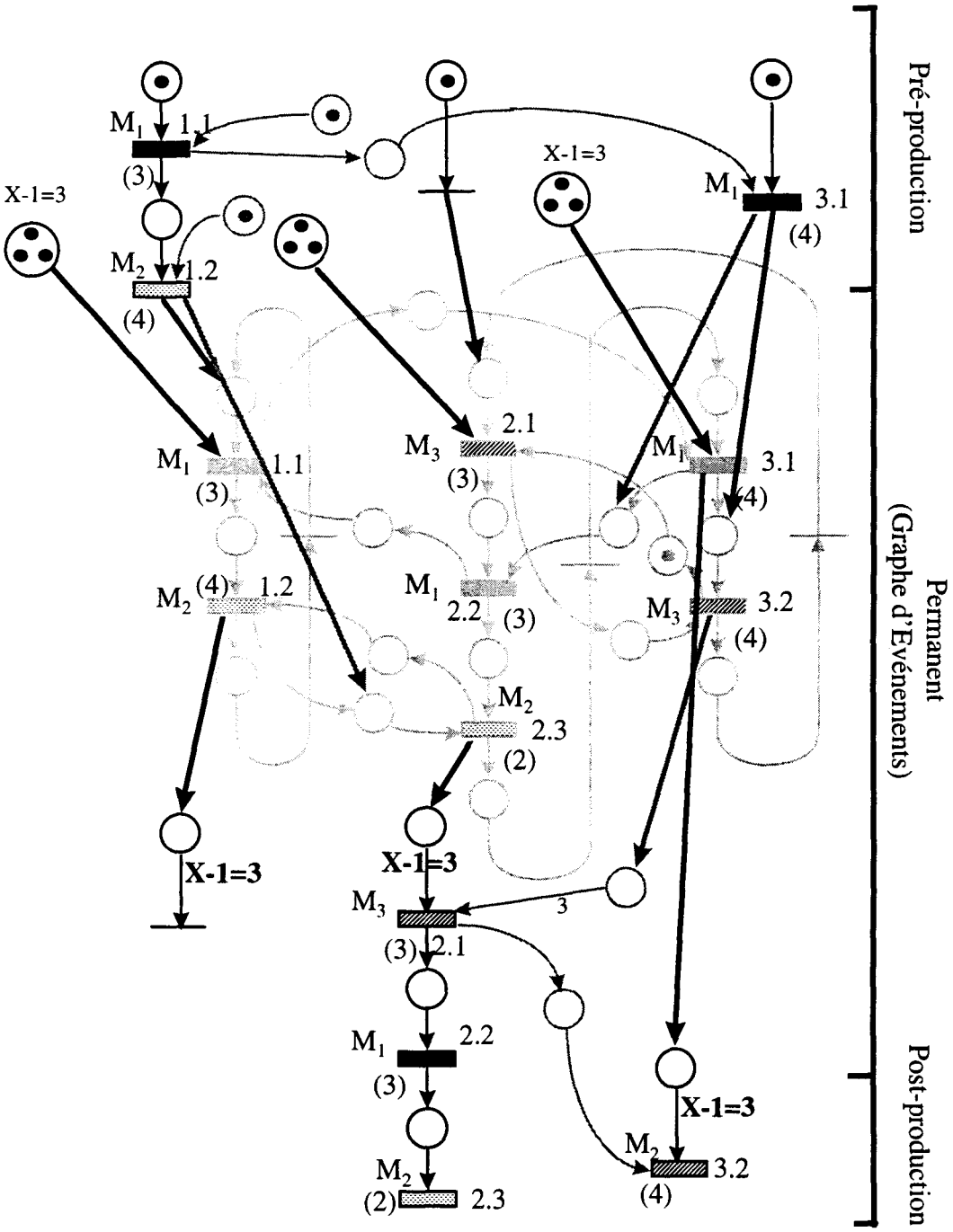


Figure IV-20 : Modélisation Réseau de Petri de la commande totale

IV.8 Conclusion

Nous retiendrons de cette étude que la détermination du régime transitoire est tout aussi combinatoire que celle du régime permanent. Cependant, nous avons pu émettre un certain nombre d'hypothèses restrictives qui ont permis, à la fois, de réduire la combinatoire et de garantir l'optimum du temps total de production, toujours moyennant le respect des ces hypothèses. Nous avons aussi proposé plusieurs bornes pour les durées des différents régimes afin d'assister la recherche du makespan optimal et de juger de la qualité des résultats obtenus.

L'heuristique que nous avons élaborée assure la minimisation du critère principal de performances en l'occurrence le temps total de production mais pénalise le critère secondaire de maximisation de la durée du régime permanent. Néanmoins, la difficulté majeure dans l'étude des régimes transitoires réside dans le fait qu'ils sont totalement dépendants du reste de l'étude. En effet, tout au long de la démarche d'évaluation de performances et d'ordonnancement, il est difficile de prévoir a priori si les délais de production pourront être respectés.

Nous avons représenté, cf. Figure IV-21, l'ensemble des bornes du makespan que nous avons défini en fin de chaque phase d'étude. Nous pouvons remarquer que si une borne inf. intéressante est accessible dès la fin de la phase 2, la seule borne max. significative du makespan est obtenue après la phase d'ordonnancement ce qui est relativement tard pour une remise en cause due à un éventuel dépassement de la date due.

Dans ce cas, deux solutions sont envisageables : la première consiste à constater que nous avons optimisé au mieux les performances du système (en tenant compte du compromis performances / combinatoire) en optimisant au maximum le transitoire. Quant à la seconde, elle réside dans la relance totale du calcul d'ordonnancement en relaxant la contrainte de la taille des horizons de production par exemple.

Dans ce sens, nous pensons, dans une perspective d'informatisation de la méthode, que la mise en place d'un système expert peut sembler intéressante. En effet, dans un atelier de production donné, le nombre I_{max} de types de pièces à réaliser est connu à l'avance, en dehors de la création de nouveaux types (dans ce cas nous pourrions y ajouter un à deux produits). Comme nous avons contraint la taille de l'horizon de production à dix pièces, il

est évident qu'au bout d'un certain temps d'exploitation nous retrouverons plus ou moins les mêmes productions déjà optimisées.

Nous pouvons alors constituer une base de données des productions déjà optimisées (caractérisées par l'horizon E, le temps de cycle minimal CT, une période du régime permanent, les régimes pré et post-production correspondant au min. du makespan) que l'on pourra enrichir au fur et à mesure du temps.

Ainsi, lors de la prise en charge d'une nouvelle production, l'étape de planification fine décomposera cette production en petites productions "cycliques". Sur un horizon déjà déterminé, nous utilisons le résultat stocké dans la base (le temps de cycle minimal CT, une période du régime permanent, les régimes pré et post-production correspondant au min. du makespan qui sont tous indépendants du nombre de répétitions X) sinon nous le calculons et le conservons pour la suite.

Dans le même sens, il deviendrait possible d'optimiser conjointement les régimes de post-production et de pré-production afin de proposer des solutions « contractées » de régime inter-productions pour les productions les plus courantes.

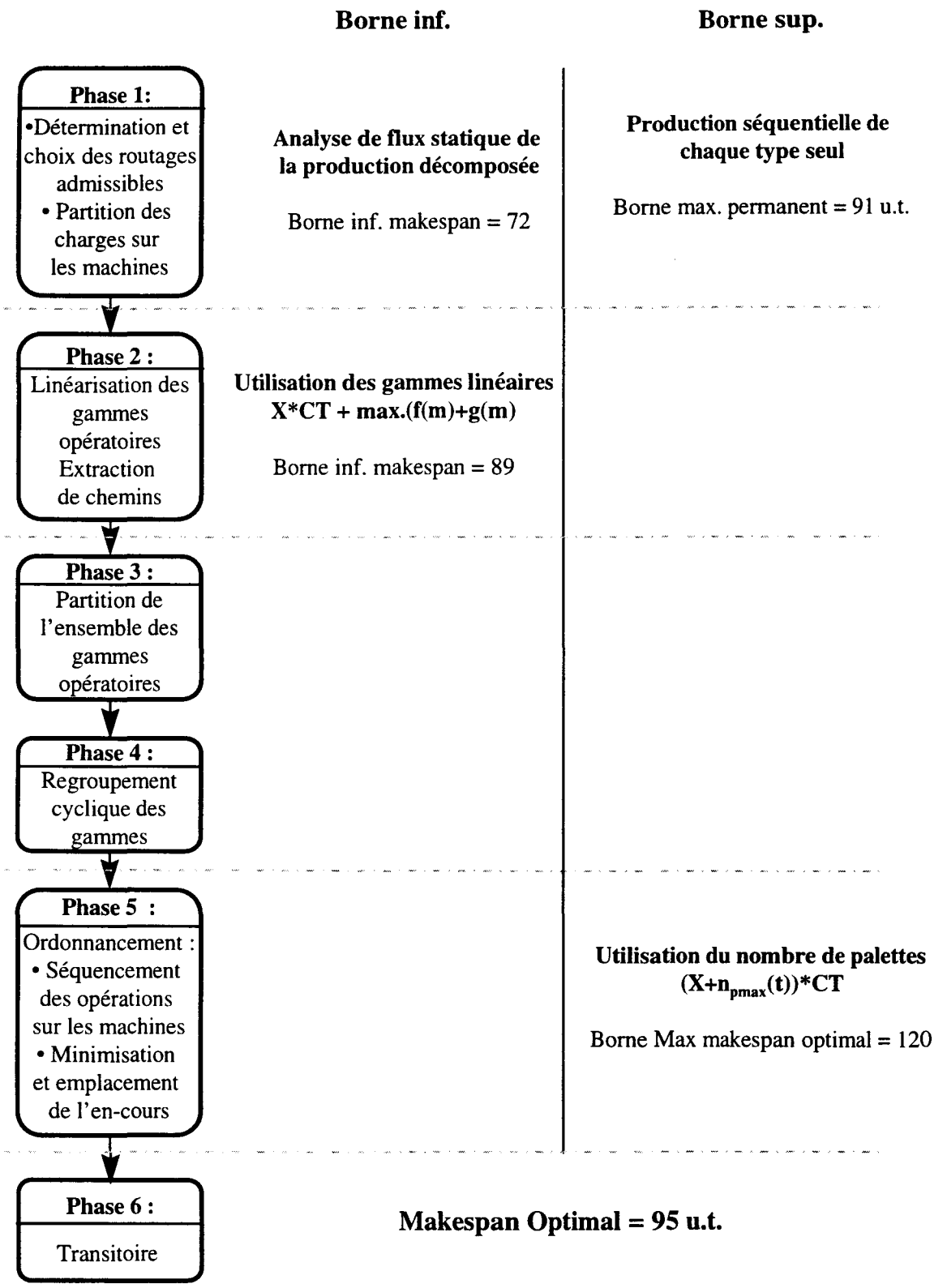


Figure IV-21 : Schéma récapitulatif de l'obtention des différentes bornes

Conclusion et Perspectives

Conduite de flux des Systèmes Flexibles de Production Manufacturière à l'aide d'une commande prévisionnelle

En conclusion, nous dresserons le bilan des apports de ce mémoire, puis, nous présenterons les perspectives et nouvelles directions de recherche que nous proposons concernant la détermination d'un **commande cyclique prévisionnelle** des **Systèmes Flexibles de Production Manufacturière** soumis à des demandes stables de **petites ou moyennes séries**. L'objectif de cette commande est d'optimiser un certain nombre de critères de performances en particulier le temps total de production et l'en-cours utilisé dans l'atelier de production.

Conclusion

Au cours de ce mémoire, nous nous sommes attaché à présenter les résultats de nos travaux, menés au sein de l'équipe Evaluation des Performances du LAIL, dans la suite logique des thèses de H. Ohl et H. Camus. C'est le raison pour laquelle nous avons adopté la démarche linéaire, progressive et structurée d'évaluation de performances et d'ordonnancement pour, tout d'abord, l'approfondir et par la suite en étendre les potentialités d'application. Cette linéarisation de la résolution a pour objet la recherche d'une réduction incontournable de complexité.

Ce travail réalisé avec un regard critique sur la hiérarchisation, antérieurement proposée, de résolution des différents indéterminismes nous a conduit à *justifier a posteriori cette linéarisation* et à *restructurer la démarche* linéaire de résolution, cf. **Chapitre I**. Cette démarche est constituée de 6 phases de résolution. Nous avons conservé entièrement la première étape de planification fine, proposée par H. Camus. Cette phase se charge de considérer une demande initiale quelconque pour la découper en une ou plusieurs commandes cycliques.

Les critères de performances sont donc découplés. Nous prenons alors en compte, dans cette phase, des critères tels que la minimisation des durées des régimes permanents, la simplicité de la commande, ... pour ne traiter l'optimisation de l'en-cours et le temps total de production que dans les phases ultérieures.

Au cours du **Chapitre II**, nous avons considéré la résolution d'une commande cyclique comportant les phases 2, 3 et 4 qui correspondent à la levée des indéterminismes dus aux flexibilités opératoires et aux ressources de transport. Nous avons alors effectué, pour chaque phase, une *étude de la combinatoire* et proposé une *représentation mathématique* qui a permis de *déterminer le nombre de solutions* attendues (ou au moins des bornes de ce nombre) ainsi qu'*un algorithme pour l'extraction de ces solutions* pour l'une de ces trois phases. Nous avons ainsi pu évaluer de façon plus précise les *gains de complexité* réalisés.

Dans le **Chapitre III**, nous avons traité la phase suivante de résolution à savoir l'ordonnancement cyclique proprement dit chargé d'optimiser l'en-cours du système tout en respectant le temps de cycle optimal (débit optimal). Nous avons alors présenté puis mis en œuvre *un nouvel algorithme d'ordonnancement cyclique*. L'application de cette heuristique sur un exemple illustratif de taille significative, puis sur des exemples bibliographiques, a permis de mettre en évidence la combinatoire de résolution, la complétude et l'efficacité de l'algorithme et plus généralement les apports originaux de l'approche de résolution proposée.

Après avoir passé en revue, à titre de comparaison, quelques exemples de référence, nous avons continué notre étude par l'extension de la phase d'ordonnancement aux opérations de transfert. En effet, jusque là, nous avions supposé que les opérations de transfert et de stockage entre les différentes opérations de transformation étaient de durées négligeables se déroulant en temps masqué ; Ceci dans le but d'isoler le problème d'*ordonnancement des opérations de transfert*, pour réduire une combinatoire déjà très complexe.

Disposant de résultats probants sur l'acheminement des pièces, nous avons pu étudier l'incidence des problèmes posés par les opérations de transfert (blocage par saturation, meilleur moment pour l'introduction des opérations de transfert ...). Nous avons alors poursuivi ce chapitre en proposant une *heuristique d'ordonnancement avec opérations de transfert* qui reste à implanter.

Le **Chapitre IV** concerne la question originale de l'*optimisation des régimes transitoires*. Cette étape est indispensable pour l'obtention d'une commande totale constituée de régimes permanents cycliques optimisés et de phases transitoires de démarrage, d'arrêt et éventuellement de passage d'un régime cyclique à un autre.

Nous avons ainsi étudié et résolu le problème de lancement et d'arrêt de régimes cycliques isolés en proposant différentes bornes de calcul des régimes transitoires et en présentant une *heuristique de détermination de la commande totale minimisant le temps total de production*. Les régimes transitoires entre deux régimes cycliques ont été analysés afin de dégager les principales difficultés. Cette question reste cependant à résoudre.

Perspectives

Les perspectives des travaux de recherches que nous avons mené peuvent être décomposées en trois classes.

En premier lieu, les perspectives à court terme concernent la mise en oeuvre des différentes phases sous forme d'applications informatiques. En effet, il reste, dans ce domaine à terminer la formalisation mathématique plus fine des phases 2 et 3 en vue de la détermination d'algorithmes d'extraction des différentes solutions comme nous l'avons proposé pour la phase 4. De même, l'heuristique d'ordonnancement avec opérations de transfert reste à valider par une implantation et une application sur différents jeux d'essais comme nous l'avons effectué pour l'heuristique d'ordonnancement sans transfert. Il conviendrait également de réaliser une étude mathématique de la complexité de l'algorithme d'ordonnancement, afin de mieux maîtriser la combinatoire de la résolution.

Les travaux que nous menons actuellement sont dans la logique de ces perspectives et concernent l'implantation de la détermination des régimes de Pré-production et de Post-production, en collaboration avec l'équipe *Modélisation et spécification* du LAIL. L'algorithme que nous cherchons à appliquer est donc basé sur une recherche d'accessibilité dans les Réseaux de Petri à l'aide de la programmation par contraintes, cf. [JAF 94], [BEN 95] et [VAR 96], qui permet de couper au plus tôt les solutions inutiles et donc d'accélérer la résolution, mais en même temps on ne veut pas qu'on « tue » trop vite une bonne solution. Ainsi, l'application, réalisée sous VISUAL C++ et ILOG SOLVER, permet de trouver les paquets de tir (STEPS) des séquences d'accessibilité entre un marquage initial et un marquage final.

Dans la cas du régime Pré-production, par exemple, l'état initial est composé de pièces brutes, l'état final est caractérisé par la date de début du régime permanent. L'avantage d'une telle méthode, par rapport à un ordonnancement au plus tôt classique, est qu'elle est basée sur un modèle RdP et donc qu'elle permet de prendre en compte les flexibilités opératoires. De plus, par la propagation des contraintes, l'arbre de recherche est largement plus réduit que dans le cas d'une méthode énumérative.

L'informatisation totale de la démarche (regroupement de tous les modules dans une seule application) reste donc une priorité, elle est indispensable en vue d'une mise en oeuvre industrielle de cette approche.

En second lieu, il serait intéressant, compte tenu des possibilités d'applications industrielles, d'étendre la démarche aux problèmes d'assemblage. Dans ce cas, la difficulté résidera dans le fait que l'en-cours n'est pas constant dans le système (durant l'assemblage, deux pièces sont assemblées pour n'en faire qu'une !).

Une troisième et dernière piste concerne la conception des architectures de transport. En effet, si le système de transport est mal adapté à la production envisagée alors les opérations de transfert deviennent critiques et pénalisantes. Par conséquent, il est nécessaire d'intégrer une phase complémentaire de conception et de dimensionnement du système de transport. Cette étude est d'autant plus difficile que les productions sont variables en raison de la nature même des Systèmes Flexibles de Production Manufacturière.

Bibliographie

Références bibliographiques

[AHM 96] AHMAD S. B., MOALLA M. ET COURVOISIER M.
Approche multimodèle pour la commande des ateliers flexibles, Journal Européen des Systèmes Automatisés, Vol. 30, n°9, pp.1201-1232, 1996.

[ALA 82] ALAIWAN H. et MEMMI G.
Algorithmes de recherche des solutions entières positives d'un système linéaire d'équations homogènes, Revue technique Thomson-CSF, 1982, Vol. 14, n° 2, pp.125-135.

[ALA 89] ALAIWAN H. et TOUDIC J.M.
Recherche des semi-flots, des verrous et des trappes dans les Réseaux de Petri, Technique et Science Informatiques, 1989, Vol. 4, n° 1, pp.103-112.

[AMA 92] AMAR S., CRAYE E. et GENTINA J.-C.
A method of hierarchical specification and prototyping of FMS, IEEE-ETFA 92, Melbourne, Australie, 1992.

[AMA 94] AMAR S.
Systèmes automatisés et flexibles de production manufacturière: méthode de conception du système de coordination par prototypage orienté objet de la partie procédé, Thèse de Doctorat en Productique, Automatique et Informatique Industrielle, Université de Lille 1, 1994.

[ARB 90] ARBIB, C., LUCERTINI, M. et NICOLO F.
Workload balance and part-transfer minimization in flexible manufacturing systems, The International Journal of Flexible Manufacturing Systems, Vol. 3, pp.5-25, 1990.

[AUS 94] AUSFELDER C.
Contribution à la conception d'un système de conduite pour les systèmes flexibles de production manufacturière : modélisation et validation de la commande, Thèse de Doctorat, Université de Lille 1, 1994.

[BAR 92] BARBIER F. et JAULENT P.
Les techniques orientées objet et CIM, Editions Eyrolles, 1992.

[BEL 65] BELLMANN R.
Dynamic Programming, Princeton University Press, Princeton, New Jersey, 1965.

[BEL 82] BELLMANN R., ESOGBUE A.O. et NABESHIMA I
Mathematical Aspects of Scheduling and Applications, Pergamon Press, 1982.

[BEL 88] BEL E., BENSANA D., et DUBOIS D.
Construction d'ordonnancements prévisionnels : un compromis entre approches classiques et systèmes experts, RAIRO, APII, 1988, Vol. 22, N°5, pp. 509-534.

[BEN 93] BENZANKEN V. Et DOUCET A.
Bases de données orientées objet : origines et principes, Armand Colin, 1993.

[BEN 95] BENASSER A., BON Ph., LEFORT A. Et YIM P.
Un Algorithme d'Accessibilité pour les réseaux de Petri : Application aux SED, Journée d'étude Affectation et Ordonnancement, Tours, 1995, pp.165-187.

[BER 96] BERRUET P. et LY F.
Terminologie et Approche fonctionnelle de la Supervision, Note Interne NI / 96 / 3, Laboratoire d'Automatique et d'Informatique industrielle de Lille, septembre 1996.

[BIG 97] BIGAN M., CORBEEL D. et BOUREY J.P.
Integration of view points by using an object oriented approach, 9th Symposium on Information Control in Manufacturing, INCOM'98, Metz, France, juin 1998.

[BIL 93] BILLAUT J.-C. et ROUBELLAT F.
Significant States and Decision Making for Real Time Workshop Scheduling, Proceedings of the 7th Annual European Computer Conference on Computers in Design, Manufacturing and Production, Paris-Evry, France, mai 1993, pp.493-500.

[BIL 95] BILLAUT J.-C., ROUBELLAT F. et VILLAUMIE M.
Prise en compte de l'affectation dans le logiciel d'ordonnancement Temps Réel Ordo, Journées d'Etude du GdR Automatique, Affectation et Ordonnancement, Tours, France, septembre 1995, pp.113-131.

[BIL 98] BILLAUT J.-C., T'KINDT V., RICHARD P. et PROUST C.
Three exact methods and an effcient heuristic for solving a bicriteria flowshop scheduling problem, CESA'98 : IMACS Multiconference on Computational Engineering in Systems Applications, Nabeul-Hammamet, Tunisie, Avril 1998. pp.371-377.

