THÈSE

présentée à

L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE FLANDRES ARTOIS

Pour obtenir le titre de

DOCTEUR

Spécialité: Automatique

par

YANG Yang

ingénieur ENSEEIHT

ALLOCATION OPTIMALE DES TACHES POUR LA COOPERATION DE DEUX ROBOTS DANS UNE CELLULE FLEXIBLE D'ASSEMBLAGE

Soutenue le 8 janvier 1988 devant la commission d'examen

Membres du Jury:	P.	VIDAL	Président et Rapporteur
	P.	LOPEZ	Rapporteur
	M.	STAROSWIECKI	Directeur de recherche
	J.G.	POSTAIRE	Examinateur
	D.	JOLLY	Examinateur

A Maizhi

A mes parents

AVANT-PROPOS

Le travail de recherche présenté dans ce mémoire a été effectué au centre d'automatique de l'Université des Sciences et Techniques de Lille Flandres Artois.

Avant d'exposer cette étude, je tiens à exprimer ma profonde gratitude à Monsieur le professeur Pierre VIDAL, Directeur du centre d'Automatique, de m'avoir accueilli dans son laboratoire et d'avoir accepté la présidence du jury.

Je tiens à exprimer ma profonde reconnaissance à Monsieur le professeur Marcel STAROSWIECKI qui, par ses qualités humaines, son dynamisme, sa culture, sa rigueur dans le travail, m'a beaucoup aidé, orienté dans l'élaboration de ce mémoire. J'ai beaucoup appris tout au long de mon travail passé sous sa direction. Je l'en remercie sincèrement.

Que Monsieur Pierre LOPEZ, Professeur à l'INSA de Toulouse, accepte mes vifs remerciements pour avoir accepté de juger ce travail et de participer au jury de thèse. Je lui en suis trés reconnaissant.

J'adresse également mes remerciements à Monsieur Jack-Gérard POSTAIRE, professeur de l'Université de Lille I, pour la confiance qu'il m'a apporté en acceptant de participer à la commission d'examen.

Que Monsieur Daniel JOLLY, Maître assistant à l'Université de Lille I, accepte mes remerciements pour l'intérêt qu'il m'apporte à ce travail.

Enfin, Je remercie particulièrement Mademoiselle Mireille BAYART à qui j'exprime toute mon amitié.

J'adresse mes vifs remerciements au personnel et aux collègues du centre d'Automatique, en particulier Monsieur Bernard CEURSTEMONT, Madame Annick PIGNON pour ses nombreuses aides, et Monsieur Jean HOUZE pour la réalisation matérielle de cette thèse.

SOMMAIRE

INTRODUCTION GENERALE

L'atelier flexible est plus qu'une mode. Il a été imaginé par des précurseurs comme un moyen efficace pour introduire de profonds changements dans les usines.

On peut définir un atelier flexible comme un système de production automatisé, géré en temps réel par ordinateur et susceptible de fabriquer différents modèles(type) d'objets d'une même "famille".

Selon les personnes, l'atelier flexible est défini de façons différentes, mais l'objectif commun consiste à apporter aux entreprises qui produisent des gammes variées de pièces en quantité moyenne, les caractéristiques et les avantages des méthodes de production sur lignes transferts associées à la souplesse des machines individuelles.

De cet objectif général découlent un certain nombre de conséquences : réduction du coût de la main-d'oeuvre directe et indirecte, meilleure utilisation des équipements, réduction des en-cours, application pratique des techniques de pointe, installation progressive suivant les exigences de la production ou de la gamme, contrôle de la gestion sur l'ensemble de la production.

L'atelier flexible, à terme, sera le prolongement naturel de la conception et de la fabrication assistées par ordinateur. Ce sera alors l'assurance de créer et de fabriquer un produit au moindre coût.

Dans un atelier flexible, les décisions au coup par coup sont prises par le système. Il est fondamental de souligner qu'elles sont indépendantes de toute intervention humaine. Ces décisions multiples concernent non seulement les manutentions, mais aussi la qualité et le contrôle, le stockage des outillages et des produits, en plus des opérations classiques d'usinage.

Dans un sens, on pourrait penser qu'une ligne transfert est un atelier flexible. Mais ce serait ignorer un élément fondamental qui différencie l'atelier flexible de tout autre système de production: la possibilité d'accepter des pièces différentes, dans un ordre aléatoire. En d'autres termes, l'atelier flexible peut être conçu, si nécessaire, pour produire n'importe quelle pièce, dans un ordre quelconque.

Par définition, un atelier flexible est capable de prendre en compte simultanément des pièces différentes, avec les outils et outillages appropriés mis à disposition en temps utile sur la machine adaptée, selon un ordre établi par la gamme.

L'atelier flexible est plus qu'une technologie de groupe pilotée par un ordinateur: C'est un système de production à très haut niveau d'intégration. La fonction de l'ordinateur est de déterminer les besoins de l'atelier de production, et d'y répondre par la mise à disposition des moyens nécessaires : outils, outillages, moyens de manutention et modules de contrôle.

Sur la base des connaissances actuelles, il n' y a pas une dimension optimale pour un atelier flexible. La dimension dépend à la fois des besoins et des possibilités. Le nombre de machines peut même n'être que de 1 ou 2 . Ce noyau peut servir de point de départ pour ceux qui souhaitent une approche progressive de l'atelier flexible.

En général, le nombre de machines est plus important, de 3 à 10. L'une des plus grandes réalisations à ce jour compte 18 centres d'usinage. Il n'est pas surprenant alors que le coût des études et de l'installation puisse être très élevé[ING.83].

Dans le cadre du pôle productique, le laboratoire d'Automatique de l'U.S.T. de Lille, développe un atelier flexible expérimental dans le domaine de la confection. La particularité de cet atelier par rapport à ceux conçus dans les industries mécaniques est la manipulation d'objets non rigides et déformables, que sont les pièces de tissus, de cuir, etc...

Le système de production que nous étudions dans cette thèse est une cellule de production flexible, destinée à l'assemblage automatique à l'aide de robots. La flexibilité du système nécessite que la conduite de la cellule soit réalisée en temps réel.

L'objectif que nous nous sommes fixé concerne, d'une part la modélisation d'une telle cellule, d'autre part la conduite du système en temps réel, de manière à minimiser le temps d'exécution des tâches d'un montage.

Le mémoire est structuré de la manière suivante : au chapitre I, après quelques généralités sur les ateliers flexibles, nous présentons les problèmes relatifs à la conduite de la cellule envisagée, enfin nous décrivons le site expérimental.

Au deuxième chapitre nous présentons quelques méthodes utilisées dans les systèmes de production puisqu'on peut envisager plusieurs types de modèles pour un même système selon les buts de l'analyse. Nous envisageons trois méthodes typiques pour analyser notre cellule, puis nous choisissons les systèmes de files d'attente pour conduire la cellule dans

les chapitres suivants. Dans ce sens, nous présentons quelques connaissances sur la théorie des files d'attente.

Après avoir modélisé la cellule par un système de file d'attente, nous considérons la gestion de la cellule au troisième chapitre. La cellule flexible d'assemblage envisagée dans ce chapitre est composée d'un seul robot qui doit exécuter les n tâches, soit indépendantes, soit modélisées par un graphe potentiel-tâches. Premièrement, nous supposons que les n tâches sont indépendantes; pour minimiser le temps d'exécution de ces tâches, il faut minimiser le temps d'oisiveté du robot, ce qui conduit à commander le taux d'arrivée des ressources. Deuxièmement, à partir d'un graphe potentiel-tâches, nous considérons que les tâches exécutables, sont les clients du système de file d'attente. Dans ce cas, pour minimiser le temps d'exécution des n tâches, il faut choisir une tâche optimale parmi les m tâches exécutables à l'instant où le robot est libre. Chaque tâche exécutable réalisée apporte une certaine quantité de travail, on l'appelle le gain du système. La tâche prioritaire à chaque instant t est celle qui apporte le gain maximal.

Nous exposons enfin dans le quatrième chapitre, le cas de deux robots qui travaillent en coopération. La coopération de deux robots que nous envisageons se traduit par le fait qu'une partie des tâches (les tâches communes) peut être exécutée par l'un ou l'autre des deux robots. Dans le système de file d'attente modélisé, on a alors un problème d'affectation dynamique des tâches communes, de façon à minimiser les temps d'oisiveté des robots. Dans ce sens, nous analysons d'abord la probabilité d'affectation des tâches exécutables dans le cas stationnaire en supposant que le modèle est M/G/2. Nous utilisons ensuite les méthodes de S.E.P. (séparation et évaluation progressive) ou la programmation dynamique pour affecter les tâches communes en temps réel. Dans la dernière partie, nous affectons directement les tâches communes aux deux files en utilisant les résultats du chapitre III. Enfin, nous considérons les problèmes de conflit du système.

CHAPITRE I

PRESENTATION DES SYSTEMES DE PRODUCTION FLEXIBLES

1 - INTRODUCTION

L'atelier flexible est un système de production qui répond au besoin de fabrication de produits diversifiés: la production unitaire, la production petite et moyenne série qui est nouvellement envisagée. Ce type de production nécessite la réalisation du meilleur compromis possible entre les deux objectifs a priori antinomiques que sont la productivité et la flexibilité. L'objectif de productivité est relativement facile à définir: le but est d'obtenir, à partir d'un atelier donné, la productivité maximale. Ceci nécessite bien évidemment une automatisation poussée. Il faut en fait noter qu'il n'y a pas "une flexibilité" mais plusieurs; on peut citer, entre autres, les critères de flexibilité suivants:

- faculté de traiter simultanément plusieurs produits,
- souplesse d'adaptation à de nouveaux produits: flexibilité atelier/produits,
- faculté des machines à traiter des tâches différentes: flexibilité machines/tâches,
- possibilité d'effectuer une même tâche sur plusieurs machines: flexibilité tâches/machines.

Cette liste est loin d'être exhaustive. Des critères divers peuvent être donnés en fonction de la demande envisagée.

D'après une étude faite dans [DUP.82], sur les systèmes de production flexibles réalisés dans plusieurs pays, il ressort trois grands axes de développement:

- modules ou cellules flexibles,
- lignes transfert flexibles,
- systèmes flexibles où l'emplacement des moyens de production n'obéit pas à une structure linéaire.

Si on s'intéresse à une classification des ateliers flexibles, on peut encore distinguer l'atelier flexible d'usinage et l'atelier flexible d'assemblage. Pour chacune de ces classes d'ateliers flexibles, on peut préciser le mode de production; si toutes les pièces ont des cheminements (ordre de passage sur les différentes machines) identiques, l'atelier est dit de type "flowshop", c'est-à-dire atelier à flot; si les cheminements sont particuliers à chaque type de pièces, il est dit de type "jobshop", c'est-à-dire atelier à tâches.

Un atelier flexible est un système de production comportant:

- un ensemble de stations ou postes de travail: centre d'usinage, machine de contrôle, station d'assemblage, poste de chargement, retournement, déchargement...

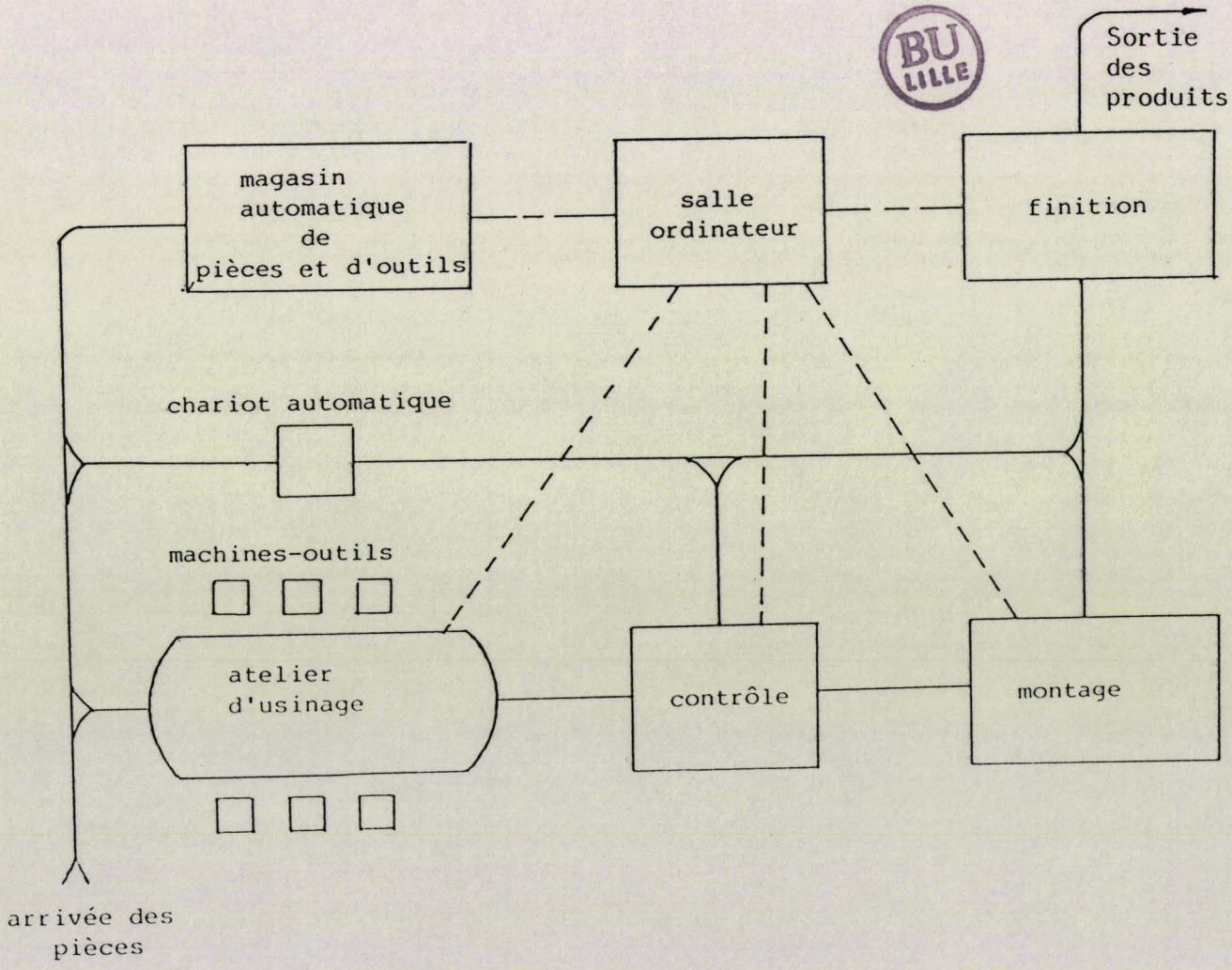
- un réseau de transport automatisé: chariots filoguidés, tapis roulants...

- des aires de stockage pour les pièces en attente d'un traitement (en cours): stock centralisé et/ou réparti.

- des ressources en outillage: un magasin central d'outils et/ou des magasins propres à chaque machine.

- un système informatique destiné au contrôle et à la commande de l'atelier flexible. Il devra réaliser la commande informatique de tout le système, mais il aura aussi à sa charge la conduite de la production encore appelée gestion de l'atelier flexible.

Un schéma classique d'atelier flexible est donné par la figure 1.1.



un schéma classique d'atelier flexible

fig.1.1

En résumé, un atelier flexible peut se décomposer en trois éléments: [BON.85]

- un système de fabrication,

- un système de manutention,

- un système de pilotage.

La combinaison de ces trois facteurs en constitue la base. En analysant ce principe et en le prenant à la lettre on peut arriver à une quantité de solutions pratiquement infinie. Depuis quelques années le vocable d'"atelier flexible" a connu un succès de presse considérable. Cela peut donner à penser au grand public que c'est une invention récente et déjà très répandue. Par ailleurs, l'absence de définition précise permet actuellement de couvrir une quantité impressionnante d'applications. Le moindre chargeur automatique sur une machine, la transforme immédiatement en cellule flexible. Le moindre système de manutention un tant soit peu automatisé métamorphose une série de moyens de fabrication traditionnels en atelier automatisé flexible.

Pour en revenir au réel, la première caractéristique d'un tel atelier est une capacité à produire des pièces différentes, par opposition aux machines spéciales ou transferts qui ne peuvent exécuter que des produits identiques dans des limites fixées au préalable.

Il existe en effet une multitude d'ateliers de petite série qui fabriquent des produits très divers. La moyenne série est aussi un domaine dans lequel avec plus ou moins de bonheur on trouve des unités réalisant des pièces très diverses. Même en grande série un certain niveau de flexibilité a toujours été recherché puisque généralement les matériels utilisés sont capables de produire des variantes d'un même ensemble, ou d'un même objet.

Tout moyen de production quel que soit la gamme de travail qu'il exécute doit remplir les fonctions suivantes[GRO.81] :

- Fabriquer: usiner, emboutir, souder, peindre, assembler, forger, laver,etc...

- Manipuler: placer les pièces en position de travail avec la précision nécessaire à la fabrication, qui dépend bien sûr du type d'opération à réaliser; en fin de tâche, replacer le produit sur le système de manutention.

- Manutentionner: transporter une pièce d'un poste au suivant, au stockage ou vers l'atelier aval.

- Stocker: les pièces étant différentes, leurs temps de fabrication ne seront pas identiques, donc le flux ne pourra être rendu pseudocontinu qu'en contrôlant un tampon de pièces interne à l'atelier.

- Contrôler: vérifier le respect de la qualité du produit, avoir une possibilité de stratégie de contrôle variable entre le prélèvement et le 100% . Ceci afin d'assurer le retour d'informations sur la qualité qui permettra une amélioration constante.

- Piloter la production: diriger chaque pièce vers le poste de travail ou le magasin qui lui convient , en fonction de son état d'avancement dans la gamme, de la disponibilité des machines, de son niveau de priorité, etc...

L'atelier flexible peut revêtir des formes et des destinations très variées depuis la production en forte cadence jusqu'aux systèmes réalisant des pièces prototypes.

Le problème de fabrication est excessivement vaste et la solution n'est pas unique. Il n'existe donc pas un atelier flexible, mais un concept de flexibilité.

Il reste à définir l'objectif de production d'un tel système. L'objectif du système de conduite d'un atelier flexible peut se définir de la façon suivante: maximiser la productivité tout en assurant des ratios de production entre les différents types de pièces. Cette définition d'un atelier flexible peut paraître quelque peu restrictive. On pourra plus généralement parler de système flexible de production pour un système qui diffère sensiblement de la définition proposée.

2 - PRESENTATION DE LA CELLULE DE PRODUCTION FLEXIBLE ENVISAGEE

Dans le cadre du pôle productique régional du Nord-Pas-de-Calais, le laboratoire d'Automatique de l'U.S.T de Lille, développe un atelier flexible expérimental dans le domaine de la confection.

Cet atelier est composé de plusieurs postes de travail interconnectés de manière à prendre en charge tout le processus de confection:

- alimentation et inspection en continu de la matière première;

- découpe en continu selon un placement dynamique, tenant compte de l'inspection;

- préhension et stockage des pièces;

- assemblage automatique à l'aide de robots;

- couture et évacuation.

Chaque poste de travail soulève un certain nombre de problèmes théoriques et pratiques qui occupent actuellement plusieurs chercheurs du centre d'Automatique, notamment l'étude d'un système de préhension de pièces déformables, le développement d'algorithmes de placement avant la découpe, la coopération entre robots en vue de l'assemblage des pièces et l'étude d'un système de supervision qui permet par le biais d'un réseau local de communication, de coordonner les différentes fonctions dévolues à chacun de ces postes de travail.

Notre cellule de fabrication intervient dans cet atelier flexible, au niveau des opérations d'assemblage.

Le but de notre étude est la modélisation et la gestion d'une cellule flexible d'assemblage.

Les moyens de production utilisés dans la cellule de fabrication sont des robots. Les robots doivent coopérer sur un même poste de travail, la cellule comprend alors deux ou plusieurs robots. Dans ce sens, il convient de développer un moniteur de coopération qui sera chargé de gérer les affectations des tâches aux robots, en fonction de l'environnement et des ré-ordonnancements décidés par le système de gestion de production.

2.1 - Problèmes posés sur la cellule envisagée

La mise en place d'une cellule flexible soulève une multitude de problèmes de toutes sortes.

Il faut tout d'abord définir l'approvisionnement en ressources, c'est-à-dire la préparation par un robot des ressources nécessaires à l'exécution d'une opération par un autre robot. Le problème posé est relatif à l'allocation dynamique des ressources. En effet, l'arrivée des pièces sur un tapis roulant crée un stock dynamique. De ce fait, nous risquons de perdre des pièces qui ne peuvent être utilisées immédiatement dans le montage, et du temps puisqu'il faudra attendre qu'une pièce, directement utilisable, soit en position de prise, pour activer les robots, on en déduit la nécessité de créer une zone qui permettra au robot de stocker les pièces qui seront utilisées ultérieurement. Une zone de stockage pourrait être associée à chacun des robots, pour une meilleure coopération, nous plaçons le stock en zone commune afin que les pièces soient stockées et directement accessibles par l'un ou l'autre des robots.

La création de ce stock nécessite la résolution du problème d'écoulement de stock (priorité entre le stock statique et le stock dynamique, par exemple) de la limitation de la zone de stockage et des critères d'organisation du stock (pas n fois la même pièce).

L'horizon de stockage et la taille du stock dépendent du temps, et tiennent compte du type de production, sur cycle ou par flux, et des éventuels changements de production.

L'ordonnancement des tâches est le deuxième problème posé par la réalisation d'une cellule flexible. Le problème d'ordonnancement des tâches est en fait, un problème d'optimisation dans lequel il s'agit de réaliser une action, en optimisant un ou plusieurs critères donnés, sous le respect de certaines contraintes.

Si pour les ateliers dits "classiques" où l'ordonnancement est statique, des méthodes analytiques peuvent être appliquées, il en est autrement des ateliers flexibles, où l'ordonnancement ne peut être que dynamique.

Les approches étudiées pour développer l'ordonnancemment dynamique sont:

- la simulation du système de production. [DAL.82] [ARA.84]

- les heuristiques,[SOU.82][SIM.84]

- les méthodes analytiques dans lesquelles nous regroupons les méthodes déterministes combinatoires, les méthodes fondées sur la théorie des files d'attente, etc...[DUM.74] [AMM.85][LEV.82][DUP.83][KUS.85].

Le problème que l'on étudie concerne une cellule de production flexible, qui nécessite obligatoirement un ordonnancement ou une affectation dynamique.

Dans le cas où deux robots ou plus travaillent en coopération, les problèmes soulevés sont:

- l'allocation dynamique des tâches et des ressources,

- le contrôle et le diagnostic.

L'affectation des tâches aux robots, des ressources aux tâches (pièces à assembler par exemple), ou aux robots (outils partageables par les robots) est un problème essentiel dans la conduite d'une cellule de production. C'est de la manière dont il est résolu que dépendent les performances globales de la cellule.

Il s'agit le plus souvent du problème d'optimisation d'un coût, d'un délai de réalisation ou d'un autre critère. Ceci conduit dans la plupart des cas à la recherche d'une maximisation du parallélisme dans la réalisation des tâches.

La coopération entre robots dans une cellule de production exige un contrôle strict de l'état des divers éléments évoluant dans cette cellule, afin de permettre le bon déroulement des opérations.

Suivant le résultat du contrôle, un diagnostic doit pouvoir être établi afin qu'une décision soit prise le plus rapidement possible et ainsi éviter que d'autres décisions soient fondées sur une base de données incohérentes, parce qu'elle n'a pas été remise à jour à temps.

D'autres problèmes peuvent encore être cités, mais nous nous limitons à l'exposé de ceux qui nous semblent être les principaux.

2.2 - Système de conduite

Un système de conduite est composé d'un système d'information et d'un système de décision.

Le système d'information représente l'état de l'unité d'assemblage. Le système de décision représente un ensemble de modules qui sont activés lorsqu'il y a un changement d'état dans l'unité d'assemblage.

Le système est conçu pour permettre au personnel chargé de la mise en oeuvre de l'unité d'assemblage de programmer une série de tâches sans se soucier de la fonction, décision, optimisation. Il ne s'agit plus alors que de décrire les travaux à effectuer, le système prend en charge la commande des robots et l'optimisation de leur fonctionnement.

L'objectif que nous voulons atteindre est la minimisation du temps d'exécution d'un montage.

2.2.1 - Description des tâches

Une tâche est une action élémentaire de l'assemblage. Dans notre cas, une tâche est aussi une action élémentaire du robot ("pick and place" par exemple).

A. la gamme d'assemblage

L'ensemble des tâches d'un montage compose la gamme d'assemblage. La gamme d'assemblage est représentée sous forme de graphe qui est composé de noeuds et d'arcs.

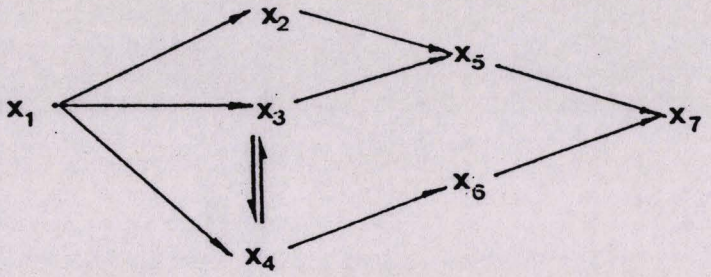
Les noeuds représentent des tâches élémentaires.

Les arcs définissent les relations entre les noeuds. Celles-ci sont de deux types:

relation d'ordre arc simple

relation d'équivalence arc double

exemple:



Dans cet exemple, x_2 peut être réalisée lorsque x_1 est terminée. x_3 et x_4 nécessitent aussi que x_1 soit terminée, mais les tâche x_3 et x_4 sont liées par certaines relations de synchronisation.

On a donc

$$G=\{X,U\}$$

avec $X=\{x_1,x_2,x_3,\ldots x_N\}$ ensemble des actions élémentaires de l'assemblage.

$$U=\{(x_i,x_j)/x_i R\ x_j\}$$

$$V=\{(x_i,x_j)/x_i R' x_j\}$$

avec R un préordre et R' un préordre symétrique.

- x_i R x_j signifie que x_j est successeur de x_i
  x_i est antécédent de x_j

- Dans le cas des tâches liées, on aura:

  x_i R x_j et x_j R x_i

On note cette relation d'équivalence:

x_i R' x_j

B. caractéristiques des tâches [DJE.86]

L'évolution dans le graphe qui correspond à la réalisation d'une tâche est soumise à deux contraintes:

contraintes de précédence et/ou synchronisation,

contraintes de ressources.

Les ressources nécessaires à la réalisation d'une tâche sont les suivantes:

les robots qui réalisent la tâche,

la matière première,

l'èspace commun, géré comme une ressource commune.

Les tâches peuvent se trouver dans l'un des états suivants:

Tâche bloquée: la tâche appartient à un assemblage qui n'est pas en cours de fabrication,

Tâche non exécutable: tous les antécédents de la tâche ne sont pas réalisés, et toutes ses ressources propres ne sont pas disponibles,

Tâche pré-exécutable(a): une tâche est pré-exécutable par rapport aux antécédents si l'ensemble des antécédents de cette tâche sont réalisés.

Tâche pré-exécutable(r): une tâche est pré-exécutable par rapport aux ressources si la(les) ressource qui peut être consommée par cette tâche est présente dans le système.

Tâche exécutable: les deux conditions suivantes sont vérifiées:

- antécédents réalisés;
  et
- ressources propres disponibles,

Tâche en cours: la tâche est en cours d'exécution, les ressources partageables dont elle a besoin sont reservées suivant une certaine planification.

Tâche terminée: la tâche est correctement réalisée.

2.2.2 - DESCRIPTION DE L'ENVIRONNEMENT

L'environnement des robots est constitué par les deux stocks, statique et dynamique, ainsi que par la zone commune.

1. Etat du stock statique

Le stock statique est représenté par une liste qui comprend l'ensemble des pièces présentes sur le stock, ainsi que leurs caractéristiques, par exemple, des pièces textiles devant composer un vêtement qui se différencient par leur type, la taille, la nature, la couleur etc... du tissus dans lequel chacune d'entre elle a été découpé, les coordonnées de position et orientation, les paramètres de prise, etc...

2. Etat du stock dynamique

Dans cette liste, nous avons les pièces en position sur le tapis, ainsi que leurs caractéristiques.

3. Etat de la zone commune

La zone commune est accessible indifféremment par l'un quelconque des robots de la cellule. Il convient de décrire son état d'occupation, cette description sera utilisée pour la génération de trajectoires sans collision.

2.2.3 - DESCRIPTION DES ROBOTS

Les robots sont considerés comme des systèmes à aptitudes multiples. A chaque instant, nous devons connaître les aptitudes de chacun des robots, l'activité ou l'état de panne, le temps d'indisponipilité (maintenance) ou d'occupation, la pince, les outils disponibles (pour un éventuel changement d'outil).

CONCLUSION

Le système d'information dispose des renseignements suivants sous forme de listes:

gamme d'assemblage
liste des tâches bloquées
liste des tâches pré-exécutables(a),(r)
liste des tâches exécutables
liste des tâches en exécution
état du stock statique
état du stock dynamique
état de la zone commune
état des robots

Le système de décision doit permettre l'ordonnancement et l'affectation des différentes tâches aux différents robots; compte tenu des informations disponibles, il optimise la décision. Le système d'information doit mettre à jour les informations compte tenu des modification et de l'évolution de l'environnement.

2.2.4 - AFFECTATION DES TACHES

L'affectation est le problème principal dans le système de décision. Il dispose de la liste des opérations exécutables sur lesquelles, il doit opérer un ordonnancement qui satisfait aux critères retenus, en vue de leur exécution par les robots.

A. Critère d'optimisation

L'affectation des tâches est réalisée suivant un critère d'optimisation.

Plusieurs critères peuvent être pris en considération:

Critère temps minimum de production:

Suivant la production à réaliser et la structure de la cellule ce critère pourra être étudié sous différentes formes.

Pour un processus seul (robot, dans notre cas), on pourra considérer un assemblage unique ou un flux de production (production par "lots").

Pour plusieurs processus, ayant plusieurs montages à réaliser, on pourra considérer l'ensemble des montages comme un assemblage. On pourra également travailler sur flux, ce dernier pouvant être constitué d'un même nombre de montages pour chacun des processus, ou d'un ensemble de montages (n montage d'un type pour m montages différents).

Critère sur les temps de travail:

En fonction des durées des tâches à réaliser, on peut répartir les charges de travail des différents robots pour obtenir une égalisation du temps de travail en minimisant l'inactivité de certains robots et la suractivité (usure) d'autres.

Le critère que nous choisissons est la minimisation du temps d'exécution d'un montage.

B.Paramètres utilisés pour l'affectation

Compte tenu du critère choisi, il existe un ensemble de paramètres qui interviennent dans l'affectation d'une tâche.

Nombre de successeurs.

A partir de la gamme d'assemblage, on associe à une tâche une ou plusieurs tâches successeurs immédiats et sa réalisation contribue à les rendre préexécutable(a). Il semble alors judicieux de choisir parmi les tâche exécutables celles dont le nombre de successeurs immédiats est plus élevé.

Un autre paramètre que nous jugeons intéressant est celui du nombre de successeurs total (immédiats et autres), dont les ressources propres sont disponibles dans la cellule (magasin ou stock intermédiaire). Ces successeurs ont l'avantage de devenir directement exécutables lorsque leurs antécédents sont réalisés.

Coèfficient de déblocage.

La réalisation de chacun des antécédents d'une tâche (si elle en a plusieurs), contribue à la rendre préexécutable(a). On associe ainsi, à chacun de ces antécédents, un coefficient dit de déblocage. Il dépend de l'état, réalisé ou non des antécédents de la tâche et prend sa valeur maximale lorsqu'il reste un seul antécédent non-réalisé.

Temps d'exécution d'une tâche

Le temps d'exécution d'une tâche peut varier suivant le robot qui réalise cette tâche,l'exécution de la tâche étant liée au déplacement et à un éventuel changement d'outil.

Fréquence d'arrivée des pièces

La féquence d'apparition des pièces, qui est liée à la production amont, nous permettra de choisir une tâche de stockage à la place d'une tache de montage, lorsqu'une pièce "rare" sera présente en position de prise par un robot.

3 - PRESENTATION DU SITE EXPERIMENTAL

La cellule de fabrication qui a été implantée sur le site expérimental est composée d'un certain nombre d'éléments robotiques, péri-robotiques et de calcul.

a) Deux robots SCEMI 4C06

Ce sont des manipulateurs de type bras articulé rigide, à quatre degrés de liberté, correspondant à deux rotations principales du bras et à deux mouvements relatifs à l'organe terminal: une rotation de celui-ci et une translation verticale le long de son axe.

L'espace de travail de chacun des robots est asymétrique, ce qui a impliqué un aménagement particulier de leur implantation afin d'obtenir une zone de travail commune la plus grande possible.

Leur disposition est telle qu'elle offre trois plans de travail possibles, dont un est commun.

Leur programmation s'effectue à l'aide d'un langage de robotique de haut niveau, LM(langage de manipulation) [MAZ.81][MIR.84] développé à l'IMAG.

b) Un système de vision VISIOMAT

C'est un système développé par MATRA, qui utilise un langage de programmation de haut niveau, LV (langage de vision).

