N° d'ordre : 293

THESE

présentée à

L'UNIVERSITE DE SCIENCES ET TECHNIQUES DE LILLE FLANDRES-ARTOIS

en vue de l'obtention du titre de

DOCTEUR

Spécialité : Productique, Automatique, Informatique Industrielle

par

BAYART Mireille

Ingénieur EUDIL

Conception d'un système de pilotage Temps Réel pour une cellule d'assemblage automatisé

soutenue le 20 Décembre 1988 devant la Commission d'Examen

Membres du jury :			
	P.	VIDAL	Président
	L.	PUN	Rapporteur
	J.C.	GENTINA	Rapporteur
	M.	STAROSWIECKI	Directeur de recherche
	R.	GESLOT	Invité
	D.	JOLLY	Invité

A mes parents

AVANT-PROPOS

Le travail présenté dans ce mémoire a été effectué au Centre d'Automatique de l'Université de Sciences et Techniques de Lille-Flandres-Artois.

Je tiens à exprimer ma profonde reconnaissance à Monsieur le Professeur P. VIDAL, directeur du Centre d'Automatique de Lille, pour l'accueil qu'il m'a réservé dans son laboratoire et pour l'honneur qu'il me fait de présider le jury de cette thèse.

Je tiens à exprimer ma profonde gratitude à Monsieur L. PUN, Professeur à l'Université de Bordeaux, pour avoir bien voulu apporter sa contribution au jugement de ce travail.

Je remercie très vivement Monsieur le Professeur J.C. GENTINA, Directeur de l'Institut Industriel du Nord, pour avoir accepté d'examiner ces travaux et de faire partie de ce jury.

Je suis très reconnaissante à Monsieur M. STAROSWIECKI, Professeur à l'Université de Lille, pour la patience et la confiance qu'il a eues à mon égard, le soutien et la qualité de l'encadrement qu'il m'a accordés.

Que Monsieur D. JOLLY, Maître de Conférence à l'université de Lille accepte mes remerciements pour l'intérêt qu'il a porté ce travail.

J'exprime toute ma gratitude à Monsieur R. GESLOT, Directeur Adjoint du département Production Mécanique du C.E.T.I.M., qui me fait un grand honneur en participant à ce jury.

Je tiens à remercier très amicalement les personnels des laboratoires d'I.E.E.A. et métrologie de l'E.N.S.A.M. pour la sympathie et l'amitié que j'ai trouvées en eux et tout particulièrement Messieurs J. P. FLUSIN et D. WALBROU.

Mes remerciements vont enfin à Madame A. PIGNON du Centre d'Automatique pour l'aide qu'elle m'a toujours apportée et à Monsieur BOUVET et Madame COGOI de l'E.N.S.A.M. qui assure la reproduction de ce mémoire.

SOMMAIRE

CHAPITRE 2 - MODELISATION DE LA CELLULE

CHAPITRE 3 - MODULE DE DECISION

CHAPITRE 4 - ARCHITECTURE FONCTIONNELLE

Notations utilisées

X_t	état du produit en cours à l'instant t
R_t	état des robots à l'instant t
E_t	état de l'environnement à l'instant t
U_t	commandes admissibles à l'instant t
t	instant d'observation
T	ensemble des tâches
A(i)	ensemble des antécédents de i,
A(i)	cardinal de l'ensemble A(i)
Af(i,t)	ensemble des antécédents de i qui sont finis à l'instant t
Af(i,t)	cardinal de l'ensemble Af(i,t)
S(i)	ensemble des successeurs de i
RC	ensemble des ressources consommables
RC(t)	ensemble des ressources consommables présentes à l'instant t
R(i)	ensemble des ressources de i
Γ^{-1}(i)	ressource consommable utilisée par i
RR(i)	ensemble des ressources récupérables nécessaires à l'exécution de i
RC_k	ensemble des ressources consommables de type k
n_k	cardinal de l'ensemble RC_k
t_i	date de début de la tâche i
p_i	durée de la tâche i
c_i	date de fin de tâche i
$CR_i(t)$	capacité du robot i à l'instant t
SS(t)	ensemble des ressources dans le stock statique à l'instant t
SD(t)	ensemble des ressources dans le stock dynamique à l'instant t
EXEC(t)	ensemble flou des tâches exécutables à l'instant t
<u>EXEC(t)</u>	ensemble des tâches strictement exécutables à l'instant t
EXEC(t)	ensemble flou des tâches qui ne sont pas strictement exécutables à l'instant t
<u>EXECI(t)</u>	ensemble des tâches strictement exécutables à l'instant t par le robot I, I = 1, 2
PREA(t)	ensemble flou des tâches pré-exécutables par rapport aux antécédents à l'instant t
PRER(t)	ensemble des tâches pré-exécutables par rapport aux ressources à l'instant t
ATT(t)	ensemble des tâches en attente d'exécution à l'instant t
ENC(t)	ensemble des tâches en cours d'exécution à l'instant t

$P_s(t,\delta t)$	ensemble des ressources sorties de la cellule pendant l'intervalle $[t,t+\delta t]$
$P_e(t,\delta t)$	ensemble des ressources entrées dans la cellule pendant l'intervalle $[t,t+\delta t]$
T_i	date de disponibilité du robot i, i = 1, 2
T_v	temps d'accessibilité à la zone commune
T_{tk}	temps de transport de la ressource k de l'entrée de la cellule au poste de prise
T_{ak}	temps d'attente de la ressource k au poste de prise
T_{uk}	date d'utilisation de la ressource k située sur le stock dynamique
Φ(EXEC)	mesure de la quantité de travail apporté
Φ_k(EXEC)	mesure de la quantité de travail apporté par une ressource de type k
β	coefficient d'actualisation

INTRODUCTION

GENERALE

La diminution de la durée de vie des produits associée à l'augmentation du nombre de variantes des produits fabriqués ont rendu le concept de flexibilité essentiel pour l'assemblage automatisé.

Cette flexibilité ne peut être obtenue qu'en transformant le produit et les équipements de production, et la situation est correctement résumée dans (AND 86) : "la question de savoir si le système de production doit être adapté aux produits ou les produits adaptés au système est la même que celle de l'oeuf et la poule, qui commence le premier ?"

Cette remarque justifie les deux grandes catégories de travaux effectués dans le cadre de l'assemblage automatisé, celle relative à la conception du produit et celle relative à la réalisation du produit.

C'est dans cette deuxième classe que se situe l'étude que nous présentons dans ce mémoire, à savoir, la conception d'un système de pilotage pour une cellule flexible d'assemblage dans laquelle deux robots coopèrent à la réalisation d'un produit commun.

La participation de plusieurs robots à l'exécution d'un travail offre un certain nombre d'avantages. Elle permet, par exemple, si de nombreux outils sont nécessaires d'éviter de fréquents changements. Elle autorise la réalisation de tâches exigeant la disponibilité de deux robots telles que le transport d'objets encombrants, l'insertion d'un composant par un robot dans un élément maintenu par l'autre, etc. Si, de plus, chaque robot est capable de réaliser seul l'ensemble des tâches, une panne de l'un d'eux n'entraîne pas un arrêt de la production, et n'aura pour effet qu'une augmentation du temps d'exécution des tâches.

En fait, au niveau matériel, la flexibilité ne provient que des robots, ceux-ci sont considérés comme des systèmes universels mais les équipements périphériques, pinces, outils, procédés d'alimentation ne possèdent pas, en général, la même adaptabilité et versatilité.

La flexibilité d'une cellule d'assemblage dépend également de la puissance des logiciels de commande et de gestion.

Quelle que soit la complexité du montage à effectuer, il peut toujours être décomposé en sous-assemblages plus simples, eux-mêmes décomposés en tâches élémentaires. Il existe couramment des contraintes de réalisation entre les tâches mais également des possibilités de parallélisme qui engendrent diverses séquences d'assemblage.

Généralement, l'ordonnancement est réalisé "hors ligne", puis chargé sur les moyens de production. Tout incident provoque alors l'arrêt de production. La flexibilité du système de conduite devrait en fait permettre de détecter l'incident (panne d'un robot, problème dans l'alimentation en ressources consommables, ...) de déduire les modifications sur l'ordonnancement, puis de commander le système en conséquence.

Un autre élément important est la notion de temps d'exécution d'une tâche par un robot. Il n'est pas toujours possible de connaître à l'avance, la durée exacte d'une tâche. Selon les paramètres de prise, l'usure des outils, la précision requise, les variations des positions des pièces, ..., des fluctuations dans le temps vont apparaître. L'ordonnancement statique doit alors être réalisé sur une valeur maximum, ce qui peut créer pendant l'exécution de la tâche, un temps d'oisiveté du robot.

Un ordonnancement dynamique permet la prise en charge de ces différents éléments, dans la mesure où l'ordonnancement est réalisé en ligne en fonction de la disponibilité des robots, de l'arrivée des pièces et, de façon générale, des modifications de l'environnement.

La flexibilité de la cellule est alors obtenue grâce au système de pilotage Temps Réel, qui effectue un ordonnancement dynamique.

Dans le cadre du pôle productique, le laboratoire d'Automatique de l'U.S.T. de Lille-Flandres-Artois a développé un atelier flexible expérimental dans le domaine de la confection. Notre étude s'applique à un élément de cet atelier, à savoir le poste d'assemblage automatique à l'aide de deux robots.

Dans le premier chapitre, nous présentons les différents problèmes relatifs à l'assemblage automatisé, au niveau de la conception et des systèmes de production. Nous décrivons ensuite le site expérimental, support de notre étude, ainsi que les caractéristiques principales de la cellule d'assemblage.

La structure du système de conduite proposée pour le pilotage de la cellule fait l'objet de la première partie du second chapitre. Puis la modélisation, élaborée à partir de graphes d'état, des différents constituants de la cellule est détaillée. Les tâches sont regroupées, selon leur état, en différents ensembles flous ou vulgaires, que le module d'information actualise à chaque modification de l'état de la cellule.

Dans le troisième chapitre est défini le problème de la conduite de la cellule, c'est à dire l'ordonnancement-affectation des tâches aux robots en fonction de l'assemblage en cours, de l'arrivée des ressources consommables et de la disponibilité des ressources partageables. L'objectif est la minimisation du temps global de réalisation du produit.

La sélection est basée sur la quantité de travail apporté au système par l'exécution d'un couple de tâches. Pour limiter l'attente des robots, l'affectation est réalisée de manière prévisionnelle.

La méthode de résolution est développée pour une représentation de la gamme d'assemblage par graphe, puis nous envisageons le cas d'une description par arbre.

Le quatrième chapitre aborde la structure fonctionnelle du système de pilotage proposé. Les différentes fonctions composant les modules d'information et de décision ainsi que leurs interconnexions sont présentées.

CHAPITRE 1

L'ASSEMBLAGE AUTOMATISE

I INTRODUCTION

Aujourd'hui, pour des motifs économiques (abaissement du coût, amélioration de la productivité pour faire face à la concurrence internationale), techniques (amélioration de la qualité,..) et sociaux (amélioration des conditions de travail en supprimant les tâches répétitives et pénibles), les processus de fabrication des produits industriels sont de plus en plus automatisés.

Cependant, la notion de systèmes flexibles de production désigne des structures industrielles profondément différentes et il est difficile de recenser toutes les installations. Le concept de flexibilité est, de plus, différent d'un pays à l'autre (DUP 83).

Le nombre de cellules flexibles de production implantées augmente (BON 85), elles sont généralement caractérisées par un chargement et un déchargement automatiques et un stock de pièces qui leur donnent une certaine autonomie.

Les systèmes flexibles désignés sous le terme de F.M.S. (Flexible Manufacturing System) font leur apparition avec par exemple en France, l'atelier de Renault Véhicules Industriels à Bouthéon (DUP 82), l'atelier Citroën de Meudon ou l'usine Bull de Villeneuve d'Ascq (IAC 87).

L'étape suivante, qui est encore au stade de l'expérimentation, est l'usine entièrement automatisée et flexible nommée C.I.M. (Computer Integrated Manufacturing) dans la littérature. Elle intègre l'ensemble des fonctions de l'entreprise, de la conception à la réalisation des produits en passant par la gestion du système de production (RUS 87).

La plupart des travaux réalisés dans ce domaine concernent des applications mécaniques (MER 87), (OKH 84). De plus, si on excepte l'insertion de composants sur les circuits imprimés, exemple souvent rencontré dans la littérature (SAN 83), (SOL 84), l'assemblage reste essentiellement manuel.

Le système de pilotage que nous présentons dans ce mémoire, est destiné à une cellule flexible d'assemblage, nous nous sommes donc intéressés aux travaux effectués dans ce domaine.

Dans la première partie de ce chapitre, nous présenterons, sans prétendre être exhaustif, différents travaux de recherche sur l'assemblage automatique. Nous verrons ainsi que pour un tel processus, les éléments à assembler doivent être conçus en vue d'une automatisation de leur montage. D'autre part, s'il est nécessaire de tenir compte du système d'alimentation et des caractéristiques du robot, la flexibilité d'une cellule d'assemblage dépend également des logiciels de commande et de gestion. Dans un deuxième temps, nous décrirons la cellule flexible qui est utilisée comme base pour notre étude.

II STRUCTURE FONCTIONNELLE

II.1. La conception du produit

Lorsque l'on désire automatiser un assemblage, se pose le problème du choix entre plusieurs systèmes :

- Les machines automatiques classiques destinées à un type de produit.

- Le robot sophistiqué que l'on programmera selon les différents produits.

- L'utilisation de robots simples aux performances limitées.

Si la machine spécialisée reste prépondérante dans les assemblages en grande série, le robot est de plus en plus utilisé quand la flexibilité est le critère principal de choix.

Dans ce cas, c'est la simplicité qui l'emporte et les industriels optent généralement pour un robot élémentaire moins performant mais aussi moins cher qu'un robot complexe. Les robots simples type "pick and place" ont des performances plus faibles, ils sont conçus pour assurer des opérations d'insertion dans des plans orthogonaux, ils réalisent des assemblages élémentaires et nécessitent donc une révision complète du produit à monter. La reprise de la conception est une étape utile pour la robotisation de l'assemblage, elle engendre souvent une simplification de la construction. On peut citer le cas d'Hitachi (PER 86) qui, en redessinant le mécanisme mobile d'une tête de lecture de magnétoscopes, a réduit de 460 à 379 le nombre de composants élémentaires.

La diminution du nombre de composants permet une facilité d'automatisation de l'assemblage en diminuant le nombre de systèmes d'alimentation et le nombre d'opérations à effectuer. Dans ce sens, elle réduira le prix de revient du produit tout en améliorant sa fiabilité.

Pour automatiser une opération d'assemblage, il est souvent nécessaire de revoir la conception du produit. Il existe de nombreuses régles à observer pour faciliter l'automatisation (AND 86), (LAS 84), (FOR 84). A titre d'exemple, nous en citerons quelques unes :

- S'il est facile pour un opérateur de tourner un assemblage et de travailler sur toutes les faces, ceci nécessitera plus de degrés de liberté au robot et un équipement périphérique adapté. Une première considération sera donc d'éviter ces rotations et de concevoir un montage uni-directionnel pour permettre l'utilisation d'un robot plus simple et une diminution du temps de cycle.

- En dépit de leur exactitude et répétabilité, les robots ne peuvent jamais se déplacer exactement en un point précis. Les variations des pièces, fabriquées avec une certaine tolérance, et celles du robot doivent être considérées dans la conception du produit ; ainsi, par exemple deux pièces chanfreinées s'ajusteront plus aisément même si elles ne sont pas parfaitement alignées.

- L'utilisation de pièces symétriques permet une alimentation et un assemblage dans n'importe quelle direction et réduit ainsi le nombre de capteurs.

- Pour les pièces asymétriques, on exagérera cette asymétrie pour faciliter le transport et la prise par un effecteur.

- Les pièces qui doivent être jointes le seront plus facilement si elles sont prévues pour être assemblées par un simple encliquetage au lieu d'utiliser un système de fixation type boulon. Il est possible d'intégrer les éléments de fixation (cliquets, ergots) dans la fabrication des produits. Ceci élimine une partie des composants et des systèmes d'alimentation donc réduit le coût de l'installation et augmente la cadence du dispositif d'assemblage.

- Si l'utilisation d'attaches de type boulons, vis, rivets ne peut être évitée, on prévoira l'emploi de fixations de même type et même taille pour l'ensemble de l'assemblage. Un dégagement adéquat de l'outil autour de ces attaches devra alors être envisagé.

Ces quelques régles font partie d'un ensemble utilisé pour la conception du produit. D'autres études ont été réalisées pour classer les opérations d'assemblage selon leur axe privilégié et les types de mouvements. Ainsi il apparait que 80 % des pièces sont introduites selon un axe vertical (60 % vers le bas, 20 % vers le haut), sur les 20 % restants, 10 % des composants sont mis en place horizontalement et les 10 % derniers selon un axe quelconque (JEA(b) 86).

On trouve également une classification des assemblages selon la méthode de fabrication (OWE 86), ainsi sont distingués :

- le montage sur cadre, les composants sont assemblés sur un châssis comme pour les calculateurs. C'est l'assemblage de plusieurs éléments discrets sur un objet tridimensionnel plus grand.

- le montage par empilement pour lequel les éléments sont assemblés de façon séquentielle, les uns sur les autres, suivant une direction unique.

- le montage sur support comme pour les circuits intégrés, il désigne l'assemblage de plusieurs éléments discrets sur un objet bidimensionnel plus grand.

- le montage modulaire où les sous-assemblages sont combinés de façon différente en fonction du produit à réaliser. Il se caractérise par l'assemblage de plusieurs sous-assemblages entre eux.

Quel que soit le type de montage, l'automatisation des tâches d'assemblage nécessite souvent une reconception des produits (JEA(a) 86) qui sera combinée avec l'élaboration et le choix des composants de la cellule d'assemblage, (dispositif d'alimentation, de maintien, préhenseurs, robots, ..).

II.2. L'alimentation des pièces

La forme de stockage des pièces la plus courante reste le vrac volumétrique. De ce fait, l'alimentation est actuellement réalisée soit manuellement, soit à l'aide de dispositifs automatiques spécialisés qui fournissent les pièces isolées dans une position et orientation bien déterminées.

Un produit est constitué d'un ensemble de matériaux que T. OWEN classe en trois groupes (OWE 86) :

- des éléments discrets (vis par exemple) dont l'alimentation est réalisée par des distributeurs comme les bols vibrants, les dispositifs de distribution par gravité, les tournevis automatiques.

- des matériaux consommables (colles, adhésifs), l'alimentation est alors effectuée par des conteneurs, la quantité de matériau dispensé est alors programmée.

- des composants qui sont distribués par :

* des convoyeurs qui indexent un objet devant le poste de travail ou fonctionnent en continu,

* des chariots automoteurs filoguidés qui permettent une plus grande flexibilité.

Les pièces sont alors disposées en rangées, sur palettes, dans des plateaux alvéolés, ..., de manière à avoir une position connue.

D'autres systèmes d'alimentation sont réalisés pour des éléments particuliers (GRO 75),(JAC 84).

Là encore, dans la conception du produit, on prévoira un dispositif d'alimentation qui facilite le transport, le positionnement et la préhension des pièces. Dans ce sens, les pièces alimentées par glissement ne devront pas s'enchevêtrer, ni se coincer aussi certaines régles seront à respecter, par exemple :

- éviter les bords pointus pour que les pièces ne s'accrochent pas entre elles.

- utiliser des pièces à base plate afin que les pièces ne se chevauchent pas.

- prévoir le centre de gravité le plus bas possible pour augmenter la stabilité des composants pendant l'alimentation et l'assemblage.

- faciliter la stabilisation et empêcher les chevauchements en créant, par exemple, au centre de la pièce, une rainure parallèle à la direction du transporteur.

- ...

Outre ces considérations, le choix du système de manutention devra être adapté. Des études par simulation sont effectuées pour faciliter cette sélection (ABB 87) et pour définir la stratégie de lancement de produits dans l'atelier (LEV 82).

II.3. L'implantation du robot

Le cahier des charges permet le choix d'un robot, par la définition des effecteurs, de la morphologie, de la périphérie et des liaisons homme-machine.

Rapidité, précision, répétabilité font partie des qualités de base exigées d'un robot d'assemblage. En fait, un robot très rapide ne travaillera qu'un court instant à sa vitesse nominale aussi la vitesse n'est pas un bon critère et on s'oriente vers la mesure du temps de cycle (SEP 85) qui correspond à la réalisation d'une opération : prise de pièce, transport et positionnement précis. Il est à noter également qu'une précision plus importante du manipulateur coûtera plus cher que la réalisation d'un chanfrein au niveau d'une pièce.

On notera à ce sujet les travaux de (JEA(b) 86) sur les caractéristiques opératoires des robots d'assemblages.

On pourrait citer dans ce paragraphe, les nombreux travaux effectués sur les préhenseurs, les poignets compliants, les capteurs, ... En effet, dans la conception du produit, le problème des préhenseurs intervient également et si certaines pièces (plates, larges, ..) sont faciles à saisir par des systèmes magnétiques ou par ventouse, pour les autres, on prévoira des trous, des rainures ou des pattes pour faciliter la saisie.

Ces différentes recherches appartiennent au domaine de la mécanique et de la fabrication. Nous présenterons dans le paragraphe suivant, les systèmes de commande qui relèvent du domaine de l'automatique et de l'informatique.

III STRUCTURE INFORMATIONNELLE

III.1. Notion de tâche

Un robot industriel doit être capable d'exécuter des tâches matérielles. En effet à la différence d'un ordinateur, il doit réaliser des interventions physiques sur un "environnement" de travail (LHO 82). On lui demandera également d'exécuter des tâches diversifiées à la différence d'une machine automatique traditionnelle conçue pour réaliser une tâche spécifique.

Un des points critiques de l'automatisation de l'assemblage est justement le développement de programmes d'exécution de tâches diverses et complexes pour un ou plusieurs robots.

La commande des robots utilisés pour la réalisation d'un assemblage donné nécessite plusieurs étapes :

- la décomposition de l'assemblage en différents sous-assemblages,

- la description de chaque sous-assemblage en un programme de niveau manipulateur,

- la spécification des synchronisations entre les actions élémentaires.

La modélisation des processus d'assemblage est un sujet d'études actuellement répandu, et on trouve de nombreuses approches dans la littérature.

Ainsi, on peut citer les travaux de M. TEMANI (TEM 85) sur l'élaboration automatique de gammes de montage d'un produit à partir d'une base de données comportant des informations sur le processus de fabrication, la définition de la nomenclature et des traitements spécifiques associés à ce produit.

Les travaux de P. MAGUET (MAG 82) sur la décomposition en assemblages élémentaires et leur enchaînement sont, quant à eux, orientés vers la robotisation des postes d'assemblage.

Les études de A. BOURJAULT (BOU 84), (BOU(a) 87) concernent l'élaboration automatique des séquences opératoires d'assemblage à partir des liaisons fonctionnelles entre les composants.

Enfin, la décomposition en tâches élémentaires pourra être réalisée à l'aide de l'Intelligence Artificielle (KEM 85). C'est l'objectif du Système Expert LMAD (ARN 84) qui devrait permettre la décomposition d'une tâche manufacturière en tâches élémentaires avec leur ordre d'exécution, le traitement de ces tâches permettant de déduire les actions à effectuer à l'aide d'une cellule robotisée. Les séquences programmées seront alors transformées en unités de production pour la commande de procédés sous LMAC (BON 82), (BON 84), (ELL 83).

Ces différentes études des processus d'assemblage ont pour but la définition des différentes tâches de production nécessaires à l'application et le choix des constituants de la cellule. Parallèlement à la résolution de ce problème, il est nécessaire de concevoir un système de contrôle-commande automatique en temps réel de l'ensemble de la cellule.

Les travaux de recherche s'orientent pour cela vers un système autonome, capable de s'adapter à un environnement variable en essayant en particulier de rendre les robots plus "intelligents".

III.2. Système de contrôle-commande

La conduite en temps réel d'une cellule est un problème de commande en boucle fermée dans la mesure où l'on doit réagir aux perturbations en prenant en compte l'état réel de l'atelier.

Dans ce sens, le système de contrôle-commande assure :

- la mesure des grandeurs nécessaires pour observer l'évolution de la cellule,

- la prise en compte des événements survenant aléatoirement,

- l'évaluation des décisions à partir des informations précédentes,

- la génération des grandeurs de commande pour assurer le fonctionnement de la cellule.

Parmi les fonctions de contrôle, nous pouvons citer :

- le contrôle de l'état du produit en cours,

- la vérification de la qualité des produits (détection de mauvaises pièces),

- la connaissance de l'état des machines et/ou robots (libre, occupé, en panne),

- le contrôle de la qualité et de l'état des outils,

- le contrôle du système d'alimentation,

- ....

Ces informations sont reçues en provenance des capteurs.

Au niveau de la commande, les fonctions sont essentiellement :

- l'affectation des tâches aux moyens de production,

- la génération des commandes pour les robots et/ou machines,

- la gestion du système de transport en fonction des besoins,

- la gestion des changements d'outils,

- la communication avec un opérateur,

- ....

Parmi ces différentes fonctions, nous allons détailler plus précisément le problème de la commande des robots.

III.3. Commande des robots

On s'intéresse ici aux robots programmables parmi lesquels on distingue différentes classes selon la méthode de programmation (PAR 83), (HAU 87) :

- par apprentissage, l'opérateur positionne, par déplacement manuel ou assisté, le robot dans les configurations qui doivent être enregistrées ou lui "montre" la trajectoire qu'il doit accomplir.

Dans ce cas, la programmation est rigide dans le sens, où le déroulement des séquences est toujours le même. L'interface avec l'environnement est alors réalisé par

attente d'événements extérieurs et/ou envoi de signaux vers l'extérieur (signaux logiques) pour déclencher une action.

L' apprentissage est réalisé sur le site de production (programmation sur site ou "on-line"). Les techniques que nous allons présenter à présent, entrent dans le cadre de la programmation hors-site (off-line), c'est-à-dire sans utilisation du robot ni de son système de contrôle.

- par simulation, grâce aux systèmes de C.A.O. qui à partir de la modélisation de la tâche et du robot, permettent de tester et vérifier plusieurs scénarios et de programmer le robot d'une manière se rapprochant de la précédente, un simulateur graphique remplaçant le robot. Si actuellement, ces systèmes offrent une programmation sûre et efficace des robots industriels, des améliorations sont attendues quant à l'intégration des capteurs et à la prise en compte des incertitudes dans le fonctionnement d'une cellule robotisée.

- par langage, à partir du jeu d'instructions disponibles, l'utilisateur écrit un programme qui est ensuite interprété pour piloter les actions du robot.

Dans cette catégorie, il existe également différents niveaux :

- La programmation explicite :

au niveau actionneur, l'utilisateur programme les déplacements des actionneurs de façon coordonnée ou non,

au niveau effecteur, où l'on programme les déplacements de l'outil,

Dans ces modes, les séquences de mouvements du robot ainsi que les interactions avec les systèmes périphériques sont bien définies.

- La programmation implicite :

au niveau objet où les spécifications des tâches sont données en termes d'opérations sur les objets. Les mouvements du robot sont alors générés à partir de la description de l'environnement et de la modélisation des pièces à manipuler. En ce qui concerne les programmes d'assemblage, des études sont menées pour résoudre les problèmes tels que la saisie, le transport et le montage de l'objet à assembler (DUF 84), (LAP 80), (LOZ 76), (LOZ(b) 83).

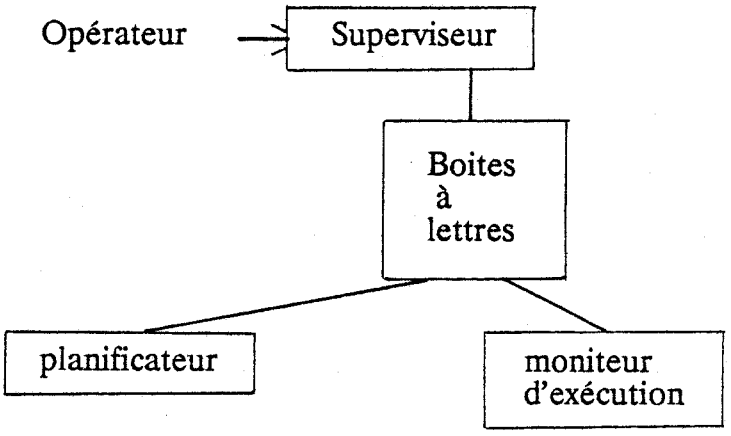
au niveau objectif, c'est la généralisation de la programmation au niveau objet, et seul l'objectif final est stipulé sans détailler les étapes intermédiaires.

L'idée d'apporter une "intelligence" au robot n'est pas récente et l'on peut par exemple, citer les travaux de Ejiri (EJI 72) sur le contrôle d'un robot qui analyse les dessins des trois vues d'objets et décide comment manipuler ces objets pour réaliser un assemblage.

Les exemples actuels s'appuient sur des outils d'Intelligence Artificielle tels que les générateurs de plans d'actions qui définissent les suites d'actions à réaliser pour atteindre l'objectif voulu. La planification peut être réalisée hors ligne, (CLE 86), (DUE 86) et le système est généralement organisé de manière hiérarchique avec au niveau supérieur la planification de l'exécution des tâches et au niveau inférieur les programmes d'exécution des mouvements des robots. Ce plan est ensuite transféré au superviseur de la cellule qui en assure la réalisation et le contrôle.

La génération de plans peut également être effectuée "en ligne" (VER 80). Cependant, si on reconnait à l'Intelligence Artificielle sa capacité à résoudre des problèmes complexes et la possibilité d'augmenter au fur et à mesure la base de connaissances, un des points critiques reste sa lenteur. Pour résoudre ce problème, une approche de structure parallèle a été développée par l'équipe d'I.A. et Robotique de Toulouse (PIC 86) pour gérer une cellule flexible d'assemblage composée d'un robot, d'un tapis roulant, d'une table tournante et d'une zone d'assemblage.

Le système MUCAR est divisé en trois modules écrits en LISP et fonctionnant en parallèle.



Le superviseur dépêche le travail aux deux autres modules. Le planificateur établit le plan d'exécution, compte tenu de l'état courant ou de l'état anticipé, c'est à dire correspondant à l'état en fin d'exécution du plan en cours. Le moniteur d'exécution établit le plan d'actions élémentaires du manipulateur.

III.4. Contrôle

Tout système de production est soumis à des perturbations. Celles-ci, diverses et nombreuses, ont été recensées dans (BEH 83) et concernent des problèmes sur les ressources mécaniques, informatiques, ..

Ces perturbations entraînent un mode de fonctionnement dégradé qui peut être géré automatiquement, suivant le type de pannes, ou manuellement. Il n'existe pas encore de systèmes permettant une analyse de tout incident et une prise de décision pour la poursuite de l'exécution après correction de la cause de l'anomalie.

Les pannes prises en compte généralement sont celles liées aux machines, aux outils et aux matières premières (THO 80).

Un autre axe de recherche sur la fonction " intelligence " est le contrôle de bonne exécution et la capacité de réagir aux différents incidents.

Des solutions sont proposées par une approche Intelligence Artificielle (AIR 84), (COR 79), (GLI 84) où les différents incidents pouvant se produire sont traités.

Ces différents travaux sur la conception, le choix des équipements, la commande et le contrôle ont pour objectif l'augmentation de la flexibilité des systèmes d'assemblage. Nous allons voir, à présent, que l'implantation contribue à l'augmentation de la flexibilité et que le choix de l'installation dépend des objectifs de production.

IV LES LIGNES D'ASSEMBLAGES

Si on trouve couramment des lignes d'assemblage où chaque module effectue un seul pas dans le procédé de montage, une alternative à ce modèle est la ligne d'assemblage parallèle composée de cellules multitâches identiques dans chacune desquelles l'ensemble des pas est réalisé. Nous allons présenter les avantages de ce type de cellule compte tenu de l'évolution du produit.

IV.1. ligne d'assemblage série

Dans une ligne d'assemblage série (fig. I.1.), chaque module effectue une tâche du procédé d'assemblage. Le produit est transporté le long de la chaîne de poste en poste au moyen d'un convoyeur ou de palettes. A chaque station, la palette est arrêtée et l'opération réalisée. Compte tenu de la simplicité de leur tâche, les robots utilisés dans ces postes peuvent n'avoir que quelques degrés de liberté, des capacités de décision limitées et un langage de programmation de bas niveau. Ils sont équipés de l'outil nécessaire pour effectuer l'opération qui leur est allouée.

Dans une telle configuration, le taux de production est fonction du poste le plus lent et une panne entraîne généralement l'arrêt de la chaîne.

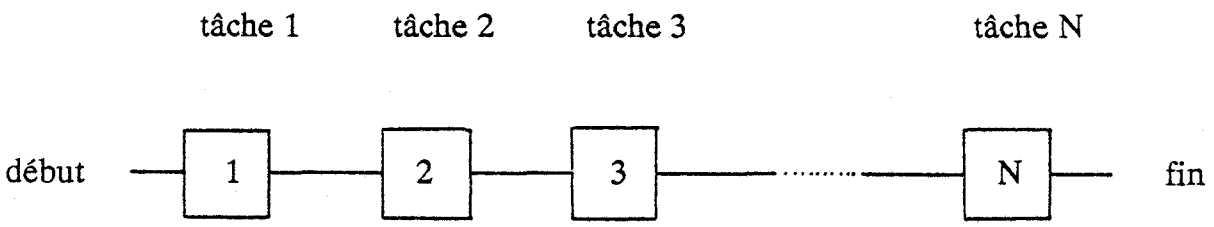


fig. I.1.

IV.2. ligne d'assemblage parallèle

Par opposition aux lignes d'assemblage série, une ligne d'assemblage parallèle (fig. I.2.) est composée de plusieurs cellules identiques qui effectuent toutes les opérations sur le produit.