[BOI 91] BOIS S.
Intégration de la gestion des modes de marche dans le pilotage d'un système automatisé de production, Thèse de Doctorat, Université de Lille 1, 1991.

[BOU 93a] BOUREY J.-P.
Méthode de conception de la commande de systèmes flexibles de production manufacturière, Habilitation à Diriger des Recherches, Université de Lille 1, 1993.

[BOU 93b] BOUHCHOUCH A., FREIN Y. et DALLERY Y.
Evaluation de performances de lignes de production bouclées par une méthode analytique, APII, Vol. 27, n° 3, pp.315-342.

[BOU 91] BOURCERIE M., GODON A., BERRUÉ J., DELANCHY B. et FERRIER J.-L.
PETRI PARTNER : les réseaux de Petri, de la simulation à la commande, sur IBM PC et compatibles, Revue d'automatique et de productique appliquée, Vol. 4, n° 3/1991, pp.233-241.

[BOU 96] BOUHCHOUCH A., FREIN Y. ET DALLERY Y.
Performance evaluation of closed tandem queueing networks with finite buffers, Performance Evaluation, Vol. 26, pp.115-132, 1996.

[CAM 90] CAMPOS J.
Performance bounds for synchronized queueing networks, Ph.D. thesis, Departamento de Informática e Ingeniería de Sistemas, Universidad de Zaragoza, Espagne, décembre 1990.

[CAM 93] CAMUS H.
Optimisation du routage des pièces dans un atelier flexible par la méthode du simplexe, mémoire de DEA de Productique, Université de Lille 1, juin 1993.

[CAM 96a] CAMUS H., OHL H., KORBAA O. et GENTINA J.-C.
Cyclic schedules in Flexible Manufacturing Systems with groups of identical machines, Rensselaer's Fifth International Conference on Computer Integrated Manufacturing and Automation Technology, CIMAT'96, Grenoble, France, mai 1996, Vol. 1, pp.345-350.

[CAM 96b] CAMUS H., OHL H., KORBAA O. et GENTINA J.-C.
Cyclic schedules in Flexible Manufacturing Systems with flexibilities in operating sequences, First International Workshop on Manufacturing and Petri Nets, 17th International conference on Application and Theory of Petri Nets, Osaka, Japon, juin 1996, pp.97-116.

[CAM 96c] CAMUS H., OHL H., KORBAA O. et GENTINA, J.-C.
Petri Net Modeling of flexible Operating Sequences in a F.M.S. and Implications for the search of cyclic schedules, Symposium on Discrete Events and Manufacturing Systems, CESA'96 IMACS Multiconference on Computational Engineering in Systems Applications, Lille, France, juillet 1996, pp.218-225.

[CAM 97] CAMUS H.
Conduite de Systèmes Flexibles de Production Manufacturière par composition de régimes permanents cycliques : modélisation et évaluation de performances à l'aide des Réseaux de Petri, Thèse de Doctorat, Université de Lille 1, mars 1997.

[CAM 98] CAMUS H., MUGARZA J.C., TERUEL E., GENTINA J.C. et SILVA M.
Scheduling Petri Nets Models and Structural Analysis, CESA'98 : IMACS Multiconference on Computational Engineering in Systems Applications, Nabeul-Hammamet, Tunisie, avril 1998

[CAR 88] CARLIER J. et CHRETIENNE P.
Problèmes d'ordonnancement: modélisation / complexité / algorithmes, Editions Masson, Paris, France, 1988.

[CAR 96] CARLIER J., PERIDY L. et PINSON E.
Recent developments on the Job-shop scheduling problem, Rensselaer's Fifth International Conference on Computer Integrated Manufacturing and Automation Technology, CIMAT'96, Grenoble, France, mai 1996, Vol. 1, pp.249-254.

[CAS 96] CASTELAIN E.
Contribution à la conduite des systèmes flexibles de production manufacturière, Habilitation à diriger des recherches, Université de Lille 1, juillet 1996.

[CHR 85] CHRETIENNE P.
Analyse des régimes transitoire et asymptotique d'un graphe d'événements temporisé, Technique et Science Informatiques, 1985, Vol. 4, n° 1, pp.127-142.

[CHR 97] CHRETIENNE P., COFFMAN E. G., LENSTRA J. K. et LIU Z.
Scheduling Theory and its applications, Wiley editorials, 1997.

[COM 71] COMMONER F., HOLT A.W., EVEN S. et PNUELI A.
Marked directed graphs, Journal of Computer and System Sciences, 1971, Vol. 5, n° 5, pp.511-523.

[COM 74] COMTET L.
Advanced Combinatorics : The Art of Finite and Infinite Expansions, D. Reidel, 1974.

[CRA 94] CRAYE E.
Contribution au contrôle / commande de Systèmes Flexibles de Production Manufacturière, Habilitation à diriger des recherches, Université de Lille 1, décembre 1994.

[CRU 91] CRUETTE D.
Méthodologie de conception des systèmes complexes à événements discrets : application à la conception et à la validation de la commande de cellules flexibles de production dans l'industrie manufacturière, Thèse de Doctorat, Université de Lille 1, 1991.

[DAV 92] DAVID R. et ALLA H.
Du grafcet aux réseaux de Petri, Hermes, 1992.

[DEA 93] DE ARAUJO S.L., BOUHCHOUCH A., DI MASCOLO M. et FREIN Y.
On the Analysis of a Stochastic Petri Net Modeling a Ressource Sharing Situation, IEEE Conference on Systems, Man et Cybernetics, Octobre 1993,Le Touquet, France, Vol. 1, pp.301-306.

[DEL 96] DELAVAL M. et LEFORT A.
Car sequencing problem two approaches : Hyper nets and simulated annealing, CIMAT'96, Grenoble, France, mai 1996, pp.174-179.

[DIC 93] DI CESARE F., HARHALAKIS G., PROTH J.-M., SILVA M. et VERNADAT F.B.
Practice of Petri Nets in manufacturing, Chapman et Hall, London, 1993.

[DIM 92] DI MASCOLO M., FREIN Y. et DALLERY Y.
Queueing Network Modeling and Analysis of Kanban Systems, IEEE, pp.202-211.

[DUB 90] DUBOIS D. et STECKE K.E.
Dynamic analysis of repetitive decision-free discrete-event processes : applications to production systems, Annals of Operations Reseach, 1990, Vol. 26, pp.323-347.

[ELK 93] ELKHATTABI S.
Intégration de la surveillance de bas niveau dans la conception des systèmes à événements discrets: application aux systèmes de production flexibles, Thèse de Doctorat, Université de Lille 1, 1993.

[EOW 95] EOWYN (LABesançon, LAILille, LITours)
Intégration des différents niveaux de commande dans le pilotage d'un système de production : de la planification prédictive au pilotage réactif et à la supervision, Projet MESR DSPT8 1995 - 1996.

[ERS 76] ERSCHLER J., ROUBELLAT F. et VERNHES J.P.
A decision making process for the real time control of process unit, International Journal of Production Research, 1976, Vol. 14, n° 2.

[ERS 82] ERSCHLER J., LEVEQUE D. et ROUBELLAT F.
Periodic loading of Flexible Manufacturing Systems, IFIP Congress, APMS, Bordeaux, France, 1982, pp.327-339.

[FDI 89] FDIDA S. et PUJOLLE G.
Modèles de systèmes et de réseaux, Performances et Files d'Attentes, Editions Eyrolles, Paris, France, 1989.

[FLO 85] FLORIN G. et NATKIN S.
Les Réseaux de Petri stochastiques, Technique et Science Informatiques, 1985, Vol. 4, n° 1, pp.143-159.

[FRE 88] FREIN Y., DALLERY Y., PIERRAT J.-J. et DAVID R.
Optimisation du routage des pièces dans un atelier flexible par des méthodes analytiques, APII, 1988, Vol. 2, n° 5, pp.489-508.

[GAM 94] GAMACHE A.
Bases de données, Architectures Modèles Langages, Les presses de l'Université de Laval, 1994.

[GEN 88] GENTINA J.-C., BOUREY J.-P. et KAPUSTA M.
Colored adaptive structured Petri Nets - a tool for the automatic synthesis of hierarchical control of Flexible Manufacturing Systems, Computer Integrated Manufacturing Systems, Butterworth Heinemann, February 1988, Vol. 1, n° 1, pp.39-47.

[GEN 97] J.-C. GENTINA, E. CASTELAIN, O. KORBAA, H. CAMUS, A. LEFORT et M. DELAVAL
Intégration des différents niveaux de commande dans le pilotage d'un système de production : de la planification prédictive au pilotage réactif à la supervision, Note Interne NI / 97 / 1,, LAIL, Lille, France, 1997.

[GOL 86] GOLTZ, U.
Synchronic distance, Advances in Petri Nets, LNCS n° 254, 1986, pp.338-358.

[GOT 93] GOTHA
Les problèmes d'ordonnancement, Recherche opérationnelle, 1993, Vol. 27, n° 1, pp.77-150.

[GIU 96] GIUA A.
Petri Nets techniques for Supervisory Control of Discrete Event Systems, First International Workshop on Manufacturing and Petri Nets, 17th International conference on Application and Theory of Petr Nets, Osaka, Japon, juin 1996, pp.1-30.

[GUP 88] GUPTA J.N.P.
Two stage hybrid flowshop scheduling problem, Operations Research, 1988, Vol 39, n°4, pp.359-364.

[HAM 91] HAMMADI S.
Une méthode d'ordonnancement minimisant les temps d'attente et de transit dans les systèmes de production flexible de type job-shop, Thèse de Doctorat, Université de Lille 1, 1991.

[HAN 89] HANEN C.
Optimizing microprograms for recurrent loops on pipelined architectures using timed Petri Nets, in Advances in Petri Nets, LNCS 424, 1989, Springer Verlag, New York, U.S.A., pp.236-261.

[HAN 94] HANEN C.
Problèmes d'ordonnancement cycliques, Habilitation à diriger des recherches, LIPT, Institut Blaise Pascal, Université VI, Université VI, Février 1995.

[HAP 95] HAPPIETTE M. et ZENG X.
Ordonnancement pour le problème flow-shop, RAIRO, APII, 1995, Vol. 29, n 6, pp.623-639.

[HIL 87] HILLION H.P., PROTH J.-M. et XIE X.L.
A heuristic algorithm for the periodic scheduling and sequencing job-shop problem, 26th IEEE Conference on Decision and Control, Los Angeles, U.S.A., 1987, pp.612-617.

[HIL 88] HILLION H.P. et PROTH J.-M.
Analyse de fabrications non linéaires et répétitives à l'aide des Graphes d'Evénements Temporisés, RAIRO, Vol 22, n° 2, septembre 1988.

[HIL 89a] HILLION H.P. et PROTH J.-M.
Performance evaluation of job-shop systems using timed event-graphs, IEEE Transactions on Automatic Control, 1989, Vol. 34, n° 1, pp.3-9.

[HIL 89b] HILLION H.
Modélisation et analyse des systèmes de production discrets par les Réseaux de Petri temporisés, Thèse de Doctorat en Mathématiques, Université Pierre et Marie Curie, Paris VI, 1989.

[HUV 94] HUVENOIT B.
De la conception à l'implantation de la commande modulaire et hiérarchisée de systèmes flexibles de production manufacturière, Thèse de Doctorat en Productique, Université de Lille 1, 1994.

[JAF 94] JAFFAR J. Et MAHER M.J.
Constraint Logic Programming : A sruvey, Journal Logic Programming, 1994, Vol 19/20, pp.503-581.

[JOH 54] JOHNSON S.M.
Optimal two and three stage production schedules with set-up times included, Naval Research Logistics Quarterly, 1954, Vol 28, n°1, pp.61-68.

[IVA 93] IVAN J.
Le génie logiciel orienté objet, ACM press, 1993.

[KAP 88] KAPUSTA M.
Génération assistée d'un graphe fonctionnel destiné à l'élaboration structurée du modèle de la partie commande pour les cellules de production flexibles dans l'industrie manufacturière, Thèse de Doctorat, Université de Lille 1, 1988.

[KAU 68] KAUFMANN A.
Introduction à la combinatorique en vue de ses applications, Dunod, Paris France, 1968.

[KER 96] KERMAD L.
Contribution à la supervision et à la gestion des modes et des configurations des Systèmes Flexibles de Production Manufacturière, Thèse de Doctorat, Université de Lille 1, janvier 1996.

[KHA 96] KHANSA W., DENAT J.-P. et COLLART DUTILLEUL S.
Analysis of Robustness using the periodic functioning of timed Event Graphs, Rensselaer's Fifth International Conference on Computer Integrated Manufacturing and Automation Technology, CIMAT'96, Grenoble, France, mai 1996, Vol. 1, pp.180-185.

[KIM 83] KIMEMIA J. et GERSHWIN S.B.
An algorithm for the computer control of a Flexible Manufacturing System, IIE Transactions, Vol. 15, n° 4, 1983, pp.353-362.

[KOR 95] KORBAA O.
Recherche des circuits élémentaires dans un flow-shop, mémoire de DEA de Productique, Université de Lille 1, juin 1995.

[KOR 97a] KORBAA O., CAMUS H. et GENTINA J.-C.
FMS Cyclic Scheduling with Overlapping production cycles, Second International Workshop on Manufacturing and Petri Nets, 18th Int Conf. on Application and Theory of Petri Nets, ICATPN 97, Toulouse, France, pp.35-53.

[KOR 97b] KORBAA O., CAMUS H. et GENTINA J.-C.
Transient State Study for Cyclic Schedules : Bounds and Optimization, IEEE International Symposium on Assembly and Task Planning, ISATP'97, Marina del Rey, Californie, USA, Aout 1997, pp.188-193.

[KOR 97c] KORBAA O., CAMUS H. et GENTINA J.-C.
Heuristic for the Resolution of the General FMS Cyclic Scheduling Problem, IEEE International Conference on Systems Man and Cybernetics, Orlando, USA, Octobre 1997. Vol. n°3. pp. 2903-2908.

[KOR 98] KORBAA O., CAMUS H. et GENTINA J.-C.
Heuristic for the Computation of Transient States for Cyclic Schedules, CESA'98 : IMACS Multiconference on Computational Engineering in Systems Applications, Nabeul-Hammamet, Tunisie, Avril 1998, pp.563-568.

[KRO 97] KROL JOZAGA P.
Heuristique de recherche d'un ordonnancement optimal, mémoire de DEA de Productique, Université de Lille 1, septembre 1997.

[LAF 91] LAFTIT S.
Graphes d'événements déterministes et stochastiques: application aux systèmes de production, Thèse de Doctorat en Mathématiques, Paris IX Dauphine, 1991.

[LAF 92] LAFTIT S., PROTH J.-M. et XIE X.L.
Optimization of invariant criteria for event graphs, IEEE Transactions on Automatic Control, Vol 37, n° 1, pp.547-555.

[LAF 93] LAFTIT S., PROTH J.-M. et XIE X.L.
Marking Optimization in Timed Event Graphs, Lecture Notes in Computer Science, 1993, Vol 674.

[LEE 94] LEE D.Y. et DICESARE F.
Scheduling flexible manufacturing systems using Petri nets and Heuristic search, IEEE Transactions on Robotics and Automation, Vol. 10, n° 2, 1994, pp.123-132.

[LOP 92] LOPEZ P., ERSCHLER J. et ESQUIROL P.
Ordonnancement de tâches sous contraintes : une approche énergétique, R.A.I.R.O. APII, 1992, Vol. 26, n° 5-6, pp.453-481.

[MAN 96] MANIER H. et BAPTISTE P.
The Influence of the Lay-out on the Performances of a Few Methods of Control of a Flexible Manufacturing Ring, Symposium on Discrete Events and Manufacturing Systems, CESA'96 IMACS Multiconference on Computational Engineering in Systems Applications, Lille, France, juillet 1996, pp.570-574.

[MAR 82] MARTINEZ J. et SILVA M.
A simple and fast algorithm to obtain all invariants of a generalized Petri Net, in Application and Theory of Petri nets, Eds. C. Girault et. W. Reisig, Springer Verlag, New York, 1982, pp.301-310.

[MAU 95] MAURINO M.
La gestion des données techniques, Technologies du concurrent engineering, Masson, 1995.

[MES 97] MESGHOUNI K., HAMMADI S. et BORNE P.
Extension for Job-shop scheduling, IEEE International Conference on Systems Man and Cybernetics, Orlando, USA, Octobre 1997. Vol. n°1. pp. 720-725.

[MOU 97] MOUZE A.
Mise en oeuvre de la complexité à l'ordonnancement cyclique, mémoire de fin d'études, Ecole Centrale de Lille, septembre 1997.

[MUN 91] MUNIER A.
Résolution d'un problème d'ordonnancement cyclique à itérations indépendantes et contraintes de ressources, RAIRO, Recherche Opérationnelle, 1991, Vol. 25, n° 2, pp.161-182.

[MUR 89] MURATA T.
Petri Nets: properties, analysis and application, Proceedings of the IEEE, 1989, Vol. 77, n° 4, pp.541-580.

[NDI 97] NDIAYE D.
Système de gestion des données techniques d'un système productique, mémoire de DEA de Productique, Université de Lille 1, Septembre 1997.

[OHL 94] OHL H., CAMUS H., CASTELAIN E. et GENTINA J.-C.
A heuristic algorithm for the computation of cyclic schedules and the necessary WIP to obtain optimal cycle time, Rensselaer's Fourth International Conference on Computer Integrated Manufacturing and Automation Technology, CIMAT 1994, octobre 1994, Troy, New York, U.S.A., Vol. 1, pp.339-344.

[OHL 95a] OHL H.
Fonctionnements répétitifs de systèmes flexibles de production manufacturière : Analyse et Optimisation des performances à l'aide des Réseaux de Petri, Thèse de Doctorat, Université de Lille 1, septembre 1995.

[OHL 95b] OHL H., CAMUS H., CASTELAIN E. et GENTINA J.-C.
Petri Net modeling of ratio-driven Flexible Manufacturing Systems and implications on the WIP for cyclic schedules, IEEE Conference on Systems, Man and Cybernetics, octobre 1995, Vancouver, British Columbia, Canada, Vol. 4, n° 5, pp.3081-3086.

[PLU 94] PLUS O.
Minimisation du nombre de marques dans les graphes d'événements, mémoire de DEA de Productique, Université de Lille 1, septembre 1994.

[PRO 94] PROTH J.-M. et XIE X.L.
Les Réseaux de Petri pour la conception et la gestion des systèmes de production, Editions MASSON, 1994.

[PRO 95] PROTH J.-M. et MINIS I.
Production management in a Petri net environment, RAIRO/ Recherche opérationnelle, Vol. 29, n° 3, 1995, pp.321-352.

[PRO 96] PROTH J.-M. et SAUER N.
Continuous approach of scheduling problem based on Petri nets, Proceedings 5th IEEE Int. Conf. On Emerging Technologies and Factory Automation (ETFA'96), Hawaii, 1996, pp.717-723.

[RAM 80] RAMAMOORTHY C.V. et HO G.S.
Performance evaluation of asynchronous concurrent systems using Petri Nets, IEEE Transactions on Software Engineering, 1980, Vol. SE-6, n° 5, pp.440-449.

[ROU 94] ROUBELLAT F., BILLAUT J.-C. et VILLAUMIE M.
Ordonnancement d'atelier en temps réel : d'ORABAID à ORDO, Journées d'Etude du GT3, Ordonnancement et Entreprise, Applications Concrètes et Outils pour le Futur, Toulouse, France, juin 1994.

[SER 89] SERAFINI P. et UKOVICH W.
A mathematical model for periodic scheduling problems, SIAM J. Disc. Math, Novembre 1989,Vol 2, n° 4, pp.550-581.

[SHE 92] SHEN L., CHEN Q. et LUH. L.Y.S.
Truncation of Petri nets models for simplifying computation of optimum scheduling problems, Computer in Industry, 1992, pp. 25-43.

[SIF 80] SIFAKIS J.
Performance evaluation of system using Nets, in Net theory and Applications, LNCS, Springer Verlag, Berlin, 1980, pp.307-319.

[SIL 87] SILVA M.
Towards a synchrony theory for p/t nets, in Concurrency and Nets, Eds K. Voss, Springer Verlag, Berlin, 1987, pp.435-460.

[SIL 89] SILVA M. et VALETTE R.
Petri Nets and Flexible Manufacturing, in Advances in Petri Nets, LNCS 424, Springer Verlag, New York, 1989, pp.374-417.

[SIL 96a] SILVA M. et TERUEL E.
Petri Nets for the Design and Operation of Manufacturing Systems, Rensselaer's Fifth International Conference on Computer Integrated Manufacturing and Automation Technology, CIMAT'96, Grenoble, France, mai 1996, Vol. 1, pp.330-343.

[SIL 96b] SILVA M. et TERUEL E.
A Systems Theory Perspective of Discrete Event Dynamic Systems : The Petri Net Paradigm, Symposium on Discrete Events and Manufacturing Systems, CESA'96 IMACS Multiconference on Computational Engineering in Systems Applications, Lille, France, juillet 1996, pp.1-12.

[STE 91] STECKE K.E. et KIM I.
A flexible approach to part type selection in flexible flow systems using part mix ratios, Int. J. Prod. Res., 1991, Vol. 29, n° 1, pp.53-75.

[STE 92] STECKE K.E.
Procedures to determine part mix ratios for independent demands in Flexible Manufacturing Systems, IEEE Transactions on Engineering Management, 1992, Vol. 39, n° 4, pp.359-369.

[TAW 95] TAWEGOUM R.
Contrôle temps réel de déroulement des opérations dans les systèmes de production flexibles, Thèse de Doctorat, Université de Lille I - Ecole Centrale de Lille, avril 1995.

[TER 97] TERUEL E., COLOM J. M. et SILVA M.
Choice-Free Petri Nets : A Model for deterministic concurrent Systems with Bulk Services and Arrivals, IEEE Trans. On System, Man and Cybernetics, Vol. 27, n 1, janvier 1997.

[TOG 92] TOGUYENI A.
Surveillance et diagnostic en ligne dans les ateliers flexibles de l'industrie manufacturière, Thèse de Doctorat, Université de Lille 1, 1992.

[TOU 82] TOUDIC J.M.
Algorithmes d'algèbre linéaire pour l'analyse structurelle des Réseaux de Petri, Revue technique Thomson-CSF, 1982, Vol. 14, n° 1, pp.137-155.