Il comprend un numériseur d'images, un extracteur de contours, un processeur de commande, construit autour d'un microprocesseur 16 bits.

c) Un tapis roulant

Il est linéaire et accessible aux deux robots. Des cellules photo-électriques sont installées sur son support afin de signaler la présence d'un objet devant les robots ou sous la caméra.

d) Calculateur: HP 1000

L'ensemble des éléments de la cellule est géré par l'intermédiaire d'un calculateur HP1000 de type A600, construit autour du microprocesseur 2801.

Il travaille sous le système d'exploitation multi-tâches temps réel RTE.A.

L'organisation physique de la cellule de fabrication doit être telle que les différents problèmes cités et qui sont inhérents à la coopération entre les robots, puissent apparaître et donc être étudiés.

La structure que nous avons choisie est celle représentée par le figure 1.2.

Cette organisation offre de multiples possibilités d'expérimentation, tout en autorisant une augmentation croissante de la difficulté.

En dehors de l'atelier expérimental dans lequel elle est située, la cellule peut s'intégrer facilement dans d'autres structures plus importantes, comme par exemple une chaîne de production modulaire.

Elle illustre donc parfaitement une réalité industrielle dont on observe la naissance actuellement.

Le but de cette cellule est de permettre la fabrication d'un ou de plusieurs produits simultanément. Ceux-ci nécessitent généralement différentes opérations (assemblage, collage,...) qui doivent obligatoiremant être compatibles sur un même site.

Les pièces ou la matière première nécessaires sont supposées produites en amont de la cellule et arrivent de façon aléatoire sur un convoyeur.

4 - CONCLUSION

L'étude de la conduite d'une cellule de production flexible utilisant les robots engendre un certain nombre de problèmes dont les principaux ont été exposés.

L'essentiel du problème concerne la conduite de la cellule, c'est-à-dire, l'ordonnancement dynamique des tâches et l'allocation des tâches aux robots.

Pour cela, on a besoin de méthodes et d'outils permettant une analyse quantitative et rationnelle du comportement des systèmes automatisés de production.

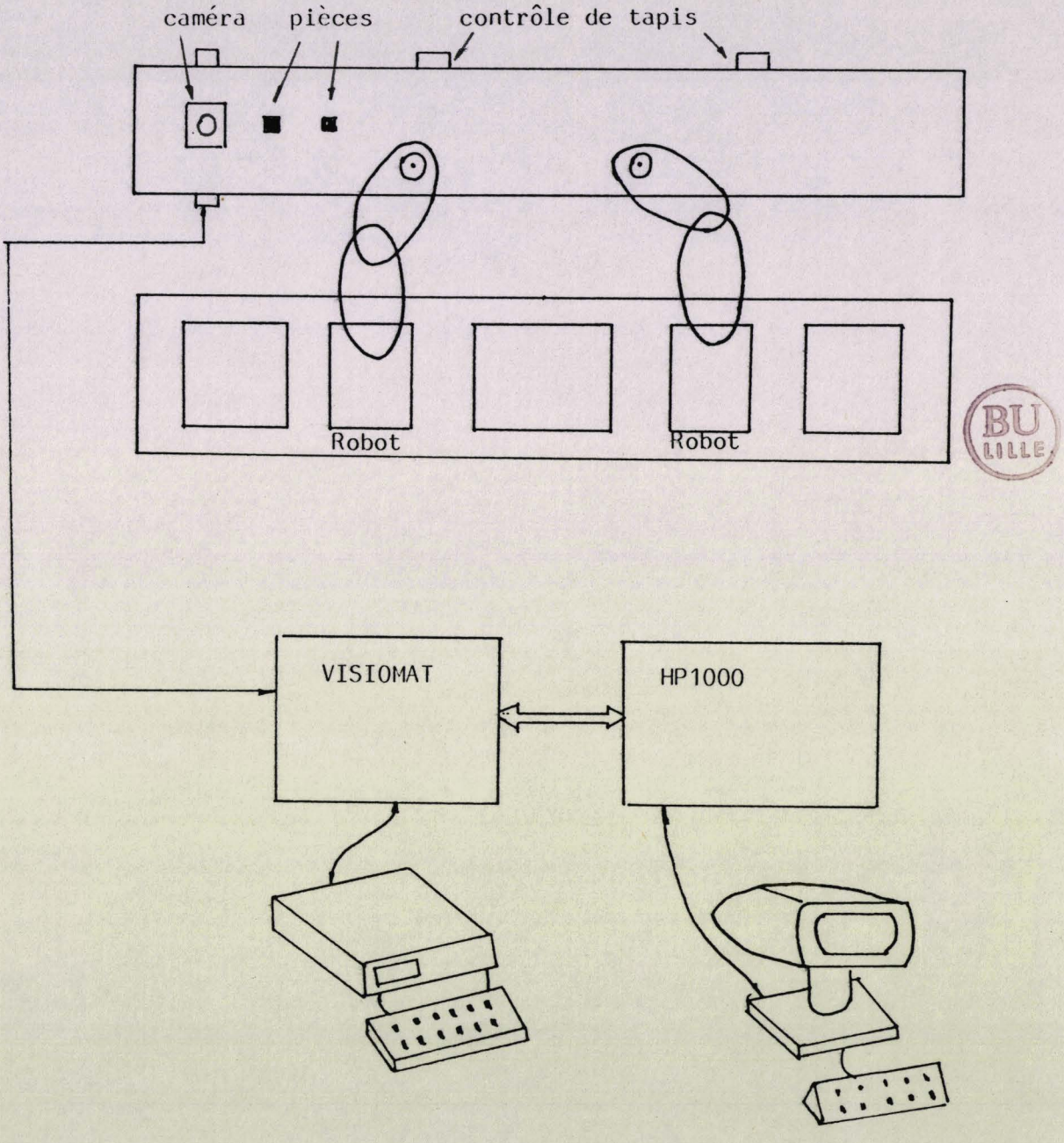


fig.1.2

CHAPITRE II

MODELISATION DES SYSTEMES DE PRODUCTION FLEXIBLES

1 - INTRODUCTION [BEL.85][BOU.86]

Les études d'aide à la conception et au pilotage des systèmes automatisés de production font, en général, appel à des modèles. Ces modèles sont des représentations du comportement dynamique du système réel.

Pour un même système, on peut envisager plusieurs types de modèles selon les préoccupations de l'étude, l'état d'avancement de celle-ci et les degrés de finesse souhaités pour la représentation. On a pu ainsi évaluer plus clairement leur pouvoir descriptif et leur applicabilité pour l'étude des systèmes automatisés de production. On assure ainsi leur cohérence réciproque et on facilite leur utilisation aux différents stades d'avancement de l'étude.

Pour tous ces modèles, le système est composé d'objets (ou entités ou ressources) caractérisés par des attributs dont certains peuvent être variables dans le temps. les valeurs de ces attributs variables définissent l'état de l'objet et donc du système. Pendant tout intervalle de temps ou l'état ne varie pas, on dit qu'il est engagé dans une activité. Au contraire, un événement a lieu à l'instant ou cet état varie. Un objet ne peut donc être engagé que dans une seule activité. C'est pourquoi au lieu d'objet, on peut aussi parler de ressources.

Pour décrire la dynamique du système, il faut ensuite représenter les mécanismes qui vont gouverner les changements d'état. Pour les systèmes de production automatisés, ces mécanismes sont de deux types:

- Une partie d'entre eux est dictée par des contraintes technologiques (gamme d'usinage par exemple). On peut parler de "règles opératoires".

- Les autres sont les règles de gestion et de pilotage dont le choix va déterminer les performances du système en productivité grâce à un meilleur contrôle du flux de production.

Dans ce chapitre, nous présenterons les divers types de modèles,et certaines méthodes d'optimisation utilisées dans les systèmes de production. Dans ce sens, nous rappelons quelques éléments de la théorie de files d'attente, qui seront utilisés dans les chapitres suivants. Enfin, nous présenterons la modélisation de la cellule d'assemblage envisagée.

2 - MODELES POUR LA DESCRIPTION ET L'ANALYSE DES SYSTEMES DE PRODUCTION

Le pilotage d'un système de production consiste généralement à générer une suite de décisions pour résoudre les problèmes d'ordonnancement et les problèmes d'affectation. De nombreux modèles ont été étudiés dans la littérature pour décrire et analyser un système de production.

Modèles déterministes

- Théorie des graphes: CPM, PERT, MPM,

- Modèles algébriques

  Algèbre des Dioides

  Modélisation pour optimisation: algèbre linéaire,etc

- Modèles à événements discrets

  Réseaux de Pétri, etc

Modèles stochastiques

- Processus de Markov

- La théorie des files d'attente...

Les approches utilisées dépendent des systèmes de production et des hypothèses retenues pour leur description:

- les temps opératoires, connus ou aléatoires,
- l'arrivée des pièces, de façon déterministe ou aléatoire,
- la définition du modèle, continu ou discret,
- la possibilité de stockage intermédiaire, fini ou infini.
- etc...

Pour un même système automatisé de production, on peut ainsi envisager plusieurs approches.

2.1 Modèles déterministes [MAR.78]

Si on suppose que le séquencement des différentes opérations à réaliser est connu avant le début d'exécution des tâches, les modèles déterministes suivants peuvent être considérés :

2.1.1 - Théorie des graphes [DIB.70][ROY.64]

La théorie des graphes est souvent utilisée pour décrire et analyser un système de production, afin de résoudre les problèmes d'ordonnancement. La résolution d'un problème d'ordonnancement requiert de multiples implications numériques: chaque tâche du projet est caractérisée par, au moins trois types de grandeurs:

- sa durée;
- la quantité de chacun des moyens, corps de métiers, matériels, coût, requis pour sa réalisation;
- son intervalle de réalisation (début au plus tôt, fin au plus tard).

L'exécution des tâches est soumise à des contraintes pour réaliser un ordonnancement. Les principaux types de contraintes sont donnés dans [ROY.64] [CAR.82]. Ce sont les contraintes de type potentiel, de type disjonctif, et enfin de type cumulatif.

1) contraintes potentielles

Elles se présentent sous deux formes d'une part les contraintes de localisation temporelle, d'autre part les contraintes de succession.

Les contraintes de localisation temporelle signifient, en pratique, qu'une tâche i ne peut débuter avant une date fixée l (date de début au plus tôt).

$$t_i \geq l$$

Une autre contrainte de ce type peut concerner la date limite avant laquelle une tâche doit être terminée (date de fin au plus tard).

Une contrainte de succession entre les tâches i et j signifie que la tâche j ne peut débuter avant la fin de la tâche i, on aura:

$$t_j \geq t_i + d_i \qquad d_i : \text{ la durée de la tâche i.}$$

2) contraintes disjonctives

Considérons deux tâches i et j, et supposons que leur exécution nécessite l'emploi d'un matériel unique. Les deux tâches ne pourront donc être réalisées en même temps; en d'autres termes, les intervalles de temps:

$$(t_i, t_i+d_i) \text{ et } (t_j, t_j+d_j)$$

ne peuvent avoir de partie commune.

En général, ces contraintes portent sur un assez grand nombre de couple de tâches prises parmi celles qu'un matériel doit accomplir. Dans la mesure où l'ordre de succession de ces dernières est l'une des inconnues du problème, il s'avère impossible d'utiliser la remarque précédente pour éliminer les contraintes disjonctives.

3) contraintes cumulatives

Soit K un moyen quelconque(corps de métiers, matériel, financement, matière première...); soit $\Gamma_K(t)$ la quantité maximale disponible de moyen K à l'instant t (t étant compris entre le début et la fin de la réalisation du projet); soit δ_{ik} la quantité du moyen K requis pour l'exécution de la tâche i; à chaque instant t, le cumul des quantités δ_{ik} du moyen K, requises pour l'exécution des tâches i en cours, doit être inférieur ou égal à la quantité maximale disponible $\Gamma_K(t)$. Ce que l'on peut écrire sous la forme:

$$\delta_{ik} \leq \Gamma_K(t) \qquad \forall t$$

Ce type de contraintes est, en général, plus nuancé que les contraintes potentielles. Il est souvent possible, en cas de besoin, d'avoir recours à des heures supplémentaires, de sous-traiter certaines opérations de tirer parti d'une polyvalence possible de la main d'oeuvre ou du matériel.

Lorsque seules les contraintes potentielles sont considérées, l'ordonnancement des tâches minimisant le délai total d'exécution est facilement déterminé par les méthodes CPM/PERT ou MPM . Celles-ci sont basées sur la recherche d'un chemin de valeur maximale dans un graphe évalué.

Conclusion: La théorie des graphes est une méthode déterministe pour résoudre les problèmes d'ordonnancement dans le système de production. Pour un ensemble de tâches parfaitement déterminé, on peut le modéliser par un graphe puis calculer l'ensemble des dates de début au plus tôt ,l'ensemble des dates de début au plus tard , et les marges de ces tâches. Le calcul d'un chemin critique permet de minimiser le délai total d'exécution. Cette méthode est limitée dans la mesure où les dates au plus tôt ne sont pas connues (les matières premières arrivent de façon aléatoire, par exemple). Dans ce cas, on préfère utiliser les méthodes stochastiques pour résoudre les problèmes d'ordonnancement.

2.1.2 - Modèles algébriques

Les méthodes algébriques peuvent être utilisées pour modéliser un système de production. Parmi les méthodes utilisées, citons l'algèbre des Dioides et l'algèbre linéaire.

a) Algèbre des Dioides

Un processus de production peut être modélisé par l'algèbre des Dioides, en général, sous les conditions suivantes.

1) les diverses tâches peuvent être obtenues à l'aide un graphe potentiel-tâches [ROY.70].

2) l'ensemble des tâches est fini, et définit les sommets d'un graphe sans cycle.

C'est la méthode du chemin critique, qui a fondé le succès de l'analyse de type PERT. Récemment, on a pu étendre ce type d'approche à des graphes orientés avec circuits, et analyser ainsi des processus de production répétitifs, tels qu'on peut les rencontrer dans certains ateliers flexibles .

On considère un système à événements discrets qui peut être décrit par des ensembles finis A et R d'activités et de ressources engagées dans ces activités, et on suppose que la logique de changement d'état est complètement déterministe. Si x_j est la date au plus tôt de début de l'activité $n^o j$, P(j) l'ensemble des activités qui précédent j, et R(j) l'ensemble des ressources qui commencent leur cycle d'activité en j; x_j est défini donc par

$$x_j=\max(\max_{i\in P(j)} x_i+p_i, \max_{r\in R(j)} u_r) \qquad (2.1)$$

où p_i est la durée de l'activité i et u_r la date de disponibilité de la ressource r.

Si on considère les vecteurs X et U de composantes x_i et u_r, on peut réécrire l'équation (2.1) sous forme matricielle.

$$X=XA \oplus UB \qquad (2.2)$$

où $\oplus$ symbolise le maximum et le produit matriciel est de type (max,+), c'est-à-dire $(M.N)_{ij}=\max_k(M_{ik}+N_{ik})$.

La matrice carrée A est définie par ses coefficients

$A_{ij}=P_i$ si $i\in P(j)$

$=-\infty$ sinon

La matrice rectangulaire B est définie par ses coefficients

$B_{ri}=0$ si $r\in R(i)$

$=-\infty$ sinon

L'équation (2.2) se résout simplement sous la forme

$$X=UBA^* \qquad (2.3)$$

où A^* contient les durées des plus longs chemins entre activités (le graphe potentiel-tâches associé à la matrice A est sans cycle). A^* s'obtient par un algorithme de plus long chemin. La connaissance du vecteur U détermine donc les valeurs des x_i.

Si on répète les activités de l'ensemble A, on note X_n le vecteur des dates au plus tôt de début des activités pour la n-ième fois; U_n contient les dates de disponibilité des ressources pour la n-ième fois (c'est-à-dire après avoir parcouru n-1 fois leur cycle d'activité). U_n s'exprime en fonction de X_{n-1} sous la forme

$$U_n=X_{n-1}C \qquad (2.4)$$

où $c_{ir}=p_i$ si i est la dernière activité effectuée à l'aide de r

$=-\infty$ sinon

On construit donc une équation matricielle récurrente sur une algèbre de type (max, +), étudiée par [GUN.79] qui est un cas particulier de structure de dioïde . Il suffit pour cela de relier (2.3) et (2.4), en éliminant X_{n-1}, sous la forme

$$U_n=U_{n-1}BA^*C$$
$$=U_{n-1}M \qquad (2.5)$$

La donnée du vecteur U_0 définit l'état initial du système.

L'étude de telles équations récurrentes est menée dans [COH.83][COH.85] . On montre que les systèmes décrits par des équations de type (2.5) ont un comportement périodique atteint après un transitoire de durée finie au sens ou il existe un entier d,un réel positif λ , et un entier n_0 tels que

$$n>n_0 \quad i\in R \ , \ x_{i,n}=x_{i,n-d}+d\lambda \qquad (2.6)$$

est analogue à une valeur propre, car on vérifie l'existence de vecteurs V tel que

$$\lambda V=VM \quad (\text{avec}(\lambda V)_i=V_i+\lambda)$$

Le régime permanent décrit par (2.6) est complètement différent d'un régime permanent stochastique. Il s'agit ici

d'un cycle limite qui se répète, tandis que le régime permanent stochastique est évalué en moyenne.

Il existe des algorithmes simples qui permettent d'obtenir les valeurs de d et λ . On peut ainsi calculer les régimes permanents d'ateliers automatisés soumis à des lancements en production périodique. Ce type de processus a été étudié dès 1960 par [CUN.60] [CUN.62].

b) Algèbre linéaire

Les problèmes de production modélisés sous la forme d'un programme linéaire sont fréquemment rencontrés. La forme générale en est la suivante:

Compte tenu des ressources disponibles (hommes, machines, matières premières, etc...) pour une période de temps fixée(une semaine, par exemple), en quelle quantité une entreprise devrait-elle manufacturer ses produits pour optimiser son objectif?

exemple de production

Soient:

X_j= quantité d'un produit j donné, fabriqué par un processus donné, dans une période de planification (il est possible que plusieurs processus partageant les mêmes facilités puissent être utilisés pour fabriquer le même produit).

T_i= temps total disponible sur les machines de type i pendant la période considérée.

B_i= quantité de matière première de type i disponible

a_{ij}= quantité de matière première de type i pour fabriquer une unité du produits j.

t_{ij}= temps d'occupation de la machine i pour la fabrication d'une unité du produit j.

et si on suppose que n produits différents peuvent être fabriqués, en utilisant m types de machines et que k types de matières premières sont requises, alors les contraintes imposées par les ressources disponibles sont:

$$\sum_{j=1}^{n} t_{ij} x_j \leq T_i \qquad i=1,\ldots m \qquad (2.7)$$

$$\sum_{j=1}^{n} a_{ij} x_j \leq B_i \qquad i=1,\ldots k \qquad (2.8)$$

En supposant que l'entreprise veut maximiser ses profits, si P_j dénote le profit, associé au produit j, alors le programme à résoudre est:

$$\max \sum_{j=1}^{n} P_j x_j \qquad x_j \geq 0 \ , \ j=1,\dots n \qquad (2.9)$$

satisfaisant les contraintes précédentes.

2.1.3 - Modèles à événements discrets

Un système peut être décrit par un modèle à événements discrets si les changements des états ne peuvent avoir lieu qu'à certains instants et de façon discontinue. C'est en particulier le cas des modèles de système de production dans lesquels on ne s'intéresse qu'aux dates de début et de fin d'un ensemble de tâches.

Pour décrire la dynamique du modèle, il faut considérer les mécanismes qui vont gouverner les changements d'état. Pour chaque événement on doit définir dans quelles activités vont s'engager les différents objets libérés par cet événement(fin de l'activité précédente). On définit ainsi des contraintes de précédence entre les différentes activités. Les "trajectoires" ou suites d'activités, décrites par les objets doivent respecter ces contraintes. Celles-ci vont introduire un ensemble de transitions possibles entre états du modèle. Divers types de modèles peuvent être obtenus selon la façon dont seront décrits les mécanismes de changements d'état.

Les réseaux de Petri [BRA.83][CHR.83] sont une méthode systématique de description de systèmes discrets qui possèdent les deux caractéristiques suivantes:

- La synchronisation de phénomènes se déroulant en parallèle;

- Le non-déterminisme, c'est-à-dire la nécessité de faire appel à des procédures de décision pour faire évoluer le système dans certaines configurations.

Les systèmes automatisés de production semblent donc tout à fait susceptibles d'être décrits à l'aide de réseaux de Petri [CAV.81][CAR.83] ([DAL.83] utilise le <GRAFCET>). Les applications traditionnelles des réseaux de Petri sont les protocoles de communication entre calculateurs et l'analyse de procédures algorithmiques. A partir des travaux de Brams[BRA.83], la modélisation d'un système à événements discrets à l'aide d'un réseau de Petri peut se faire selon les conventions naturelles suivantes [DUB.83]:

- les activités seront représentées par des transitions. la durée entre le début et la fin de l'activation de la transition correspond à la durée de l'activité;

- les jetons du réseau de Petri représenteront les objets. A un objet donné correspondront autant de jetons qu'il aura d'attributs variables;

- les places du réseau de Petri seront les conditions qui permettront à l'activité de démarrer. Pour cela il faudra que chaque place contienne un jeton.

- l'ensemble des places que peut parcourir un jeton correspond à l'ensemble des valeurs que peut prendre l'attribut de l'objet concerné.

2.2 Méthodes stochastiques

La complexité des éléments constituant un système automatisé de production donne à certaines variables un caractère aléatoire (la fréquence d'arrivée des pièces, les temps opératoires, par exemple). Dans ce sens, les méthodes stochastiques peuvent être proposées pour résoudre les problèmes d'ordonnancement, d'allocation de ressources,etc...

2.2.1- Méthode de Programmation Markovienne

La programmation Markovienne s'applique à des processus à espace d'états et de temps discrets et est basée sur le principe d'optimalité de BELLMAN.

En programmation dynamique, on résout la récurrence d'un programme en commençant par l'étape finale, en supposant un horizon fini T. On peut donc considérer qu'à un instant t donné, où le système est dans un des M états possibles, la probabilité de transition d'un état i à l'instant t, vers un état j à l'instant t+1, est donnée par:

$$P(Y_{t+1}=j/Y_t=i)=P_{ij} \qquad \text{(dans le cas stationnaire)}$$

On suppose que cela engendre un coût C_{ij} de la transition de i vers j.

On définit $V_{T-t}(i)$, comme le coût moyen engendré lorsque i est l'état de départ. Il est égal à la somme du coût de la transition vers l'état j, à l'instant t+1 et de la valeur de l'état à cet instant, c'est-à-dire $V_{T-(t+1)}(j)$.

Or le passage à l'état j n'est qu'estimé par une probabilité P_{ij}.

A partir donc de la définition de l'espérance mathématique d'une variable aléatoire discrète, on peut écrire que:

$$V_{T-t}(i)=\sum_{j=1}^{M} P_{ij}(C_{ij}+V_{T-t-1}(j)) \qquad (2.10)$$

Lorsqu'en plus, on a des décisions à prendre à chaque instant pour faire évoluer le système d'un état vers un autre, on aura un problème décisionnel Markovien.

Soit U_t, l'ensemble des décisions admissibles à l'instant t. La probabilité de transition P_{ij}, comme le coût C_{ij} dépendent maintenant de la décision u_t de U_t. On aura alors $P_{ij}(u_t)$ et $C_{ij}(u_t)$.

D'où

$$V_{T-t}(i) = \underset{u_t \; U_t}{OPt} \left\{ \sum_{j=1}^{M} P_{ij}(u_t)(C_{ij}(u_t)+V_{T-t-1}(j)) \right\} \quad (2.11)$$

Pour modéliser un système de production, cette approche a été plus particulièrement étudiée par l'institut de Contrôle et Systèmes de l'Université Technique de WROCLAW(Pologne), dans le cadre d'une collaboration avec notre Laboratoire [BUB.85][REY.84][REY.84bis].

2.2.2 - Théorie des files d'attente

Les files d'attente peuvent être utilisées comme outil de modélisation lorsque le système est stochastique. Les premiers systèmes de production étudiés par la théorie des files d'attente, font l'objet des travaux de Jackson [JAC.63], Gordon-Newell [GOR.67] et Koenigsberg [KOE.59]. C'est dans le cadre de l'analyse des performances de systèmes informatiques que la théorie des files d'attente a été la plus développée.

En général, pour modéliser un système de production par un système de files d'attente, on considère les machines et le système de transport comme les guichets, et les pièces comme les clients. L'état instantané est décrit par le nombre de clients en attente ou en service à chaque guichet.

Deux aspects peuvent être étudiés par un modèle de file d'attente:

- les performances du système (la conception du système)

- le pilotage du système

Le premier aspect a pour but d'étudier la distribution des temps d'oisiveté et d'occupation des serveurs, le nombre moyen d'objets en attente, la durée moyenne des temps d'attente, afin de planifier le système pour maximiser sa productivité.

Le deuxième aspect est l'étude de la discipline de service, de la répartition des ressources pour optimiser l'évolution du système.

Le modèle mathématique est en général de type Markovien, donc plus simple à étudier. L'ensemble des états est alors facilement identifié, et la logique de changements d'état se traduit par des probabilités de transition affectées à chaque événement possible. On suppose que ces événements ne peuvent se produire simultanément. Les probabilités de transition se calculent à partir de la connaissance des durées moyennes des activités (opérations d'usinage, de manutention...), des règles opératoires que sont les gammes d'usinage, et des règles de pilotage, qui s'expriment sous forme de discipline de service aux stations. En réalité, il est très difficile de considérer le système de production comme un modèle Markovien. Dans ce cas, les modèles que l'on utilise ne permettent de calculer que les probabilités asymptotiques des états, d'où on déduit des indices de performance moyenne sur une très longue période de fonctionnement.

Pour les modèles de type non-markovien, les deux classes d'outils suivantes sont utilisées pour modéliser et évaluer les performances d'un système:

- les méthodes de simulation

- les méthodes analytiques

En réalité, ces deux méthodes peuvent être utilisées dans tous les systèmes de file d'attente. La méthode de simulation la plus utilisée est la méthode de Monte-carlo. C'est une méthode statistique, qui permet d'obtenir la performance du système, le nombre moyen de clients, le temps moyen d'attente, etc, à l'aide de la simulation.

Les méthodes analytiques employées dans l'étude des systèmes de production, utilisent l'analyse opérationnelle, introduite par Denning et al [DEN.77] et Byzen et al [BUZ.80] et développée par Dallerz [DAL.84].

De plus en plus, les travaux récents sur le pilotage des systèmes ont pour objectif d'optimiser le système en temps réel. Dans la théorie des files d'attente, par opposition aux modèles statiques, on développe les modèles de contrôle, à l'aide des méthodes de recherche opérationnelle, de programmation dynamique, des graphes, et d'autre méthodes dynamiques, pour résoudre les problèmes de discipline de service, d'admission des clients et d'affectation [HAR.75] [STI.80][STI.85].

3 - LES FILES D'ATTENTE [KAU.59][TAK.62][GNE.68][BUD.77]

Les files d'attente ou queues sont des phénomènes familiers que nous observons très fréquemment dans notre activité personnelle; mais on les rencontre aussi dans de nombreux problèmes économiques, militaires, sociaux, etc . La figure 2.1 présente un modèle général de file d'attente. L'ensemble des files d'attente et des stations constitue le système d'attente ou système, représenté dans le cadre pointillé de la figure 2.1.

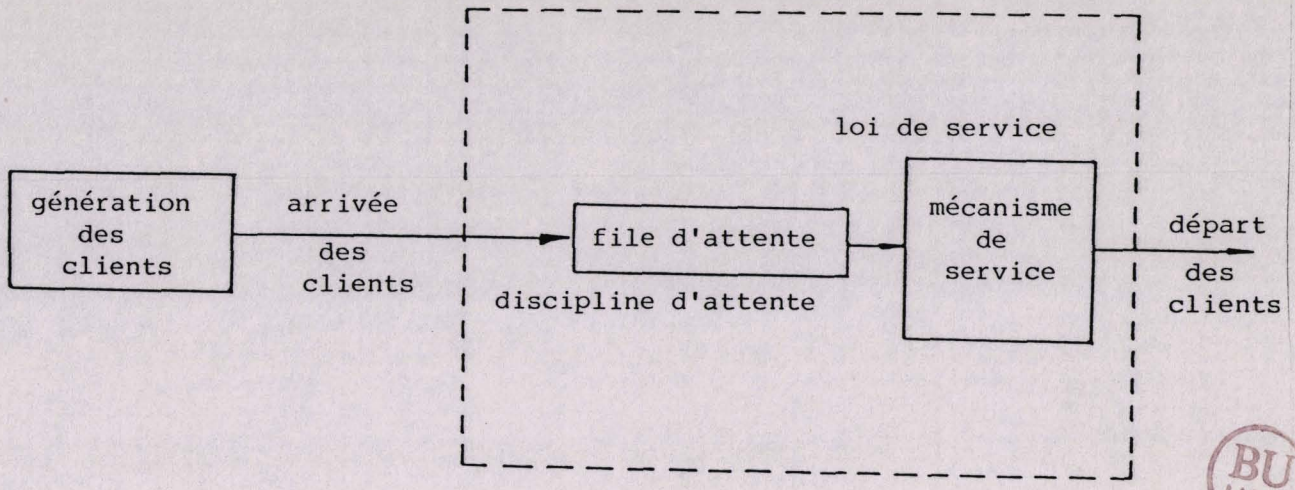


fig.2.1

Présentons plus précisément les trois parties constitutives d'un système de files d'attente.

3.1 - Processus des arrivées de clients

Les arrivées des clients peuvent être:

a) séparées par des intervalles de temps égaux;

b) séparées par des intervalles de temps inégaux mais déterminés;

c) séparées par des intervalles de temps inégaux connus en probabilité; on dit alors que les intervalles sont aléatoires.

Pour le dernier cas, il existe plusieurs modèles différents. Nous supposons toujours que le temps t est dans l'intervalle $[0,\infty)$, $\tau_1, \tau_2, \cdots \tau_n \cdots$ sont les instants d'arrivée des clients. Les intervalles de temps $Q_n = \tau_{n+1} - \tau_n$ $(n=0,1,\cdots, \tau_0=0)$ sont indépendants mutuellement et sont des variables aléatoires positives avec la fonction de distribution:

$$P\{Q_n \leq x\} = F(x) \qquad (n=1,2,\ldots)$$

En général la variable aléatoire $Q_0 = \tau_1$ peut avoir une fonction de distribution

$$P\{\tau_1 \leq x\} = \hat{F}(x)$$

qui est différente de F(x). Généralement $\hat{F}(x)$ dépend de l'état initial du processus. Dans ce cas, on dit que le processus d'entrée est récurrent.

Dans le cas où on suppose que $F(x)=\hat{F}(x)$, les processus suivants peuvent être considérés:

$$\text{i)} \quad F(x)=\begin{cases} 1-\exp(-\lambda x) & \text{si } x\geq 0 \\ 0 & \text{si } x<0 \end{cases} \qquad (2.12)$$

On dit que $\{\tau_n\}$ est un processus de Poisson.

$$\text{ii)} \quad F(x)=\begin{cases} 1-\sum_{j=0}^{m-1} \exp(-\lambda x)\,\frac{(\lambda x)^j}{j!} & \text{si } x\geq 0 \\ 0 & \text{si } x<0 \end{cases} \qquad (2.13)$$

Dans ce cas, $\{\tau_n\}$ est un processus d'Erlang.

iii) Si $F(x)$ est arbitraire, on dit que $\{\tau_n\}$ est un processus de Palm.

iv) Si on suppose que le temps moyen d'intervalle $\beta=\int_0^\infty x\,dF(x)$ est défini et $\hat{F}(x)=F^*(x)$

où
$$F^*(x)=\begin{cases} \frac{1}{\beta}\int_0 [1-F(y)]dy & \text{si } x\geq 0 \\ 0 & \text{si } x<0 \end{cases} \qquad (2.14)$$

On dit que c'est un processus récurrent homogène.

3.2 - Discipline d'attente

Quand les clients arrivent dans le système et que le(ou les) serveur(s) est occupé, ils peuvent quitter le système (Lossing system) ou attendre dans la file d'attente (Waiting system).

Pour les clients qui attendent dans la file d'attente (Waiting system), il existe plusieurs disciplines de service en fonction de leur ordre d'arrivée:

a) Une politique de service P.A.P.S (premier arrivé - premier servi): les clients sont servis dans l'ordre d'arrivée.

b) Une politique D.A.P.S (dernier arrivé-premier servi).

c) Une politique de service aléatoire: le serveur choisit le client aléatoirement, comme par exemple dans le cas d'un standard qui sert un téléphone.

d) Une politique de service prioritaire.

etc.

Pour le système d'attente, la longueur de la file d'attente peut être infinie ou limitée (m par exemple); la file peut être unique ou au contraire, le système peut comprendre plusieurs files.

3.3 - Mécanisme de service

Les serveurs qui servent les clients peuvent présenter différentes configurations comme le montre la figure 2.2.

a) Une file-un serveur

b) Plusieurs files - plusieurs serveurs(en parallèle)

c) Une file - plusieurs serveurs (en parallèle)

d) Plusieurs serveurs en cascade

e) Plusieurs serveurs en mixte

Le temps de service peut être déterministe ou aléatoire. Pour un temps de service aléatoire, il faut connaître sa distribution, soit par l'expérience, soit par un modèle théorique (probabilité). Comme pour le processus des arrivées, on suppose que la distribution du temps de service est stable, c'est-à-dire que ses paramètres ne sont pas influencés par le temps.

