ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE MÉCANIQUE
M. ING.

PAR
ZIANE BENSALAH

CONCEPTION ET MISE EN ŒUVRE D'UN SYSTÈME DE BASE DE CONNAISSANCES POUR L'ORDONNANCEMENT DYNAMIQUE DE GESTION DE PRIORITÉS, BASÉ SUR L'APPROCHE ALGORITHMIQUE

MONTRÉAL, DÉCEMBRE 2000

CE MÉMOIRE A ÉTÉ ÉVALUÉ
PAR UN JURY COMPOSÉ

- M Dao Thien-My, directeur de recherche
  Département de génie mécanique à l'École de technologie supérieure

- M. Youssef, A Youssef, professeur
  Département de génie mécanique à l'École de technologie supérieure

- M. Guy Fournier, ingénieur
  Service du génie industriel à IBM microélectronique, usine de Bromont

IL A FAIT L'OBJET D'UNE PRÉSENTATION DEVANT CE JURY ET UN PUBLIC
LE 5 OCTOBRE 2000
À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

CONCEPTION ET MISE EN ŒUVRE D'UN SYSTÈME DE BASE DE CONNAISSANCES POUR L'ORDONNANCEMENT DYNAMIQUE DE GESTION DE PRIORITÉS, BASÉ SUR L'APPROCHE ALGORITHMIQUE

Ziane Bensalah

(SOMMAIRE)

Ce rapport présente un modèle de système de base de connaissances d'ordonnancement dynamique pour l'optimisation de la production dans un environnement manufacturier de plusieurs produits. L'objectif de ce travail est de fournir à l'utilisateur un outil d'aide à la décision dans le choix de calendriers de production réalisables et optimaux.

Nous avons présenté une recherche bibliographique pour essayer de donner un aperçu sur ce que se fait aujourd'hui dans ce domaine en plein expansion et un passage en revue de possibilités existantes. La recherche bibliographique aborde la complexité d'un système manufacturier "job shop". Nous présentons une classification des problèmes d'ordonnancement et les algorithmes utilisés pour la résolution de ces problèmes en fonction des critères économiques à optimiser. Des exemples ont été présentés pour faciliter la compréhension.

L'arrivée de la nouvelle technologie et son utilisation pour l'automatisation des systèmes de production a suscité un grand intérêt dans le développement des systèmes d'ordonnancement dynamique en temps réel. Comme le système de production change de statut imprévisiblement, alors le système de base de connaissances pour l'ordonnancement dynamique que nous proposons sera une solution pour la résolution des problèmes d'ordonnancement complexes. L'approche que nous proposons pour la construction de ce système est une combinaison des heuristiques et des règles de base, afin de pourvoir l'utilisateur de conseils et d'avis sur les décisions de gestion des priorités sur les lignes de production. Cet outil permet aussi à ce dernier de maîtriser la situation du changement dynamique de l'environnement manufacturier. En outre, les informations générées sont utilisées pour le contrôle de la performance du système de production.

La méthodologie que nous avons utilisée, est assez différente de ce qui se fait à ce jour dans l'ordonnancement des systèmes de production; une tentative de montrer une voie de recherche qui pourrait constituer une alternative intéressante dans l'avenir.

DESIGN AND IMPLEMENTATION OF KNOWLEDGE BASED DYNAMIC SCHEDULING SYSTEM FOR PRIORITIES MANAGEMENT, BASED ON THE ALGORITHMIC APPROACH

(ABSTRACT)

This document presents a model of knowledge based dynamic scheduling system for the production optimization in the manufacturing environment of several products. The objective of this work is to provide to the user a tool of decision-making aid in the choice of realizable and optimal promises to pay of production.

We presented a library search trying to give one seen on what is done today in this field into full expansion and a passage in review with existing possibilities. The library search approaches the complexity of a job shop manufacturing system. We present a classification of the scheduling problems and algorithms used for the resolution of these problems according to the economic criteria to optimize. Examples were given to facilitate comprehension.

The arrival of the new technology and its use for production systems automation aroused a great interest in the development of the real time dynamic scheduling systems . As the production system changes statute, then our knowledge-based dynamic scheduling system, which we propose, will be a solution for the resolution of the complex scheduling problems . The approach that we propose for the construction of this system is a combination of basic heuristic and rules to provide the user with consulting and opinion on the possible decisions of the priorities' management on the production lines to control the situation of the dynamic change of the manufacturing environment. Moreover, generated information is used for the performance control of the production system.

The methodology presented here is rather different from what is done to date in the production scheduling systems, an attempt to show a voice of search, which could constitute an interesting alternative in the future.

REMERCIEMENTS

Je tiens à exprimer mes remerciements les plus vifs et les plus sincères :

- Au professeur Dao Thien My qui a accepté de me diriger et de me guider lors de la réalisation de ce projet. Son bon jugement et son enthousiasme ont été pour moi une source de motivation.

- À messieurs Étienne Lemieux, directeur de génie industriel et planning et Guy Fournier, chef de projet développement d'IBM Bromont qui m'ont bien accueilli en me permettant de réaliser ce projet et de parfaire mes connaissances.

- À tous le personnel de l'ingénierie, de la fabrication et du génie industriel de l'usine d'IBM pour avoir si bien répondu à mes questions.

Finalement, je remercie tous les membres de ma famille pour leur patience et leur support moral.

DÉDICACE

Ce mémoire est dédié tout particulièrement ***à ma femme et mes enfants***, qui ont été ***les témoins*** de cette passionnante aventure.

Je remercie ma femme, qui à fait preuve de compréhension dans des moments difficiles qui ont jalonné ce travail. Je la remercie également et surtout pour son soutien et sa présence discrète; ce qui a permis à ce mémoire d'aboutir et au projet de se réaliser dans les temps.

TABLES DES MATIÈRES

LISTE DES TABLEAUX

Page

LISTE DES GRAPHIQUES

Page

LISTE DES FIGURES

Page

LISTE DES SYMBOLES ET DES ABRÉVIATIONS

r_j	Date de disponibilité
p_j	Temps de traitement
d_j	Délai de livraison
w_j	Facteur d'importance associé aux coûts
n	Nombre de tâches
m	Nombre de machines
W_j	Temps d'attente
C_j	Temps d'achèvement de la tâche
F_j	Temps de passage de la tâche dans l'atelier
L_j	La Tardiveté de la tâche
T_j	Retard de la tâche
U	Nombre de tâche en retard
S_{ij}	Temps de mise en course
PAPS	Premier arrivé, premier servi
TTPC	Temps de traitement le plus court
TTPL	Temps de traitement le plus long
TTPPC	Temps de traitement pondéré le plus court
TTPPL	Temps de traitement pondéré le plus long
MDM	Marge dynamique minimale
RC	Ratio critique
CAR	Coût apparenté au retard
CARTMC	Coût apparenté au retard avec le temps de mise en course

INTRODUCTION

Les systèmes de base de connaissances représentent une des branches de l'intelligence artificielle. Souvent référés comme des systèmes experts, ils sont largement utilisés dans plusieurs domaines tels que la médecine, la géologie, la chimie, l'ingénierie et la production.