[VAL 94] VALENTIN C.
Modeling and analysis methods for a class of hybrid dynamic systems, Symposium Automatisation des processus mixtes : Les systèmes dynamiques hybrides, ADPM'94, Bruxelles, Belgique, novembre 1994, pp.221-226.

[VAR 96] VARNIER C.
Extensions de « Hoist Scheduling Problem » cyclique : Résolution basée sur un traitement des contraintes disjonctives en Programmation Logique avec Contraintes, Thèse de Doctorat, Université de Franche-Comté, n° 502, janvier 1996.

[XIE 89] XIE X.L.
Real time scheduling and routing for Flexible Manufacturing Systems with unreliable machines, RAIRO, Recherche Opérationnelle, 1989, Vol. 23, n° 4, pp.355-374.

[ZAY 95] ZAYTOON J., MOUGHAMIR S., VILLERMAN-LECOLIER G.
Extension de la méthode OMT au génie automatique, Génie Logiciel, N°17, septembre 1995

Notations et Abréviations

1-cyclique — Les machines répètent à chaque cycle exactement le même fonctionnement (mêmes opérations aux mêmes dates)

Buffer — File d'attente (du type : premier arrivé, premier servi) qui sert au stockage des pièces en attendant la libération de la machine ou à la sortie de la machine

Card(E) — Cardinal de l'ensemble E = nombre d'éléments de l'ensemble

CASPAIM — Conception Assistée des Systèmes de Production Automatisés en Industrie Manufacturière

Cellule — (*de transport*) ressource unitaire de transport (capacité = une palette)

Charge — *d'une machine* : somme des temps opératoires des opérations réalisées par la machine

CT — Cycle Time = *fr.* Temps de Cycle (optimal)

DDGL(i,DDRP) — Date de **Début** de la Gamme Linéaire i pour une date de début du régime permanent DDRP.

DDRP : — Date de Début du Régime Permanent

DFGL(i,DDRP) — Date de **Fin** de la Gamme Linéaire i pour une date de début du régime permanent

Flexibilité — (*de gamme*) non unicité du procédé de fabrication : possibilité de produire le type de pièces avec au moins deux gammes linéaires différentes

FRT-net — **F**ree **R**elated **T**-semiflow net : *fr.* Réseau à T-semiflot librement reliés

Gamme — Ensemble des chemins possibles pour fabriquer un type de produit

Gamme linéaire	séquence linéaire d'opérations (avec ordre stricte) permettant de réaliser un produit
Gantt dual	Diagramme de Gantt représentant le comportement de l'en-cours auquel on associe un autre diagramme représentant le comportement des machines
Horizon	(*de production, cyclique, E*) *:* Nombre de pièces à produire par cycle durant le régime permanent
Hors ligne	*prévisionnel, déterministe* : les temps de calcul ne permettent pas de prendre en compte l'évolution instantanée du système
Inter-productions	Régime transitoire qui permet d'enchaîner deux régimes permanents successifs
LAIL	Laboratoire d'Automatique et d'Informatique industrielle de Lille
Machine menante	(*critique, goulot*) : machine dont la charge est supérieure ou égale à toutes les autres machines
Macro-gamme	Séquence ordonnée de gammes = Ensemble de gammes linéaires qui sont transportées par les mêmes palettes avec un ordre « cyclique » défini de passage sur ces palettes
Makespan	Temps total de production
MRP	Manufacturing Resource Planning
NP-complet	Non Polynomial complet
NP-difficile	Non Polynomial difficile
Ordonnancement	Séquencement des opérations sur les machines et détermination du nombre d'en-cours ainsi que son emplacement
Permanent	régime cyclique, obtenu par l'heuristique d'ordonnancement, respectant le temps de cycle optimal
PFM	Production Flexible Manufacturière
P.G.C.D.	Plus Grand Commun Diviseur

Planification fine Etape qui prend en compte les caractéristiques du système (machines disponibles et intervalles de temps de production) et ses contraintes (demande initiale, ...) pour établir des régimes permanents (*horizon de production*, *CT*, *X*)

P.P.C.M. Plus Petit Commun Multiple

Pré-production Régime transitoire qui permet d'atteindre le régime permanent à partir d'un système vide

Production isolée Production qui n'est pas précédée ni succédée par une autre (départ = arrivée = système vide)

Post-production Régime transitoire qui permet de terminer un régime permanent pour aboutir à un système vide

Ressource (de transformation) : Ensemble d'une ou plusieurs machines identiques capables d'effectuer les mêmes opérations avec les mêmes temps opératoires

RdP Réseau de Petri

RFA Réseaux de Files d'Attente

RP Régime Permanent

SED Systèmes à Evénements Discrets

SFPM Système Flexible pour Production Manufacturière

Transitoire isolé Transitoire ne correspondant à aucun régime permanent. Sert à des très petites production (unitaires) afin de respecter les contraintes de production initiales

Tronçon (*de transport*) : partie de l'anneau de transport, formée d'une ou plusieurs cellules et gérée en FIFO, ne possédant qu'une entrée (en amont) et une sortie (en aval). Capacité = nombre de cellules.

X Nombre de répétitions de l'horizon de production

UML Unified Modeling Language

u.t. unité de temps

Index

Index Alphabétique

Annexe I

Les classes de décision correspondent aux différentes flexibilités du problèmes :

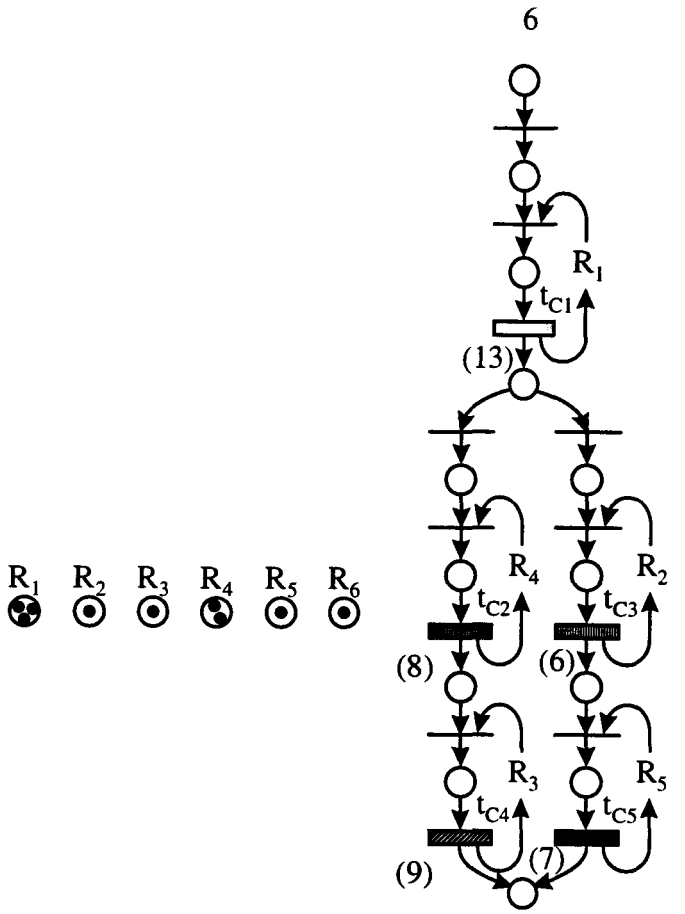
- **Classe D_1 :** *Choix des produits à fabriquer simultanément (Planification fine)* : cette flexibilité s'intéresse à la question : **Quels produits** fabriquer simultanément durant une production ?
- **Classe D_2 :** *Choix des ratios de production* : cette classe pose le problème de **combien de pièces** de chaque type doit-on fabriquer durant un cycle de fonctionnement ?
- **Classe D_3 :** *Résolution des flexibilités de gamme* : **quels procédés** de fabrication utiliser pour chaque pièce ?
- **Classe D_4 :** *Affectation d'une opération à une machine particulière* : cette classe s'intéresse à la question **quelles machines** utiliser pour chaque procédé de fabrication (dans les où certaines machines existent en plusieurs exemplaires) ?
- **Classe D_5 :** *Ordonnancement* : une fois que nous savons quelles machines utiliser, nous nous intéressons à **quand et comment utiliser ces machines** (dates et ordre de réalisation des opérations) ?
- **Classe D_6 :** *En-cours* : cette classe pose le problème de **quelles palettes utiliser** ? **combien** ? **pour quelles pièces et dans quel ordre** ?
- **Classe D_7 :** *Transitoire* : **comment lancer et arrêter la production** ?

Annexe II

II.1 Calcul des bornes pour l'exemple illustratif

• Borne max. du makespan optimal (hors régimes transitoires)

Elle correspond à une fabrication de toutes les pièces du type A suivies du type B puis C., cf. § I.4.2. Le calcul se fait comme si on produisait toutes les pièces d'un même type en un cycle. Pour le **type C**, nous avons à fabriquer 6 pièces avec les contraintes suivantes :

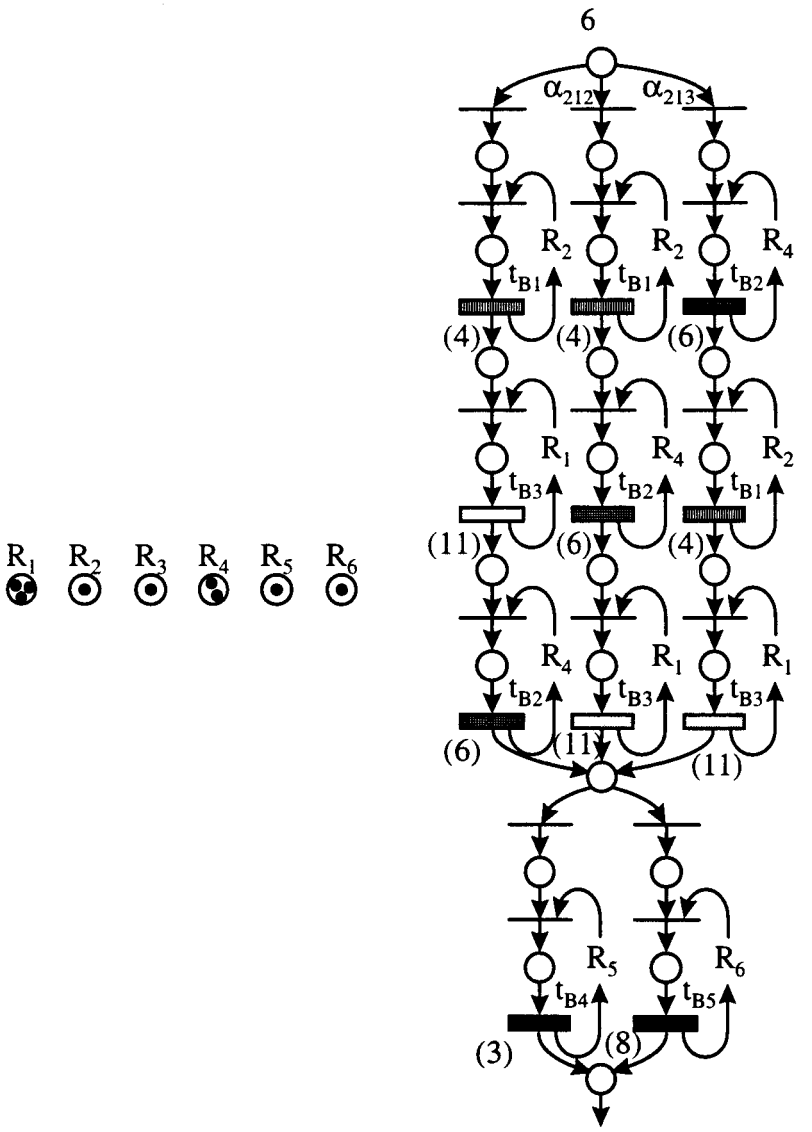


Il suffit alors d'appliquer un calcul de flux directement en discret :

$$
\begin{cases}
0 \le n \le 6 \\
R_1 \ : \ Z = (13*2)*3 = 26*3 \ \text{(26 u.t. par machine)} \\
R_2 \ : \ Z = 6*n \ \text{(n est le nombre de pièces dans cette branche)} \\
R_3 \ : \ Z = 9*(6-n) \\
R_4 \ : \ Z = 8*(6-n) \ \text{(pour deux machines)} \\
R_5 \ : \ Z = 7*n \\
R_6 \ : \ Z = 0
\end{cases}
$$

Une étude rapide du système (une inconnue) donne un temps de cycle optimal de 27 u.t., pour n = 3, et la machine menante est R3.

Pour le **type B**, nous avons à fabriquer 6 pièces avec les contraintes suivantes :

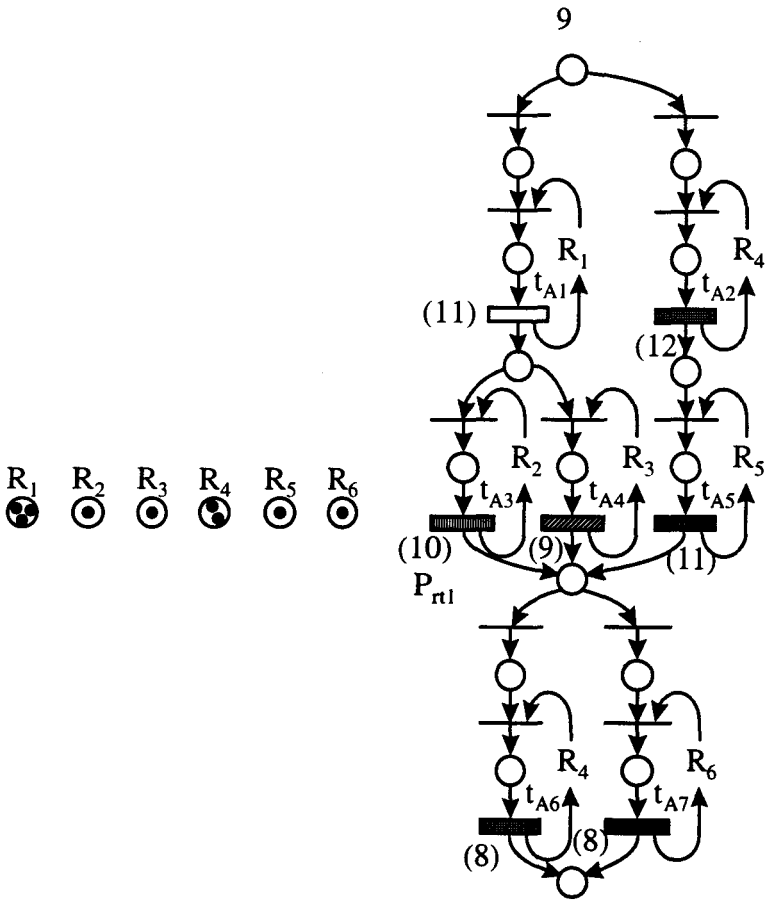


Le calcul du flux en discret donne alors :

$$
\begin{array}{l}
0 \le n \le 6 \\
R_1 \ : \ Z = (11*2)*3 = 22*3 \ \text{(22 u.t. par machine)} \\
R_2 \ : \ Z = 4*6 = \ 24 \\
R_3 \ : \ Z = 0 \\
R_4 \ : \ Z = (6*3)*2 = \ 18*2 \ \text{(18 u.t. par machine)} \\
R_5 \ : \ Z = 3*n \\
R_6 \ : \ Z = 8*(6-n)
\end{array}
$$

Dans ce cas la solution est triviale : la ressource critique est R2 et le temps de cycle associé est égal à 24 u.t.

Finalement, concernant le **type A**, nous avons à fabriquer 9 pièces avec les contraintes suivantes :



$$
\begin{cases}
0 \le n_1 \le 9 \\
0 \le n_2 \le 9 \\
0 \le n_{1.1} \le n_1 \\
R_1 \ : \ Z = 11 * n_1 \quad \text{(pour 3 machines)} \\
R_2 \ : \ Z = 10 * n_{1.1} \\
R_3 \ : \ Z = 10 * (n_1 - n_{1.1}) \\
R_4 \ : \ Z = 12 * (9 - n_1) + 8 * (9 - n_2) \quad \text{(pour 2 machines)} \\
R_5 \ : \ Z = 11 * (9 - n_1) \\
R_6 \ : \ Z = 8 * (9 - n_2)
\end{cases}
$$

Le calcul est bien entendu plus compliqué que les deux cas précédents étant donné la présence de 3 variables. Le résultat optimal, correspond à $n_1 = 6$, $n_{1.1} = 2$ et $n_2 = 4$. D'où :

$$
\begin{cases}
R_1 \ : \ Z = (11 * 2) * 3 = 22 * 3 \quad \text{(22 u.t. par machine)} \\
R_2 \ : \ Z = 10 * 2 = 20 \\
R_3 \ : \ Z = 10 * 4 = 40 \\
R_4 \ : \ Z = 12 * 3 + 8 * 5 = 76 \text{ u.t. (40 u.t. pour une machine et 36 pour l'autre)} \\
R_5 \ : \ Z = 11 * 3 = 33 \\
R_6 \ : \ Z = 8 * 5 = 40
\end{cases}
$$

Par conséquent :

$$\text{Borne max. du makespan optimal} = \underbrace{40}_{\substack{\text{production}\\ \text{du type A}}} + \underbrace{24}_{\substack{\text{production}\\ \text{du type B}}} + \underbrace{27}_{\substack{\text{production}\\ \text{du type C}}} = 91 \text{ u.t.}$$

Equation 1 : Borne max. du makespan optimal hors régimes transitoires

•La borne inf. théorique du makespan

Concernant la borne inf. théorique, elle est déterminée par le calcul de flux, cf. [FRE 88], d'une production de **21 pièces** dont $\frac{3}{7}$ de type A, $\frac{2}{7}$ de type B et autant de type C avec une **prise en compte du caractère discret**, c'est-à-dire que nous considérons les ratios $\frac{3}{7}$, $\frac{2}{7}$ et $\frac{2}{7}$ mais ne conservons que les horizons admissibles à savoir qu'un horizon de 12 A, 8 B et 8 C, quoique compatible avec les ratios, n'est pas admissible puisqu'il dépasse la demande initiale de 9 A, 6 B et 6 C.

Cette prise en compte de la production discrète nous impose un facteur multiplicatif ν, de l'horizon minimal de production {3, 2, 2}, inférieur à trois. En effet, comme nous produisons en tout 3*{3, 2, 2}, trouver le flux optimal pour un facteur multiplicatif de 4 par exemple n'est pas réaliste puisque nous ne pourrons même pas finir un seul cycle. Les calculs, cf. [CAM 97], nous donnent un temps de cycle, pour le facteur multiplicatif ν=3, égal à 69 u.t.

Borne inf. théorique du makespan = 69 u.t.

II.2 Démonstrations des Lemmes du Chapitre II

Lemme II.1 :

Le nombre de solutions du système : $\begin{cases} \forall i \quad x_i \in IN \\ \sum_{i=1}^{m} x_i = p \end{cases}$ où m et p sont deux entiers naturels non nuls est égal à : $T(m,p)=C_{m+p-1}^{m-1} = C_{m+p-1}^{p} = \frac{(m+p-1)!}{(p)!*(m-1)!}$.

Démonstration[1] :

D'abord, afin de mieux comprendre le lemme nous allons l'appliquer avec m=2 et p=3.

L'équation $x_1+x_2 = 3$ admet les solutions suivantes : (0,3), (1,2), (2,1) et (3,0).

Ensuite, pour démontrer ce lemme, nous allons partitionner $T(m,p)$ en deux sous-ensembles selon la valeur du premier élément x_1 :

- cas où $x_1 = 0$: l'équation s'écrit alors sous la forme $\sum_{i=2}^{m} x_i = p$ et le nombre de solutions est égal à $T(m-1,p)$.
- cas où $x_1 \neq 0$: on définit alors $x'_1=x_1-1$ et l'équation s'écrit sous la forme $\sum_{i=1}^{m} x_i - 1 = (x_1 - 1) + \sum_{i=2}^{m} x_i = x'_1 + \sum_{i=2}^{m} x_i = p - 1$ et le nombre de solutions est égal à $T(m,p-1)$.

Comme les deux sont disjoints et indépendants, on en déduit que $T(m,p) = T(m,p-1) + T(m-1,p)$ et le nombre de solutions est totalement défini par cette équation récurrente à laquelle on ajoute les conditions initiales : $T(m,0) = 1$ et $T(1,p) = 1$. En effet, le nombre de solutions de l'équation $\sum_{i=1}^{m} x_i = 0$ est égal à 1 et consiste à mettre tous les x_i égaux à zéro. De même, l'équation : $x_1 = 1$ admet comme ensemble de solution le singleton {1}. Le résultat final se démontre facilement par récurrence en utilisant le formule précédemment établie ■

Proposition II.1 :

la borne supérieure du *nombre total de solutions T* est donnée par :

$$T \leq \min\left(\frac{\prod_{j=1}^{m} T1(n,\beta_j)}{\max_j (T1(n,\beta_j))}, \frac{\prod_{i=1}^{n} T2(m,\lambda_i)}{\max_i (T2(m,\lambda_i))} \right)$$

[1] D'autres démonstrations, différentes de celle que nous présentons, sont possibles, cf. [COM 74] et [MOU 97].

Lemme II.2 :

Le nombre de partitions possibles dans un ensemble de N éléments identiques est égal à p(N,N). Ce nombre est une ***borne inférieure*** *du cas général de partitions dans un ensemble de N éléments quelconques.*

Démonstration :

La première affirmation est facilement prouvée en posant a(i) = l(i) (pour les a(i)≠0) et en utilisant l'Equation II-5). Quant à la seconde, elle résulte du fait que dans le cas général, outre la détermination du nombre de partitions possibles, affecter un élément à un ensemble plutôt qu'à un autre peut donner une nouvelle combinaison. Par conséquent, pour chaque partition, il faudra déterminer toutes les façons d'affecter les différents éléments aux différents ensembles ■

Lemme II.3 :

avec la définition précédente de p(N,N), on obtient :

$$p(1,q)=1,\ p(N,1)=1$$
et
$$p(N,q)=p(N,q-1)+p(N-q,q)$$

Démonstration :

Soit a(i) une **suite croissante** finie de q entiers naturels vérifiant : $\sum_{i=1}^{q} a(i) = N$. Nous allons calculer le nombre p(N,q) de solutions possibles de construction de a(i). On distingue **deux cas disjoints** :

- a(1) = 0 : alors le nombre de solutions est égal à p(N,q-1).
- a(1) ≠ 0 : alors tous les a(i) sont différents de 0 (puisque a(i) est croissante). On peut donc soustraire une unité à chaque élément de la somme ce qui nous donne $\sum_{i=1}^{q}(a(i)-1) = \sum_{i=1}^{q} a'(i) = N-q$ et le nombre de solutions est de p(N-q,q).