En général, on connaît la distribution du temps de service, supposée indépendante du processus des arrivées. On note x_n le temps de service du n-ième client, et on définit:

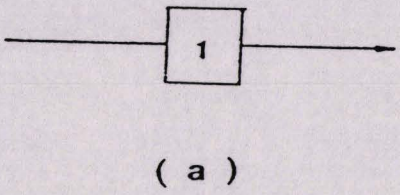
$$P\{x_n \leq x\} = H(x)$$

Le temps moyen de service est:

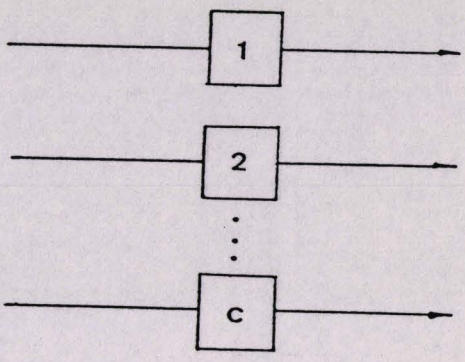
$$\alpha = \int_0^\infty x \, dH(x)$$

et la variance du temps de service est:

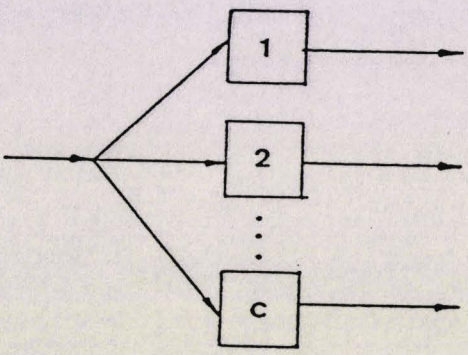
$$\sigma_\alpha^2 = \int_0^\infty (x-\alpha)^2 \, dH(x)$$



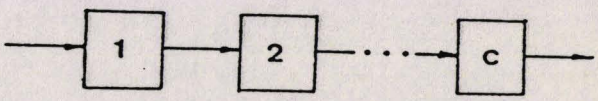
(a)



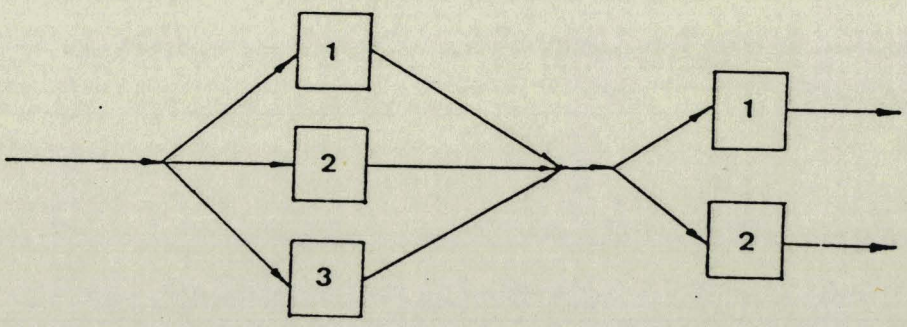
(b)



(c)



(d)



(e)

fig.2.2

Un cas particulier est celui dans lequel le temps de service a une distribution exponentielle. C'est-à-dire:

$$H(x)=\begin{cases}1-\exp(-\mu x) & \text{si } x\geq 0\\ 0 & \text{si } x<0\end{cases} \qquad (2.15)$$

Dans d'autre cas, on considère que le temps de service a une distribution gamma:

$$H(x)=\begin{cases}1-\sum_{j=1}^{m-1}\exp(-\mu x)\dfrac{(\mu x)^j}{j!} & \text{si } x\geq 0\\ 0 & \text{si } x<0\end{cases} \qquad (2.16)$$

3.4 - Les problèmes étudiés par la théorie de file d'attente

Lorsque les clients arrivés ne peuvent pas être servis immédiatemment et s'ils sont admissibles dans le système, on aboutit à des phénomènes d'attente. A cause du caractère aléatoire des arrivées de clients et des temps de service, on peut dire que les phénomènes d'attente sont presque inévitables.

Si l'on ajoute des équipements de service, il faut augmenter les investissements au risque d'aboutir au gaspillage du serveur oisif; si les équipements de service ne sont pas suffisants, le phénomène d'attente devient grave et crée des inconvénients pour les individus. Finalement, les gestionnaires doivent considérer un compromis entre ces deux circonstances, vérifier que le traitement est convenable et analyser des stratégies pour améliorer la qualité du service et en réduire le coût.

La théorie des files d'attente a été développée pour résoudre les problèmes évoqués. Les problèmes que la théorie des files d'attente étudie comprennent les trois parties suivantes:

1) Le problème des caractères On étudie la régularité des probabilités: La distribution de la longueur de la file d'attente, la distribution du temps d'attente et la distribution du temps de service. Il convient de distinguer les phénomènes transitoires et stationnaires.

2) Le problème d'optimisation Il comprend deux sortes d'optimisation: L'optimisation statique et l'optimisation dynamique. La première conduit à la conception optimale et la seconde à la gestion optimale.

3) Les analyses statistiques de file d'attente-identification On analyse un système de file d'attente pour vérifier quel modèle lui convient, afin d'appliquer à ce système les résultats fournis par la théorie.

3.5 - Le classement des modèles de file d'attente [KEN.51]

Pour un modèle donné, les trois caractères les plus importants sont:

1) Les arrivées de clients, qui sont caractérisées par la distribution d'intervalle de temps entre deux clients successifs.

2) La caractéristique du serveur, qui est définie par la distribution du temps de service.

3) Le type de file d'attente, auquel correspond le nombre des serveurs.

D.G.Kendall a donné une méthode pour classer les modèles. Le symbole qu'il utilise est:

$$X/Y/Z \qquad (2.17)$$

X: la distribution du temps d'intervalle entre les arrivées des clients
Y: la distribution du temps de service
Z: le nombre des serveurs

Les symboles usuellement adoptés pour les distributions d'intervalle de temps entre les arrivées des clients et du temps de service sont:

M: la distribution exponentielle

D: déterministe

E_k: la distribution de k-ième Erlang

GI: la distribution générale à tirages indépendants

G: la distribution générale

exemple: M/M/1 exprime que la distribution d'intervalle de temps entre les arrivées des clients est exponentielle et le temps de service est la distribution exponentielle avec un serveur. D/M/C: l'intervalle de temps entre les arrivées des clients est déterministe et le temps de service est la distribution exponentielle avec C serveurs.

3.6 - Le modèle utilisé dans la cellule d'assemblage

Pour analyser le problème d'affectation de la cellule dans le cas stationnaire, nous allons utiliser un modèle M/G/1 dans le chapitre suivant.

Dans ce modèle, la distribution du temps d'intervalle entre les arrivées des clients est exponentielle, c'est-à-dire que les arrivées sont Poissonniennes, et la distribution du temps de service est générale, de moyenne E[T] et de variance

Var[T]. C'est un modèle non-Markovien, pour lequel, Pollaczek et Khinchin ont donné les résultats suivants [GRO.74][KLE.76]:

Dans le cas $\rho=\lambda E[T]<1$, le nombre moyen de clients dans le système est Ls

$$L_s= \rho + \frac{\rho^2+\rho^2 Var[T]}{2(1-\rho)} \qquad (2.18)$$

Une autre expression est:

$$L_s= \rho + \rho^2 \frac{1+C_b^2}{2(1-\rho)} \qquad C_b^2= \frac{Var[T]}{E^2[T]} \qquad (2.19)$$

Utilisons la formule de Little [LIT.61], le temps moyen d'attente dans le système est $W_s=L_s/\lambda$, et le temps moyen d'attente dans le file est Wq, le nombre moyen de clients dans la file est Lq, on a donc:

$$Wq=Ws-E[T]$$

$$Lq= \lambda Wq$$

4 - LES METHODES D'OPTIMISATION

Les problèmes d'ordonnancement sont souvent résolus à l'aide des méthodes d'optimisation. Nous introduisons certaines méthodes dans ce paragraphe.

4.1 - Méthodes de programmation linéaire à variables mixtes

C'est une des méthodes les plus couramment utilisées dans un système de production. L'ordonnancement est décrit à l'aide d'un programme linéaire utilisant deux types de variables:

- les variables, temps de début d'exécution (t_i), et durée d'exécution de l'opération (d_i). Elles expriment les contraintes de succession. Si j est le successeur de i, on a $t_j=t_i+d_i$.

- les variables booléennes, qui expriment les contraintes disjonctives sur les moyens (une machine ne peut réaliser qu'une seule opération à la fois).

L'objectif peut être choisi comme:

minimiser la durée de réalisation du projet;
minimiser le coût du projet;
etc...

C'est donc une méthode qui permet d'obtenir la solution optimale, mais son coût en temps de calcul devient prohibitif si la taille de l'atelier est relativement grande. Il semble assez maladroit de résoudre les problèmes d'ordonnancement sous contraintes sur les moyens, à l'aide de cette méthode[SOU.81].

4.2 - Méthode de séparation et évaluation progressive [ROY.69]

La méthode de séparation et évaluation progressive (SEP) est une technique combinatoire d'optimisation. Elle est essentiellement une méthode efficace d'énumération partielle des solutions réalisables d'un problème décisionnel. Cette méthode est utilisée pour affecter les tâches aux robots dans le chapitre IV.

La méthode SEP a été appliquée à un grand nombre de problèmes d'optimisation, les problèmes d'affectation, les problèmes d'ordonnancement, les problèmes d'optimisation dans les réseaux.

Définitions et notations:

X: Un ensemble fini de variables de décision discrètes,
Contrainte implicite: Une contrainte qui est automatiquement satisfaite par la façon dont l'algorithme est construit, par exemple le fait que les éléments de X sont des variables 0-1.
Contrainte explicite: Une contrainte qui requiert une procédure spéciale dans l'algorithme pour en tenir compte.
Solution: Un ensemble de valeurs pour X qui satisfait toutes les contraintes implicites.
Solution réalisable: Un ensemble de valeurs pour X qui satisfait toutes les contraintes.
Solution optimale: Une solution réalisable qui minimise la fonction économique.

Algorithme général

Un algorithme de séparation et évaluation progressive définit un ensemble de règles pour:

1) séparer des noeuds en d'autres noeuds
2) évaluer les bornes inférieures pour ces nouveaux noeuds
3) choisir le prochain noeud intermédiaire à séparer
4) identifier les noeuds qui contiennent seulement des solutions non réalisables ou non optimales
5) identifier les noeuds finaux qui contiennent une solution optimale.

4.3- Méthodes d'analyse combinatoire

Le principe de base de ces méthodes est d'analyser l'ensemble des solutions possibles, pour en dégager des sous-ensembles remarquables et de réduire ainsi la dimension du problème à résoudre.

La propriété qui permet de déterminer ces sous-ensembles est la dominance [THO.80][ERS.82][ERS.85]. Elle repose sur deux conditions:

- la dominance par rapport à l'admissibilité (contraintes temporelles, et sur les moyens);

- la dominance par rapport aux critères.

Les conditions nécessaires et suffisantes d'admissibilité sont établies dans [ERS.82]. Dans [COU.79] et [FON.80] sont démontrées les conditions nécessaires de dominance à travers l'étude des structures pyramidales des diagrammes de GANTT pour le premier et la recherche de circuits dans un graphe pour le second.

Ces méthodes sont intéressantes dans un cadre d'aide à la décision puisqu'elles peuvent proposer plus d'une solution satisfaisante et permettent donc au décideur d'orienter son choix.

4.4 - Méthode de programmation dynamique [BEL.57][KAU.59]

La programmation dynamique est une technique mathématique conçue pour prendre une suite de décisions mutuellement reliées entre elles en vue d'optimiser un objectif donné. Comme les méthodes de séparation et évaluation progressive, elle constitue une exploration intelligemment structurée de l'espace des solutions réalisables d'un problème d'optimisation donnée. On ne peut pas résoudre tous les programmes dynamiques à l'aide d'un seul algorithme, comme c'est le cas pour la programmation linéaire. La programmation dynamique est plutôt une approche générale pour la résolution de problèmes complexes, et la forme des équations utilisées varie avec chaque nouvelle situation.

Comme la méthode de séparation et évaluation progressive, nous utilisons cette méthode dans le chapitre IV.

4.5 - Méthodes heuristiques

L'inconvénient majeur des méthodes présentées est leur incapacité à résoudre des problèmes de dimensions réalistes à cause d'une part, des besoins de place en mémoire qui risquent d'être importants, et d'autre part,du temps mis pour obtenir la solution optimale.

Les méthodes heuristiques pallient cet inconvénient mais malheureusement sans garantir un résultat optimal.

"Une heuristique est une règle de choix permettant d'agir en l'absence d'un résultat théorique sûr"[SIM.84].

Elle consiste généralement en le classement des opérations "ordonnançables", c'est-à-dire celles dont les antécédents sont déjà ordonnancés, selon une priorité donnée. Celle-ci peut être une règle PAPS (premier arrivé, premier servi), ou selon les temps d'exécution croissants, ou leurs dates au plus tard croissantes, etc...

La recherche de solutions sous-optimales, n'obéit donc pas à un besoin abstrait de traiter des problèmes complexes mais à un besoin réel de trouver des solutions aux problèmes qui se posent, dans l'industrie par exemple, et qui doivent être résolus. C'est tout l'intérêt de ces méthodes.

4.6 - Simulation [VAL.85]

> "La simulation est une technique numérique pour élaborer des expériences sur ordinateur. Elle implique l'utilisation de modèles logiques et mathématiques qui décrivent le comportement de systèmes administratifs ou économiques (ou de leurs sous-systèmes sur une période de temps prolongée)."[NAY.66]

La conception d'ateliers flexibles est un travail complexe. Dès la phase de pré-étude des problèmes de dimensionnement (stocks, système de transport) doivent souvent être résolus par simulation. La simulation peut également être utilisée pendant le fonctionnement du système pour réaliser des fonctions de surveillance et d'évaluation de décisions d'ordonnancement.

Dans les modèles à événements discrets pour la simulation, on aboutit à trois approches différentes [BEL.85] [FIS.78][MAT.74][PRI.79][BEL.83(G)].

Approche par événement Dans cette approche, il faut spécifier:

l'analyse du système puis de son fonctionnement qui permet de répertorier les différents types d'événements (changements d'état) pouvant avoir lieu au cours de la vie du système;

la modélisation de la logique de changement d'état qui correspond à ces divers types d'événements.

Approche par activités Cette approche peut être utilisée dans la mesure où le système ne répertorie plus les événements mais les différents types d'activités que l'on peut rencontrer dans son fonctionnement. La modélisation du changement d'état se fait en spécifiant les conditions sur l'état du système nécessaire à la réalisation du début ou de la fin de chacune de ces différentes activités.

Approche par processus dans cette approche, au lieu de définir la logique de changement d'état relative à chaque événement ou chaque début d'activité, on décrit séparément cette logique pour certaines séquences d'événements prédéterminées ou processus.

De nombreux langages spécifiques utilisent ces approches: GPSSFORTAN,SIMSCRIPT,GASPIV,ECSL,QNAP,SLAM,...

Parmi ces approches, la plus connue et la plus utilisée est l'approche par événement. Un bon outil de dialogue entre les mécaniciens et les ingénieurs système au niveau des systèmes mécaniques de convoyage est fourni par les Réseaux de Pétri. Nous introduisions dans ce chapitre la description par Réseaux de Petri et son application dans la cellule envisagée.

5 - MODELISATION DE LA CELLULE FLEXIBLE D'ASSEMBLAGE

5.1 - Utilisation de l'algèbre des DIOIDES

La cellule flexible d'assemblage considérée est composée de deux robots, de trois plans de travail, un propre à chacun des robots et un disposé en zone commune.(figure 2.3)

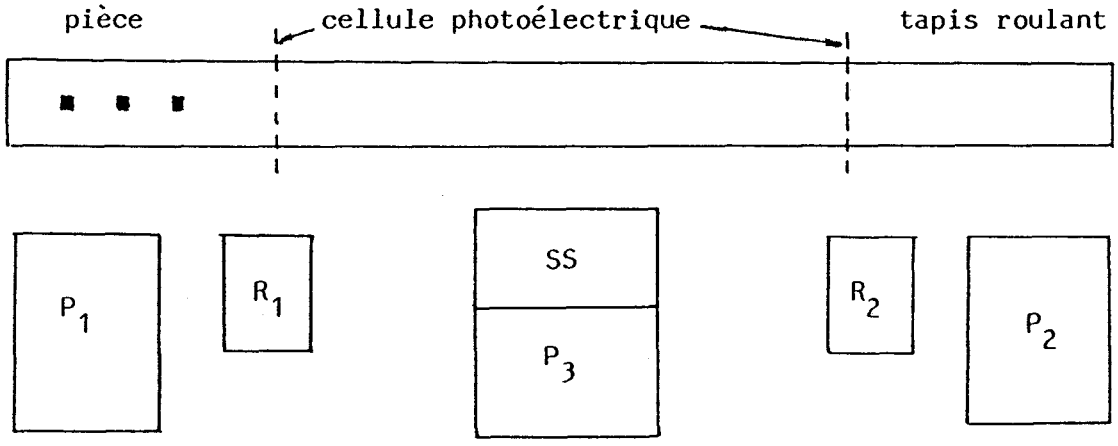


fig.2.3

Comme nous l'avons signalé au chapitre précédent, nous modélisons la gamme d'assemblage, sous forme d'un graphe. Les noeuds représentent des actions (tâches) élémentaires d'assemblage et les arcs définissent les relations entre les

noeuds (Nous ne considérons que la relation d'ordre, c'est-à-dire l'arc simple).

Les données associées à une tâche j sont:

- Durée de l'exécution d_j;
- Localisation temporelle de l'exécution avec:
  - . Une date d'exécution au plus tôt x_j ;
  - . Une date d'exécution au plus tard y_j ;
- Liste des robots capables de réaliser la tâche ;
- Liste des ressources associées à la tâche ;

Soient: A(j) l'ensemble des antécédents de la tâche j ;

R(j) l'ensemble des ressources nécessaires pour la tâche j ;

S(j) l'ensemble des successeurs de la tâche j ;

x_j et y_j peuvent être évalués de la façon suivante:

$$x_j=\max\left(\max_{i\in A(j)} x_i+d_i \ , \ \max_{r\in R(j)} u_r\right) \qquad (2.20)$$

où u_r est la date de disponibilité de la ressource r. Dans notre cas les ressources sont les robots, les pièces sur le tapis roulant ou sur le stock statique.

Notons que les pièces arrivant de façon aléatoire, la date d'exécution au plus tôt de la tâche j x_j sera donc aléatoire. De la même façon, la date d'exécution au plus tard est donnée par:

$$y_j=\max\left(\max_{i\in G(t)\setminus S(j)\cup\{j\}} x_i+d_i \ , \ \max_{r\in R(j)} u_r\right) \qquad (2.21)$$

où G(t) est l'ensemble des tâches qui ne sont pas encore réalisées à l'instant t.

Dans la cellule que nous envisageons, la date de disponibilité des ressources dépend de la date d'arrivée des pièces et de la date de disponibilité des robots. Soient U1 et U2 les vecteurs représentant les dates d'arrivée des pièces et les dates de disponibilité des robots.

Si on considère que x_f est la date de début au plus tôt de la dernière tâche du montage, le problème d'ordonnancement des tâches s'exprime par:

Trouver U2=f(U1) pour minimiser x_f sous les contraintes (2.20).

les dates de disponibilité des ressources étant aléatoires, on pourrait évaluer les dates de début des tâches en moyenne en utilisant les probabilités d'arrivée des pièces. Mais cette solution ne convient pas pour piloter la cellule en temps réel.

Il semble que cette approche puisse être utilisée dans le cas déterministe: les routes que suivent les pièces dans l'atelier sont connues, et l'ordre des pièces devant les machines est imposé. D'éventuels temps de préparation sur les machines, les durées de transport, l'existence de machines identiques peuvent aussi être intégrés aux modèles. De plus, les algorithmes (de type <plus court chemin>) mis en jeu par cette approche pour l'analyse des processus de production répétitifs sont simples. Enfin l'approche déterministe ici évoquée, permet une première analyse de la conduite de l'atelier, à travers les choix de tâches aux machines [BEL.85].

Cette approche est limitée dans le sens où elle ne peut modéliser les systèmes stochastiques. Cela suppose des systèmes de production suffisamment fiables pour que le régime périodique stationnaire décrit par (2.6) puisse être maintenu. Dans le cadre de la production en moyenne série où le plan de production est suffisamment stable, on peut penser que la technologie deviendra suffisamment robuste pour que l'atelier automatisé fonctionne réellement comme un automate sur un intervalle de temps significatif, mais il est clair que la plupart des ateliers existants sont loin de cet idéal, et que leur étude détaillée nécessite le recours à des outils moins analytiques, plus généraux, et plus souples.

5.2 - Modélisation de la cellule par Réseaux de Pétri

La modélisation de la cellule d'assemblage que nous avons envisagée précédemment est considérée dans le groupe de coopération entre robots du Laboratoire d'automatique de l'U.S.T de Lille [STA.85(a)][DJE.86][BAY.87] à partir d'une description par graphes d'état:

Soient a et r , deux variables logiques correspondant respectivement à : antécédents réalisés et ressources propres disponibles.

Le passage de l'état non-exécutable à l'état exécutable s'effectue lorsque la fonction logique a.r est satisfaite. Les états intermédiaires correspondant aux fonctions $\bar{a}.r$ et $a.\bar{r}$ sont pris en compte séparément par le module de décision.

Pour une opération de production, reprenons, en les complétant, les caractéristiques des tâches la constituant. Pour chaque tâche, les états suivants peuvent être distingués:

Bloqué: L'opération appartient à un assemblage qui n'est pas en cours de fabrication.

Non exécutable: Tous les antécédents de l'opération ne sont pas réalisés, et toutes ses ressources propres ne sont pas disponibles.

Il correspond à la fonction logique: $\bar{a}.\bar{r}$.

Pré-exécutable: Cet état intermédiaire correspond en fait à deux situations:

- antécédents réalisés
  ou
- ressources propres disponibles

c'est-à-dire $a\bar{r}+\bar{a}r$

On distingue en fait les deux états suivants:

pré-exécutable(a) $a\bar{r}$
pré-exécutable(r) $\bar{a}r$

Exécutable: Les deux conditions suivantes sont vérifiées:

- Antécédents réalisés
  et
- Ressources propres disponibles

c'est-à-dire $a \cdot r$

En attente: La décision de réaliser l'opération est prise, mais les ressources partageables ne sont pas toutes disponibles. L'affectation peut être remise en cause si l'évolution de l'état de la cellule diffère de celle qui est prévue.

En cours: La tâche est en cours d'exécution, les ressources partageables dont elle a besoin sont réservées suivant une certaine planification.

Suspendu: La réalisation de l'opération est suspendue (par exemple on doit attendre des ressources qui étaient prévues présentes mais qui ne sont pas disponibles). Suivant l'évolution de la cellule, l'opération peut se terminer normalement ou deviendra non réalisable.

Terminée: L'opération est correctement réalisée.

Non réalisable: L'opération ne peut plus être poursuivie. Après un appel d'urgence, le système devra être réinitialisé. Ceci entraîne que toutes les opérations d'un même assemblage, quelque soit leur état, reviennent dans l'état initial.

L'évolution d'une opération entre ces différents états peut être décrite à l'aide du graphe d'état suivant.

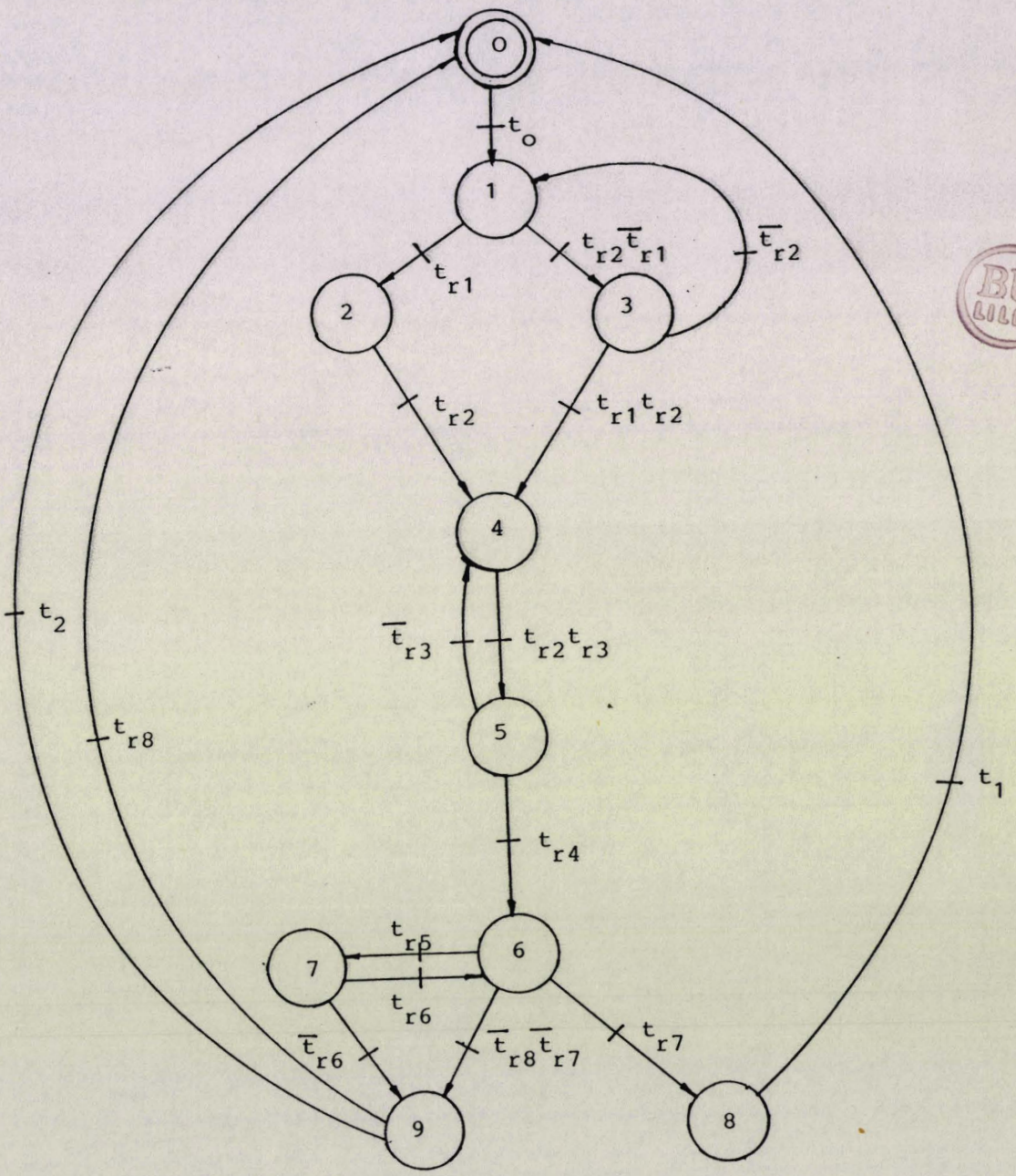


fig.2.4

0	bloqué	5	en attente
1	non exécutable	6	en cours
2	pré-exécutable(a)	7	suspendu
3	pré-exécutable(r)	8	terminé
4	exécutable	9	non réalisable

t_0: la décision de commencer l'assemblage a été prise,
t_{r1}: tous les antécédents de l'opération sont dans l'état terminé,
t_{r2}: les ressources consommables de l'opération sont présentes dans la cellule,
t_{r3}: la décision d'exécuter l'opération est prise,
t_{r4}: les ressources partageables nécessaires à l'exécution de l'opération sont réservées. Les ressources sont:

- un ou plusieurs robots
- la zone commune

t_{r5}: certaines ressources nécessaires à la poursuite de l'opération ne sont pas disponibles,
t_{r6}: les conditions de continuation de l'opération sont vérifiées et l'exécution peut être reprise,
t_{r7}: la tâche est correctement terminée,
t_{r8}: après une mauvaise exécution, l'opération peut être réexécutée,
t_1: réinitialisation lorsque l'assemblage est fini,
t_2: réinitialisation après une mauvaise exécution.

Les réseaux de Pétri peuvent être utilisés de façon systèmatique pour modéliser les systèmes automatisés de production. La représentation par réseau de Pétri permet aussi de décrire des modèles à événements discrets dont le fonctionnement n'est pas complètement déterminé par les règles opératoires. La représentation par réseau de Pétri permet, à l'aide de techniques appropriées [BRA.83] de valider certains aspects du fonctionnement du système de production modélisé:

- absence de configuration bloquée (<deadlocks>)

- caractère borné du nombre de jetons dans le réseau

- accessibilité des différentes places

Notons cependant que ces aspects sont en général étudiés indépendamment de la temporisation. Le caractère temporisé des réseaux de Pétri nécessite certaines modifications de ces techniques [CHR.83] [CAR.83].

Néanmoins la représentation par réseau de Petri ne permet pas d'exprimer la règle de décision qui sera utilisée pour choisir entre les transitions que l'on peut activer. Cette approche ne peut pas résoudre le problème d'ordonnancement, le problème de priorité, par exemple, le problème d'optimisation du système. Par ailleurs, le risque d'explosion du nombre de places rend toute exploitation des réseaux obtenus impossible, et il est quelquefois difficile de représenter certaines règles.

5.3 - Modélisation de la cellule d'assemblage par un système de file d'attente

La cellule flexible d'assemblage envisagée peut être modélisée par un système de file d' attente.

1) Les clients sont les tâches exécutables qui dépendent des tâches pré-exécutables par rapport aux antécédents et des tâches pré-exécutables par rapport aux ressources (figure 2.5).

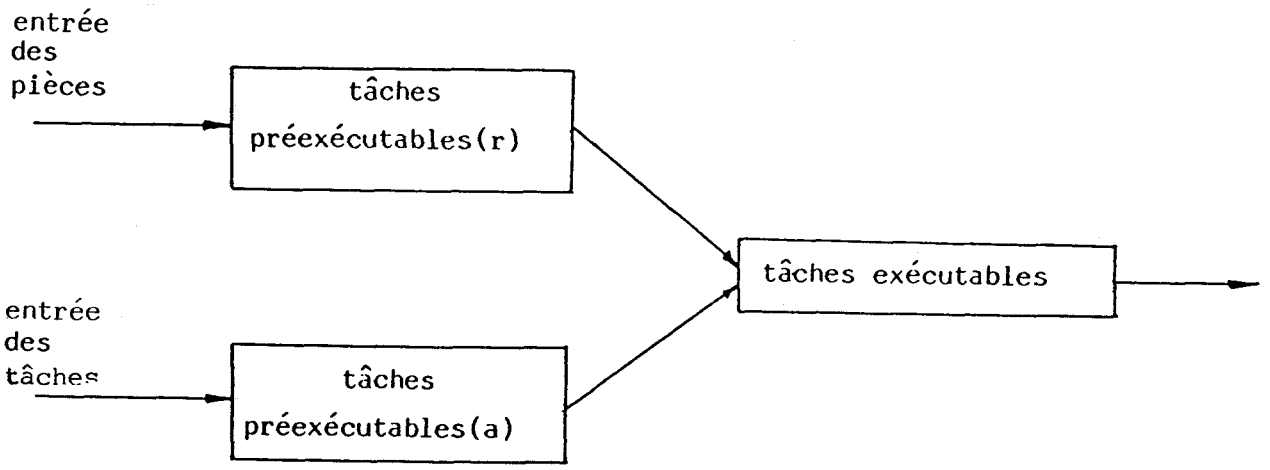


fig.2.5

Les intervalles de temps d'arrivée des tâches exécutables sont inconnus puisque les arrivées de pièces sont aléatoires, mais ils peuvent être évalués par les probabilités.

2) La discipline d'attente permet de servir les clients en P.A.P.S, en priorité ou par toute autre discipline. Dans notre cas, pour minimiser le temps d'exécution d'un montage, on considère la discipline prioritaire.

3) Les serveurs du système sont les robots. Nous considérons dans cette thèse le cas où un ou deux robots exécutent les tâches. Dans le cas où la cellule comprend deux robots, il faut affecter les tâches exécutables (figure 2.6).

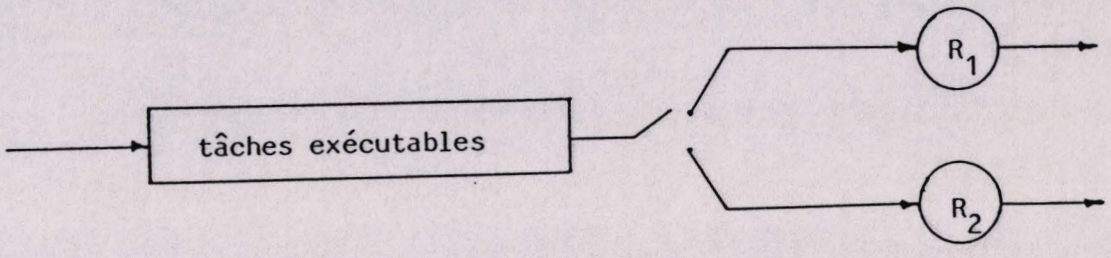


fig.2.6

Le modèle général considéré par le système de file d'attente est donné figure 2.7 . Nous allons analyser le pilotage de ce système dans les chapitres suivants.