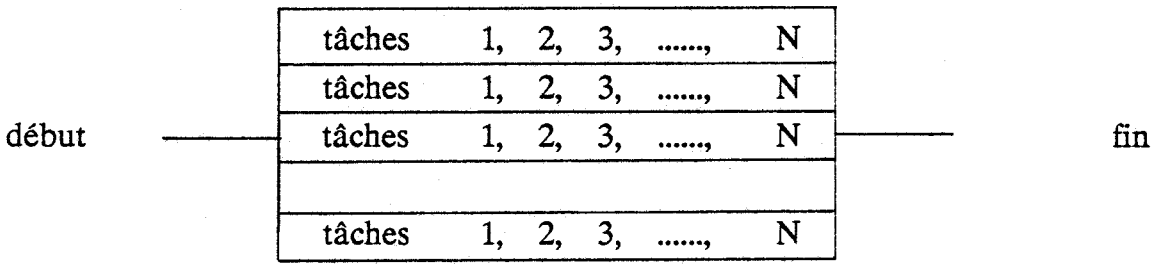


fig. I.2.

Chaque cellule est dotée des équipements nécessaires à la réalisation de toutes les opérations d'assemblage. Les robots, munis d'un changeur d'outils, doivent être capables d'effectuer les différentes tâches et de prendre des décisions. Pour cela, un langage de programmation de haut niveau est nécessaire.

Le taux de production des cellules en parallèle est la somme des taux des différentes branches, une panne d'un des robots ne bloquant que la cellule à laquelle il appartient, il ne réduit que le taux de production de celle-ci.

La mise en service, plus complexe que dans le cas d'un assemblage série, est effectuée sur une cellule puis reproduite sur les autres.

IV.3. justification économique

D'un coût plus élevé, la ligne parallèle possède un avantage majeur qui résulte de sa flexibilité et sa capacité à s'adapter à la demande de production.

Considérons les diagrammes de cycle de vie d'un produit typique (fig. I.3.) :

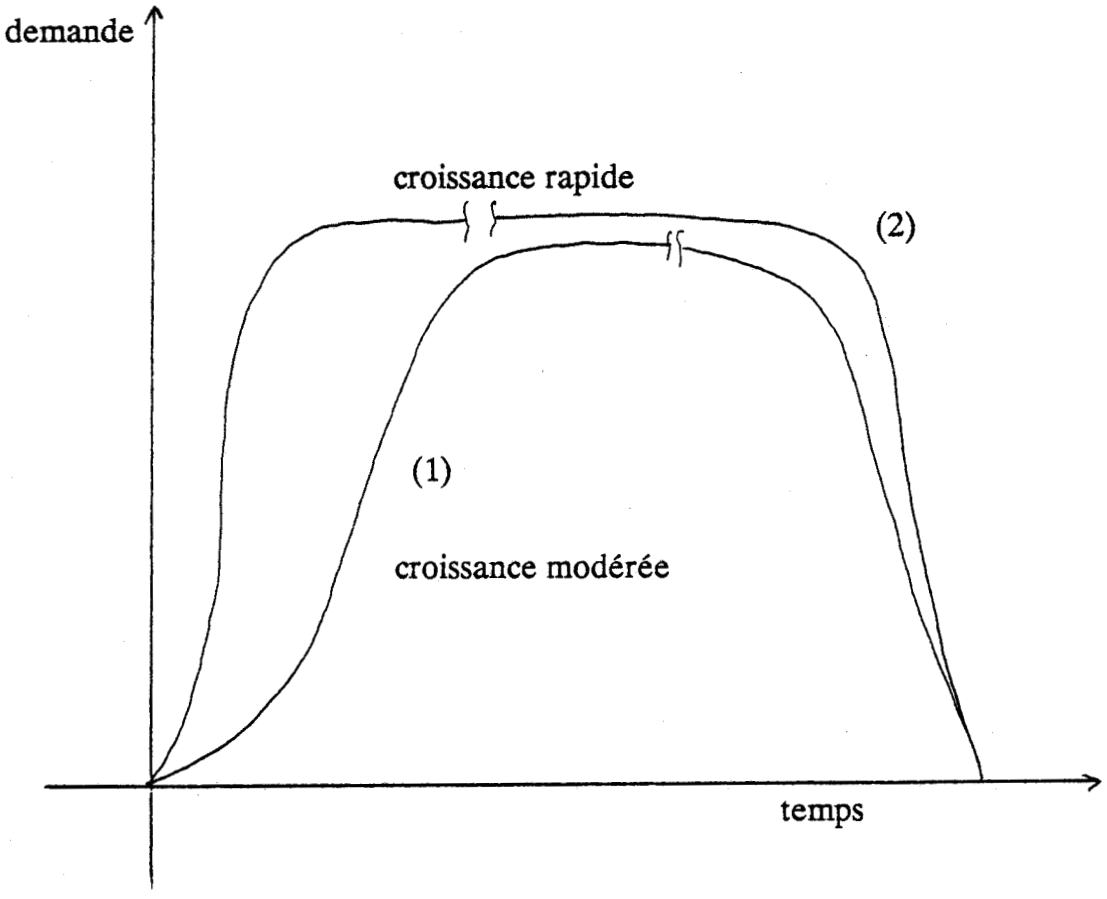
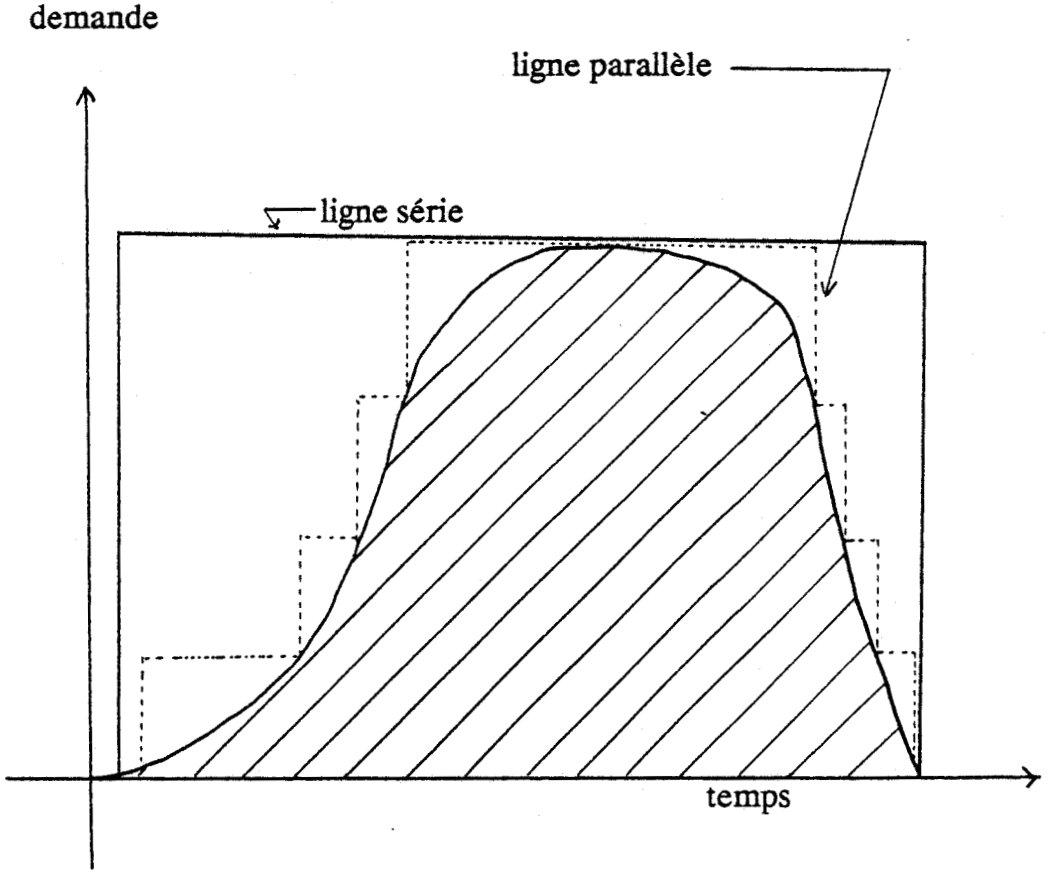


fig. I.3.

La courbe (1) représente l'évolution d'un produit, dont la demande faible au départ augmente au fur et à mesure que le produit devient connu, puis la demande reste stable jusqu'à la fin de la vie du produit.

La seconde courbe représente un produit dont le succès est immédiat et on arrive très tôt au maximum de production.

Pour une ligne d'assemblage série, l'installation est prévue pour assurer la production quand la demande a atteint son maximum. Pendant la phase de croissance du produit, en particulier si celle-ci est modérée, les équipements sont sous utilisés. Par ailleurs, les postes ne pourront être reconfigurés qu'à la fin de la vie du produit (fig. I.4.).



Implantation d'une ligne d'assemblage fig. I.4.

L'avantage de l'assemblage parallèle est la capacité de s'étendre par étapes selon la croissance du produit. On ajoute des cellules suivant l'évolution de la demande, et en fin de production on reconfigure pour une autre application les cellules qui ne sont plus nécessaires.

Enfin, pour une ligne série, l'implantation est basée sur des projections de marché. Actuellement, il est de plus en plus difficile de faire des projections de demande de produit avec une grande exactitude. La capacité d'une ligne parallèle à s'ajuster à la demande offre donc un autre avantage qui est de réduire le risque pour le capital investi. Une telle ligne à été développée par Motorola (SOL 84) pour fabriquer des modules hybrides pour des équipements de radiocommunications. Une cellule est composée de deux robots et le procédé d'assemblage comporte 12 opérations élémentaires. Le temps de production espéré se situe au milieu des deux extrêmes correspondant d'une part, au temps nécessaire dans le cas d'une ligne série c'est-à-dire

la somme des temps de chacune des opérations, d'autre part, à la moitié de cette somme, optimum obtenu si les deux robots travaillaient toujours simultanément.

C'est dans le cadre d'un assemblage parallèle que nous situons la cellule, support expérimental de notre étude, que nous présentons dans le prochain paragraphe. Après une description des différents éléments qui la composent, nous détaillerons divers problèmes liés à leur implantation.

V DESCRIPTION DU SITE EXPERIMENTAL

La cellule de fabrication qui est implantée sur le site expérimental est composée d'un certain nombre d'éléments robotiques, péri-robotique et de calcul, on y trouve :

a) Deux robots SCEMI AC06

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 leur 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.

Le principal inconvénient de ce langage est d'être "monotâche". Les robots sont donc activés séquentiellement. La seule possibilité de parallèlisme est l'utilisation de l'instruction "sansattente" qui permet de lancer un robot juste après l'autre sans attendre la fin de l'action de ce dernier. Actuellement pour résoudre ce problème, des travaux sont en cours en vue de l'implantation d'un bi-interpréteur LM de manière à pouvoir commander les deux robots en parallèle (JOL 88).

Ils sont équipés d'un changeur d'outils qui leur permet de réaliser différentes tâches d'assemblage à partir des composants acheminés par un tapis roulant.

b) un système de vision VISIOMAT

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

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

Au passage d'une pièce devant la caméra, le système de vision procéde à une prise d'image qui permet d'identifier et de localiser la pièce : détermination du type de la pièce, de son orientation et de sa position.

c) un tapis roulant

Il est linéaire et accessible aux deux robots. Il est jalonné de cellules de détection qui sont placées sous la caméra et devant chacun des robots.

d) un calculateur HP1000

L'ensemble des éléments de la cellule est géré par un calculateur HP1000 de la série A600, qui travaille sous le système d'exploitation multitâches RTEA.

La structure que nous avons choisie est illustrée par la figure I.5.

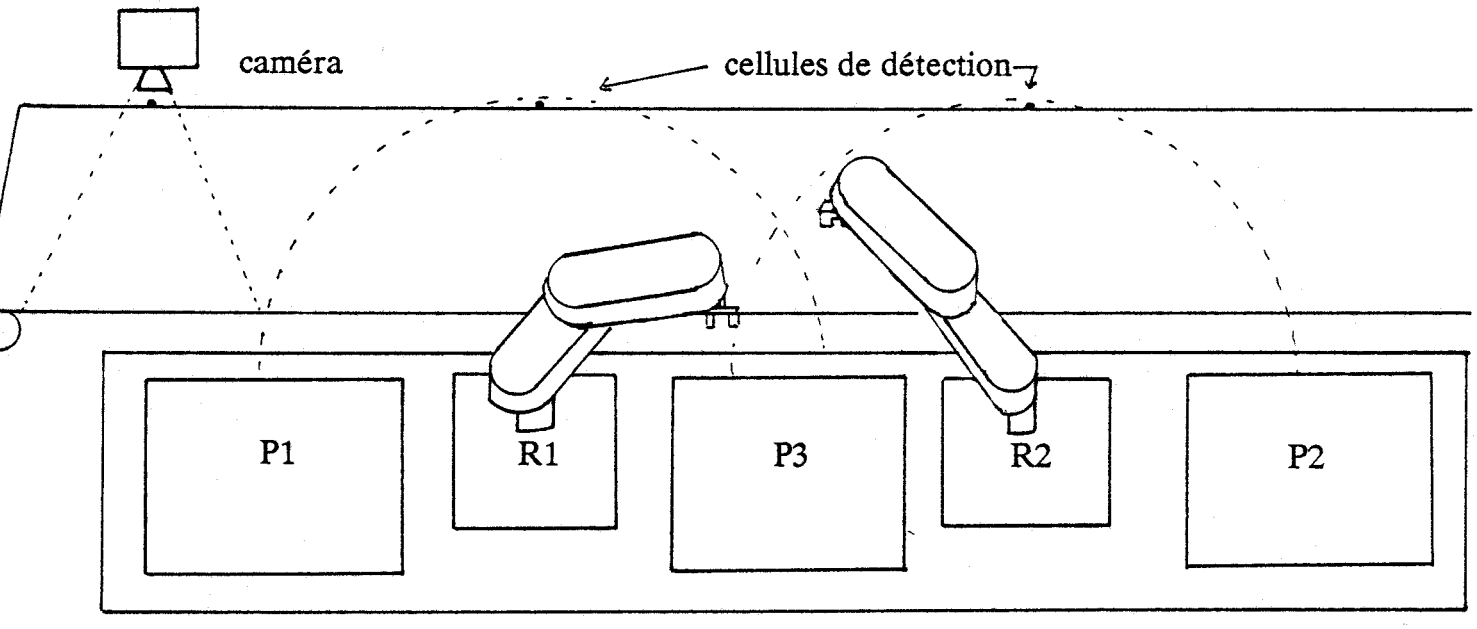


fig. I.5.

RESEAU INDUSTRIEL
DE COMMUNICATION
ENTRE MACHINES

Caractérisation Matière

Plan de coupe

Forme et position

Contrainte de stockage

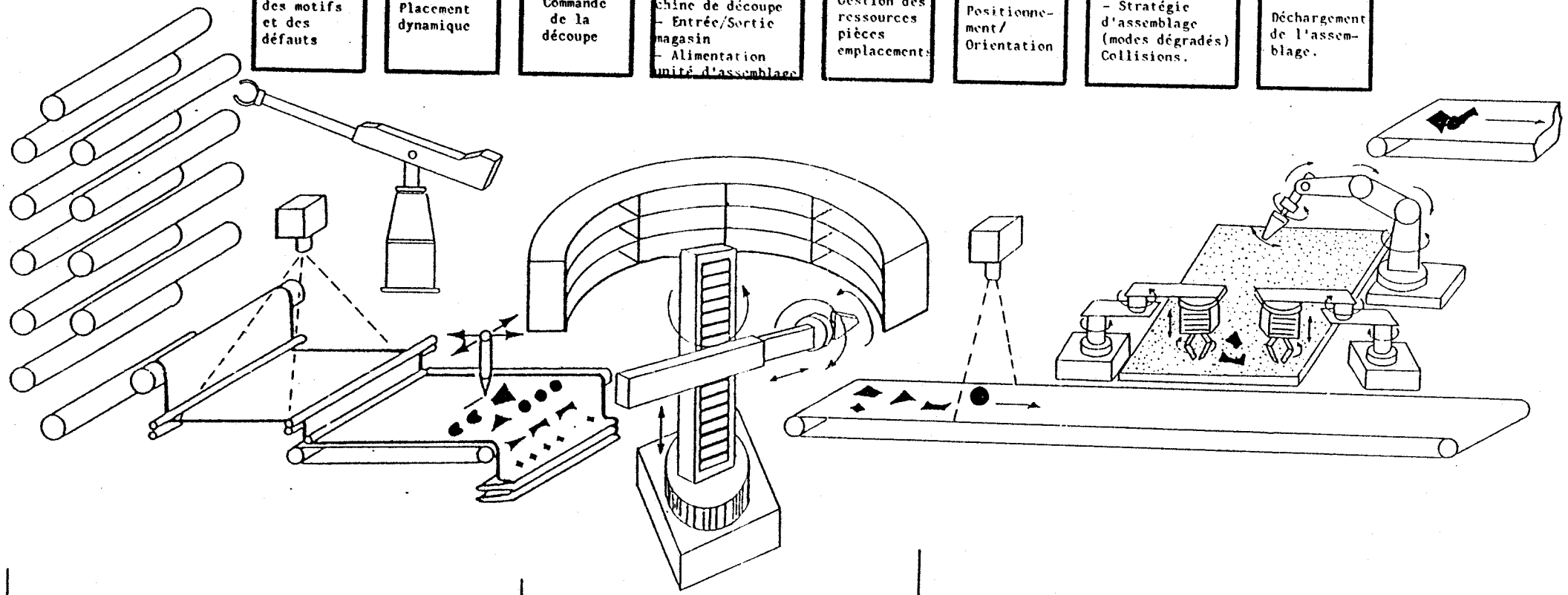
Caractérisation des pièces

Type de pièce transférée

Positionnement pièces

Validation de l'assemblage

Traitement d'image	Unité de Calcul	Unité de pilotage	Unité de transfert	Unité de stockage	Traitement d'image	Coopération entre robots	Unité d'évacuation
Détection des motifs et des défauts	Placement dynamique	Commande de la découpe	- Déchargement machine de découpe - Entrée/Sortie magasin - Alimentation unité d'assemblage	Gestion des ressources pièces emplacements	Positionnement/ Orientation	- Stratégie d'assemblage (modes dégradés) Collisions.	Déchargement de l'assemblage.



UNITE DE PREPARATION

M A G A S I N A G E

UNITE D'ASSEMBLAGE

La cellule présentée est la base de notre étude des cellules flexibles de fabrication. Elle autorise de multiples expérimentations. Son organisation pose un certain nombre de problèmes que nous allons détailler.

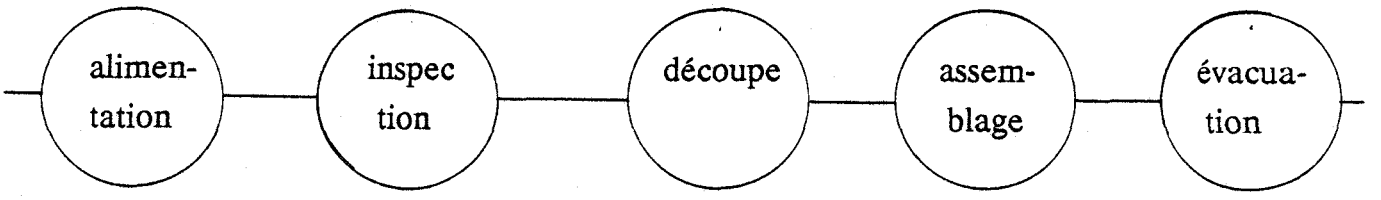
VI PROBLEMES LIES A L'IMPLANTATION

VI.1. alimentation de la cellule

La cellule présentée est un des postes de travail de l'atelier flexible expérimental dans le domaine de la confection développé dans le cadre du pôle productique régional du Nord-Pas de Calais.

Cet atelier est composé de plusieurs postes de travail interconnectés de manière à prendre en charge tout le processus de fabrication, c'est à dire :

- 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 travaillant en coopération,
- couture et évacuation.



Implantation de la cellule fig. I.6.

Le poste d'assemblage est tributaire du poste de découpe qui produit les pièces selon le résultat de l'inspection et donc, dans un ordre indéterminé.

Pour tenir compte de cet environnement non déterministe, nous travaillons sous l'hypothèse que l'arrivée des pièces est aléatoire. Cette condition peut sembler surprenante, à priori, mais nous l'avons retenue car elle nous permet de travailler dans

un cadre général, et de concevoir un système de pilotage pouvant palier aux différents problèmes survenant sur l'alimentation. De la même manière, nous considérons que les composants à assembler sont quelconques et nous ne les limitons pas à des pièces de tissus.

On notera que l'hypothèse d'arrivée aléatoire a été envisagée dans l'étude d'autres cellules d'assemblage (ALA 83), (LOZ(a) 83), (MAI 85).

Le transport des pièces est réalisé par un même tapis pour les deux robots. Ceci permet, lors d'une panne d'un robot, la poursuite de la fabrication, le robot restant en service ayant accès à l'ensemble des composants. En fait, on généralisera en considérant que le même tapis peut alimenter plusieurs cellules ne réalisant pas toutes le même produit.

Sur ce tapis, les pièces à assembler sont disposées dans une position d'équilibre stable à l'entrée de la cellule. Le système de vision permet alors de déterminer et communiquer le type, le positionnement et l'orientation de la pièce entrée.

VI.2. Stock tampon

La disponibilité des ressources étant aléatoire, nous créons un stock intermédiaire, placé en zone commune et donc accessible à chacun des robots.

Ce stock permet aux robots d'entreposer une pièce disponible sur le tapis mais qui n'est pas utilisable directement sur un des montages, on assure ainsi une continuité du fonctionnement de la cellule. Ce stock est à capacité finie.

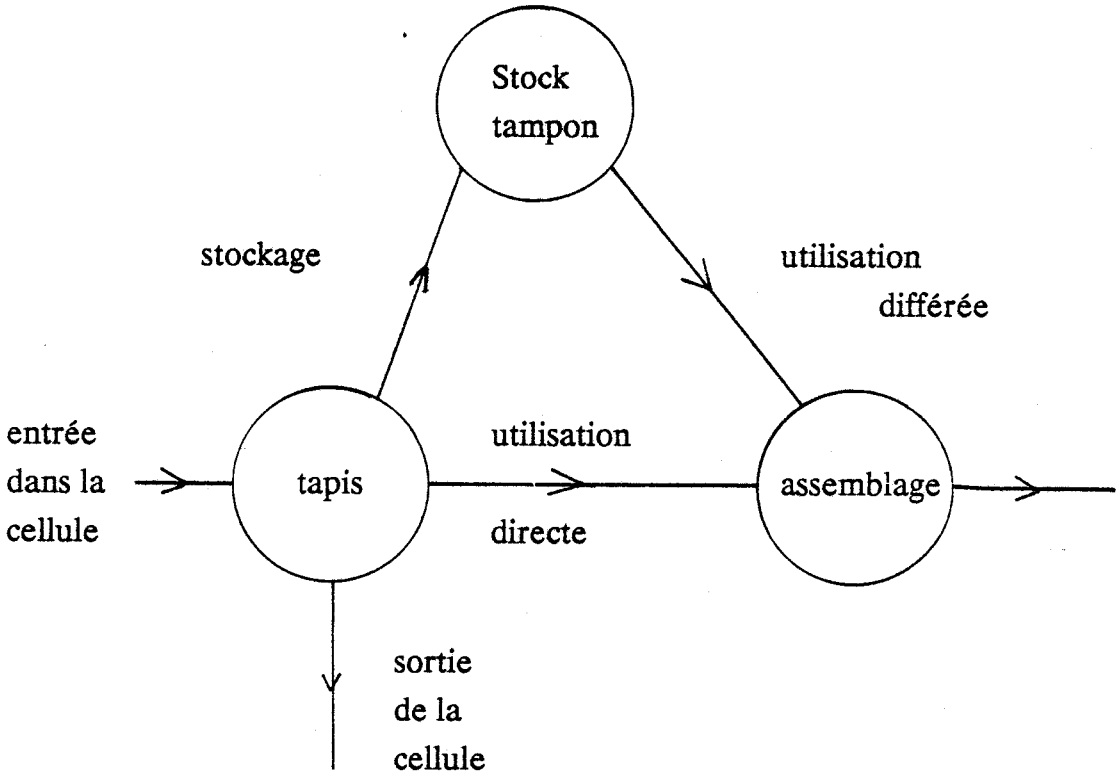
VI.3. Prise des objets

Les objets que les robots manipuleront sont pris à des positions fixes. Le module de décision permet l'affectation de l'objet qui peut être (fig. I.7.) :

- prendre pour un des montages,
- prendre pour stocker,
- laisser passer l'objet.

Cette dernière décision est nécessaire :

- pour les modifications de production,
- pour les objets présentant un défaut,
- lorsque plusieurs pièces de même type se présentent et qu'il n'est pas possible de les stocker,
- pour les objets non reconnus,
- pour les objets destinés à une cellule en aval, ...



Utilisation d'une pièce fig. I.7.

VI.4. Les robots

Les robots sont à aptitudes multiples et variables dans le temps. Certaines d'entre elles leur sont communes, ce qui permet, lors d'une panne ou d'une indisponibilité d'une de ces capacités, de continuer les assemblages.

La réalisation par les robots des opérations de montage nécessite une décomposition en mouvements élémentaires, compte tenu des positions et orientations initiales et finales de la pièce à assembler.

Ces manipulations peuvent nécessiter la participation des deux robots (DAU 84), (ZAP 83).

A chaque opération de la gamme correspond un ensemble d'actions des robots. Connaissant les aptitudes des robots, on peut donc décomposer la gamme d'opérations en gammes d'actions élémentaires directement exécutables par le robot.

VI.5. Décomposition des tâches

1. Cas d'une tâche exécutable par un robot

L'opération ou la tâche est décomposée en une suite d'actions de niveau effecteur de la façon suivante :

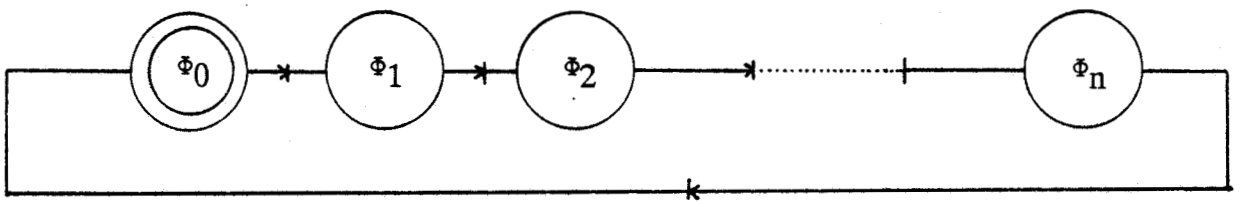


fig. I.8.

A chacune des actions élémentaires Φ_i correspond une procédure paramétrée écrite en LM. Une tâche est une séquence non préemptive d'actions élémentaires. Lorsqu'un des robots perd une de ses capacités, soit l'ensemble de l'opération peut être réalisée par l'autre robot, soit l'opération ne peut plus être exécutée, il est alors nécessaire de réinitialiser le système.

Lorsque les deux robots sont dans ce mode, ils exécutent leur tâche de façon parallèle mais sans synchronisation. Il est alors nécessaire de superviser les trajectoires pour éviter les collisions.

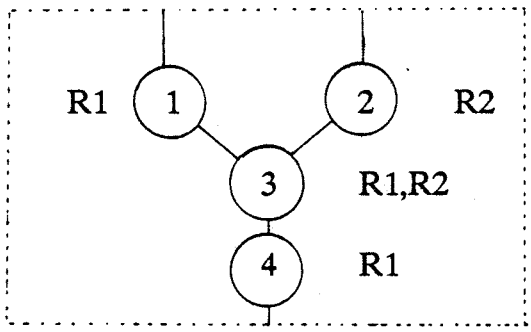
L'évitement d'obstacles dynamiques est un sujet complexe et difficile à mettre en oeuvre et exige des moyens de calcul importants (PAR 84).

L'organisation de la cellule avec les trois plans de travail nous permettra de sélectionner des tâches donnant lieu à des trajectoires sans collision.

2. cas d'une tâche nécessitant une synchronisation des deux robots

Lorsque l'on décompose une opération de montage en une suite d'actions robots, certaines de ces actions nécessitent un seul robot tandis que d'autres imposeront la présence des deux. Dans le cas d'une tâche réalisable par un robot, celui-ci est occupé pendant la durée de cette tâche. Dans le cas d'une tâche non préemptive, pour éviter la réservation des deux manipulateurs durant l'exécution de cette tâche, on définit un planning d'utilisation des ressources.

Exemple : Soit la décomposition suivante d'une opération de montage à deux mains, nous avons :



durée de 2 < durée de 1

On associe à cette tâche le planning suivant :

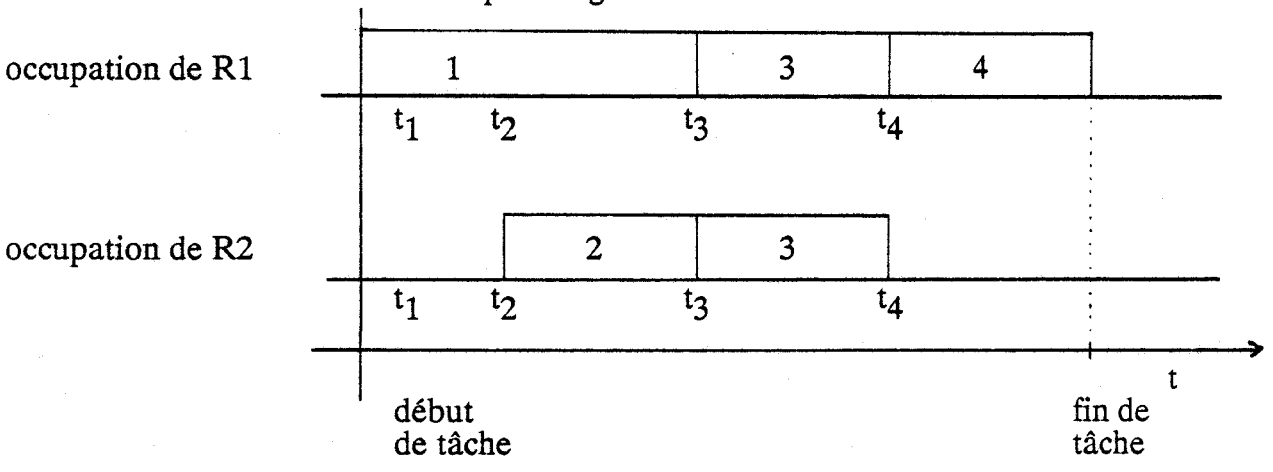


fig. I.9.

Dans ce cas, la réalisation de la tâche nécessite un fonctionnement parallèle des deux robots avec synchronisation. Ainsi, dans l'exemple, pour limiter l'attente du robot 1, la tâche 2 devra être terminée à l'instant t_3, et dans ce cas, le robot 2 sera libéré à t_4.

Deux modes de synchronisation peuvent être envisagés :

a - Une synchronisation à temps discrets où un certain nombre de rendez-vous sont prévus. Une application de ce type d'échanges est réalisée pour la prise de ressources par deux robots. Au point de rencontre, les robots s'arrêtent, échangent leurs signaux de synchronisation puis continuent leurs déplacements jusqu'au prochain point de rendez-vous, comme illustré sur le schéma suivant :

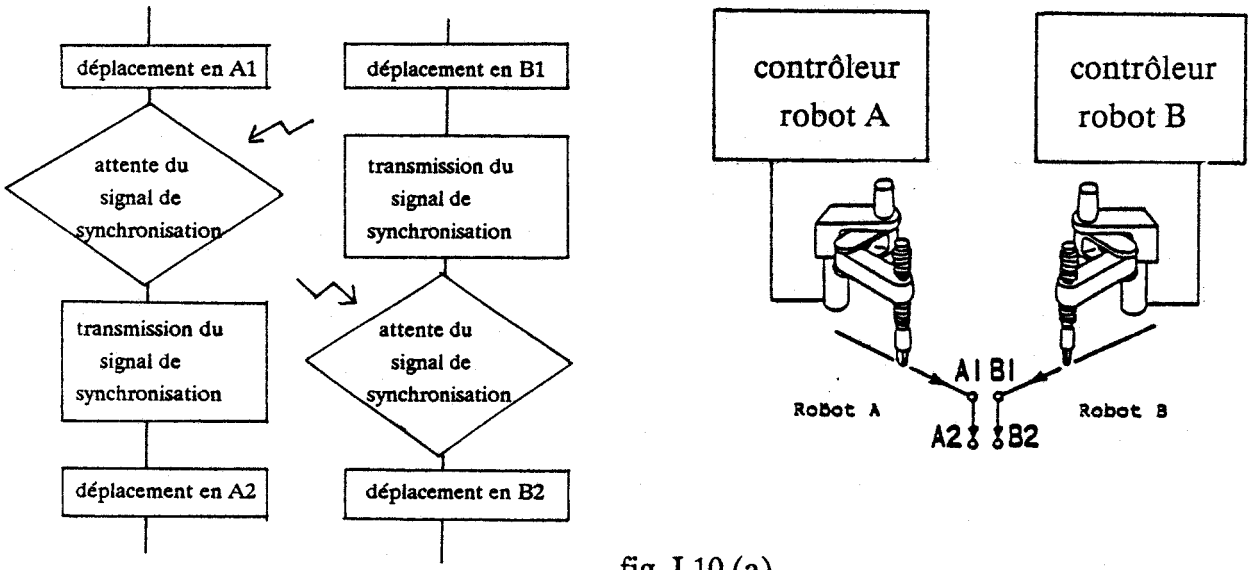


fig. I.10.(a)

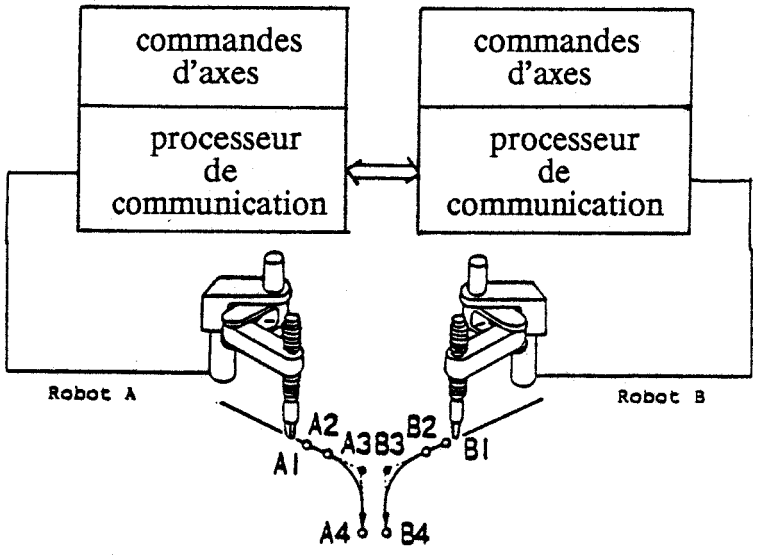


fig. I.10.(b)

Pour ce type de synchronisation, une autre approche a été développée par (KOH 86) dans laquelle d'une part, un processeur de contrôle de communication est associé au contrôleur de robot afin que les échanges des signaux de synchronisation soient effectués en parallèle avec l'exécution des mouvements des robots, d'autre part, pour un même déplacement, deux trajectoires sont calculées. L'une qui intègre le déplacement horizontal puis la descente (A1, A2, A3, A4) et l'autre qui est la trajectoire lissée (A1, A2, A4) ainsi selon l'état de l'autre robot, une sélection du mouvement le mieux adapté est réalisée ; ceci permet de diminuer la durée des déplacements puisqu'au moins un des robots ne s'arrête pas.

b - Une "synchronisation continue" dans laquelle les référentiels des deux robots sont liés en permanence. Le module de supervision doit alors générer les trajectoires de chacun des robots en contrôlant les capteurs qui leur sont associés. Dans (CLA 86), une méthode est présentée où l'un des robots est défini comme maître et l'autre comme esclave.