Les deux cas étant disjoints, le nombre total de partitions d'un entier naturel N en q entiers positifs est donc égal à la somme des deux cas.

Proposition II.2 :

Le nombre C(N,q) de structures différentes dans une partition de N éléments en q ensembles est donné par la forme suivante :

$$C(q,q)=1\ ,\ C(N,1)=1$$
et
$$C(N,q)=C(N-q,q)+p(N-q,q-1)$$

Démonstration :

On utilise le même principe que pour p(N,q). On considère une suite a(i) **croissante** finie de q éléments entiers **strictement** positifs telle que $\sum_{i=1}^{q} a(i) = N$. Nous procédons à la soustraction de 1 à chaque élément de la somme ($\sum_{i=1}^{q}(a(i)-1)=\sum_{i=1}^{q} a'(i)=N-q$) pour obtenir l'un des deux cas disjoints :

- $a'(1) \neq 0$: le nombre de solutions est alors égal à C=(N-q,q)
- $a'(1) = 0$: comme a'(2) peut être également nul ainsi que a'(3) ... le problème s'écrit : $\sum_{i=2}^{q} a'(i)=\sum_{j=1}^{q-1} a''(j)=N-q$ où $a''(j)$ est **une suite croissante finie d'entiers positifs ou nuls**. Le nombre de solutions est donc égal à p(N-q,q-1).

Etant donné l'intersection vide entre les deux cas, le résultat final est obtenu par la réunion des deux ensembles ■

Lemme II.4 :

$$\text{Si } \text{pgcd}(\delta_1, \ldots, \delta_n)=1, \text{ alors } \xi=\frac{\left(\sum_{i=1}^{n}\delta_i\right)!}{\prod_{i=1}^{n}(\delta_i)!}*\frac{1}{\left(\sum_{i=1}^{n}\delta_i\right)}.$$

Démonstration :

1. Si toutes les gammes étaient distinctes, le nombre de regroupements ordonnés serait égal à $\left(\sum_{i=1}^{n}\delta_i\right)!$, ce qui correspond à évaluer le nombre d'arrangements de n éléments pris dans un ensemble à n éléments : soit $A_n^n = n!$.

2. Puisqu'il existe δ_l gammes G_l identiques, il faut retirer toutes les permutations possibles entre ces éléments. Il faut donc diviser par $(\delta_l)!$. La prise en compte de tous les types de gammes nous amène donc à diviser par $\prod_{i=1}^{n}(\delta_i)!$, pour ne garder que les regroupements ordonnés distincts.

Considérons la relation d'équivalence $\mathcal{R}$: « est équivalent à une permutation cyclique près ». Les solutions appartenant à la même classe d'équivalence donnent le même regroupement cyclique. Comme $\text{pgcd}(\delta_1,...,\delta_n)=1$, il ne peut pas y avoir de permutation interne, c'est-à-dire des permutations d'ordre inférieur à $\sum_{i=1}^{n}\delta_i$.

Par conséquent, le nombre de classes d'équivalences ou de macro-gammes obtenues à partir du regroupement $\mathfrak{C} = \left\{\overbrace{G1,...G1}^{\delta_1 \text{ éléments}},\overbrace{G2,...G2}^{\delta_2 \text{ éléments}},...,\overbrace{Gn,...,Gn}^{\delta_n \text{ éléments}}\right\}$ est égal à :

$$\xi = \frac{\left(\sum_{i=1}^{n}\delta_i\right)!}{\prod_{i=1}^{n}(\delta_i)!} * \frac{1}{\left(\sum_{i=1}^{n}\delta_i\right)} \blacksquare$$

Lemme II.5 :

Si $\text{pgcd}(\delta_1, ..., \delta_n)=\delta$ avec δ **nombre premier**, alors $\xi = \xi_\delta + \xi_1$, avec ξ_δ le nombre de classes d'équivalences des permutations cycliques d'ordre $\frac{\sum_{i=1}^{n}\delta_i}{\delta}$ exactement et ξ_1 celui des permutations cycliques d'ordre $\left(\sum_{i=1}^{n}\delta_i\right)$ exactement.

$$\xi_\delta = \frac{\left(\sum_{i=1}^{n}\frac{\delta_i}{\delta}\right)!}{\prod_{i=1}^{n}\left(\frac{\delta_i}{\delta}\right)!} * \frac{1}{\sum_{i=1}^{n}\frac{\delta_i}{\delta}} \quad et \quad \xi_1 = \left(\frac{\left(\sum_{i=1}^{n}\delta_i\right)!}{\prod_{i=1}^{n}(\delta_i)!} - \frac{\left(\sum_{i=1}^{n}\frac{\delta_i}{\delta}\right)!}{\prod_{i=1}^{n}\left(\frac{\delta_i}{\delta}\right)!}\right) * \frac{1}{\sum_{i=1}^{n}\delta_i}.$$

Démonstration :

1. Contrairement au cas étudié précédemment, il existe ici des classes d'équivalence dont la cardinalité est inférieure à $\sum_{i=1}^{n}\delta_i$. En effet celles-ci sont liées à des permutations cycliques d'ordre inférieur à $\sum_{i=1}^{n}\delta_i$. Comme ici $\text{pgcd}(\delta_1,...,\delta_n)=\delta$ avec δ premier, les seules autres permutations cycliques à étudier sont celles d'ordre $\frac{\sum_{i=1}^{n}\delta_i}{\delta}$. La cardinalité de ces classes d'équivalences est donc égale à $\frac{\sum_{i=1}^{n}\delta_i}{\delta}$. Ces classes viennent de celles trouvées sur l'ensemble $\mathfrak{E}|_\delta = \left\{\underbrace{G_1, ..., G_1}_{\delta_1/\delta \text{ fois}}, \underbrace{G_2, ..., G_2}_{\delta_2/\delta \text{ fois}}, ..., \underbrace{G_n, ..., G_n}_{\delta_n/\delta \text{ fois}}\right\}$. Comme δ est premier, nous sommes sûrs de ne pas trouver dans les regroupements ordonnés $M|_\delta$ de permutations internes, c'est-à-dire des classes d'équivalence des permutations cycliques d'ordre inférieur à $\frac{\sum_{i=1}^{n}\delta_i}{\delta}$. Le nombre de classes d'équivalences est donc donné par le lemme 4. D'où $\xi_\delta = \frac{\left(\sum_{i=1}^{n}\frac{\delta_i}{\delta}\right)!}{\prod_{i=1}^{n}\left(\frac{\delta_i}{\delta}\right)!} * \frac{1}{\sum_{i=1}^{n}\frac{\delta_i}{\delta}}$.

2. Nous recherchons maintenant les classes d'équivalence des permutations cycliques d'ordre $\sum_{i=1}^{n}\delta_i$ exactement. Pour cela, il faut retirer de l'ensemble des regroupements ordonnés ceux considérés précédemment, c'est-à-dire ceux des

classes d'équivalence des permutations cycliques d'ordre égal à $\frac{\sum_{i=1}^{n}\delta_i}{\delta}$. La cardinalité de ces ensembles est donc $\frac{\left(\sum_{i=1}^{n}\frac{\delta_i}{\delta}\right)!}{\prod_{i=1}^{n}\left(\frac{\delta_i}{\delta}\right)!}$. L'ensemble des regroupements ordonnés a pour cardinalité $\frac{\left(\sum_{i=1}^{n}\frac{\delta_i}{\delta}\right)!}{\prod_{i=1}^{n}\left(\frac{\delta_i}{\delta}\right)!}$. Donc quand nous ôtons les regroupements ordonnés considérés précédemment, il reste $\frac{\left(\sum_{i=1}^{n}\delta_i\right)!}{\prod_{i=1}^{n}(\delta_i)!}-\frac{\left(\sum_{i=1}^{n}\frac{\delta_i}{\delta}\right)!}{\prod_{i=1}^{n}\left(\frac{\delta_i}{\delta}\right)!}$ regroupements ordonnés appartenant à des classes d'équivalence de cardinalité égale exactement à $\sum_{i=1}^{n}\delta_i$. D'où le nombre de classes d'équivalences des permutations est donc égal à

$$\xi_1=\left(\frac{\left(\sum_{i=1}^{n}\delta_i\right)!}{\prod_{i=1}^{n}(\delta_i)!}-\frac{\left(\sum_{i=1}^{n}\frac{\delta_i}{\delta}\right)!}{\prod_{i=1}^{n}\left(\frac{\delta_i}{\delta}\right)!}\right)*\frac{1}{\sum_{i=1}^{n}\delta_i}. \blacksquare$$

Proposition II.3 :

Si $\text{pgcd}(\delta_1, ..., \delta_n)=\delta$ et si l'ensemble des diviseurs de δ est $\{d_{\delta 1}=\delta, ..., d_{\delta k}=1\}$, alors le nombre total de macro-gammes obtenues à partir de $\mathfrak{E}$ est donné par $\xi=\sum_{i=1}^{k}\xi_{d_{\delta i}}$, où $\xi_{d_{\delta i}}$ représente le nombre de classes d'équivalences d'ordre $\frac{\sum_{k=1}^{n}\delta_k}{d_{\delta i}}$ et vaut :

$$\xi_{d_{\delta i}}=\left(\begin{array}{l}\text{nombre de permutations ordonnés pour } \mathfrak{E}\big|_{d_{\delta i}} \\ -\text{ nombre de permutations ordonnés pour } \mathfrak{E}\big|_{d_{\delta j}} \text{ pour tous les } d_{\delta j} \text{ multiples de } d_{\delta i}\end{array}\right)*\frac{1}{\sum_{k=1}^{n}\frac{\delta_k}{d_{\delta i}}}$$

Lemme II.6 :

Soient $\mathcal{X}$ et $\mathcal{Y}$ deux regroupements cycliques, ils sont équivalents pour la relation d'équivalence $\mathcal{R}$ définie si et seulement si :

il existe un entier naturel $i \in \{0,...,N-1\}$ tel que $\mathcal{Y}$ est l'image de $\mathcal{X}$ par $\mathcal{F}^i$:

$$\mathcal{X} \mathcal{R} \mathcal{Y} \Leftrightarrow \exists\, i \in \{0,...,N-1\} \text{ tel que } \mathcal{F}^i.\mathcal{X}=\mathcal{Y}$$

Démonstration :

Evidente : $\Rightarrow$ si $\mathcal{X} \mathcal{R} \mathcal{Y}$ alors $\mathcal{Y}$ est une permutation cyclique de $\mathcal{X}$

$\Leftarrow \mathcal{F}^i.\mathcal{X}=\mathcal{Y}$ alors $\mathcal{Y}$ est une permutation cyclique de $\mathcal{X}$ et donc $\mathcal{X} \mathcal{R} \mathcal{Y}$ ■

Par conséquent la famille de regroupements équivalents à $\mathcal{X}$ (appelée classe d'équivalence de $\mathcal{X}$) est obtenue par : $\{\mathcal{X}, \mathcal{F}.\mathcal{X}, \mathcal{F}^2.\mathcal{X}, ..., \mathcal{F}^{N-1}.\mathcal{X}\}$.

Notons que cette famille peut avoir des éléments identiques si $\mathcal{X}$ contient des composantes identiques. Dans ce cas on obtient : $\exists i < N$ tel que $\mathcal{F}^i.\mathcal{X} = \mathcal{X}$.

Proposition II.4 :

$\mathcal{X}$ et $\mathcal{Y}$ étant deux regroupements cycliques.

$$\mathcal{X} \text{ et } \mathcal{Y} \text{ sont non équivalents} \Leftrightarrow \forall\, i \in \{0,...,N-1\} \quad \mathcal{F}^i.\mathcal{X} \neq \mathcal{Y}$$

Démonstration :

Se démontre facilement avec la proposition précédente :

$\Rightarrow \mathcal{X}$ non équivalent à $\mathcal{Y}$ supposons que $\exists\, i \in \{0,...,N-1\}$ tel que $\mathcal{F}^i.\mathcal{X}= \mathcal{Y}$

alors d'après la première proposition $\mathcal{X} \mathcal{R} \mathcal{Y}$ ce qui n'est pas le cas.

$\Leftarrow \mathcal{F}^i.\mathcal{X} \neq \mathcal{Y}$ supposons que $\mathcal{X} \mathcal{R} \mathcal{Y}$

alors d'après la première proposition $\exists\, i \in \{0,...,N-1\}$ tel que $\mathcal{F}^i.\mathcal{X}=\mathcal{Y}$

ce qui n'est pas le cas ■

II.3 Détail du nombre des solutions obtenues

Dans cette partie nous allons détailler le nombre de solutions obtenues à chaque phase.

II.3.1 Linéarisation des gammes

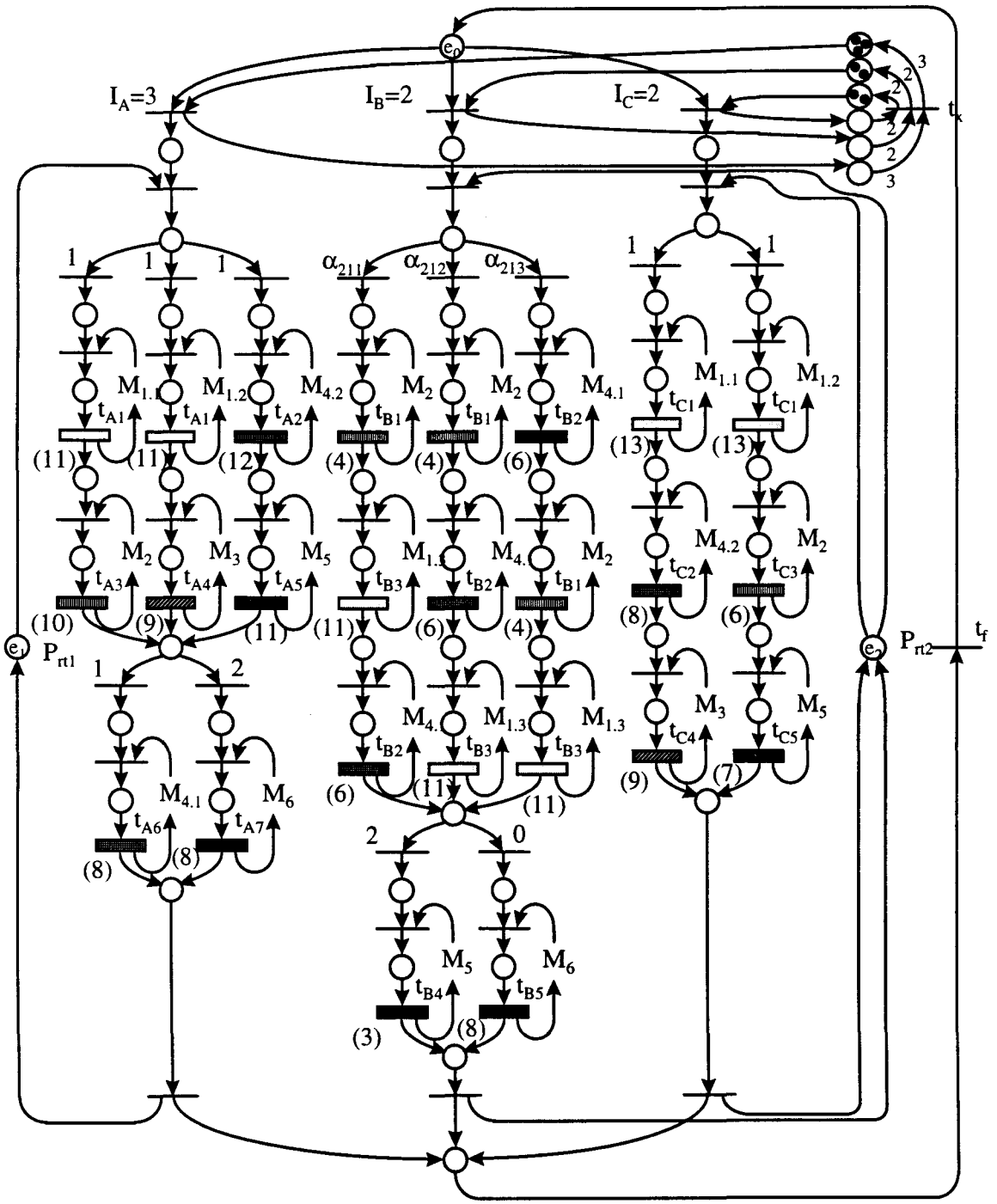


Figure 1 : Résultat de la planification fine

Pour commencer nous allons voir le nombre de solutions issues de la phase de linéarisation de gammes et correspondant à un régime permanent fixé à la fin de la planification fine.

Pour la gamme C aucun choix n'est possible. Il y a donc une seule possibilité. En ce qui concerne la gamme A, nous avons montré qu'il y a 3 possibilités de la construire, cf. Chapitre II.4.2. Concernant la gamme B, nous avons également montré qu'il y 6 solutions distinctes. Ainsi, comme ces trois gammes sont indépendantes le nombre de solutions à la fin de cette phase est le produit des nombres de solutions soit 18 solutions possibles. La figure suivante résume ces solutions. Parmi ces 18 solutions nous distinguons 9 solutions avec deux gammes B identiques et 9 avec deux gammes de B distinctes.

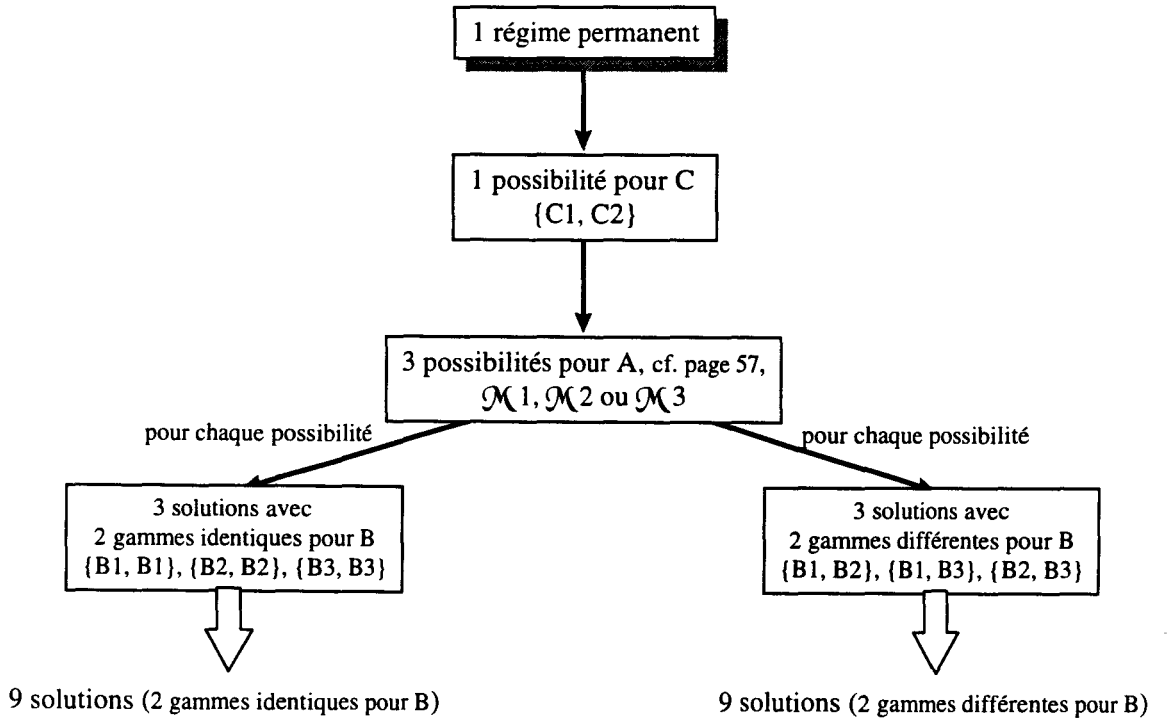


Figure 2 : nombre de solutions à la fin de la linéarisation des gammes

II.3.2 Partition des gammes

Cette phase cherche à déterminer le nombre de partitions possibles dans un ensemble de trois éléments {A1, A2, et A3} et dans un ensemble de quatre éléments (formé des deux exemplaires de B et des deux exemplaires de C.