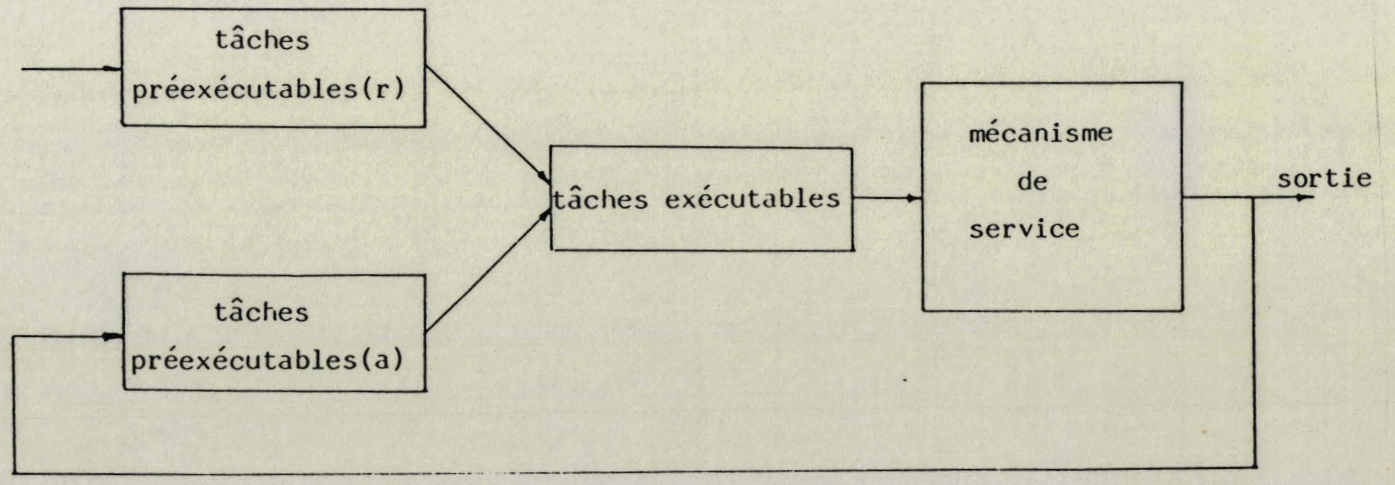


fig.2.7

La théorie des files d'attente est une partie très importante de la recherche opérationnelle, utilisée largement pour le transport, la production, le service. Elle s'intéresse

aux systèmes d'exploitation d'ordinateurs, aux flux de trafic de véhicules, aux lignes de production, données de télécommunication, ateliers flexibles, etc. Grâce au développement de la science informatique, les ordinateurs, des systèmes de commande, l'application et le développement de la théorie des files d'attente ne cessent de croître. Pour un système automatisé de production, on s'intéresse aux problèmes d'attente, de demande de service, de priorité, d'allocation des ressources. Le problème de file d'attente comprend le processus des arrivées, la discipline d'attente, le mécanisme de service, les caractéristiques élémentaires des files d'attente. La théorie des files d'attente en liaison avec l'analyse des performances, l'analyse statistique, l'analyse d'évolution, la simulation, la théorie de la commande, constitue un outil efficace pour modéliser et tenter d'optimiser un système stochastique.

6 - CONCLUSION

Dans ce chapitre, nous avons considéré diverses méthodes pour modéliser un système de production flexible. Le développement assez récent des études dans le domaine des systèmes automatisés de production a mis en évidence la multiplicité des modèles adaptés à l'étude d'un ou plusieurs aspects du fonctionnement dynamique de ces systèmes: l'algèbre des Dioides (PERT), le réseaux de PETRI et le système de file d'attente. Pour modéliser la cellule que nous avons envisagée, nous utiliserons une représentation par graphes d'état pour faire la simulation et un modèle de files d'attente pour conduire la cellule, puisque c'est un système stochastique. Dans les chapitres suivants, nous analyserons les modèles des systèmes de file d'attente pour conduire la cellule envisagée, c'est pourquoi nous en avons rappelé certains résultats élémentaires.

CHAPITRE III

CONTROLE D'UNE CELLULE A UN SERVEUR

1 - INTRODUCTION

L'atelier flexible est un système de production complexe. Il est donc important de disposer d'outils qui permettent de modéliser et d'évaluer les particularités d'un tel système. Les deux classes d'outils existantes sont d'une part la simulation et d'autre part les méthodes analytiques. C'est cette deuxième classe à laquelle nous nous intéressons particulièrement. Le but de notre étude est la conduite d'une cellule flexible à l'aide des modèles de la théorie des files d'attente.

La cellule flexible d'assemblage que nous étudions dans ce chapitre est composée d'un seul robot qui doit exécuter les n tâches d'un montage. Les ressources propres associées aux tâches sont les pièces qui arrivent aléatoirement sur le tapis roulant. L'objectif de fonctionnement du système est de minimiser le temps d'exécution du montage. Cette cellule peut être considérée comme un système de file d'attente avec un serveur. Les deux disciplines d'attente que nous considèrerons sont:

- PAPS (premier arrivé, premier servi): les clients (les tâches exécutables dans notre cas) sont servis en fonction de leur ordre d'arrivée,

- discipline prioritaire: les clients sont servis dans l'ordre des gains décroissants associés à chacun d'eux, il s'agit d'un problème d'ordonnancement dynamique.

Sur le problème de file d'attente à priorité dynamique, [COB.54] a donné le premier un modèle dans lequel il existe r classes de clients. Les clients de haute priorité peuvent déplacer les clients de basse priorité sous la condition de non-préemptivité. Puis [BEL.71] et [HAR.75] ont étudié et développé ce modèle. Beaucoup de modèles relatifs au problème de la priorité dynamique envisagée par un système de file d'attente se basent sur la théorie de Markov (à temps continu ou discret). La théorie des processus de Markov permet d'établir les équations du système puis de les résoudre en fonction des états du système. Mais dans les cas concrets, il existe des systèmes de file d'attente qui sont plus complexes: ils peuvent avoir plus d'un serveur ou un nombre de places limité dans le système. Les clients peuvent être refusés par le système ou quitter le système après y être entrés. Des clients différents peuvent avoir des priorités différentes, etc. De nombreux travaux sur le problème du contrôle dynamique du système de file d'attente ont été effectués ces dernières années:[ROS.70][KLE.79][STI.85].

Nous considérons premièrement dans ce chapitre un système de file d'attente utilisant une politique PAPS. On suppose dans cette approche que les tâches à réaliser sont indépendantes. Les tâches peuvent être servies dans l'ordre des arrivées de pièces dans le système. Le contrôle de ce système consiste alors à commander l'arrivée des pièces.

Dans le cas où les différentes tâches nécessaires aux assemblages forment une gamme de fabrication qui est décrite à l'aide d'un graphe orienté acyclique, le système de file d'attente modélisé utilise une discipline prioritaire. Chaque tâche exécutable réalisée apporte une certaine quantité de travail, on l'appelle le gain du système. Le tâche prioritaire à chaque instant t est celle qui a le gain maximal.

2 - MODELISATION D'UNE CELLULE FLEXIBLE PAR UN SYSTEME DE FILE D'ATTENTE

Nous avons modélisé la cellule flexible par un système de file d'attente dans le chapitre précédent. Donnons à présent plus de détails sur cette approche.

Hypothèses:

1) Les pièces arrivent sur le tapis roulant de façon aléatoire, on peut estimer la probabilité d'arrivée de chaque pièce de type j à l'instant t, $P_j(t)$ (j=1,2,...m) ou la distribution des intervalles de temps entre deux arrivées successives de chaque type de pièce.

2) Le temps d'exécution de chaque tâche est connu. Dans le cas où ceci n'est pas vérifié, on pourra estimer la distribution des temps de réalisation de chaque type de tâche.

La figure ci dessous rappelle ce modèle .

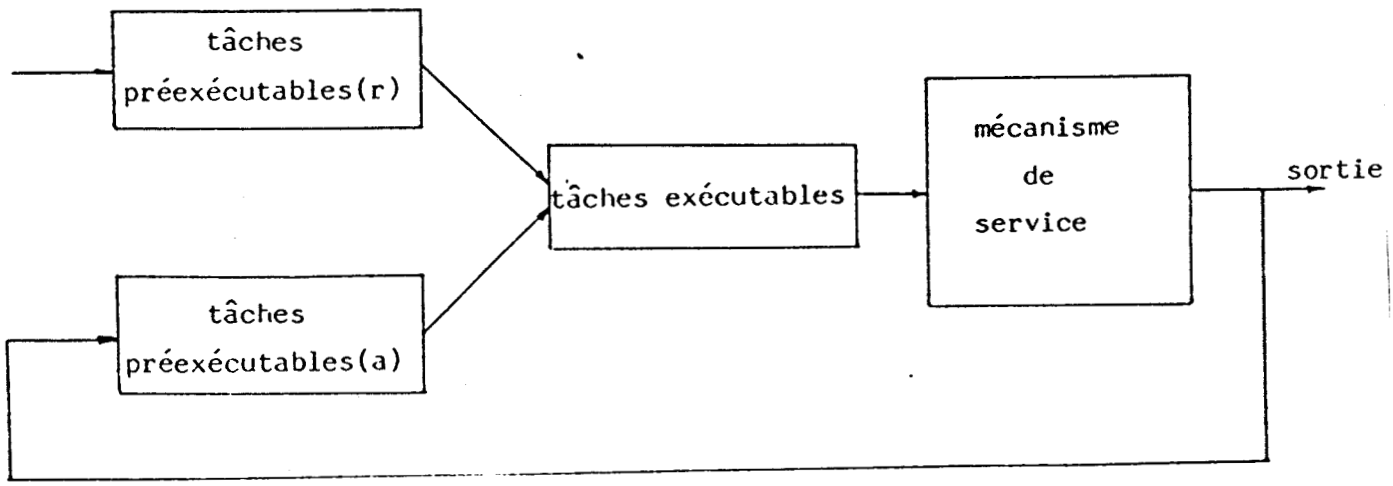


fig.3.1

Dans ce système, il existe deux files, l'une est la file d'arrivée des pièces, l'autre est la file des tâches. Dans le mécanisme de service, on trouve le robot. Les pièces proviennent d'un magasin ou des stocks de la cellule, les arrivées de tâches dépendent de l'état d'avancement du graphe décrivant les tâches à réaliser.

Les pièces arrivent les unes après les autres. Elles doivent attendre dans la file lorsque le robot est occupé. On suppose de plus que la capacité de stockage sur le tapis est limitée (M pièces, par exemple).

Les tâches arrivent dans le système, soit l'une après l'autre, soit en groupe. Ceci dépend des tâches pré-exécutables(a) à l'instant t. La capacité de cette file n'est pas limitée, toutes les tâches pré-exécutables(a) peuvent entrer dans le système.

Nous nous intéressons au fonctionnement dynamique du système et nous étudions l'ordonnancement/affectation dynamique à l'aide de la théorie des files d'attente.

D'une part, pour réaliser n tâches indépendantes, nous étudions le contrôle du système avec un robot qui respecte une politique de service PAPS.

D'autre part, pour exécuter les n tâches d'un montage, nous considérons l'ordonnancement dynamique dans le système de file d'attente. Dans ce cas, le système ne respecte pas la politique de service PAPS.

L'objectif est alors de minimiser le temps d'exécution des n tâches.

3 - CONTROLE D'UN SYSTEME DE FILE D'ATTENTE PAR UN ROBOT QUI REALISE n TACHES INDEPENDANTES

On suppose ici que le robot doit réaliser n tâches indépendantes, et que la discipline de service respecte une politique PAPS. Les pièces arrivées dans le système seront utilisées les unes après les autres pour l'exécution des tâches par le robot. La minimisation du temps d'exécution des n tâches consiste à minimiser le temps d'oisiveté du robot.

En considérant que le robot n'est pas en panne pendant l'exécution des tâches, nous cherchons les conditions pour que le robot travaille sans temps d'attente.

Soient $X=\{x_1,x_2,x_3,\ldots x_n\}$ l'ensemble des tâches

$T=\{T_1,T_2,\ldots T_n\}$ les temps d'exécution des tâches

$R=\{r_1, r_2, \ldots r_s\}$ l'ensemble des ressources consommables nécessaires à l'exécution des tâches. $s \leq n$.
On a:
s=n à chaque tâche correspond sa ressource propre
s<n plusieurs tâches peuvent consommer la même ressource

$Q_i (i=1,2,\ldots n)$ le nombre de tâches i à réaliser. Ce nombre est fini mais peut être considéré comme infini dans certains cas.
(régime permanent, par exemple)

Le graphe des tâches est représenté ci-dessous (r(i) est la ressource de R utilisée par la tâche i)(figure 3.2).

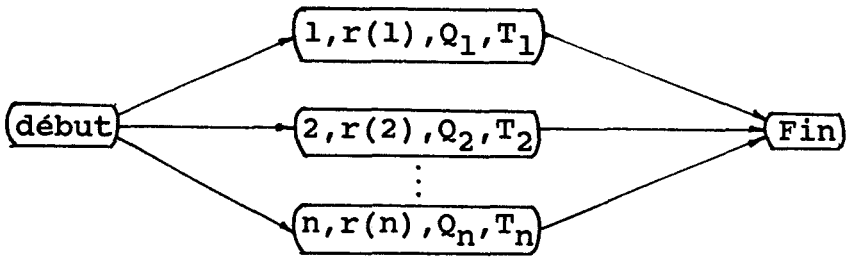


fig.3.2

Soit $X(t)=(x_1(t), x_2(t), \ldots x_m(t), 0, 0, \ldots 0)^T$ le vecteur de dimension M+1 décrivant la file d'attente à l'instant t: $x_1(t)$ est la tâche en cours d'exécution, $x_j(t)$ $j \geq 2$ est la tâche exécutable en $j^{ème}$ position dans la file.

Le travail présent dans le système est par définition la durée de traitement des tâches exécutables à l'instant t.

En tenant compte du fait que ces tâches comprennent la tâche en cours de service, la quantité de travail présente dans le système à l'instant t est:

$L_t = \sum_{i=1}^{m} T_i - \Delta_t$ où Δ_t est le temps écoulé depuis le commencement de la tâche en cours par le robot.

L'évolution de l'état des tâches exécutables peut être analysée comme suit:

Soit E une variable logique représentant l'événement entrée d'une pièce

Soit F une variable logique représentant l'événement fin de l'exécution de la tâche en cours

A l'instant t, nous avons les quatre possibilités suivantes:

E	F
0	0
0	1
1	0
1	1

L'état des tâches exécutables du système à l'instant t+dt est:

$$X(t+dt)=AX(t)+E\bar{F}f(X(t),r,Q_r(t))+EFD_sf(X(t),r,Q_r(t)) \quad (3.1)$$

où

$$D_s=\left(\begin{array}{c|ccc} 0 & 1 & & 0 \\ & & \ddots & \\ & & & 1 \\ \hline 0 & & & 0 \end{array}\right) \qquad \text{et } A=(1-F)I+FD_s$$

$$f(X(t),r,Q_r(t))=(\underbrace{0\ldots0}_{m}\ X_{m+1}(t),0\ldots0)^T$$

et r représente la ressource entrante

Deux cas doivent alors être distingués:

a) Chaque tâche consomme une ressource qui lui est propre: $r(i)=r_i \qquad r_i\neq r_j \quad$ si $i\neq j$

$X_{m+1}(t)=x_i \qquad$ si $r=r_i$ et $Q_i(t)>0$

$X_{m+1}(t)=0 \qquad$ sinon

b) Des tâches différentes peuvent consommer la même ressource: $i,j(i\neq j)\ \{1,\cdots n\}$ et $k\in\{1,\cdots s\}$ tels que $r(i)=r(j)=r_k$.

Si $r=r_i$, en notant $x_{ri}\subset X$ l'ensemble des tâches utilisant la ressource r_i, on a :

$X_{m+1}(t)=x_{io}$ tel que $Q_{io}=\max_{k\in x_{ri}} Q_k$ si $Q_{io}>0$

$X_{m+1}(t)=0$ sinon

Pour connaître la situation du robot, nous évaluons la quantité de travail présente dans le système à l'instant t+dt sous l'hypothèse qu'entre t et t+dt une ressource au plus a pu pénétrer dans le système:

$$\begin{aligned} L_{t+dt}&=L_t-dt+L_e(t,t+dt) && \text{si } L_t\geq dt \\ L_{t+dt}&=L_e(t,t+dt) && \text{si } L_t<dt \end{aligned} \qquad (3.2)$$

$L_e(t,t+dt)$: quantité de travail introduite dans la file entre t et t+dt

$L_e(t,t+dt)=Tx_{m+1}$

Nous supposons que la quantité de travail à t_o, où le robot commence à travailler, est $L_o(L_o>0)$. Il est certain que le robot travaille sans interruption entre t_o et t_1 avec $t_1=t_o+L_o$ (figure 3.3).

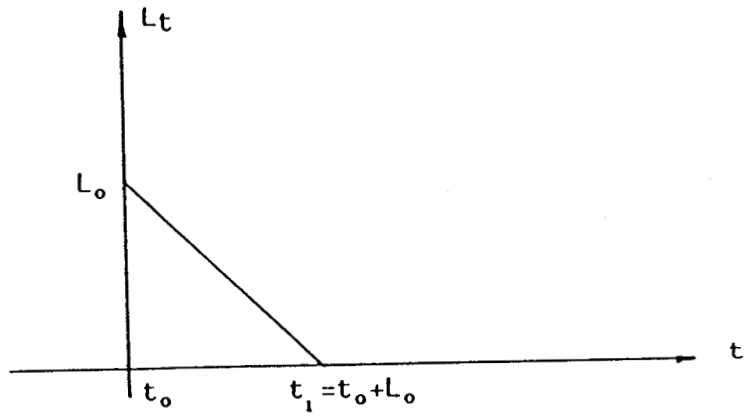


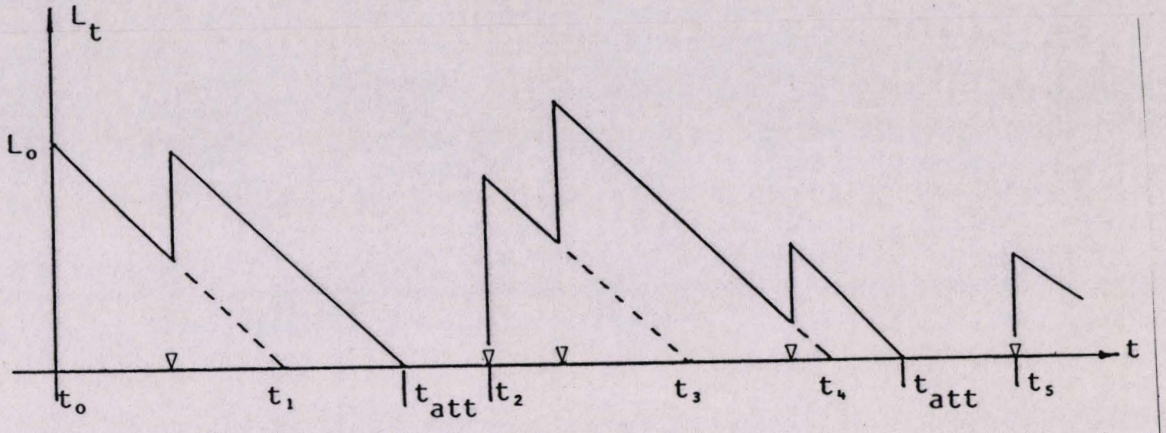
fig.3.3

Pour contrôler le système, nous notons que $L_e(t_i,t_{i+1})$ est la quantité de travail introduite dans la file entre t_i et t_{i+1} .

t_{i+1} peut être défini par:

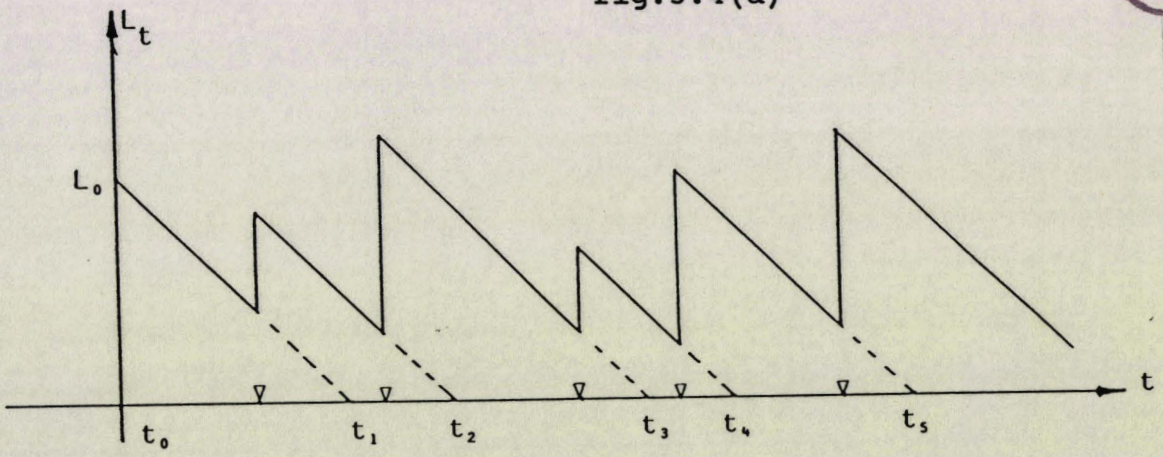
$$t_{i+1}=t_i+L_i+t_{att}$$

où t_{att} est le temps d'attente du robot. Si $t_{att}=0$, le robot travaille sans interruption.(figure 3.4(a), 3.4(b))



$t_{att} \neq 0$

fig.3.4(a)



$t_{att}=0$

▽ : l'arrivée d'une pièce dans la file

fig.3.4(b)

Sur la figure 3.4(a), le temps d'attente est positif. Pendant ce temps, le robot est obligé d'arrêter son travail pour attendre une arrivée de pièce. Sur la figure 3.4(b), le

temps d'attente est nul, le robot travaille donc sans interruption. La condition nécessaire et suffisante pour qu'il en soit ainsi est:

1) $L_e(t_i,t_{i+1})>0$ $\quad i=0,1,2,...$

Par ailleurs, la condition de non débordement de la file d'attente s'écrit:

2) $m<M+1$

Le contrôle de ce système consiste à commander l'arrivée des pièces pour satisfaire les conditions 1) et 2). Le système de contrôle est alors celui de la figure 3.5.

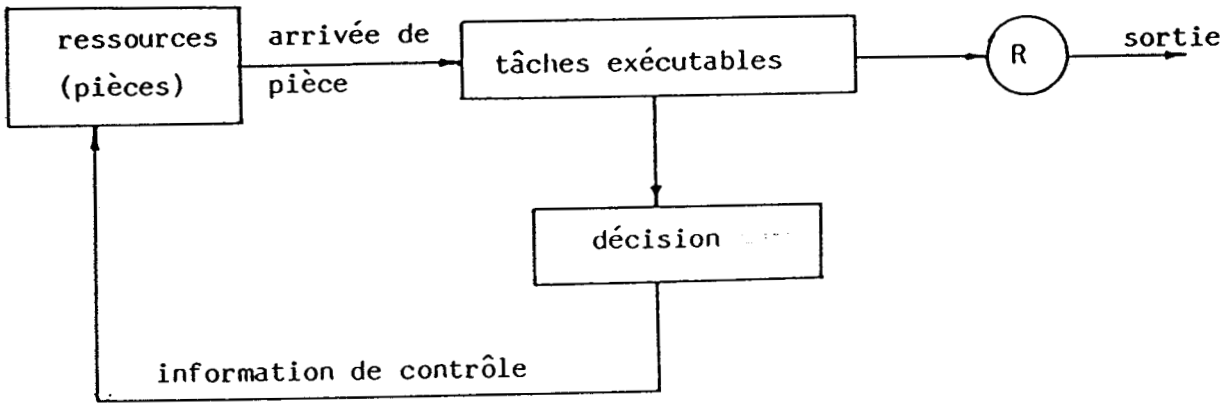


fig.3.5

Conclusion Pour minimiser le temps d'exécution de n tâches indépendantes, il faut commander le taux d'arrivée des pièces. A chaque instant t_i $(i=0,1,2,...)$, on peut donc donner une information de contrôle pour que la file contienne au moins une pièce avant t_{i+1} avec $t_{i+1}=t_i+L_i$.

4 - CONTROLE D'UN SYSTEME DE FILE D'ATTENTE PAR UN ROBOT QUI NE RESPECTE PAS UNE POLITIQUE DE SERVICE PAPS [STA.87]

Le système envisagé est composé d'un seul robot qui exécute n tâches d'un montage. Les différentes tâches nécessaires aux assemblages forment alors ce que l'on appelle la gamme de fabrication. Celle-ci, qui traduit l'enchaînement de toutes les opérations à réaliser est décrite à l'aide d'un graphe orienté acyclique G(N,A), où N est l'ensemble des noeuds et A celui des arcs [BER.83].

Ce graphe fait apparaître:

- toutes les opérations à réaliser, correspondant aux noeuds N de G;

- les relations de succession entre ces opérations représentées par les arcs A de G.

Pour réaliser les tâches du montage, les pièces associées sont indispensables. Elles arrivent aléatoirement sur un tapis roulant. Le modèle envisagé est représenté à la figure 2.3.

Nous avons défini les états des tâches au chapitre I. Nous précisons dans ce paragraphe les ensembles de tâches qui sont dans le même état, en utilisant le concept de sous ensemble flou [KAU.73] .

On note:

PEA(t): l'ensemble flou des tâches pré-exécutables par rapport aux antécédents à l'instant t,

PER(t): l'ensemble flou des tâches pré-exécutables par rapport aux ressources (les pièces) à l'instant t.

EXE(t): l'ensemble flou des tâches exécutables à l'instant t.

On décide de représenter les ensembles de tâches PEA(t), PER(t), EXE(t) comme des ensembles flous pour la raison suivante:

PEA(t): une tâche est pré-exécutable par rapport aux antécédents lorsque tous ses prédécesseurs dans le graphe potentiel tâche sont exécutés. On dira qu'une telle tâche est strictement pré-exécutable(a) et on notera cet ensemble $\underline{PEA}(t)$. Pour les autres tâches, nous introduisons un coefficient d'appartenance à l'ensemble des tâches pré-exécutables(a) qui dépend du nombre de prédécesseurs effectivement réalisés par rapport au nombre total de prédécesseurs. L'idée sous jacente est qu'une tâche dont presque tous les prédécesseurs sont réalisés est "plus proche" de l'exécutabilité qu'une tâche pour laquelle aucun des prédécesseurs n'est encore réalisé. On notera $\widetilde{PEA}(t)$ l'ensemble des tâches dont le coefficient de pré-exécutabilité(a) est strictement inférieur à 1. L'ensemble flou PEA(t) est alors donné par:

$$PEA(t)=\underline{PEA}(t)\cup\widetilde{PEA}(t) \qquad (3.3)$$

PER(t): une tâche est pré-exécutable par rapport aux ressources lorsque toutes les ressources consommables nécessaires à son exécution sont présentes dans la cellule. On note $\underline{PER}(t)$ l'ensemble de ces tâches, strictement pré-exécutables(r). Comme ci-dessus, on estime qu'à l'instant t, une tâche pour laquelle la probabilité d'arrivée de ses

ressources est grande est "plus proche" de l'exécutabilité qu'une tâche dont la probabilité d'arrivée des ressources est faible. On note $\underset{\sim}{PER}(t)$ l'ensemble des tâches dont le coefficient de pré-exécutabilité(r) est strictement inférieur à 1, l'ensemble flou PER(t) est alors donné par:

$$PER(t)=\underline{PER}(t)\cup PER(t) \qquad (3.4)$$

L'ensemble flou des tâches exécutables est alors:

$$EXE(t)=PEA(t)\cap PER(t) \qquad (3.5)$$

On verra par la suite comment les différents coefficients d'appartenance sont définis.

La procédure de décision retenue consiste à choisir à l'instant t une tâche x apportant le travail maximal. Cette tâche doit bien sûr être strictement exécutable:

$x\in\underline{EXE}(t)$ où $\underline{EXE}(t)$ est l'ensemble des tâches strictement exécutables à t

4.1 - Situation des pièces dans le système

Hypothèses: Les pièces arrivent dans le système de façon aléatoire. Pour chaque type de pièce la probabilité $p_j(t)$ (j=1,2,...s,j est le type de pièce) est connue ou bien on peut estimer sa distribution d'arrivée. Les pièces qui ne sont pas utilisées attendent dans la queue. La discipline de service ne respecte pas le PAPS, elle dépend de la demande des tâches.

Chaque type de pièce peut être consommé par une seule tâche ou plusieurs tâches. Dans les paragraphes qui suivent, nous utilisons la première hypothèse. Le deuxième cas sera étudié au § 5.

4.2 - Situation des tâches dans le système

Les tâches sont les actions élémentaires de l'assemblage. Dans le système de file d'attente envisagé, les tâches exécutables entrent dans le système l'une après l'autre ou en groupe, en fonction des tâches débloquées et des pièces arrivées dans le système.

L'ensemble des tâches strictement exécutables à l'instant t définit la file d'attente:

$$X(t)=(x_1(t),x_2(t),\cdots x_m(t),0\cdots 0)^T$$

Nous avons alors:

$$X(t+dt)=AX(t)+\Sigma(t,t+dt) \qquad (3.6)$$

$$A=\begin{cases} I_{M+1} & \text{aucune tâche ne se termine pendant } [t,t+dt). \\ D_s & \text{la tâche i se termine pendant } [t,t+dt). \end{cases}$$

$$D_s=\left(\begin{array}{c|c} I_{i-1} & 0 \\ \hline 0 & \begin{matrix} 0 & 1 & & 0 \\ & & \ddots & \\ & & & 1 \\ & & & 0 \end{matrix} \end{array}\right)$$

$$I_{M+1}=\begin{pmatrix} 1 & & & \\ & 1 & & 0 \\ & & \ddots & \\ 0 & & & 1 \end{pmatrix}$$

$\Sigma(t,t+dt)$: est un vecteur représentant les tâches exécutables introduites dans le système entre t et t+dt. Cet ensemble dépend:

1) des tâches débloquées pendant [t,t+dt)
2) des pièces arrivées pendant [t,t+dt).

Soit L_t la quantité de travail présente à l'instant t, la quantité de travail à t+dt est:

$$\begin{aligned} L_{t+dt}&=L_t-dt+L[\Sigma(t,t+dt)] && \text{si } L_t\geq dt \\ L_{t+dt}&=L[\Sigma(t,t+dt)] && \text{si } L_t<dt \end{aligned} \qquad (3.7)$$

$L[\Sigma(t,t+dt)]$: La quantité de travail introduite dans le système, entre t et t+dt.

4.3 - Priorité dans la discipline de la file d'attente

Dans un grand nombre de systèmes de file d'attente, on utilise le service prioritaire pour réduire le temps d'attente des clients ou optimiser le critère donné. Différents cas peuvent être distingués[KLE.76].

1) La discipline de la priorité absolue (préemption). Dans ce cas, si un client prioritaire arrive dans le système, le client moins prioritaire dont le service est en cours doit suspendre son service. Le serveur sert immédiatement le client

prioritaire. Après avoir servi ce client, il continue à servir le client suspendu.

2) La discipline de la priorité ordonnée (non-préemption). Quand le client prioritaire arrive dans le système, le client moins prioritaire en cours de service peut continuer et terminer son service, puis le serveur sert immédiatement le plus prioritaire.

3) La discipline de la priorité mixte. Si lorsque le client prioritaire arrive dans le système, le temps restant à courir pour le service du client en cours est inférieur à un certain seuil, on fait jouer la non-préemption. Dans le cas contraire, le service en cours est interrompu pour servir immédiatement le client prioritaire.

La discipline de priorité que nous utilisons dans ce chapitre est toujours non-préemptive.

Pour définir la priorité de la file d'attente, en général, il faut établir certaines fonctions associées aux clients pour comparer les degrés de priorité des clients[BEL.73]. Dans la cellule que nous envisageons, chaque tâche exécutable réalisée peut apporter une certaine quantité de travail, que nous appelons le gain du système. Le gain peut être défini par les tâches pré-exécutables par rapport aux tâches antécédentes et les tâches pré-exécutables par rapport aux ressources. Pour déterminer la priorité des tâches, nous comparons la valeur du gain associé à chacune des tâches strictement exécutables. Parmi les tâches dans la file à chaque instant t, la tâche la plus prioritaire est celle qui apporte le gain maximal. Pour définir le gain associé au choix d'une tâche i, il faut calculer l'évolution résultante des ensembles PEA, PER, et EXE, qui permettra la connaissance de la quantité de travail entrant dans la file.

4.4 - Evolution de PEA

PEA(t) évolue si et seulement si des tâches se terminent puisque l'ensemble des successeurs de la tâche terminée voient leur coefficient d'appartenance à PEA augmenter.

$PEA(t_o)$ ensemble des tâches préexécutables par rapport aux antécédents à l'instant t_o

$PEA(t_i)$ ensemble des tâches préexécutables par rapport aux antécédents à l'instant t_i où la tâche i se termine.

Soient:

S_i ensemble des successeurs de la tâche i

$\bar{S}_i$ les autres tâches

$x \in S_i$ un successeur de i

$S^{-1}(x)$ ensemble des antécédents de x

$S_o^{-1}(x,t)$ ensemble des antécédents de x non réalisés à l'instant t, (chacun des éléments de $S_o^{-1}(x,t)$ a un coefficient d'appartenance à EXE).

