FCPPG : Générateur de planificateurs de tâches pour une cellule flexible de production.

Samia Aoudia [2], Djamila Ouelhadj [1] et Abdelgnani Souilah [1]

[1] **Centre de Développement Des Technologies Avancées**
128, Chemin Mohamed Gacem, El Madania, Alger, Algérie
[2] **Institut National d'Informatique**
B.P. 68M Oued smar, 16270 Alger, Algérie

Mots clefs

Cellule flexible de production, Planification de tâches, Générateur de planificateurs, Intelligence artificielle, Logique des prédicats, Programmation orienté objet.

Résumé

Dans cet article, nous décrivons un générateur de planificateurs de tâches dédié au cellules flexibles de production où des tâches d'usinage, d'assemblage et de manutention sont exécutées. Notre système s'inspire de l'approche systèmes experts. Cela nécessite la définition de trois modules superposés: un langage de description de la connaissance du domaine, un langage de description de l'application et un mécanisme de raisonnement. Le langage de représentation de la connaissance est une hybridation entre trois formalismes: les objets, la logique des prédicats et les règles de production. Le moteur d'inférence du premier ordre fonctionne en chaînage arrière, son mécanisme de filtrage est optimisé par des heuristiques.

Abstract

This paper describes a task planners generator dedicated to flexible production cells where tasks like machining, assembling and material handling are processed. Our system is inspired from expert system approach. This needs the definition of tree superposed moduls: a knowledge description language, a data or appliction description language and a reasonning mechanism. The knowledge description language is a hybridization between three formalisms: objects, predicat logic and production rules. The first order inference engine runs with backward chaining and incorporates a matching mechanism optimised with some developed heuristics.

---

1. Introduction

La recherche en productique tend vers une intégration de toute la chaîne allant de la conception jusqu'à la fabrication du produit fini et même sa commercialisation dans des systèmes informatiques. Ceci rentre dans le cadre d'une discipline communément appelée «Computer Integrated Manufacturing» (CIM) [1]. Parmi les maillons de cette chaîne nous retrouvons les systèmes générateurs de plans de tâches de fabrication. Nous nous intéressons, dans ce travail à

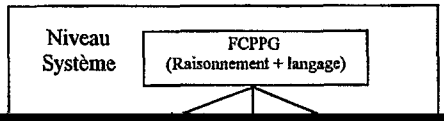
la planification de tâches dans les systèmes cellulaires de production et en particulier aux tâches effectuées à l'intérieur des cellules [9]. Ces tâches ont la particularité d'être complémentaires, d'avoir la pièce comme dénominateur commun et d'être liés à travers leurs ressources. Ces types

fonction de la demande, et d'autre part, ces types de produits sont réalisés simultanément. Du fait que la cellule est flexible, ces trois principales tâches sont planifiées et replanifiées globalement ou partiellement si l'un des cas suivants se présente:

- ajout ou suppression d'un produit dans la famille,
- forte fluctuation de la demande de produit,
- ajout ou suppression d'une ressource dans la cellule,
- surgissement d'aléas dans la cellule,
- modification de l'agencement de la cellule.

3. Approche de résolution

Pour assurer une bonne coordination entre les différents types de plans à générer, et pour optimiser le temps de traitement et améliorer la qualité de la solution finale, nous nous proposons de concevoir un générateur de planificateurs de tâches FCPPG (Flexible Cell Process Planner Generator) prenant en charge les trois principaux types de tâches :usinage, assemblage, et manutention. Ce système repose sur une approche IA, en particulier la méthodologie système expert. Cela suppose le développement d'un langage de description des connaissances du monde et d'un moteur d'inférences [3][4]. Le langage va permettre de décrire les connaissances du domaine de la production nécessaires à la planification, à savoir : les objets, les relations entre objets et les opérateurs de planification. Le moteur d'inférences est le raisonnement qui permet de générer le plan de tâches recherché. Notre système se veut ouvert et non spécifique, dans le sens ou il comportera trois niveaux (fig. 2) à savoir le niveau système, le niveau expert et le niveau utilisateur.