Nous avons démontré que le nombre de partitions dans le premier ensemble est égal à 5, cf. Chapitre II.5.2, il nous reste donc à déterminer les partitions possible du deuxième ensemble. Pour cela, nous distinguons deux cas disjoints selon que les deux exemplaires

formant la gamme B soient différents ou pas. Dans le premier sous cas nous avons montré que le nombre de solutions est égal à 15, cf. Chapitre II.5.2. La figure suivante rappelle ces solutions :

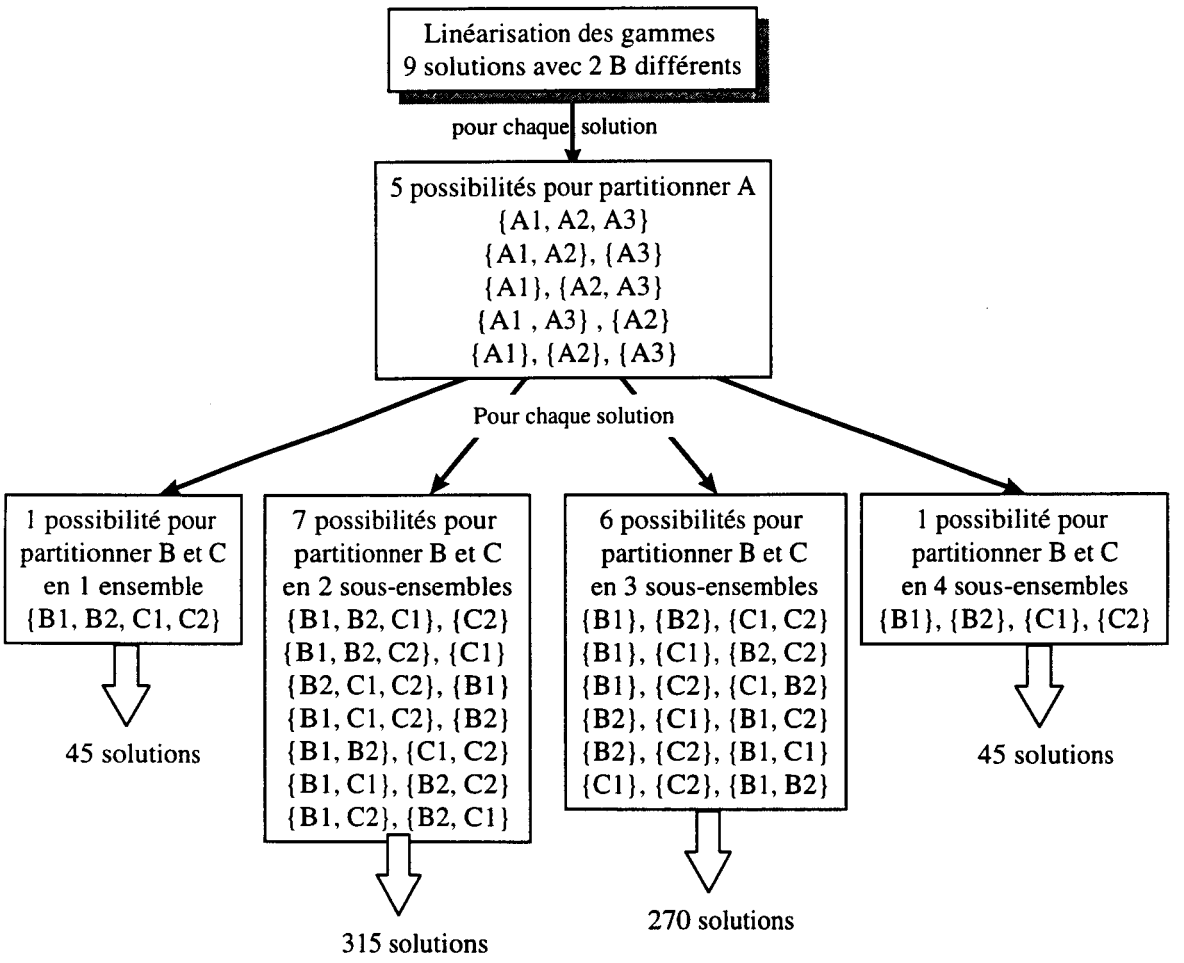


Figure 3 : nombre de solutions à la fin de la partition des gammes (cas où les deux gammes de B sont différentes)

Ainsi, pour le sous cas comprenant deux exemplaires de B différents nous obtenons 5*15= 75 solutions pour chaque graphe linéarisé. Soit 75*9 = 675 solutions possibles en tout.

En ce qui concerne le second sous-ensemble, contenant deux B identiques, nous représentons figure 4 les différentes solutions qui sont au nombre de 11. Par conséquent, le nombre de solutions de ce sous-ensemble est égal à 5 (solutions pour A) * 11 (solutions pour B et C) = 55 solutions possible pour chaque graphe linéarisé. Soit 55*9 (graphes linéarisés possibles avec deux B identiques) = 495.

En conclusion, le nombre de solutions non équivalentes à la fin de cette phase de partition de gammes est égal à : 675 + 495 = 1170 solutions

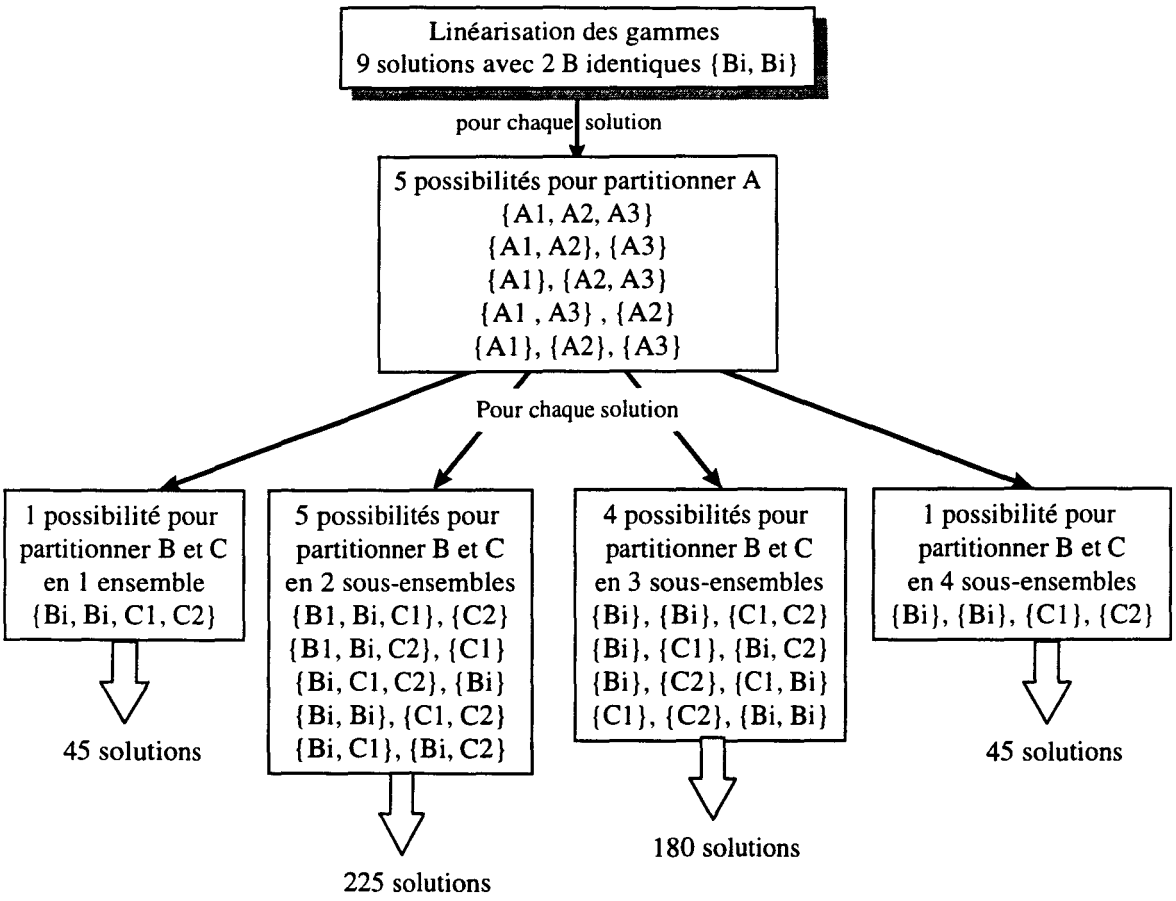


Figure 4 : nombre de solutions à la fin de la partition des gammes (cas où les deux gammes de B sont identiques)

II.3.3 Regroupement Cyclique

Une fois les ensembles partitionnés, il nous reste à déterminer les ordres cycliques dans chaque sous-ensemble (macro-gamme). Pour cela nous allons conserver le classement des solutions en deux sous cas en fonction de la nature des deux exemplaires de B (identiques ou différents).

Nous savons que à partir de 3 gammes nous pouvons obtenir plusieurs regroupements cycliques. En effet, si nous avons deux gammes G1 et G2 faire G1-G2 ou G2-G1 revient au même étant donné le caractère cyclique du problème. De même, si nous avons 3 gammes dont deux identiques, les solutions G1-G1-G2, G1-G2-G1 et G2-G1-G1 sont équivalentes.

Ainsi, pour la gamme A, nous avons 6 solutions possibles à partir des 5 partitions trouvées à la fin de la phase précédente : seule la partition {A1, A2, A3} donne lieu à deux regroupements non équivalents (A1-A2-A3) et (A1-A3-A2). Il nous reste donc à trouver les regroupements possibles pour B et C.

II.3.3.1 Cas où les deux B sont différents

Afin de faciliter la lecture des solutions nous allons nous baser sur la décomposition en nombre de sous-ensembles, cf. figure 3.

- Partition de {B1, B2, C1, C2} en 4 sous-ensembles : étant donné que chaque sous-ensemble est formé d'un élément, il y a une seule solution.

- Partition de {B1, B2, C1, C2} en 3 sous-ensembles : chaque sous-ensemble est formé de 1 ou 2 éléments alors chaque solution donne une seule possibilité.

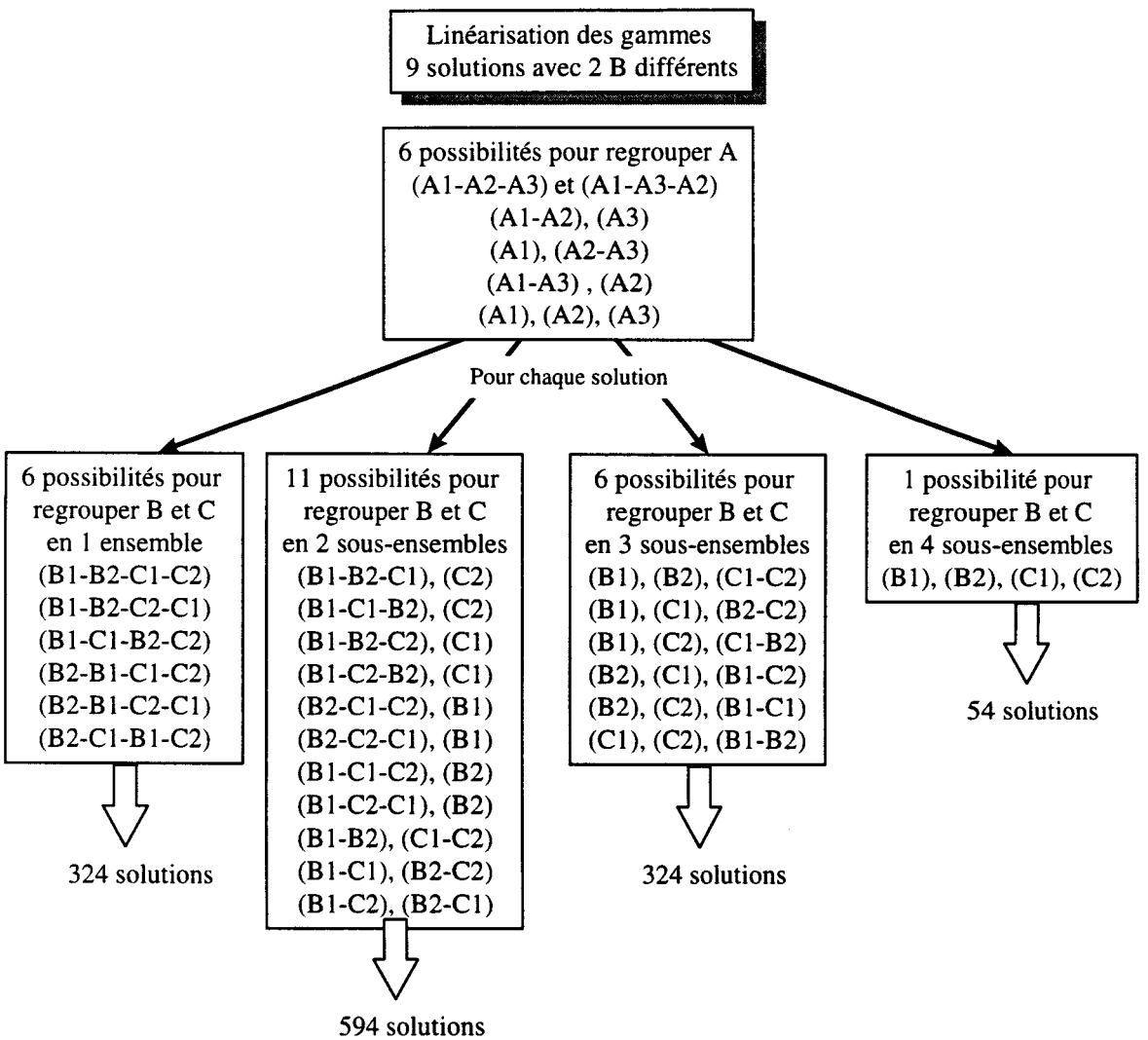


Figure 5 : nombre de regroupements cycliques (deux B différents)

- Partition de {B1, B2, C1, C2} en 2 sous-ensembles : seules les solutions contenant un sous-ensemble de 3 éléments donnent plusieurs solutions les 7 partitions de départ donnent alors 11 regroupements cycliques.

- Partition de {B1, B2, C1, C2} en 1 ensemble : dans ce cas précis nous somment en présence d'un ensemble de 4 éléments distincts : le nombre de solutions est donc, cf. Chapitre II.6.1 lemme 4 :

$$\frac{(4)!}{\prod_{i=1}^{4}(1!)} * \frac{1}{\sum_{i=1}^{4}1} = \frac{4!}{4} = 3! = 6 \text{ solutions}$$

Le nombre de solutions pour ce premiers cas est donc égal à (6+11+6+1)*6 solutions pour chaque graphe linéarisé. Soit 144 * 9 = 1296 solutions.

II.3.3.2 Cas où les deux B sont identiques

Etant donné que les deux représentants de B sont identiques, nous les notons Bi.

- Partition de {Bi, Bi, C1, C2} en 4 sous-ensembles : étant donné que chaque sous-ensemble est formé d'un élément, il y a une seule solution.

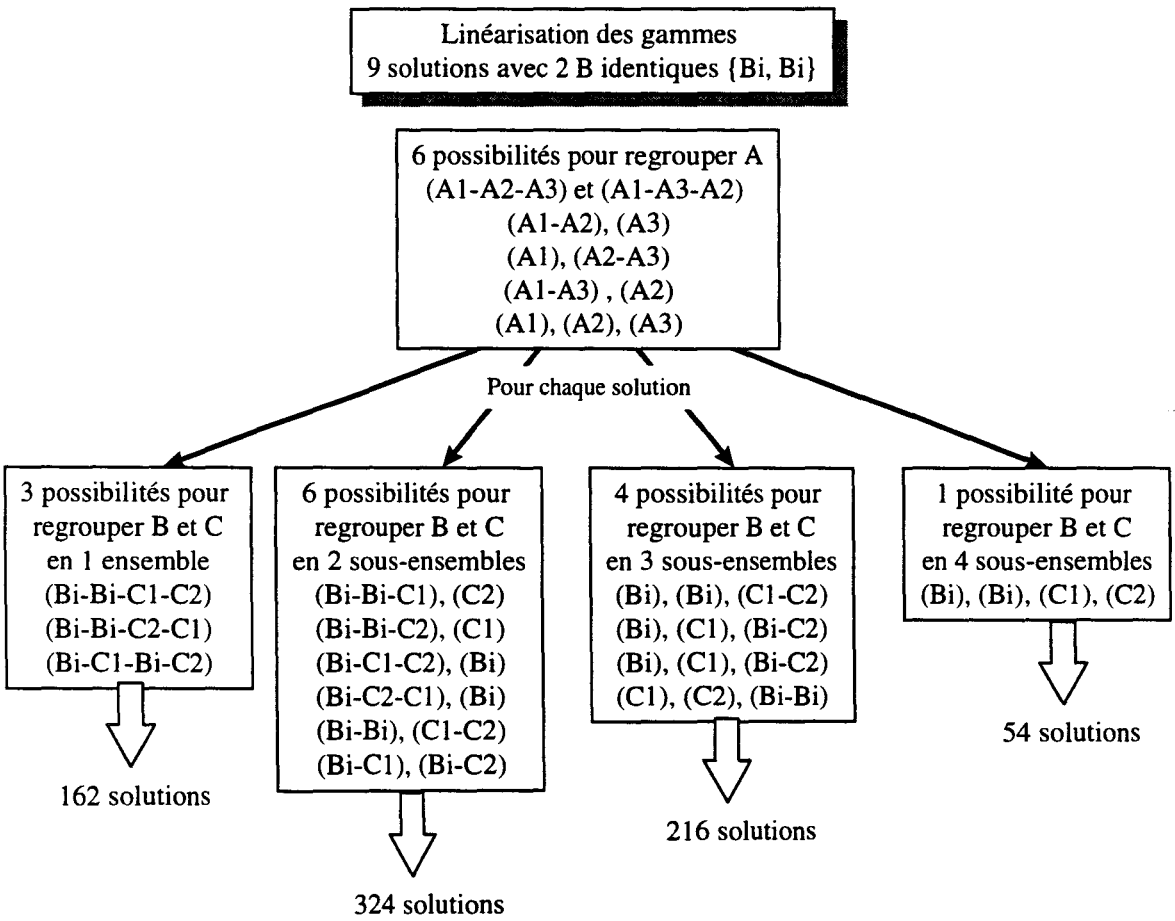


Figure 6 : nombre de regroupements cycliques (deux B identiques)

- Partition de {Bi, Bi, C1, C2} en 3 sous-ensembles : chaque sous-ensemble est formé de 1 ou 2 éléments alors chaque solution donne une seule possibilité.

- Partition de {Bi, Bi, C1, C2} en 2 sous-ensembles : seules les solutions contenant un sous-ensemble de 3 éléments distincts donnent plusieurs solutions. Les 5 partitions de départ donnent alors 6 regroupements cycliques.

- Partition de {B1, B2, C1, C2} en 1 ensemble : dans ce cas précis nous somment en présence d'un ensemble de 3 éléments distincts dont un en double exemplaire : le nombre de solutions est donc, cf. Chapitre II.6.1 lemme 4 :

$$\frac{(1+1+2)!}{1!*1!*2!}*\frac{1}{1+1+2}=\frac{4!}{4*2}=3 \text{ solutions}$$

Le nombre de solutions pour ce premier cas est donc égal à (3+6+4+1)*6 solutions pour chaque graphe linéarisé. Soit 84 * 9 = 756 solutions.

Le nombre final de graphes ordonnançables issus de la solution initiale obtenue de la planification fine est égal à 756+1296 = 2052 solutions.

Annexe III

III.1 Enoncé de l'algorithme

L'énoncé de l'algorithme telle qu'il a été présenté par H. Camus dans son mémoire de thèse, cf. [CAM 97] :

Paramètres de l'algorithme :

- $OP(j,k)$ — k-ème opération de la j-ème macro-gamme,
- $OPM(op_t)$ — machine associée à l'opération op_t,
- $OPMG(op_t)$ — macro-gamme associée à l'opération op_t,
- $\mu(op_t)$ — durée de l'opération op_t,
- M — nombre de machines, (variable associée : i)
- H — nombre de types de ressources de transport, (variable associée : h)
- K_h — nombre de macro-gammes pour le h-ème type de ressource de transport (variable associée : a),
- π^* — profondeur maximale de l'arbre de recherche,
- $CT = CT(RP_p)$ — temps de cycle à respecter,
- C_{WIP} — coût associé au placement d'un en-cours,
- $C_{macr.}$ — coût associé à la perte d'une unité de marge de macro-gammes,
- $C_{mach.}$ — coût associé à la perte d'une unité de marge de machines.

Variables de l'algorithme :

- *ESTG, EST, ESC* — **variables de type état de l'ordonnancement en-cours** :
  ce type est à champ multiple.
  ESC est la variable courante,
  ESTG est la variable utilisée pour conserver l'état de l'ordonnancement avant le placement des séquences prévisionnelles d'opérations,

EST est une variable locale utilisée pour conserver les états d'ordonnancement au cours de la récursivité.

Les différente caractéristiques de cet état sont :

.$OP_{placé}$: l'ensemble des opérations déjà placées,

.$OP_{restant}$: l'ensemble des opérations restantes,

.$\varepsilon_M(M_i)$: la marge de la machine M_i, $\forall i$,

.$NID(M_i)$: nombre d'intervalles de disponibilité de la machine M_i, $\forall i$,

.$I_{d_l}(M_i)$: l-ème intervalle de disponibilité de la machine M_i, $\forall i$, $\forall l$

.$d_d(i,l)$: date de début de l'intervalle de disponibilité $I_{d_l}(M_i)$, $\forall i$, $\forall l$,

.$\mu(I_{d_l}(M_i))$: durée de l'intervalle de disponibilité $I_{d_l}(M_i)$, $\forall i$, $\forall l$,

.$WIP(M_a^h)$: niveau d'en-cours utilisé pour placer les opérations déjà ordonnancées sur la macro-gamme $M_a^h, \forall h, \forall a$,

.$d_d(M_a^h)$: date de disponibilité de la macro-gamme $M_a^h, \forall h, \forall a$,

.$\varepsilon_G(M_a^h)$: marge totale de la macro-gamme $M_a^h, \forall h, \forall a$.

- π profondeur effective de l'arbre de recherche,
- σ séquence prévisionnelle (initialisée à π opérations à placer),

avec $\sigma(s)$: la s-ème opération de la séquence σ,

- $\Omega(\pi)$ ensemble des séquences prévisionnelles non équivalentes de profondeur associée égale à π, avec, pour chaque opération, la politique de placement utilisée.
- $\Omega(\pi)(1)$ première séquence de $\Omega(\pi)$ avec les politiques de placement de chaque opération.
- *pré*$(\Omega(\pi))$ première séquence d'opérations de $\Omega(\pi)$,
- *compteur* nombre de feuilles terminales atteintes pour un ensemble $\Omega(\pi)$,
- $Cout_1$ coût associé à l'en-cours déjà utilisé,
- $Cout_2$ coût associé à la marge de machines,
- $Cout_3$ coût associé à la marge de macro-gammes,

- $Cout_{total}$ somme des trois coûts précédents : coût associé à un état de l'ordonnancement d'une feuille terminale,
- $Cout_{meilleur}$ meilleur coût retenu des feuilles terminales issues de $\Omega(\pi)$,
- $\sigma_{meilleur}$ séquence associée à $Cout_{meilleur}$ (initialisée à $\varnothing$) avec pour chaque opération la politique de placement.