Une base de connaissances est définie comme un système informatisé qui tente d'égaler l'habileté d'un expert dans la prise de décision.

Dans le premier chapitre, nous abordons la problématique de l'ordonnancement comme étant une activité très importante par le rôle et la place qu'elle occupe à l'intérieur d'une organisation. Nous définissons les critères économiques à optimiser dans le processus de prise de décision.

Le chapitre 2 traitera les caractéristiques de l'environnement manufacturier "job shop" et les algorithmes de résolution de problème d'ordonnancement dans les différents contextes.

Dans le chapitre 3 qui consiste l'élément essentiel de notre travail, nous avons proposé une architecture tandem d'un système de base de connaissances pour l'ordonnancement dynamique. Le but est de doter l'utilisateur d'un outil flexible pour la résolution des différents problèmes d'ordonnancement. L'usager aura uniquement à répondre à des questions posées par l'interface du système qui se chargera par la suite à choisir les algorithmes appropriés pour la résolution du problème posé.

Dans le chapitre 4, nous présentons une étude de cas pour la résolution d'un problème d'ordonnancement avec l'utilisation de l'outil que nous avons développé. Dans cette partie, nous essayons de donner un aperçu du mode d'utilisation de ce dernier et de montrer à l'usager une façon d'interpréter les résultats que génère ce système.

Dans la dernière partie, nous résumons les perspectives à venir quant à l'utilisation de la base de connaissances dans les systèmes d'ordonnancement. Le but de notre travail est d'arriver à fournir à l'usager un outil économique et simple à l'utilisation, ne nécessitant pas la

connaissance informatique ni celle de la recherche opérationnelle. Une connaissance de base des systèmes manufacturiers et celle des objectifs à atteindre sont suffisantes.

CHAPITRE 1

LA PROBLÉMATIQUE

1.1 Introduction

À l'intérieur d'une organisation, l'ordonnancement se rapporte à l'établissement du "timing" pour l'utilisation des ressources spécifiques de cette organisation telles que les équipements, les installations et les ressources humaines.

Dans n'importe quelle organisation, l'ordonnancement exige un regard sur la nature de toutes ses activités. Par exemple, les manufacturiers doivent ordonnancer la production, qui implique le développement des calendriers de production pour la main d'œuvre, les machines, les achats, la maintenance et ainsi de suite. Les hôpitaux doivent ordonnancer les admissions, les affectations des infirmières et les services de soutien comme la sécurité, la maintenance et l'entretien. Les établissements de l'éducation doivent planifier les salles de cours, les instructions. Et les secrétaires des avocats, des médecins, des dentistes doivent planifier les rendez-vous.

Dans la hiérarchie de prise de décision, celle de l'ordonnancement est l'étape finale dans le processus de transformation manufacturière. Beaucoup de décisions sur le "design" du système et des opérations sont faites avant la décision de l'ordonnancement. Elles incluent les capacités du système, la sélection des équipements, la sélection et la formation des employés, le "design" des produits et services. En conséquence, les décisions de l'ordonnancement doivent être prises en fonction des contraintes établies par beaucoup d'autres décisions.

Généralement, l'objectif de l'ordonnancement est de trouver un compromis entre des buts conflictuels, qui incluent l'utilisation efficace de l'équipe, des équipements, des installations, et la minimisation du temps d'attente, des inventaires, du temps de traitement et des coûts.

1.2 L'ordonnancement dans un environnement "Job Shop"

L'environnement "job shop" est un système de production où les équipements sont regroupées par types d'opérations et les produits ne suivent pas nécessairement la même séquences de production.

Les caractéristiques d'un tel environnement sont considérablement différentes de celles des systèmes de production à grand volume ou à volume intermédiaire. Les produits sont fabriqués sur commandes, qui habituellement diffèrent en terme de besoins de transformation, de matériels, de temps de fabrication, et de séquence de traitement et de mise en route. À cause de ces circonstances, l'ordonnancement dans un environnement "job shop" est généralement très complexe qui est déterminé par l'impossibilité d'établir des calendriers de production avant de recevoir les commandes.

Le traitement des tâches dans un système de production "job shop" donne aux planificateurs deux issues de base :

1) Comment distribuer la charge de travail aux stations ?
2) Quelle séquence de traitement des tâches utiliser ?

L'ordonnancement concerne la détermination de l'ordre dans lequel les tâches (commandes) seront traitées sur la station de travail. La figure 1.1 montre un processus simple d'ordonnancement dans un atelier.

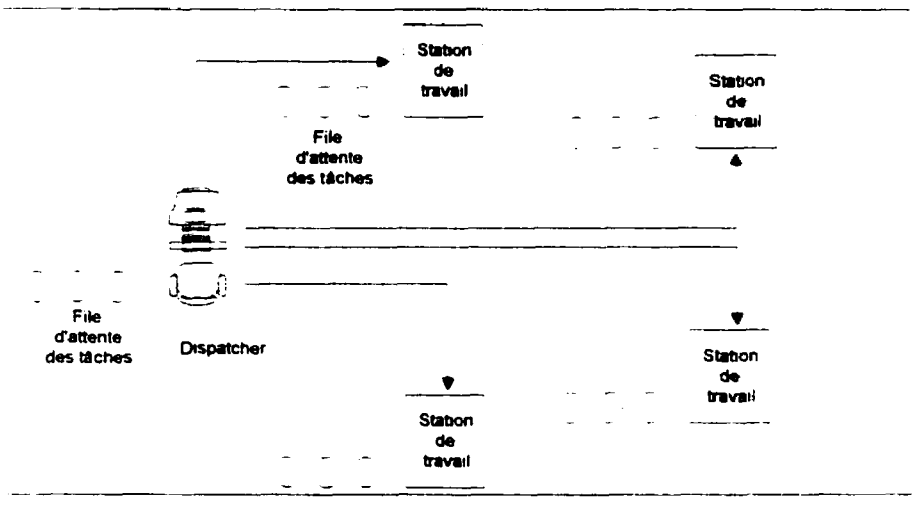


Figure1.1: Processus d'ordonnancement dans un atelier:

Dans un environnement où l'ensemble des commandes à exécuter évoluent dynamiquement et où la brièveté de la procédure d'ordonnancement est un impératif, les techniques de règles de priorité sont abondamment utilisées.

Une règle de priorité est une méthode d'ordonnancement qui, à chaque fois qu'une machine est libre, choisit la prochaine opération à y exécuter parmi l'ensemble de celles qui sont en attente d'y être traitées à l'aide d'une fonction de priorité telles que :

- TTPC temps de traitement le plus court ou,
- DLPR délai de livraison le plus rapproché etc.