Etant donnée une cellule flexible de production

5. Langage d'expression des connaissances.

La complexité et la variété des connaissances propres aux domaines de production (usinage, assemblage, manutention), nécessite la conception d'un langage adoptant une représentation hybride :

- représentation orientée objets pour la description des objets du monde,
- logique des prédicats pour l'expression des relations,
- règles de production pour la représentation des connaissances opératoires. Ces règles sont, à leur tour, exprimées par des variables (objets structurés) et des prédicats.

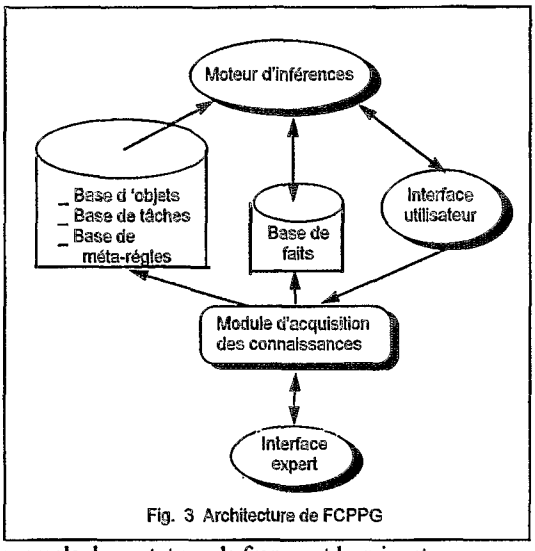


Fig. 3 Architecture de FCPPG

5.1. description des objets et des relations du monde: Base d'objets

Un objet du monde est représenté par un frame. Un frame est l'entité génératrice (prototype) capable d'engendrer les instances du frame par application d'un processus d'instantiation [2]. Un exemple du prototype de frame est le suivant.

```
((Nom frame )
(type frame)
(attribut1(type valeur))
.....
(attributn(type valeur))
(liste-sous-frames))
```

Le frame décrit un objet hiérarchiquement. Cette notion de hiérarchie est essentiellement utilisée lorsqu'on manipule des objets pouvant être subdivisés en un certain nombre de composants. La hiérarchie de composition (lien d'héritage) est schématisée par un arbre (fig. 4) . Un exemple de description du frame pièce est donné dans figure 5.

Une relation du monde est décrite par un prédicat suivi de la liste de ses arguments.

Exemple : SUR (pièce 1, pièce 2)

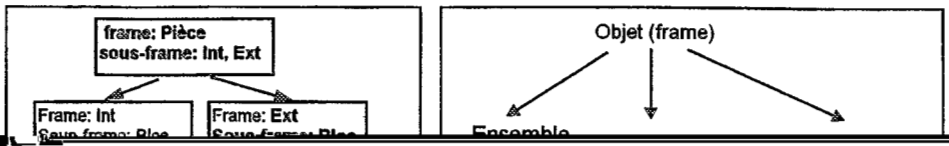
5.2. Description des connaissances opératoires

Les connaissances représentent le savoir-faire sur le domaine. On distingue deux types de connaissances opératoires :

- les tâches de planification (opérateurs),
- les méta_règles : règles de sélection de tâches.

Elles sont exprimées par des règles de production . Le bloc condition traduit la situation que doit vérifier le monde(état initial ou final) pour le déclenchement de la conclusion. Le bloc condition est formé d'un ensemble de prémisses. Deux prémisses successives sont implicitement liées par le connecteur ET. La partie condition sera vérifiée si toutes les prémisses qui le composent sont vérifiées, elle est fausse si l'une au moins de ses prémisses est fausse.