L'étude de ce module de synchronisation est lié au bi-interpréteur LM dont l'implantation est en cours.

Dans la suite, nous travaillerons sur la gamme d'assemblage, c'est à dire que nous n'entrons pas au niveau des tâches élémentaires ; nous utiliserons indifféremment les termes de tâches ou d'opérations.

VI.6. Les assemblages

L'objectif de production détermine, pour chaque cellule, les types et quantités d'assemblages à fabriquer. Ces différents assemblages seront réalisés sur les trois plans accessibles aux robots. Cependant, nous ne nous intéresserons pas aux problèmes mécaniques d'assemblage, aussi le terme d'assemblage est à considérer dans le sens plus général de montage.

VII CONCLUSION

Le but de notre étude est le pilotage de la cellule que nous venons de présenter. Il s'agit donc d'affecter des tâches aux robots compte tenu de l'état courant de la cellule et selon une fonction objectif donnée.

C'est un problème d'ordonnancement-affectation dynamique, dont la résolution nécessite une modélisation de la cellule. Celle-ci fait l'objet du prochain chapitre.

CHAPITRE 2

MODELISATION DE LA CELLULE

I INTRODUCTION

"La conduite d'atelier se situe à la limite entre ce qu'il est courant d'appeler la partie "gestion" regroupant les décisions stratégiques qui font appel à l'intuition humaine et la partie "commande" qui représente des décisions plus répétitives souvent réalisées par calculateur et directement appliquées au système de production" (KAL 85).

Le système de pilotage que nous développons dans ce mémoire doit permettre l'intégration des fonctions "gestion" et "commande", sa structure est présentée dans la première partie de ce chapitre. Dans un deuxième temps, nous présentons une modélisation de la cellule. Celle-ci basée sur des graphes d'états nous permet de définir le module d'information du système de pilotage.

II SYSTEME DE PILOTAGE

La conduite de production consiste à piloter et coordonner les moyens de production afin de réaliser dans les délais prévus et au moindre coût, les objectifs fixés par l'unité de production.

Le système de pilotage est généralement décomposé en trois niveaux correspondant aux grandes étapes de la planification de production (DOU 79), (CAM 85), (SOE 77).

- Le niveau long terme qui permet d'élaborer le plan directeur de production et les moyens à mettre en oeuvre.

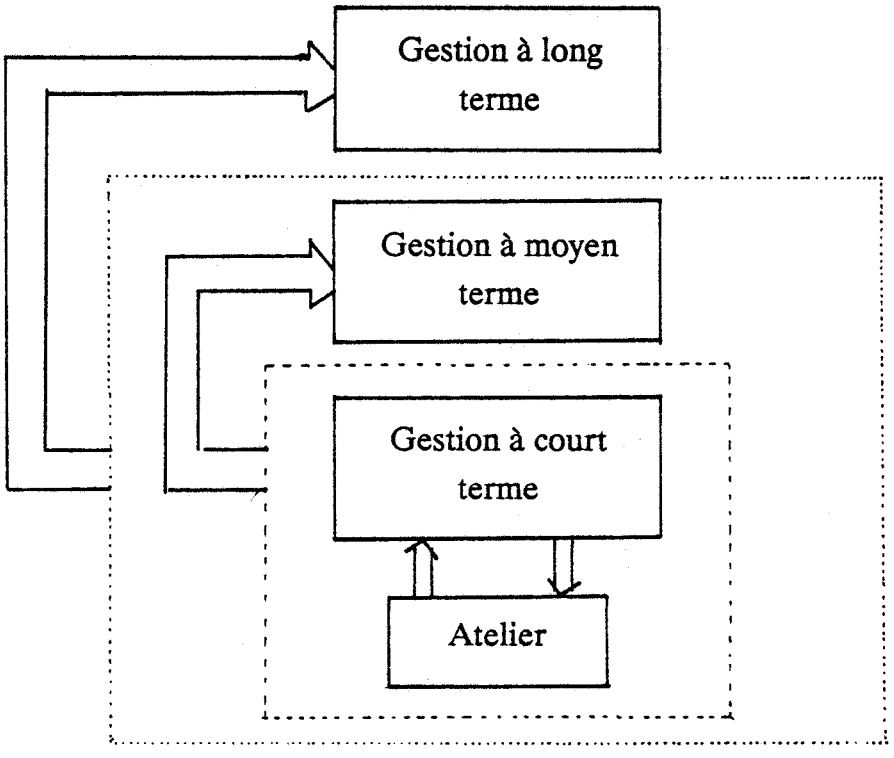
Ce premier niveau permet d'adapter l'outil de production au carnet de commande.

- Le niveau moyen terme qui détermine les approvisionnements et l'ordonnancement des moyens de fabrication. A ce niveau seront définis :

les dates au plus tard de mise à disposition de chaque produit,
le jalonnement des opérations,
les charges.

- Le niveau court terme qui fournit l'ordonnancement à court terme (séquencement des opérations effectives) et le lancement en fabrication.

Entre chaque niveau, une coordination est établie et la structure est couramment la suivante :



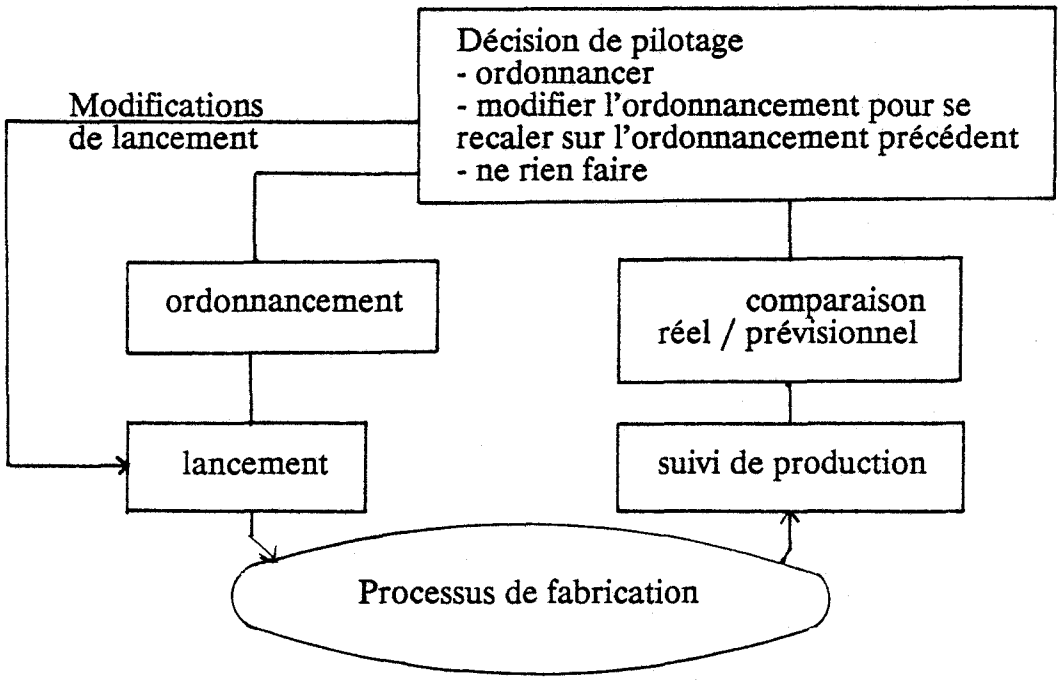
Décomposition de la gestion
d'un système de production. fig. II.1.

Un quatrième niveau peut être introduit (CAM 85) pour définir la commande Temps Réel qui consiste à déclencher la réalisation effective des opérations. On y trouve, l'opération à réaliser sur un poste et le mode opératoire détaillé.

Dans notre étude, nous nous sommes placés au niveau gestion à court terme. On peut remarquer que si la fonction ordonnancement-lancement est primordiale dans un système de conduite de production, c'est aussi celle qui est le plus rarement implantée dans les systèmes informatisés.

Pour être efficace, cette fonction devrait être considérée comme un processus

dynamique bouclé que l'on peut schématiser ainsi (DOU 83) :



Boucle de pilotage. fig. II.2.

En fait, cette fonction est celle qui pose le plus de problèmes au niveau de son implantation dans une entreprise. Ceux-ci apparaissent, soit au niveau de l'ordonnancement par une mauvaise adaptation de celui-ci, soit au niveau du lancement des tâches en décalage par rapport à l'état réel de l'atelier, soit au niveau du suivi de production par transmission incorrecte des informations.

Pour résoudre ces difficultés, les recherches s'orientent vers des méthodes d'analyse et de conception des systèmes de pilotage (ARA 84).

Les tendances actuelles délaissent les systèmes centralisés pour des ordonnancements hiérarchisés et répartis.

Avant de détailler l'approche que nous avons retenue, nous allons présenter quelques travaux réalisés dans ce domaine.

III EXEMPLES DE SYSTEMES DE PILOTAGE

III.1. Modèle développé au LAAS de Toulouse

Pour la gestion de production d'unités fabricant des objets diversifiés en petites et moyennes séries, les études du Laboratoire de systèmes automatisés de production ont pour objectif d'assurer le pilotage en temps réel des flux de produits.

Leur approche basée sur une des techniques d'aide à la décision (COU 79), (ERS 85), (FON 80), (THO 80) consiste à décomposer chaque niveau de décision en deux fonctions selon la figure II.3.

La fonction "aide à la décision" analyse, compte tenu des contraintes du niveau supérieur, les caractéristiques de l'ensemble des solutions admissibles en vue de définir à chaque instant un ensemble d'actions possibles.

La fonction "prise de décision" choisit parmi les actions sélectionnées par la fonction précédente, une décision à partir des considérations propres au niveau.

En pratique, la première fonction qui assure une cohérence entre les différents niveaux peut être totalement automatisée, tandis que la deuxième, compte tenu du problème de choix ne peut pas être intégralement automatisée et doit être soumise en partie à un décideur.

Dans ce sens, les travaux développés pour ce système d'aide à la décision permettent de définir des ordonnancements admissibles. Ces derniers sont composés d'un ensemble de tâches permutables qui vérifient l'objectif de respect des dates limites d'exécution. Ainsi, l'organisation finale est laissée au niveau de l'aide à la décision.

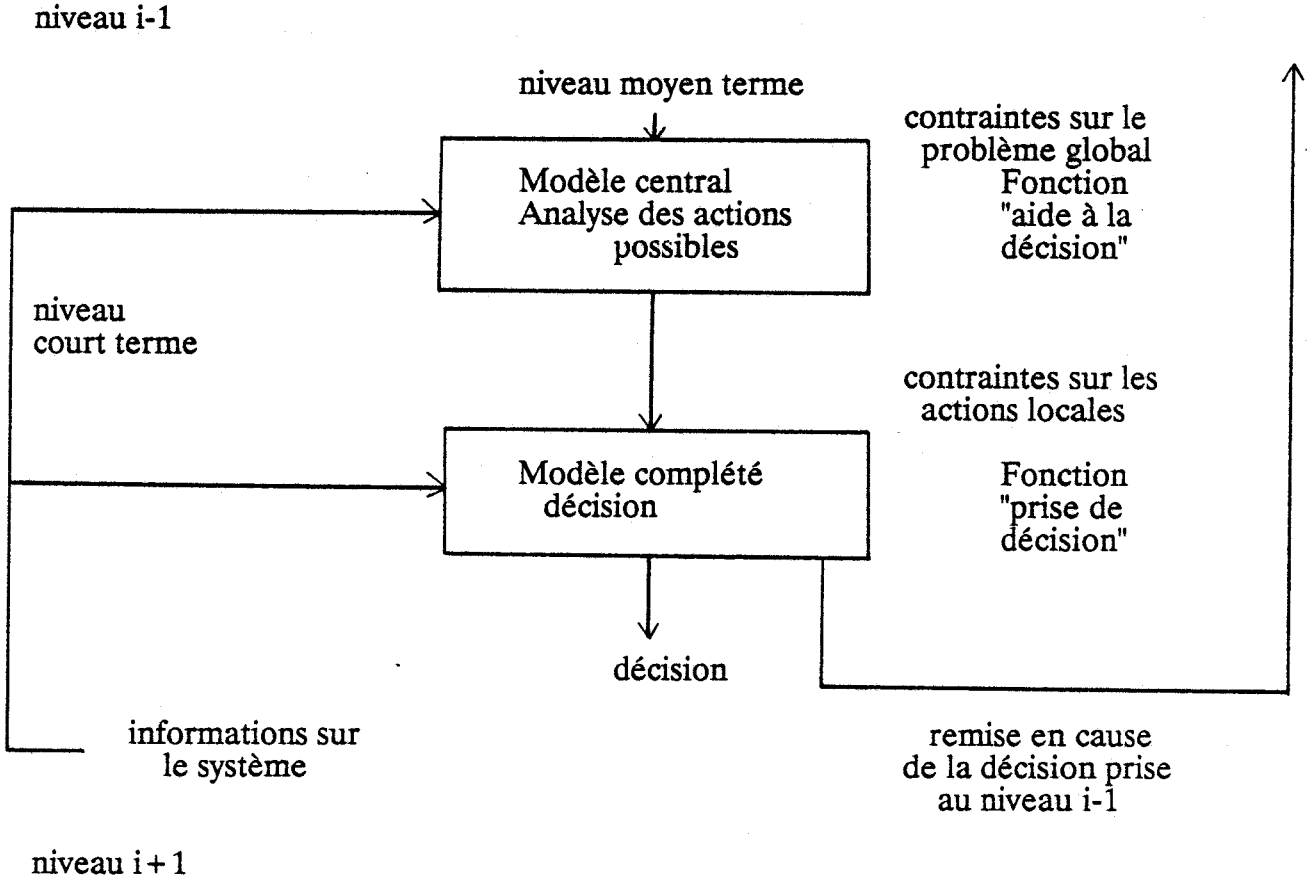
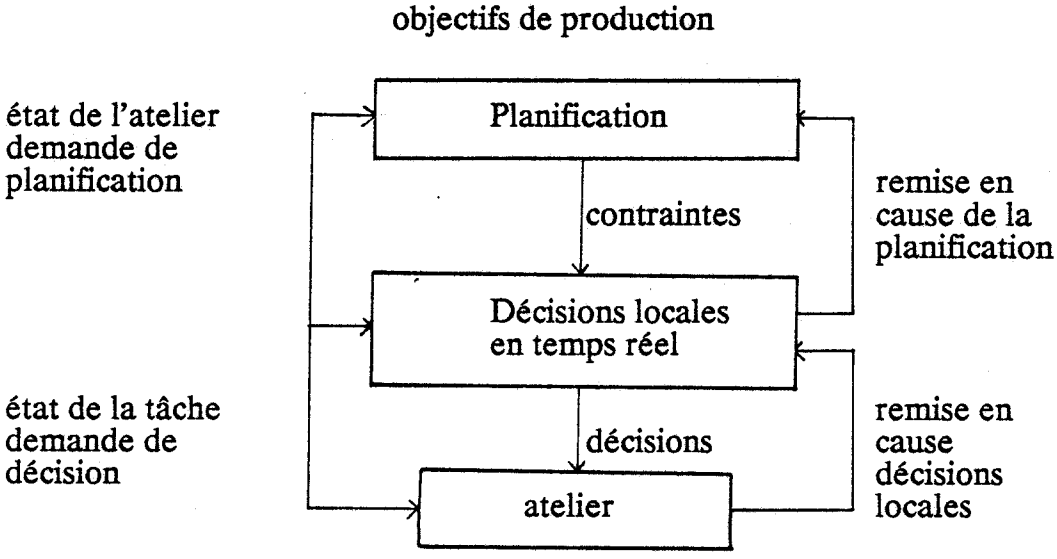


fig. II.3.

III.2. Logiciel de simulation SAGA

Dans l'équipe de Méthodes et Outils pour la Production Automatisée du Laboratoire de Toulouse, un logiciel de simulation et d'aide à la gestion d'ateliers SAGA (MON 85) a été développé. L'idée est de laisser aux modules inférieurs une certaine autonomie. Même si leur vision du problème est restreinte, ceci permet d'éviter les remises en cause des décisions du niveau supérieur.

La structure retenue comporte deux niveaux ; un niveau supérieur qui correspond à la fonction planification, un niveau inférieur qui assure la fonction décision pilotant directement l'atelier.



Structure hiérarchisée Planification Décision. fig. II.4.

III.3. Système CODECO du laboratoire de Grenoble

L'équipe Conduite Décentralisée Coordonnée d'ateliers du Laboratoire d'Automatique de Grenoble propose une structure à deux niveaux de décision s'insérant dans une structure hiérarchisée globale du système de production.

La décomposition est réalisée suivant deux axes. Un horizontal qui correspond à la division du système physique en centres de charge. Un vertical associé à une répartition du système de décision en une structure hiérarchisée avec, comme critère de décomposition, l'horizon de prise de décision, associé à une période de temps au bout de laquelle les décisions sont remises en cause. Chaque niveau reçoit les décisions du niveau supérieur et définit les cadres de décision pour le niveau inférieur. Les centres de décision sont alors reliés par deux types de relation, les relations inter-niveaux ou coordination, les relations intra-niveau ou coopération (KAL 85).

La structure interne d'un centre de décision est alors la suivante :

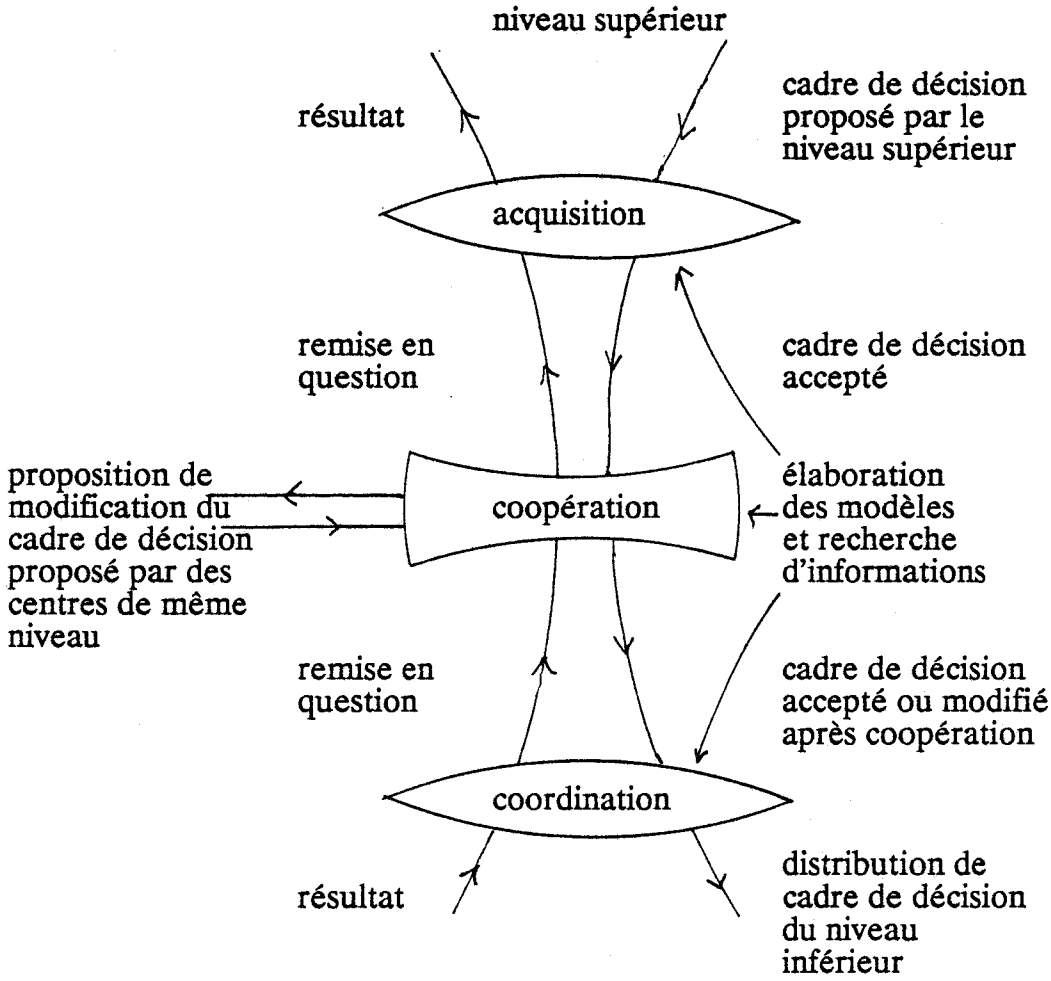


fig. II.5.

Au niveau court terme, pour surmonter les problèmes "d'écart" entre les fonctions d'ordonnancement et de lancement, la conduite d'atelier est décomposée en deux axes associés respectivement :

- à la coordination de l'atelier,

- aux conduites locales des machines

Les méthodes citées précédemment sur la recherche de tâches permutables sont utilisées.

En cas de perturbations, la coopération entre les centres de lancement voisins

tente de résoudre les problèmes et si ce n'est pas possible, un appel au coordinateur est fait pour que la décision soit reconsidérée.

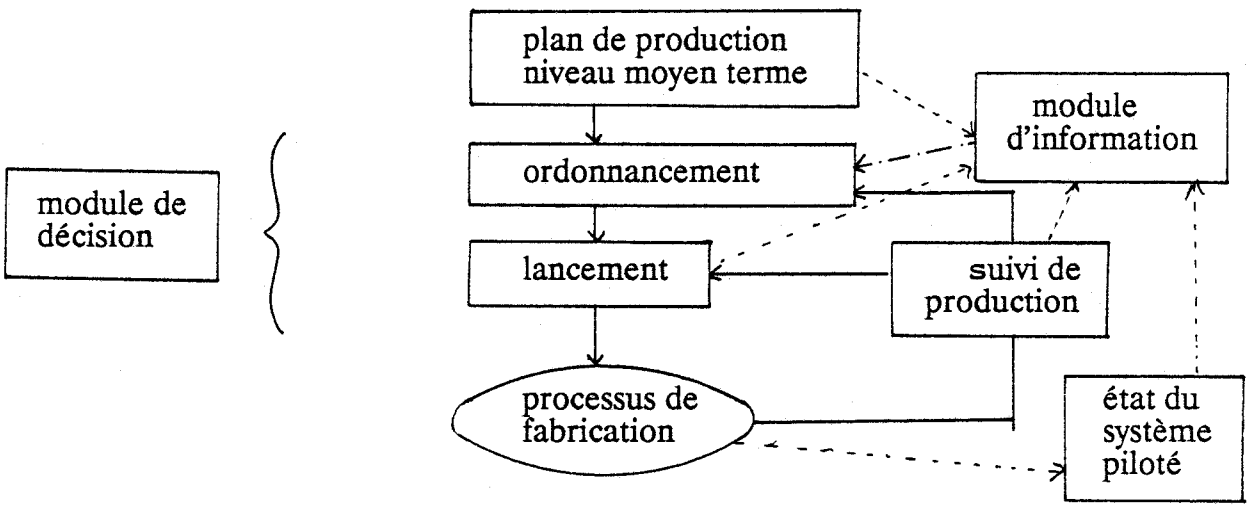
III.4. Solution retenue

Le modèle de décision que nous avons élaboré s'intègre également dans une structure hiérarchisée. Compte tenu du fait que notre étude ne porte que sur un élément de l'atelier flexible de fabrication, nous nous sommes situés au niveau gestion à court terme et plus exactement à celui de la cellule étudiée.

Dans ce sens, la structure inspirée des travaux précédents en diffère par le fait qu'elle ne s'applique pas à un atelier mais à une cellule d'assemblage. L'objectif est de laisser au système de pilotage, une autonomie quant à l'ordonnancement des tâches, au lieu de ne lui laisser qu'un rôle d'exécutant passif.

Par conséquent, le niveau moyen terme définit pour chacune des cellules, sur un horizon donné, la quantité de produits à fabriquer selon les types. Le système de décision associé à chaque cellule doit alors ordonnancer et affecter, les différentes opérations aux différents moyens en tenant compte en permanence de l'état réel de la cellule (avancement de la production, état des ressources, des stocks, ..).

La structure est alors la suivante :



Système de pilotage fig. II.6.

On réalise ainsi un ordonnancement et une affectation dynamiques des tâches aux robots. Pour cela, le système de pilotage est composé d'un module d'information qui permet de connaître à tout instant l'état des différents composants de la cellule, et d'un module de décision qui, compte tenu des données en provenance du module d'information, est chargé d'effectuer l'ordonnancement.

Nous allons présenter le module d'information dont la construction s'appuie sur une modélisation par graphe d'état de la cellule. Le module de décision fera l'objet du prochain chapitre.

IV. MODULE D'INFORMATION

IV.1. Introduction

Le problème de pilotage peut être posé de la façon suivante :

Sur un horizon donné (un jour, un poste), la cellule de production doit fabriquer un certain nombre de produits différents. Le système de conduite permet alors l'affectation et l'ordonnancement des différentes opérations en fonction des moyens de l'atelier. Il doit être capable de réagir aux différents aléas et pour cela, il doit connaître à tout instant l'état du système.

Compte tenu de la complexité du système, la première étape de notre travail a été de répertorier les différents événements qui pouvaient se produire et analyser les changements d'états correspondants.

Il existe différents outils de représentation graphique (BEL 85) permettant de décrire l'évolution des systèmes de production, on peut citer le grafcet utilisé dans (DAL 82), les réseaux de Pétri (OH 82). Dans notre étude, nous avons effectué une modélisation par des graphes d'états et de transitions qui permettent d'évaluer en permanence l'état des divers constituants de la cellule (DJE 85), (STA(c) 85).

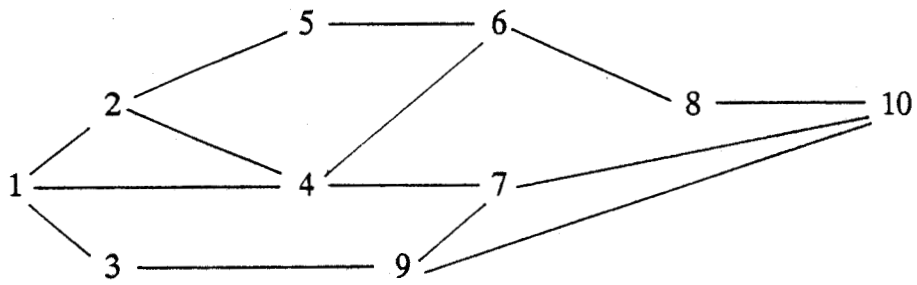
Parmi les différents états, certains se rapprochent de ceux utilisés pour les tâches informatiques aussi nous avons choisi comme référence le rapport SCEPTRE (SCE 84). Le même vocable a été employé pour les états semblables et d'autres termes ont été définis pour les états spécifiques que nous avons introduits.

Au niveau des événements, on peut distinguer les deux types suivants : ceux dont la date d'occurrence est prévue (fin de tâche) et ceux dont l'instant d'occurrence est inconnu avant l'apparition (panne, arrivée de pièces).

IV.2. Gammes d'assemblage

La réalisation d'un assemblage nécessite un certain nombre d'opérations élémentaires liées par des relations de précédence. Dans beaucoup de cas, l'assemblage n'est pas une succession linéaire de tâches, certaines d'entre elles pouvant être réalisées en parallèle.

La modélisation que nous présentons dans ce chapitre, est basée sur une représentation de la gamme d'assemblage, sous forme de graphe tel que :



Gamme d'assemblage à 10 opérations fig. II.7.

L'ensemble des tâches est représenté par un graphe orienté acyclique G (T,A) où T est l'ensemble des noeuds et A celui des arcs (BER 63).

Ce graphe fait apparaître :

Toutes les opérations à réaliser, correspondant aux noeuds de T.

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

L'obtention d'une gamme d'assemblage est souvent manuelle au sein d'un bureau des méthodes. La génération automatique dans ce domaine est limitée à l'étude de variantes d'un même groupe de produits (CAM 87).

Une autre approche est proposée par l'équipe Assemblage de l'unité "Microsystèmes et Robotique" de Besançon (BOU 84). Elle repose sur la recherche et

l'exploitation de toutes les gammes d'assemblage satisfaisant un ensemble de contraintes. La méthode est basée sur les notions suivantes :

- Tout produit fini est un ensemble *C* de M composants élémentaires $\{c_1, c_2, ..., c_i, ...c_M\}$ tels que deux composants quelconques c_i et c_j sont liés par une ou plusieurs liaisons mécaniques.

- Notion de liaisons fonctionnelles

Définition : Il existe une liaison fonctionnelle et une seule entre c_i et c_j s'il existe au moins une liaison mécanique entre ces deux constituants.

Une liaison fonctionnelle peut donc recouvrir plusieurs liaisons mécaniques. L'établissement d'une telle liaison s'effectue par une action fonctionnelle.

Une séquence opératoire d'assemblage (ou gamme) est alors caractérisée par une suite de n actions fonctionnelles réalisées en série et/ou parallèle.

A partir des plans du produit et des contraintes liées à la mise en oeuvre des actions d'assemblage, le graphe des liaisons fonctionnelles est établi. La méthode proposée permet de trouver les contraintes de réalisabilité exprimées sous forme de relations n-aires R et S.

Celles-ci sont des généralisations des deux relations binaires R et S définies par :

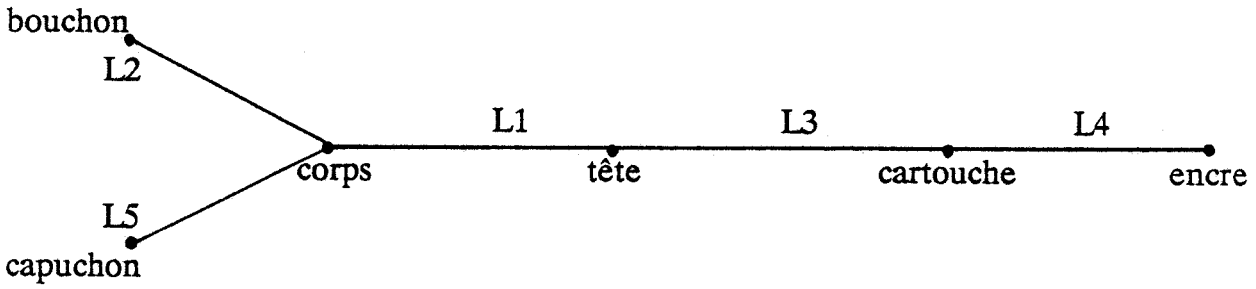
R : "Il est impossible de réaliser L_i lorsque L_j est établie."

S : "Il est impossible de réaliser L_i lorsque L_k n'est pas établie".

L'exploitation, à l'aide de fonctions logiques, par génération de plans ou par traduction en réseaux de Pétri, des contraintes de réalisation de chaque action permet d'établir toutes les séquences opératoires d'assemblage (BOU 86), (BOU(a) 87), (BOU(b) 87).

Les gammes d'assemblage obtenues ont une structure arborescente exprimant à chaque rang les choix d'actions fonctionnelles qui se présentent.

A titre d'exemple, la figure II.8. représente le graphe des liaisons fonctionnelles d'un crayon à bille et la figure II.9. les gammes d'assemblage (exemple extrait de (BOU 84)).



Crayon à bille : graphe des liaisons fonctionnelles. fig II.8.

1	5	3	4	2 (1)
3	5	4	2	(2)
	4	5	2	(3)
		2	5	(4)
2	3	4	1	5 (5)
3	2	4	1	5 (6)
4	2	1	5	(7)
	1	2	5	(8)
		5	2	(9)
1	5	4	2	(10)
	4	2	5	(11)
		5	2	(12)

Gammes d'assemblage du crayon à bille. fig. II.9.

La connaissance de ces diverses possibilités pour le montage d'un produit permet une certaine flexibilité dans la réalisation automatique de celui-ci, dans la mesure où selon les pannes, les problèmes d'alimentation, ..., il sera possible d'assembler les constituants dans des ordres différents.

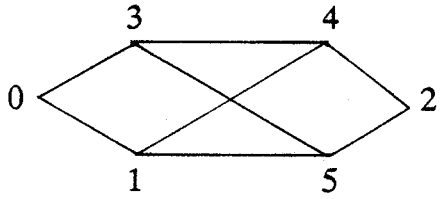
Sur cette structure, il existe des actions permutables qui sont définies par :"Deux actions AF_i et AF_j permutent aux rangs α et $(\alpha+1)$ dans deux séquences, si ces séquences sont identiques, sauf aux rangs α et $(\alpha+1)$ pour lesquels l'ordre d'exécution de AF_i et AF_j est inversé".

Les actions permutables sont groupées dans un rectangle et reliées symboliquement par une double flèche, ce qui permet d'une part de diminuer le nombre de branches, d'autre part par récurrence d'obtenir des séquences comportant un maximum de permutations.