Nous utiliserons certaines règles de priorité dans la construction de notre système de base de connaissances pour l'optimisation de la productivité dans un environnement de fabrication "job shop". Elles seront étudiées plus en détail dans le chapitre 2.

Tandis que l'affectation des tâches est définie comme l'assignation des ces dernières aux stations de travail et aux différentes machines qui les constituent, (figure 1.1). Dans le cas où la tâche devrait être traitée uniquement par une station spécifique, la décision d'affectation ne présente pas de grandes difficultés. Cependant, les problèmes surviennent, quand deux ou

plusieurs tâches doivent être traitées et il y a un nombre de stations de travail capables d'exécuter le travail exigé. Dans de tels cas, le gestionnaire des opérations a besoin de méthodes d'assignation des tâches aux stations. Au moment des affectations, les gestionnaires cherchent souvent un arrangement qui minimisera:

- les coûts reliés au traitement des tâches et la mise en course des équipements;
- le temps libre des stations de travail;
- le temps d'achèvement de la tâche, dépendamment de la situation.

L'ordonnancement a pour objet de déterminer l'ordre de traitement des tâches dans les divers centres de travail et dans les stations de travail individuelles à l'intérieur de ces centres. Bien que les décisions d'affectation déterminent les machines ou les stations de travail qui serviraient au traitement des tâches spécifiques, mais elles n'indiquent pas l'ordre dans lequel les tâches en attente devant les centres de travail doivent être traitées.

1.3 Rôle de l'ordonnancement

L'ordonnancement joue un rôle très important dans une organisation. C'est un processus de prise de décision qui existe dans:

- plusieurs systèmes de fabrication et de production;
- les environnements de traitement de l'information;
- le milieu du transport et de la distribution et;
- dans d'autres types d'industrie de service.

L'établissement des cédules de production influence sur les objectifs de l'entreprise. Il est vraiment payant d'investir du temps dans la construction d'un bon calendrier de production plutôt que d'en prendre un arbitrairement.

D'habitude, le choix d'un plan de production a un impact significatif sur la performance du système de production tels que:

- l'efficacité des équipements;
- l'utilisation rationnelle de la main d'œuvre etc.

1.4 La place de l'ordonnancement à l'intérieur d'une organisation

La fonction de l'ordonnancement dans une organisation ou un système est en interface avec plusieurs autres fonctions importantes telles que :

- les méthodes de production;
- la gestion des stocks des matières premières;
- les approvisionnements;
- la production;
- la maintenance des équipements.

Dans ce qui suit, nous décrivons un environnement manufacturier "job shop" et le rôle du processus d'ordonnancement dans un tel environnement.

Dans un système manufacturier, les commandes sont réalisées et sont translatées dans des tâches associées à des dates d'échéance. Ces tâches sont souvent traitées sur des machines dans un centre de travail suivant un ordre ou une séquence. Elles peuvent attendre devant les machines pour être traitées si ces dernières sont occupées. Des préemptions peuvent survenir quand les tâches avec une grande priorité arrivent aux machines et doivent être traitées immédiatement. L'ordonnancement détaillé des tâches qui doivent être traitées dans un système de production est nécessaire pour maintenir l'efficience et le contrôle des opérations.

La fonction de l'ordonnancement est affectée par le processus de planification de la production. Ce processus traite le plan à moyen et à long terme de toute l'organisation. Il doit considérer les niveaux des inventaires, les prévisions et les besoins en ressources à optimiser, le mix-produit et l'allocation des ressources à long terme au niveau supérieur.

L'ordonnancement reçoit aussi les intrants de contrôle du plancher de production. Si des événements inattendus surviennent sur le plancher de production, tels que les pannes de machines ou les temps de traitement plus longs que prévus, ils doivent être pris en compte, parce qu'ils peuvent avoir un impact important sur les plans de production. Les manufactures

modernes ont souvent élaboré leurs systèmes d'informations de fabrication sur place. La figure 1.2, page 9 montre le flux d'informations dans un système manufacturier.

L'usine a un ordinateur central et une base de données centrale. Les ordinateurs personnels, les stations et les terminaux sont connectés en réseau à l'ordinateur central, lesquels sont utilisés pour retrouver les informations de la base de données ou introduire de nouvelles données. Habituellement, la fonction d'ordonnancement est faite sur un ordinateur personnel ou une station reliée à l'ordinateur principal de l'usine. Les terminaux peuvent être connectés à l'ordinateur de l'ordonnancement pour accéder aux départements, à l'information nécessaire pour l'ordonnancement et pour permettre aux départements se pourvoir du système d'ordonnancement avec les informations pertinentes telles que le statut des machines, les changements dans les données de la commande.