Procédure Rechercher_meilleur_ordo_de_la_séquence (σ, ESC)

- **Si(1)** $\sigma = \varnothing$
- **alors(1)**

  | */* S'il n'y a plus d'opérations à placer, une nouvelle feuille terminale vient*
  | *d'être atteinte. Il faut évaluer le coût de l'état d'ordonnancement pour*
  | *éventuellement le retenir s'il représente le meilleur coût pour le moment. */*
  | $compteur = compteur + 1$
  | $Cout_1 = C_{WIP} * \sum_{\forall h, \forall a} ESC.WIP\left(M_a^h\right)$
  | $Cout_2 = C_{mach.} * \sum_{i=1}^{M} card\left(ESC.OP_{\text{restant}}\Big|_{OPM(j,k)=M_i}\right) * ESC.\varepsilon_M\left(M_i\right)$
  | $Cout_3 = C_{macr.} * \sum_{\forall h, \forall a} card\left(ESC.OP_{\text{restant}}\Big|_{OP(j,k)\in M_a^h}\right) * ESC.\varepsilon_G\left(M_a^h\right)$
  | $Cout_{total} = Cout_1 + Cout_2 + Cout_3$
  | **Si(2)** $Cout_{total} < Cout_{meilleur}$
  | **alors(2)**
  | | $Cout_{meilleur} = Cout_{total}$
  | | $\sigma_{meilleur} = \Omega(\pi)(1)$
  | **Fin alors(2)**
- **Sinon(1)**

  | */* Il reste encore des opérations de la séquence* σ *à placer. Il faut ordonnancer*

| *la prochaine opération suivant les différentes politiques de placement. */*

| **Si(2)** $ESC.OP_{placé}\big|_{OPM(\sigma(1))} = \varnothing$,

| **alors(2)**

| */* Si aucune opération de la machine associée à la prochaine opération à placer n'a encore été ordonnancée, cette opération est placée directement à la date de disponibilité de la macro-gamme correspondante pour éviter de perdre de la marge de macro-gammes et d'augmenter inutilement l'en-cours. Ce placement modifie la date de début et la durée de l'unique intervalle de disponibilité de la machine associée */*

| | $ESC.OP_{restant} = ESC.OP_{restant} - \{\sigma(1)\}$

| | $ESC.OP_{placé} = ESC.OP_{placé} \cup \{\sigma(1)\}$

| | Modifier $ESC.d_d(OPM(\sigma(1)),1)$ */* date de début du seul intervalle de disponibilité de la machine associée à l'opération $\sigma(1)$ */*

| | Modifier $ESC.\mu(I_{d_1}(OPM(\sigma(1))))$ */* durée de l'intervalle de disponibilité */*

| | Modifier $ESC.d_d(OPMG(\sigma(1)))$ */* date de disponibilité de la macro-gamme associée à $\sigma(1)$ */*

| | Ajouter si nécessaire un en-cours dans $ESC.WIP(OPMG(\sigma(1)))$

| | Indiquer la politique de placement de $\sigma(1)$ dans $pré(\Omega(\pi))$

| | *Rechercher_meilleur_ordo_de_la_séquence* $(\sigma \setminus \sigma(1), ESC)$

| **Fin alors(2)**

| **Sinon(2)**

| | */* Politique de placement, si c'est possible, de la prochaine opération de la séquence prévisionnelle à la date de disponibilité de la macro-gamme associée. Si cela conduit à un ordonnancement réalisable, placement de l'opération suivante de la séquence à l'aide de la récurrence */*

| | $EST = ESC$

| | **Si(3)** il existe un intervalle de disponibilité ($ESC.I_{d_1}(OPM(\sigma(1)))$) de la

| | machine $OPM(\sigma(1))$ comprenant l'opération $\sigma(1)$ placée à la date de

| | disponibilité de la macro-gamme associée $OPMG(\sigma(1))$ et dont la date de
| | début ne coïncide pas avec la date de disponibilité de $OPMG(\sigma(1))$ ou dont
| | la date de fin ne coïncide pas avec la date de fin de l'opération $\sigma(1)$,
| | **alors(3)**
| | | $ESC.OP_{restant} = ESC.OP_{restant} - \{\sigma(1)\}$
| | | $ESC.OP_{placé} = ESC.OP_{placé} \cup \{\sigma(1)\}$
| | | Incrémenter $ESC.NID(OPM(\sigma(1)))$ */* nombre d'intervalles de*
| | | *disponibilité de la machine* $OPM(\sigma(1))$ */
| | | Créer un nouvel intervalle terminant l'ancien $ESC.I_{d_l}(OPM(\sigma(1)))$ (date
| | | de début et durée)
| | | Modifier $ESC.\mu(I_{d_l}(OPM(\sigma(1))))$
| | | Modifier $ESC.d_d(OPMG(\sigma(1)))$
| | | Modifier si nécessaire $ESC.\varepsilon_M(OPM(\sigma(1)))$
| | | Ajouter si nécessaire un en-cours dans $ESC.WIP(OPMG(\sigma(1)))$
| | | Indiquer la politique de placement de $\sigma(1)$ dans $pré(\Omega(\pi))$
| | | **Si(4)** la condition CNS d'existence d'un ordonnancement réalisable est
| | | vérifiée,
| | | **alors(4)**
| | | *Rechercher_meilleur_ordo_de_la_séquence* $(\sigma \setminus \sigma(1), ESC)$
| | | **Fin alors(4)**
| | $ESC = EST$
| | **Fin alors(3)**
| | /* *Placement de l'opération* $\sigma(1)$ *dans tous les intervalles pouvant*
| | *l'accueillir selon deux façons : au plus tôt et au plus tard* */
| | **Pour(1)** tous les intervalles $ESC.I_{d_l}(OPM(\sigma(1)))$ tels que
| | $ESC.\mu(I_{d_l}(OPM(\sigma(1)))) \geq \mu(\sigma(1))$, faire
| | | /* *Politique de placement de la prochaine opération au début de*

| | | *l'intervalle de disponibilité* $ESC.I_{d_l}(OPM(\sigma(1)))$ *de la machine associée.*

| | | *Après vérification de la CNS d'existence d'un ordonnancement*

| | | *prévisionnel, placement de l'opération suivante de la séquence avec la*

| | | *récurrence */*

| | | $ESC.OP_{\text{restant}} = ESC.OP_{\text{restant}} - \{\sigma(1)\}$

| | | $ESC.OP_{placé} = ESC.OP_{placé} \cup \{\sigma(1)\}$

| | | Modifier $ESC.d_d(OPM(\sigma(1)),l)$

| | | Modifier $ESC.\mu(I_{d_l}(OPM(\sigma(1))))$

| | | **Si** $ESC.\mu(I_{d_l}(OPM(\sigma(1))))=0$ **alors** supprimer $ESC.I_{d_l}(OPM(\sigma(1)))$

| | | Modifier $ESC.d_d(OPMG(\sigma(1)))$

| | | Modifier si nécessaire $ESC.\varepsilon_G(OPMG(\sigma(1)))$

| | | Modifier si nécessaire $ESC.\varepsilon_M(OPM(\sigma(1)))$

| | | Ajouter si nécessaire un en-cours dans $ESC.WIP(OPMG(\sigma(1)))$

| | | Indiquer la politique de placement de $\sigma(1)$ dans $pré(\Omega(\pi))$

| | | **Si(3)** la condition CNS d'existence d'un ordonnancement réalisable est

| | | vérifiée,

| | | **alors(3)**

| | | | *Rechercher_meilleur_ordo_de_la_séquence* $(\sigma \setminus \sigma(1), ESC)$

| | | **Fin alors(3)**

| | | $ESC = EST$

| | | */* Politique de placement de la prochaine opération à la fin de l'intervalle*

| | | *de disponibilité* $ESC.I_{d_l}(OPM(\sigma(1)))$ *de la machine associée. Après*

| | | *vérification de la CNS d'existence d'un ordonnancement prévisionnel,*

| | | *placement de l'opération suivante de la séquence avec la récurrence */*

| | | $ESC.OP_{\text{restant}} = ESC.OP_{\text{restant}} - \{\sigma(1)\}$

| | | $ESC.OP_{placé} = ESC.OP_{placé} \cup \{\sigma(1)\}$

| | | Modifier $ESC.\mu(I_{d_l}(OPM(\sigma(1))))$

| | | **Si** $ESC.\mu\left(I_{d_1}\left(OPM(\sigma(1))\right)\right)=0$ **alors** supprimer $ESC.I_{d_1}\left(OPM(\sigma(1))\right)$

| | | Modifier $ESC.d_d\left(OPMG(\sigma(1))\right)$

| | | Modifier si nécessaire $ESC.\varepsilon_G\left(OPMG(\sigma(1))\right)$

| | | Modifier si nécessaire $ESC.\varepsilon_M\left(OPM(\sigma(1))\right)$

| | | Ajouter si nécessaire un en-cours dans $ESC.WIP\left(OPMG(\sigma(1))\right)$

| | | Indiquer la politique de placement de $\sigma(1)$ dans $pré(\Omega(\pi))$

| | | **Si** la condition CNS d'existence d'un ordonnancement réalisable est

| | | vérifiée,

| | | **alors(3)**

| | | *Rechercher_meilleur_ordo_de_la_séquence* $(\sigma \setminus \sigma(1), ESC)$

| | | **Fin alors(3)**

| | *ESC = EST*

| | **Fin Pour(1)**

| **Fin Sinon(2)**

- **Fin Sinon(1)**

Initialisation

- $ESC.OP_{placé} = \varnothing$
- $ESC.OP_{\text{restant}} = \left\{OP(j,k), \forall j, \forall k\right\}$
- $\forall i \in \{1..M\}$, $ESC.NID\left(M_i\right)=1$, $ESC.d_d(i,1)=0$ et $ESC.\mu\left(I_{d_1}\left(M_i\right)\right)=CT$
- $\forall h \in \{1..H\}, \forall a \in \left\{1..K_h\right\}$, $ESC.d_d\left(M_a^h\right)=0$, $ESC.WIP\left(M_a^h\right)=1$
- Calculer $ESC.\varepsilon_M\left(M_i\right)$, $\forall i \in \{1..M\}$
- Calculer $ESC.\varepsilon_G\left(M_a^h\right)$, $\forall h \in \{1..H\}, \forall a \in \left\{1..K_h\right\}$

Algorithme de placement

- **Tant que(1)** $card(OP_{restant}) \neq \varnothing$, faire

  | Déterminer la profondeur π des séquences prévisionnelles :

  | $\pi = \min(\pi^*, card(OP_{restant}))$

  | Calculer l'ensemble des séquences prévisionnelles σ représentant de chaque

  | classe d'équivalence : $\Omega(\pi)$

  | $compteur = 0$

  | $Cout_{meilleur} = \infty$

  | $ESTG = ESC$

  | **Tant que(2)** $card(\Omega(\pi)) \neq \varnothing$, faire

  | | $ESC = ESTG$

  | | *Rechercher_meilleur_ordo_de_la_séquence* $(\sigma = pré(\Omega(\pi)), ESC)$

  | | $\Omega(\pi) = \Omega(\pi) - \{\Omega(\pi)(1)\}$

  | **Fin Tant que(2)**

  | $ESC = ESTG$

  | Placer réellement la première opération de la meilleure séquence $\sigma_{meilleur}$ avec

  | la politique de placement précisé et modifier *ESC* en conséquence.

- **Fin Tant que(1)**

III.2 Application de l'heuristique d'ordonnancement

III.2.1 Tableau de valeurs

Nous donnons ici les valeurs, résultat de l'application de notre heuristique d'ordonnancement sur l'exemple illustratif, de l'en-cours optimal, du nombre de branches étudiées, du nombre de séquences non équivalentes et du temps de calcul, en fonction de la profondeur de recherche. Cette profondeur de recherche varie entre ses deux valeurs extrêmes : profondeur égale à 1 pour la valeur minimale et profondeur égale au nombre maximal d'opérations, à réaliser durant un cycle, pour la valeur maximale.

Profondeur	En-cours	Temps de Calcul	Branches	Séquences
1	12	0,27	90	68
2	10	0,54	842	425
3	10	0,87	1917	712
4	11	1,42	3893	1040
5	11	2,25	5081	1170
6	11	3,07	7657	1495
7	10	4,45	12344	1881
8	10	7,96	22791	2514
9	10	24,12	53173	3633
10	10	66,26	138491	4907
11	10	129,94	279651	6045
12	10	188,84	437194	7126
13	10	303	874097	8647
14	10	1530	4097357	11211
15	10	1245	2732606	11449
16	10	2700	6773281	14236
17	10	6293	17294236	16029
18	9	7571	19401721	16107
19	10	20000	59959256	18328
20	9	38270	77234390	17487
21	9	50335	148601340	16547
22	9	72968	258344594	13927
23	9	134672	317621137	8511

III.2.2 Explication des variations de la combinatoire

Nous allons expliquer le phénomène, de non monotonie du nombre de séquences non équivalentes en fonction de la profondeur de recherche, sur un exemple « d'école ». Soit les deux gammes suivantes :

Gamme 1 : Op1 (M1, 1u.t.), Op2 (M2, 1 u.t.)

Gamme 2 : Op3 (M3, 1 u.t.).

Avec une profondeur de 2 nous obtenons le résultat suivant : à la première itération l'heuristique trouve deux séquences non équivalentes[1] Op1-Op2 et Op1-Op3. Dans les deux cas l'algorithme place Op1 en premier lieu et passe à l'itération suivante où il trouve une seule séquence[2] Op2-Op3 pour laquelle il place Op2. À la dernière itération

[1] Op3-Op1 est équivalente à Op1-OP3 car les deux opérations utilisent deux machines différentes et proviennent de deux macro-gammes différentes.

[2] Op3-Op2 est équivalente à Op2-Op3 pour les mêmes raisons.

l'algorithme dispose d'une seule opération donc une seule séquence. Au total le nombre de séquences traitées est égal à 4.

Passons maintenant à une profondeur égale à 3. A la première itération, nous avons une seule séquence[1] Op1-Op2-Op3. On place donc Op1 pour obtenir, à la deuxième itération, la séquence Op2-Op3 et pour finir, à la dernière itération, la séquence Op3. Le nombre de séquences non équivalentes a donc baissé malgré l'augmentation de la profondeur. Les branches terminales ne sont pas à l'abri de ces variations brusques et imprévisibles en fonction de la profondeur de recherche.

III.2.3 Exemples bibliographiques de référence

Dans cette partie, nous présentons les exemples bibliographiques qui ont été utilisés a titre de comparaison avec notre heuristique, cf. Chapitre III.2.4 Tableau III.1.

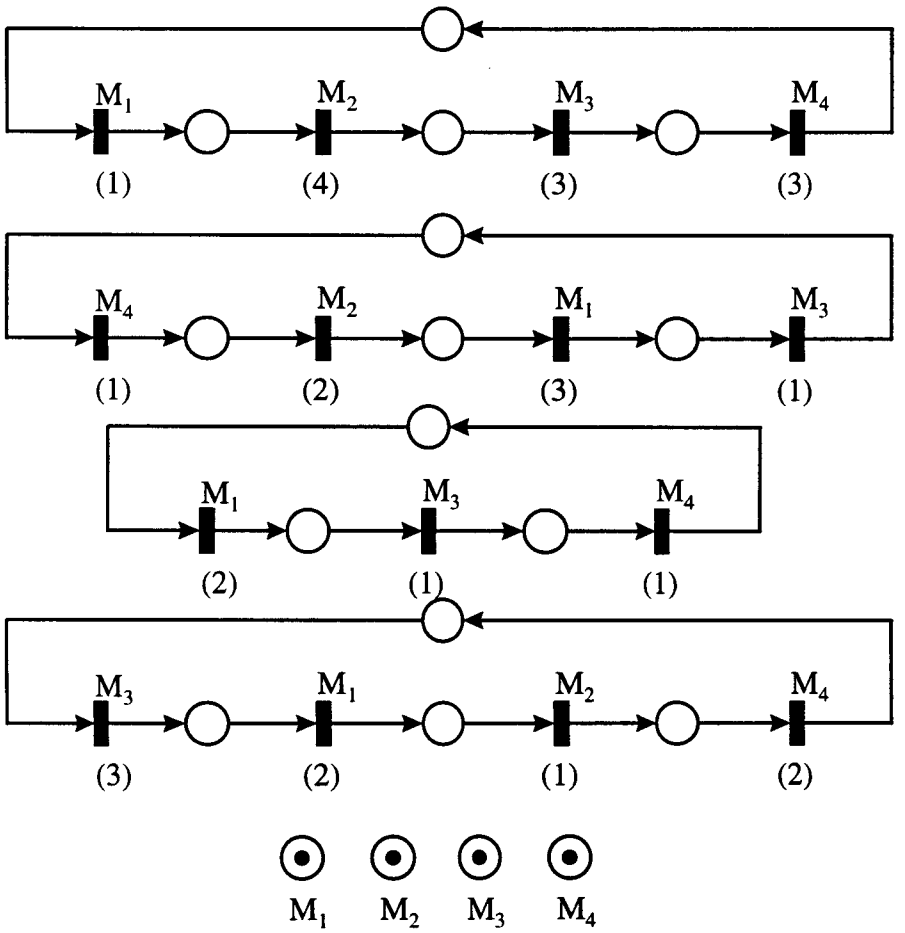


Figure 1 : Exemple [HIL 87]

[1] Les séquences possibles sont Op1-Op2-Op3, Op1-Op3-Op2 et Op3-Op1-Op2 (attention aux contraintes de précédence de Gamme 1) ces trois séquences sont toutes équivalentes puisque les opérations utilisent des machines différentes.

Le premier exemple, cf. [HIL 87], contient 4 gammes utilisant 4 machines (au total 15 opérations) :

- G1 : M_1 (1 u.t.), M_2 (4 u.t.), M_3 (3 u.t.), M_4 (3 u.t.).
- G2 : M_4 (1 u.t.), M_2 (2 u.t.), M_1 (3 u.t.), M_3 (1 u.t.).
- G3 : M_1 (2 u.t.), M_3 (1 u.t.), M_4 (1 u.t.).
- G4 : M_3 (3 u.t.), M_1 (2 u.t.), M_2 (1 u.t.), M_4 (2 u.t.).

Quant au second, cf. [HIL 88], il contient 4 gammes utilisant 3 machines (au total 9 opérations) :

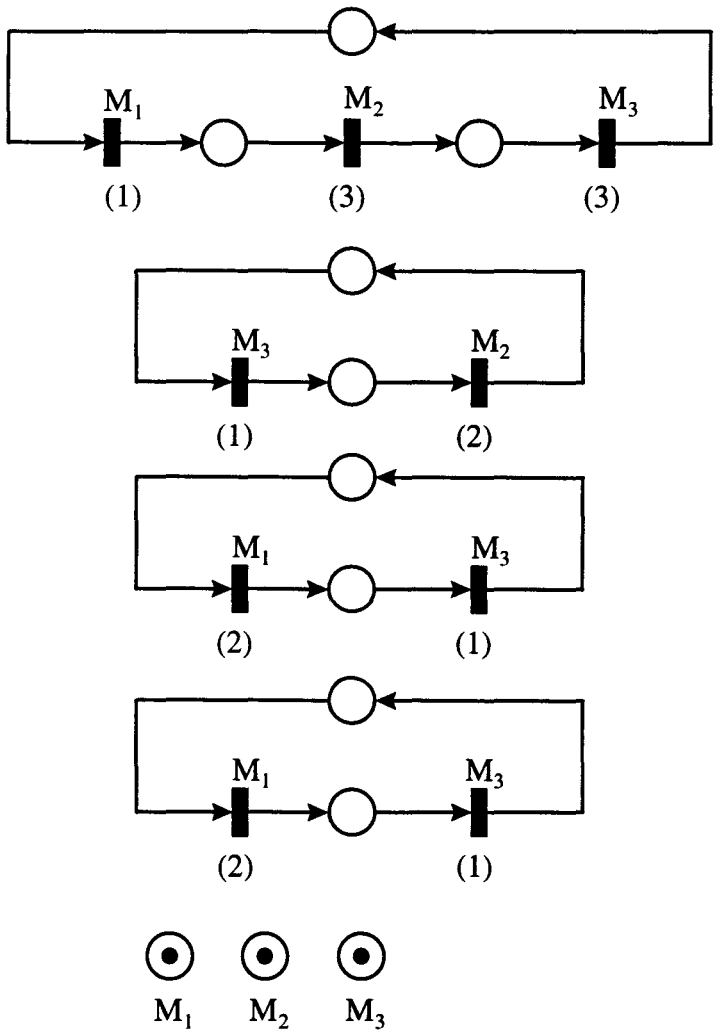


Figure 2 : Exemple [HIL 88]

- G1 : M_1 (1 u.t.), M_2 (3 u.t.), M_3 (3 u.t.).
- G2 : M_3 (1 u.t.), M_2 (2 u.t.).

- G3 : M_1 (2 u.t.), M_3 (1 u.t.).
- G4 : M_1 (2 u.t.), M_3 (1 u.t.).

Le troisième exemple, cf. [VAL 94], contient 5 gammes utilisant 3 machines (au total 13 opérations) :

- G1 : U_1 (2 u.t.), M_1 (3 u.t.), M_2 (2 u.t.).
- G2 : U_1 (2 u.t.), M_1 (3 u.t.), M_2 (2 u.t.).
- G3 : U_1 (2 u.t.), M_1 (3 u.t.), M_2 (2 u.t.).
- G4 : M_1 (1 u.t.), U_1 (2 u.t.).
- G5 : M_1 (1 u.t.), U_1 (2 u.t.).