Ainsi, dans l'exemple cité, nous avons :

permutations	séquences correspondantes
1 <----> 3 ---- 5 <----> 4 ---- 2	2, 3, 10, 12
4 ---- 5 <----> 2	3, 4, 11, 12
2 <----> 3 ---- 4 ---- 1 ---- 5	5, 6
1 ---- 3 <----> 5 ---- 4 ---- 2	1, 2
3 ---- 1 <----> 4 ---- 5 <----> 2	8, 9, 11, 12
3 ---- 2 <----> 4 ---- 1 ---- 5	6, 7
3 ---- 4 ---- 1 <----> 2 ---- 5	7, 8

A chacune de ces branches, nous pouvons associer un graphe, par exemple :



3 --- 1 --- 4 --- 5 --- 2

3 --- 1 --- 5 --- 4 --- 2

1 --- 3 --- 4 --- 5 --- 2

1 --- 3 --- 5 --- 4 --- 2

Graphe d'assemblage du crayon à bille fig. II.10.

gammes correspondantes

Nous pouvons générer un ensemble de séquences à partir d'un graphe, il y a donc une relation entre un graphe et un ensemble de séquences.

En fait, la différence essentielle est que, dans une représentation sous forme d'arbre une opération intervient plusieurs fois par contre dans un graphe une opération n'apparaît qu'une seule fois. Ceci provient du fait que l'arbre représente l'ensemble des séquences possibles et le graphe correspond à un sous ensemble.

Dans cette étude, nous proposons une procédure appliquée à un graphe. Nous pourrons l'utiliser pour une structure arborescente en appliquant le calcul à chacun des graphes, une autre solution pour limiter les calculs, consiste à sélectionner parmi l'ensemble des graphes, celui qui représente le maximum de séquences possibles.

IV.3. Modélisation

La cellule de production flexible est décrite au moyen des graphes d'état associés au produit à réaliser, aux robots, à l'environnement. Elle est modélisée par le quadruplet :

$$< X_t , R_t , E_t , U_t >$$

X_t représente l'état du produit en cours
R_t représente l'état des robots
E_t représente l'état de l'environnement
U_t représente l'état des commandes admissibles.
t instant d'observation

IV.3.1. Etat du produit en cours

Il est défini par l'état de l'assemblage et celui de chacune des opérations.

IV.3.1.1. Etat du produit

L'assemblage est composé de n opérations. On note *T* l'ensemble des n tâches (ou opérations). L'évolution de l'assemblage peut être représentée par le graphe

suivant :

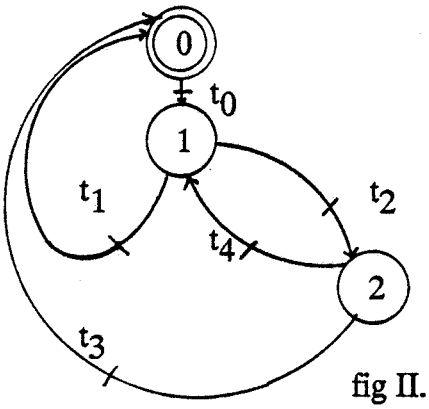


fig II.11.

Etats

0 bloqué

1 exécution en cours

2 suspendu

Transitions :

t_0 : La décision de commencer cet assemblage a été prise.

t_1 : Quelle que soit la tâche i appartenant à *T*, i est terminée. L'assemblage est fini.

t_2 : Il existe une opération i appartenant à *T* telle que i soit dans l'état non réalisable.

t_3 : Une réinitialisation de l'assemblage est nécessaire suite à la non-réalisation d'une opération. t_3 correspond à cette réinitialisation qui nécessite l'intervention d'un opérateur.

t_4 : Après intervention d'un opérateur, l'assemblage peut être poursuivi.

La description des états d'un assemblage est complétée par un graphe d'état associé à chacune des opérations qui le composent.

IV.3.1.2. Etat d'une opération

1. Caractéristiques d'une tâche

A chaque assemblage est associé un graphe acyclique dont les sommets représentent les opérations et les arcs les contraintes entre les tâches. Le graphe possède un sommet final f et on lui adjoint un sommet initial s.

Soient i, j deux opérations du graphe, nous dirons que la tâche i précède la tâche j (i < j) s'il existe un arc d'extrémité initiale i et d'extrémité finale j.

L'ensemble des antécédents de la tâche i est noté $A(i)$ et est défini par :

$A(s) = \phi$

$i \neq s \quad A(i) = \{ j / j < i \}$

La tâche i peut être réalisée si et seulement si toutes les tâches de $A(i)$ sont terminées.

On note $Af(i,t)$ l'ensemble des tâches antécédentes de i qui sont finies à l'instant t.

L'ensemble des successeurs de la tâche i est défini par :

$S(t) = \phi$

$i \neq f \quad S(i) = \{ j / i < j \}$

Aucune tâche de l'ensemble $S(i)$ ne peut être commencée tant que la tâche i n'est pas terminée.

On définit ainsi les contraintes de précédence entre les tâches.

La réalisation d'une tâche i demande la disponibilité, à différents instants, d'un certain nombre des ressources. Soit $R(i)$ l'ensemble de ces ressources, il est défini par :

$R(i) =$ {(ressource, date d'appropriation, date de libération)}

Les dates d'appropriation et de libération sont définies par rapport à la date de début d'exécution de la tâche.

Sur cet ensemble, on distingue :

- les ressources de type consommable : ce sont celles qui ne sont utilisables qu'une seule fois (pièce, matière première). La décomposition des opérations élémentaires d'assemblage est telle qu'une tâche nécessite au plus une ressource. En fait, une opération consiste généralement à assembler une pièce au montage en cours. Une tâche qui nécessite aucune ressource consommable est par exemple, celle qui consiste à évacuer le montage lorsqu'il est terminé. On note *RC* l'ensemble des ressources consommables.

On définit l'application suivante :

Γ : RC -----> $P(T)$ $P(T)$ ensemble des parties de T
k -----> $\Gamma(k)$

$\Gamma(k)$ est l'ensemble des opérations utilisant la ressource consommable de type k.

Γ^{-1} : $P(T)$ -----> RC

{i, j, ..., n} -----> les ressources consommables par les tâches de cette partie de T.

Par abus de langage et pour simplifier l'écriture, nous définissons :

Γ^{-1} : T -----> RC
i -----> $\Gamma^{-1}(i)$

$\Gamma^{-1}(i)$ est la ressource utilisée par i.

Cas 1 : A chaque tâche correspond un type de ressources, Γ^{-1} est une application bijective.

Cas 2 : Plusieurs tâches peuvent utiliser le même type de ressources, Γ^{-1} est une application surjective.

Par la suite, nous utiliserons la notation Cas I (I = 1, 2) lorsqu'il y aura lieu de faire la différence.

- les ressources de type récupérable : ce sont celles qui sont réutilisables dés que la tâche qui les occupait, les a libérées (robots, zone commune, outils). Une opération peut nécessiter la synchronisation des actions élémentaires des deux robots. On note $RR(i)$ l'ensemble des ressources récupérables utilisées par la tâche i.

Nous avons : $R(i) = \{\Gamma^{-1}(i)\} \cup RR(i)$.

Une opération est non préemptive.

Pour chaque opération, nous disposons des paramètres de la tâche (exemple : temps probables d'exécution, spécifications du type d'outil, ...) et d'un planning d'utilisation des ressources.

Nous notons :

t_i : la date de début d'exécution de la tâche i,
p_i : la durée de la tâche i,
c_i : la date de fin de la tâche i avec $c_i = t_i + p_i$

La date de début de l'exécution de l'opération i vérifie l'équation (BEL 85) :

$$t_i \geq \max \left(\max_{j \in A(i)} (t_j + p_j) , \max_{r \in R(i)} T_r \right)$$

avec T_r la date de disponibilité de la ressource r.

La date de début d'une tâche dépend donc

- de la date d'arrivée de la ressource consommable,
- de la date de libération des ressources partageables,
- de la date de fin d'exécution des antécédents.

2. Etat d'une opération

Chacun des robots peut exécuter deux types d'opérations. Celles qui ont pour conséquence de faire évoluer l'assemblage, que l'on appellera opération de montage et celles qui permettent une régularisation de la production que l'on notera opération de stockage.

Nous allons présenter les différents états des opérations puis un graphe décrivant les conditions de changements d'états.

Une opération sera dans l'un des états suivants :

bloqué : Lorsqu'un assemblage est bloqué, toutes ses opérations sont dans l'état bloqué.

non exécutable : L'exécution de l'assemblage est en cours, une opération est dans cet état tant que tous ses antécédents ne sont pas réalisés ou que la ressource consommable nécessaire à son exécution n'est pas dans la cellule.

pré-exécutable : (a) Les antécédents de l'opération sont réalisés.

pré-exécutable : (r) La ressource consommable que l'opération utilise est présente dans la cellule.

exécutable : Les deux conditions précédentes sont réalisées. La ressource consommable est présente et les antécédents sont terminés.

en attente : Une tâche est dans cet état lorsque la décision de réaliser l'opération est prise, mais que les ressources partageables ne sont pas toutes disponibles.

en cours : La tâche est en cours d'exécution, les ressources partageables nécessaires sont prévues présentes selon le planning d'utilisation.

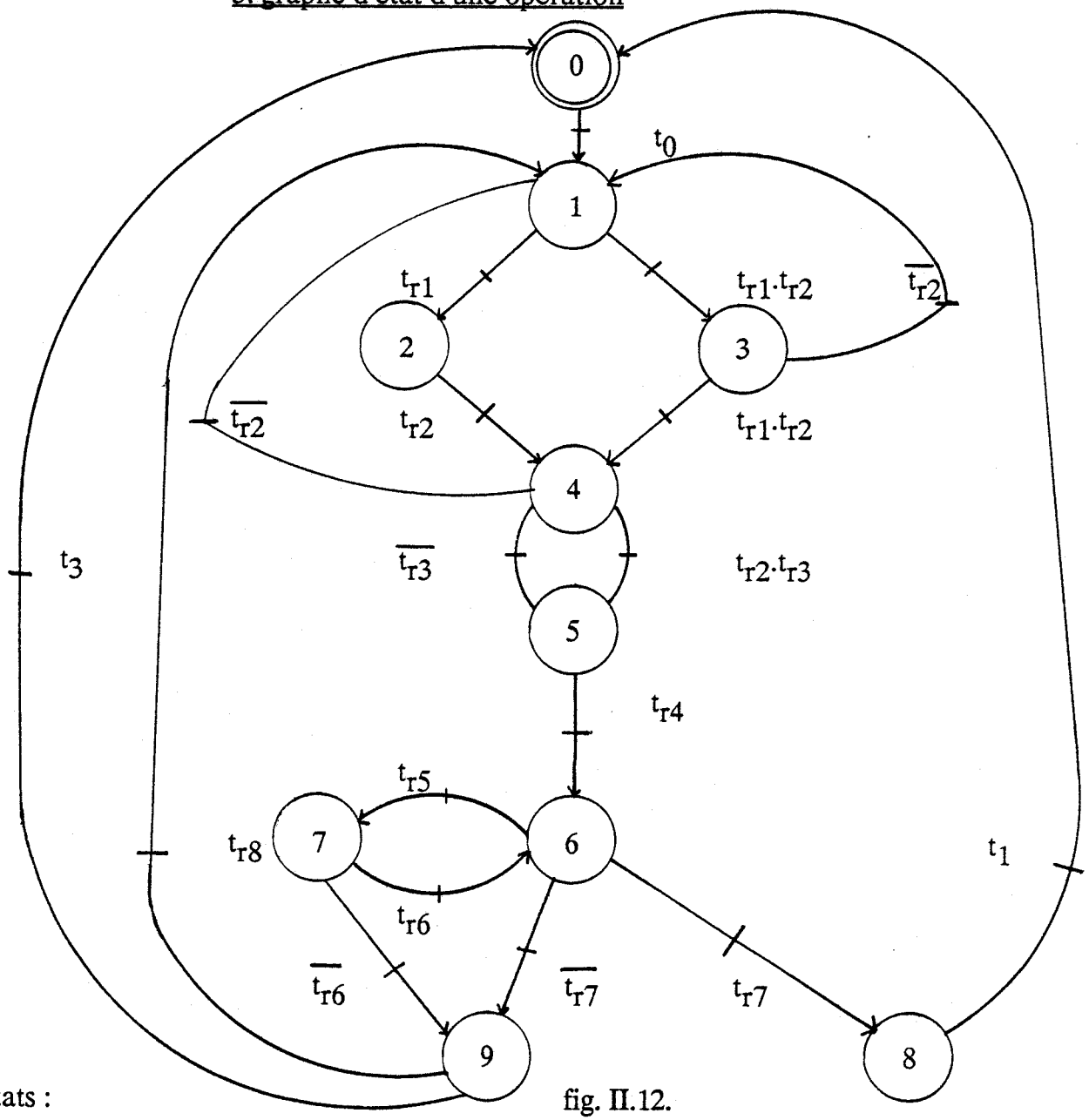
suspendu : La réalisation de l'opération est suspendue car les conditions de continuation ne sont pas remplies (exemple : absence d'une ressource partageable).

terminé : L'opération est correctement réalisée.

non réalisable : L'opération ne peut plus être poursuivie. Le système doit être réinitialisé globalement.

L'évolution d'une opération entre ces différents états est décrite à l'aide du graphe d'état suivant :

3. graphe d'état d'une opération



Etats :

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

fig. II.12.

Transitions :

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} La ressource consommable de l'opération est présente dans la cellule.

$\overline{t_{r2}}$ Si plusieurs tâches consomment la même ressource et que celle-ci est attribuée à une autre tâche, l'opération passe de l'état pré-exécutable (r) à l'état non exécutable.

(*Remarque* : une opération qui est dans l'état pré-exécutable (a) ne peut devenir non exécutable).

t_{r3} La décision d'exécuter l'opération est prise. La décision peut être remise en cause ($\overline{t_{r3}}$) si l'évolution de l'état de la cellule diffère de celle qui est prévue avant la réservation de toutes les ressources.

t_{r4} Les ressources partageables nécessaires à l'exécution de l'opération sont réservées.

t_{r5} Certaines conditions de continuation de l'opération ne sont pas remplies (ex. : indisponibilité d'une ressource consommable consécutive à une panne, bris d'outil, ..). Suivant l'évolution de la cellule, l'opération pourra se terminer normalement ou deviendra non exécutable.

t_{r6} Les conditions de continuation de l'opération sont vérifiées et l'exécution peut être poursuivie.

t_{r7} La tâche est correctement terminée.

t_{r8} Après une exécution incorrecte, la tâche peut être ré-exécutée.

t_1 Réinitialisation lorsque l'assemblage est fini.

t_3 Réinitialisation après une mauvaise exécution. Dans ce cas, toutes les opérations du même assemblage, quel que soit leur état, reviennent dans l'état bloqué.

différent de [illegible]

4. cas d'une opération de stockage

En fonction de la place en stock, et des assemblages que l'on réalise, on définit un certain nombre de tâches de stockage.

Le graphe de ces opérations est similaire à celui des opérations de montage. La seule différence est qu'une opération de stockage est dans l'état pré-exécutable (a) lorsqu'il existe une place en stock pour ce type de ressource; la transition t_{r1} devenant : "il existe au moins une place en stock pour ce type de ressource".

IV.3.2. Etat des robots

On peut de la même façon définir un graphe d'état associé à chacun des robots.

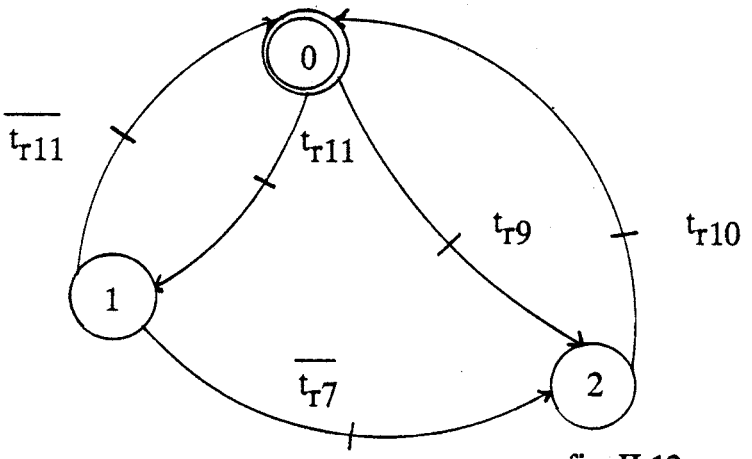


fig. II.13.

états :

0 libre

1 occupé

2 indisponible

Définition des états :

libre : Le robot est disponible.

Le robot est dans cet état lorsqu'il vient de terminer une opération, ou après une fin de maintenance et il n'a pas été réservé pour une opération.

occupé : Le robot est réservé pour l'exécution d'une opération.

indisponible :Le robot n'est pas disponible car, par exemple, il est à la disposition du service d'entretien ou sous diagnostic suite à une exécution incorrecte.

Signification des transitions :

t_{r9} décision de mise en maintenance.

t_{r10} fin de maintenance ou diagnostic avec mise à jour éventuelle des capacités du robot.

t_{r11} Le robot appartient à la liste des ressources nécessaires à l'une des tâches sélectionnée et est réservé.

$\overline{t_{r11}}$ Le robot n'est plus utilisé pour la tâche en cours.

Les autres transitions ont la même signification que celles du graphe d'état des opérations de montage.

A chacun des robots sont associés d'une part un identificateur, d'autre part, les paramètres suivants :

- la liste des outils qui sont à sa disposition,
- l'outil qui lui est associé à l'instant t.

Le système de décision utilise ces informations pour connaître, à chaque instant, les capacités du robot, c'est-à-dire les actions élémentaires qu'il peut exécuter (vissage, collage, assemblage, suivi de trajectoire, ...). Nous les noterons $CR_i(t)$.

$CR_i(t)$ est modifié si après une panne, il existe des actions élémentaires que le robot i n'est plus apte à réaliser.

La relation $CR_1(t_0) \cap CR_2(t_0) \neq \phi$ est vérifiée à l'instant $t_0 = 0$.

Le fait qu'une partie des tâches puissent être effectuées par l'un ou l'autre des robots.permet, lors d'une panne sur un des robots, que le système de production ne soit pas obligatoirement interrompu.

IV.3.3. Etat de l'environnement

L'environnement correspond aux ressources consommables des opérations de la gamme qui sont présentes dans la cellule, c'est à dire soit les pièces qui sont sur le tapis roulant que nous appellerons stock dynamique, soit celles qui sont sur le stock situé en zone commune que nous appellerons stock statique.

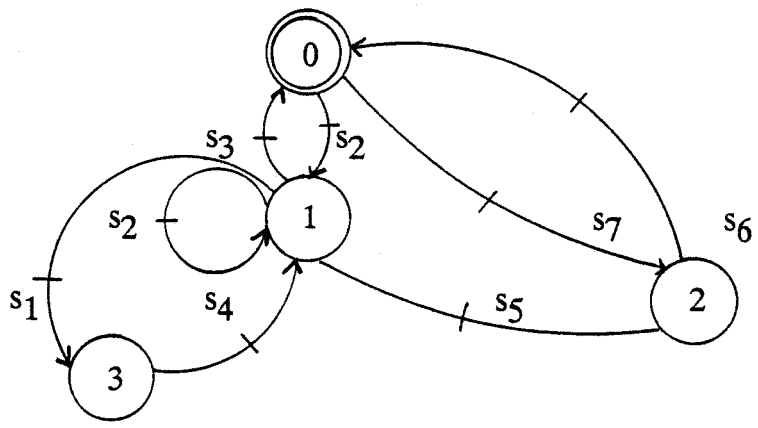
Chacun des stocks évolue selon les entrées, les utilisations directes ou différées et les sorties de ressources. Nous rappelons que certaines pièces peuvent sortir de la cellule, soit parce qu'elles présentent un défaut, soit pour une utilisation par une autre cellule, etc .

On note SS(t) (respectivement SD(t)), l'ensemble des ressources présentes sur le stock statique (respectivement sur le stock dynamique) à l'instant t.

A chacune des ressources est associé un ensemble de paramètres caractéristiques relatifs à :

- son identification
- son orientation
- sa position courante.

En fait, les stocks étant, tout comme les robots, des ressources récupérables, nous pouvons leur associer un graphe d'état décrivant leur évolution. Pour le tapis :



états :

0 libre

1 occupé

2 indisponible

3 suspendu

graphe d'état du stock dynamique fig II.14.

libre : Le stock dynamique est libre lorsqu'aucune tâche ne lui a été attribuée.

occupé : La fonction transport réalise une opération.

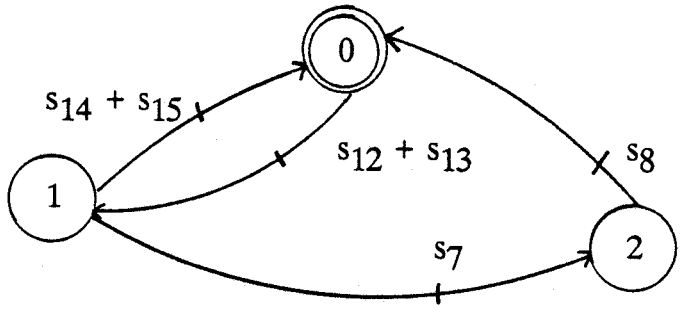
suspendu : La fonction transport a été interrompue par l'entrée d'une ressource dans la cellule.

indisponible : Le tapis est en panne ou en maintenance.

Les transitions sont définies par :

- s_1 entrée d'une ressource dans la cellule,
- s_2 une tâche est attribuée au tapis,
- s_3 la tâche est terminée,
- s_4 fin de traitement de l'entrée,
- s_5 incident,
- s_6 fin d'indisponibilité,
- s_7 décision de mise en maintenance,

Contrairement au stock dynamique, la zone commune n'est pas un composant actif. On lui associe un graphe permettant de savoir à tout instant si elle est accessible ou non.



<u>états</u> :

0 libre

1 occupé

2 indisponible

graphe d'état du stock statique fig II.15.

Le stock statique ne réalise jamais de tâche, certaines transitions sont différentes du cas dynamique, ainsi :

- s_{12} stockage d'une ressource,
- s_{13} utilisation d'une ressource pour un montage,
- s_{14} prise d'une ressource terminée.
- s_{15} opération de stockage achevée

IV.3.4. Ensemble des commandes admissibles

Pour un assemblage donné, en cours d'exécution, nous connaissons l'état des n opérations appartenant à T et celui des opérations de stockage.

Nous disposons de deux robots et le problème consiste alors à sélectionner parmi les tâches qui sont dans l'état exécutable, les prochaines qui seront réalisées.

On note :

EXEC(t) l'ensemble des opérations exécutables à l'instant t,

ATT(t) le sous-ensemble de EXEC(t) dont les éléments sont les deux opérations en attente d'exécution.

Le problème de décision est alors de définir l'ensemble ATT(t), quel que soit t, compte tenu de l'état des robots, de l'environnement et de l'assemblage à cet instant.

L'ensemble des commandes admissibles est :

$$U(t) = \{ u(t) / u(t) = (u_1(t), u_2(t)) \}$$

avec $u_i(t) \in EXEC(t) \cap CR_i(t)$

On pose $U_i(t) = \{ u_i(t) \} = EXEC(t) \cap CR_i(t)$

On a alors $u(t) \in (U_1(t) * U_2(t))$

En fait, nous aurons :

$$u(t) \in U(t) \subset (U_1(t) * U_2(t))$$

En effet, certaines contraintes doivent être respectées, par exemple,

- $u_1(t) = u_2(t)$, lorsque l'on réalise un travail à deux mains

- $u_1(t) \neq u_2(t)$ dans le cas contraire

- $u_1(t)$ et $u_2(t)$ ne peuvent utiliser une même ressource consommable si à l'instant t, elle est présente en un seul exemplaire.

Les tâches sont choisies de façon à optimiser un certain objectif, le problème est donc :

$$\forall t, \text{ trouver } f(u^*(t)) = \underset{u(t) \in U(t)}{\text{opt}} (f(u(t)))$$

L'ensemble des commandes admissibles varie dans le temps selon les différents événements tels que arrivée ou sortie d'une pièce, début ou fin de tâche, panne de robot, ...

Chacune de ces modifications de l'état de la cellule provoque une activation du module d'information qui effectue alors une mise à jour des différents états. Pour simplifier cette actualisation nous avons regroupé les tâches, selon leur état, en différents ensembles.

IV.4. Représentation par ensembles flous

En fait, pour qu'une tâche donnée soit dans l'état exécutable, deux conditions doivent être vérifiées :

1) chacun de ses antécédents doit être réalisé,

2) la ressource consommable nécessaire à son exécution doit être présente dans la cellule.

A la fin de l'exécution d'une tâche, certaines opérations deviendront pré-exécutables par rapport aux antécédents voire exécutables si les ressources qu'elles nécessitent sont présentes.

En effet, la réalisation d'une tâche qui possède un ou plusieurs successeurs, contribue au déblocage de ceux-ci. Certains d'entre-eux pourront alors être exécutés dès la fin de la tâche considérée, d'autres ne seront que partiellement débloqués.

Pour tenir compte de ces modifications, nous utilisons la notion d'ensembles flous (KAU 73).

Le référentiel considéré comprend l'ensemble des tâches à réaliser.

Définissons, à l'instant t, les ensembles suivants :

- PREA(t) : ensemble flou des tâches pré-exécutables par rapport aux antécédents. Cet ensemble est construit sur le référentiel en définissant la fonction d'appartenance d'une tâche à cet ensemble. Ce coefficient d'appartenance d'une tâche à l'ensemble flou PREA(t) est fonction de :

- Af(x,t) le nombre d'antécédents de x terminés à t,

- A(x) le nombre total d'antécédents.

Cette fonction doit être telle que :

* $\mu_{PREA(t)}(x) = 0$ lorsqu'aucun antécédent n'a été réalisé, ou que la tâche est terminée.

* $\mu_{PREA(t)}(x) = 1$ lorsque toutes les tâches prédécesseurs de x, sont terminées.

* f(Af(x,t),A(x)) est une fonction croissante du nombre d'antécédents de x terminés à l'instant t.

La figure ci dessous donne des allures possibles de cette fonction.

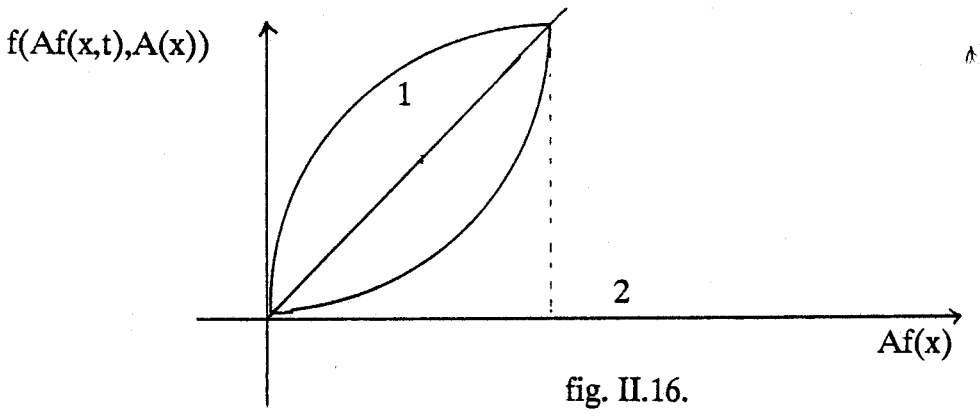


fig. II.16.

Si on examine la fonction $\Delta\mu(x)/\Delta Af(x,t)$, nous obtenons le "rendement marginal" relatif à la réalisation d'un antécédent.

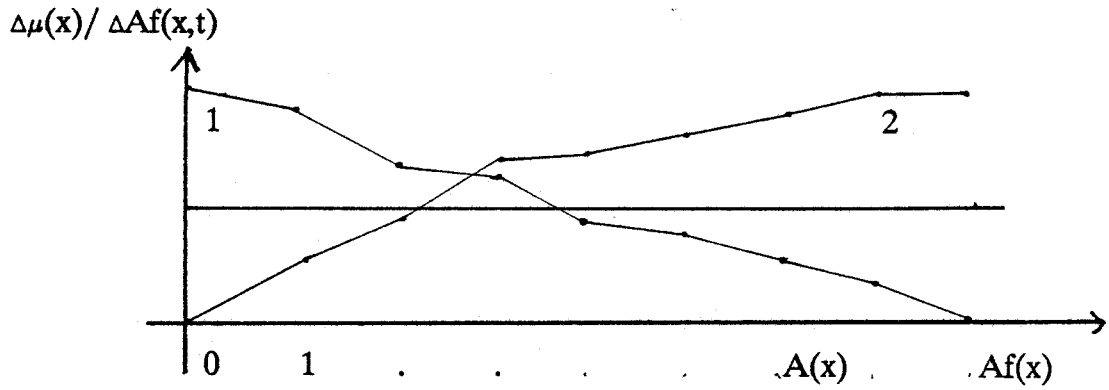


fig. II.17.

Dans le cas 1, la réalisation du premier antécédent d'une tâche apporte plus que celle d'un second ou n ième antécédent d'une autre tâche, ceci encourage à commencer de nouveaux ensembles d'antécédents. La courbe 1 se rapporte donc à des décisions favorables à la "mise en chantier".

Dans le second cas, l'inverse se produit et il est plus favorable de choisir l'antécédent d'une tâche dont un ou plusieurs prédécesseurs sont déjà réalisés. Ceci correspond à un encouragement à terminer des ensembles d'antécédents déjà entamés. La courbe 2 correspond à des décisions favorables à la "fin de chantier".

Le cas linéaire correspond à une attitude neutre et nous avons :

$$\mu_{PREA(t)}(x) = \frac{Af(x)}{A(x)} \qquad 0 \le \mu(x) \le 1$$

Par la suite, nous nous placerons dans ce cas pour avoir une fonction linéaire.

- PRER(t) : ensemble des tâches pré-exécutables par rapport aux ressources.

On peut modéliser cet ensemble par un ensemble flou si on définit sur le référentiel, une fonction d'appartenance à cet ensemble qui prend en compte les temps d'arrivée des ressources par une modélisation de type probabilité. Compte tenu des calculs engendrés par cette modélisation, (YAN 88), dans notre étude, nous ne considérons pas cet ensemble comme un ensemble flou. Les ressources ne sont pas directement affectées aux tâches. Dans ce sens, une tâche appartient à l'ensemble des pré-exécutables par rapport aux ressources dés que la ressource consommable nécessaire à sa réalisation est présente dans la cellule. Remarquons que cette même ressource rend aussi pré-exécutables toutes les tâches suceptibles de l'utiliser.

- EXEC(t) : ensemble flou des tâches exécutables. Nous avons :

$$EXEC(t) = PREA(t) \cap PRER(t)$$

avec $\mu_{EXEC(t)}(x) = \min(\mu_{PRER(t)}(x), \mu_{PREA(t)}(x))$

et $\mu_{PRER(t)}(x) \in \{0, 1\}$ $\mu_{PREA(t)}(x) \in [0, 1]$

Lorsque la tâche x est terminée, nous avons :

$$\mu_{\underline{EXEC(t)}}(x) = 0$$

Dans cet ensemble, nous distinguerons les trois sous-ensembles suivants : $\underline{EXEC(t)}$, $\underset{\sim}{EXEC}(t)$ et ATT(t).

On note $\underline{EXEC(t)}$ le sous ensemble de EXEC(t) défini de la façon suivante :

$\mu_{\underline{EXEC(t)}}(x) = 1$ si et seulement si $\mu_{EXEC(t)}(x) = 1$

$\mu_{\underline{EXEC(t)}}(x) = 0$ si et seulement si $\mu_{EXEC(t)}(x) < 1$

Cet ensemble représente les tâches qui peuvent réellement être exécutées à l'instant t.

ATT(t) est une partie de cet ensemble, il comprend les deux prochaines tâches à effectuer telles qu'elles ont été sélectionnées par le module de décision.

$$ATT(t) \subset \underline{EXEC(t)}$$

$\underset{\sim}{EXEC}(t)$ est le sous-ensemble de $\underline{EXEC(t)}$ qui contient les tâches telles que :

$\mu_{\underset{\sim}{EXEC}(t)}(x) = \mu_{EXEC(t)}(x)$ si et seulement si $\mu_{EXEC(t)}(x) < 1$

$\mu_{\underset{\sim}{EXEC}(t)}(x) = 0$ si et seulement si $\mu_{EXEC(t)}(x) = 1$

$$EXEC(t) = \underline{EXEC(t)} \cup \underset{\sim}{EXEC}(t)$$

avec $\mu_{EXEC(t)}(x) = \max(\mu_{\underline{EXEC(t)}}(x), \mu_{\underset{\sim}{EXEC}(t)}(x))$

Enfin, nous définissons ENC(t), le couple de tâches en cours de réalisation. Les tâches de ATT sont affectées aux robots à la libération de ceux-ci. Une tâche de l'ensemble ATT passe dans ENC dès que sa ressource robot devient libre.

V CONCLUSION

Dans ce chapitre, le module d'information du système de pilotage a été présenté. La modélisation par graphe d'état des différents éléments de la cellule a facilité la sélection des données pertinentes. Leur mise à jour est effectuée par activation du module d'information à l'occurrence de certains événements. Il est en effet nécessaire d'avoir une base de données correctes à tout instant, si l'on veut gérer la cellule en temps réel.

Cet ensemble d'informations nous permet de réaliser un ordonnancement des tâches à exécuter, compte tenu de l'état global de la cellule. Ceci est le rôle du module de décision que nous présentons dans le chapitre suivant. La solution retenue pour résoudre ce problème d'ordonnancement dynamique y est détaillée. C'est l'élément fondamental du module car c'est de la méthode utilisée que dépendent les performances du système.

CHAPITRE 3

MODULE DE DECISION

I INTRODUCTION

Le système de pilotage est conçu pour réaliser un ordonnancement et une affectation dynamiques des tâches aux différents robots. Ces décisions temps réel dépendent des résultats du suivi de production, et donc de l'état des composants de la cellule. Les informations nécessaires à la décision sont regroupées et gérées par le module d'information que nous avons défini au chapitre précédent.

Le module de décision que nous allons maintenant présenter, détermine l'ordre dans lequel les robots vont effectuer les différentes tâches nécessaires à la réalisation de la production demandée en satisfaisant les contraintes. Avant de proposer une solution à ce problème d'ordonnancement, nous exposerons brièvement les différents modèles.