$S_1^{-1}(x,t)$ les antécédents de x terminés à t

$$PEA(t_i)=PEA(S_i,t_i)-\{i\} \cup PEA(S_i,t_i)$$
$$=PEA(S_i,t_o)-\{i\} \cup PEA(S_i,t_i)$$

Nous définissons $PEA(S_i,t_i)$ au moyen de δ_o et δ_1 comme suit:

$$x \in S_i \qquad \mu_{PEA(ti)}(x)=f(\delta_o(x,t_i),\delta_1(x,t_i))$$

$$\delta_1(x,t_i)=|S_1^{-1}(x,t_i)|$$
$$\delta_o(x,t_i)=|S_o^{-1}(x,t_i)|$$

La fonction $f(\delta_o,\delta_1)$ vérifie:

1) $0 \leq f(\delta_o,\delta_1) \leq 1$

avec $f(\delta,0)=0$

$f(0,\delta)=1$

$0 < f(\delta_o,\delta_1) < 1$ si $0<\delta_o<\delta$

2) $f(\delta_o,\delta_1)$ est une fonction croissante de δ_1 et on a:

$$x \in S_i \qquad \delta_1(t_i)=\delta_1(t_o)+1$$

La figure 3.6 donne 3 allures possibles de la fonction $f(\delta_o,\delta_1)$. Le cas linéaire correspond à une attitude neutre entre les "mises en chantier" et les "fins de chantier", il s'exprime par:

$$\mu_{PEA(t_i)}(x)=\frac{\delta_1(x,t_i)}{\delta(x)} \quad =\frac{\delta_1(x,t_o)+1}{\delta(x)}$$
$$=\mu_{PEA(t_o)}(x)+\frac{1}{\delta(x)}$$

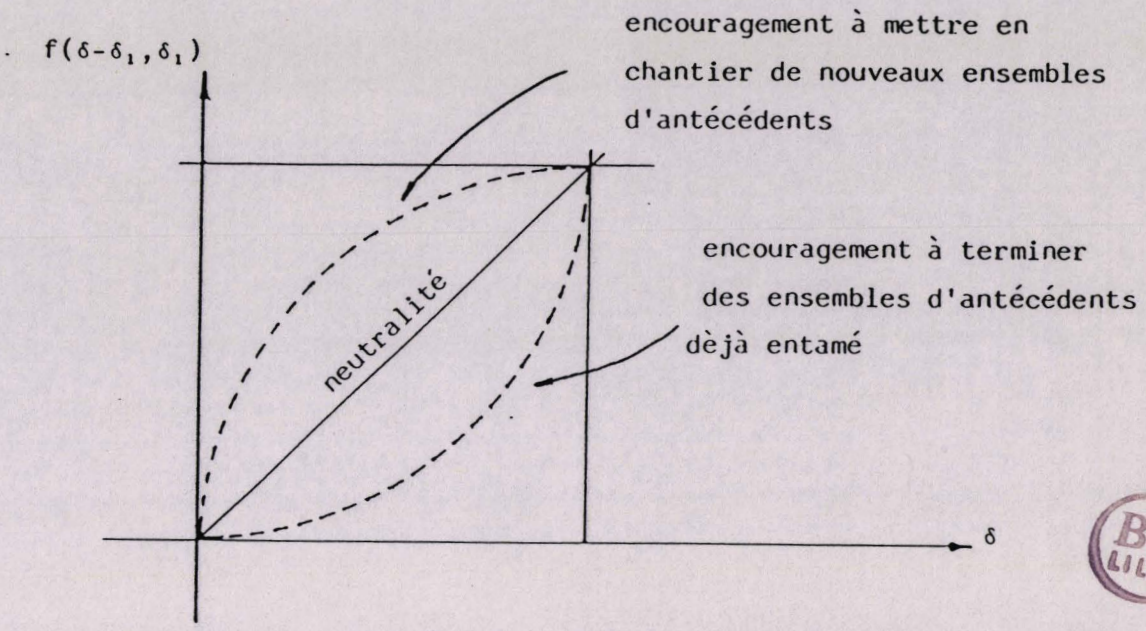


fig.3.6

Résultat:

$PEA(t_i)$:

$x \notin S_i \quad \mu_{PEA(t_i)}(x) = \mu_{PEA(t_o)}(x)$

$x \in S_i \quad \mu_{PEA(t_i)}(x) = \mu_{PEA(t_o)}(x) + \Delta_x$

avec $\Delta_x = \frac{1}{\delta(x)}$ dans le cas de neutralité.

$x=i \quad \mu_{PEA(t_i)}(x) = 0$

4.5 - Evolution de PER

Dans le cas où à une tâche correspond une seule ressource, on définit simplement PER par $\mu_{PER}(x)$=Proba {ressource nécessaire à la réalisation de x soit présente à l'instant t}

L'évolution du vecteur des coefficients d'appartenance à PER se confond donc avec l'évolution du vecteur des probabilités d'arrivée des ressources.

$P(t)=(p_1(t), p_2(t), \cdots p_n(t))$ le vecteur des probabilités, où $p_i(t)$ est la probabilité pour que la ressource i soit présente dans le système à l'instant t.

4.6 - Evolution de EXE

On a $EXE(t)=PEA(t)\cap PER(t)$

Les coefficients d'appartenance se calculent alors simplement sous la forme:

$$\mu_{EXE(t)}(x)=\mu_{PEA(t)}(x)*\mu_{PER(t)}(x) \qquad (3.8)$$

où * est une opération vérifiant:

$a*1=1*a=a \qquad a\in[0,1]$

On pourra prendre par exemple *=min ou *=·.

4.7 - Gain du système

Soit i une tâche candidate et dont la durée de réalisation est T_i. Pour évaluer la quantité de travail apportée dans la file d'attente par les tâches qui deviennent exécutables à $t+T_i$,(évolution de EXE), deux cas sont à distinguer:

a) $j\in\underline{EXE}(t+T_i)$ La tâche j pourra effectivement être choisie pour être exécutée à $t+T_i$. Elle apporte donc une quantité de travail réelle à la file dont la valeur peut être donnée par T_j ou plus généralement une fonction $f(T_j)$.

b) $j\in\underline{\underline{EXE}}(t+T_i)$ La tâche j ne pourra pas être choisie pour être exécutée à $t+T_i$, elle fournit seulement un travail futur à la file, qui est donné par une valeur actualisée de $f(T_j)$.

Actualisation

Nous considérons une tâche j pour laquelle $j\in EXE(t+T_i)$ et soit Δ le temps au bout duquel $\mu_{EXE(t+Ti+\Delta)}(j)=1$. Une approche classique permet de définir le gain actualisé de la tâche j par $f(T_j)e^{-\beta\Delta}j$ où β est un coefficient d'actualisation [KAU.59][BEL.73].

Le problème est alors l'estimation de la valeur de l'intervalle de temps Δ tel que

$$\mu_{PEA(t+T_i+\Delta)}(j) = \mu_{PER(t+T_i+\Delta)}(j)=1$$

Au bout de ce délai, la tâche j deviendra exécutable si on réalise la tâche i à l'instant t.

i) On peut connaître le délai au bout duquel $\mu_{PER(t+Ti+\Delta)}(j)=P_j(t+Ti+\Delta)=1$, en effet, $p_j(t)$ est supposée connue, soit Δ_p ce délai.(figure 3.7)

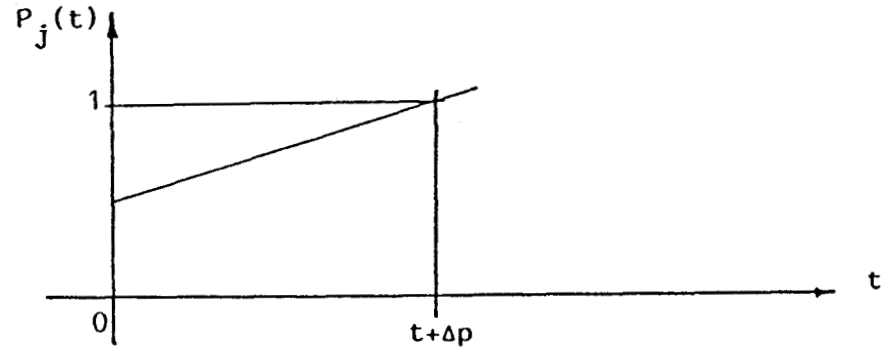


fig.3.7

ii) Le délai au bout duquel $\mu_{PEA(t+Ti+\Delta)}=q_i=1$ dépend des décisions suivantes, en effet, il est nécessaire que l'ensemble des antécédents de j soit exécuté. Ce délai n'est pas connu à l'avance, mais il peut être minoré et majoré.

<u>minoration</u>

Le délai minimal correspond au temps d'exécution des tâches de $A_j(t+T_i)$, ensemble des antécédents de j non encore réalisés à l'instant $t+T_i$.

Soit $\Delta\overline{q}$ ce délai

$$\Delta\overline{q} = \max_{k\in A_j(t+T_i)} T_k$$

<u>majoration</u>

Le délai maximal correspond au cas le plus défavorable pour j, c'est-à-dire le cas où toutes les tâches ne possédant ni j ni un de ses successeurs, comme antécédents seront exécutées avant j.

Soit G(t) l'ensemble de toutes les tâches non réalisées à l'instant t et S_j l'ensemble de tous les successeurs de j.

Le temps d'exécution de toutes les tâches qui peuvent être réalisées avant la tâche j est donné par:

$$T_w = \sum_{k\in G(t)\setminus S_j U(j)} T_k$$

D'autre part, ces tâches doivent éventuellement attendre l'arrivée de ressources pour être exécutées, nous devons donc ajouter un temps d'attente T_a au temps T_w.

Le temps d'attente est calculé à partir des probabilités d'arrivée de ressources.

$$T_a = \sum_{k \in G(t) \setminus S_j U\{j\}} t_{ak}$$

Le délai maximal est alors:

$$\Delta_q^+ = T_w + T_a$$

Conclusion Le délai Δ appartient à l'intervalle suivant:

$$\Delta \in (\max(\Delta_p, \Delta_q^-), \max(\Delta_p, \Delta_q^+))$$

$$= (\Delta_{min}, \Delta_{max})$$

Une estimation de ce délai peut alors être réalisée ainsi:

1. estimation "optimiste" $\Delta = \Delta_{max}$
2. estimation "pessimiste" $\Delta = \Delta_{min}$
3. estimation "moyenne" $\Delta = \alpha \Delta_{min} + \gamma \Delta_{max}$

avec $\alpha + \gamma = 1$, $\alpha, \gamma > 0$

Pour chaque tâche j de couple d'exécutabilité (p_j, q_j) à l'instant t, on peut estimer le délai Δ_j au bout duquel la quantité de travail Q_j sera apportée dans la queue.

Cette tâche procure alors un gain $f(T_j)e^{-\beta\Delta_j}$. Le gain total apporté au système par le choix de i à l'instant t est alors:

$$Q_i(t) = \sum_{j \in S_i} f(T_j)\, e^{-\beta\Delta_j} \qquad (3.9)$$

5 - CAS OU PLUSIEURS TACHES PEUVENT CONSOMMER LA MEME RESSOURCE [YAN.87]

Nous avons défini le gain du système dans le paragraphe précédent et nous avons donné un processus d'actualisation du gain dans le cas où à une tâche correspond une ressource.

Dans ce paragraphe, nous supposerons qu'un même type de ressource peut être consommé par une ou plusieurs tâches à

l'instant t. Dans ce cas le coefficient d'appartenance à l'ensemble des tâches préexécutables par rapport aux ressources n'est pas celui du cas précédent. Il dépend du nombre de tâches, du nombre de ressources, etc...

A partir de PEA(t),PER(t), nous allons analyser l'évolution des tâches exécutables, c'est-à-dire appartenant à l'ensemble:

$$EXE(t)=PEA(t)\cap PER(t)$$

5.1 - Evolution de PEA

L'évolution de PEA(t) ne dépendant que des tâches terminées et de la donnée du graphe est identique à celle observée dans le cas précédent.

5.2 - Evolution de PER

Plusieurs tâches peuvent nécessiter un même type de ressources. Ces tâches appartiendront à l'ensemble PER(t) dès qu'il y aura au moins une ressource de ce type dans la cellule.

A l'instant t_o, nous connaissons le nombre et le type des ressources présentes dans la cellule. L'ensemble $PER(t_i)$ dépend des arrivées de pièces entre t_o et t_i dont nous connaissons les probabilités. Dans ce sens, l'ensemble PER(t) est défini comme un ensemble flou:

Soit x_k une tâche nécessitant une ressource de type k

à l'instant t_o,

$\mu_{PER(t_o)}(x_k)=1$ s'il existe au moins une ressource de type k dans la cellule.

$\mu_{PER(t_o)}(x_k)=0$ autrement

- Soit x_i une tâche candidate, utilisant une ressource de type 1 et dont la date de fin est t_i. On note X_1 l'ensemble des tâches consommant la ressource 1.

Pour toutes les tâches ne nécessitant pas une ressource de type 1, $(x_k \notin X_1)$, on a:

$$\mu_{PER(t_i)}(x_k)=1 \quad \text{si} \quad \mu_{PER(t_o)}(x_k)=1$$

$$\mu_{PER(t_i)}(x_k)=prob(n_k(t_i)\geq 1) \quad \text{si} \quad \mu_{PER(t_o)}(x_k)=0$$

avec $prob(n_k(t_i)\geq 1)$, la probabilité pour que le nombre de ressources du type k consommé par x_k soit supérieur à 1 à l'instant t_i.

- Pour les tâches utilisant une ressource de type $l(x_k \in X_l)$, on a

- s'il existe au moins deux ressources de type l à l'instant t_o

$$\mu_{PER(t_i)}(x_k) = \mu_{PER(t_o)}(x_k) = 1$$

- s'il n'existe qu'une ressource de type l à l'instant t_o

$$\mu_{PER(t_i)}(x_k) = prob(n_k(t_i)\geq 1)$$

Remarque: Pour chaque type de ressource, on peut s'intéresser au stock présent de cette ressource. En notant:

$n_k(t_o)$, nombre de ressources de type k présentes à t_o,

$n_k(t_i)$, nombre de ressources de type k présentes à t_i.

et $\bar{n}_k(t_i)$, nombre moyen de ressources de type k présentes à t_i, on a:

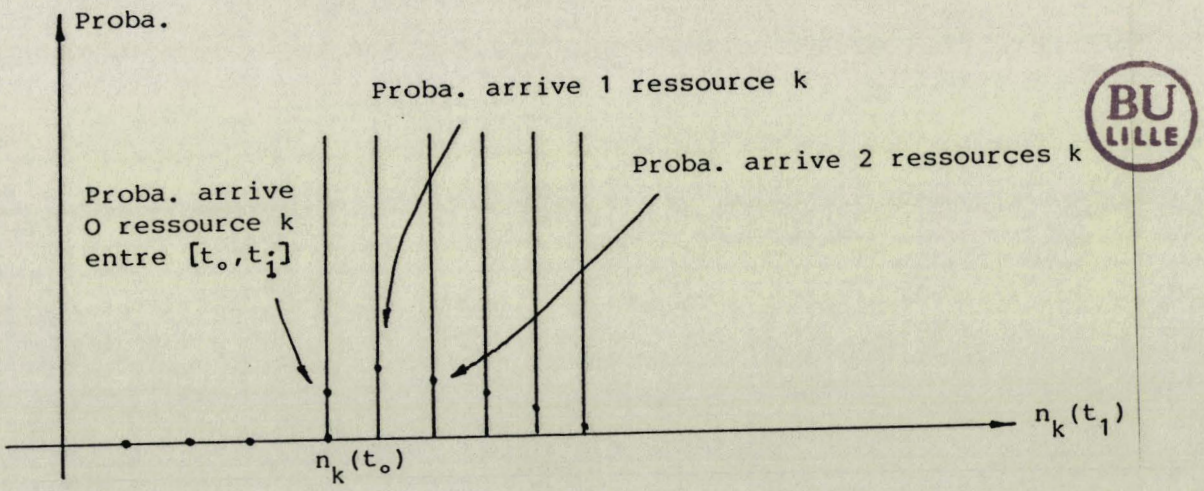


fig.3.8

1) $x \notin X_l$

$$\bar{n}_k(t_i) = n_k(t_o)P_o + (n_k(t_o)+1)P_1 + \cdots + (n_k(t_o)+q)P_q$$

$$\bar{n}_k(t_i) = n_k(t_o) + P_1 + 2P_2 + \cdots + qP_q$$

où P_j, $j=1,2,\cdots q$, sont les probabilités qu'il arrive j ressources de type k entre t_o et t_1.

2) $x \in X_1$

Ces x voient disparaître de façon certaine la ressource de type l utilisée par i.

D'autre part, des ressources peuvent apparaître dans la cellule entre t_o et t_i. On a:

$n_l(t_o)$=nombre de ressources de type l présentes à t_o.

$n_l(t_i)$=nombre aléatoire de ressources qui seront présentes à t_i.

$$\bar{n}_l(t_i)=n_l(t_o)-1+P_1+2P_2+\cdots+qP_q$$

Conclusion:

Nous savons évaluer les ensembles PEA et PER, compte tenu des différents événements intervenant dans le système:

- arrivée d'une ressource
- exécution d'une tâche

nous pouvons en déduire l'évaluation de l'ensemble EXE(t).

On a $EXE(t)=PEA(t) \cap PER(t)$

On définit alors le coefficient d'appartenance d'une tâche à l'ensemble EXE(t) par

$$\mu_{EXE(t)}(x)=\mu_{PEA(t)}(x) * \mu_{PER(t)}(x)$$

5.3 - Evaluation du gain du système

Nous avons évalué le gain du système dans le paragraphe 4.4 dans le cas où à chaque tâche correspond une ressource. Nous supposons maintenant que plusieurs tâches peuvent consommer un même type de ressource. L'évaluation du gain dans ce cas est analysée dans ce paragraphe.

Soit $\underline{EXE}(t_o)$ l'ensemble des tâches strictement exécutables à l'instant t_o.

Le problème devient:

$$y \in \underline{EXE}(t_o)$$

Quelle est la quantité de travail apportée par la tâche y ? C'est-à-dire la quantité de travail à $t_i=t_o+T_y$.

Pour chaque type de ressource k$\in$R, on peut calculer le nombre moyen de ressources $\bar{n}_k(t_i)$ ou la probabilité que n_k ressources soient présentes dans le système à l'instant t_i, où Prob$\{n_k(t_i)\geq 1\}$ à l'instant t_i. On connait également S_y, l'ensemble des successeurs de y qui utilisent la ressource k.

On sait calculer PEA(t_i) et PER(t_i), on a alors:

$$EXE(t_i)=PEA(t_i)\cap PER(t_i)$$

Pour évaluer la quantité de travail, nous pouvons encore écrire:

$$EXE(t_i)=\underline{EXE}(t_i)\cup\underset{\sim}{EXE}(t_i)$$

$\underline{EXE}(t_i)$ correspond au travail présent à t_i

$\underset{\sim}{EXE}(t_i)$ correspond au travail attendu.

5.3.1 Quantité de travail présente dans EXE(t_i)

$$\underline{EXE}(t_i)=\bigcup_{k\in R}\underline{EXE}(t_i,k)$$

EXE(t_i,k) l'ensemble des tâches strictement exécutables à l'instant t_i et utilisant la même ressource k.

Soit $N(t_i,k)=|\underline{EXE}(t_i,k)|$

Soit $n_k(t_i)$ le nombre de ressources k présentes à l'instant t_i. C'est un nombre aléatoire, de moyenne $\bar{n}_k(t_i)$.

La quantité de travail apportée par EXE(t_i) est:

$$Q(EXE(t_i))=\sum_{k\in R}Q(EXE(t_i,k)) \qquad (3.10)$$

Il s'agit donc de calculer Q(EXE(t_i,k)), c'est-à-dire la quantité de travail présente lorsque N(t_i,k) tâches sont en compétition pour $n_k(t_i)$ ressources.

$$Q(EXE(t_i,k))=\varphi(N(t_i,k),n_k(t_i)) \qquad (3.11)$$

où N(t_i,k) est déterministe

$n_k(t_i)$ est aléatoire, cette valeur peut être estimée par les probabilités ou la moyenne.

a) en probabilité

On note P_j ($j=0,1,2,\cdots$) la probabilité qu'il arrive j ressources de type k au système entre t_o et t_i.

On aura alors $n_k(t_i)=n_k(t_o)$ avec la probabilité P_o. Le gain correspondant sera alors $\varphi(N(t_i,k),n_k(t_o))$. De même, avec la probabilité P_1 on aura $n_k(t_i)=n_k(t_o)+1$ et le gain $\varphi(N(t_i,k),n_k(t_o)+1)$, etc$\cdots$. Pour simplifier les notations on pose $N=N(t_i,k)$ $n_k^o=n_k(t_o)$; $n_k=n_k^1(t_i)$.

La fonction $\varphi(N,n_k^1)$ est aléatoire, on peut en calculer la moyenne:

$$E[\varphi(N,n)]=\sum_{i=0}^{\infty} P_i\,\varphi(N,n_k+i) \qquad (3.12)$$

Pour ce faire, il faut définir la fonction $\varphi(N,i)$, N et i étant des entiers.

b) en moyenne

On définit:

$$Q(EXE(t_i,k))=\varphi(N,\bar{n}_k^1) \qquad (3.13)$$

où $\bar{n}_k^1$ est la moyenne de n_k^1, il faut alors définir $\varphi(N,n_k)$ où $\bar{n}_k^1$ n'est pas forcement un entier.

Dans les deux cas, il convient de définir la fonction $\varphi(N,n)$ quantité de travail présente lorsque N tâches strictement exécutables sont en compétition pour n ressources supposées strictement présentes.

Cette fonction est telle que:

$$\varphi(N,0)=0 \qquad \forall N$$

$$\varphi(0,n)=0 \qquad \forall n$$

$$\varphi(N,n)=\sum_{\underline{EXE}(t_1,k)} f(t_j) \qquad \forall (N,n) \text{ tel que } n\geq N$$

Il reste à définir $\varphi(N,n)$ lorsque N>n, c'est-à-dire lorsque seule une partie des tâches candidates pourront recevoir une ressource. Il est alors évident que:

$$\varphi(N,n)=\max_{\mathcal{P}_n(A)} \sum_{(\underline{EXE}(t_i,k)} f(Tj) \qquad (3.14)$$

où $\mathcal{P}_n(A)$ est l'ensemble des parties à n éléments de l'ensemble A .

Ce problème d'optimisation se résout très simplement: il suffit de classer par ordre décroissant les $f(T_j)$ des tâches de $\underline{EXE}(t_i,k)$ et de prendre les n premières.

Appliquons ceci aux cas précédents.

Le cas a) correspond au seul mode de calcul justifiable. $\varphi(N,i)$ étant défini d'après ce qui précède seulement pour i entier, le calcul suivant a) ne pose donc aucun problème.

Pour le cas b), ce mode de calcul n'a aucune justification sauf la simplicité et la rapidité. (Pour le justifier, il faudrait que φ soit linéaire par rapport à n $\varphi(N,P_i)=P_i\varphi(N,i)$)

L'utilisation du mode de calcul b) implique une extension de la définition de la fonction à l'ensemble des n réels.

Notons $n=E(n)+D(n)$

où E(n) est la partie entière de n,
et D(n) est sa partie décimale.

On étend la définition de la façon suivante:

$$\varphi(N,n)=\varphi(N,E(n))+D(n)\cdot f(T_{E(n)+1}) \qquad (3.15)$$

les T_j sont numérotées selon l'ordre $T1\geq T2\geq T3\geq\cdots$ etc.

5.3.2 Quantité de travail attendue dans $EXE(t_i)$

Notons $\underset{\sim}{EXE}(t_i,k)$ ensemble des tâches j telles que: $0<\mu_{EXE(t_1)}(j)<1$ et utilisant la même ressource de type k à l'instant t_i.

On a: $\underset{\sim}{EXE}(t_i)=\bigcup_{k\in R}\underset{\sim}{EXE}(t_i,k)$

On note $N_k(t_i)=|EXE(t_i,k)|$ Nombre de tâches k pour lesquelles $0<\mu_{EXE(t_i,k)}(x)<1$.

L'ensemble de toutes les tâches en compétition à t_i pour la ressource k est :

$\underline{EXE}(t_i,k)$ $\underset{\sim}{EXE}(t_i,k)$

avec $|\underline{EXE}(t_i,k) \cup \underset{\sim}{EXE}(t_i,k)| = N(t_i,k) + \underset{\sim}{N}(t_i,k)$

et $n_k(t_i)$ nombre de ressources de type k présentes à t_i.

a) - $n_k(t_i) \leq N(t_i,k)$: les ressources sont attribuées aux tâches de $\underline{EXE}(t_i,k)$ d'où on aura:

$$Q(\underset{\sim}{EXE}(t_1,k)=0$$

b) - $n_k > N(t_i,k)$

Soit $\lambda_k(t_i)=n_k(t_i)-N(t_i,k)$, le nombre des ressources en surnombre. Il convient de calculer la quantité de travail apportée par l'affectation des $\lambda_k(t_i)$ ressources aux $\underset{\sim}{N}(t_i,k)$ tâches qui deviendront strictement exécutables.

Si une tâche j devient exécutable strictement au bout d'un temps $_j$, la quantité de travail qu'elle apporte, $f(T_j)$ ne deviendra disponible qu'au bout du temps Δ_j, nous la comptabilisons alors sous sa forme actualisée:

$$f(T_j)e^{-\beta\Delta_j}$$

Les délais Δ_j peuvent être calculés comme au paragraphe précédent.

Le problème de l'affectation optimale des ressources aux tâches se résout très simplement lorsque $\lambda_k(t_i) \geq \underset{\sim}{N}(t_i,k)$.

Dans ce cas, la quantité de travail apportée est:

$$\sum_{j=1}^{\underset{\sim}{N}(t_i,k)} f(T_j)e^{-\beta\Delta_j}$$

Dans le cas où $\lambda_k(t_i) < \underset{\sim}{N}(t_i,k)$ on peut songer à affecter les ressources aux tâches au fur et à mesure que celles-ci deviennent strictement exécutables. Les tâches étant numérotées dans l'ordre des Δ_j croissants, la quantité de travail apportée est alors:

$$\sum_{j=1}^{\lambda_k(t_i)} f(T_j)e^{-\beta\Delta_j}$$

Une telle procédure n'optimise pas l'affectation des ressources. En fait, l'affectation optimale correspond aux

choix du meilleur sous ensemble de EXE à $\lambda_k(t_i)$ éléments. On classe pour cela les tâches selon les valeurs décroissantes de $f(T_j)e^{-\beta \Delta j}$. Soit $J(\lambda_k(t_i))$ l'ensemble des $\lambda_k(t_i)$ premiers éléments; la quantité de travail optimale s'écrit alors:

$$\sum_{J(\lambda_k(t_i))} f(T_j)e^{-\beta\Delta_j} \qquad (3.16)$$

6 - PROBLEME DE DECISION DANS LE SYSTEME DE FILE D'ATTENTE

Le problème de décision est un problème de commande optimale, ou il s'agit pour un critère donné, d'optimiser une certaine fonction objectif.

Dans le système de file d'attente que nous avons envisagé, l'objectif est la minimisation de la durée de réalisation de n tâches. Le but peut être atteint en minimisant le temps d'inactivité du robot. Pour cela, nous maximisons le nombre des tâches exécutables. Le problème est alors de choisir parmi les tâches exécutables, celle qui permettra au plus grand nombre de tâches de devenir à leur tour exécutables.

La particularité de notre problème concerne la prise en compte de la dynamique du système étudié. On a donc un problème d'ordonnancement dynamique dans la file d'attente. Dans ce sens, l'ordre de service est déterminé par le choix de la tâche la plus prioritaire, c'est-à-dire celle qui débloque la plus grande quantité de travail.

Le problème de décision dans la file est résolu par un exécutif temps réel, composé de deux parties:

- Un système d'information. En effet, pour un état des tâches exécutables à l'instant t, X(t), on doit avoir des informations sur le coefficient d'activabilité des tâches suivantes et sur la probabilité d'arrivée des ressources consommables.

- Un système de décision, qui a pour fonction, le choix parmi les décisions admissibles, de celle qui satisfait le mieux les critères donnés.

L'activation du module de décision est réalisée lorsque le robot est libre, puisqu'on doit alors lui affecter une opération de la file d'attente des tâches exécutables.

Pour choisir une tâche parmi n qui attendent dans la file, on a un grand nombre de paramètres à considérer, ce qui rend ce choix difficile.

L'approche multicritère [ROY.80][LAG.78][LAG.80] peut être utilisée pour ordonnancer les tâches du système.

[STA.85(bis)] a utilisé cette approche pour choisir une tâche parmi plusieurs. Les critères qu'il a considérés sont:

a. $J_1(j)$ nombre de successeurs immédiats de j.

b. $J_2(j)$ nombre d'opération de S(j) dont la ressource propre est disponible à l'instant t.

c. $J_3(j)$ coefficient de déblocage.

d. $J_4(j)$ temps de réalisation de l'opération j.

e. $J_5(j)$ temps d'attente moyen d'une ressource propre par l'opération j.

où j est une tâche exécutable.

Dans le cas d'un système où on ne peut effectuer une analyse quantitative précise, cette méthode est utile. On peut donner des coefficients de pondération aux différents critères, mais la détermination de leur valeur respective est complexe. La méthode est très sensible aux variations des valeurs des poids affectés aux critères.

Dans le système de file d'attente que nous avons modélisé, nous choisissons un critère unique. L'algorithme de décision consiste à maximiser l'ensemble des tâches exécutables à chaque instant t, c'est-à-dire à choisir la tâche qui apporte la quantité de travail maximale à la file.

Nous savons que l'on aboutira à une politique non optimale si l'on ne considère que les tâches strictement exécutables. C'est pourquoi nous considérons EXE comme un ensemble flou qui comprend les tâches effectivement exécutables et celles qui le deviendront dans le futur.

On définit EXE(t) l'ensemble des tâches strictement exécutables à l'instant t.

$$\underline{EXE}(t) \subset EXE(t)$$

L'algorithme de décision calcule le gain du système, c'est-à-dire:

$$j \in \underline{EXE}(t) \qquad Q_j(t) = Q(EXE(t+T_j))$$

La meilleure tâche à l'instant t pour le robot est donnée par:

$$j^* \quad \text{telle que} \quad Q_{j^*}(t) = \max_{j \in \underline{EXE}(t)} Q_j(t) \qquad (3.17)$$

7 - CONCLUSION

En utilisant le modèle de file d'attente introduit au chapitre précédent, nous proposons dans ce chapitre un mode de gestion de la cellule d'assemblage flexible.

L'objectif du fonctionnement de ce système est de minimiser le temps d'exécution de n tâches, cela conduit à minimiser l'oisiveté du robot.

Lorsque toutes les tâches sont indépendantes, c'est-à-dire qu'elles peuvent être réalisées en parallèle, la discipline d'attente est PAPS. Le contrôle du système consiste alors à commander l'arrivée des ressources.

Lorsque les tâches sont liées par des relations de succession, il faut définir la priorité dans la file. Pour cela, à chaque instant, le module de décision choisit parmi les tâches exécutables, celle qui sera effectuée lorsque le robot sera libre. La meilleure tâche est celle qui apporte le gain maximal à la file. Le gain est composé d'une partie de travail immédiate, c'est-à-dire réalisable juste après exécution de cette tâche, et d'une autre partie de travail futur qui dépend des tâches qui seront exécutées, des arrivées de ressources dont on connait la probabilité, de l'utilisation de ces ressources qui peuvent être consommées soit par une tâche unique, soit par différentes tâches en compétition.

CHAPITRE IV

CONTROLE D'UNE CELLULE A DEUX SERVEURS

1 - INTRODUCTION

Le but de notre étude dans ce chapitre est la conduite d'une cellule flexible d'assemblage dans laquelle deux robots coopèrent. La coopération est ici entendue dans le sens suivant: Une partie des tâches à exécuter(les tâches communes) peuvent être réalisées soit par l'un, soit par l'autre des robots. Le critère que nous choisissons est la minimisation du temps d'exécution d'un montage.

Dans ce chapitre, la cellule d'assemblage est modélisée par un système de file d'attente. La gestion de la cellule conduit au problème d'affectation dynamique, dans lequel on affecte les tâches aux deux robots de façon à minimiser leur oisiveté. Pour un modèle standard, par exemple M/M/2, on a prouvé que l'affectation optimale dans le cas dynamique est d'envoyer un client à la plus courte queue à l'instant t où il arrive [EPH.80]. Mais la plupart des systèmes ne suivent pas ce modèle dans les cas concrets.

La cellule d'assemblage peut être considérée comme un modèle M/G/2 sous certaines conditions: on suppose que les tâches exécutables arrivées dans le système respectent une distribution de Poisson et que le temps de service des robots respectent une distribution générale sur un horizon infini. Avec ces hypothèses, nous utilisons la formule de Pollaczek-Khinchin pour calculer les probabilités d'affectation au sein du système. Le critère que nous choisissons est de minimiser le nombre total des clients dans les files d'attente; compte tenu des hypothèses concernant le processus des arrivées de clients, cela correspond à maximiser le taux de service de la cellule.

La cellule d'assemblage que nous envisageons, en général, ne satisfait pas la condition ci-dessus. Cependant, un certain nombre d'informations sont connues: la durée de chaque tâche, les aptitudes des robots,etc... Pour affecter les tâches communes,sous l'hypothèse qu'une tâche consomme une ressource correspondante, les méthodes de PSEP (procédure d'optimisation par séparation et évaluation progressive) ou de programmation dynamique peuvent être utilisées. D'autre part, sous l'hypothèse qu'un même type de ressource peut être utilisé par une ou plusieurs tâches, nous préférons utiliser les résultats du chapitre III, pour choisir une tâche parmi les tâches exécutables lorsqu'un robot est libre. Dans ce cas, il faut traiter les conflits, puisqu'une même tâche ou une même ressource peut être affectée aux deux robots.