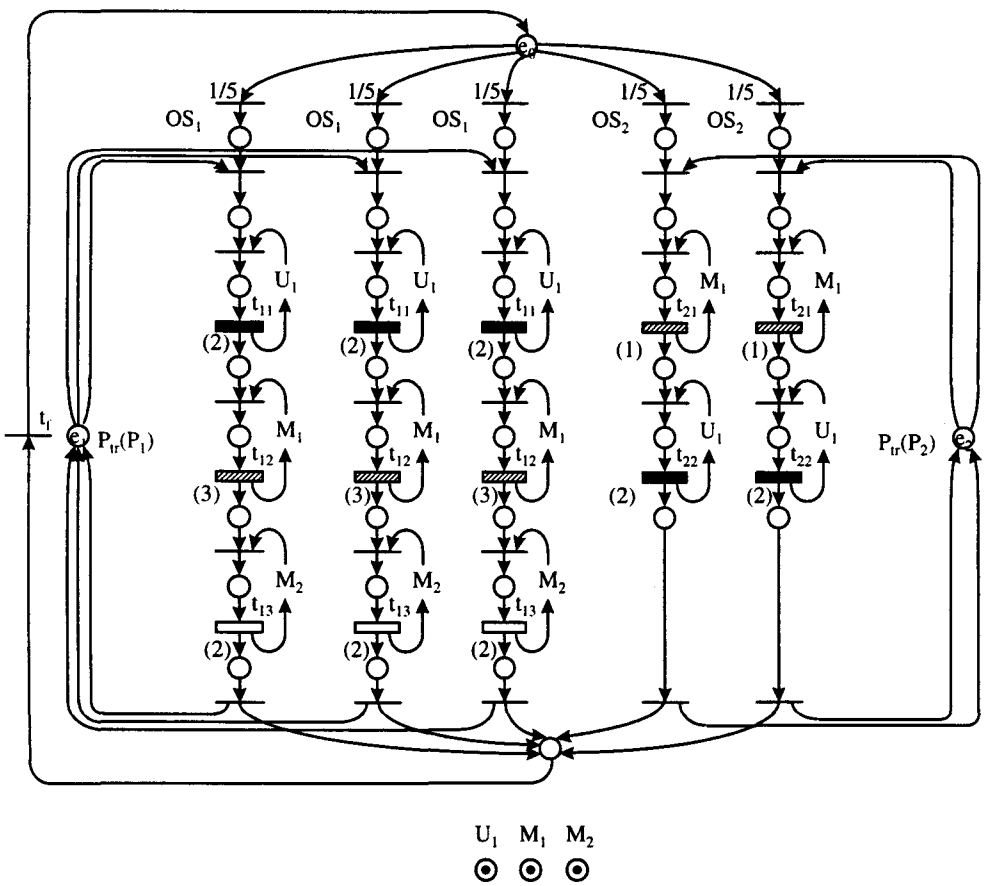


Figure 3 : Exemple [VAL 94]

Finalement, le dernier exemple est tiré de la Thèse de H. Ohl, cf. [OHL 95a]. Il contient deux gammes utilisant 6 machines (au total 10 opérations) :

- G1 : R_1 (18 u.t.), $R_{3.1}$ (14 u.t.), R_2 (12 u.t.), $R_{3.1}$ (14 u.t.), R_2 (10 u.t.), $R_{3.2}$ (14 u.t.).
- G2 : R_4 (5 u.t.), R_6 (4 u.t.), R_4 (5 u.t.), R_6 (4 u.t.).

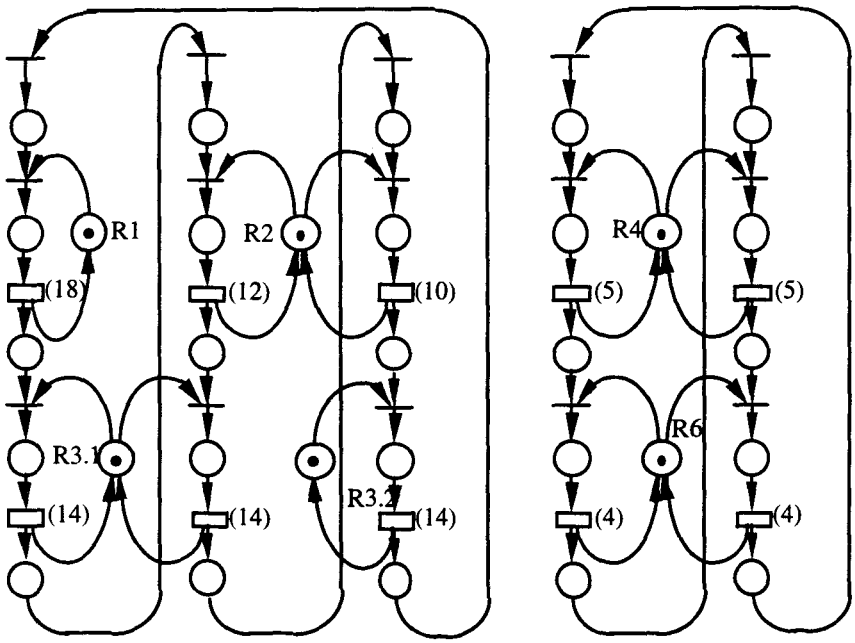


Figure 4 : Exemple [OHL 95a]

III.2.4 Apport du regroupement cyclique

Afin de mettre l'accent sur l'apport du regroupement cyclique, nous allons appliquer, sur le même exemple, différentes méthodes : Algorithme de Hillion, celui de Valentin et notre heuristique avec regroupement cyclique. L'exemple de départ est tiré de la bibliographie, cf. [VAL 94] et figure 3, il est formé de 5 gammes sur 3 machines :

- G1 : U_1 (2 u.t.), M_1 (3 u.t.), M_2 (2 u.t.).
- G2 : U_1 (2 u.t.), M_1 (3 u.t.), M_2 (2 u.t.).
- G3 : U_1 (2 u.t.), M_1 (3 u.t.), M_2 (2 u.t.).
- G4 : M_1 (1 u.t.), U_1 (2 u.t.).
- G5 : M_1 (1 u.t.), U_1 (2 u.t.).

L'application de l'heuristique de Hillion donne 6 palettes pour respecter le temps de cycle optimal, cf. [KOR 97a] :

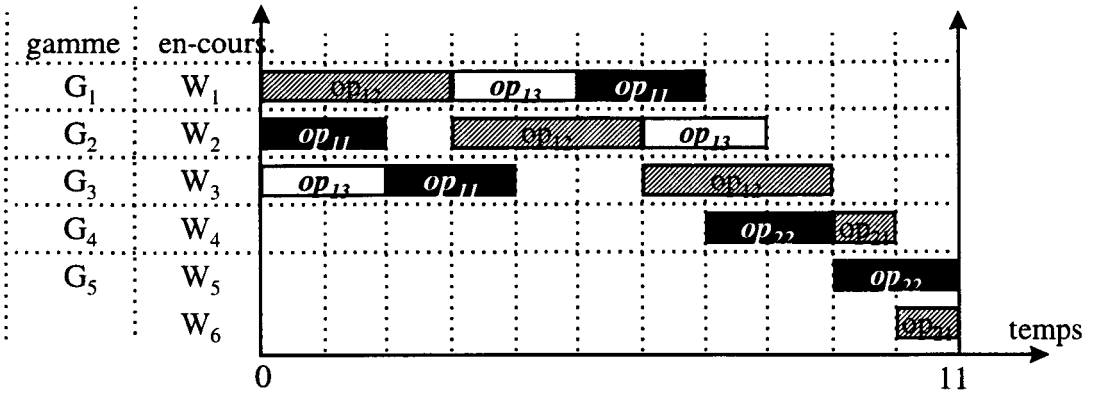


Figure 5 : Ordonnancement obtenu par l'algorithme de Hillion

En ce qui concerne l'heuristique de Valentin, elle nous donne un résultat meilleur puisqu'elle se contente de 5 palettes pour respecter le temps de cycle :

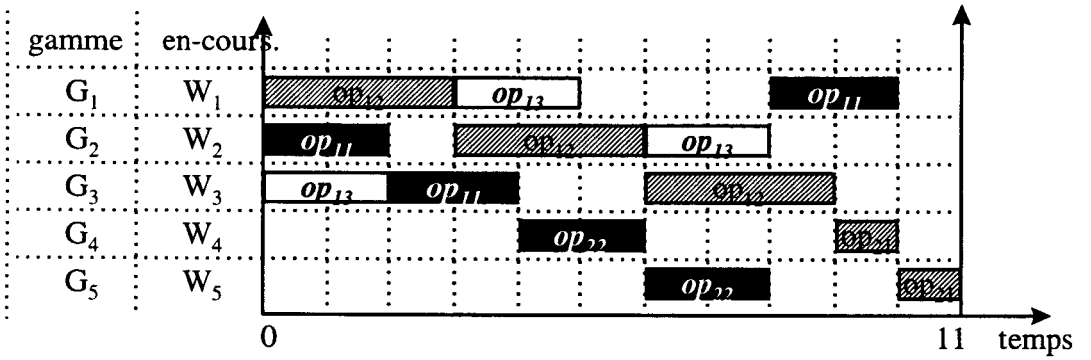
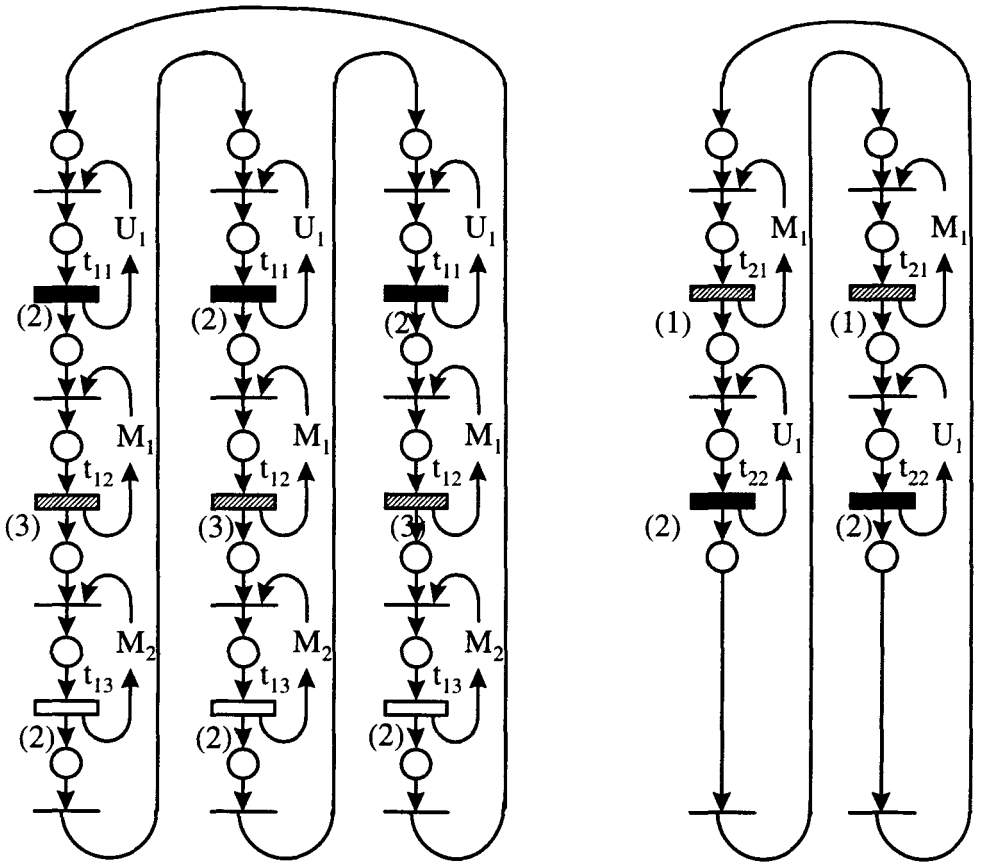


Figure 6 : Ordonnancement par la méthode de Valentin

Ce résultat est le meilleur qu'on peut espérer dans cette configuration puisqu'il est égal à la borne inférieure d'en-cours. Cependant, en considérant que les trois premières gammes sont identiques, rien ne nous empêche de leur affecter les mêmes palettes et de même pour les deux dernières gammes. Ainsi, on peut construire deux macro-gammes à partir de ces 5 gammes initiales :

- M1 : G1-G2-G3 ;
- M2 : G4-G5.

Notons qu'il existe une seule solutions de regroupement car les gammes formant une macro-gamme sont identiques. Nous obtenons alors le schéma suivant :



Macro-gamme M_1 Macro-gamme M_2

U_1 M_1 M_2

Figure 0-7 : Regroupement cyclique

En conséquence, l'application de notre heuristique, à cette nouvelle configuration, donne résultat sensiblement meilleur puisque il requiert uniquement 3 palettes pour respecter le temps de cycle optimal :

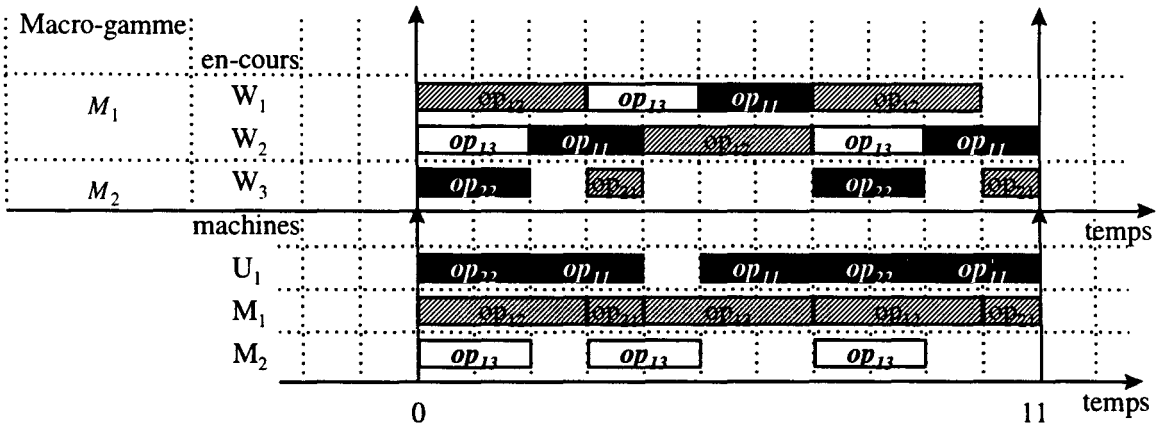


Figure 0-8 : Ordonnancement final.

Annexe IV

IV.1 Démonstrations des Lemmes du Chapitre IV

Rappels :

- Toutes les dates ainsi que les durées des opérations sont multiples de 1.
- Le nombre de palettes utilisé par une gamme i à une date t ($n_p(i,t)$) est une fonction constante discontinue à la date de fin et à la date de début de la gamme i.

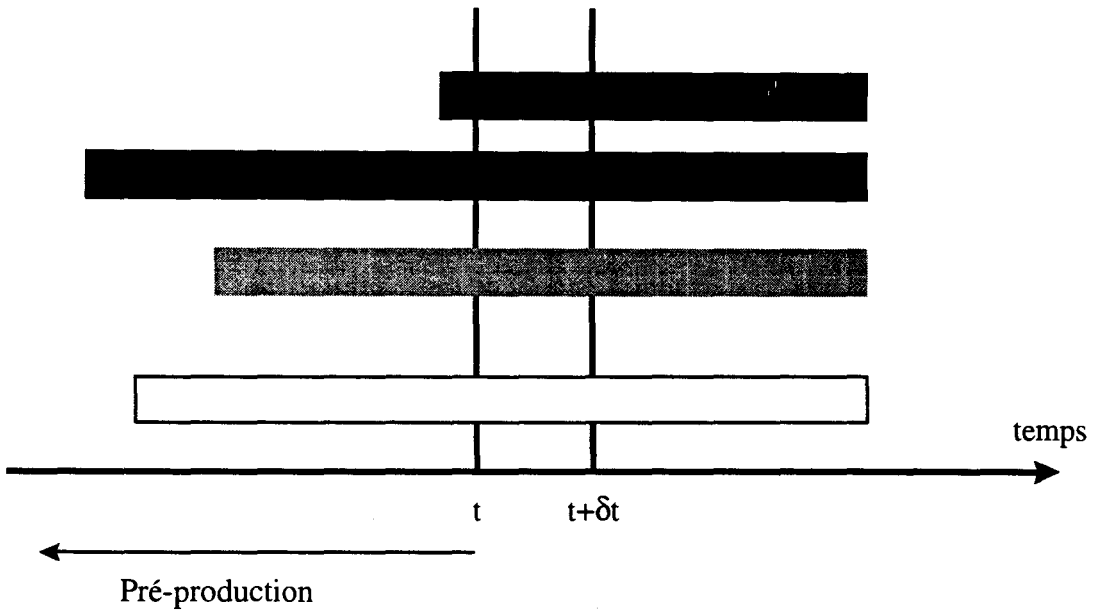
Lemme IV.1

Les bornes inf. et sup. Ainsi que ***la durée du régime pré-production sont croissantes par intervalles.***

Démonstration :

soit $t \in [0, CT]$ et $\delta t \geq 0$ tel que il n'existe pas de gamme i / $t < DFGL(i,0) \leq t + \delta t$ (pas de date de fin de gamme entre t et $t + \delta t$).

Nous savons donc qu'en passant de DDRP = t à DDRP = $t + \delta t$ alors nous ne diminuons pas le nombre d'opérations à lancer durant le pré-production. En effet, comme nous ne traversons pas de date de fin de gamme, alors nous n'avons pas à différer une gamme vers la fin de la production donc nous ne pouvons que augmenter le nombre d'opérations :

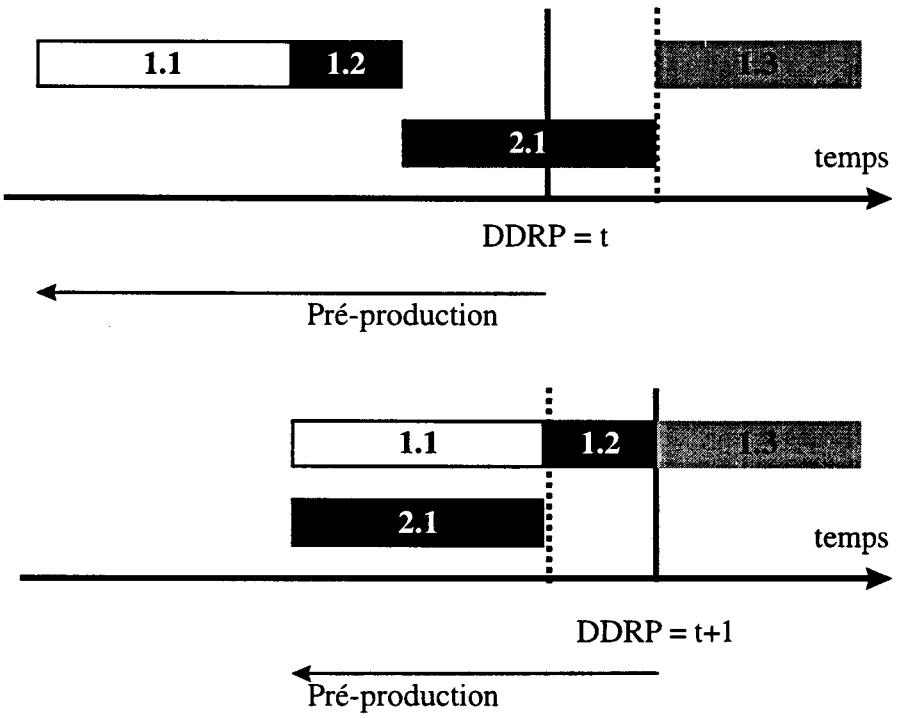


Comme nous augmentons le nombre d'opérations alors la somme de charges par palette augmente ⇒ la borne min. du pré-production est croissante. Le même raisonnement est

applicable à la borne sup. Pour déduire qu'elle est également croissante. Il est également clair que les points de discontinuité des deux bornes sont des dates de fin de gammes.

Quant à la durée optimale du régime pré-production, elle est bien entendu **croissante** (même raisonnement que précédemment) mais **ses points de discontinuité sont des dates de fin d'opérations**. En fait, en passant de t à t+1, si nous ne traversons pas de date de fin d'opération, alors nous n'augmentions pas le nombre d'opérations dans le pré-production et la durée du régime pré-production est soit constante soit augmente de 1.

Cependant, en traversant une date de fin d'opération, soit nous gardons le cas précédent soit nous pouvons introduire une discontinuité dans la durée du régime de pré-production. En effet, si en passant de DDRP = t à DDRP = t + δt nous faisons passer une opération entièrement dans le pré-production, i.e. opération 2.1 figure ci-dessous, alors nous pouvons changer le séquencement des opérations sur les machines et par conséquent modifier sensiblement la durée du régime pré-production :



En conclusion, entre deux dates de discontinuité, le régime pré-production est croissant (de coefficient directeur = 1) ou constant. Ses dates de discontinuité sont des dates de fin d'opérations ■

Lemme IV.2

*La **durée du régime permanent est strictement décroissante par morceaux**. Le coefficient directeur est égal à -1 et les points de discontinuité sont des Dates de Fin de Gammes (DFGL).*

Démonstration :

$$\text{Durée Régime Permanent } (DDRP) = \min_i \left\{ \left(X - n_p(i, DDRP) \right) * CT + DDGL(i, DDRP) \right\}$$

En passant de DDRP = t à DDRP = t+1 nous ne pouvons que rencontrer un des cas suivants :

1. ∃ gamme i / t+1 = DFGL(i,0).
2. ∃ gamme i / t+1 = DDGL(i,0).
3. 1 et 2 sont faux

Rappelons que $DDGL(i,t) = DDGL(i,0) - t \quad \{+\text{CT} \quad \text{si} < 0\}$ d'où :

$$\left(X - n_p(i,t) \right) * CT + DDGL(i,t) = -\left(n_p(i,t) * CT + t \right) + \overbrace{X * CT + DDGL(i,0)}^{\text{terme constant}} \{+ CT \text{si} < 0\}$$

Pour le troisième cas, nous savons qu'en passant de t à t+1 nous ne changeons pas le nombre de palettes donc la durée du régime permanent est de la forme - t + C^{te}.

Pour le deuxième cas, à t+1 la palette n'est pas encore prise (même cas que le précédent) par contre à t + 1 + δt le nombre de palettes est augmenté de 1 cependant comme t+1 = DDGL(i,0) alors :

$$\begin{aligned} DDGL(i, t+1+\delta tt) &= DDGL(i,0) - t - 1 - \delta t \quad \{+\text{CT} \quad \text{si} < 0\} \\ &= t + 1 - t - 1 - \delta t \quad \{+\text{CT} \quad \text{si} < 0\} \\ &= -\delta t \quad \{+\text{CT} \quad \text{si} < 0\} \\ &= CT - \delta t \end{aligned}$$

alors :

$$\left(X - n_p(i, t+1+\delta t) \right) * CT + DDGL(i, t+1+\delta t) = -\left(\left(n_p(i,t) + 1 \right) * CT + t + 1 + \delta t \right) + \overbrace{X * CT + DDGL(i,0) + CT}^{\text{terme constant}}$$
$$= \left(X - n_p(i, t+1) \right) * CT + DDGL(i,0) - \delta t$$

Par conséquent, la durée du régime permanent est strictement décroissante de coefficient directeur égal à -1. Ses dates de discontinuité sont des dates de fin de gammes.