II PROBLEME D'ORDONNANCEMENT

"Ordonnancer, c'est programmer dans le temps l'exécution d'une réalisation décomposable en tâches, en attribuant des ressources à ces tâches et en fixant en particulier leurs dates de début tout en respectant des contraintes données" (CAR 82).

Selon les types d'ateliers, la connaissance ou non du nombre de tâches, le nombre de machines (fixe ou variable), on choisira un modèle d'ordonnancement. On distingue en particulier les problèmes déterministes où les données sont fixées et où l'on ne considère pas les modifications de nature aléatoire, des problèmes stochastiques où certains éléments ne sont connus qu'en terme de probabilité (arrivée des tâches, délais, temps de fabrication, capacité des machines en raison de pannes éventuelles,...).

De nombreuses méthodes ont été élaborées (DUM 74), (KUS 85) (NEP 78), (CAR 82) pour déterminer des ordonnancements d'un nombre fini de tâches sur un horizon fixé. Ce sont des méthodes d'ordonnancements de type statique, c'est à dire pour lesquels aucune modification ne peut être introduite au programme de fabrication durant la période considérée.

Lorsque les tâches arrivent de façon aléatoire, on utilise des méthodes d'ordonnancements dynamiques (SOU 81), (SOU 82). Dans ce cas, l'ordonnancement est effectué périodiquement sur l'ensemble des tâches connues.

Pour repondre aux differentes perturbations (panne,...) qui peuvent se produire au sein d'une cellule on utilise des methodes d'ordonnancement dynamique

L'avantage d'un ordonnancement dynamique est donc, dans notre cas, de permettre d'affecter en temps réel les différentes tâches aux robots, en admettant l'occurrence de pannes. Le problème se pose alors de la façon suivante : Sélectionner les tâches à affecter aux robots afin de satisfaire un critère tout en respectant les contraintes de fabrication.

Les principales mesures de performance prises en compte dans les problèmes d'ordonnancement sont :

- l'utilisation des moyens de production,
- le respect des délais,
- les temps morts,
- le montant des en-cours de fabrication, des stocks de matières premières et des produis finis,
- les pertes de temps en reconversion de machines,
- le temps total de fabrication de chaque produit,
- les transferts de tâches dans l'atelier,
- les effets des pannes des machines ou plus généralement les aléas de fabrication.

Toutes ces mesures ne peuvent pas être optimisées en même temps. Il est cependant possible de combiner plusieurs de ces données pour obtenir une fonction globale de coût (BER 71).

Les techniques d'Intelligence Artificielle sont également utilisées pour créer des ordonnancements prévisionnels (BEL 87).

III ALLOCATION DE TACHES

III.1. Introduction

Dans notre étude, le critère retenu est la minimisation du temps total de réalisation d'un ensemble de produits donnés. En fait, nous avons considéré que la minimisation du temps total était liée à la minimisation du temps d'oisiveté des robots.

Les pièces arrivant de façon aléatoire dans la cellule, il en sera de même pour les tâches à effectuer. Pour éviter l'oisiveté des robots, nous proposons donc de maximiser,

à chaque instant, la quantité de travail disponible dans la cellule. Ceci nous conduit à maximiser le nombre de tâches que les robots pourront exécuter lorsqu'ils seront libres.

Ainsi posée l'affectation s'apparente à un problème d'optimisation monocritère dont la résolution ne présente à priori aucune difficulté. En fait, la sélection de la tâche qui maximisera le nombre de tâches exécutables à la libération des robots requiert la prise en compte d'un grand nombre de paramètres.

L'analyse du problème nous a permis d'en déterminer un certain nombre :

- *Nombre de successeurs*

S'il faut considérer le nombre de successeurs immédiats c'est à dire ceux qui deviendront pré-exécutables par rapport aux antécédents après réalisation de la tâche, il est intéressant de tenir compte du nombre total de successeurs (immédiats ou non) dont les ressources consommables sont dans la cellule; ils sont pré-exécutables par rapport aux ressources et deviendront directement exécutables à la fin de la réalisation de leurs antécédents et ainsi de suite, comme le montre le principe d'optimalité de Bellman.

- *Coefficient de déblocage*

Nous avons vu qu'une tâche pouvait avoir plusieurs antécédents ; chacun d'entre eux contribue au déblocage de la tâche, le dernier antécédent permet le passage de la tâche dans l'état préexécutable strict par rapport aux antécédents. Le coefficient de déblocage est une mesure de la distance entre l'état actuel et l'état pré-exécutable.

- *Temps d'exécution d'une tâche*

Le temps d'exécution d'une tâche peut varier suivant le robot qui réalise cette tâche, les robots n'étant pas obligatoirement identiques.

- *Fréquence d'arrivée des pièces*

La fréquence d'apparition des pièces qui est liée à la production amont pourrait également intervenir sur le choix d'une opération de stockage de préférence à une tâche de montage lorsqu'une pièce "rare" se présente sur le stock dynamique. Ce paramètre serait en fait une mesure d'utilité du stock.

- *Espace libéré sur le stock*

Si les pièces sont de tailles différentes, on peut également envisager d'affecter aux pièces en stock statique un coefficient fonction de l'espace occupé ; l'utilisation d'une ressource de ce stock contribue alors à l'augmentation de la zone de stockage.

- *Niveau de priorité du montage*

Dans le cas de plusieurs travaux exécutés en parallèle sur les différents plans de travail, selon la production souhaitée et l'avancement des assemblages, on peut définir une priorité relative des montages.

Ces différents exemples nous permettent de constater qu'il est nécessaire de considérer plusieurs paramètres pour satisfaire le critère retenu qui est la minimisation du temps de réalisation de l'ensemble des tâches.

La classification des critères proposée par Conway (CON 67) distingue ceux relatifs aux tâches et ceux relatifs aux machines. Le critère que nous avons adopté appartient à la première catégorie. La répartition des charges de travail des différents robots de manière à obtenir une égalisation du temps de travail est un exemple de critère relatif aux machines. Dans ce cas, on minimise l'inactivité de certains robots d'une part, mais également la suractivité (et donc l'usure) des autres et par voie de conséquence on réduit les risques de panne de ces robots. Selon les critères, les paramètres à considérer sont différents.

Le problème à résoudre se pose de la façon suivante :

Etant donné l'ensemble de tâches exécutables à l'instant t, déterminer la ou les "meilleures" tâches à affecter aux robots de manière à minimiser le temps total de réalisation.

III.2. Approche multicritères

Plusieurs approches ont été développées dans l'équipe de coopération entre robots. Un premier algorithme, basé sur une méthode d'optimisation multicritères, a été développé par M. DJEGHABA (DJE 86), (STA(b) 85). Il permet l'affectation des

tâches en considérant pour chacune d'elles, les critères suivants :

- nombre de successeurs d'une opération,
- coefficient de déblocage, fonction des antécédents non encore réalisés
- nombre de successeurs ayant leurs ressources consommables disponibles,
- temps de réalisation de l'opération,
- temps d'attente moyen d'une ressource propre, compte tenu de la fréquence d'arrivée des ressources et de la probabilité qu'une ressource soit utilisée par une autre cellule,

Il existe de nombreuses méthodes (BER 71), (GUI 77), (ROY 80) pour résoudre les problèmes multicritères telles que : hiérarchisation des critères, agrégation des critères, relation de surclassement, ... Ce sont essentiellement des méthodes d'aide à la décision qui permettent au décideur d'atteindre son objectif.

Dans l' approche adoptée, le système de décision est basé sur un algorithme utilisant la notion de surclassement. La méthode ELECTRE 1 (ROY 70) permet la sélection et l'attribution à un robot, lorsque celui-ci est libre, de la "meilleure" tâche à exécuter. Le problème principal de cette méthode est la détermination des pondérations des différents critères.

III.3. Modélisation par files d'attente (MAR 78)

Un deuxième algorithme, pour lequel la cellule a été modélisée par un système de files d'attente a été étudié par Y. YANG (YAN 88). Dans ce modèle, les robots sont les serveurs, les tâches exécutables les clients. Deux méthodes ont été proposées :

- une allocation à deux niveaux avec d'une part, une répartition des tâches qui consiste à affecter les tâches communes à l'une ou l'autre des files d'attente associées aux robots, et d'autre part, une discipline prioritaire au sein de chacune des files.

- une allocation de l'ensemble des tâches communes à chacune des files, avec la même discipline de service que précédemment et lors d'un choix identique, la résolution des conflits avec prise en compte du deuxième choix.

La politique de service est élaborée à partir d'un calcul de priorité, basé sur la quantité de travail apportée à la file d'attente (STA(a) 87), (YAN 87). Compte tenu du fait que l'exécution d'une tâche permet le déblocage d'un certain nombre d'antécédents,

le but de la méthode est d'évaluer par une heuristique, la quantité de travail qui entrera dans les files d'attente. Le problème est alors d'évaluer ce gain qui se compose d'une part d'un travail immédiat dans le sens où les tâches débloquées pourront effectivement être réalisées, et d'autre part d'un travail futur dans la mesure où certaines tâches ne sont que partiellement débloquées et ne seront exécutables qu'au bout d'un certain temps que l'on peut borner.

Dans ces conditions, le calcul du gain apporté par une tâche est élaboré à partir du nombre de tâches débloquées et de la probabilité d'arrivée des ressources.

III.4. Approche retenue

Pour le système de pilotage que nous présentons, la procédure, inspirée des travaux précédents, lève un certain nombre d'hypothèses et prend en considération d'autres conditions quant à la décision d'affectation.

- *Affectation prévisionnelle*

Nous avons vu qu'une tâche est exécutable lorsque ses antécédents sont terminés et que les ressources qu'elle requiert sont disponibles. La sélection des tâches à affecter se fait parmi cet ensemble de tâches exécutables. Cependant, une tâche ne peut être exécutée que lorsque les ressources récupérables utiles à sa réalisation sont disponibles. La décision d'affectation peut donc être prise à différents instants selon la disponibilité de ces ressources.

Dans les travaux précédents, l'affectation est définitive, dans la mesure où l'activation du module de décision est réalisée lorsque l'un des robots est libre et les ressources récupérables disponibles.

Dans ce travail, l'affectation est calculée de manière prévisionnelle, c'est à dire que pendant que le robot exécute sa tâche, nous sélectionnons la suivante.

La procédure développée prend en compte l'arrivée dynamique des pièces et donc le temps d'accessibilité à ces ressources. Dans ce sens, cette sélection anticipée permet de gérer l'avance du tapis pendant le travail des robots et ainsi de minimiser les temps d'attente de ressources consommables.

- *Affectation d'un couple de tâches*

Le but du module de décision étant d'affecter les tâches à effectuer, en assurant la coopération entre les deux robots et en minimisant le temps total de réalisation d'un ensemble de montages, il nous a paru souhaitable de considérer les deux robots ensemble. En effet, les robots ayant d'une part un objectif commun, d'autre part un même système d'alimentation, la prise de décision pour chacun des robots indépendamment des charges de l'autre apparait sous-optimale.

Dans ce sens, au lieu d'ordonner les tâches de manière à ce que chacun des robots optimise le critère, nous sélectionnerons les tâches par couple en prenant en compte tous les éléments de la cellule.

En fait, à chaque instant la décision ne porte que sur un couple, il est en effet inutile de prévoir à plus long terme, puisque l'on ne connait ni l'ordre d'arrivée des ressources ni les instants de pannes des robots.

- *Méthode utilisée*

Le but de la décision est de sélectionner parmi les tâches exécutables à l'instant t ou au plus tard à la fin des tâches en cours, les deux "meilleures" tâches qui seront affectées aux robots dés que ceux-ci seront libres; le "meilleur" couple étant celui qui apportera la quantité de travail maximale dans la cellule (BAY 87).

Le problème consiste alors à déterminer une mesure de la quantité de travail apportée par un couple de tâches. Ce gain étant basé sur une estimation de l'ensemble des tâches exécutables, nous nous intéresserons, dans un premier temps à l'évolution des ensembles, puis nous exposerons le calcul du gain associé à un couple. Dans un premier temps, les calculs sont développés pour une gamme décrite par graphe, puis nous présentons le cas particulier d'une structure arborescente.

IV APPROXIMATION DES DIFFERENTS ENSEMBLES

Pour prévoir les tâches qui seront exécutées à la libération de chacun des robots, nous devons estimer l'ensemble des tâches exécutables tel qu'il sera à ces instants.

Nous avons : $EXEC(t) = PREA(t) \cap PRER(t)$

Cet ensemble évolue dans le temps selon les événements suivants :

- entrée ou sortie d'une pièce en stock,
- début ou fin d'exécution d'une tâche.

L'évolution de l'ensemble EXEC dépend donc de l'évolution de chacun des ensembles qui le composent. Examinons ces différents ensembles.

IV.1. Evolution de l'ensemble PREA(t)

Les tâches i et j étant en cours de réalisation, nous pouvons calculer l'ensemble PREA(t) tel qu'il sera aux instants c_i et c_j.

Nous supposons que $c_i < c_j$.

L'ensemble $PREA(c_i)$ dépend de :

- $PREA(t_i)$
- les tâches débloquées par i.

Sous l'hypothèse qu'aucune panne n'intervient, ces deux ensembles sont parfaitement connus, il en sera donc de même pour $PREA(c_i)$.

L'ensemble $PREA(c_j)$ dépend de :

- $PREA(t_j)$
- des tâches débloquées par i et j
- des tâches éventuellement débloquées dans l'intervalle $[c_i, c_j]$.

Sous l'hypothèse qu'aucune panne n'intervient pendant l'exécution de i et j, les deux premiers sous ensembles sont parfaitement connus, le troisième ne le sera que dans certains cas.

- Sous ensembles des tâches débloquées pendant $[c_i, c_j]$

Nous avons supposé que $c_i < c_j$, à l'instant c_i nous affecterons donc au robot libéré, la tâche i+1. Plusieurs cas sont alors à considérer.

1. $c_{i+1} > c_j$

Dans ce cas, aucune tâche n'est débloquée dans l'intervalle $[c_i, c_j]$ et ce sous ensemble est vide.

2. $c_{i+1} = c_j$

L'ensemble des tâches débloquées à c_j sera l'union des ensembles PREA(t_i) et de l'ensemble des tâches débloquées par i, i+1 et j.

3. $c_{i+1} < c_j$

L'ensemble des tâches débloquées dans l'intervalle $[c_i, c_j]$ comprend alors les tâches débloquées par i+1 et celles qui le seront dans l'intervalle $[c_{i+1}, c_j]$.

Dans tous les cas, à l'instant de décision, le choix de la tâche i+1 n'est pas effectué. Aussi nous ne pouvons pas connaître par avance le cas (1,2 ou 3) qu'il faut considérer.

Nous travaillons donc sur une estimation de l' ensemble PREA en considérant le cas 1, c'est à dire celui où aucune tâche n'est débloquée pendant $[c_i, c_j]$. C'est une position pessimiste puisqu'elle correspond au cas le plus défavorable.

Si l'on se place dans une attitude neutre (cf Ch II, §IV), le coefficient d'appartenance est une fonction linéaire des antécédents finis à l'instant t.

L'évolution de l'ensemble PREA(t) est décrite par

$$x \in S(i) \qquad \mu_{PREA(c_i)}(x) = \mu_{PREA(t_i)}(x) + \frac{1}{A(x)}$$

$x \notin S(i)$ $\qquad \mu_{PREA(c_i)}(x) = \mu_{PREA(t_i)}(x)$

$x = i$ $\qquad \mu_{PREA(c_i)}(x) = 0$

IV.2. Evolution de l'ensemble PRER(t)

Nous avons vu qu'une tâche appartenait à l'ensemble PRER(t) lorsque la ressource consommable nécessaire à sa réalisation était dans un des stocks. L'évolution de cet ensemble est donc fonction du flux des ressources, détaillons ces variations :

Soient $RC(t)$, l'ensemble des différentes ressources consommables présentes à l'instant t,

$RC_k(t)$, l'ensemble des ressources de type k présentes à l'instant t, on en note n_k le nombre d'exemplaires,

$P_s(t, \delta t)$, l'ensemble des pièces sorties de la cellule pendant un intervalle de temps $[t, t+\delta t]$,

$P_e(t, \delta t)$, l'ensemble des pièces entrées dans la cellule durant ce même intervalle.

L' ensemble PRER(t) est modifié aux instants suivants :

IV.2.1. Entrée d'une pièce

A l'instant $t+\delta t$ où entre une pièce p dans la cellule, nous avons :

$$PRER(t+\delta t) = PRER(t) \cup \{x/p \in P_e(t, \delta t), p \notin RC(t), p = \Gamma^{-1}(x)\}$$

En effet, d'après la définition (Ch. II, § IV.3.), l'ensemble des tâches pré-exécutables par rapport aux ressources n'est modifié que lorsqu'il entre une pièce d'un type qui n'est pas encore dans la cellule.

IV.2.2. Utilisation d'une ressource

Soit la tâche i utilisant une ressource p appartenant à RC_k, connaissant $PRER(t_i)$, calculons $PRER(c_i)$.

1) Cas où i est une tâche de montage

Soit $n_k > 1$, entre $[t_i, c_i]$, il n'a pas été utilisé plus d'une ressource de type k.

Si $p \in RC_K(t_i)$, $n_k > 1$, alors $p \in RC_K(c_i)$, et :

$$PRER(c_i) = PRER(t_i) \setminus \{ i \}$$

Soit $n_k = 1$, dans ce cas, la seule ressource de type k présente à l'instant t_i a été utilisée, les tâches appartenant à $PRER(t_i)$ et utilisant ce type de pièce deviennent non exécutables.

Si $p \in RC_K(t_i)$, $n_k = 1$, alors $p \notin RC_K(c_i)$, et :

$$PRER(c_i) = PRER(t_i) \setminus \{ x/p = \Gamma^{-1}(i), p \in RC_k(t_i), p = \Gamma^{-1}(x)\}$$

2) Cas où i est une tâche de stockage

La pièce reste disponible et :

$$PRER(c_i) = PRER(t_i)$$

IV.2.3. Sortie d'une pièce

Si, à l'instant t, on choisit une ressource p sur le stock dynamique pour exécuter une tâche i, alors les pièces situées en aval de la ressource vont quitter la cellule. La sortie de la pièce n'a de conséquence sur l'ensemble PRER que si elle était l'unique exemplaire de son type ($n_k = 1$).

Dans ce cas, à l'instant t où une pièce sort de la cellule, nous avons :

$$PRER(t+\delta t) = PRER(t) \setminus \{x/q \in Ps(t,\delta t) \cap RC_k(t), q = \Gamma^{-1}(x)\}$$

Les résultats précédents nous permettent d'estimer l'ensemble PRER tel qu'il sera à c_i si la tâche i est sélectionnée à l'instant t. Ainsi nous avons :

$$PRER(c_i) = PRER(t) \setminus \{ x/p = r^{-1}(i), p \in RC_k(t), p = r^{-1}(x) \}$$

$$\setminus \{ x/q \in Ps(t, \delta t_i) \cap RC_k(t), q = r^{-1}(x) \}$$

$$\cup \{ x/q \in P_e(t, \delta t_i), q \notin RC(t), q = r^{-1}(x) \}$$

avec $\delta t_i = c_i - t$, $RC_k(t)$ tel que $n_k = 1$

Cependant, il est à noter que si la sortie des pièces de la cellule est gérée par le système de décision, l'entrée des pièces est un phénomène aléatoire. C'est pourquoi, de la même façon que pour l'ensemble PREA(t), nous travaillerons sur une approximation de PRER(t) ; nous nous plaçons dans le cas le plus défavorable puisque nous considérons qu'aucune pièce n'entrera dans l'intervalle $[c_i, c_j]$.

IV.3. Evolution de l'ensemble des tâches exécutables

L'ensemble flou des tâches exécutables est défini par :

$$EXEC(t) = PREA(t) \quad PRER(t)$$

avec $$\mu_{EXEC(t)}(x) = \min (\mu_{PREA(t)}(x), \mu_{PRER(t)}(x))$$

Compte tenu du fait que nous avons deux robots, que certaines tâches ne sont réalisables que par un seul de ceux-ci, et d'autres par les deux, les ensembles précédents peuvent s'exprimer par :

$$EXEC(t) = EXEC1(t) \cup EXEC2(t)$$

$$PREA(t) = PREA1(t) \cup PREA2(t)$$

$$PRER(t) = PRER1(t) \cup PRER2(t)$$

EXECI(t), PREAI(t), PRERI(t) ensembles des tâches qui seront exécutées par le robot I, I = 1, 2.

L'ensemble des tâches exécutables est modifié à chaque événement transformant l'un des ensembles PREA(t) et PRER(t).

Nous avons vu que l'ensemble des tâches préexécutables par rapport aux antécédents était modifié à la fin de l'exécution d'une tâche et que cette évolution pouvait être prévue lorsque l'on commence cette tâche.

En ce qui concerne l'ensemble des tâches préexécutables par rapport aux ressources, il est modifié à chaque entrée ou sortie de pièce et à chaque début de tâche.

L'évaluation de l'ensemble EXEC(t) est directement liée à ces ensembles et par voie de conséquence ne pourra qu'être estimé.

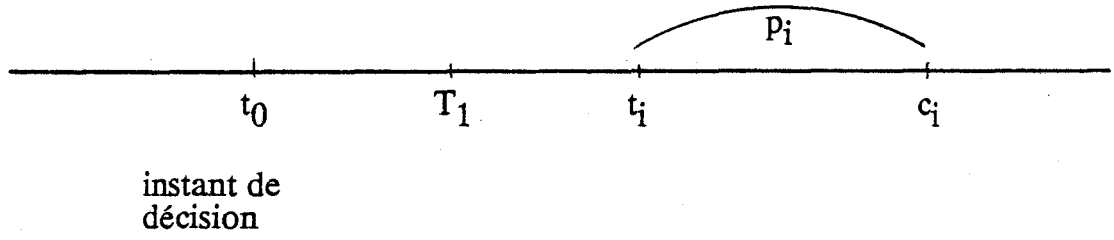
V GAIN ASSOCIE A UN COUPLE DE TACHES

V.1. Représentation par graphe

Pour clarifier l'exposé, nous présenterons dans un premier temps, l'apport d'une tâche puis nous étudierons le cas d'un couple de tâches.

V.1.1. Retour associé à l'exécution d'une tâche

Soit T_1 l'instant auquel le robot devient libre, t_i la date à laquelle la tâche i pourra réellement commencer, c_i la date de fin de la tâche i,



La sélection de la prochaine tâche à affecter au robot pourrait être réalisée à l'instant T_1 où le robot devient libre. Toutefois, il est souhaitable de décider plus tôt, (instant t_0) quelle sera la prochaine tâche à effectuer, de manière à ce que la ressource nécessaire puisse être en position de prise, à l'instant T_1. Ceci permet de gagner le

temps de transport mais pose néanmoins le problème d'une éventuelle remise en cause de la décision si une nouvelle ressource entre dans la cellule.

Le choix de réaliser la tâche i entraîne l'apport d'une quantité de travail disponible à l'instant t_i, (la tâche i elle-même) ainsi que d'une quantité de travail dont une partie ne sera disponible au plus tôt qu'à l'instant c_i.

En effet, la tâche possède des successeurs qui seront totalement ou partiellement débloqués à la date c_i. Parmi ces successeurs, un certain nombre deviendront exécutables car les ressources nécessaires à leur réalisation sont ou seront présentes dans les stocks.

L'évaluation de l'ensemble PREA traduit le déblocage des tâches successeurs de i.

Le choix de la tâche i peut également entraîner une diminution de la quantité de travail dans la mesure où certaines tâches ne seront plus préexécutables par rapport aux ressources, en particulier lorsque :

- La tâche i nécessite une ressource de type k qui est dans le stock dynamique. La prise de cette ressource entraîne la perte de toutes les pièces situées en aval. Si parmi elles, certaines sont uniques dans leur type, des tâches de PRER(t_0) deviendront non exécutables.

- La tâche i nécessite une ressource de type k qui existe, à l'instant t_0 de décision, en un seul exemplaire dans les stocks. Compte tenu de la définition que nous avons adoptée, la prise de cette pièce pour l'exécution de la tâche i implique que toutes les tâches appartenant à l'ensemble PRER(t_0) et nécessitant le même type de ressources reviennent dans l'état non exécutable.

L'évolution de l'ensemble PRER rend compte de la diminution du nombre de tâches consécutive au choix de i.

Remarque : une tâche de stockage n'augmente pas la quantité de travail dans le système puisque les pièces du stock statique sont prises en compte dans PRER. Au contraire, le choix de stocker une pièce permettra de ne pas perdre une certaine quantité de travail existante dans la cellule à un instant.

En fait, nous nous plaçons sous une hypothèse pessimiste, dans la mesure où nous ne considérons pas les entrées des pièces qui peuvent rendre certaines tâches

exécutables. Cette hypothèse correspond à une attitude "prudente" dans le cadre de la commande en environnement incertain (BER 57), (GUI 68).

La quantité de travail globale apportée au système est donc une fonction des ensembles précédents. Pour l'évaluer, nous avons choisi de l'exprimer en temps, en distinguant deux composantes :

Un travail présent :

Il est fonction de la durée de la tâche et de la date de début de tâche (en fait $t_i - t_0$). Cependant, nous savons qu'une tâche, quelle qu'elle soit, ne peut commencer avant l'instant T_1. Aussi, les calculs seront exécutés en référence à T_1 qui correspond à l'instant de libération du robot.

La tâche i ne devient exécutable strictement qu'à l'instant t_i, le travail présent relatif à cette tâche n'est donc disponible qu'au bout du temps $(t_i - T_1)$. Nous le considérons donc sous sa forme actualisée.

Nous avons alors, travail présent = $p_i \, . \, e^{- \beta (t_i - T_1)}$

L'influence du paramètre β dont dépend le coefficient d'actualisation $e^{- \beta t}$ est présentée en annexe.

Un travail futur :

Il est fonction des tâches débloquées à c_i, à savoir :

travail futur = $\Phi [EXEC(c_i)] \, . \, e^{- \beta (t_i + p_i - T_1)}$

$$= \Phi [EXEC(c_i)] \, . \, e^{- \beta (c_i - T_1)}$$

où Φ [EXEC] est une mesure de la quantité de travail qui sera présente dans EXEC.

L'évaluation de la quantité globale de travail apportée au système par le choix d'exécuter la tâche i est donnée par :

$$R(i) = (c_i - t_i) e^{- \beta (t_i - T_1)} + \Phi [EXEC(c_i)] \, e^{- \beta (c_i - T_1)}$$

Le gain apporté par le travail futur est fonction de la durée des tâches débloquées à c_i. Nous avons vu que certaines d'entre elles seront exécutables, et que d'autres ne seront que partiellement débloquées. Pour tenir compte de ce travail fourni

mais non exécutable à c_i, nous pondérons la durée des tâches par leur coefficient d'appartenance à l'ensemble EXEC.

En fait, si j est une tâche non exécutable à c_i, j deviendra exécutable au bout d'un temps δj. Cette valeur dépend des décisions futures et des arrivées de ressources. Dans la mesure où nous n'avons aucune information sur l'arrivée des pièces, nous ne pouvons pas borner cette date. Le calcul, dans le cas d'un robot, et utilisant les probabilités d'entrée des ressources est proposé dans (STA(a) 87). Dans cette étude, nous choisissons $\delta_j = 0$, quel que soit j.

Ainsi,nous définissons :

- <u>cas 1</u> (Γ-1 bijective)

$$\Phi(EXEC) = \sum_x \mu_{EXEC(c_i)}(x)*(c_x - t_x)$$

$$= \sum_x \mu_{EXEC(c_i)}(x) * p_x$$

- <u>cas 2</u> (Γ-1 surjective)

Dans ce cas, pour chaque ressource de type k (présente à n_k exemplaires) apparaissent dans EXEC, les tâches utilisant cette ressource (présentes à N_k exemplaires). Ces deux nombres n'étant pas forcément égaux, il convient de définir une fonction Φ_k[EXEC] pour chaque type de ressource k. Celle-ci exprimera la quantité de travail présente lorsque $N_k(t)$ tâches sont en compétition pour $n_k(t)$ ressources.

$\Gamma(k)$ est l'ensemble de tâches nécessitant une ressource de type k.

Nous avons :

$$\mu_{EXEC(t)}(x) = \min\left(\mu_{PRER(t)}(x), \mu_{PREA(t)}(x)\right)$$

Si $n_k(t) = 0$ $\quad \forall x \in \Gamma(k), \quad \mu_{EXEC(t)}(x) = \mu_{PRER(t)}(x) = 0$

Si $\Gamma(k) \cap PREA(t) = \phi$ $\quad \forall x \in \Gamma(k), \quad \mu_{EXEC(t)}(x) = \mu_{PREA(t)}(x) = 0$

Si $n_k(t) > N_k(t)$ $\quad \Phi_k[EXEC] = \sum_{x \in \Gamma(k)} \mu_{EXEC(t)}(x) \cdot p_x$

Il nous faut maintenant définir Φ_k [EXEC] lorsque le nombre de ressources de type k est inférieur au nombre de tâches utilisant ce type de ressources.

Nous devons ici distinguer deux cas : celui où N_k tâches au moins sont exécutables au sens strict ($\mu_{EXEC(t)}=1$) et celui où seule une partie des N_k tâches peuvent être immédiatement effectuées.

- 1 - Les Nk tâches sont strictement exécutables, plusieurs solutions peuvent être envisagées :

- estimation "optimiste"

On classe les $N_k(t)$ tâches selon l'ordre décroissant de leur apport en quantité de travail et on affecte les $n_k(t)$ ressources aux $n_k(t)$ premières tâches. On obtient :

$$\Phi_k\,[EXEC] = \max_{P_n(\Gamma(k))} \sum_{x \in \Gamma(k)} \mu_{EXEC(t)}(x) \cdot p_x$$

où $P_n(\Gamma(k))$ est l'ensemble des parties à $n_k(t)$ éléments de l'ensemble $\Gamma(k)$

c'est à dire : $$\Phi_k\,[EXEC] = \max_{P_n(\Gamma(k))} \sum_{x \in \Gamma(k)} p_x$$

(étant donné que $\mu_{EXEC(t)}(x) = 1$)

Cette estimation est optimiste dans le sens où l'on considère que les prochaines décisions affecteront les n_k ressources aux N_k tâches ayant le plus important apport à EXEC(t). Il est fort probable que ceci ne se produise pas, mais cette solution nous donne une majoration de Φ_k[EXEC].

On peut également chercher une minimisation de cette quantité de travail :

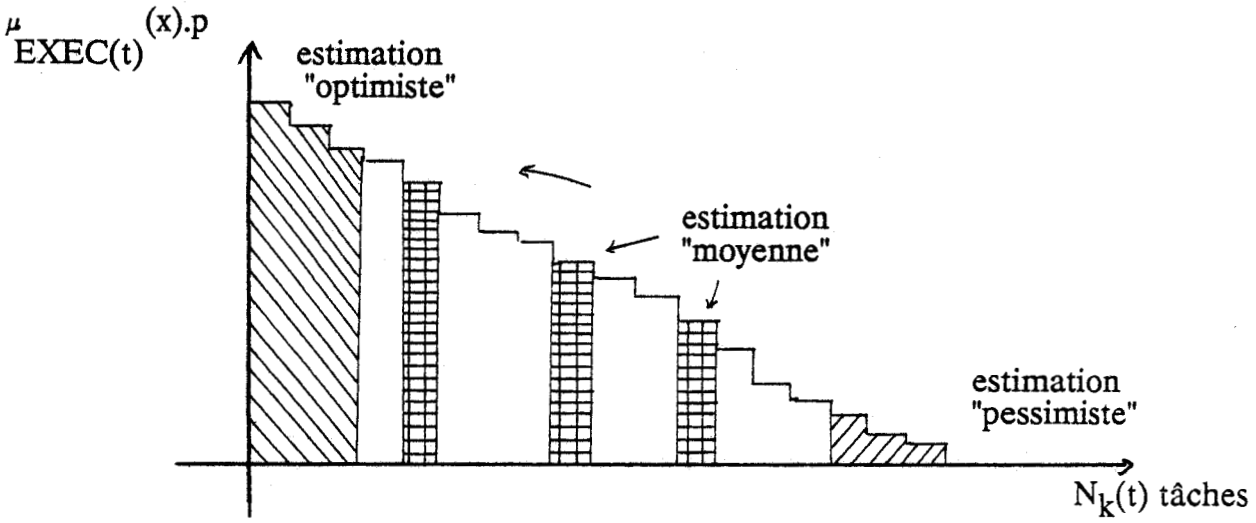
- estimation "pessimiste"

Elle consiste à affecter les $n_k(t)$ ressources aux $N_k(t)$ tâches représentant la plus faible quantité de travail, alors :

$$\Phi_k\,[EXEC] = \min_{P_n(\Gamma(k))} \sum_{x \in \Gamma(k)} \mu_{EXEC(t)}(x) \cdot p_x$$

soit ($\mu_{EXEC(t)}(x) = 1$) :

$$\Phi_k\,[EXEC] = \min_{P_n(\Gamma(k))} \sum_{x \in \Gamma(k)} p_x$$



Affectation de $n_k(t)$ ressources à $N_k(t)$ tâches fig. III.1.

- estimation "moyenne"

Cette solution nous donnera une valeur comprise entre les deux précédentes, ainsi :

$\Phi_k\,[EXEC] = \gamma_1\,\Phi_k max + \gamma_2\,\Phi_k min$ avec $\gamma_1 + \gamma_2 = 1$, γ_1, $\gamma_2 > 0$

γ_1 coefficient d'optimisme, γ_2 coefficient de pessimisme.

- 2 - Les N_k tâches ne sont pas strictement exécutables et seules $(N_k - M_k)$ ont un coefficient d'appartenance à EXEC(t) = 1.

Si $n_k \leq (N_k - M_k)$ le calcul précédent s'applique puisqu'il nous faut choisir n_k tâches parmi $(N_k - M_k)$ strictement exécutables.

Si $n_k > (N_k - M_k)$, nous affectons $(N_k - M_k)$ ressources aux $(N_k - M_k)$ tâches strictement exécutables. Le problème est alors de définir la quantité de travail Φ_k' lorsque M_k tâches qui ne sont pas strictement exécutables sont en compétition pour m_k ressources avec $m_k = (n_k - (N_k - M_k))$.