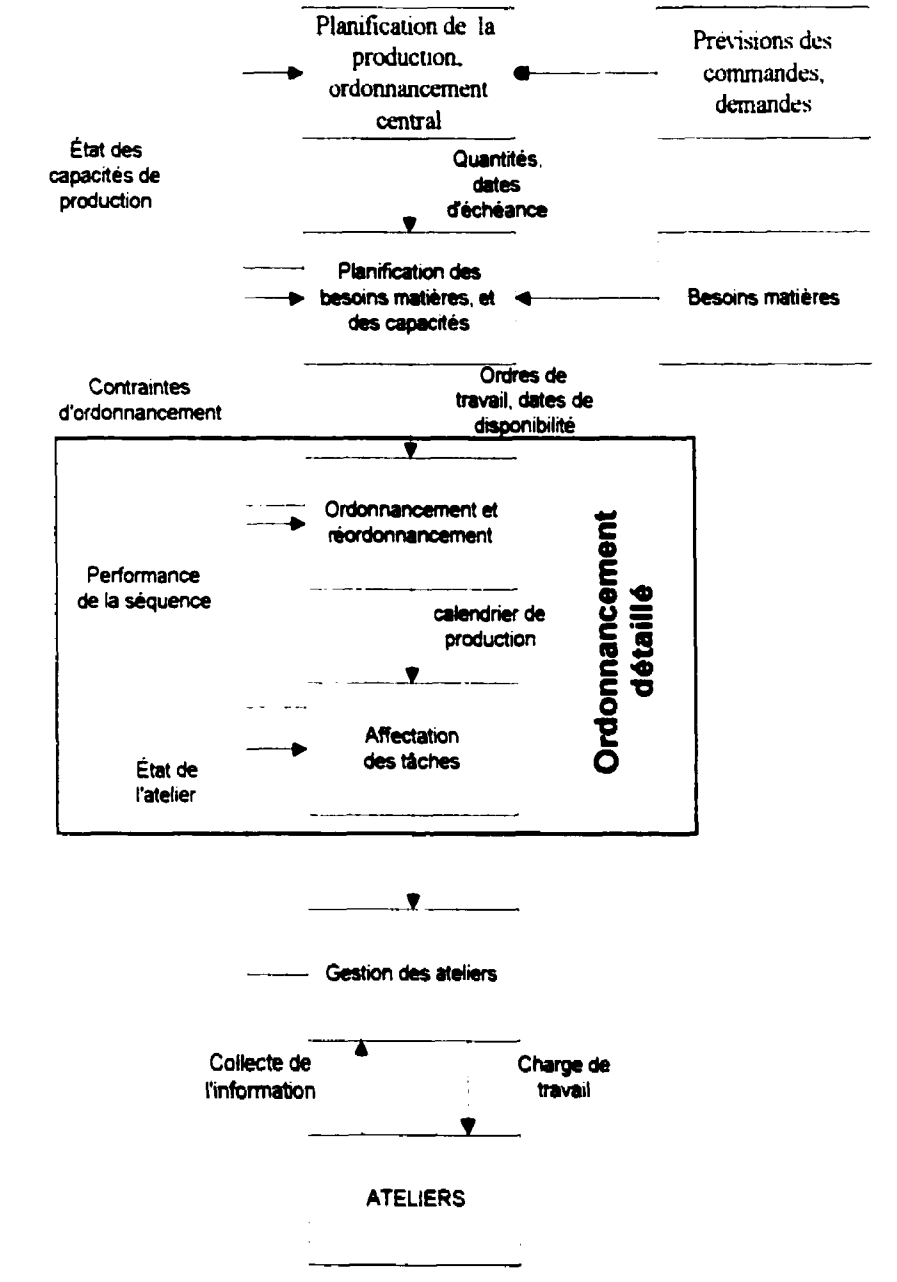


Figure 1.2 : Flux d'informations dans un système manufacturier

1.5 Classification des problèmes d'ordonnancement

Il serait plus commode d'utiliser des notations simples pour représenter les types de problèmes dans un environnement "job shop". Nous classifierons les problèmes selon les 4 paramètres suivants : n/m/A/B. Notation utilisée par French [2]; où

- n: est le nombre de tâches;
- m: est le nombres de machines;
- A: décrit le modèle de flux ou la discipline;
- F: pour le cas de "flow shop", i.e. l'ordre des machines est le même ;
- P: pour le "flow shop" de permutation, i.e. l'ordre des machines et des tâches sont les mêmes, et aucun dépassement des tâches n'est permis;
- G: le cas de "job-shop" où il n'y a pas des restrictions sur la forme des contraintes technologiques;
- B: décrit les mesures de performance ou fonctions économiques par lesquelles l'ordonnancement est évalué.

Exemple 1-1:

$7/2/G/C_{max}$

7: nombre de tâches ;

2 : nombre de machines ;

G : problème de « job-shop » ;

C_{max} : représente la fonction objective à optimiser « Minimisation du « makespan ».

1.6 Critères économiques dans la prise de décision

Il n'est pas facile de statuer sur nos objectifs de l'ordonnancement. Il y a beaucoup d'entre eux qui sont complexes et souvent en conflit. Mellor (1966) a listé 27 objectifs d'ordonnancement distincts [2]. Nous essayons d'être très exhaustifs. Alors nous indiquerons dans nos termes généraux quelques critères par lesquels nous baserons nos jugements sur nos succès. Ces critères seront suffisants pour motiver les définitions mathématiques de nos mesures de performances. Ces critères de performance sont résumés dans le tableau 16 de la

page 61 et seront utilisés dans le processus de la prise de décision quant au choix du calendrier de production optimal.

Nous préférons prendre comme critère, les dates de livraison des produits. Autrement dit, il est préférable de prendre les pénalités de retard qui sont très importants comme indicateur de performance. Nous essayons de minimiser la période totale de l'ordonnancement parce que, une fois toutes les tâches ont été traitées, les machines peuvent être prêtes pour traiter d'autres. Nous essayons de minimiser le temps libre des machines. Ce qui signifie que le capital investi n'est pas entièrement utilisé. Nous essayons aussi de minimiser les coûts des inventaires reliés aux coûts de stocks des matières premières et ceux des en-cours de production qui sont contraints d'attendre devant les machines. Nous pourrions aussi assurer un taux uniforme d'activité à travers la période de l'ordonnancement.

Avant de définir les mesures de performance dans les termes mathématiques précis, nous avons besoin de plusieurs définitions et de notations qui sont:

r_j : temps auquel la tâche j sera disponible ;

p_{ij} : temps de traitement de la tâche j sur la machine i ;

d_j : date d'échéance j, i.e. date de livraison prévue ;

aj : allocation pour la tâche j. C'est la période allouée pour traiter la tâche j entre la date de disponibilité et la date d'échéance.

$$a_j = d_j - r_j \tag{1.1}$$

W_{jk} : temps d'attente de la tâche j à la $k^{ième}$ position ;

W_j : temps d'attente total de la tâche j ;

$$W_j = \sum_{k=1}^{m} W_{jk} \tag{1.2}$$

C_j : temps d'achèvement de la tâche j ;

$$C_j = r_j + \sum_{k=1}^{m} (W_{jk} + p_{ij(ki)}) \tag{1.3}$$

F_j : temps de passage de la tâche j dans l'atelier.

$$F_j = C_j - r_j = \sum_{k=1}^{m} (W_{jk} + p_{hj(k)}) \qquad (1.4)$$

L_j : la tardiveté de la tâche j. C'est la différence entre le temps d'achèvement et le délai de livraison de la tâche j.

$$L_j = C_j - d_j \qquad (1.5)$$

Note:

- Quand la tâche j a achevé son traitement avant sa date de livraison, elle est en avance et L_j est négatif ($L_j < 0$).
- La tâche j est en retard, quand elle achève son traitement après sa date de livraison et L_j est positif ($L_j > 0$).