Lemme IV.3

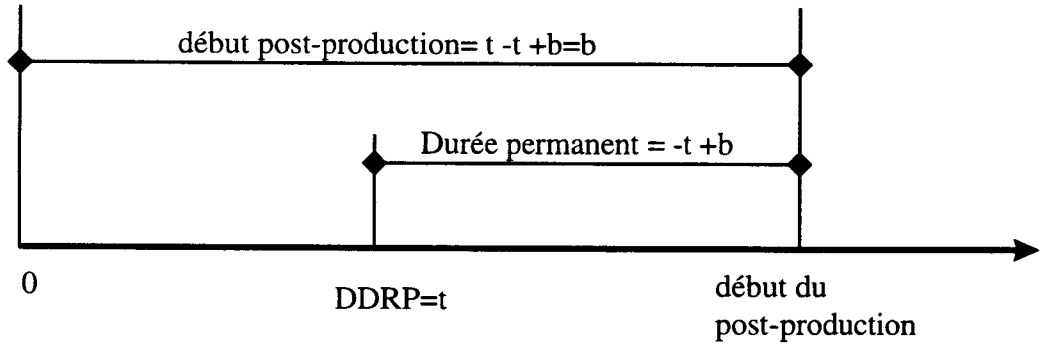
Les bornes inf. et sup. et ***la durée du régime post-production*** *sont* ***constantes par intervalles.*** *Les points de discontinuité sont des Dates de Fin de Gammes Linéaires (DFGL).*

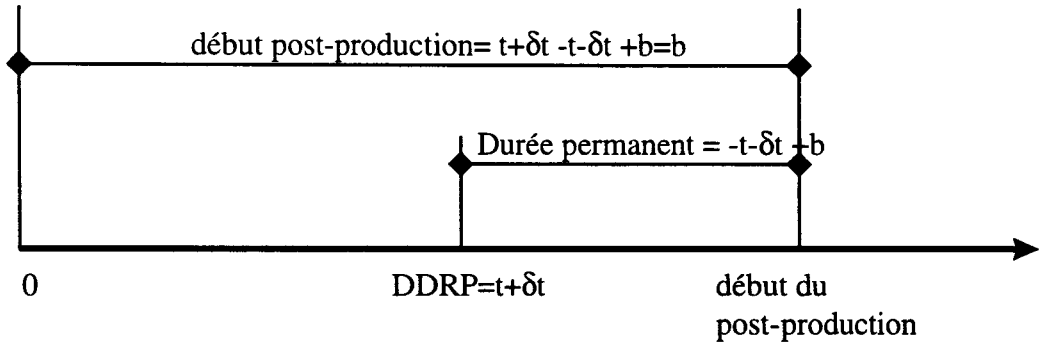
Démonstration :

Soit $t \in [0, CT]$ et $\delta t \geq 0$ tel que il n'existe pas de gamme i / $t < DDGL(i,0) \leq t + \delta t$ (pas de date de fin de gamme entre t et $t + \delta t$).

Nous venons de voir dans la démonstration précédente que, entre deux dates de fin de gammes, la durée du régime permanent est de la forme -x +b.

Ainsi par rapport à un même repère fixe la date de début du régime post-production est la même :





donc pour DDRP = t et DDRP = $t + \delta t$ nous avons exactement les mêmes opérations pendant les régimes de post-production. En conséquence les bornes et la durée du régime post-production sont constantes entre deux dates de fin de gammes.

En fait traverser une date de fin de gamme revient à différer une gamme vers la fin de la production et donc à changer les opérations du post-production ■

Lemme IV.4

La durée optimale du régime transitoire est ***croissante par intervalles.***

Démonstration :

le résultat découle directement des lemmes 1 et 3. En effet, la durée du régime transitoire est la somme des régimes post et pré-production ■

Lemme IV.5 :

Entre deux dates de fin de gamme successives le makespan est décroissant.

Démonstration :

Soit t_1 et t_2 tel que $\nexists$ une gamme i avec $t_1 \leq DFGL(i,0) \leq t_2$

d'après le lemme 1 nous avons :

$$\text{Durée pré-production}(t_1) \geq \text{Durée pré-production}(t_2) - (t_2 - t_1)$$

d'après le lemme 2 nous avons :

$$\text{Durée permanent}(t_1) = \text{Durée permanent}(t_2) + (t_2 - t_1)$$

d'après le lemme 3 nous avons :

$$\text{Durée post-production}(t_1) = \text{Durée post-production}(t_2)$$

en faisant la somme, terme à terme, des trois lignes :

$$\text{Makespan}(t_1) \geq \text{Makespan}(t_2)$$ ■

Proposition IV.1

$\exists$ une gamme i / ***le makespan optimal est atteint*** *pour DDRP = DFGL(i,0)-1*

Démonstration : Par l'absurde

supposons que $\nexists$ une gamme i / le makespan optimal est atteint pour DDRP=DFGL(i,0)-1.

Soit t_0 la date de début de régime permanent pour laquelle le makespan optimal est atteint (cette date existe puisqu'on a un ensemble discret fini et que pour un tel ensemble le min. existe et est atteint)

soit i_0 et i_1 deux gammes tels que $DFGL(i_0,0) \leq t_0 < DFGL(i_1,0)$ (deux dates de fin de gamme successives).

Comme toutes les dates sont multiples de 1 alors $t_0 \leq DFGL(i_1,0) - 1$ et il n'y pas de date de fin de gamme entre t_0 et $DFGL(i_1,0) - 1$. En conséquence, d'après le **lemme 5**

$$\text{makespan (DDRP} = DFGL(i_1,0) - 1) \leq \text{makespan (DDRP} = t_0)$$

Or makespan (DDRP = t_0) est optimal (par hypothèse) donc

$$\text{makespan (DDRP} = DFGL(i_1,0) - 1) = \text{makespan (DDRP} = t_0) = \text{makespan optimal}$$

ce qui contre dit l'hypothèse de non existence de i / le makespan optimal est atteint pour DDRP=DFGL(i,0)-1 ■

IV.2 Détermination de la commande finale de l'exemple de la thèse

Temps de cycle = 24u.t. X = 3.

Pour déterminer la borne inf. nous avons à étudier les machines critiques : M1.1, M1.2, M2 et M5 :

$$\text{Borne inf.} = \underbrace{X(q) * CT(RP_q)}_{\substack{\text{charge d'une} \\ \text{machine critique}}} + \max_{\substack{\text{machine} \\ \text{critique m}}} \left(f(m) + g(m)\right) = 3*24 + 17 = 89 \text{ u.t.}$$

$$f(m) = \min_{l \in \{1,\ldots,I_{max}\}} \left(\sum_{n=1}^{prem(GL_q(l),m)-1} Z(l,n) \right) : \begin{array}{l}\text{durées opératoires nécessaires} \\ \text{pour atteindre la machine critique m}\end{array}$$

$$g(m) = \min_{l \in \{1,\ldots,I_{max}\}} \left(\sum_{n=dern(GL_q(l),m)+1}^{NO(GL_q(l))} Z(l,n) \right) : \begin{array}{l}\text{durées opératoires nécessaires} \\ \text{pour terminer la pièce après m}\end{array}$$

m	f(m)	g(m)	f(m)+g(m)
M1.1	0	17	17
M1.2	0	15	15
M2	0	8	8
M5	12	0	12

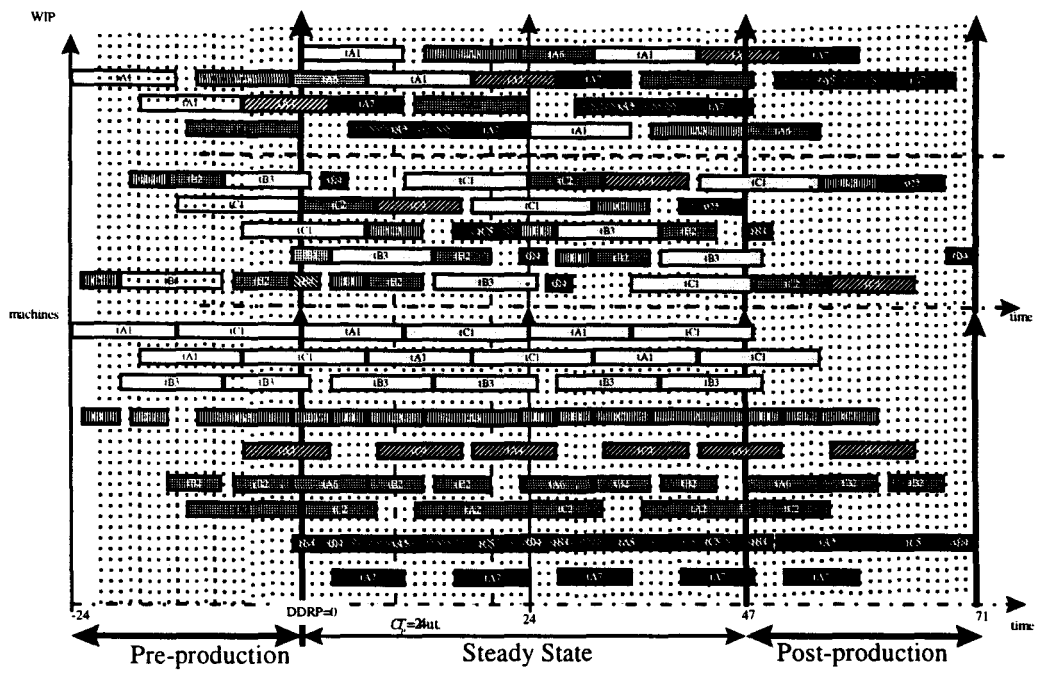
L'application de l'heuristique nécessite d'étudier l'ensemble {DFGL(i,0) / i}. L'ensemble des dates de fin de gamme {DFGL(i,0)} = {0, 2, 5, 7, 11, 17, 23} soit {DFGL(i,0) / i} = {-1, 1, 4, 6, 10, 16, 22} = {1, 4, 6, 10, 16, 22, 23}.

Le détail des différentes valeurs en fonction des dates importantes :

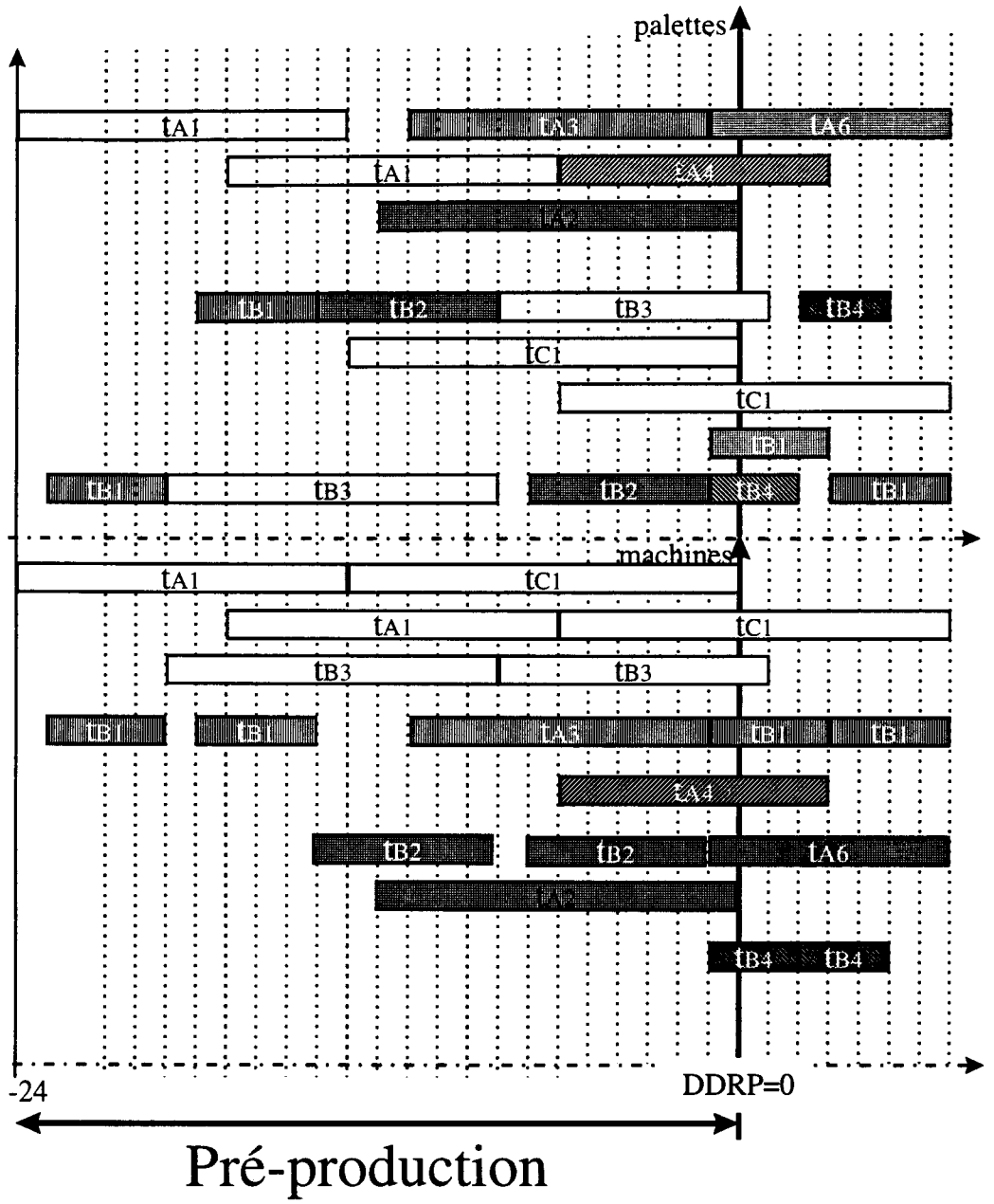
DDRP	pré	post	transitoire	permanent	makespan
0	24	24	48	47	95
1	25	24	49	46	95
2	24	26	50	46	96
4	26	26	52	44	96
5	27	26	53	43	96
6	28	26	54	42	96
7	24	24	48	48	96
10	27	24	51	45	96
11	24	24	48	48	96
16	29	24	53	43	96
17	24	29	53	43	96
22	29	29	58	38	96
23	30	29	59	37	96

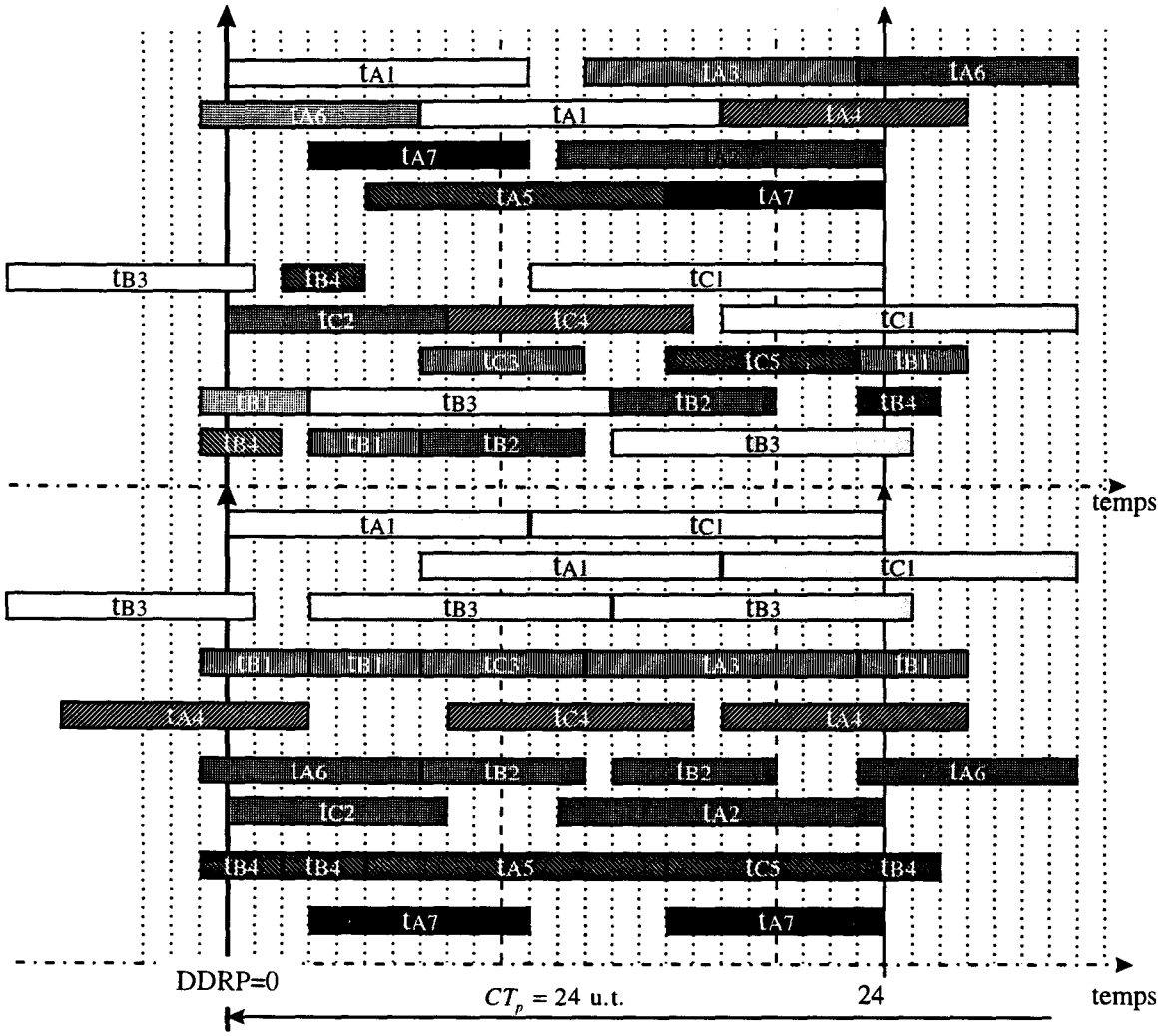
Par conséquent le makespan optimal est égal à 95 u.t. il est obtenu pour DDRP ∈ {0, 1}.

Nous donnons la commande finale correspondant à DDRP = 0 :

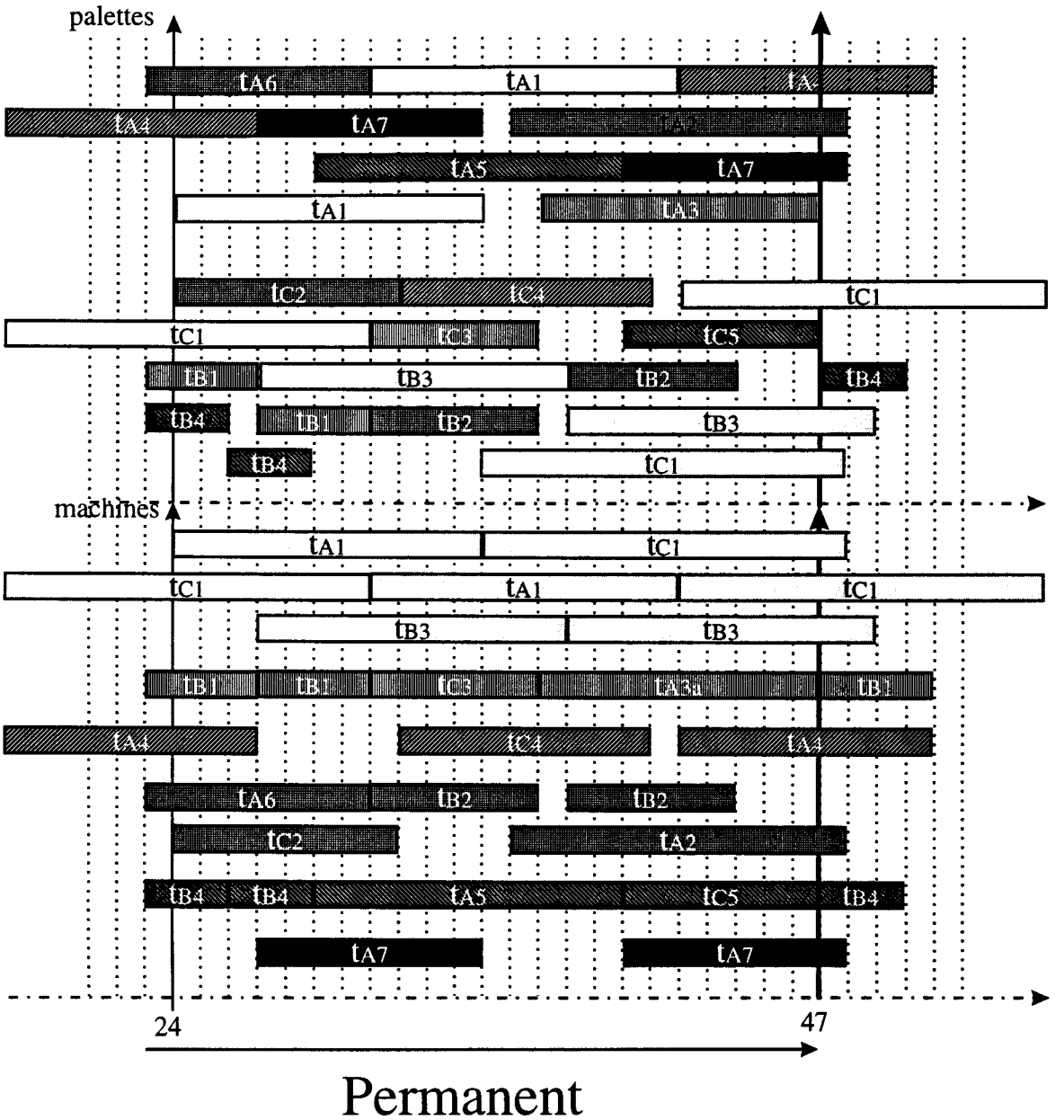


Afin de permettre une meilleur lecture de cette commande finale nous la représentons ici par parties :




tA1
tA3
tA6
tA6
tA1
tA4
tA7
tA5
tA7
tB3
tB4
tC1
tC2
tC4
tC1
tC3
tC5
tB1
tB3
tB2
tB4
tB4
tB1
tB2
tB3
temps
tA1
tC1
tA1
tC1
tB3
tB3
tB3
tB1
tB1
tC3
tA3
tB1
tA4
tC4
tA4
tA6
tB2
tB2
tA6
tC2
tA2
tB4
tB4
tA5
tC5
tB4
tA7
tA7
DDRP=0
CT_p = 24 u.t.
24
temps

Permanent


palettes
machines
24
47
Permanent