De la même façon que précédemment, nous pouvons évaluer cette valeur et nous avons alors :

- *Une estimation "optimiste"* où l'on classe les $M_k(t)$ tâches selon l'ordre décroissant de leur apport et on affecte les ressources restantes aux $m_k(t)$ tâches représentant le plus grand apport.

On obtient : $$\Phi'_k[EXEC] = \max_{P_m(\Gamma(k))} \sum_{x \in \Gamma(k)}^{R} \mu_{EXEC(t)}(x) \cdot p_x$$

où $P_m(\Gamma(k))$ est l'ensemble des parties à $m_k(t)$ éléments de l'ensemble $\Gamma(k)$.

- *Une estimation "pessimiste"* qui consiste à affecter les $m_k(t)$ ressources aux $M_k(t)$ tâches correspondant au plus faible apport :

$$\Phi_k'[EXEC] = \min_{P_m(\Gamma(k))} \sum_{x \in \Gamma(k)}^{R} \mu_{EXEC(t)}(x) \cdot p_x$$

- *Une estimation "moyenne"* qui nous donne une valeur comprise entre les deux précédentes, ainsi :

$$\Phi_k'[EXEC] = \gamma_1 \Phi_k'max + \gamma_2 \Phi_k'min \quad \text{avec } \gamma_1 + \gamma_2 = 1, \ \gamma_1, \ \gamma_2 > 0$$

Dans ce cas, nous calculons :

$$\Phi_k[EXEC] = \sum_{\substack{x \in \Gamma(k) \\ \omega/\mu_{EXEC}(x)=1}} p_x + \Phi_k'[EXEC]$$

Pour l'ensemble des ressources, nous obtenons :

$$\Phi[EXEC] = \sum_k \Phi_k[EXEC]$$

Cette mesure permet la prise en compte des tâches qui seront strictement exécutables à l'instant c_i et le travail futur apportée par les tâches pour lesquelles $(\mu_{EXEC(c_i)}(i) < 1)$ à l'instant c_i.

Elle est calculée pour toutes les tâches qui seront exécutables à l'instant T_1 de libération du robot.

V.1.2. Retour associé à l'exécution d'un couple de tâches

Dans la mesure où nous sélectionnons les tâches par couple, et que le critère est de maximiser la quantité de travail apportée au système, il nous faut maintenant calculer le retour associé à l'exécution d'un couple de tâches (i, j).

Nous savons calculer le gain consécutif à l'exécution d'une tâche. Pour un couple, l'apport R(i,j) est fonction :

- d'un travail présent relatif au robot 1 et disponible à l'instant T_1 :

$$p_i \, e^{-\beta(ti-T1)},$$

- d'un travail futur relatif au robot 1 et disponible à c_i :

$$\Phi\,[EXEC(c_i)]\, e^{-\beta\,(ci-T1)},$$

- d'un travail présent relatif au robot 2 et disponible à l'instant T_2 :

$$p_j \, e^{-\beta(tj-T2)},$$

- d'un travail futur relatif au robot 2 et disponible à c_j :

$$\Phi\,[EXEC(c_j)]\, e^{-\beta\,(cj-T2)}.$$

Si on suppose $T_2 > T_1$, on constate que si la réalisation de la tâche i est sûre (ceci est vrai si la décision n'est plus modifiée entre t_0 et T_1), celle de j est incertaine puisque l'on sait qu'il y aura, au plus tard, une nouvelle décision à l'instant T_1. Pour tenir compte de cette incertitude, nous pondérons la quantité de travail apportée par j par le coefficient d'actualisation : $e^{-\beta\,(T2-T1)}$.

Le travail présent consécutif à l'exécution du couple (i, j) est alors :

$$p_i \cdot e^{-\beta\,(t_i-T_1)} + p_j \cdot e^{-\beta\,(t_j-T_1)}$$

Compte tenu de la définition de la fonction Φ [EXEC], les quantités de travail futur des deux robots ne s'additionnent pas.

$$\text{Soit } \Phi(c_i) = \Phi\,[EXEC(c_i)] \quad e^{-\beta\,(c_i-T_1)}$$

$$\text{et } \Phi(c_j) = \Phi[EXEC(c_j)] \quad e^{-\beta (c_j - T_1)}$$

La quantité disponible à $c_{min} = \min(c_i, c_j)$ est plus sûre que celle débloquée à $c_{max} = \max(c_i, c_j)$.

Nous posons $\Phi(i,j)$ la quantité de travail futur consécutive à l'exécution de (i,j).

$$\Phi(i,j) = f(\Phi(c_{min}), \Phi(c_{max}))$$

Si on adopte $\Phi(i,j) = \Phi(c_{min})$, on ne considère que le travail sûr, mais ceci correspond au travail futur relatif à une seule tâche.

Si on choisit $\Phi(i,j) = \Phi(c_{max})$, on prend en compte le travail futur débloqué par les deux tâches. Cependant, le travail sûr est alors pondéré du même coefficient que le travail incertain.

En fait, la quantité de travail fournie à l'instant c_{max}, est égale à la somme du gain sûr, disponible à c_{min}, et de celui incertain obtenu dans l'intervalle (c_{min}, c_{max}), aussi :

$$\Phi(i,j) = \Phi(c_{min}).\ e^{-\beta (c_{min} - T_1)} + \Delta\Phi.\ e^{-\beta (c_{max} - T_1)}$$

avec $\Phi(c_{max}) = \Phi(c_{min}) + \Delta\Phi$

Dans ces conditions, nous avons :

$$R(i,j) = p_i.\ e^{-\beta (t_i - T_{min})} + p_j.\ e^{-\beta (t_j - T_{min})}$$

$$+ \Phi[EXEC(c_{min})]\ e^{-\beta (c_{min} - T_{min})} + \Delta\Phi\, e^{-\beta (c_{max} - T_{min})}$$

avec $T_{min} = \min(T1, T2)$.

Généralisation : dans le cas de n robots, on classe les c_i i=1 à n, selon leur ordre croissant, nous avons alors :

$$R(1,2,..,n) = \sum_{n} p_i . \; e^{-\beta (t_i - T_{min})} + \sum_{n} \Delta\Phi (i, i+1). \; e^{-\beta (c_{i+1} - T_{min})}$$

avec $\Delta\Phi (i, i+1) = \Phi (c_{i+1}) - \Phi (c_i)$, i= 0 à n, $\Phi (c_0) = 0$,

Tmin = min (T_j) j =1 à n, Tj, date de disponibilité du robot j.

V.2. Cas d'une structure arborescente

La procédure présentée s'appuie sur une description des tâches à effectuer sous forme de graphe. Partant d'un point initial, le problème consiste alors à exécuter toutes les tâches, qui sont les noeuds, pour atteindre l'état final. Dans cette représentation, une opération peut posséder plusieurs antécédents.

Dans une description arborescente, le problème se pose différemment dans la mesure, où il est nécessaire de réaliser toutes les tâches d'un chemin qui relie le point de départ, état initial, au point d'arrivée, état final.

Nous avons vu au chapitre 2 qu'il était possible d'effectuer un regroupement de l'ensemble des séquences par des graphes en utilisant les permutations possibles et donc en faisant apparaître les possibilités de parallélisme.

L'arbre représentant l'ensemble des séquences vérifiant les conditions de réalisabilité peut ainsi être représenté par plusieurs graphes. Le problème qui se pose alors est de réaliser entièrement un de ces graphes.

Une solution consiste à sélectionner parmi l'ensemble des graphes, celui qui correspond au nombre maximal de séquences et à appliquer à ce graphe la procédure développée.

En fait, dans la mesure où nous savons sélectionner le meilleur couple pour un graphe donné, nous pouvons effectuer ce choix sur tous les graphes possibles. Puis nous

retenons sur les différents couples optimaux, le meilleur couple c'est à dire celui qui apporte une quantité de travail maximale.

A chaque étape, le choix du couple optimal élimine les graphes incompatibles avec la solution choisie, ceci correspond en fait, au choix d'un chemin sur l'arbre.

Connaissant l'apport d'un couple de tâches, nous allons calculer la date effective de début des tâches.

VI DATE DE DEBUT REEL D'UNE TACHE

Nous avons vu au chapitre II que :

$$t_i \geq \max\left(\max_{j \in A(j)} (t_j + p_j), \max_{r \in R(i)} T_r \right)$$

T_r date de disponibilité de la ressource r.

Pour connaître la date effective de début d'une tâche, nous allons calculer ces différents temps.

VI.1. Temps d'accessibilité aux ressources récupérables

La date d'accessibilité aux robots est donnée par T_1 pour le robot 1, T_2 pour le robot 2 qui sont les temps d'occupation des robots.

Les plans de travail propres à chacun des robots sont toujours disponibles, le plan commun n'est quand à lui accessible que lorsque la zone est libérée, notons T_v cette date.

Ces temps T_1, T_2 et T_v sont connus par le planning d'utilisation des ressources.

Remarque : Si une des tâches demande la disponibilité des deux robots, la seconde tâche ne pourra commencer qu'à la libération d'un des robots.

VI.2. Date d'arrivée d'une ressource consommable

La pièce nécessaire à l'exécution d'une tâche peut se trouver, soit en stock statique, soit en stock dynamique. Nous connaissons à chaque instant la position de cette ressource, le problème est alors de déterminer la date à laquelle elle sera disponible effectivement, c'est à dire :

pour une pièce en stock dynamique, l'instant où elle sera en position de prise devant le robot qui l'utilise pour effectuer une tâche.

pour une pièce en stock statique, la date de disponibilité de la zone commune qui est utilisée en exclusion mutuelle, dans la mesure où l'on ne dispose pas, actuellement, de trajectoire anti collision.

VI.2.1. prise d'une pièce sur le stock dynamique

Le tapis peut être schématisé de la façon suivante :

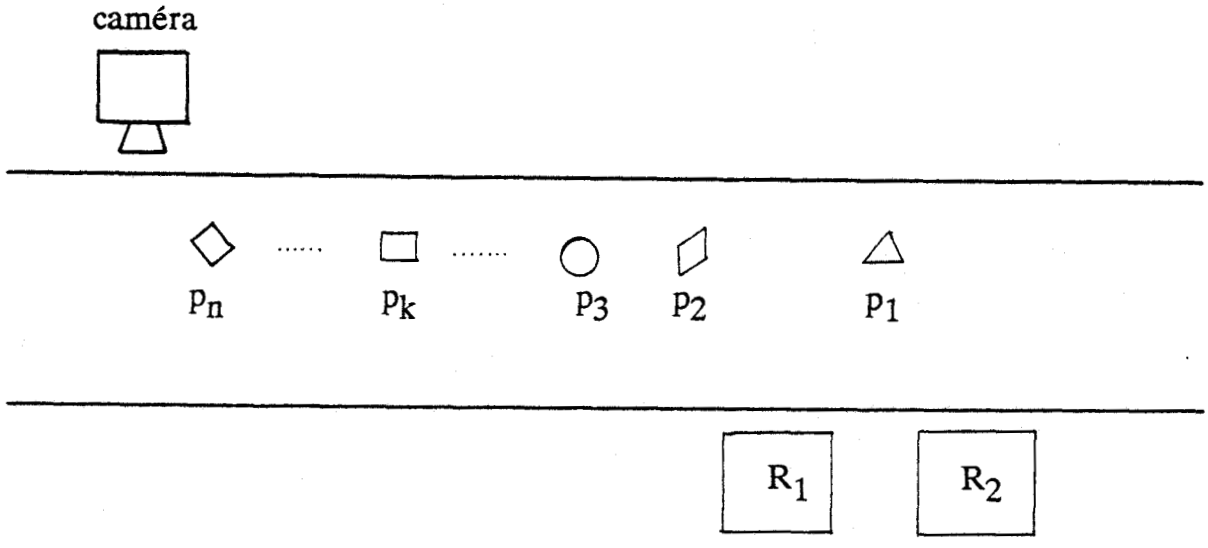


fig.III.2.

VI.2.1.1. cas d'une tâche

La date d'arrivée de la pièce p_k devant le robot qui la saisira est fonction :

- du temps de transfert Tt_k de sa position courante à la position de prise qui est fonction de la vitesse du tapis.

- des arrêts éventuels du tapis devant la caméra pour reconnaissance de la ou des pièces qui entrent dans la cellule.

Ce dernier temps ne peut être évalué puisque les entrées sont aléatoires.

Il affecte de la même valeur, l'ensemble des pièces situées sur le tapis, donc n'intervient pas dans la compa-raison de deux pièces appartenant au stock dynamique.

Le problème se pose lorsque l'on compare les temps de disponibilité d'une pièce en stock statique et d'une pièce en stock dynamique.

Dans ce cas, négliger les arrêts entraîne une sous estimation du temps de transport. Pour compenser ces arrêts, nous introduisons un facteur de correction α.

Ce coefficient est fonction d'une part du temps de transport d'une ressource de l'entrée de la cellule à un des robots, d'autre part du nombre de ressources présentes à un instant sur le tapis. En effet, chacune d'entre elles a provoqué un arrêt du tapis pour reconnaissance, le nombre moyen de ressources définit donc le temps moyen d'identification qui s'ajoute au temps de transport.

Dans ces conditions, nous pouvons calculer α, en prenant par exemple :

$$\alpha = 1 + \frac{\text{temps moyen d'identification}}{\text{temps moyen de transport}}.$$

Connaissant la position de la pièce, nous pouvons évaluer $\alpha\, Tt_K$. La date d'utilisation de la ressource Tu_K dépend de la disponibilité du robot j qui doit saisir cette ressource.

- s'il est libre, et que l'on néglige le temps de prise, la date d'utilisation Tu_k est estimée par $\alpha\, Tt_k$.

- s'il n'est pas disponible, le temps d'attente de la pièce devant le poste de prise dépend de la date de libération du robot, T_j.

Nous avons alors :

$$Tu_k = \max (\alpha Tt_k , T_j) \quad j = 1, 2$$

VI.2.1.2. cas d'un couple de tâches utilisant des ressources appartenant au stock dynamique.

Pour un couple de tâches nécessitant les ressources p_k et p_l pour leur réalisation, la date d'utilisation des ressources dépend :

- du temps de transfert Tt_k, Tt_l des ressources de leur position courante à leur poste de prise respectif.

- du temps d'arrêt du tapis devant l'un des robots pour la saisie d'une ressource.

Cet arrêt est causé par la première pièce qui arrive en position de prise c'est à dire celle dont le temps de transfert est égal à :

$$\min (Tt_k, Tt_l)$$

Soit p_k cette pièce.

La durée de l'arrêt dépend de la date de saisie par le robot de la ressource, qui est fonction de la disponibilité de celui-ci et correspond au temps d'attente de la ressource au poste de prise Ta_k.

Nous avons :

$Ta_k = T_j - Tt_k$ si $Tt_k < T_j$

$Ta_k = 0$ si $Tt_k > T_j$

Si le robot 1 utilise la ressource k,

$$Tu_k = \max (Tt_k , T_1)$$

et $$Tu_l = \max (Tt_l + Ta_k , T_2)$$

Connaissant les temps d'occupation T_1 et T_2 des robots 1 et 2 et la position courante des ressources, nous pouvons calculer les dates effectives d'utilisation des ressources.

VI.2.2. Date de prise d'une pièce en stock statique

Par définition, une pièce située en stock statique à un temps de transfert nul. Les conditions nécessaires à la saisie de cette pièce sont alors fonction de la disponibilité du robot et de l'accessibilité à la zone commune.

Dans ces conditions, nous avons :

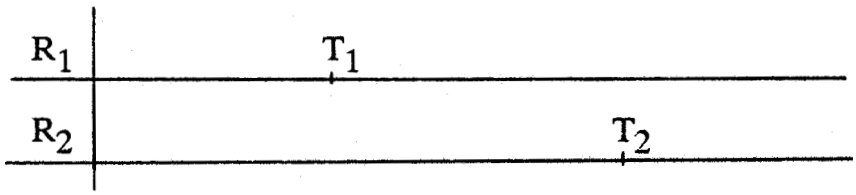
$$Tu_k = \max (T_j, T_v)$$

T_v date de libération de la zone commune.

Remarque : si les deux tâches d'un couple nécessitent une ressource en zone commune, ces dernières seront saisies séquentiellement. L'ordre est alors celui d'arrivée des robots dans la zone commune.

VI.3. Date de fin d'exécution d'antécédents

Soit la répartition suivante des temps :



Les prochaines tâches à effectuer commenceront au plus tôt à T_1 pour le robot 1 et à T_2 pour le robot 2.

Nous avons vu qu'il pouvait exister un temps d'attente pour l'arrivée d'une ressource ou pour accéder à la zone commune.

Dans la mesure où ces temps peuvent être supérieurs à T_2, on peut accepter des temps d'attente pour l'arrivée d'antécédents, c'est à dire attendre les tâches qui seront débloquées à T_2.

En fait, on pourrait attendre également le déblocage d'autres tâches par R_2, mais on considère qu'elle n'ont aucune chance d'être choisies, aussi nous ne tenons compte

que des tâches débloquées à T_1 et T_2. Aussi, nous avons :

$$i \in EXEC1(T_{max}), j \in EXEC2(T_{max}) \text{ et } T_{max} = \max(T_1, T_2)$$

La date de fin d'exécution des antécédents est alors égale à :

0	si tous les prédécesseurs sont terminés,
T_1	si 1 exécute le dernier antécédent,
T_2	si 2 exécute le dernier antécédent,
$\max(T_1, T_2)$	si les deux robots exécutent les deux derniers antécédents de la tâche.

VI.4. Date de début au plus tôt de tâche

La date de début au plus tôt d'une tâche nécessitant une ressource p_k et exécutée par le robot i est égale au maximum des temps précédemment cités.

VII. SELECTION D'UN COUPLE DE TACHES

A ce stade, nous savons calculer les différents paramètres utiles au calcul du gain résultant de l'exécution d'un couple de tâches (i, j).

Il existe néanmoins des contraintes à respecter pour la sélection d'un couple de tâches.

En effet, nous avons vu que la réalisation d'une tâche nécessite :

- au plus une ressource consommable (située en zone commune ou sur le tapis)
- un ou deux robots
- une plan d'assemblage propre à un robot ou en zone commune.

Les tâches (i,j) sont telles que :

$$i \in EXEC1(T_{max}), j \in EXEC2(T_{max}) \text{ et } T_{max} = \max(T_1, T_2)$$

Le premier impératif évident est $i \neq j$.

La contrainte suivante concerne les ressources consommables, les tâches i et j ne peuvent utiliser une même ressource présente en un seul exemplaire.

Pour les ressources partageables, il n'y a aucune condition particulière car la procédure d'affectation calcule les dates de début et fin de tâches en tenant compte des différentes données.

Ainsi lors de l'utilisation, par une tâche, de la zone commune, pour la prise d'une ressource ou la réalisation d'un assemblage, si l'autre tâche requiert la zone, elle attend que celle-ci soit libérée.

De la même façon, si une des tâches du couple demande la disponibilité des 2 robots, l'évaluation des dates de début prend en considération le fait que l'une des tâches sera forcément réalisée après l'autre.

Ces différentes conditions sont prises en charge pour la sélection des couples.

Ainsi, nous avons :

1 - $\forall i \in EXEC1(T_{max}), \forall j \in EXEC2(T_{max})$ $T_{max} = \max(T_1, T_2)$

Déterminer l'ensemble C de couples (i,j), i compatible avec j,

2 - Pour chacun des couples obtenus, calculer R(i,j),

3 - Sélectionner sur C, le "meilleur couple" soit (i^*,j^*), défini par : (i^*,j^*) apporte un gain $R^*(i,j)$ tel que

$$R^*(i,j) = \max_{(i,j) \in C} R(i,j)$$

4 - $ATT(t) = \{ i^*, j^* \}$

Une affectation de couple est effectuée à chaque entrée d'une pièce dans le stock dynamique, si cette ressource modifie l'ensemble des tâches exécutables. En fait, même s'il n'y a pas de nouvelles tâches exécutables, le calcul est repris puisque les gains R(i,j) ($(i,j) \in C$) peuvent être modifiés.

A l'instant $T_{min} = \min (T_1, T_2)$ où le module de lancement affecte une tâche au robot libre, l'algorithme de calcul est également relancé. Les couples sont alors sélectionnés parmi l'ensemble :

$$(EXEC1(T'_{max}),EXEC2(T'_{max})) \text{ avec } T'_{max} = \max (T'_1, T'_2),$$

T'_1 et T'_2 correspondant aux nouvelles dates de libération des robots. En fait, nous avons $T'_1 = T_1$ ou $T'_2 = T_2$.

VIII. CONCLUSION

Le module de décision permet le choix du meilleur couple de tâches à réaliser à la libération des robots. Le calcul prend en compte l'arrivée dynamique des ressources. Le critère retenu pour le "meilleur couple" est basé sur la maximisation de la quantité de travail présent et futur consécutive à l'exécution du couple de tâches.

Le calcul proposé s'applique à une description des tâches à effectuer sous forme d'un graphe. L'application à une structure arborescente nécessite un regroupement des séquences possibles sous forme de graphes.

L'annexe 2 présente des exemples d'application de la procédure de décision.

CHAPITRE 4

ARCHITECTURE FONCTIONNELLE

I INTRODUCTION

Le module de décision, élément moteur du système de pilotage, étant élaboré, les phases suivantes sont la conception et la réalisation de ce système. La première nécessité était la modification de l'interpréteur LM (langage de manipulation) puisque celui dont nous disposions pour la commande des deux robots ne permettait qu'un déroulement séquentiel des programmes, même si les robots se déplaçaient en parallèle. Un bi-interpréteur LM a donc été écrit (JOL 88), son implantation est en cours.

Dans ce chapitre, nous présentons la structure fonction-nelle du système de pilotage, qui sera prochainement installé.

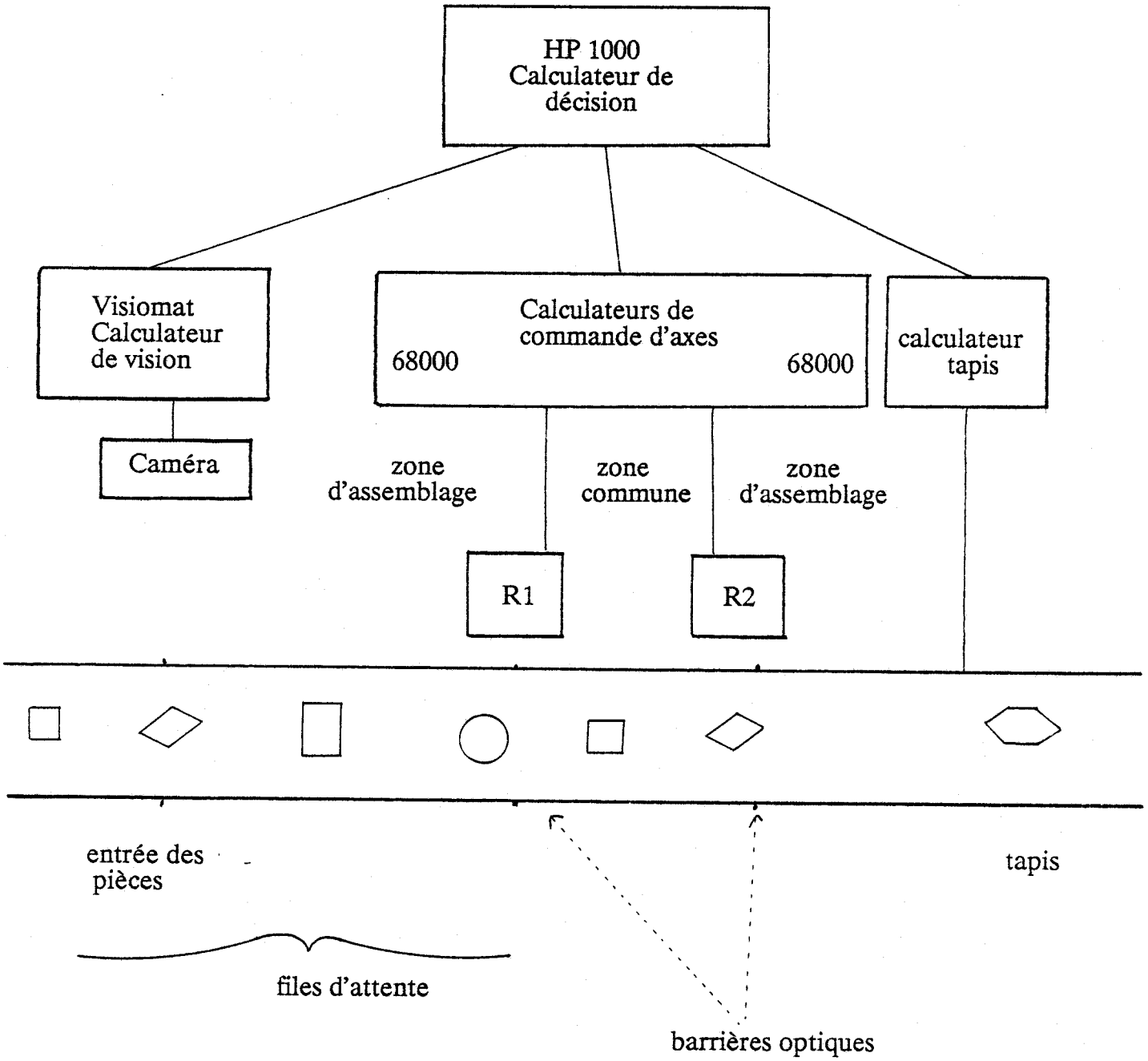
II DESCRIPTION MATERIELLE

La cellule expérimentale de montage est composée :
- de deux robots 4 axes SCEMI,
- d'un système de vision (Visiomat),
- d'un tapis roulant.

C'est un des éléments de l'atelier flexible expérimental implanté au centre d'Automatique à l'U.S.T.L. Il est piloté par un calculateur HP 1000.

La caméra est associée à un calculateur de traitement d'images (Calculateur Vision) et le tapis possède son propre calculateur de gestion et contrôle (Calculateur Tapis). Ils sont tous deux reliés par liaison série au superviseur (HP 1000) qui, par ses possibilités multi-tâches, assure la coordination de l'ensemble ainsi que le calcul des trajectoires des robots. Deux liaisons à haute vitesse lui permettent de transmettre les paramètres de commande des axes à deux calculateurs spécialisés construits autour de microprocesseurs 68000. La figure IV.1. décrit la structure matérielle de la cellule.

Calculateur de gestion d'atelier



Architecture matérielle fig. IV.1.

III STRUCTURE DU SYSTEME DE CONDUITE (STA(a) 85), (STA(b) 87)

Le système de conduite est composé de différents éléments affectés chacun à une fonction : information, décision, exécution, communication.

Il est organisé de façon modulaire et chaque système est activé selon les événements intervenant dans la cellule. Pour chacun de ces systèmes, nous allons présenter les différents modules qui le composent avant de détailler les fonctions qu'ils réalisent ainsi que les conditions d'activation.

Le système d'information comprend les trois modules de mise à jour correspondant respectivement aux robots, aux assemblages, aux stocks, le module d'identification et ceux de contrôle et diagnostic.

Le système de décision a pour rôle le choix des opérations à lancer et de ce fait, contient le module d'allocation de tâches et celui de lancement.

Le système d'exécution permet l'interfaçage avec les systèmes physiques, on y trouve la fonction transport et la commande des robots.

L'interfaçage avec le monde extérieur réalisé via un opérateur ou via d'autres calculateurs est assuré par le système de communication.

III.1 Le système d'information

La gestion en temps réel du système nécessite une connaissance exacte, à tout instant, des informations utiles au système de décision. Celles-ci sont fournies par le système d'information qui a pour rôle l'acquisition, le traitement et le stockage des données nécessaires au fonctionnement de la cellule de production. Ces différentes fonctions sont assurées par les modules détaillés ci-dessous.

III.1.1. Modules de mise à jour

Ils assurent l'actualisation de l'état des robots, de l'état des assemblages et de l'état des stocks statique et dynamique.

La mise à jour des données se fait selon le principe suivant :

III.1.1.1. Etat des robots

Nous avons vu au chapitre II que les différents états dans lequel un robot peut se trouver sont libre, occupé, indisponible.

A tout instant, chaque robot est défini par son état et par la liste de ses aptitudes. La mise à jour de ces états est alors effectuée à l'occurrence de certains événements qui provoquent un changement, à savoir :

Informations	générées par	transitions
début de tâche	module lancement	libre --> occupé
fin de tâche correcte	module d'exécution module de contrôle	occupé --> libre
fin d'indisponibilité	opérateur	indisponible -->libre
défaut, panne	module diagnostic	occupé ->indisponible
maintenance	opérateur	libre -> indisponible

L'actualisation de la liste des aptitudes est, quant à elle, effectuée si nécessaire par le module de diagnostic après une panne ou par l'opérateur après maintenance.

III.1.1.2. Etat des assemblages

Le lancement d'une production suppose la gamme de fabrication connue de la cellule. Celle-ci est décrite par un graphe de précédence définissant les opérations à effectuer ainsi que leur enchaînement.

L'état du produit à réaliser est décrit par l'état des opérations qui le composent. La mise à jour de l'état des assemblages est donc exécutée à chaque fois qu'une tâche change d'état, à savoir :

< Fonctionnement normal >

Informations	générées par	transitions
lancement de production	opérateur	bloqué --> non exécutable
fin correcte de tous les antécédents	module d'exécution module de contrôle	non exécutable --> pré-exécutable (a)
entrée de la ressource	module d'identification	non exécutable --> pré-exécutable (r)
sortie d'une ressource	module de décision	exécutable --> non exécutable
sélection de la tâche	module de décision	exécutable --> en attente
changement de décision	module de décision	en attente --> exécutable
début de tâche	module lancement	en attente --> en cours
fin de tâche correcte	module d'exécution module de contrôle	en cours --> terminé
assemblage fini	opérateur	terminé --> bloqué

< mode défaillant >

Informations	générées par	transitions
conditions de poursuite non remplies	module de contrôle	en cours --> suspendu
conditions de poursuite vérifiées	module diagnostic	suspendu --> en cours
poursuite impossible	module diagnostic	suspendu --> non réalisable
fin de tâche incorrecte	module diagnostic	en cours --> non réalisable
ré-exécution après incident	module diagnostic	non réalisable --> non exécutable
défaut "fatal"	opérateur	non réalisable --> bloqué

Nous pouvons remarquer sur ce tableau que certains événements qui entraînent une mise à jour de l'état des assemblages, intervenaient déjà dans la mise à jour des robots (exemples : début de tâche, fin de tâche, panne, ...).

III.1.1.3. Etat de l'environnement

L'environnement comprend le stock statique, commun aux deux robots et le stock dynamique. L'état de ces stocks est défini par la liste des informations relatives aux pièces qu'ils contiennent. Pour chacune d'entre elles, en plus d'un identificateur, on dispose de paramètres de position et d'orientation qui sont élaborés et fournis par le système de vision.

Ces listes sont mises à jour à l'occurrence des événements suivants :

Evénements	transmis par	Stock concerné
entrée d'une pièce	module d'identification	dynamique
	module d'exécution module de contrôle	statique
sortie d'une pièce	module de décision	dynamique
utilisation d'une ressource	module d'exécution module de contrôle	dynamique statique
incident	module de contrôle et/ou opérateur	dynamique statique

Le stock statique est accessible indifféremment à l'un quelconque des robots de la cellule. Nous avons vu que le système de décision tenait compte du fait qu'actuellement nous ne disposons pas de trajectoires anti-collision pour la sélection des tâches. Il est cependant nécessaire de gérer l'accès à cette zone, ceci est fait par l'utilisation d'un sémaphore d'exclusion mutuelle (CRO 79), (LEM 84) dont le fonctionnement sera exposé par la suite.

Les différents modules de mise à jour que nous venons de présenter sont initialisés par l'opérateur ou un autre calculateur grâce au système de communication.

III.1.2. Module d'identification

Son rôle est de fournir à la fonction mise à jour du stock dynamique les informations concernant le type et l'orientation de la pièce identifiée.

La reconnaissance est activée par le calculateur tapis qui stoppe la fonction transport lorsqu'il a détecté une pièce entrant dans la cellule.

L'événement "fin de reconnaissance" sollicite le système d'information pour mise à jour du stock dynamique. Une lecture des paramètres élaborés par la fonction reconnaissance est alors effectuée, ainsi qu'une lecture du contenu du compteur du calculateur tapis de manière à connaître de façon précise la position de la ressource.

Lorsque l'échange d'informations est terminé, un signal "fin de transmission" autorise la relance du tapis.

La figure IV.2. montre les différents échanges de la fonction reconnaissance avec la fonction transport et le système d'information.

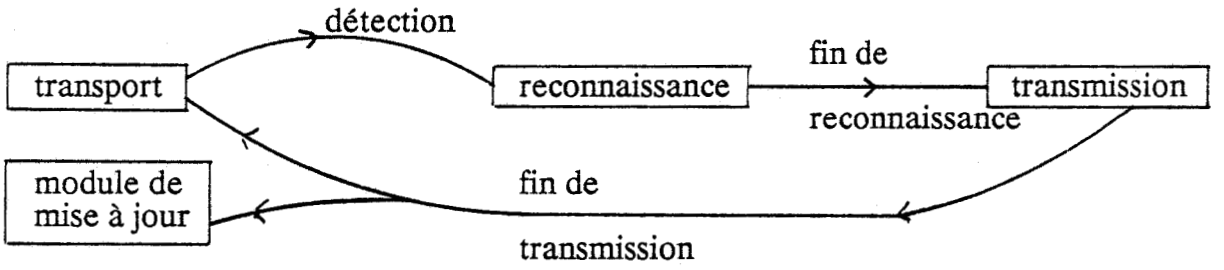


fig. IV.2.

III.1.3. Modules contrôle et diagnostic

Pour effectuer un suivi en temps réel de l'évolution de la cellule, deux modules seront développés et associés au module d'information. Il s'agit du module de contrôle qui permet à partir des capteurs de vérifier la bonne exécution des opérations et du module de diagnostic.

Le contrôle est activé à chaque début de tâche par le module lancement qui lui fournit les données concernant la tâche qui va être exécutée.