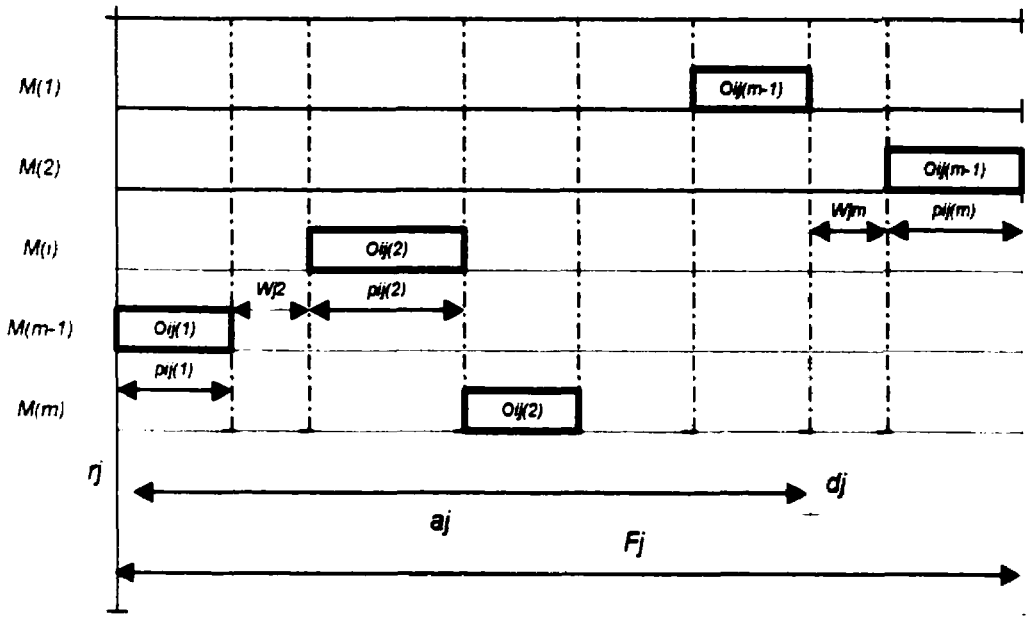
T_j : le retard de la tâche j;

$$T_j = \max(L_j, 0) \qquad (1.6)$$

E_j : l'avance de la tâche j;

$$E_j = \max(-L_j, 0) \qquad (1.7)$$

Ces quantités pour une tâche typique sont représentées par le diagramme de Gantt et sont illustrées dans le graphique 1 suivant.



Graphique 1: Diagramme de Gantt pour une tâche (j) typique.

- Oij(m) représente l'ordre de traitement de la tâche j sur la machine i à la kième position.
- L'ordre de traitement est donné par les contraintes technologiques $M_{(m-1)}$, M_1, M_m, M_1, M_2.
- Les temps d'attente W_{j1} et W_{j3} sont égaux à zéro, par contre, W_{j2} et W_{jm} sont différents de zéro.
- Pour cette tâche j, Tj = Lj i.e. que la tâche j a complété son traitement après la date d'échéance dj.

$$C_j - d_j > 0 \Rightarrow E_j = 0$$

1.6.1 Critères basés sur le temps d'achèvement de la tâche

Les principaux critères rattachés à cette catégorie sont:

F_{total} : le temps de passage total des tâches *j* dans l'atelier;

$$F_{total}=\sum_{j=1}^{n}(C_j-r_j) \tag{1.8}$$

C_{max} : la durée de temps maximal pour l'exécution des tâches j;

$$C_{max}=\max(C_j) \tag{1.9}$$

F_{moyen} : le temps de passage moyen de la tâche j dans l'atelier;

$$F_{moyen}=\frac{\sum_{j=1}^{n}(C_j-r_j)}{n} \tag{1.10}$$

C_{moyen} : la durée de l'intervalle de temps moyen nécessaire pour l'achèvement des tâches j.

$$C_{moyen}=\frac{\sum_{j=1}^{n}C_j}{n} \tag{1.11}$$

La minimisation de F_{total} implique essentiellement la minimisation du coût de l'ordonnancement apparenté directement au temps nécessaire pour le traitement de toutes les tâches. Celle de C_{max} implique la minimisation du coût de l'ordonnancement lequel dépend de combien de temps le système a consacré pour traiter la tâche la plus longue. Dans le cas où toutes les tâches seraient disponibles ($r_j=0$), C_{max} et F_{max} sont égaux. Par contre, quand les dates de disponibilités des tâches sont différentes, C_{max} et F_{max} sont plutôt distincts.

La minimisation du temps moyen de passage F_{moyen} implique que le coût de l'ordonnancement est directement lié au temps moyen nécessaire pour le traitement de la tâche. La minimisation du temps moyen d'achèvement du traitement de la tâche j, C_{moyen} est équivalent à F_{moyen}.

Nous considérons les mesures pondérées. Nous reconnaissons que certaines tâches sont plus importantes que d'autres. Alors nous suggérons de minimiser la somme pondérée telle que:

Coût total de l'ordonnancement

$$\sum_{j=1}^{n}w_j*C_j \tag{1.12}$$

Coût total des retards

$$\sum_{j=1}^{n} w_j * T_j \tag{1.13}$$

où: w_j est le facteur de pondération, généralement associé à un coût.

1.6.2 Critères basés sur les délais de livraison

Étant donné que le coût de l'ordonnancement est habituellement apparenté de combien nous manquons les dates prévues de livraison, évidemment les mesures de performances sont:

Somme des retards :

$$\sum_{j=1}^{n} T_j = \sum_{j=1}^{n} [(C_j - d_j) > 0] \tag{1.14}$$

Retard moyen des tâches :

$$\frac{\sum_{j=1}^{n} T_j}{n} \tag{1.15}$$

Temps moyen des retards :

$$\frac{\sum_{j=1}^{n} T_j}{n} \tag{1.16}$$

Il y a des pénalités encourues quand la tâche est en retard. L'objectif raisonnable sera de minimiser le nombre de tâches en retard ou le coût moyen des retards, soient:

Nombre de tâches en retard :

$$\sum_{j=1}^{n} U_j = \begin{cases} 1 \text{ si } C_j - d_j > 0 \\ 0 \text{ autrement} \end{cases} \tag{1.17}$$

Coût moyen des retards :

$$\frac{\sum_{j=1}^{n} w_j * T_j}{\sum_{j=1}^{n} U_j} \tag{1.18}$$

Où : U_j est la tâche en retard.

1.6.3 Critères basés sur les coûts d'inventaire et d'utilisation

Ici, nous désirions minimiser *Nw(moyen)*, le nombre moyen des tâches en attente devant les machines ou *Nu(moyen)*, le nombre moyen des tâches inachevées. Ces deux mesures sont apparentées aux coûts des encours. Nous pouvions être plus concernés par la minimisation de nombre des tâches qui ont complété leur traitement, *Nc(moyen)*, car en général, nous réduisons les coûts d'inventaire des produits finis. Si notre but était de s'assurer de l'utilisation efficace des équipements, alors nous choisissions de maximiser *Np(moyen)*, le nombre moyen des tâches actuellement prêtes à être traitées à n'importe quel instant, ou de minimiser le temps libre des machines.

Temps libre des machines :

$$C_{max} - \sum_{j=1}^{n} p_j \tag{1.19}$$

Finalement, nous introduisons certaines variables qui comptent le nombre de tâches dans diverses situations à n'importe quel instant t.

Nw(t) est le nombre de tâches en attente devant les machines ou non disponibles pour être traitées à l'instant t.

Np(t) est le nombre de tâches actuellement en cours de traitement à l'instant t.