2 - DESCRIPTION D'UNE CELLULE FLEXIBLE D'ASSEMBLAGE

La cellule flexible d'assemblage considérée est décrite au paragraphe 5.1 du chapitre II. Nous donnons ici plus de détails :

La cellule est composée de deux robots qui travaillent ensemble sur trois plans de travail, un propre à chacun des robots et un disposé en zone commune. L'arrivée des pièces est aléatoire et est effectuée par un tapis roulant. Un système de vision permet l'identification des pièces qui entrent dans la cellule. La succession des opérations d'assemblage est décrite par un graphe potentiel-tâches. Pour chaque tâche, on connait:

- Le robot qui exécute la tâche, soit: R_1 ou R_2. Dans le cas où la tâche peut être exécutée par l'un ou l'autre des robots, on note $R_1 \cup R_2$

- Le temps d'exécution d'une tâche.

Les différents état d'une tâche sont ceux décrits dans le chapitre I.

Les robots ont deux états:

- libre

- occupé

Le critère que nous avons retenu est la minimisation du temps d'exécution d'un montage.

3 - MODELISATION DE LA CELLULE PAR UN SYSTEME DE FILE D'ATTENTE

La cellule que nous venons de décrire peut être modélisée par un système de file d'attente (chapitre II). Les clients sont les tâches exécutables, les serveurs sont les deux robots.

Nous définissons dans ce chapitre:

α ensemble des tâches exécutées par R_1(robot 1)

β ensemble des tâches exécutées par R_2(robot 2)

γ ensemble des tâches exécutées par un des deux robots ($R1 \cup R2$)

avec $\alpha+\beta+\gamma=n$

n ensemble des tâches d'un montage

Les tâches exécutables qui appartiennent effectivement à α(ou β) peuvent entrer dans la file de R1 (ou R2). Pour les tâches de γ, elles pourront être exécutées par un

des deux robots lorsqu'elles seront exécutables. Dans ce cas, le problème de minimisation du temps d'exécution consiste à répartir les tâches de δ aux deux robots. Nous allons analyser ce problème dans le paragraphe suivant.

4 - PROBLEME D'ALLOCATION DES TACHES EXECUTABLES

La modélisation par un système de file d'attente est largement utilisée pour analyser ou contrôler les problèmes d'allocation. Ceux-ci peuvent être le partage de ressources dans un ordinateur, la régulation du trafic de véhicules, l'augmentation de la productivité ou la minimisation des temps d'exécution sur une ligne de production ou dans un atelier flexible.

Dans ce paragraphe, nous utilisons le modèle de file d'attente pour analyser et contrôler la cellule d'assemblage. Les clients des deux files d'attente sont les tâches exécutables. L'allocation aux serveurs (les robots) doit permettre la minimisation du temps d'oisiveté des robots, ce qui correspond à la minimisation du temps d'exécution.

Dans un premier temps, nous analysons le cas stationnaire dans lequel on suppose que les clients arrivent dans le système en respectant un débit poissonnien. Le temps de service est une distribution générale sur un horizon infini. La répartition des éléments de δ dans les files d'attente de R_1 et R_2 est réalisée selon un critère qui est fonction du nombre moyen de tâches dans les queues [STI.85].

Dans un second temps, nous utilisons les données du système concernant la liste des tâches, les temps d'exécution,... pour répartir les tâches de δ dans les files de R1 et R2. L'affectation de ces tâches en temps réel peut être réalisée par la programmation dynamique ou par une méthode du type P.S.E.P (procédure d'optimisation par séparation et évaluation progressive).

4.1 - Allocation des clients dans le cas stationnaire

Beaucoup de travaux ont été consacrés à ce problème au cours des dernières années [CRA.71] [LIP.71] [LOW.71] [LIP.77][HAL.80] [BEL.83] [LIN.84]. Pour le modèle que nous avons envisagé dans le cas stationnaire, nous avons les hypothèses suivantes.

1) Les tâches exécutables arrivent dans le système avec les taux poissonniens $\lambda_1', \lambda_2', \lambda_3$.

où λ_1' est le taux de tâches α

λ_2' est le taux de tâches β

λ_3 est le taux de tâches δ

2) Chaque robot exécute la tâche avec un temps aléatoire T_i,(i=1,2). On s'intéresse uniquement à un calcul sur le long terme, dans ce sens on considère pour les deux temps de service, leur moyenne et leur variance.

$$E[T_1^j]=\frac{1}{\mu_1} \qquad E[T_1^{j2}]=\frac{b_1}{\mu_2} \qquad \text{pour le robot 1}$$

$$E[T_2^j]=\frac{1}{\mu_2} \qquad E[T_2^{j2}]=\frac{b_2}{\mu_2} \qquad \text{pour le robot 2}$$

où j est une tâche exécutable.
b_i(i=1,2) est constant ($b_i \geq 1$)

Donc, le modèle envisagé est M/G/2 queues sur un horizon infini (fig 4.1). Les tâches de α (ou ß) arrivent dans la queue 1 (ou 2) avec le taux λ_1'(ou λ_2'). Les tâches communes, c'est-à-dire les tâches de δ , arrivent au système avec le taux λ_3. Une tâche dans cette partie, qui arrive à l'instant t par exemple, peut être envoyée à la queue 1 avec la probabilité a (ou à la queue 2 avec 1-a). a est alors la variable de contrôle du système des files d'attente. Les tâches de chaque queue peuvent être servies en PAPS ou en priorité que nous avons étudiée au chapitre III en supposant que les taux de service sont μ_1 et μ_2. La minimisation du temps d'oisiveté des robots consiste à minimiser le nombre moyen des tâches dans le système [LAZ.83] [STI.85] [GHO.85].

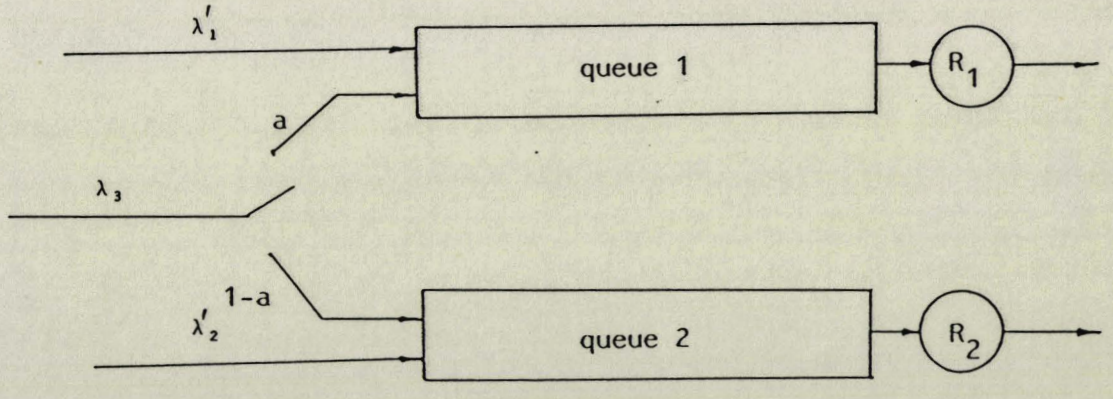


fig. 4.1

Le critère choisi est:

$$\text{minimiser} \quad \sum_{i=1}^{2} L_i(\lambda_i)$$

$L_i(\lambda_i)$ nombre moyen de tâches dans la queue i y compris la tâche en cours.

avec

$$\lambda_1 = \lambda_1' + a\lambda_3$$

$$\lambda_2 = \lambda_2' + (1-a)\lambda_3$$

$$\bar{\lambda} = \lambda_1' + \lambda_2' + \lambda_3 = \lambda_1 + \lambda_2 \qquad 0 \le \lambda_i < \mu_i \quad (i=1,2)$$

$a \ge 0$ la probabilité pour envoyer les tâches de γ à la queue 1.

Le problème de l'allocation statique dans un système général de file d'attente devient très complexe pour i>2. Dans [HAL.76] [BER.83(D.P)], une méthode numérique a été utilisée pour résoudre ce problème avec un algorithme de programmation non-linéaire. Pour notre problème (i=2), nous analysons la quantité de travail pour réaliser l'allocation optimale.

Utilisons la formule de Pollaczek-Khinchin [KLE.76] [COX.61] [GRO.74] qui permet de calculer le nombre moyen de tâches dans le système pour un modèle M/G/1 en supposant que le taux d'arrivée des clients est λ et que le taux de service moyen est μ.

Cette formule est rappelée ci dessous:

$$L(\lambda) = \frac{\lambda}{\mu} + \frac{\lambda^2 b}{2\mu(\mu-\lambda)}$$

Pour notre problème, nous avons:

$$L(\lambda_1) = \frac{\lambda_1}{\mu_1} + \frac{\lambda_1^2 b_1}{2\mu(\mu_1-\lambda_1)}$$

$$L(\lambda_2) = \frac{\lambda_2}{\mu_2} + \frac{\lambda_2^2 b_2}{2\mu(\mu_2-\lambda_2)}$$

Dans le cas stationnaire ($\lambda_1, \lambda_2, \mu_1, \mu_2, b_1, b_2,$ sont constants), la variable est a avec $0 \le a \le 1$.

On note:

$$L_a = L(\lambda_1) + L(\lambda_2) = \frac{\lambda_1}{\mu_1} + \frac{\lambda_1^2 b}{2\mu_1(\mu_1-\lambda_1)} + \frac{\lambda_2}{\mu_2} + \frac{\lambda_2^2 b}{2\mu_2(\mu_2-\lambda_2)} \qquad (4.1)$$

Pour trouver la valeur optimale, on cherche la valeur qui donne $\frac{dL_a}{da} = 0$

C'est-à-dire:

$$\frac{1}{\mu_1}\frac{d\lambda_1}{da} + \frac{b_1 2\lambda_1(\frac{d\lambda_1}{da})2\mu_1(\mu_1-\lambda_1) + 2\mu_1\frac{d\lambda_1}{da}\lambda_1^2 b_1}{4\mu_1^2(\mu_1-\lambda_1)^2}$$

$$+ \frac{1}{\mu_2}\frac{d\lambda_2}{da} + \frac{b_2 2\lambda_2(\frac{d\lambda_2}{da})2\mu_2(\mu_2-\lambda_2) + 2\mu_2\frac{d\lambda_2}{da}\lambda_2^2 b_2}{4\mu_2^2(\mu_2-\lambda_2)^2} = 0$$

$$\frac{d\lambda_1}{da} = \lambda_3 \qquad \frac{d\lambda_2}{da} = -\lambda_3$$

d'où:

$$\lambda_3(\frac{1}{\mu_1} - \frac{1}{\mu_2}) + \frac{(2b_1\lambda_1(\mu_1-\lambda_1) + b_1\lambda_1^2)\mu_1\lambda_3}{2\mu_1(\mu_1-\lambda_1)^2}$$

$$- \frac{(2b_2\lambda_2(\mu_2-\lambda_2) + b_2\lambda_2^2)\mu_2\lambda_3}{2\mu_2(\mu_2-\lambda_2)^2} = 0$$

En simplifiant λ_3, on obtient:

$$\left(\frac{1}{\mu_1} - \frac{1}{\mu_2}\right) + \frac{(2b_1\lambda_1(\mu_1-\lambda_1) + b_1\lambda_1^2)\mu_1}{2\mu_1(\mu_1-\lambda_1)^2}$$

$$- \frac{(2b_2\lambda_2(\mu_2-\lambda_2) + b_2\lambda_2^2)\mu_2}{2\mu_2(\mu_2-\lambda_2)^2} = 0$$

C'est une équation de degré 4 en a .Elle peut être résolue numériquement en utilisant des procédures classiques (algorithme de Newton Raphson par exemple). La résolution analytique est possible facilement sous des hypothèses simplificatrices par exemple dans le cas particulier $\mu_1=\mu_2=\mu$ $b_1=b_2=b$.

Dans ce cas on obtient:

$$\frac{2\lambda_1(\mu-\lambda_1) + \lambda_1^2}{(\mu-\lambda_1)^2} - \frac{2\lambda_2(\mu-\lambda_2) + \lambda_2^2}{(\mu-\lambda_2)^2} = 0$$

$\lambda_1+\lambda_2=\bar{\lambda}$ constante $\lambda_1,\lambda_2 \geq 0$. La solution optimale est $\lambda_1=\lambda_2$.

Preuve:

$$L_a(\lambda_1=\lambda_2=\lambda) = \frac{2\lambda}{\mu} + \frac{b\lambda^2}{\mu(\mu-\lambda)}$$

$\forall\ \lambda_1,\lambda_2 \quad \lambda_1 \neq \lambda_2$

$$L_a(\lambda_1 \neq \lambda_2) = \frac{\lambda_1}{\mu} + \frac{\lambda_1^2 b}{2\mu(\mu-\lambda_1)} + \frac{\lambda_2}{\mu} + \frac{\lambda_2^2 b}{2\mu(\mu-\lambda_2)}$$

$$= \frac{\lambda_1+\lambda_2}{\mu} + \frac{b}{2\mu}\left(\frac{\lambda_1^2}{\mu-\lambda_1} + \frac{\lambda_2^2}{\mu-\lambda_2}\right)$$

$$\lambda_1+\lambda_2 = 2\lambda$$

Donc $L_a(\lambda_1 \neq \lambda_2) - L_a(\lambda_1 = \lambda_2)$

$$\frac{b}{2\mu}\left(\frac{\lambda_1^2}{\mu-\lambda_1} + \frac{\lambda_2^2}{\mu-\lambda_2} - \frac{2\lambda^2}{\mu-\lambda}\right)$$

On a $\frac{b}{2\mu} > 0$

et

$$\frac{\lambda_1^2(\mu-\lambda_2)+\lambda_2^2(\mu-\lambda_1)}{(\mu-\lambda_1)(\mu-\lambda_2)} - \frac{2\lambda^2(\mu-\lambda)}{(\mu-\lambda)^2} \Big| >$$

$$\frac{\lambda_1^2(\mu-\lambda_2)+\lambda_2^2(\mu-\lambda_1)-2\lambda^2(\mu-\lambda)}{(\mu-\lambda)^2}$$

Puisque $\mu^2-(\lambda_1+\lambda_2)\mu+\lambda_1\lambda_2 < \mu^2-2\lambda\mu+\lambda^2$

$(\lambda-\mu)^2>0$ on a:

$$\lambda_1^2(\mu-\lambda_2)+\lambda_2^2(\mu-\lambda_1)-2\lambda^2(\mu-\lambda)$$
$$= 2\mu\lambda^2+2\lambda^3-2\lambda_1\lambda_2(\mu+\lambda)$$
$$= 2\mu\frac{(\lambda_1+\lambda_2)^2}{4}+\frac{2(\lambda_1+\lambda_2)^3}{8}-2\lambda_1\lambda_2\frac{\lambda_1+\lambda_2}{2}-2\lambda_1\lambda_2\mu$$
$$= \frac{\mu}{2}(\lambda_1-\lambda_2)^2+\frac{1}{4}(\lambda_1+\lambda_2)(\lambda_1-\lambda_2)^2$$
$$= (\lambda_1-\lambda_2)^2\left(\frac{\mu}{2}+\frac{1}{4}(\lambda_1+\lambda_2)\right)>0$$

Donc $\lambda_1=\lambda_2=\lambda$ est la solution optimale pour minimiser $L_a(\lambda)$.

On a $\lambda_1'+a\lambda_3 = \lambda_2'+(1-a)\lambda_3$

$$a=\frac{\lambda_3+\lambda_2'-\lambda_1'}{2\lambda_3}$$

i) si $\lambda_1'=\lambda_2'$ $\quad a=\frac{1}{2}$

ii) si $\frac{\lambda_3+\lambda'_2-\lambda'_1}{2\lambda_3} > 1$ a=1

iii) si $\lambda_3+\lambda'_2-\lambda'_1 \leq 0$ a=0

Conclusion

Connaissant λ'_1, λ'_2, λ_3, on calcule a et on affecte les tâches de aux queues 1 et 2 selon la probabilité a,(1-a) avec:

$a= \frac{\lambda_3+\lambda'_2-\lambda'_1}{2\lambda_3}$ si cette valeur est comprise entre 0 et 1.

a=1 si $\frac{\lambda_3+\lambda'_2-\lambda'_1}{2\lambda_3} > 1$

a=0 si $\lambda_3+\lambda'_2-\lambda'_1 \leq 0$

4.2 - Allocation des clients(tâches exécutables) dans le cas dynamique

Pour réaliser l'allocation optimale dans le cas dynamique, la politique qui consiste à envoyer les clients à la plus courte queue peut être utilisée. C'est une politique optimale dans un cas très spécial: le modèle M/M/2 [WIN.77] [WEB.78] [EPH.80]. Dans le cas où $\mu_1 \neq \mu_2$,[W.LIN][LIN.84] et [B.HAJEK][HAJ.84] ont étudié le modèle M/M/2. Ils ont montré que la politique optimale dans le système est celle du type seuil, on peut fournir un client au serveur le plus rapide s'il est disponible pour le service, mais le serveur le plus lent sera utilisé si et seulement si la longueur de la queue n'excède pas une valeur de seuil, le calcul est donc facile.

La cellule flexible d'assemblage est un système très complexe. Le taux d'arrivée des tâches exécutables n'est peut-être pas poissonnien, d'autres données peuvent être très utiles (par exemple, le temps de service des tâches est déterministe, et les tâches successeurs d'une tâche sont connues,...). Pour contrôler le système en temps réel, nous utilisons donc les méthodes suivantes pour affecter les tâches de ϑ au robot 1 ou robot 2.

1) La méthode d'optimisation pour affecter les tâches de ϑ à la queue 1 ou à la queue 2 à l'instant t.

hypothèses:

a) La longueur de travail (en temps) est connue pour la queue i (i=1,2), y compris le temps de service.

$L_t=(L_t^1,L_t^2)$ Si Δ_j^i est la durée d'exécution de la tâche j par le robot i on a:

$L_t^1=\Sigma\Delta_j^1+\Delta_{j'}^1$,

$j\in q_1$ et j':tâche en cours

$L_t^i(i=1,2)$ la longueur de travail(en temps) dans la queue i à l'instant t.

b) Les tâches exécutables à l'instant t $m=m_1Um_2Um_3$ sont connues avec les longueurs de travail:

$l_t(m_1)$, $l_t(m_2)$, $l_t(m_3)$

$m_1\in\alpha$, $m_2\in\beta$, $m_3\in\delta$

c) Les tâches qui appartiennent à m_3 peuvent être transférées d'une queue à l'autre.

d) Les deux robots ont la même capacité de travail.

e) Une tâche utilise une ressource correspondante.

Problème: trouver une partition optimale de l'ensemble m_3 en deux classes $m_3(R_1)$, $m_3(R_2)$

telle que: $l_t(m_1)+l_t(m_3(R_1))=L_t^1$

$l_t(m_2)+l_t(m_3(R_2))=L_t^2$

Critère: minimiser le temps d'exécution des n tâches d'un montage réalisé par deux robots.

2) A chaque instant t où un robot est libre, il choisit la meilleure tâche à exécuter.

hypothèses:

a) Les tâches exécutables à l'instant t sont $m=m_1Um_2Um_3$, $m_1\in\alpha$, $m_2\in\beta$, $m_3\in\delta$.

b) Les tâches de m_3 peuvent être choisies par les deux robots à l'instant t. C'est-à-dire que nous avons:

$M_t^1=\{m_1\}U\{m_3\}$

$M_t^2=\{m_2\}U\{m_3\}$

M_t^i (i=1,2): Tâches exécutables de la queue i à l'instant t.

c) Plusieurs tâches peuvent consommer un même type de ressource.

Problème: trouver la meilleure tâche à faire exécuter au robot qui est libre.

On choisit le même critère que dans 1).

Cette méthode est présentée dans le paragraphe §5.

4.2.1 - Méthodes d'optimisation pour affecter les tâches

Dans ce paragraphe, nous utilisons les méthodes d'optimisation (la méthode d'optimisation par séparation et évaluation(PSEP) et la méthode de programmation dynamique) pour affecter les tâches de δ en temps réel. Un résultat sous optimal peut être obtenu.

Dans un premier temps, nous supposons que la situation des tâches suivantes ne peut pas être prévue, puis nous estimons un pas des tâches suivantes.

1 -Politique optimale sans prédiction de la situation des tâches suivantes

Dans ce cas, il faut affecter les tâches exécutables présentes dans le système.

formulation du problème

La longueur du travail à l'instant t est L_t.

$$L_t=(L_t^1,L_t^2)$$

$$L_t^1=l_t(m_1)+l_t(m_3(R_1))$$

$$L_t^2=l_t(m_2)+l_t(m_3(R_2))$$

$$l_t(m_3)=\sigma_1+\sigma_2+\cdots+\sigma_n=\sum_{k=1}^{n}\sigma_k$$

σ_k la durée d'exécution de la $k^{ième}$ tâche de m_3.

$$L=L_t^1+L_t^2$$

Le temps d'exécution minimal pour deux robots est:

$$J^* = \min\ (\max\ \{L_t^1,L_t^2\}) \geq \frac{L}{2}$$

Preuve: On suppose $L_t^1 \leq L_t^2$

$$J^*=L_t^2 = L_t^1+\Delta$$

$$2J^*=L+\Delta$$

$$J^*= \frac{L}{2} + \frac{\Delta}{2} \geq \frac{L}{2}$$

$\forall\ L_t^1, L_t^2$

$$\max\{L_t^1, L_t^2\} = \frac{L_t^1+L_t^2}{2} + \frac{|L_t^1-L_t^2|}{2}$$

$$= \frac{L}{2} + \frac{|L_t^1-L_t^2|}{2}$$

d'où $\min(\max\{L_t^1, L_t^2\}) \Longleftrightarrow \min|L_t^1-L_t^2|$

Décision: $u=(u_1,u_2,\cdots,u_k,\cdots,u_n)$

avec $u_k=1$ si σ_k est affectée à la queue 1,

$u_k=0$ si σ_k est affectée à la queue 2.

Le critère est $J^* = \min|L_t^1-L_t^2|$

On a $L_t^1 = l_t(m_1) + \sum_{k=1}^{n} \sigma_k u_k$

$$L_t^2 = l_t(m_2) + \sum_{k=1}^{n} \sigma_k - \sum_{k=1}^{n} u_k\sigma_k$$

d'où $L_t^1-L_t^2 = l_t(m_1)+2\sum_{k=1}^{n} u_k\sigma_k - (l_t(m_2)+\sum_{k=1}^{n}\sigma_k)$

$$= 2l_t(m_1)+2\sum_{k=1}^{n} u_k\sigma_k - L$$

puisque $L = l_t(m_1) + l_t(m_2) + \sum_{k=1}^{n}\sigma_k$

On a donc $J^* = \min_{u\in\{0,1\}^n} \left|l_t(m_1) + \sum_{k=1}^{n} u_k\sigma_k - \frac{L}{2}\right|$ (4.2)

Conclusion: A chaque instant t, on désire affecter les tâches de m_3 aux queues 1 et 2 de façon à minimiser le critère. On peut utiliser pour cela la méthode P.S.E.P ou la programmation dynamique.

a) la méthode P.S.E.P [ROY.69]

Cette méthode consiste à parcourir l'ordre des solutions possibles et à choisir le chemin optimal.

Nous définissons:

$$L_t^1(0) = l_t(m_1) \qquad L_t^2(0) = l_t(m_2)$$

La racine de l'ensemble des sommets est $L_t^1(0)=l_t(m_1)$. Le principe est la séparation de chaque sommet de l'ordre par 1 ou 0.

Arborescence:

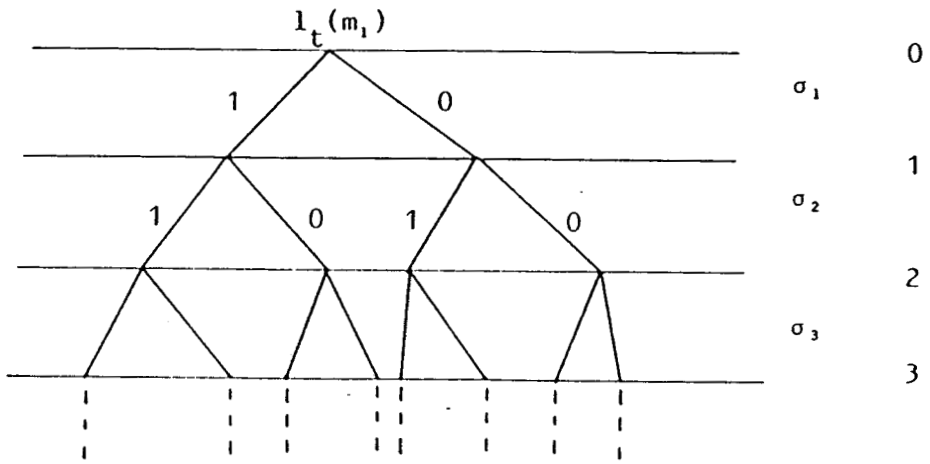


fig.4.2

On note $$v_i=l_t(m_1) + \sum_{k=1}^{i} \sigma_k \qquad (4.3)$$

$$v_o=l_t(m_1)$$

$$w_i=\sum_{i+2}^{n} u_k\sigma_k \qquad (4.4)$$

Algorithme 1 On cherche d'abord la meilleure solution parmi l'ensemble des solutions V:

$$v_t=\{u/u=(1,1,\cdots,\underset{\uparrow}{1},0,\cdots 0),\ i\in\{1,\cdots n\}\}$$
$$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i$$

Pour cela,on choisit i tel que:

$$|v_i - \frac{L}{2}| \leq |v_{i-1} - \frac{L}{2}|$$

et

$$|v_i - \frac{L}{2}| \leq |v_{i+1} - \frac{L}{2}|$$

$$|v_i - \frac{L}{2}| = \begin{cases} \Delta_1^- & \text{si } v_i \leq \frac{L}{2} \\ \Delta_1^+ & \text{si } v_i \geq \frac{L}{2} \end{cases} \qquad (4.5)$$

Si $\Delta_1 = 0$ $\quad u^* = (1,1,\cdots \underset{\uparrow}{1},0,0,\cdots 0)$
$i^{ème}$ position

u^* est la décision optimale.

Si $\Delta_1^- \neq 0$ et $\Delta_1^+ \neq 0$, on trouve la première contrainte:

$$|l_t(m_1) + \sum_{k=1}^{n} \sigma_k u_k - \frac{L}{2}| \leq \Delta_1 \quad \Delta_1 \in (\Delta_1^+, \Delta_1^-)$$

puis on élimine la branche suivante et on considère le sous ensemble de sommet i (si $\Delta_1 = \Delta_1^-$) ou i-1 (si $\Delta_1 = \Delta_1^+$).

Par exemple, si $\Delta_1 = \Delta_1^-$, la racine du sous ensemble de sommet i est

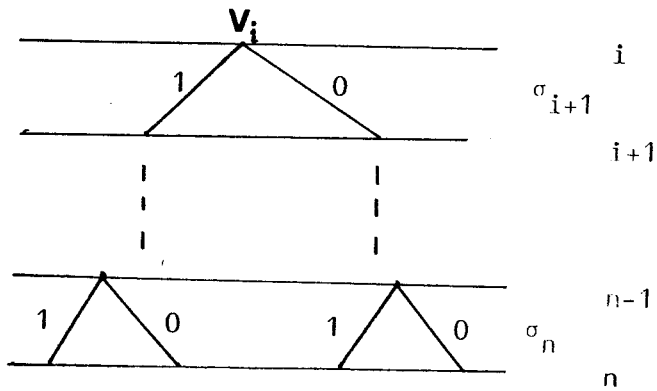


fig.4.3

Nous utilisons le même processus pour trouver Δ_2 tel que:

$$\Delta_2 = \left| v_i + w_i - \frac{L}{2} \right|$$

1) $\Delta_2 = 0$ la solution optimale est :

$$u = (1, \cdots \underset{i}{1}, \underset{i+1}{0}, u_{i+2}, u_{i+3}, \cdots u_n)$$

2) $\Delta_2 < \Delta_1$ la contrainte devient:

$$\left| l_t(m_1) + \sum_{k=1}^{n} u_k \sigma_k - \frac{L}{2} \right| \leq \Delta_2$$

3) $\Delta_2 \geq \Delta_1$ on garde la première contrainte. La décision suivante est $u = (1, \cdots \underset{i-1}{1}, \underset{i}{0}, u_{i+1}, u_{i+2}, \cdots u_n)$.

Dans les deux derniers cas, il faut continuer à examiner les sous ensembles de sommets (i-1,i-2,...2,1,0)pour trouver la solution optimale.

exemple: $l_t(m_1)=3 \quad l_t(m_2)=2$

$$l_t(m_3) = \sigma_1 + \sigma_2 + \sigma_3 + \sigma_4 + \sigma_5 = 3+3+2+3+4 = 15$$

$$L = l_t(m_1) + l_t(m_2) + l_t(m_3) = 20$$

$$\frac{L}{2} = 10$$

$$J^* = \min_{u \in \{0,1\}^5} \left| 3 + \sum_{1}^{5} \sigma_k u_k - 10 \right|$$

1) On cherche d'abord la meilleure solution parmi l'ensemble des solutions V.