Il est chargé de transmettre un bilan de fin d'opération : au module de mise à jour quelle que soit l'exécution (correcte ou non) et au module diagnostic dans le cas d'une exécution incorrecte.

Cette détection peut être faite par exemple, par tout ou rien, avec la comparaison de la durée moyenne d'une tâche à celle de la tâche en cours, par surveillance d'exécution à l'aide d'un chien de garde, par analyse de signaux, par des tests sur les données fournies par différents capteurs, ...

En fait, plusieurs étapes sont nécessaires :

- la détection du défaut,
- l'isolation de la défaillance c'est à dire trouver l'endroit de la panne,
- le diagnostic intervient ensuite pour expliquer les causes du défaut,
- Enfin, une solution pour remédier au problème doit être proposée.

Les méthodes utilisées pour la surveillance des pannes de processus font appel aux procédures de tests, aux techniques de traitement du signal et de la reconnaissance des formes (EAG 88). Si pour certaines défaillances (bris d'outils, absence de pièce,...), une solution peut être élaborée (par exemple, par modifications du déroulement d'un programme), le problème est plus difficile lorsque le défaut (par exemple, la chute d'un objet) a causé des dommages aux autres constituants de l'environnement, (LOZ(b) 83), (LOZ 76), (COR 79).

Le module diagnostic devrait être un module "expert" qui suivant les informations qui lui sont transmises par le module contrôle, est capable de donner la marche à suivre pour éventuellement remédier à une défaillance. Dans le cas d'une exécution incorrecte, si l'action ne peut être poursuivie, un appel à l'opérateur est réalisé. Un exemple de module travaillant sur une base de connaissance est proposé dans (LOP 86).

III.2. Le système de décision

Compte tenu des informations qui lui sont transmises par le système d'information, il a pour rôle :

- de sélectionner le couple de tâches à affecter aux robots lorsqu'il deviendront libre ; ceci est effectué par le module d'allocation de tâches.

- de lancer la réalisation des tâches par le système d'exécution, ceci est le rôle du module de lancement.

III.2.1. Module d'allocation de tâches

Ce module exécute la procédure que nous avons développée dans cette étude. A partir de la liste des tâches exécutables, il sélectionne le couple de tâches à affecter aux

robots à leur libération, selon l'état courant de la cellule qu'il connait par l'intermédiaire des modules de mise à jour.

Il détermine les deux prochaines tâches à affecter aux robots, c'est à dire l'ensemble ATT.

Il est activé lors de certaines modifications de la liste des tâches exécutables, à savoir :

- lorsqu'une pièce entre dans la cellule,
- lorsqu'une tâche est terminée.

Cette liste est par ailleurs modifiée par les décisions suivantes :

- sélection d'une tâche,
- sortie d'une ressource.

III.2.1.1. Activation par entrée d'une ressource consommable

Plusieurs cas sont à analyser :

- 1. Si cet événement intervient durant un calcul du système de décision, et que ATT est vide, le calcul est interrompu puis repris avec les données actualisées. Simultanément, lorsque la transmission d'informations est terminée, le tapis avance jusqu'à ce qu'une ressource soit en position de prise devant l'un des robots.

Il est alors possible que durant l'avance du tapis, une autre ressource entre, provoquant ainsi une nouvelle interruption du calcul. En fait, nous verrons que quand aucune décision n'a été prise, le tapis s'arrête lorsqu'une ressource est en position de prise devant un robot, aussi nous sommes sûr que le calcul se terminera au plus tard à ce moment-là.

- 2. Si l'entrée d'une pièce se produit alors qu'un ensemble ATT a été sélectionné.

Les tâches affectées à l'ensemble ATT peuvent être remises en cause.

L'arrivée d'une nouvelle ressource modifiant la base de données, un nouveau calcul est effectué. Deux solutions sont alors possibles :

- soit le calcul confirme le choix précédent,

- soit il infirme ce choix.

Dans le premier cas, nous ne gagnons rien et nous perdons, au pire, le temps de calcul.

Dans le second cas, nous trouvons une solution qui est meilleure que la précédente, aussi nous la retenons. On notera que même si la situation se réitère, c'est à dire qu'une autre ressource arrive, nous n'aboutirons jamais à une situation de blocage compte tenu du fait que la fonction bénéfice est une fonction croissante et bornée.

III.2.1.2. Activation par fin de tâche

L'occurrence de cet événement entraîne plusieurs possibilités selon la manière dont la tâche a été exécutée. Ainsi, nous distinguerons l'exécution correcte de l'exécution incorrecte.

Dans le premier cas :

- si une sélection est en cours de calcul (ATT est vide), celui-ci n'est pas interrompu ; en effet, l'algorithme de décision prend en compte l'état des assemblages tel qu'il sera à la libération des robots et de ce fait, la fin correcte d'une tâche ne modifie pas la base de données.

- sinon, on affecte une des tâches de ATT au robot libre, c'est à dire à ENC et on exécute un nouveau calcul.

Dans le cas d'une exécution incorrecte :

- qu'il y ait un calcul en cours ou non, on doit nécessairement refaire une affectation puisque les estimations de l'état de la cellule ne sont pas validées. Si un calcul est en cours, il sera donc interrompu.

On remarquera que selon les situations, il est possible que sur les deux tâches sélectionnées par le système de décision à un instant donné, une seule soit réalisée puisqu'une nouvelle affectation peut modifier l'autre.

III.2.2. La fonction lancement

Le rôle de la fonction lancement est de transmettre aux modules d'exécution et de contrôle, la tâche à réaliser par la ressource partageable libérée.

Pour les robots, ceci correspond au passage des tâches de ATT à l'ensemble ENC. Si une affectation est en cours de calcul, le robot reste libre.

Cette fonction fournit les informations concernant la ressource consommable requise (situation (stock statique ou stock dynamique), position, orientation), ainsi que celles relatives à l'exécution de la tâche.

Elle est activée par un signal de fin d'activité d'un robot émis par les modules d'exécution et de contrôle.

Pour le tapis d'alimentation, ceci correspond à un nombre de pas à effectuer. IL est activé par un signal de fin de saisie d'une ressource en provenance du module de commande des robots.

III.3. L'exécution

III.3.1. La fonction transport

Son rôle est d'une part, la commande du tapis qui est utilisé pour le transport des pièces à l'intérieur de la cellule compte tenu des consignes attribuées par le système de décision ; d'autre part, l'incrémentation d'un compteur qui permet au module de mise à jour d'actualiser la position des différentes pièces présentes sur le tapis.

Le système de décision fournit à la fonction transport la tâche qu'il doit exécuter c'est à dire le nombre de pas à effectuer :

- pour que la pièce nécessaire à une tâche de ATT ou ENC se trouve en position de prise devant le robot qui réalisera la tâche,

- ou si ATT est vide et que les tâches de ENC ont leurs ressources, pour que la pièce la plus proche d'un des robots se trouve en position de prise.

Dans ces conditions, la tâche de fond de la fonction transport est la commande d'avance du tapis d'un nombre donné de pas. Lorsque cet objectif est atteint, le tapis est arrêté, la fonction transport attend un nouvel ordre.

Cette tâche est interrompue lorsqu'une ressource est détectée à l'entrée de la cellule. Un signal est alors envoyé pour activer la fonction reconnaissance. La réception de l'information "fin de transmission" (après envoi des paramètres d'identification et lecture du compteur de position du calculateur tapis) entraîne la commande d'avance du tapis jusqu'à ce qu'une pièce soit devant un des robots. Puis le tapis est arrêté et attend un nouvel objectif qui selon le cas sera celui qui était en cours avant l'interruption ou un autre si une nouvelle décision a été effectuée.

Le tapis est équipé de cellules de détection situées à l'entrée de la cellule, pour arrêter le transport, et devant la position de prise de chacun des robots. Dans ce dernier cas, elles sont utilisées pour contrôler l'arrêt à l'endroit prévu.

La figure IV.3. représente les échanges entre la fonction transport et les autres modules.

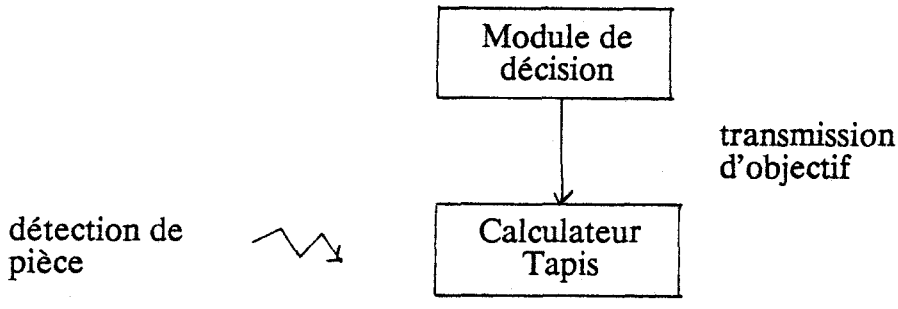


fig. IV.3.

Une autre approche a été développée par R. FERHATI (FER 86) dans laquelle une zone atteignable pour chacun des robots était définie, de manière à limiter l'attente du robot pour la prise d'une ressource (fig. IV.4.).

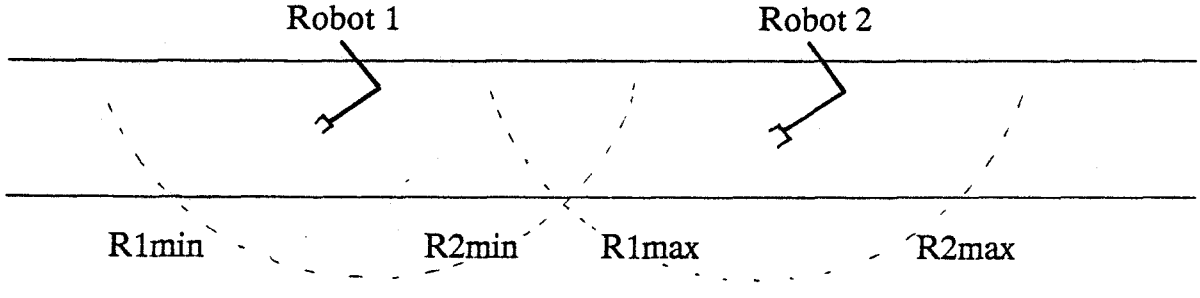


fig. IV.4.

Ainsi, trois cas sont à considérer :

- si le robot I se libère et que la ressource n'est pas dans sa zone atteignable, l'avance du tapis continue jusqu'à ce que la pièce soit en RImin,

- si le robot se libère et que la ressource est dans sa zone, le tapis est arrêté,

- lorsque la ressource requise arrive en RImax et que le robot I n'est toujours pas libre, l'arrêt du tapis est commandé.

Cette approche séduisante n'a pas été retenue dans notre étude car elle exige de nombreux calculs au niveau de la fonction transport mais également des échanges importants d'informations entre cette fonction et le module de décision.

On peut cependant remarquer qu'une extension de cette démarche pourrait amener à déterminer les zones optimales d'arrêts, à savoir celles qui permettent la prise des pièces d'un couple de tâches en minimisant l'attente des robots (ABB 87).

III.3.2. Le module de commande des robots

III.3.2.1. Le langage LM

Classé dans "les langages dits de haut niveau informatique" dans le panorama présenté dans (HAU 87), le LM est un langage spécialisé dans la commande des robots, qui autorise une programmation structurée voisine du PASCAL (LOZ(a) 83).

La structure géométrique de l'environnement est représentée en coordonnées cartésiennes. Ceci permet de décrire facilement la position et l'orientation des objets ou de l'outil du robot. Les déplacements sont spécifiés en termes de transformations entre le repère lié à l'outil et celui associé à la position finale.

C'est un langage compilé puis interprété qui était prévu à l'origine pour gérer un seul robot.

Dans le cadre de notre application, la société ITMI qui commercialise le système a modifié le LM afin d'y intégrer la gestion des deux robots. Cependant, les programmes étaient exécutés de manière séquentielle ; l'option "sansattente" qui permet de continuer

l'exécution du programme sans attendre la fin du déplacement en cours, permettait un pseudo-parallélisme au niveau des mouvements des deux robots.

La structure de l'interpréteur est la suivante :

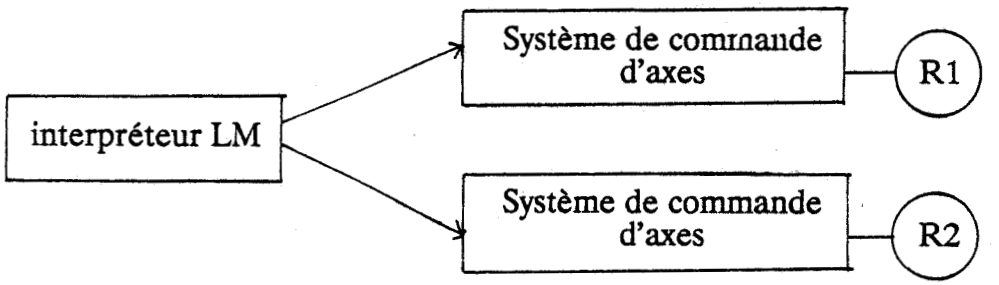


fig. IV.5.

III.3.2.2. Le bi-interpréteur (JOL 88), (KAC 88)

Pour le pilotage de la cellule robotisée, il était nécessaire de développer une nouvelle structure d'exécution qui supporterait le déroulement de programme en parallèle, et par voie de conséquence une réelle coopération entre les deux robots.

Pour résoudre ce type de problème, il était possible de créer un nouveau langage ou de modifier la structure interne de l'interpréteur LM. La solution retenue par J.P. GERVAL avec LCOOP, langage de coopération (GER 87), possède l'inconvénient d'associer un calculateur par robot. En collaboration avec la société ITMI, la solution du bi-interpréteur a été adoptée, avec la contrainte d'une entière compatibilité avec l'interpréteur existant. Les deux robots sont alors commandés à partir d'un même calculateur multi-tâches.

La structure du bi-interpréteur est la suivante :

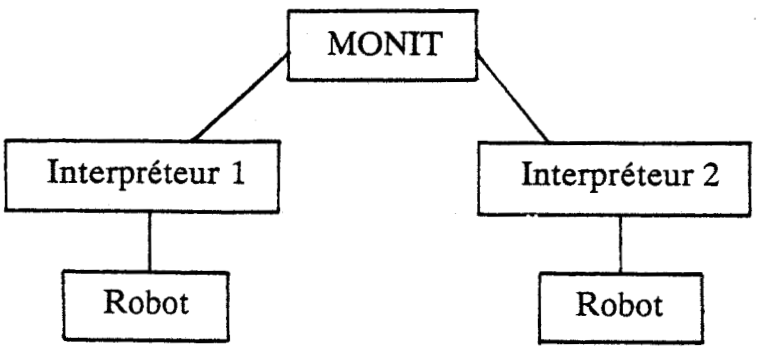


fig. IV.6.

Le rôle du module MONIT est de lancer l'exécution de l'interpréteur LM. Il est chargé de toutes les initialisations des variables partageables et de l'activation d'autres modules utilisés pour gérer les déplacements, les échanges d'informations avec le système de commande d'axes et la commande manuelle. Pour différencier les deux interpréteurs, les données sont dédoublées, le module MONIT affecte les programmes LM vers l'un ou l'autre des interpréteurs.

Des processus de communication et de synchronisation entre tâches ont été définis pour permettre la coopération entre les robots.

III.3.2.3. Couplage du bi-interpréteur avec le module de lancement

Chaque tâche est décrite par une séquence d'opérations élémentaires, il lui correspond une procédure paramétrée écrite en LM et qui est exécutée par l'un ou l'autre des interpréteurs selon le robot qui réalise la tâche.

Le couplage du bi-interpréteur avec le module de lancement permet la transformation de la tâche choisie par le système de décision en une liste ordonnée d'opérations élémentaires avec les paramètres adéquats.

III.3.2.4. Exécution d'une tâche

Lorsque la fonction lancement a transmis les paramètres de la tâche sélectionnée au module de commande des robots, celui-ci charge dans la pile le code qui doit être interprété et active l'interpréteur correspondant au robot qui réalise cette tâche. Les trajectoires sont alors élaborées et les paramètres de mouvement transmis au système de contrôle des axes.

Cette action se déroule tant que le module de contrôle ne décèle aucune anomalie. La détection d'un défaut arrête l'interprétation de la tâche en cours et active la fonction diagnostic qui, suivant le cas, autorise la poursuite de l'assemblage, en sollicitant la mise à jour de l'état, ou fait appel à l'opérateur, pour une intervention, par l'intermédiaire de l'interface Homme/Machine.

Lors de l'exécution d'une tâche qui utilise une pièce sur le stock dynamique (tapis), une primitive de synchronisation avec la fonction transport permet au module d'être informé de la présence, devant le robot libre, de la ressource à saisir. La tâche est alors exécutée. Dés que la prise de la ressource est terminée, un signal est envoyé au système de décision qui attribue alors un nouvel objectif à la fonction transport.

La figure IV.7. représente les échanges entre la fonction transport et les autres modules.

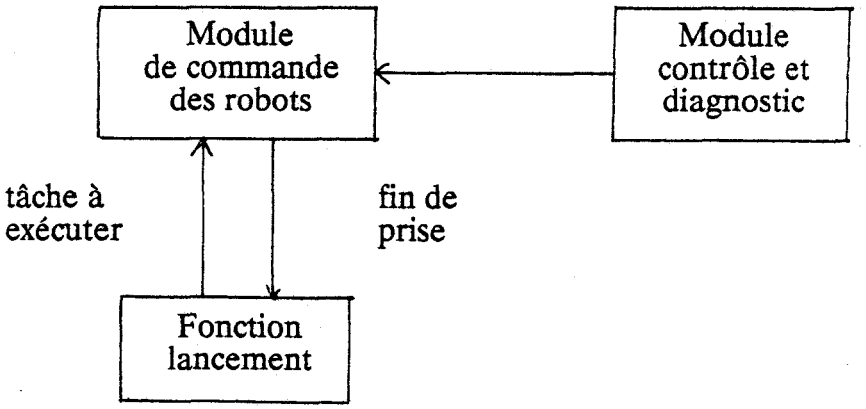


fig. IV.7.

Lorsque la zone commune est utilisée que ce soit pour une prise de pièce ou pour l'exécution d'une tâche, il est nécessaire de s'assurer qu'un seul robot est présent dans cette zone, puisque nous ne disposons pas, actuellement, de trajectoires anti-collision.

La solution retenue est l'utilisation d'un sémaphore d'exclusion mutuelle qui interdit l'accès simultané des deux robots dans cette zone (CRO 79), (LIS 83).

Un sémaphore est constitué d'une variable entière S et d'une file d'attente $f(S)$. La variable entière peut prendre des valeurs positives, négatives ou nulles.

Dans le cas de l'exclusion mutuelle, le sémaphore mutex est initialisé à 1. On peut agir sur lui grâce à deux primitives qui sont des opérations indivisibles.

P(mutex) "opération de demande de la zone"
S := S - 1
si S < 0 alors
blocage du processus par manque de ressource, mise en attente dans la file f(S).
sinon
attribution de la ressource au processus.
fin;

V(mutex) "opération de libération de la zone"
S:= S + 1
si S $\leq$ 0 alors
il y a au moins un processus dans la file f(S).
sortir un processus et lui attribuer la ressource.
fin;

Les programmes de déplacement de robot ont alors la structure suivante :

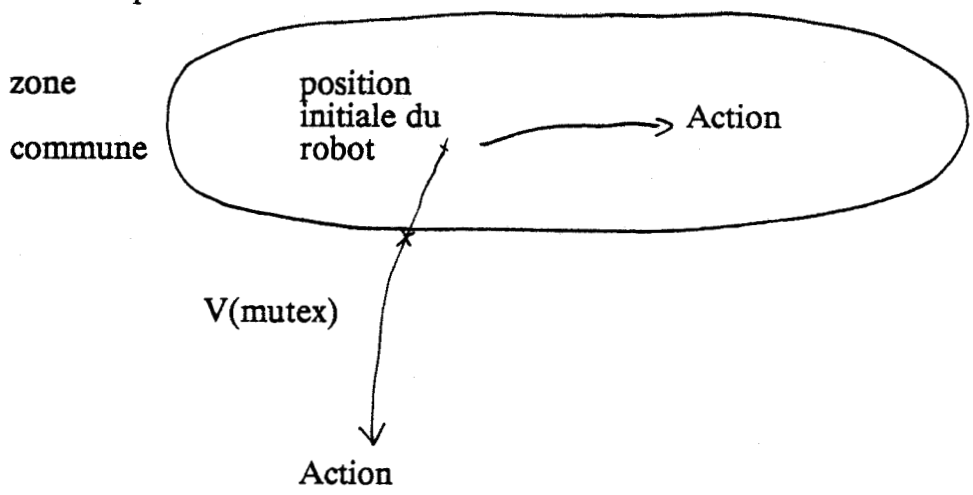
```
début
calcul de la position à atteindre;
        si cette position est dans la zone commune;
                début
                si le robot n'est pas dans la zone commune;
                        début
                        calcul de la position d'entrée dans la zone;
                        déplacer robot à cette position;
                        P(mutex);
                        fin;
                fin;
        sinon; (position hors de la zone commune)
                début
                si le robot est dans la zone commune;
                        début
                        déplacer robot à position sortie de la zone;
                        V(mutex);
                        fin;
                fin;
        déplacer robot à position à atteindre;
        action;
        fin;
fin;
```

Ainsi si la zone commune est occupée (S = O) à la suite de P(mutex), le processus va se bloquer sur la primitive P(mutex). Il la franchit lorsque le sémaphore repasse à 1 sous l'effet de l'exécution de V(mutex) par le processus concurrent.

Dans le cas 2, si la zone commune est occupée, le robot pourrait rester en position initiale. Pour gagner du temps, il se déplace jusqu'à l'entrée de la zone où il attend la libération de la ressource partageable, tout en restant dans son périmètre de sécurité.

La figure ci-dessous représente le mécanisme :

(1)



(2)

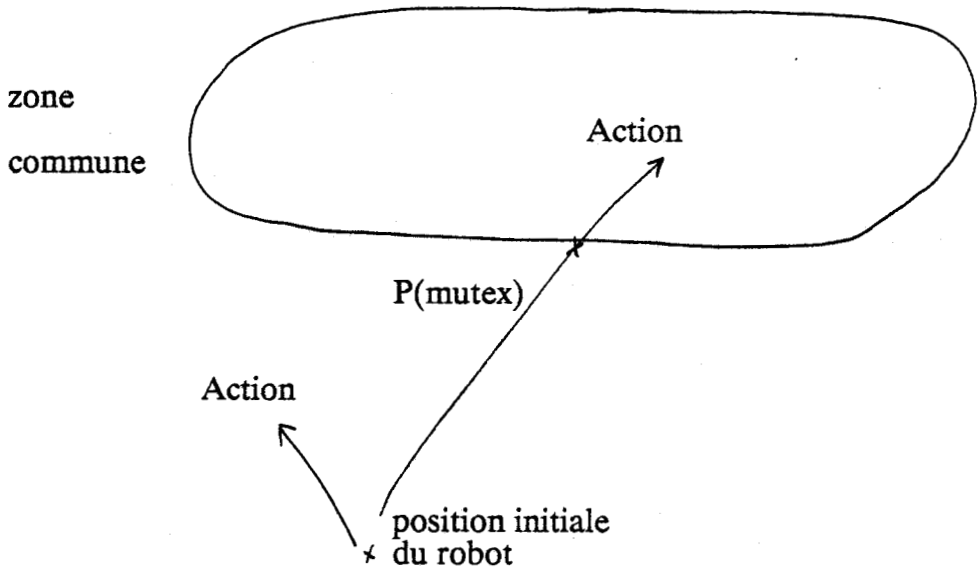


fig. IV.8.

III.4. Le système de communication

Le système de communication permet l'échange d'informations avec le calculateur de gestion d'atelier. Celui-ci contrôle l'ensemble du plan de production de

l'atelier et envoie à chacune des cellules les informations nécessaires à l'initialisation du superviseur lors du lancement d'une production, ce sont, par exemple, les paramètres suivants :

- graphes d'assemblage,

- nombre de produits à réaliser,

- identification des ressources associées à chaque tâche,

- ...

D'autre part une communication Homme/Machine permet le dialogue avec l'opérateur qui assure la conduite du système. Les informations sur l'état de la cellule, fournies par le calculateur, lui permettent une analyse de la situation, l'appréciation de son évolution et le choix des stratégie qui en résultent. Ainsi, par exemple, sur une requête du module diagnostic, lors d'une panne robot, il a la possibilité de ré-initialiser la cellule dans le cas d'un défaut "fatal" ou de décider de continuer la production selon un mode dégradé en faisant une mise à jour de la liste des aptitudes des robots.

Des renseignements journaliers, hebdomadaires, ... sont également communiqués pour assurer la gestion de production et pour faciliter la maintenance du matériel.

IV ARCHITECTURE FONCTIONNELLE

La figure IV.9. définit les différentes fonctions mises en oeuvre au niveau de la cellule ainsi que leurs interconnexions.

L'architecture présentée fait apparaître les trois systèmes qui composent une "Activité de Pilotage" telle que définie dans (BER 83), à savoir :

" Un système de décision constitué d'un ensemble d'activités de décision,

- Un système d'exécution constitué d'un ensemble d'activités d'exécution et représentant le système physique piloté par le système de décision,

- Un système d'information constitué de deux catégories d'informations : les informations internes et les informations externes."

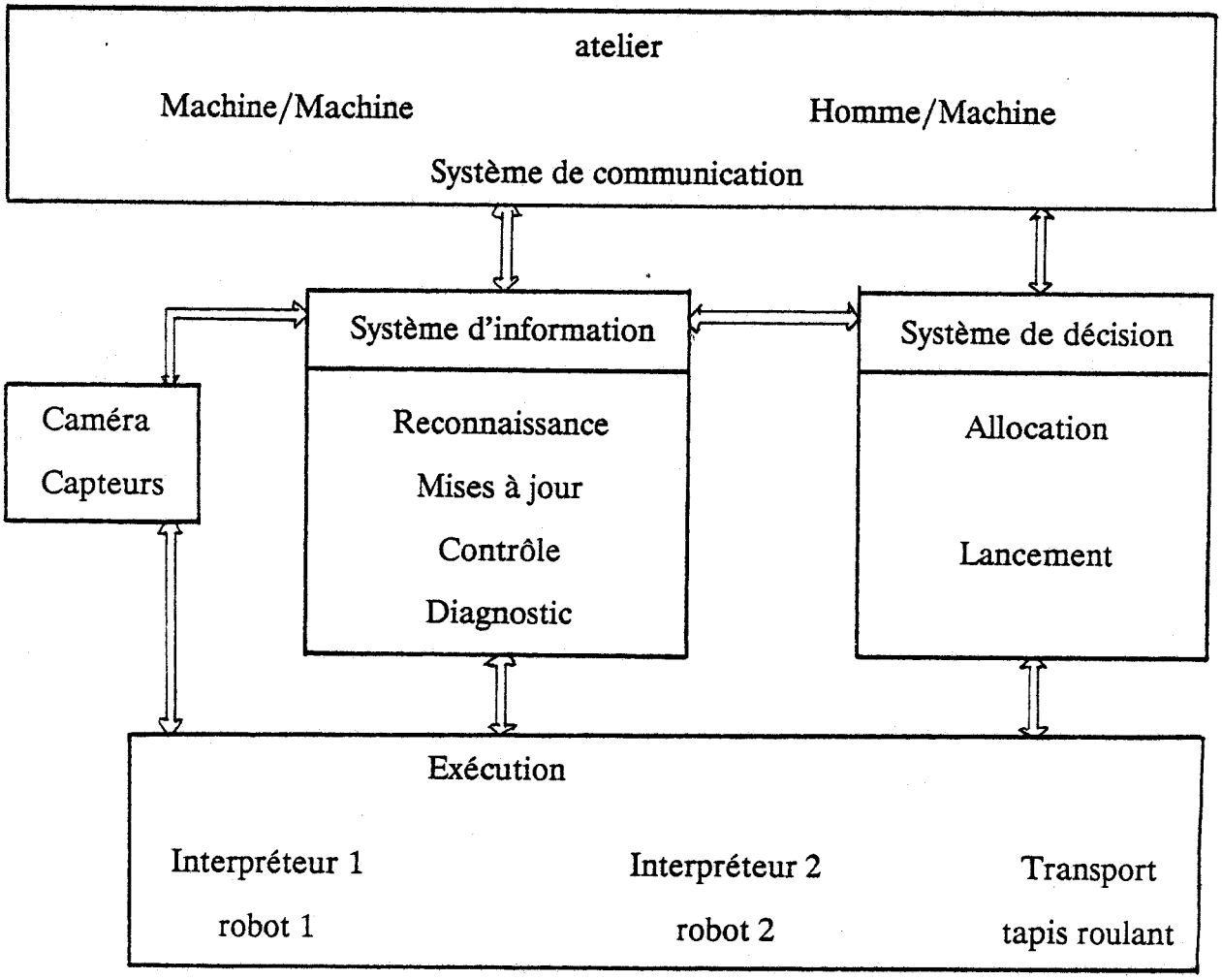


fig. IV.9.

Le module de communication est alors l'interface entre le module de décision et son environnement.

L'architecture est hiérarchisée et distribuée (DUP 86). Le système de pilotage qui contrôle la cellule est organisé de façon modulaire. Le niveau supérieur contient le superviseur qui comprend les sous-systèmes d'information, de décision et de communication. Le niveau inférieur est composé des contrôleurs du système de vision, du tapis et des robots.

V. CONCLUSION

A ce stade, les différentes fonctions ainsi que les échanges d'informations sont définis ; ce qui constitue une première étape dans la conception du système de pilotage.

En effet, la réalisation d'un projet passe par différentes phases qui vont des spécifications fonctionnelles, opératoires et technologiques, à la conception logicielle et matérielle.

Il existe peu de méthodes qui prennent en compte l'ensemble des étapes nécessaires à la réalisation logicielle et matérielle.

Cependant, nous pouvons citer la méthodologie de conception pour les applications de commande des processus complexes en temps réel développée par J.P. CALVEZ (CAL 82).

Il propose une approche constructive en cinq niveaux :

1 - Niveau de la description externe : correspondant aux spécifications de l'application.

2 - Niveau de la description fonctionnelle correspondant à la topologie de l'application (constituants et interconnexions).

3 - Niveau de la description opératoire : obtenue en ajoutant au niveau précédent la description algorithmique de chaque fonction indépendamment de toute technologie.

4 - Niveau de la description exécutive correspondant à la structure exécutive et à l'intégration de la description opératoire.

5 - Niveau de la réalisation, c'est à dire la description complète de la réalisation (matériel et logiciel).

Dans ce chapitre, nous nous sommes placés au niveau fonctionnel de la méthodologie proposée dans (CAL 82). Les différents modules qui composent le système de pilotage ont été présentés ainsi que les fonctions qu'ils réalisent. Les étapes suivantes qui permettront l'implantation définitive du système sont actuellement étudiées.

CONCLUSION

GENERALE

Le travail présenté concerne la conception d'un système de pilotage Temps Réel d'une cellule flexible d'assemblage dans laquelle deux robots coopèrent.

A partir des gammes et quantités d'assemblage, fournies par un opérateur ou un calculateur, le système proposé établit, selon l'état de l'environnement, l'ordonnancement des tâches effectuées par les robots en minimisant le temps de réalisation des assemblages

Une première phase du travail a consisté à modéliser les différents éléments de la cellule, à partir de graphes d'états. Ceux-ci représentent le comportement dynamique du système et rassemblent les éléments nécessaires au module d'information.

Dans une deuxième phase, nous avons développé une procédure permettant l'ordonnancement dynamique des tâches à effectuer. Celui-ci prend les décisions, chaque fois que survient un événement afin de satisfaire les objectifs de production et la minimisation du temps de réalisation d'un assemblage.

La notion de quantité de travail apporté au système est introduite pour sélectionner le meilleur couple des tâches à réaliser à un instant donné. La minimisation du temps d'attente des robots lors de l'arrivée des ressources est obtenue en réalisant une affectation prévisionnelle.

L'architecture retenue est hiérarchisée et distribuée, avec au niveau supérieur le superviseur et au niveau inférieur, le contrôle des robots, du système de vision et du tapis.

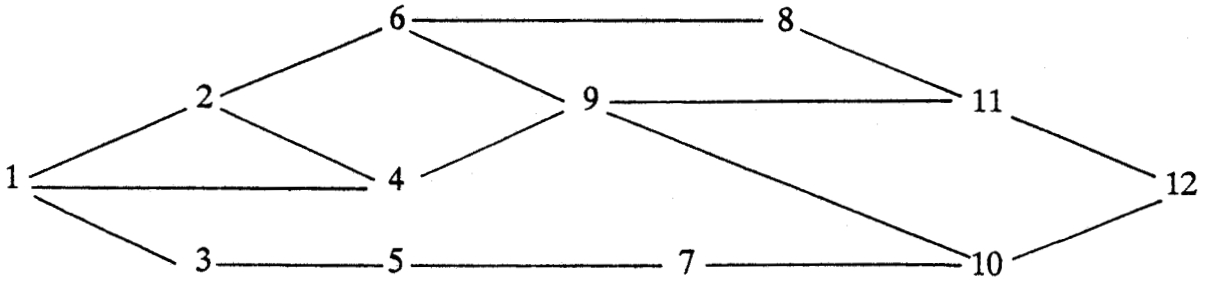
Il convient à présent de réaliser la mise en oeuvre pratique de la procédure proposée et d'approfondir l'étude des modules de contrôle et diagnostic de manière à obtenir une cellule entièrement autonome.

ANNEXES

ANNEXE 1

Influence du paramètre β

Nous considérons le graphe suivant :



Le gain apporté au système pour un seul robot est donné par :

$$R(i) = p_i \cdot e^{-\beta (t_i - T)} + \Phi(EXEC(c_i)) \cdot e^{-\beta (c_i - T)}$$

Les hypothèses considérées dans les exemples présentés sont :

L'ensemble des tâches est réalisé par un seul robot,
Les ressources consommables nécessaires sont présentes.

Dans ces conditions, $(t_i - T)$ est égal à zéro, puisque toutes les ressources sont présentes. Le gain s'écrit alors :

$$R(i) = p_i + \Phi(EXEC(c_i)) \cdot e^{-\beta p_i}$$

Dans chacun des cas, nous donnons la valeur de β et l'ordonnancement obtenu.