Nc(t) est le nombre de tâches qui ont complété leur traitement à l'instant t, et

Nu(t) est le nombre de tâches qui n'ont pas encore achevé leur traitement à l'instant t.

Alors le nombre total et le nombre moyen de tâches dans le système sont donnés par les relations suivantes:

Nombre total des tâches dans le système :

$$Nw(t) - Np(t) - Nc(t) - Nu(t) \tag{1.20}$$

Nombre moyen des tâches dans le système :

$$\frac{\sum_{j=1}^{n} C_j}{C_{max}} \qquad (1.21)$$

1.7 Conclusion

Nous pouvons conclure, de ce qui a été cité dans ce chapitre, que l'ordonnancement est une tâche très complexe et sa fonction est très importante dans tout système de production. Nous déduisons aussi que l'activité de l'ordonnancement est un processus de prise de décision qui concerne la détermination de l'ordre dans lequel les tâches seront traitées sur les stations de travail ou les machines dans le but d'optimiser un certain objectif économique.

La construction d'un système d'ordonnancement dynamique basé sur la connaissance dépend étroitement :

- de l'environnement manufacturier où nous évoluons; et
- aux critères préétablis par l'entreprise dans le but d'optimiser les objectifs économiques visés par cette dernière.

Le choix d'un tel outil a un impact très significatif sur la performance du système de production, et c'est dans ce but que nous avons défini les différents critères de performance que nous utiliserons pour évaluer les cédules de production générées par notre système.

De nos jours, dans un environnement manufacturier, l'ordonnancement doit être dynamique pour s'adapter aux changements des facteurs de production intervenant dans le système. Parmi les facteurs qui rendent le système dynamique, nous pouvons citer entre autres les suivants :

- l'arrivée aléatoire des commandes dans le système,
- l'adaptation des ressources suivant la fluctuation de la demande,
- la diversité des routages des produits,
- les pannes de machines qui peuvent causer un changement de priorité des commandes,
- les commandes imprévues,

- les commandes spécifiques.

Cependant, le chapitre qui suit traitera, en détail, un environnement manufacturier "job shop" et les critères de base pour la construction du système de l'ordonnancement à base de connaissances.

CHAPITRE 2

CARACTÉRISTIQUES DE L'ENVIRONNEMENT MANUFACTURIER "JOB SHOP"

2.1 Introduction

La révolution industrielle, où la production de masse dominait, a été révolue par les exigences accrues du marché qui sont les suivantes :

- mettre à la disposition du consommateur une diversité de produits de bonne qualité ;
- à moindre coût ;
- et dans le délai prévu.

Ces dernières ont poussé les manufacturiers à repenser leur politique de production. Le facteur le plus important qui a favorisé l'immergence de l'environnement manufacturier "job shop" était la production de petites séries engendrées par la philosophie du juste à temps, c'est à dire fabriquer des produits variés au moment de la réception de la commande.

Durant la dernière décennie, beaucoup de nouvelles technologies (robot, machines à commande numérique "CNC", systèmes de fabrication flexibles "FMS", etc.) ont été utilisées pour l'amélioration de la productivité sur le planché des usines. L'implantation de ces nouvelles technologies a créé de nouvelles classes de systèmes manufacturiers. Ces derniers sont complexes et requièrent un niveau très élevé de planification et de contrôle de la production pour être capable d'atteindre un niveau de performance élevé en terme de qualité et de quantité de produits. Dans un tel environnement, l'ordonnancement de la production est devenu aujourd'hui le centre névralgique du système de la planification et de contrôle de la production.

2.2 Problèmes d'ordonnancement

L'environnement "job shop" est caractérisé par la fabrication des produits sur commandes, qui différent en terme de besoins de transformation, de matériels, de temps de fabrication, de séquence de traitement et de mise en route. Cette séquence de traitement est en fonction des contraintes technologiques qui exigent que chaque tâche doit être traitée à travers les machines dans un ordre particulier. Mais dans un problème de "job shop" généralisé, il n'y a pas de restrictions sur ces formes de contraintes, car chaque tâche a son propre ordre de traitement.

Généralement un système de production "job shop" est une organisation fonctionnelle dans laquelle les départements ou les centres de travail sont organisés autour d'un même type d'équipement ou opération tels que, par exemple, le tournage, le fraisage, l'ajustage, l'affûtage etc. (figure 2.2, page 23)

"Notre problème d'ordonnancement est d'identifier la séquence optimale de **n** tâches (commandes) qui seront traitées sur **m** machines de même type regroupées en département de fabrication (ou centre de travail) avec laquelle le but de production sera atteint". Dans ce cas, l'évaluation montre qu'il a **n!** de combinaisons possibles de séquences d'opération [4].

Le problème d'ordonnancement dans un environnement "job shop" est complexe, non seulement par le nombre de possibilités de combinaisons de séquences de tâches, mais aussi par le fait que l'efficacité des machines est différente en fonction des opérations effectuer. Le temps des opérations des tâches sur chaque machine est différent et la priorité varie d'une tâche à l'autre.

Dans le problème d'ordonnancement "job shop" , la séquence optimale des tâches peut être trouvée quand :

- le temps total de production ou "makespan" correspondant au coût minimal est atteint;
- l'efficacité des ressources est élevée ;

- le délai de livraison des commandes est minimal;
- la priorité des tâches est respectée;
- le nombre de tâche en retard est minimal;
- le temps de passage des tâches dans l'atelier est minimal;
- le temps de mise en course des équipements correspondant au coût minimal est atteint.

L'approche classique de l'ordonnancement dans un environnement « job shop » se concentre sur les six éléments suivants:

- Les types d'arrivée des tâches (produits);
- Le nombre et la variété des machines dans l'atelier ;
- Le ratio homme-machine dans l'atelier;
- Les types de flux de produit à travers l'atelier;
- Les règles de priorité pour allouer les tâches aux machines;
- Le critère d'évaluation du calendrier de production.

Pour modéliser un problème d'ordonnancement, il y a lieu de comprendre l'importance des facteurs intervenant dans tout environnement manufacturier et qui sont:

2.2.1 Types d'arrivée des tâches

Les tâches peuvent arriver dans l'atelier, soit en lot, ou soit en pièce par pièce dans un intervalle de temps qui suit une distribution statistique. Le premier type d'arrivée des tâches est appelé statique et le second dynamique. L'arrivée statique des tâches ne signifie pas que les commandes sont placées par le client au même moment, mais elles sont assujetties d'être ordonnancées en même temps. Tandis que dans le cas dynamique, les tâches sont ordonnancées suivant leur arrivée.

2.2.2 Nombre et variété de machines

Le nombre de machines dans l'atelier affecte manifestement le processus de l'ordonnancement. S'il y a une seule machine ou un groupe de plusieurs machines qui peuvent être traitées comme une seule machine, le problème de l'ordonnancement est beaucoup plus facile. Dans le cas où le nombre et la variété des machines augmenteraient, le problème de l'ordonnancement devient vraisemblablement plus complexe.