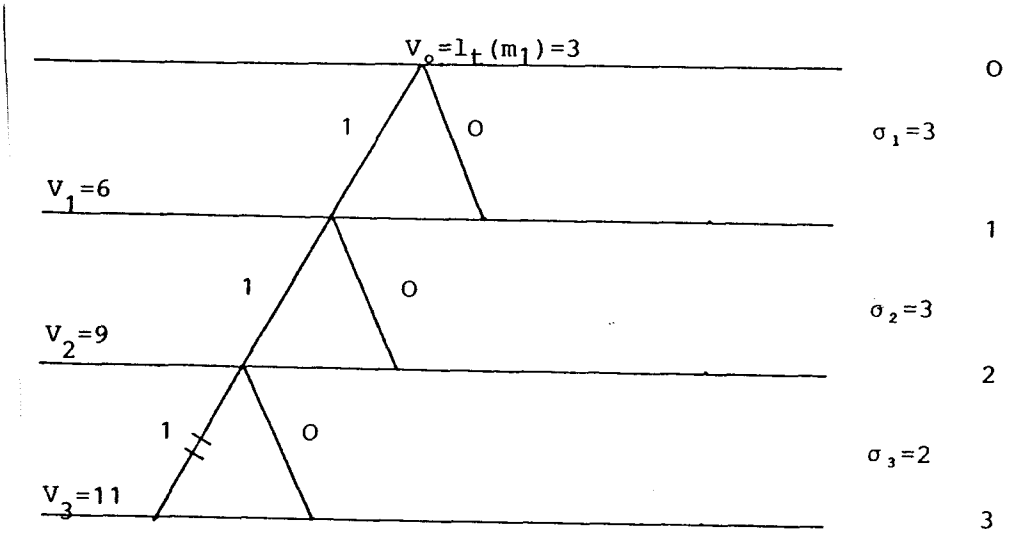


fig.4.4

v	v_o	v_1	v_2	v_3	v_4	v_5
v_k	3	6	9	11		
$\lvert v_k - \frac{L}{2} \rvert$	7	4	1	1		

La décision est u=(1,1,0,0,0) pour obtenir la première contrainte $\Delta_1=\Delta_1^-=1$

2) En éliminant la branche suivante, on recommence à partir du sommet 2, la décision devient $u=(1,1,0,u_4,u_5)$

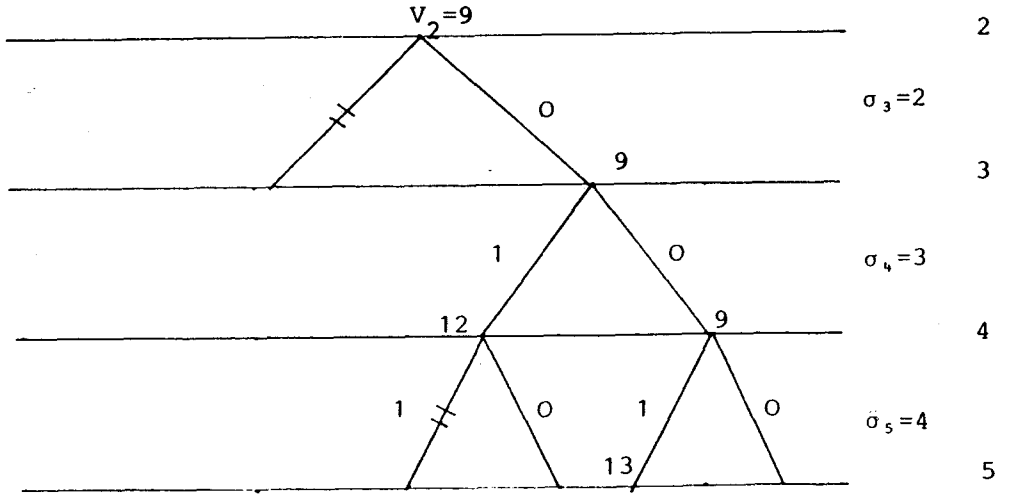


fig4.5

Pour déterminer $w_2=\Sigma u_k\sigma_k$, on peut utiliser le même processus. Par exemple, on prend $v_0'=9$, on obtient:

V'	$V_o'=V_2$	$V_1'=V_2+\sigma_4$	$V_2'=V_1'+\sigma_5$
V_k'	9	12	
$\lvert V_k'-\frac{L}{2}\rvert$	1	2	

On a donc $\Delta_2=2$ et $w_2=u_4\sigma_4+u_5\sigma_5=1\cdot 3+0=3$. La décision suivante à considérer pour cette branche est $u=(1,1,0,0,u_5)$ puisque $\Delta_2>\Delta_1$. Pour $u_5=1$, on a:

$$\Delta_3=|v_2+w_2(u_4=0,u_5=1)-10|=3.$$

3) En gardant la contrainte $\Delta=1$, on recommence à partir du sommet 1 avec $v_1=6$. La décision devient $u=(1,0,u_3,u_4,u_5)$

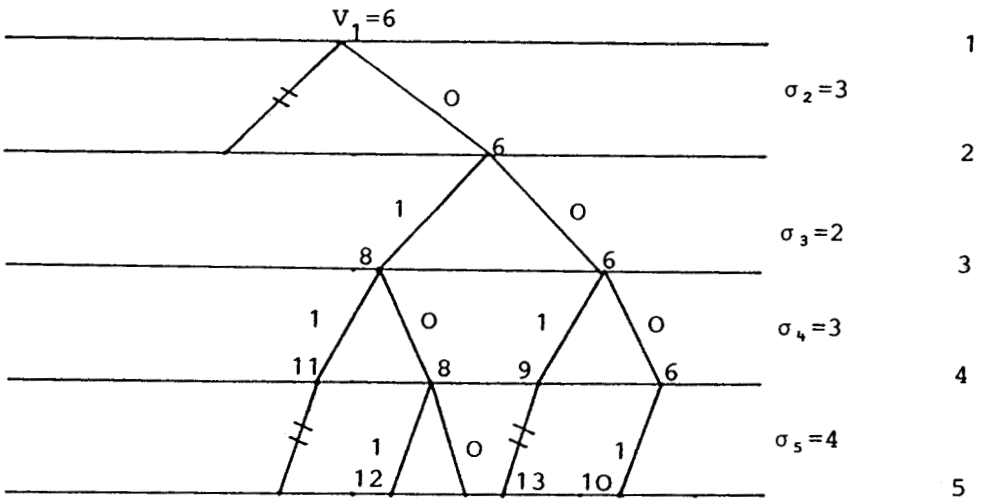


fig.4.6

A partir de $v_1=v_o'=6$, on peut obtenir $w_1=\Sigma u_k\sigma_k$ avec

$$\Delta_4=|v_1+w_1(u_3=1,u_4=1,u_5=0)-10|=1=\Delta_1$$
$$\Delta_5=|v_1+w_1(u_3=1,u_4=0,u_5=1)-10|=2>\Delta_1$$
$$\Delta_6=|v_1+w_1(u_3=0,u_4=1,u_5=0)-10|=3>\Delta_1$$
$$\Delta_7=|v_1+w_1(u_3=0,u_4=0,u_5=1)-10|=0<\Delta_1$$

Une solution optimale est u=(1,0,0,0,1)

et $J^*=|3+3+4-10|=0$

La solution optimale n'est peut être pas unique. Dans l'exemple, on a deux autres décisions optimales qui sont:

u=(0,1,0,0,1) avec $J^*=0$

u=(0,0,0,1,1) avec $J^*=0$

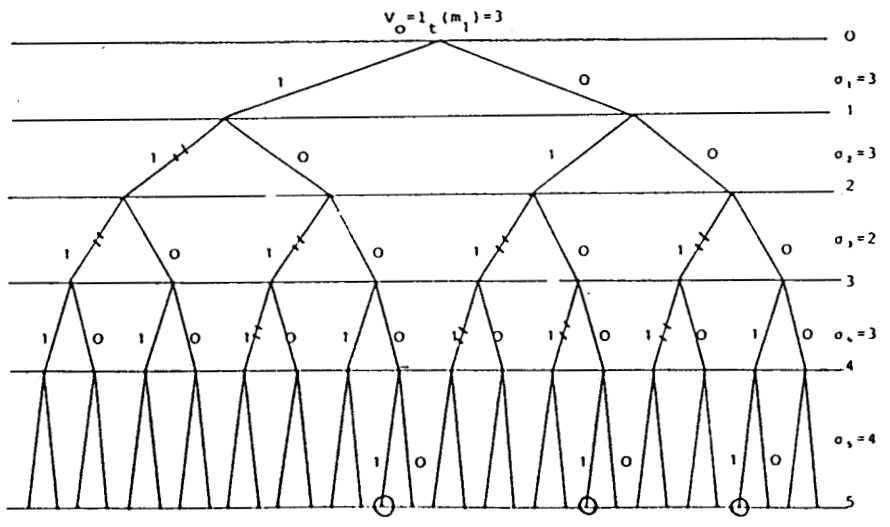


fig.4.7

b) méthode de programmation dynamique

L'affectation des n tâches peut se faire en utilisant l'approche de la programmation dynamique. Pour ce faire, on supposera que l'on affecte les tâches une par une. Compte tenu de la forme particulière du critère, nous allons définir l'algorithme à utiliser.

Comme précédemment, on note u_i la variable définissant l'affectation de la tâche i au robot 1, avec $u_i \in \{0,1\}$.

Soit:

$$J_1(l(m_1),u_1,u_2,\cdots u_n)=\left|l(m_1)+\sum_{i=1}^{n}u_i\sigma_i-\frac{L}{2}\right| \text{ le critère}$$

à minimiser, dans lequel $l(m_1)$ désigne la quantité de travail déjà affectée au robot 1, et l'indice inférieur signifie que l'on a à affecter les tâches de 1 à n. Notre but est de calculer l'affectation optimale, telle que:

$$J_1^*(x)=J_1(l(m_1),u_1^*(x),u_2^*(x),\cdots u_n^*(x))$$

$$= \min_{u_i\in\{0,1\}} \left|l(m_1)+\sum_{i=1}^{n} u_i\sigma_i-\frac{L}{2}\right| \qquad (4.6)$$

Le principe d'inclusion nous conduit à nous intéresser au critère plus général suivant:

$$J_k(x,u_k,u_{k+1},\cdots u_n)= \left|x+\sum_{i=k}^{n} u_i\sigma_i-\frac{L}{2}\right| \qquad (4.7)$$

où x désigne une quantité de travail quelconque affectée au robot 1, l'indice k signifie que seules les n-k+1 dernières tâches restent à affecter.

La solution de ce problème général dépend de la valeur de x et s'écrit:

$$J_k^*(x)=J_k(x,u_k^*(x),\cdots u_n^*(x))$$

$$= \min_{u_i\in\{0,1\}} \left|x+\sum_{i=k}^{n} u_i\sigma_i-\frac{L}{2}\right| \qquad (4.8)$$

Ecrivons le critère général sous la forme modifiée suivante:

$$J_k(x,u_k,\cdots u_n)= \left|x+u_k\sigma_k+\sum_{i=k+1}^{n} u_i\sigma_i - \frac{L}{2}\right|$$

$$= \left|y+\sum_{i=k+1}^{n} u_i\sigma_i - \frac{L}{2}\right| \qquad (4.9)$$

avec $y=x+u_k\sigma_k$

On obtient immédiatement:

$$J_k(x,u_k,\cdots u_n)=J_{k+1}(y,u_{k+1},\cdots u_n) \qquad (4.10)$$

$$\text{avec} \quad y=x+u_k\sigma_k$$

$$\forall\,(u_k,\cdots u_n)\in\{0,1\}^{n-k+1}$$

A) A partir de la relation (4.10), on obtient:

$$J_k^*(x)\leq J_k(x,u_k,\cdots u_n)=J_{k+1}(y,u_{k+1},\cdots u_n) \qquad (4.11)$$

$$\forall\,(u_{k+1},\cdots u_n)$$

Cette minoration reste vraie en particulier pour $u_i=u_i^*(y)$ $i=k+1,\cdots n$ d'où:

$$J_k^*(x)\leq J_{k+1}(y,u_{k+1}^*(y),\cdots,u_n^*(y)) \qquad \forall\, y \qquad (4.12)$$

C'est à dire:

$$J_k^*(x)\leq J_{k+1}^*(x+u_k\sigma_k) \qquad \forall\, u_k$$

On en déduit:

$$J_k^*(x)\leq \min_{u_k\in\{0,1\}} J_{k+1}^*(x+u_k\sigma_k) \qquad (4.13)$$

B) Toujours à partir de (4.10) on peut écrire:

$$J_{k+1}^*(y)\leq J_{k+1}(y,u_{k+1},\cdots u_n)=J_k(x,u_k,\cdots u_n) \qquad (4.14)$$

$$\forall\, u_i \quad i=k,\cdots n$$

En particulier prenons $u_i=u^*(x)$ pour $i=k+1,\cdots n$, on a:

$$J_{k+1}^*(x+u_k\sigma_k)\leq J_k(x,u_k,u_{k+1}^*(x),\cdots u_n^*(x)) \quad \forall\, u_k \qquad (4.15)$$

Cette inégalité lie deux fonctions de u_k pour toutes les valeurs de leur argument. On en déduit:

$$\min_{u_k\in\{0,1\}} J_{k+1}^*(x+u_k\sigma_k)\leq \min_{u_k\in\{0,1\}} J(x,u_k,u_{k+1}^*(x),\cdots u_n^*(x)) \qquad (4.16)$$

Le minimum du membre de droite est obtenu pour $u_k=u^*(x)$. On obtient donc finalement:

$$\min_{u_k\in\{0,1\}} J^*_{k+1}(x+u_k\sigma_k)\leq J^*_k(x) \qquad (4.17)$$

Les relations (4.13) et (4.17) fournissent alors:

$$\forall\ x \qquad J^*_k(x)= \min_{u_k\in\{0,1\}} J^*_{k+1}(x+u_k\sigma_k) \qquad (4.18)$$

Cette relation constitue l'algorithme de programmation dynamique recherché. En effet, il relie pour toute charge de travail initialement présente x, la répartition optimale des n-k+1 dernières tâches à celle des n-k dernières tâches (calculée à l'itération précédente).

L'initialisation de cet algorithme se fait par:

$$J^*_{n+1}(x)= \left| x- \frac{L}{2} \right| \qquad (4.19)$$

l'indice n+1 signifie (par convention) qu'il ne reste aucune tâche à répartir.

Il est évident par ailleurs que les bornes de variation de x, quantité de travail déjà affectée au robot 1 sont données par l'intervalle:

$$x\in[l(m_1),\ l(m_1)+\sum_{i=1}^{n}\sigma_i] \qquad (4.20)$$

Prenons l'exemple que nous avons étudié par la méthode SEP, nous affectons maintenant les tâches communes par la méthode de programmation dynamique.

<u>exemple:</u>

$l_t(m_1)=3 \quad \sigma_1=3, \sigma_2=3, \sigma_3=2, \sigma_4=3, \sigma_5=4 \qquad \frac{L}{2}=10$

On a donc $x\in[3,18]$

$$J^*_6(x)=|x-10|$$

Pour bien comprendre, nous avons les figures correspondantes à chaque phase du résolution.

Quelque soit la valeur de x, on trouve grâce à la figure (4.8) la valeur de $J^*_6(x)$ sur la courbe(1).

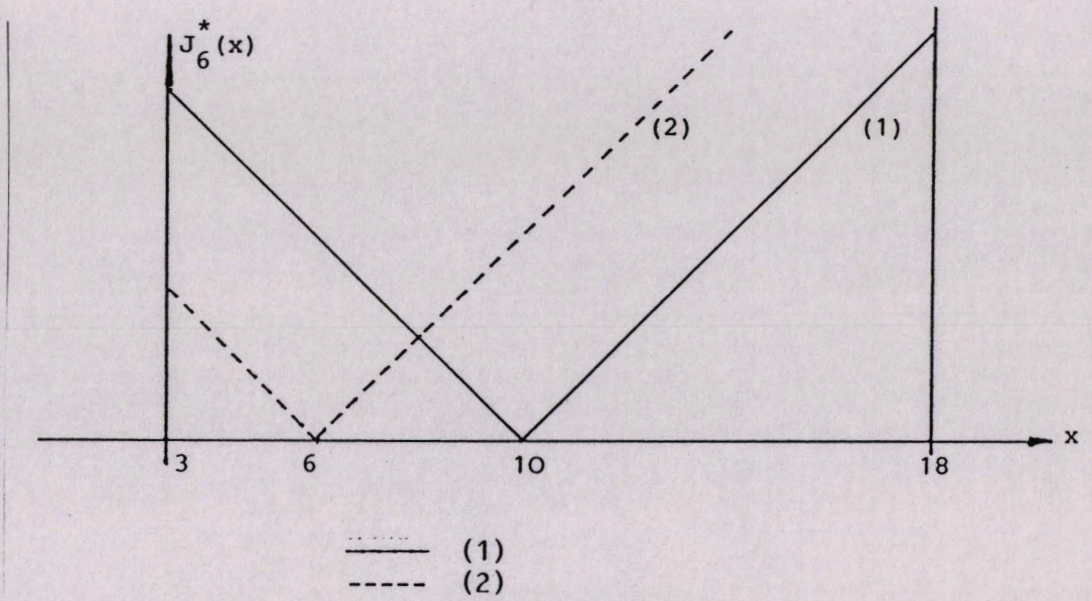


fig.4.8

$$J_5^*(x) = \min_{u_5} J_6^*(x+4u_5)$$

$$x \begin{cases} \xrightarrow{u_5=0} J_6^*(x+4u_5) = J_6^*(x) = |x-10| \\ \xrightarrow{u_5=1} J_6^*(x+4u_5) = J_6^*(x+4) = |x-6| \end{cases}$$

$u_5=0$ correspond la courbe(1) de la figure (4.8) et $u_5=0$ correspond la courbe(2) de la même figure, on a donc la valeur de $J_5^*(x)$ et la zone de décision dans la figure (4.9). De la même façon, nous avons les courbes et les zones de décisions suivantes.

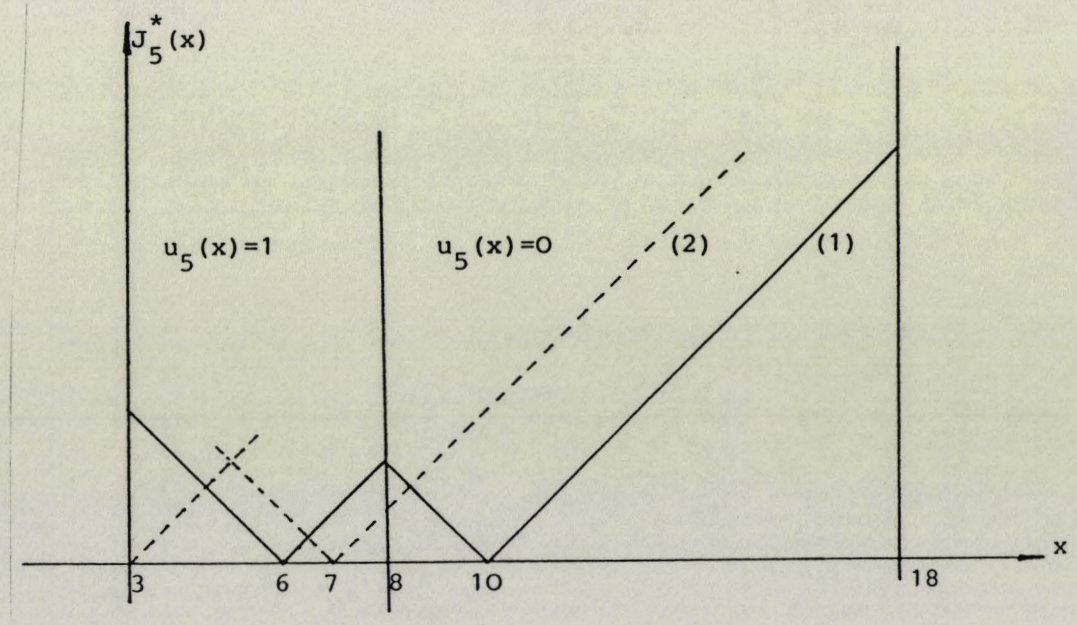


fig.4.9

$$J_4^*(x) = \min_{u_4} J_5^*(x+3u_4)$$

$$x \begin{cases} u_4=0 & J_4^*(x)=J_5^*(x) \\ u_4=1 & J_4^*(x)=J_5^*(x+3) \end{cases}$$

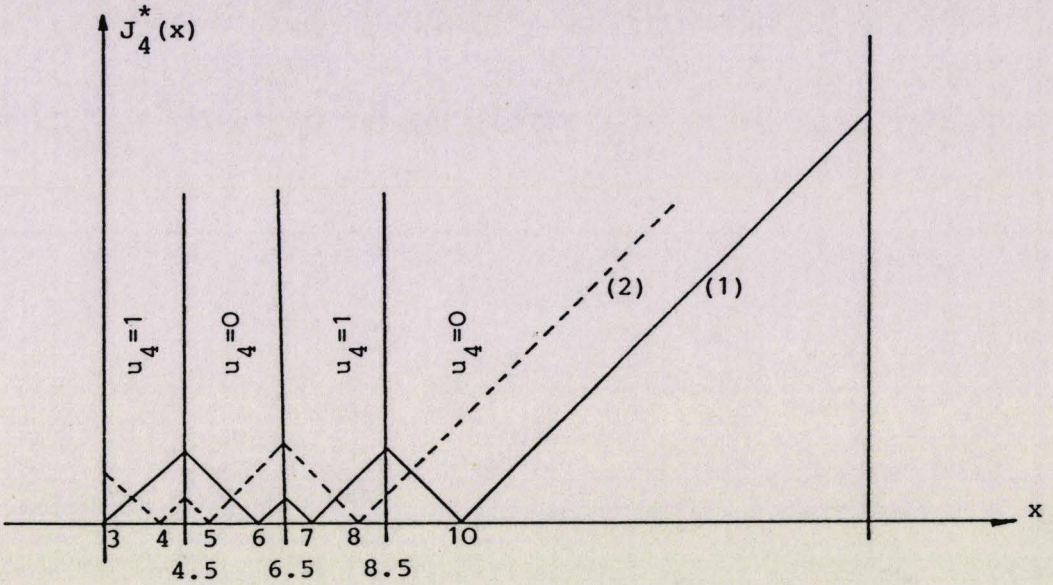


fig.4.10

$$J_3^*(x) = \min_{u_3} J_4^*(x+2u_3)$$

$$x \begin{cases} u_3=0 & J_3^*(x)=J_4^*(x) \\ u_3=1 & J_3^*(x)=J_4^*(x+2) \end{cases}$$

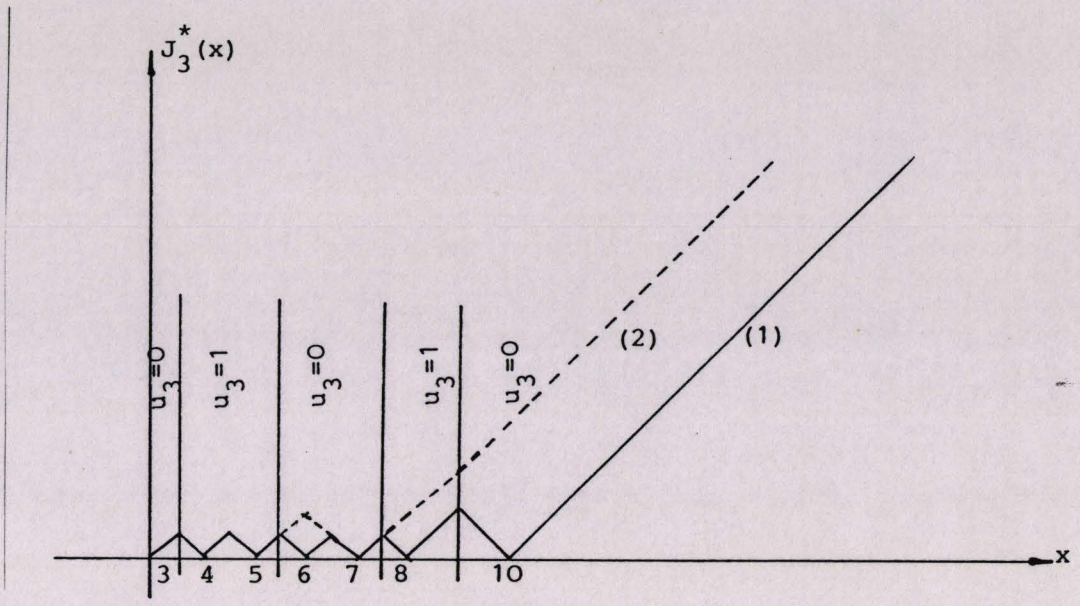


fig.4.11

$$J_2^*(x) = \min_{u_2} J_3^*(x+3u_2)$$

$$x \begin{cases} u_2=0 & J_2^*(x)=J_3^*(x) \\ u_2=1 & J_2^*(x)=J_3^*(x+3) \end{cases}$$

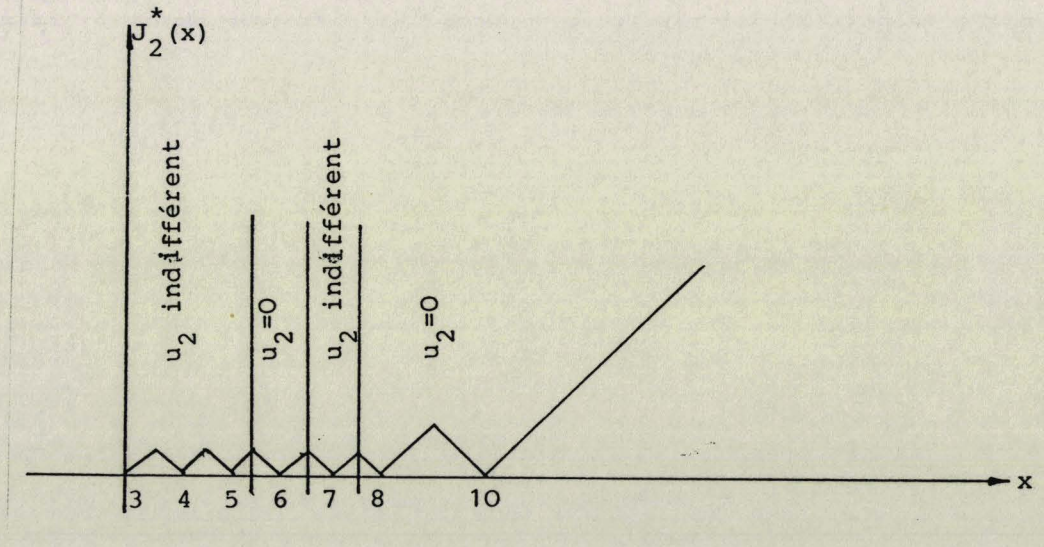


fig.4.12

$$J_1^*(x) = \min_{u_1} J_2^*(x+3u_1)$$

$$x \begin{cases} u_1=0 & J_1^*(x)=J_2^*(x) \\ u_1=1 & J_1^*(x)=J_2^*(x+3) \end{cases}$$

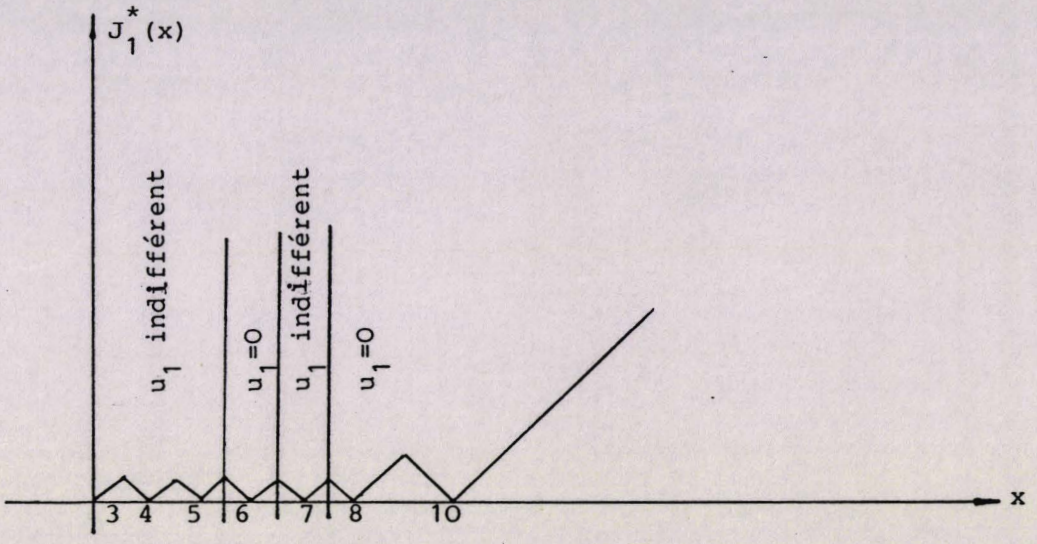


fig.4.13

En utilisant les résultats précédents, on peut trouver les solutions optimales. On commence par la valeur $x=1(m_1)=3$:

	$J_1^*(3)=0$	$u_1^*=0$	ou	1	équivalent
$u_1^*=0$	$J_2^*(3)=0$	$u_2^*=0$	ou	1	équivalent
$u_1^*=1$	$J_2^*(6)=0$	$u_2^*=0$			

On a donc

(0,0)	$J_3^*(3)=0$	$u_3^*=0$
(0,1)	$J_3^*(6)=0$	$u_3^*=0$
(1,0)	$J_3^*(6)=0$	$u_3^*=0$

Pour déterminer la décision u_4^*, on a les trois cas suivants:

$(0,0,0)$ $J_4^*(3)=0$ $u_4^*=1$

$(0,1,0)$ $J_4^*(6)=0$ $u_4^*=0$

$(1,0,0)$ $J_4^*(6)=0$ $u_4^*=0$

On a donc la décision u_5^* dans le cas suivant:

$(0,0,0,1)$ $J_5^*(6)=0$ $u_5^*=1$

$(0,1,0,0)$ $J_5^*(6)=0$ $u_5^*=1$

$(1,0,0,0)$ $J_5^*(6)=0$ $u_5^*=1$

Les trois solutions optimales trouvées sont:

$(0,0,0,1,1)$ $(0,1,0,0,1)$ $(1,0,0,0,1)$

Ces deux méthodes nous permettent d'affecter les n tâches de m_3 aux queues 1 et 2, à chaque instant t.

Nous allons maintenant étudier le cas où l'on fait un pas de prédiction, c'est-à-dire qu'on considèrera les tâches qui seront exécutables à t+1.

2. -Politique optimale sous un pas de prédiction

On suppose dans ce cas que l'on peut prédire les prochaines tâches exécutables avec une limitation d'un pas. A l'instant t, les tâches exécutables sont $m_1(t) \cup m_2(t) \cup m_3(t)$. Les instants de décision suivants peuvent être:

1) La fin d'une tâche dont au moins un successeur a sa ressource présente dans la cellule.

2) L'arrivée d'une pièce pour une tâche préexécutable par rapport aux antécédents.

Dans ces cas, on doit faire une nouvelle affectation puisque l'ensemble des tâches exécutables est modifié. En fait, nous allons voir que dans certains cas, la modification des ensembles peut être prévue.(figure 4.14)

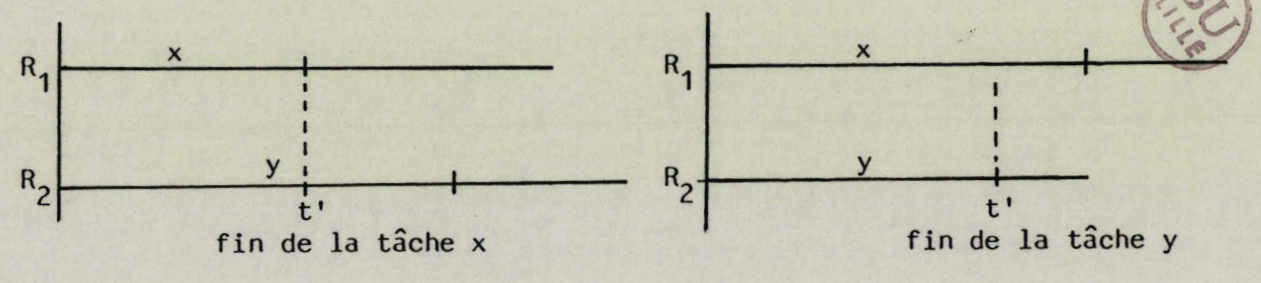


fig.4.14

A) On suppose que les pièces nécessaires aux successeurs des tâches x ou y, sont présentes dans le système. Dans ce cas, au lieu de n'affecter que les tâches exécutables à l'instant t, on calcule l'affectation sur les tâches qui seront exécutables à t'. Ces tâches sont des tâches dont la ressource est présente et dont les antécédents seront terminés à t'.

On note A^i (i=1,2,3) l'ensemble des tâches qui deviendront exécutables à t.

Pour affecter, on travail alors sur:

$l(M_1(t))=l(m_1(t))+l(A^1)$

$l(M_2(t))=l(m_2(t))+l(A^2)$

$l(M_3(t))=l(m_3(t))+l(A^3)$ à l'instant t.

Les algorithmes précédents peuvent être utilisés à l'instant t pour affecter les tâches de $M_1(t) \cup M_2(t) \cup M_3(t)$.

B) On suppose qu'il existe des tâches préexécutables par rapport aux antécédents à l'instant t. Dans ce cas, on ne peut pas prédire les tâches qui seront exécutables à t', car ceci dépend de l'arrivée des pièces. Mais, lorsqu'une ressource arrive, on relance l'algorithme d'affectation.

4.2.2 - Présence de conflits dans le système

Nous venons de voir des méthodes qui permettent d'affecter, de façon optimale, les tâches de m_3 aux queues de robots 1 ou 2. Aucun conflit n'apparait dans le système dans la mesure où une tâche consomme une ressource qui lui est propre. Dans le cas contraire, si l'on suppose qu'une même ressource peut être consommée par plusieurs tâches, un conflit peut apparaître .L'exemple suivant le montre.

exemple:

Les tâches préexécutables par rapport aux antécédents à l'instant t sont:

$\{x_1,x_2,x_3\} \subset \alpha$

$\{y_1,y_2,y_3\} \subset \beta$

$\{z_1,z_2,z_3\} \subset \delta$

Les durées de travail peuvent être déterminées lorsque ces tâches sont exécutables:

$L_t(m_1)=L(x_1)+L(x_2)+L(x_3)=2+3+2=7$ unités

$L_t(m_2)=L(y_1)+L(y_2)+L(y_3)=3+2+3=8$ unités

$L_t(m_3)=\sigma_1+\sigma_2+\sigma_3=3+2+4=9$ unités

hypothèses:

a) Trois types de pièce peuvent être consommées par les tâches:

x_2 consomme une pièce de type 1,

x_3 consomme une pièce de type 3,

$x_1, y_1, y_2, y_3, z_1, z_2, z_3$ consomment une pièce de type 2.

b) Trois pièces respectivement de type 1,2,3 sont présentes dans le système à l'instant t.

D'après les hypothèses précédentes, les 9 tâches sont exécutables à l'instant t avec:

$$L_t(m_1)=7 \qquad L_t(m_2)=8$$

$$L_t(m_3)=\sigma_1+\sigma_2+\sigma_3=3+2+4=9$$

Utilisant les méthodes P.S.E.P ou programmation dynamique, l'affectation optimale est:

queue 1: $L_t^1=L(x_1)+L(x_2)+L(x_3)+\sigma_1+\sigma_2=12$ unités

queue 2: $L_t^2=L(y_1)+L(y_2)+L(y_3)+\sigma_3=12$ unités

Nous supposons que le robot 1 est libre à l'instant t et que le robot 2 terminera son travail à t+dt (dt est très court). Si la meilleure tâche pour le robot 1 est la tâche x_1, la longueur des queues à t+dt est:

$$L_{t+dt}^1=l(x_2)+l(x_3)=5$$

$$L_{t+dt}^2=0$$

puisque toutes les tâches de la queue 2 consomment la pièce de type 2 qui n'est plus présente dans la cellule. Dans ce cas, le robot 2 est oisif jusqu'à ce que les pièces convenables arrivent dans la cellule.

Nous proposerons une solution à ce problème de conflit au paragraphe 6.

conclusion Dans ce paragraphe,nous avons analysé le problème d'allocation des tâches aux deux robots. Le problème d'affectation des tâches exécutables consiste à affecter les tâches communes. Le critère que nous choisissons est de minimiser la différence de deux queues à chaque instant t. Nous analysons d'abord, dans le cas stationnaire, la probabilité d'affectation des tâches en supposant que le modèle de file d'attente est le type de M/G/2. En utilisant les informations disponibles, nous avons, dans la deuxième partie, utilisé les méthodes de SEP et de programmation dynamique pour affecter les tâches communes à chaque instant t. Les tâches exécutables dans la queue 1 (ou queue 2) peuvent être exécutées avec le degré

prioritaire que nous avons déduit au chapitre III. Utilisant les méthodes considérées dans ce paragraphe, le problème de conflit n'apparait pas lorsqu'on suppose qu'une tâche consomme une ressource correspondante. Dans le cas où une même ressource peut être consommée par plusieurs tâches, il faut traiter le conflit sur les ressources.

5 - AFFECTATION DIRECTE DES TACHES COMMUNES

La procédure de décision étudiée jusqu'à présent peut être décomposée en deux étapes:

1 - Création de deux files d'attente par affectation des tâches communes soit à l'un soit à l'autre des robots.