Une prémisse est soit : (i) un prédicat suivi de la liste de ses arguments, (ii) une contrainte qui est un test de comparaison de la forme : <état> <Frame>.att <comparateur> <valeur>.


frame: Pièce
sous-frame: Int, Ext
Frame: Int
Frame: Ext
Objet (frame)

Tâche : chambrage
Priorité 2
*SI [(initial piece.exi.bloc-h.face-j.côte-k >0) et (inf (initial piece.int.bloc-l.face-j.*côte.val) > 12)*
et (initial piece.int.bloc-i.face-j.côte-k.val < final piece.int.bloc-i.face-j.cote-k)]
ALORS [(Initial piece.int.bloc-i.face-j.cote-k.val= finalpiece.int.bloc-i.face-j.cote-k.val)
et(Initial piece.int.bloc-i.face-j.cote-k+1.val= final piece.int.bloc-i.face-j.cote-k+1.val)
et(Initial piece.int.bloc-i.face-j+1.cote-k.val= final piece.int.bloc-i+1.face-j.cote-k.val)
et (Initial piece.int.bloc-i-1.face-j+1.cote-k.val= final piece.int.bloc-i.face-j.cote-k.val)]
où: h: numéro du dernier bloc ; i : indice du bloc concerné par le chambrage; j : relatif au numéro de la face latérale j du bloc; k :premier diamètre de la face j du bloc l : i, i+1,.....n,; n étant l'indice du dernier bloc.

Tab. 1 Règle de séléction de la tâche de chambrage

La tâche ''emmanchement'', de l'assemblage (Tab. 2), réalisant une liaison ''pivot glissant'' sera déclenchée si : (la liaison n'est pas encore réalisé) et (l'ensemble contient déjà une liaison réalisée, l'une des deux pièces concernée par cette liaison, doit être dans l'ensemble) et (la liaison en question empêche une autre liaison d'être réalisée, cette dernière doit être déjà réalisée).
Ses effets se traduisent par : (une augmenter le nombre de pièces à insérer dans l'ensemble) et (une mise à jour l'état de la pièce) et (la liaison une fois réalisée, elle sera empêchée par toute autre liaison).

Tâche: emmanchement
Priorité=1
*SI [(initial ensemble.liaison_i.etat=0) et (sup (initial ensemble.*liaison.etat=0)*
et(inf(initial ensemble.liaison_i.lieu_j.(pièce).ordre)=0)
*et (final *empêche (ensemble.liaison_i, ?x / initial x.etat =1)]*
ALORS
[(ensemble.liaison_i.etat =1) et (ensemble.ordre=initial ensemble.ordre+1)
et(ensemble.liaison_i.lieu_j.(pièce).ordre=initial ensemble.ordre)
*et (supprim *empêche (?y, initial ensemble.liaison_i))],*
TEL QUE [(j=1)]

Tab. 2 Gègle de séléction de la tâche d'emmanchement

5.2.2. Description des règles de sélection de tâches: base de méta_règles

Pour faire une restriction sur l'ensemble des tâches qui interviennent dans la génération du plan recherché, on utilise des méta-règles, qu'on appelle règle de sélection de tâches. Le modèle générique d'une règle de sélection est décrit par les champs suivants :
où «condition» représente les conditions de sélection de la tâche. On retrouve dans la partie conclusion d'une règle les noms des tâches sélectionnées, chacune suivie de la liste de ses paramètres. Une sous-règle est l'ensemble des règles de sélection des sous-tâches de la tâche.

((Nom méta-règle)
(condition)
(conclusion)
(sous-règles))

L'exemple du tableau 3 montre la règle de sélection de la tâche de chariotage cylindrique et ses sous-tâches.

5.3 Description des connaissance sur le problème: Base de faits

Elles sont relatives à l'énoncé du problème : description de tout l'environnement du problème avant l'application du processus de planification (état initial), et après application de celui-ci (état

Un exemple de description de l'état initial et de l'état final d'un ensemble de pièces à assembler est décrit dans les figures 7 et 8.