2.2.3 Ratio homme-machine

S'il y a plus d'opérateurs que de machines ou le nombre de ces derniers est égale au nombre de machines, alors l'atelier se rapporte comme un système à capacité limitée. Autrement, si le nombre de machines est supérieur à celui des opérateurs, alors il est considéré comme un système à ressource limitée. En pratique, le premier critère concerné est l'utilisation d'un opérateur sur plusieurs machines et la détermination de la meilleure façon de les allouer, exemple l'aménagement en U.

2.2.4 Types de flux des produits

En générale, nous distinguons deux types de flux de produit à travers l'atelier. Le premier appelé "flow shop", dans lequel les tâches suivent le même chemin d'une machine à l'autre, exemple d'une chaîne de montage (figure 2.1, page 23). Alors que dans le second flux appelé "job shop" le mouvement des tâches d'une machine à l'autre ne se fait pas dans la même séquence de traitement et les équipement sont regroupés par type d'opération (figure 2.2, page 23).

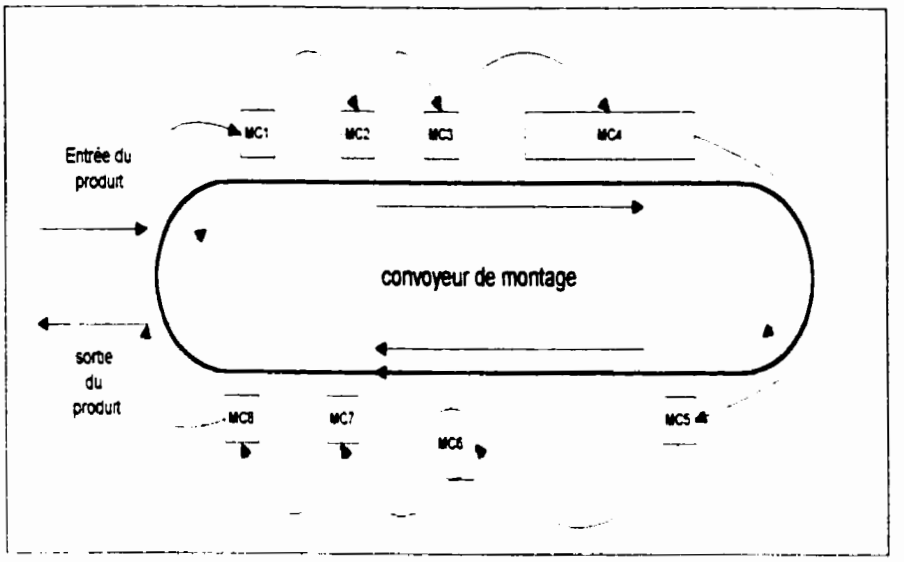


Figure 2.1: Flux de produits dans un environnement "Flow Shop"

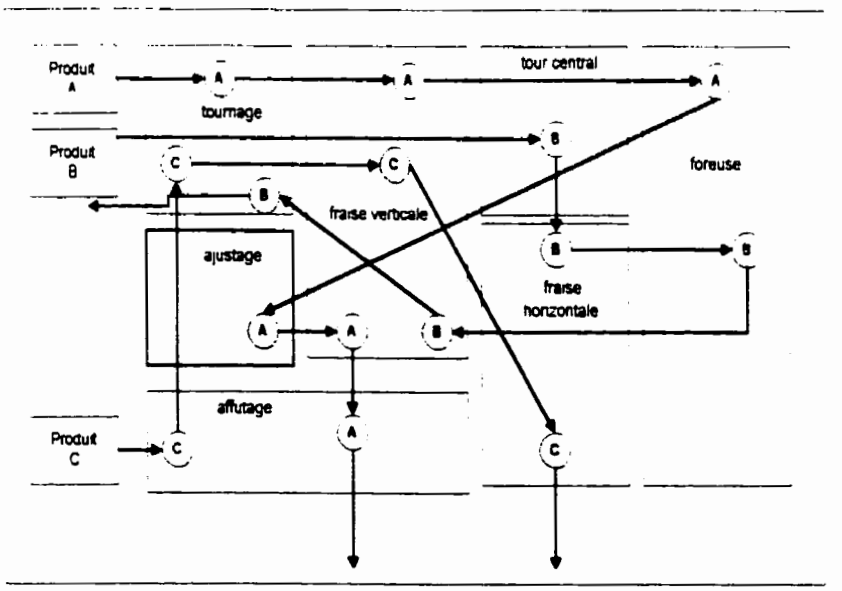


Figure 2.2: Flux de produits dans un environnement "Job Shop"

2.3 Règles de priorités et algorithmes

Dans le mécanisme d'ordonnancement suggéré dans notre document, des règles d'affectation sont utilisées pour planifier les tâches. Quand une machine devient disponible pour le traitement, une tâche qui peut être traitée sur cette machine doit être assignée à cette dernière. Si deux ou plusieurs tâches sont prêtes pour le traitement sur la même machine, une de ces dernières sera sélectionnée pour être traitée en premier par une règle d'ordonnancement qui définira la priorité des tâches.

Plusieurs heuristiques et algorithmes ont été développés pour la résolution des problèmes d'ordonnancement dans un environnement "job shop" [2]. Certains d'entre eux seront utilisés dans la construction de notre base de connaissances pour résoudre des problèmes déterministes car les temps de traitement, l'arrivée des tâches et d'autres paramètres sont considérés connus et fixés à l'avance.

Le choix des algorithmes de résolution du problème d'ordonnancement sera défini en fonction du nombre de machines constituant le système de production et des critères économiques à optimiser.

Le mécanisme de l'ordonnancement en temps réel basé sur la connaissance suggérée dans ce document, inclue 16 règles d'affectation qui sont résumées dans le tableau 15 de la page 60.

2.3.1 Algorithmes de résolution d'un problème d'ordonnancement d'une seule machine

Les modèles conçus pour résoudre le problème d'une seule machine sont importants pour plusieurs raisons. Chaque modèle a été développé dans un objectif bien défini. Il y a ceux qui visent l'optimisation du temps d'achèvement des tâches, ou la minimisation du makespan, d'autres qui visent la minimisation des retards ou du nombre de tâches en retard. L'environnement manufacturier composé d'une seul machine est un cas spécial de tous les autres environnements. Dans la pratique, les problèmes d'ordonnancement dans des

environnements complexes sont souvent décomposés en sous problèmes qui seront traités avec des modèles d'optimisation utilisés pour une seule machine.

Généralement, lorsque les tâches n'arrivent pas en même moment dans le système, la règle de priorité utilisée pour résoudre un tel problème est celle du premier arrivé, premier servi.