2 - Choix, par chacun des robots,de la tâche optimale à exécuter dès qu'il sera libéré, parmi les tâches de sa file.

Dans ce paragraphe, nous envisageons une procédure de décision dans laquelle la phase 1 a été supprimée. Comme l'indique la figure 4.15, on crée dans ce cas les deux files d'attente en affectant à chacune d'elles la totalité des tâches de m_3.

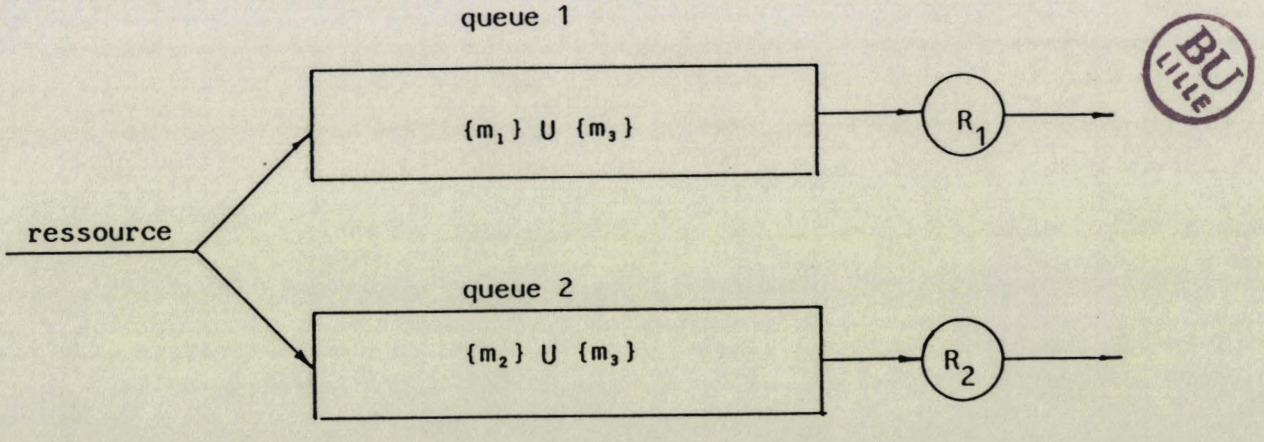


fig.4.15

Les tâches exécutables dans la queue i (i=1,2) à l'instant t sont:

$$M_t^i = \{m_i\} \cup \{m_3\} \qquad i=1,2$$

Le problème se réduit alors au choix de la meilleure tâche, avec la priorité que nous avons envisagée, au chapitre III, parmi les tâches M_t^i (i=1,2) lorsque le robot R_i(i=1,2) est libre.

Dans la mesure où les deux robots travaillent de façon coopérative, les phénomènes suivants peuvent être rencontrés:

1) des tâches communes peuvent être choisies par les deux robots à l'instant t(conflit sur les tâches);

2) les ressources présentes dans le système peuvent être consommées par les deux robots à l'instant t. (conflit sur les ressources).

On peut remarquer que le conflit sur les ressources, déjà présenté précédemment, ne se rencontre que dans le cas où plusieurs tâches peuvent consommer la même ressource, alors que le conflit sur les tâches est spécifique au mode de décision envisagé dans ce paragraphe. Dans ce mode de décision, les 3 points suivants doivent être considérés:

a) l'évolution de l'état des tâches exécutables;

b) l'évaluation du gain du système;

c) les conflits du système.

5.1 - Evolution de l'état des tâches exécutables

1. Soit $t_i(R_1)$ l'instant de fin d'exécution de la tâche i par le robot R_1;

   soit $t_j(R_2)$ l'instant de fin d'exécution de la tâche j par robot R_2.

L'ensemble des tâches pré-exécutables par rapport aux antécédents PEA(t) peut être évalué comme suit:

i) $t_i(R_1)=t_j(R_2)$

Les coefficients d'appartenance de l'ensemble des tâches PEA(t) peuvent être obtenus:

$x \notin S_i$(ou S_j) $\qquad \mu_{PEA(t_i)}(x)= \mu_{PEA(t_o)}(x)$

$x \in S_i$(ou S_j) $\qquad \mu_{PEA(t_i)}(x)= \mu_{PEA(t_o)}(x)+ \Delta_\mu(x)$

$\Delta_\mu(x)$ est défini dans le chapitre III.

x=i(ou j) $\qquad \mu_{PEA(t_i)}(x)=0$

ii) $t_i(R_1)=t_j(R_2)$ les deux tâches terminent en même temps, les coefficients d'appartenance de PEA(t) sont:

$x \notin \{S_i \cup S_j\}$ $\quad \mu_{PEA(t_i)}(x) = \mu_{PEA(t_o)}(x)$

$x \in S_i$ (ou S_j) $\quad \mu_{PEA(t_i)}(x) = \mu_{PEA(t_o)}(x) + \Delta_\mu^i(x)$ (ou $\Delta_\mu^j(x)$)

$x \in \{S_i \cap S_j\}$ $\quad \mu_{PEA(t_i)}(x) = \mu_{PEA(t_o)}(x) + \Delta_\mu^{ij}(x)$

$x=i$ ou j $\quad \mu_{PEA(t_i)}(x)=0$

Dans le cas de la neutralité, Δ_μ^{ij} peut être défini par:

$$\mu_{PEA(t_i)}(x) = \frac{\delta_1(t_i,x)}{(x)} = \frac{\delta_1(t_o,x)+2}{(x)}$$

$$= \mu_{PEA(t_o)}(x) + \frac{2}{\delta(x)}$$

$$= \mu_{PEA(t_o)}(x) + \Delta_\mu^{ij}(x)$$

Remarque: $\delta_1(t_i,x)$, $\delta_1(t_o,x)$ et $\delta(x)$ sont définis dans le chapitre III.

2. L'évaluation de l'ensemble des tâches pré-exécutables par rapport aux ressources PER(t) dans le cas où deux robots travaillent en coopération fournit les mêmes résultats que ceux obtenus au chapitre III, sauf que la disparition simultanée de deux pièces .

Soit (i,j) un couple de tâches affectées de manière prévisionnelle aux robots R_1 et R_2. Elles sont caractérisées par les données du tableau ci dessous:

tâches	exécution par	type de ressource consommée	date début	date fin
i	R_1	k	$t_o(i)$	t_i
j	R_2	l	$t_o(j)$	t_j

Dans le cas où $t_o(i)=t_o(j)$, nous pouvons évaluer PER(t) en utilisant l'approche du chapitre III à chaque instant t où une tâche termine son exécution.

Dans ce cas, on a:

- Pour toutes les tâches ne nécessitant pas des ressources de type k ou l, soit q ce type,on a:

$$\mu_{PER(t_m)}(x_q)=1 \quad \text{si} \quad \mu_{PER(t_o)}(x_q)=1$$

$$t_m=\min\{t_i,t_j\}$$

$$\mu_{PER(t_m)}(x_q)=\text{prob}(n_q(t_m)\geq 1)$$

$$\text{si} \quad \mu_{PER(t_o)}(x_q)=0$$

- Pour les tâches, utilisant une ressource de type l ou k ($l\neq k$) à l'instant t_m, on a:

- S'il existe au moins deux ressources de type l(ou k) à l'instant t_o:

$$\mu_{PER(t_m)}(x_{l\,(\text{ou } k)})= \mu_{PER(t_o)}(x_{l\,(\text{ou } k)})=1$$

- S'il n'existe qu'une ressource de type l ou k à l'instant t_o:

$$\mu_{PER(t_m)}(x_{l\,(\text{ou } k)})=\text{prob}(n_k(t_m)\geq 1)$$

- Pour les tâches, utilisant une ressource de type l lorsque l=k, on a:

-S'il existe au moins trois ressources de type l(l=k) à l'instant t_o, on a:

$$\mu_{PER(t_m)}(x_l)=1$$

- S'il n'existe que deux ressources de type l(l=k) à l'instant t_o:

$$\mu_{PER(t_m)}(x_l)=\text{prob}(n_l(t_m)\geq 1)$$

Par ailleurs,soient t_o l'instant de décision
t_f la date de fin d'exécution d'une tâche.

Nous pouvons évaluer le nombre moyen des ressources de type k à l'instant t_f:

$$\bar{n}_k(t_f)=n_k(t_o)+p_1+2p_2+\cdots qP_q-\theta$$

où $p_i (i=1,2,\cdots q)$ est la probabilité qu'il arrive une ressource de type k entre t_o et t_f.

$\theta=0$, on lance la tâche qui n'est pas le type k à l'instant t_o.

$\theta=1$, on lance une tâche de type k à l'instant t_o.

$\theta=2$, on lance deux tâches de type k à l'instant t_o.

Après avoir évalué les ensembles de PEA et PER, nous pouvons définir le coefficient d'appartenance d'une tâche à l'ensemble EXE(t) par

$$\mu_{EXE(t)}(x)=\mu_{PEA(t)}(x)*\mu_{PER(t)}(x)$$

5.2 - Evaluation du gain du système

Rappelons que le gain du système correspond la quantité de travail apportée par une tâche exécutée. D'après la figure 4.6, les tâches strictement exécutables peuvent être définies comme ci-dessous à l'instant t_o.

Soit $\underline{EXE}\ I(t_o)$ l'ensemble des tâches strictement exécutables à l'instant t_o à la queue I (I=1,2)

$$\underline{EXE}\ 1(t_o)=M^1_{t_o}=\{m_1(t_o)\}\cup\{m_3(t_o)\}$$

$$\underline{EXE}\ 2(t_o)=M^2_{t_o}=\{m_2(t_o)\}\cup\{m_3(t_o)\}$$

$y\in\underline{EXE}\ I(t_o)$, la quantité de travail apportée par la tâche y peut être calculée en fonction des situations de $PEA(t^y)$ et $PER(t^y)$.

$$EXE(t^y)=PEA(t^y)\cap PER(t^y)$$

Pour chacune des deux queues, nous pouvons calculer séparément le gain de tâche y EXE $I(t_o)$, I=1,2. La quantité de travail apportée par y dans le temps T_y peut être calculée comme le chapitre III:

$$Q[EXE\ I(t^y)]=Q[\underline{EXE}\ I(t^y)]+Q[\underline{\underline{EXE}}\ I\ (t^y)]$$

5.3 - Les conflits du système

Les conflits du système peuvent apparaître selon les hypothèses de ce paragraphe.(les tâches de m_3 sont affectées directement aux robots, et les tâches différentes consomment un même type de pièce). Les conflits apparaissent dans les cas suivants:

1) Les deux robots choisissent une même tâche à exécuter quand ils termineront leur travail en cours, évidemment cette tâche appartient à m_3.

2) Les deux robots choisissent des tâches différentes, mais celles ci consomment une même ressource; ils termineront leur travail en cours.

Ces deux types de conflit seront traités dans le paragraphe suivant.

6 - RESOLUTION DU PROBLEME DES CONFLITS

Pour résoudre les conflits, nous introduisons la notion de deuxième choix pour chacun des robots.

1) Lorsque les deux robots choisissent une même tâche z, z m_3, nous déterminons la deuxième meilleure tâche dans la queue 1 et dans la queue 2 en fonction du gain de système.

soient x la tâche de deuxième choix pour le robot R_1;

y la tâche de deuxième choix pour le robot R_2.

On a donc $x \in m_1 \cup m_3$, $y \in m_2 \cup m_3$

Pour affecter les tâches aux deux robots, nous envisageons les cas suivant:

	R_1	R_2
cas 1	z	y
cas 2	x	z

2) Lorsque les deux robots choisissent deux tâches différentes qui consomment une même ressource, nous notons:

x la tâche choisie par R_1 , $x \in m_1 \cup m_3$,

y la tâche choisie par R_2, qui consomme une même ressource que x , $y \in m_2 \cup m_3$.

x' la tâche de deuxième choix pour le robot R_1 sous la contrainte qu'elle ne consomme pas le même type de ressource que x , $x' \in m_1 \cup m_3$.

y' la tâche de deuxième choix pour le robot R_2, sous la contrainte qu'elle ne consomme pas le même type de ressource que x, $y' \in m_2 \cup m_3$.

Comme pour le conflit sur les tâches, nous envisageons alors les deux cas suivants:

	R_1	R_2
cas 1	x	y'
cas 2	x'	y

La résolution des conflits consiste alors à calculer l'intérêt du premier cas et à le comparer avec l'intérêt du deuxième cas. La meilleure des deux solutions sera retenue, elle est sans conflit par construction.

Remarque La définition de la tâche de deuxième choix est un problème sous contrainte qui peut donc fort bien ne pas avoir de solution. Il est donc possible d'obtenir des couples de tâches sans conflit de la forme (x, Φ) ou (Φ, x) où Φ désigne l'oisiveté du robot. Ceci indique l'état dans lequel se trouve le système à l'instant de décision, le conflit ne peut se résoudre qu'en laissant un des robots inactif.

7 - CONCLUSION

Pour deux robots qui travaillent au sein de la même cellule d'assemblage, la modélisation par un système à deux serveurs nous conduit à gérer les deux files d'attente de tâches exécutables. cette gestion se traduit par une allocation à deux niveaux:

1. Répartition des tâches entre les deux files d'attente;

2. Discipline prioritaire au sein de chacune des files.

Dans ce chapitre, nous avons envisagé trois approches différentes pour la répartition des tâches.

- Une approche stochastique consistant à affecter les tâches communes de façon aléatoire à l'une ou l'autre queue. Le problème est alors de définir les probabilités d'affectation. Une telle approche peut s'envisager pour un modèle M/G/2 sur un horizon infini.

- Les deux autres approches proposées tiennent compte du critère de minimisation du temps de réalisation du montage. La première effectue une optimisation à deux niveaux alors que la seconde opère sur un seul niveau d'optimisation, mais sur l'ensemble des tâches communes, pour chacun des robots.

Dans les deux cas, des conflits sur les ressources peuvent exister, alors que dans le deuxième cas on peut observer des conflits sur les tâches. Pour ces deux types de conflits, la notion du deuxième choix de chacun des robots permet une résolution simple et rapide.

CONCLUSION GENERALE

Nous nous sommes attachés dans ce travail à l'étude du problème de modélisation et de gestion d'une cellule de production flexible utilisant plusieurs robots. Il s'agit de réaliser un produit selon une gamme de fabrication donnée, chaque opération de la gamme nécessite une ressource propre(pièce) et des ressources partageables(robots, espace de travail).

Pour étudier un tel système automatisé de production, il faut faire appel à des modèles. Plusieurs types de modèles peuvent être envisagés pour un même système selon sa structure,son état d'avancement, les degrés de finesse souhaités pour la représentation, etc... Nous avons présenté d'abord quelques modèles qui peuvent être utilisés dans un système automatisé de production, enfin, nous avons analysé la cellule de production flexible par trois méthodes typiques: Algèbre des Dioides, Réseaux de Petri ,système de files d'attente.

Le problème posé se caractérise particulièrement par la prise en compte de la dynamique de la cellule et notamment par l'apparition aléatoire des opérations exécutables. C'est un problème d'ordonnancement affectation dynamique, dont l'objectif est la minimisation du temps de réalisation de l'ensemble des opérations à exécuter. Or, ceci conduit à la minimisation du temps d'oisiveté des robots et donc à la maximisation du nombre d'opérations exécutables à tout instant.

Le choix d'une opération qui après son exécution débloquera le plus grand nombre d'opérations exécutables, n'est pas évident. Principalement,il dépend des ressources, qui arrivent de façon aléatoire, et de l'état de déblocage des opérations. Pour piloter un tel système complexe, nous proposons d'utiliser la théorie des files d'attente.

La théorie des files d'attente utilisée pour résoudre un problème d'ordonnancement dynamique consiste, d'une part à contrôler le taux d'entrée des clients pour satisfaire un critère donné, d'autre part, à déterminer la priorité des clients dans la file.

Pour une file avec un robot, le calcul de priorité est basé sur la notion de quantité de travail apportée dans la queue. Le problème de décision consiste à sélectionner parmi les tâches exécutables, à chaque début de tâche et à chaque arrivée de ressource, la tâche la plus prioritaire.

Pour deux robots qui travaillent au sein de la même cellule d'assemblage, la modélisation par un système à deux serveurs nous conduit à une allocation à deux niveaux: Répartition des tâches entre les deux files d'attente et Discipline prioritaire au sein de chacune des files. Trois approches sont considérées pour la répartition des tâches: Une approche stochastique consiste à affecter les tâches communes

de façon aléatoire à l'une ou l'autre queue. Le problème est alors de définir les probabilités d'affectation en supposant que la cellule suit un modèle M/G/2. Les deux autres approches tiennent compte du critère de minimisation du temps de réalisation du montage. La première effectue une optimisation à deux niveaux alors que la seconde opère sur un seul niveau d'optimisation, mais sur l'ensemble des tâches communes pour chacun des robots. Dans les deux cas, des conflits sur les ressources peuvent exister, alors que dans le deuxième cas on peut observer des conflits sur les tâches. Pour ces deux types de conflits, la notion du deuxième choix de chacun des robots permet une résolution simple et rapide.

Il est évident que dans le cadre de ce travail, plusieurs problèmes méritent d'être approfondis, notamment:

- L'étude de la distribution d'arrivée des ressources.

- L'étude de l'influence du choix du coefficient d'actualisation pour évaluer le gain attendu.

- Le cas où les temps de service des robots sont différents.

- La prise en compte d' opérations nécessitant plusieurs robots simultanément pour sa réalisation.

L'ensemble des recherches menées dans le cadre de la coopération entre robots, conduit à doter la cellule étudiée des capacités de décision et de mise à jour de son état, qui lui permettent une certaine autonomie, et donc un comportement "intelligent" en réponse aux sollicitations de l'environnement. Cett préoccupation est de plus en plus sensible, avec le développement des recherches orientées vers l'utilisation de systèmes experts et de bases de connaissance pour le pilotage de telles cellules.

BIBLIOGRAPHIE

AMM.85 "Scheduling model for aiding real time FMS control"
JANE C.AMMONS
SMC conference IEEE 1985 TUCSON(USA)

ARA.84 "Pôle ateliers flexibles"
Rapport A.R.A,1984

BAL.80 "Public and private optimization at a service facility with approximation on congestion"
K.R.BALACHANDRAN and M.E.SHAEFER
Georgia Institute of Technology

BEL.57 "Dynamic programming"
R.E.BELLMAN
Princeton, N.J. Princeton Univ.press,1957

BEL.71 "Characterisation and computation of optimal policies for operating au M/G/1 queueing system with movable server"
C.E.BELL
Opns.Res. 19 p208-218(1971)

BEL.73 "Efficient operation of optimal-priority queueing system"
C.E.BELL
Opns.Res. 21 p777-786 (1973)

BEL.83 (G) "Evaluation, simulation et organisation d'ateliers"
G.BEL
CERT/DERA-Toulouse, Deuxièmes journées annuelles du programme automatisation et robotique avancées
BESANCON,16-17 Novembre, 1983

BEL.83 "Individual versus social optimization in the allocation of customers to alternative servers"
C.E.BELL and S.STIDHAM
Management Science vol.29 n°7 July 1983

BEL.85 "Modélisation et simulation de systèmes automatisés de production"
G.BEL et D.DUBOIS
RAIRO,Vol 19,N° 1,1985

BER.83 "Graphes"
C.BERGE
3ème Edition-Gauthier villars-1983

BER.83 (D.P) "Projected newton methods and optimalization of multicommodity flows"
D.P.BERTSEKAS and E.M.GAFNI
IEEE Trans. automat.contr.vol AC-28 p1090 (1983)

BON.85 "Les ateliers flexibles de production"
R.BONETTO
HERMES, 1985

BOU.86 "Modélisation d'un processus d'assemblage"
A.BOURJAULT et F.LHOTE
RAIRO Vol 20, N°2, 1986

BRA.83 "Réseaux de Petri: théorie et pratique"
G.W.BRAMS
Masson, Paris, 1983

BUB.85 "The control in assembly system with two robots"
Z.BUBNICKI,G.REYMAN,M.STAROSWIECKI,M.DJEGHABA
4th International conference on systems engineering,COVENTRY(ENGLAND),Sept.1985

BUD.77 "Principles of operations research for management"
BUDUICK,MOLENA,VOLLMANN
1977

BUZ.80 "Measuring and calculating queue length distribution"
J.P.BUZEN,P.J.DENNING
IEEE computer ,Avril,1980

CAR.82 "Un domaine très ouvert: les problèmes d'ordonnancement"
J.CARLIER,P.CHRETIENNE
RAIRO Rech. Opérationnelle Vol 16, 1982

CAR.83 "Modelling scheduling problems with timed PETRI nets"
J.CARLIER,P.CHRETIENNE,C.GIRAULT
4th European workshop on applications and theory of PETRI nets, Toulouse,1983,P84-103

CAV.81 "Outil CAO pour l'analyse, la spécification et la réalisation de l'automatisation d'un équipement de production"
M.CAVANNA, F.DOLLE, M.MOALLA
3e Journées scientifiques et techniques de la production automatisée, Toulouse,1981

CHR.83 "Les réseaux de PETRI temporisés"
Thése d'état, Université Pierre et Marie Curie
P.CHRETIENNE
Paris,1983

COB.54 "Priority assignment in waiting line problems"
A.COBHAM
Opns.Res. 2 P70-76 (1954)

COH.83 "Analyse du comportement périodique de systèmes de production par la théorie des Dioides"
G.COHEN,D.DUBOIS,J.P.QUADRAT,M.VIOT
Rapport de Recherche INRIA N°191,1983

COH.85 "A linear-system theoretic view of discrete-event processes"
G.COHEN, D.DUBOIS, J.P.QUADRAT,M.VIOT
22th IEEE conference on decision and control
IEEE trans.on automatic control, février,1985

COU.79 "Etude de l'existence de solutions pour certains problèmes d'ordonnancement"
C.COUZINET-MERCE
Th.D.I.,Université P.Sabatier, Toulouse,1979

COX.61 "Queues,methuen monograph"
D.R.COX and W.L.SMITH
Wiley, New York, 1961

CRA.71 "Optimal customer selection in exponential queues"
M.CRAMER
Orc Technical Report ORC 71-24, Opns. research center University of Califonia Berkeloy (1971)

CUN.60 "Process synchronisation in a steelworks a problem of feasibility"
R.A.CUNINGHAME-GREEN
Proc.2nd Int.Conf on operational rechearch,1960

CUN.62 "Describing industrial processes and approximating their steadystate behavier"
R.A.CUNINGHAME-GREEN
Opns.Res.Quart.1962,P95-100

CUN.79 "Minimax algebra"
R.A.CUNINGHAME-GREEN
Lect. notes in economics and mathematical system
Springer verlag,Berlin,1979

DAL.82 "Simulation of a flexible manufactucting system: case study of the citroen factory", MEUDON
Y.DALLERY, B.DESCOTTES-GENON,R.BONETTO
A.P.M.S-1982-IFIP CONF.BORDEAUX

DAL.83 "Recherche d'une même base de description en vue de la simulation et de la commande d'un atelier flexible:utilisation du GRAFCET"
Y.DALLERY, H.DENEUX, R.DAVID
Actes du congrès automatique ,AFCET,Besançon, France,Nov.1983

DAL.84 "Une méthode analytique pour l'évaluation des performances d'un atelier flexible"
Y.DALLERY
Thèse Docteur Ingénieur I.N.P.Grenoble,1984

DEN.77 "Operational analysis of queueing networks"
P.J.DENNING,J.P.BUZEN
Pro. of Int.Symp.on modelling and performance evaluation of computer systems, Bonn,Octo.1977

DIB.70 "Ordonnancement et potentiels méthode MPM"
M.L.DIBON
HERMANN,Paris, 1970

DJE.86 "Problemes de décision dans une cellule de production utilisant la coopération entre robots"
M.DJEGHABA
Thèse de l'Université des Sciences et Techniques de Lille Flandres Artois,1986

DUB.83 "Using PETRI nets to represent production process"
D.DUBOIS,K.STECKE
22th IEEE conference on decision and control 1983

DUM.74 "Méthodes d'ordonnancement paramétriques d'un atelier"
P.DUMAS
Th.DOC. Paris VI,1974

DUP.82 "A survey of flexible manufacturing systems"
C.DUPONT-GATELMAND
Journal of manufacturing system,1982

DUP.83 "Le concept d'atelier flexible: une approche industrielle et une approche par les modèles"
C.DUPONT
Journées A.R.A-pôle atelier flexible-1983

EPH.80 "A simple dynamic routing problem"
A.EPHREMIDES ,P.VARAIYA and J.WALRAND
IEEE Trans. on automatic control Vol 25, 1980

ERS.82 "Applying new dominance concepts to job schedule optimization"
J.ERSCHLER,G.FONTAN,C.MERCE,F.ROUBELLAT
E.J.O.R.,North,Holland N°11,1982

ERS.85 "Un nouveau concept de dominance pour l'ordonnancement de travaux sur une machine"
J.ERSCHLER,G.FONTAN,C.MERCE
RAIRO, Vol 19,N°1,1985,P15-26

FIS.78 "Principles of discrete-event simulation"
G.S.FISHMAN
Wiley,New York,1978

FON.80 "Notion de dominance et son application à l'étude de certains problèmes d'ordonnancement"
G.FONTAN
Thèse de Docteur es-sciences,
Université P.Sabatier,1980

GHO.85 "Control of arrivals to two queues in series"
H.A.GHONEIM and S.STIDHAM
European journal of operational research 21,1985

GNE.68 "Introduction to queueing theory"
B.V.GNEDENKO and I.N.KOVALENKO
Israel program for scientific translations
Jerusalem,1968

GON.79 "Graphes et algorithmes"
M.GONDRAN, M.MINOUX
EYROLLES,PARIS, 1979

GOR.67 "Closed queueing systems with exponential serveurs"
W.G.GORDON, G.F.NEWELL
Operations research 15,1967,P252-267

GRO.74 "Fundamental of queueing theory"
D.GROSS and C.M.HARRIS
John wiley & sons New York, 1974

GRO.81 "Automation et systèmes de production"
"FAO, fabrication assistée par ordinateur"
M.P.GROOVER
Volume 1, Volume 2 Hermes publiching,1981

HAJ.84 "Optimal control of two interacting service stations"
B.HAJEK
IEEE Trans. automatic control,vol AC-29,N°6, 1984

HAL.76 "Highway traffic equilibria analysed via geometric programming"
M.A.HALL and E.L.PETERSON
In traffic equilibrium methods M.A.Florian Ed
New York: Springer-Verlay (1976)

HAR.75 "Dynamic scheduling of a multiclass queue: discout optimality"
M.J.HARRISON
Opns.Res. 23 p270-282 (1975)

ING.83 "Les ateliers flexibles"
Ingersoll Engieers France
Edition de l'USINE

JAC.63 "Job-shop-like queueing systems"
J.R.JACKSON
Manag.sci.10,1963 P131-142

KAU.59 "Méthodes et modèles de la recherche opération-nelle"
A.KAUFMANN
1959

KAU.73 "Introduction à la théorie des sous-ensembles flous"
Edition Masson,1973

KEN.51 "Somes problems in the theory of queues"
D.G.KENDALL
Journal of the Royal statistical society,ser. B,13, P151-185,1951

KLE.76 "Queueing systems"
L.KLEINROCK
Vols I and II, NEW YORK: Wiley,1976

KLE.79 "Power and deterministic rules of thumb for probabilitic problems in computer communications"
L.KLEIROCK
In Proc. INT. Conf.Commun (June,1979)

KOE.59 "Production lines and internal storage:a review"
E.KOENIGSBERG
Management science 5,1959,P410-433

KUS.85 "Planning of flexible manufacturing systems"
ANDREW KUSIAK
Robotica, vol 3, 1985, pp.229-232

LAG.78 "De la logique d'agrégation des critères à une logique d'agrégation désagrégation des préférences et de jugements"
E.J.LAGREZE
Cahiers de LAMSADE sep.1978

LAG.80 "Aide à décision multicritères et systèmes relationnels de préférence"
E.J.LAGREZE,B.ROY
Cahiers de LAMSADE 1980

LAZ.83 "Optimal flow control of a class of queueing networks in equilibrium"
A.A.LAZAR
IEEE Trans. on automatic control vol.AC-28 n°11 1983

LEV.82 "Lancement periodique de produits dans un atelier flexible"
D.LEVEQUE
Thèse D.I-INSA Toulouse, 1982

LIN.84 "Optimal control of a queueing system with two heterogeneous servers"
W.LIN and P.R.KUMAR
IEEE Trans. on automatic control vol AC-29 N°8 84

LIP.71 "The streetwalker's dilemma: a job shop model"
S.A.LIPPMAN and S.ROSS
SIAM J. Appl. Math,20 p336-344 (1971)

LIP.77 "Individual versus social optimization in exponential congestion systems"
S.A.LIPPMAN and S.STIDHAM.JR
Opns.Res. 25, p233-247 (1977)

LIT.61 "A proof of the queueing formula L=λW"
J.D.C.LITTLE
Operations research 9,P383-387,1961

LOW.71 "Optimal dynamic operating policies for a M/N/S queue with variable arrival rate"
D.W.LOW
IBM Los angeles scientific center report 1971

MAT.74 "Simulation program generators"
S.C.MATHEWSON
Simulation,December,1974, P181-189

MAR.78 "Techniques et applications de la recherche opérationelle"
Alain MARTEL
Gaeton MARIN,1978

MAZ.81 "Réalisation d'un support expérimental de recherche pour le projet robotique PANDORE"
Définition et implantation du langage LM
E.MAZER
Th.3e cycle, I.N.P.G., Janvier,1981

MIR.84 "Le langage LM-manuel de référence"
J.F.MIRIBEL,E.MAZER
Editions Cepadues, 1984

NAY.66 "Computer simulation and SLAM"
T.H.NAYLOR,J.L.BOLINTFY,D.S.BURDICK and K.CHU
Wiley, 1966

PRI.79 "Introduction to simulation and SLAM"
A.B.PRITSKER,C.D.PEGDEN
Halsted press, New York,1979

REY.84 "General approach to assembly robot control"
G.REYMAN
Elsevier Science Publishery,1984

REY.84 (bis) "Application of patterm recognition algorithms for robotics assembly"
G.REYMAN
3th International conference on system engeneering, DAYTON,OHIO(USA),1984

ROS.70 "Applied probability model with optimization application"
S.Ross
San Francisco: Holden-Day (1970)

ROY.64 "Les problèmes d'ordonnancement: applications et méthodes physionomie et traitement du problème d'ordonnancement"
B.ROY
AFIRO.DUNOD,1964

ROY.69 "Procédure d'exploitation par séparation et évaluation (PSEP et PSES)"
B.ROY
RAIRO N°6, p61-90, 1969

ROY.70 "Algèbre moderne et théorie de graphes"
B.ROY
Tome 2 (Dunod, paris,1970)

ROY.80 "Systèmes relationnels de préférence en présence de critères multiples avec seuils"
B.ROY
Cahier du C.E.R.O vol 22 (1980)

SIM.84 "La reconnaissance des formes par algorithmes"
J.C.SIMMON
Edition Masson,1984

SOU.81 "Un modèle de résolution des problèmes d'ordonnancement dynamiques"
J.P.SOUBRIER
Th.D.I.Université P.Sabatier,Toulouse,1981

SOU.82 "Un algorithme de résolution des problèmes d'ordonnancement dynamiques"
J.P.SOUBRIER
RAIRO,R.O.,Vol.16,n°3,Août 1982

STA.85 "Decisions problems in the flexible assembly cell using robot cooperation"
M.STAROSWIECKI,M.DJEGHABA,M.BAYART,G.REYMAN
4th International conference on systems engineering COVENTRY(ENGLAND) Sept,1985

STA.85 (bis) "Task scheduling by multicriteria optimisation in flexible assembly cell using robot cooperation"
M.STAROSWIECKI,M.DJEGHABA,M.BAYART
15th International symposium on industrial robots
Tokyo,Japon,sept,1985

STA.87 "Dynamic scheduling of a queueing system with application to the control of a robotized assembly"
M.STAROSWIECKI,Y.YANG,M.BAYART
Seventh international congress of cybernetics and systems. London,7-11,sept,1987

STI.80 "Control of arrivals to a stochastic input-output system"
S.G.JOHANSEN and S.STIDHAM
Adv.Appl.Prob. Vol 12 P972-999 (1980)

STI.85 "Optimal control of admission to a queueing system"
S.STIDHAM
IEEE Trans. on automatic control AC-30 N°8 1985

TAK.62 "Introduction to the theory of queues"
LAJOS TAKACS
Univ.texts in the mathematical sciences New york
OXFORD University press,1962

THO.80 "Aide à la décision pour l'ordonnancement d'atelier en temps réel"
V.THOMAS
Thèse de 3^{e} cycle Univ.P.sabatier, Toulouse,1980

VAL.85 "SEDRIC:Un simulateur à événements discrets basé sur les réseaux de PETRI"
R.VALETTE,V.THOMAS,S.BACHMANN
RAIRO Vol 19 N^{O}5,1985 P423-453

WEB.78 "On the optimal assignement of customers to parallel servers"
R.WEBER
J. of Applied probability 15 p406-413,1978

WIN.77 "Optimality of the shorteste-processing-time discipline"
W.WINSTON
J.of applied probability 14,p181-189,1977

YAN.87 "Génération en temps réel du plan d'action d'un robot a partir d'un graphe potentiel-tâches"
Y.YANG,M.STAROSWIECKI,M.BAYART
Tenth IASTED international symposium robotics and automation. LUGANO,SWITZELAND,29,juin-2,july,1987