1er exemple : toutes les durées sont identiques, $p_i = 10$

Valeurs de β	ordonnancements obtenus
$\beta \leq 1.5$	1 2 6 3 5 4 9 7 8 10 11 12
$\beta = 1.6$	1 2 3 5 6 4 9 7 8 10 11 12
$2 \geq \beta \geq 1.65$	1 2 6 3 4 5 7 8 9 10 11 12
$\beta \geq 2$	1 2 3 4 5 6 7 8 9 10 11 12

2 ième exemple : toutes les durées sont identiques, $p_i = 2$

Valeurs de β	ordonnancements obtenus
$\beta \leq 7$	1 2 6 3 5 4 9 7 8 10 11 12
$\beta = 8$	1 2 3 5 6 4 9 7 8 10 11 12
$\beta = 10$	1 2 3 4 5 6 7 8 9 10 11 12

On constate que le travail futur n'intervient plus lorsque le produit $ß.p_i$ augmente ($\beta.p_i \geq 16$, dans les deux cas). En effet, dans ces conditions, $e^{-\beta.pi}$ est petit, et la valeur actualisée du travail futur également.

3 ième exemple : Dans ce cas, nous avons choisi des tâches de durées différentes.

tâches n°	1	2	3	4	5	6	7	8	9	10	11	12	moyenne
durées	8	9	11	10	12	8	10	12	9	7	13	11	10

Valeurs de β	ordonnancements obtenus
$\beta \leq 0.1$	1 2 6 3 5 8 4 9 11 7 10 12
$\beta = 0.2$	1 3 5 2 6 8 4 9 11 7 10 12
$\beta \geq 0.5$	1 3 5 7 2 4 6 8 9 11 10 12

Le choix de la valeur de β devra être tel que le coefficient $e^{-\beta pi}$ reste petit, de manière à prendre en compte la quantité de travail futur.

ANNEXE 2

Nous présentons différents exemples d'ordonnancements obtenus à partir de la procédure proposée.

1 - Conditions "favorables"

Afin de juger de l'efficacité de la méthode et donc de comparer le résultat obtenu à la solution optimale, nous avons été amené à considérer certaines hypothèses.

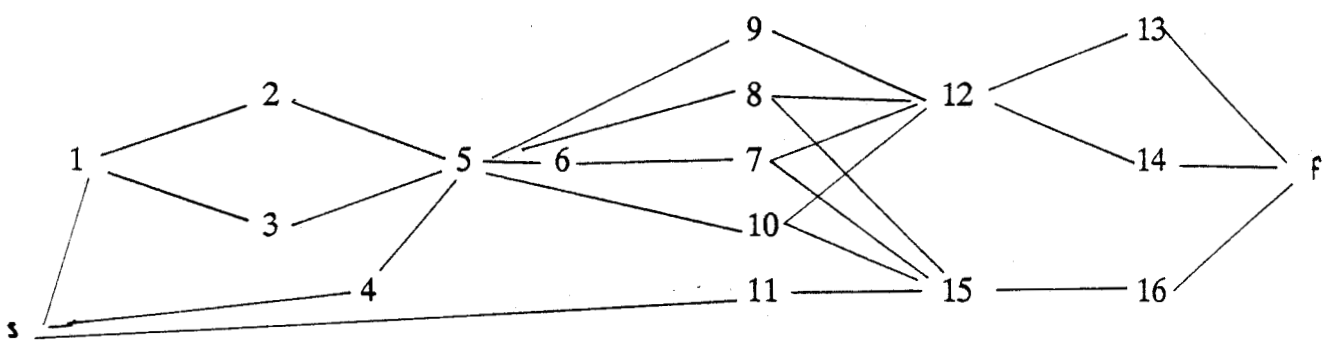
Dans un premier temps, nous considérons le cas favorable, c'est à dire celui pour lequel les hypothèses suivantes sont vérifiées :

- Les ressources consommables associées aux opérations sont disponibles et toutes présentes.

- Il n'y a pas de panne sur les robots.
- Il n'y a pas de conflit d'accès à la zone commune entre les robots.
- Les temps d'exécution sont identiques quel que soit le robot qui exécute la tâche.

Dans le premier exemple, les ordonnancements obtenus dépendent du paramètres β, pour le second, on remarque que la solution obtenue ne dépend pas du paramètre β.

Premier exemple (extrait de (GAB 82))



Graphe des tâches à réaliser

Toutes les tâches ont une durée de 2 unités.

SOLUTION OPTIMALE

Machines	ordonnancement							
1	1	2	5	6	7	9	12	13
2	4	3	11	10	8	15	16	14

Résultats obtenus pour β = **0.8**

Machines	ordonnancement							
1	1	2	5	6	7	9	12	13
2	4	3	11	10	8	15	16	14

Résultats obtenus pour β = **8**

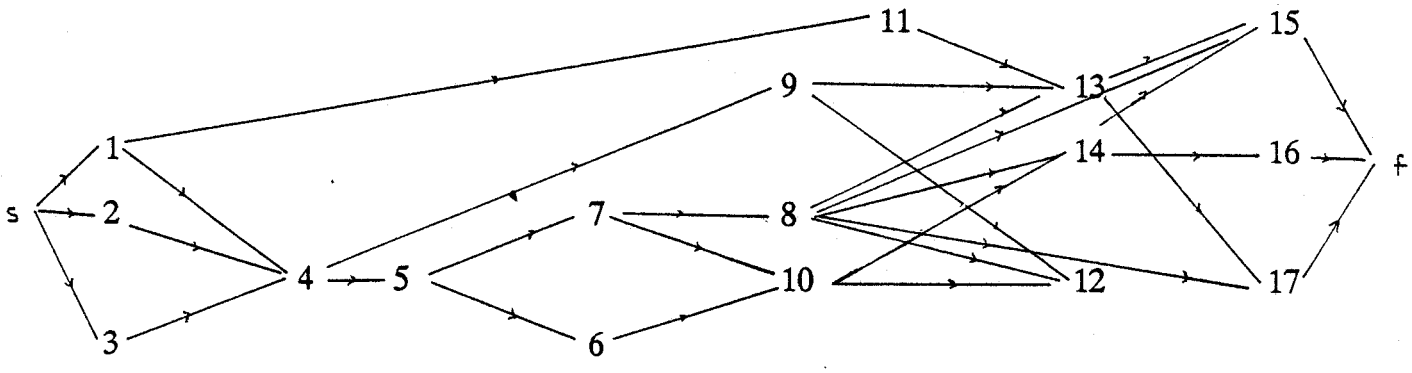
Machines	ordonnancement								
1	1	2	5	6	7	10	12	13	16
2	4	3	11	8	9		15	14	

Le premier cas donne un temps total de 16 unités, ce qui correspond au minimum.

Dans le deuxième exemple, le travail futur n'est pas considéré puisque β est important, nous obtenons alors un temps total de réalisation de 18 unités.

Le résultat optimal, en effet, ne peut pas, en général, être atteint par une politique "avide" (optimiser sur la période en cours sans tenir compte de l'état résultant).

Deuxième exemple (extrait de (CAR 82))



SOLUTION OPTIMALE

Machines	ordonnancement								
1	1	3	4	5	7	8	12	14	15
2	2	11		9	6	10	13	16	17

Résultats obtenus pour $\beta = 0.2$

Machines	ordonnancement								
1	1	3	4	5	7	8	12	14	15
2	2	11		9	6	10	13	16	17

Résultats obtenus pour $\beta = 20$

Machines	ordonnancement								
1	2		11	9	7	10	13	17	16
2	3	1	4	5	6	8	14	12	15

Quelle que soit la valeur de β, nous obtenons la solution optimale. Cet exemple est un cas particulier où la politique "avide" est la politique optimale.

Troisième exemple (extrait de (DJE 82))

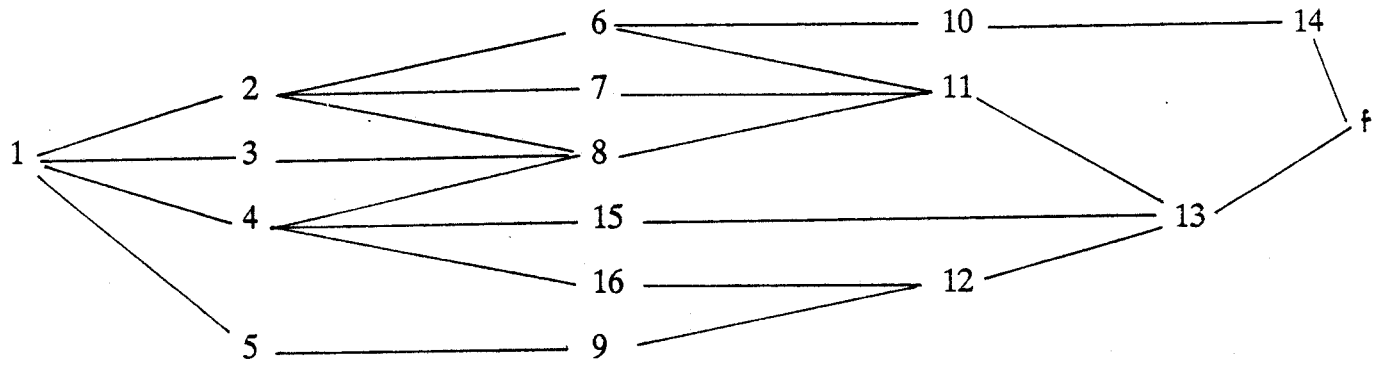
Nous proposons un dernier exemple dans lequel certaines tâches ne peuvent être exécutées que par un seul robot et d'autres par les deux robots.

La répartition des temps d'exécution est la suivante :

Opérations	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Robot 1	-	2	3	-	-	3	5	-	2	-	2	3	-	3	2	-
Robot 2	4	-	3	2	4	-	5	2	-	2	-	3	2	-	2	3

" - " la tâche n'est pas réalisable par le robot considéré.

Le graphe des opérations est le suivant :



SOLUTION OPTIMALE

Robots	ordonnancement								
1		2	6	3	9	7		11	14
2	1	5	4	15	8	10	16	12	13

Temps total = 24 unités

Résultats obtenus pour $\beta = 0.5$

Robots	ordonnancement								
1		2	6	3	7	11	9	14	
2	1	4	15	10	8	5	16	12	13

Temps total = 24 unités

Résultats obtenus pour $\beta = 10$

Robots	ordonnancement								
1		3	2	6	9	12	11	14	
2	1	5	4	7	16	8	10	15	13

Temps total = 26 unités

Dans l'approche multicritère proposée dans (DJE 86), le résultat minimum est obtenu pour une certaine combinaison des 5 coefficients utilisés. La méthode est sensible aux pondérations, et la préférence d'un critère par rapport aux autres entraînent une dégradation des performances.

L'avantage de la solution établie dans ce mémoire est qu'il n'y a qu'un seul paramètre à régler. Des calculs d'ordonnancements préalables permettront l'ajustement de ce coefficient.

2 - Conditions "défavorables"

Les résultats suivants ont été obtenu en levant certaines hypothèses évoquées précédemment. Les calculs sont appliqués au 1er exemple.

1 - Cas d'une panne d'un robot

Il est évident, dans ce cas, que pour un ordonnancement statique selon la durée de la panne, soit les tâches non réalisées sont retardées, soit l'ordonnancement ne peut plus être appliqué et il est nécessaire de faire intervenir un opérateur.

Dans l'approche proposée, si l'ensemble des tâches est réalisable par les deux robots, nous obtenons un ordonnancement qui tient compte de l'indisponibilité d'un des robots.

Ainsi, dans l'exemple suivant, le robot 1 est indisponible pendant 4 unités de temps, à partir de l'instant t = 10.

Dans le cas statique, nous obtenons :

Machines	ordonnancement									
1	1	2	5	6	7			9	12	13
2	4	3	11	10	8	15	16		14	

Temps total = 20

Par la méthode proposée, nous obtenons (β = 0.8) :

Machines	ordonnancement								
1	1	2	5	6	7			15	14
2	4	3	11	8	10	9	12	13	16

Temps total = 18

2 - Problème sur l'arrivée de ressources

Nous avons jusqu'alors considéré que les ressources étaient toutes présentes. Supposons maintenant qu'une ressource est indisponible, dans le cas d'un ordonnancement statique, de la même façon que précédemment, soit les tâches sont retardées, soit il est nécessaire de revoir l'ordonnancement établi.

L'intérêt de notre méthode est dans ce cas, de réaliser les tâches que l'on peut exécuter en attendant l'arrivée de cette ressource.

Dans l'exemple ci-dessous, on considère que la pièce nécessaire à la tâche 11 n'arrive qu'au bout de 10 unités de temps.

Pour un ordonnancement statique, nous aurions :

Machines	ordonnancement										
1	1	2	5	6	7	9			12	13	
2	4	3				11	10	8	15	16	14

Temps total = 22 unités

Par la méthode proposée, nous obtenons (β = 0.8) :

Machines	ordonnancement								
1	1	2	5	6	7	10	12	13	
2	4	3		8	9	11	15	16	14

Temps total = 18 unités

Dans ces derniers cas, la procédure prend en compte les perturbations et adapte l'ordonnancement, ce qui permet de minimiser le temps total de réalisation.

BIBLIOGRAPHIE

ABB 87 L. ABBAZZI, A. DEL TAGLIA,
The problem of feeding parallel workstations in assembly lines, 3 rd International Conference on Simulation in Manufacturing, Turin, Novembre 1987.

AND 86 M.M. ANDREASEN, T. AHM,
The relation between product design, production, layout and flexibility. 7th International Conference on Assembly Automation, Zurich, Février 1986.

AIR 84 G. AIRENTI, M. COLOMBETTI,
An approach to intelligent action execution, Artificial Intelligence and Information Control Systems of Robot, Elsever Science Publishers, 1984.

ALA 83 R. ALAMI,
Un environnement LISP pour la mise en oeuvre de systèmes complexes en robotique,
Thèse Docteur Ingénieur, LAAS Toulouse, 1983.

ARA 84 Rapport A.R.A.
Pôles Ateliers Flexibles, 1984.

ARN 84 M. ARNOUX, A. HAURAT, M.C. THOMAS,
LMAD, un Système Expert pour l'assemblage de pièces mécaniques,
Journées AFCET, L.O.O., Brest, 22-23 Novembre 1984.

BAY 87 M. BAYART, M. STAROSWIECKI, Y. YANG,
Dynamic scheduling applied to cooperating robots, 6th International Conference of F.M.S., Turin, Novembre 1987.

BEH 83 C. BERTHUL,
Etude des régimes dégradés d'ateliers flexibles, Rapport A.R.A., Pôles Ateliers Flexibles, 1983.

BEL 85 G. BEL, D. DUBOIS,
Modélisation et simulation de systèmes automatisés de production, A.P.I.I. Vol 19, n° 1, 1985.

BEL 87 G. BEL, E. BENSANA, D. DUBOIS,
Construction d'ordonnancements prévisionnels, un compromis entre approches classiques et Systèmes Experts, Rapport INRIA, Système de Production, 1987.

BER 83 C. BERARD,
Contribution à la conception de structures logicielles pour le pilotage d'atelier.
Thèse Docteur Ingénieur, Bordeaux 1, 1983.

BER 57 C. BERGE,
Théorie générale des jeux à n personnes,
Edition Gautier-Villars, 1957.

BER 63 C. BERGE,
Théorie des graphes et ses applications,
Dunod, 1963.

BER 71 G. BERNARD, M.L. BESSON,
Douze méthodes d'analyse multicritères,
R.I.R.O. Vol 3, 1971.

BON 85 R. BONETTO,
Les ateliers flexibles de production,
Hermes publishing, 1985.

BON 82 J.C. BONIN, J.P. ELLOY, A. HAURAT, P. MOLINARO,
M.C. THOMAS, M. SILLY,
Les outils de synchronisation du système LMAC pour la coopération de robots industriels,
Rapport interne 13-82, ENSM Nantes, 1982.

BON 84 J.C. BONIN,
LMAC : Système de développement de langage temps réel pour la productique - interfaces et problèmes temps réel, Thèse de Doctorat en Automatique et Informatique, ENSMM Besançon, Avril 1984.

BOU 84 A. BOURJAULT,
Contribution à une approche méthodologique de l'assemblage automatisé : Elaboration automatique de séquences opératoires, Thèse d'Etat, Laboratoire d'Automatique, Besançon, Novembre 1984.

BOU 86 A. BOURJAULT, F. LHOTTE,
Modélisation d'un processus d'assemblage,
A.P.I.I. Vol 20, n° 2, 1986.

BOU 87 (a) A. BOURJAULT, J.M. HENRIOUD,
Détermination des sous-assemblages d'un produit à partir des séquences temporelles d'assemblage,
A.P.I.I. Vol 21, n° 2, 1987.

BOU 87 (b) A. BOURJAULT, D. CHAPPE, J.M. HENRIOUD,
Elaboration automatique des gammes d'assemblage à l'aide de réseaux de Pétri,
A.P.I.I. Vol 21, n° 4, 1987.

CAL 82 J.P. CALVEZ,
Une méthodologie de conception des systèmes multi-microordinateurs pour les applications de commande en temps réel. Thèse d'Etat, Nantes, 1982.

CAM 85 J.P. CAMPAGNE, J. PEYRON, M. TEMANI,
Structuration des bases de données techniques autour de nomenclatures et gammes-mères en vue de l'élaboration automatique de gammes.
A.P.I.I. Vol 19, n° 4, 1985.

CAM 87 J.P. CAMPAGNE, C. CAPLAT,
Elaboration de gammes d'assemblage,
Rapport INRIA, Système de Production, 1987.

CAR 82 J. CARLIER, P. CHRETIENNE,
Un domaine très ouvert : les problèmes d'ordonnancement,
RAIRO, Recherche Opérationnelle, Vol 16, n° 3, 1982.

CLA 86 D. CLASSE, K. FEDMANN,
A dual armed robot system for assembly tasks,
16 th International Symposium on Industrial Robots,
Bruxelles, Septembre 1986.

CLE 86 G. CLERMONT, M. HERMANT, P. GASPART,
Fiacre : a flexible and integrated assembly cell for research and evaluation,
16 th International Symposium on Industrial Robots,
Bruxelles, Septembre 1986.

CON 67 CONWAY, MAXWELL, MILLER,
Theory of scheduling,
Addison Wesley

COR 79 P. CORTI, G. GINI, M. GINI,
Software features for intelligent industrial robots,
Kybernetes, Vol. 8, p. 149-154, 1979.

COU 79 C. COUZINET-MERCE,
Etude de l'existence de solutions pour certains problèmes d'ordonnancement, Thèse de Docteur Ingénieur, Université P. Sabatier, Toulouse, 1979.

CRO 79 CROCUS,
Système d'exploitation des ordinateurs,
Dunod, 1979.

DAL 82 Y. DALLERY, B. DESCOTES-GENON, R. BONETTO,
Simulation of a flexible manufacturing case study of citroën factory, Meudon, A.P.M.S. Advanced in Production Management Sytems, IFIP Working Conference, Bordeaux, Août 1982.

DAU 84 P. DAUCHEZ, R.ZAPATA, A. FOURNIER,
Une application particulière du modèle cinématique à la commande coordonnée de deux manipulateurs,
RAIRO, Automatique, Vol 18, n° 1, 1984.

DJE 85 M. DJEGHABA, M. STAROSWIECKI,
A management system for a flexible assembly cell using robot cooperation, International Conference on Systems, Man and Cybernetics, Tucson, (Arizona), Novembre 1985.

DJE 86 M. DJEGHABA,
Problèmes de décision dans une cellule de production flexible utilisant la coopération entre robots.
Thèse de Doctorat en Automatique, U.S.T.L. Lille, Mars 1986.

DOU 79 G. DOUMEINGTS, F. ROUBELLAT,
Système de conduite et aide à la décision,
R.A.I.R.O. Automatique, Vol 13, n° 1, 1979.

DOU 83 G. DOUMEINGTS, D. BREUIL, L. PUN,
La gestion de production assistée par ordinateur,
Hermes Publishing (France), Avril 1983.

DUE 86 G. DUELEN, U. KIRCHOFF, M. VUKOBRATOVIC, D. STOKIC,
Software system for programming and control synthesis of robots, 16 th International Symposium on Industrial Robots, Bruxelles, Septembre 1986.

DUF 84 B. DUFAY, J.C. LATOMBE,
Un système de programmation des robots par apprentissage inductif, 4 ième congrés AFCET, 1984.

DUM 74 P. DUMAS,
Méthodes d'ordonnancement paramétrique d'un atelier,
Thèse de Docteur Ingénieur, Paris VI, 1974.

DUP 82 C. DUPONT-GATELMAND,
A survey of flexible manufacturing systems,
Journal of manufacturing systems, 1982.

DUP 83 C. DUPONT-GATELMAND,
Le concept d'atelier flexible : une approche industrielle et une approche par les modèles,
Journées A.R.A., Pôles Ateliers Flexibles, 1983.

DUP 86 V.DUPOURQUE, O. ISHACIAN,
Controlling Multirobot Application from Unix,
16 th International Conference on Industrial Robots, Bruxelles, Septembre 1986.

EAG 88 P. J. EAGLE, L. H. TABRIZI,
Application of Kalman filters to assembly signature monitoring using empirical process models,
Third International Conference on CAD/CAM, Robotics and Factories of the Future, Detroit, Août 1988.

EJI 72 M. EJIRI, T. UNO, H. YODA, T. GOTO, K. TAKEYASU,
A prototype intelligent robot that assembles objets from plan drawings, I.E.E.E. Transactions on computers, Vol C-21 n°2, Février 1972.

ELL 83 J.P. ELLOY,
Un noyau exécutif temps réel réparti pour la commande de procédés industriels,
Rapport interne n° 5-83, ENSM Nantes, 1983.

ERS 85 J. ERSCHLER, G. FONTAN, C. MERCE,
Un nouveau concept de dominance pour l'ordonnancement de travaux sur une machine,
R.A.I.R.O. Vol 19, n° 1, Février 1985.

FER 86 R. FERHATI,
Coopération entre robots,
DEA Automatique, U.S.T.L. Lille, Juin 1986.

FON 80 G. FONTAN,
Notion de dominance et son application à l'étude de certains problèmes d'ordonnancement,
Thèse d'Etat, Université P. Sabatier, Toulouse, 1980.

FOR 84 T.H. FORMAN,
Robotic considerations improve defense product,
Robot 8, Détroit, Juin 1984.

GAB 82 H. N. GABOW,
An almost-linear algorithm for two-processor scheduling, Journal of the Association for Computing Machinery, Vol 29, n° 3, 1982.

GER 87 J.P. GERVAL,
Contribution à l'étude des systèmes coopératifs : LCOOP, un langage de coopération, Thèse de Doctorat en Automatique, Université de Valenciennes, 1987.

GLI 84 F. GLIVIAK, J. KUBIS, A. DICOUSKY, E. KARABINOSOVA,
A manufacturing cell management systems "CEMAS", Artificial Intelligence and Information Control Systems of Robots, Elsever Science Publishers, 1984.

GRO 75 D.D. GROSSMAN, M.W. BLASGEN,
Orienting mechanical parts by computer controlled manipulator, I.E.E.E. Transactions on Systems, Man and Cybernetics, Septembre 1975.

GUI 68 G. Th. GUILBAUD,
Eléments de la théorie mathématique des jeux,
Dunod, 1968.

GUI 77 G.L. GUIGOU,
Méthodes multidimensionnelles, Analyse de données et choix à critères multiples, Dunod, 1977.

HAU 87 A. HAURAT, J.L. PERRARD,
Panorama des langages de programmation des robots industriels.
Axes Robotique, N° 26, Septembre-Octobre 1987.

IAC 87 M. IACOVELLA,
Flexible assembly systems in the electronics industry, 6 th International Conference on F.M.S., Turin, Novembre 1987.

JAC 84 S. JACQMART,
Système flexible d'alimentation pour robots d'assemblage,
Thèse Docteur Ingénieur, ENSMM Besançon, 1984.

JEA 86 (a) R. JEANNES,
Méthodologie d'analyse des produits pour leur reconception en vue du montage automatisé
Thèse Docteur Ingénieur ENSAM PARIS, Mars 1986.

JEA 86 (b) P. JEANNIER,
Caractéristiques opératoires des robots d'assemblage,
Thèse de Docteur en Automatique, Besançon, 1986.

JOL 88 D. JOLLY, R. FERHATI, M. BAYART, M. STAROSWIECKI,
Coupling of a robotic langage with a supervisor for a simultaneous adaptive control of two robots.
Third International Conference on CAD/CAM, Robotics and Factories of the Future, Detroit, Août 1988.

KAC 88 M. KACZMAREK,
Coopération entre robots,
DEA Informatique, U.S.T.L. Lille, Juillet 1988.

KAL 85 G. KALLEL, X. PELLET, Z. BINDER,
Conduite décentralisée coordonnée d'atelier,
A.P.I.I. Vol 19, n° 4, 1985.

KAU 73 A. KAUFMANN,
Introduction à la théorie des sous ensembles flous,
Edition Masson, 1973.

KEM 85 K.G. KEMPF,
Manufacturing and Artificial Intelligence,
Robotic 1, pp 13-25, 1985.

KOH 86 M. KOHNO, A. MIYAKAWA,
Real time synchronization of two robots for coordinated assembly, 16 th International Symposium on Industrial Robots, Bruxelles, Septembre 1986.

KUS 85 A. KUSIAK,
Planning of flexible manufacturing systems,
Robotica, Vol 3, pp 229-232, 1985.

LAP 80 A. LAPORTE,
Un robot d'assemblage automatique opérant dans un univers de blocs, Thèse de Docteur Ingénieur,
Université P. Sabatier, Toulouse, 1980.

LAS 84 J.F. LASZCZ,
Product design for robotic and automatic assembly.
Robot 8, Detroit, Juin 1984.

LEM 84 P. LEMAIRE,
Coopération entre les robots,
DEA Automatique, U.S.T.L. Lille, Juin 1984.

LEV 82 D. LEVEQUE,
Lancement périodique de produits dans un atelier flexible, Thèse de Docteur Ingénieur, L.A.A.S.
Toulouse, 1982.

LHO 82 F. LHOTTE,
Recherches à caractère mécanique ou technologique,
Revue "gadz'arts 81-82", La production flexible, situations et impacts, 1982.

LIS 83 A.M. LISTER,
Principes fondamentaux des systèmes d'exploitation,
Edition Eyrolles, 1983.

LOP 86 E. LOPEZ-MELLADO, R. ALAMI,
An execution monitoring system for a flexible assembly workcell, 16 th International Symposium on Industrial Robots, Bruxelles, Septembre 1986.

LOZ 76 T. LOZANO-PEREZ,
The design of a mechanical assembly system,
AI. TR 397, Artificial Intelligence Laboratory,
M.I.T., 1976.

LOZ 83 (a) T. LOZANO-PEREZ,
Robot programming,
Proceedings of I.E.E.E., Vol. 71 n° 7, July 1983.

LOZ 83 (b) T. LOZANO-PEREZ, M.T. MASON, R.H. TAYLOR,
Automatic synthesis of fine motion strategies for robots, 1 st International Symposium on Robotic Research, Oretton Woods, 1983.

MAI 85 O.Z. MAIMON,
A multi robot control experimental system with random parts arrival,
Proceedings IEEE International Conference on Robotics Automation, 1985.

MAG 82 P. MAGUET,
Les ateliers flexibles de montage,
Thèse Docteur Ingénieur ENSAM PARIS, Novembre 1982.

MAR 78 A. MARTEL,
Techniques et applications de la Recherche Opérationnelle, Edition Gaetan-Morin, 1978.

MAZ 81 E. MAZER,
Réalisation d'un support expérimental de recherche pour le projet robotique PANDORE, Définition et implantation du langage LM,
Thèse 3 ième cycle, INPG Grenoble, Janvier 1981.

MER 87 G. MERLIN,
A study case of critical analysis, relating to an automatic assembly system, 6 th International Conference on F.M.S., Turin, Novembre 1987.

MIR 84 J.F. MIRIBEL, E. MAZER,
Le langage LM - Manuel de référence,
Editions Cepadues, 1984.

MON 85 D. MONTEIL,
SAGA : Simulation et Aide à la Gestion d'Ateliers,
Thèse de Docteur de l'INSA Toulouse, Octobre 1985.

NEP 78 P. NEPOMIATSCHY,
Résolution d'un problème d'ordonnancement à ressources variables, R.A.I.R.O. Recherche Opérationnelle, Vol 12, n° 3, Août 1978.

OH 82 G.R. OH, J. FAVREL, J.P. CAMPAGNE,
Graphic modellings by Petri nets for the production planning, A.P.M.S. Advanced in Production Management Sytems, IFIP Working Conference, Bordeaux, Août 1982.

OKH 84 D.E. OKHOTSIMSKY, S.S. KAMYNIN, E.I. KUGUSHEV,
Automatic multioperation assembly and application of visual control, Artificial Intelligence and Information Control Systems of Robots,
Elsevier Science Publishers, 1984.

OWE 86 T. OWEN, E. DOMBRE,
Les robots d'assemblage,
Hermes Publishing, 1986.

PAR 83 M. PARENT, C. LAURGEAU,
Les robots, Tome 5, Langages et méthodes de programmation, Hermes Publishing (France), 1983.

PAR 84 W.T. PARK,
State-space representations for coordination of multiple manipulator, 14 th International Symposium on Industrial Robots, Zurich, 1984.

PER 86 A. PEREZ,
Japon : Mille options en grandes séries,
Industries et Techniques, Février 1986.
Robotique d'assemblage : D'abord repenser le produit,
Industries et Techniques, Septembre 1986.

PIC 86 J.F. PICARDAT,
An A.I. robot programming architecture : MUCAR,
16 th International Symposium on Industrial Robots,
Bruxelles, Septembre 1986.

ROY 70 B. ROY,
Décisions avec critères multiples : Problèmes et méthodes, 7 ième Symposium de La Haye, 1970.

ROY 80 B. ROY, P. VINCKE,
Systèmes relationnels de préférence en présence de critères multiples avec seuils,
Cahier du C.E.R.O. Vol 22, n° 1, 1980.

RUS 87 M.G. RUSSO, R. MANUELLI,
Experimental center for system integration in C.I.M.,
6 th International Conference on F.M.S., Turin,
Novembre 1987.

SAN 83 A.C. SANDERSON, G. PERRY,
Sensor based robotic assembly systems, Research and Applications in Electronic manufacturing,
Proceedings of I.E.E.E. Vol 71, n° 7, Juin 1983.

SCE 84 SCEPTRE : Proposition de noyau normalisé pour les exécutifs temps réel, projet développé par le Bureau d'Orientation de la Normalisation en Informatique,
T.S.I. Vol. 3, n° 1, 1984.

SEP 85 M. SEPASER,
Temps de cycle d'un poste d'assemblage robotisé,
Diplôme Universitaire de Formation à la Recherche,
Université de Franche-Comté, Novembre 1985.

SOE 77 R. SOENEN,
Contribution à l'étude des systèmes de conduite en temps réel en vue de la commande d'unités de fabrication, Thèse d'Etat, U.S.T.L. Lille, 1977.

SOL 84 K.D. SOLDNER, K.L. SPLITZNAGEL,
A multi tasks robotic workcell for electronic assembly, Robot 8, Détroit, Juin 1984.

SOU 81 J.P. SOUBRIER,
Un modèle de résolution des problèmes d'ordonnancement dynamique, Thèse de Docteur Ingénieur, Université P. Sabatier, Toulouse, 1981.

SOU 82 J.P. SOUBRIER,
Un algorithme de résolution des problèmes d'ordonnancement dynamique, R.A.I.R.O. Recherche Opérationnelle, Vol 16, n° 3, Août 1982.

STA 85 (a) M. STAROSWIECKI, M. DJEGHABA, M. BAYART,
Moniteur de coopération entre robots dans une cellule flexible d'assemblage, Computer aided design and applications, IASTED, Paris, Juin 1985.

STA 85 (b) M. STAROSWIECKI, M. DJEGHABA, M. BAYART, G. REYMAN,
Decisions problems in a flexible assembly cell using robot cooperation, 4 th International Conference on Systems Engineering, Coventry, (England), Septembre 1985.

STA 85 (c) M. STAROSWIECKI, M. DJEGHABA, M. BAYART,
Task scheduling by multicriteria optimization in a flexible assembly cell using robot cooperation, 15 th International Symposium on Industrial Robots, Tokyo, (Japon), Septembre 1985.

STA 87 (a) M. STAROSWIECKI, Y. YANG, M. BAYART,
Dynamic scheduling of a queueing system with application to the control of a robotized assembly, 7 th International Congress of Cybernetics and Systems, Londres, Septembre 1987.

STA 87 (b) M. STAROSWIECKI, M. BAYART, D. JOLLY,
Integration of vision, transportation and assembly tasks in a flexible assembly cell using robot cooperation, 2nd International Conference on Robotics and Factories of the future, San Diego, Juillet 1987.

TEM 85 M. TEMANI,
Contribution à la modélisation des processus d'assemblage : élaboration automatique des gammes, Thèse 3 ième cycle, INSA LYON, Janvier 1985.

THO 80 M.C. THOMAS,
Aide à la décision pour un ordonnancement Temps Réel, Thèse 3 ième cycle, Université P. Sabatier, Toulouse, 1980.

VER 80 M. VERNET,
Contrôle d'une équipe de robots à aptitudes multiples collaborant à l'exécution d'une même tâche,
Thèse Docteur Ingénieur, INPG Grenoble, Octobre 1980.

YAN 87 Y. YANG, M. STAROSWIECKI, M. BAYART,
Génération en temps réel du plan d'action d'un robot à partir d'un graphe potentiel tâches, 10th International Symposium on Robotics and Automation, Lugano, (Suisse), Juillet 1987.

YAN 88 Y. YANG
Allocation optimale des tâches pour la coopération de deux robots dans une cellule flexible d'assemblage,
Thèse de Doctorat en Automatique, U.S.T.L. Lille, Janvier 1988.

ZAP 83 R. ZAPATA, P. DAUCHEZ, P. COIFFET,
Co-operation of robots in gripping tasks : the exchange problem, Robotica Vol. 1 pp 73-77, 1983.