Règle 1: *chariotage cylindrique1*
SI *[(finalpièce.ext.bloc_i.face_j.côte_k.val= final pièce.ext.bloc_i.face_j.côte_k+1.val)*
et (finalpièce.ext.bloc_i-1.face_j.côte_k+1.val< final pièce.ext.bloc_i.face_j.côte_k.val)
et (finalpièce.ext.bloc_i+1.face_j.côte_k.val< final pièce.ext.bloc_i.face_j.côte_k.val)]
ALORS
Chariotagecylindrique(final pièce.ext.bloc_i.face_j)

Sous_règles : sous-règle1, sous-règle2
Sous-règle1

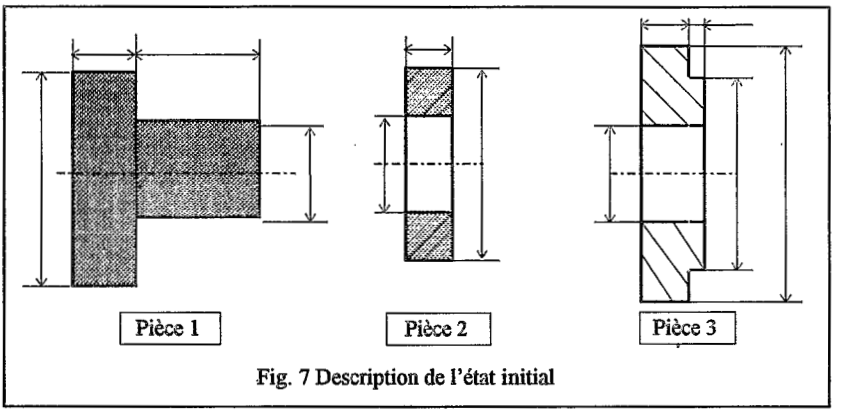


Fig. 7 Description de l'état initial

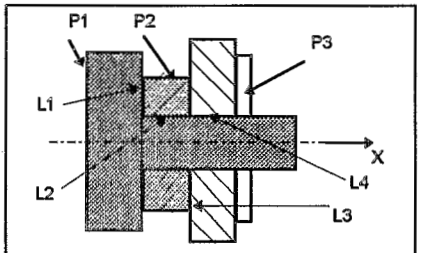


Fig. 8 Description géométrique de l'état final

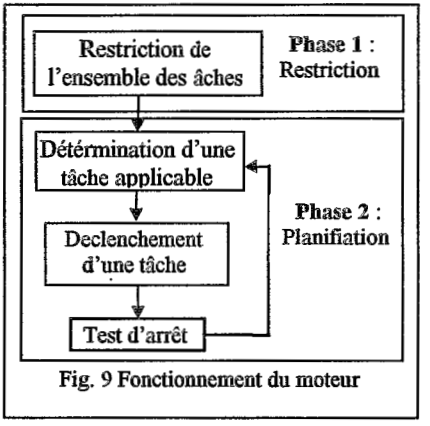


Fig. 9 Fonctionnement du moteur

6.2. Phase Planification

A ce niveau, la notion de hiérarchie entre les tâches est négligée en remettant tout à plat. Toutes les tâches et sous-tâches sont prises au même niveau.

a. Détermination d'une tâche déclenchable

Une tâche sera déclenchée si elle est sélectionnée par une méta-règle (la priorité de la tâche correspond à la priorité courante du système) ou si le filtrage de la partie condition de la tâche avec les faits de la base de faits est un sucée.

b. Exécution (déclenchement) d'une tâche

Cette phase consiste à déclencher la tâche choisie lors de la phase précédente. Le déclenchement de la tâche consiste en la prise en compte de ses effets sur l'état courant du problème. Ces effets seront réellement pris en compte si la tâche ne possède aucune sous-tâche (cas d'une tâche élémentaire), ou si ses sous-tâches sont toutes déclenchées. Un effet est soit un ajout de prédicat, une suppression de prédicat, ou un changement de valeur de l'attribut d'un objet.

c. Test de condition d'arrêt

Un arrêt du raisonnement se voit nécessaire, si le plan recherché est atteint par l'atteinte de l'état final du monde. L'état final est atteint si les valeurs des attributs de tous les objets de l'état

courant sont égales à celles de l'état final, et l'ensemble des prédicats de l'état courant est identique à celui de l'état final.
L'arrêt du système n'est pas toujours déclenché après génération du plan de tâches, ce dernier peut être causé en cas d'un plan non réalisable. Un plan est dit non réalisable si aucune tâche ne peut être déclenchée avant d'atteindre la condition d'arrêt.

7. Conclusion

Le système FCPPG est destiné à la génération de planificateurs des tâches d'usinage, d'assemblage et de manutention dans une cellule flexible de production. Ce système se basant sur une approche I.A, dispose d'un langage de description des connaissances hybride assez puissant (objets, logique des prédicats, règles de production), et d'un moteur d'inférences d'ordre 1. Les interfaces utilisateur/expert sont développés sur l'environnement Windows offrant la possibilité d'utilisation de la souris et du multi-fenêtrage. FCPPG a été validé dans les domaines de l'usinage et de l'assemblage satisfaction.

Annexe

Session de génération d'un plan d'usinage d'une pièce de révolution.

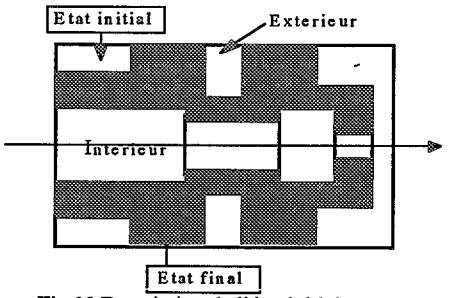


Fig.10 Description de l'état initial et final d'une piéce

Résultats de la phase de restriction

1. chariotage-cylindrique(final piece.ext.bloc2.face2, final piece.ext.bloc4.face2, final piece.ext.bloc2.face2)
2. dressage de face(final piece.ext.bloc5.face3)
3. draissage-angle2 (final piece.ext.bloc5.face2, final piece.ext.bloc4.face3)
4. rainurage (finalpiece.ext.bloc2.face3, final piece.ext.bloc3.face2, finalpiece.ext.bloc4.face1, final piece.ext.bloc1.face2, final piece.ext.bloc2.face1)
5. centrage (final piece.ext.bloc5.face2)
6. filetage (final piece.ext.bloc5.face2)
7. tronçonnage (final piece.ext.bloc1.face1)
8. perçage(final piece.int.bloc4.face2)
9. alésage cylindrique (final piece.int.bloc4.face2)
10. chambrage (final piece.int.bloc2.face2)
11. gorgeage (final piece.int.bloc3.face2, final piece.int.bloc3.face1, final piece.int.bloc1.face2, final piece.int.bloc1.face1, final piece.int.bloc1.face3)

12. taraudage (final piece.int.bloc4.face2)

Résultat de la phase de planification

10 draissage-face (final piece.ext.bloc5.face3)
20. chariotage-cylindrique(final piece.ext.bloc2.face2)
30. rainurage(finalpiece.ext.bloc1.face2final piece.ext.bloc2.face1, final piece.ext.bloc2.face3, final piece.ext.bloc3.face2 ,final piece.ext.bloc4.face1)
50. draissage-angle (final piece.ext.bloc5.face2, final piece.ext.bloc4.face3)
60. chariotage-cylindrique(final piece.ext.bloc4.face2)
70. filetage (final piece.ext.bloc5.face2)
80. centrage (final piece.ext.bloc5.face2)
90. perçage (final piece.int.bloc4.face2)
100. gorgeage (final piece.int.bloc1.face2,final piece.int.bloc1.face1, final piece.int.bloc1.face3, finalpiece.int.bloc3.face2, final piece.int.bloc3.face1)
120. chambrage (final piece.int.bloc2.face2)
130. alésage-cylindrique (final piece.int.bloc4.face2)
140. taraudage (final piece.int.bloc4.face2)
150. tronçonnage (final piece.ext.bloc1.face1)

Remerciements: Les auteurs tiennent à remercier Monsieur Mahfoud Guerbas pour l'aide précieuse qu'il leur a fournie. Ils tiennent également a remercier les examinateurs de ce travail pour leurs conseils enrichissants.

Bibliographie

[1] R. Bonetto, «Les ateliers flexibles de productions», Editions Hermes, 1987.
[2] D. Colnet, D. Léonard et K. Tomber, «Les langages à objets», Editions InterEdition , 1989.
[3] J. L. Lauriére, «Intelligence Artificielle», Tome, Editions Eyrolles, 1987.