2.3.1.1 Premier arrivée, premier servi ("PAPS"):

Les tâches (commandes) sont traitées sur la machine suivant leur ordre d'arrivée. L'algorithme stipule que: *la séquence optimale est obtenue en classant ces dernières dans l'ordre croissant des dates de disponibilité r_j de ces dernières: $Min(r_j)$* [1].

Exemple 2-2 :

Tableau I

Temps de disponibilité des tâches

Tâches (j)	Temps de disponibilité			
	1	2	3	4
r_j	0	1	3	2

La séquence des tâches à traiter sur la machine est : 1 2 4 3

Quand, toutes les tâches sont disponibles en même moment dans le système, alors les règles de priorité suivantes sont utilisées pour résoudre le problème d'ordonnancement.

2.3.1.2 Temps de traitement le plus court en premier ("TTPC")

Nous traitons en premier la tâche qui a le temps de traitement le plus court, la suivante celle qui a le second temps le plus court et ainsi de suite. [1]; [2]; [6].Alors, *nous classons les tâches dans l'ordre croissant de leurs temps de traitement p_j: $Min\ (p_j)$*

Exemple 2.2

Tableau 2

Le temps de traitement des tâches

Tâches (j)	Temps de traitement			
	1	2	3	4
p_j	4	5	8	2

L'ordre croissant des p_j est : p_4, p_1, p_2, p_3.

La séquence optimale est : 4,1,2,3

2.3.1.3 Temps de traitement le plus long en premier ("TTPL")

La séquence optimale est obtenue en classant les tâches dans l'ordre décroissant des temps de traitement p_j $Max\ (p_j)$ [1]; [2] [6].

Exemple 2.3

Nous considérons les données du tableau 2 et nous appliquons la règle TTPL.

1. L'ordre décroissant des p_j est : p_3, p_2, p_1, p_4.
2. La séquence optimale est : 3,2,1,4.

2.3.1.4 Temps de traitement pondéré le plus court en premier ("TTPPC")

Cette règle fournie une séquence optimale pour un problème de $1 \mid\mid \Sigma w_jC_j$, dont l'objectif est de minimiser la somme pondérée des temps d'achèvement des tâches j, où le w_j correspond au facteur d'importance et le C_j est le temps d'achèvement de la tâche j.

La séquence optimale est obtenue en classant les tâches dans l'ordre décroissant du rapport $\frac{w_j}{p_j}$: $Max(\frac{w_j}{p_j})$ [7].

Exemple 2.4 :

Tableau 3

Le temps de traitement,
le poids et le ratio des tâches

Tâches (j)	1	2	3	4
p_j	8	4	8	15
w_j	4	4	2	5
$\frac{w_j}{p_j}$	0,50	1,00	0,25	0,33

En classant les tâches dans l'ordre décroissant des $\frac{w_j}{p_j}$, nous obtenons la séquence optimale 2,1,4,3.

2.3.1.5 Temps de traitement pondéré le plus long en premier ("TTPPL")

La séquence optimale est obtenue en classant les tâches dans l'ordre croissant du rapport $\frac{w_j}{p_j}$: $Min(\frac{w_j}{p_j})$ [3].

Exemple 2.5

Nous considérons les mêmes données du tableau 3, et nous appliquons la règle TTPPL. En classant les tâches dans l'ordre croissant des $\frac{w_j}{p_j}$, nous obtenons la séquence optimale 3,4,1,2.

Les règles d'ordonnancement utilisées pour répondre aux délais de livraison des clients et de minimiser les retards des tâches sont les suivantes:

2.3.1.6 Date d'échéance au plus tôt ("DÉPT")

Cette règle de priorité consiste à classer les tâches dans l'ordre croissant des dates d'échéance d_j. La tâche qui a le délai de livraison le plus rapproché passe en premier sur la machine, la suivante est celle qui a la deuxième date d'échéance la plus rapprochée et ainsi de suite: $Min(d_j)$ [2]; [7].

Exemple 2.6 :

Nous considérons les délais de livraison de 4 tâches tels que illustrés dans le tableau 4.

Tableau 4 .

Délai de livraison

Tâches (j)	Délai de livraison			
	1	2	3	4
d_j	10	6	3	15

L'ordre décroissant des d_j est : d_3, d_2, d_1, d_4.

La séquence optimale est : 3,2,1,4.

2.3.1.7 Marge dynamique minimale ("MDM")

La séquence est obtenue en classant les tâches dans l'ordre croissant de la marge minimale: $Min(d_j - p_j - t)$ [1] [3] [4].

Exemple 2.7 :

Supposons que $t = r_j = 0$, cela veut dire que toutes les tâches sont disponibles à l'instant t.

Le temps de traitement, le délai de livraison et la marge sont illustrés dans le tableau 5.

Tableau 5.

Temps de traitement ,délai de livraison et marge

Tâches (j)	1	2	3	4
p_j	8	4	8	15
d_j	8	6	12	18
d_j-p_j	0,00	2,00	4,00	3,00

En classant les tâches dans l'ordre croissant des d_j-p_j, nous obtenons la séquence optimale 1,2,4,3

2.3.1.8 Ratio critique minimal de la marge dynamique (RCMMD)

Les tâches sont sélectionnées en fonction de l'ordre croissant des ratios critiques [1]; [3]:

$RC1=\frac{(d_j-p_j)}{p_j}$, i.e. la marge par le temps de traitement restant ; (2.1)

$RC2=\frac{(d_j-p_j)}{d_j}$, i.e. la marge par le délai de livraison restant ; (2.2)

$RC3=\frac{d_j}{p_j}$, i.e. le délai de livraison par le temps de traitement. (2.3)

Exemple 2.8 :

Nous supposons les mêmes données du tableau 5 et en appliquant les équations (2.1), (2.2) et (2.3), nous obtenons les ratios critiques représentés dans le tableau 6.

Tableau 6

Ratio critique de la tâche (j)

Tâches (j)	Ratio Critique			
	1	2	3	4
p_j	8	4	8	15
w_j	4	4	2	5
$\frac{w_j}{p_j}$	0,50	1,00	0,25	0,33

La séquence obtenue en appliquant les 3 règles du ratio critique est la même, soit 3,4,1,2

2.3.1.9 Coût apparenté au retard ("CAR")

Sélectionner la tâche avec le coût apparenté au retard (CAR) le plus élevé, i.e. sélectionner la tâche avec la valeur maximale de l'indice de rangement I_j [4]:

$$I_j=\frac{w_j}{p_j}\exp(-\frac{\min(d_j-p_j-t,0)}{k\bar{p}} \tag{2.4}$$

Où $\bar{p}$ le temps de traitement moyen

t dénote le temps auquel une tâche sera sélectionnée pour être traitée sur une machine disponible ;

k est appelé multiplicateur d'ajustement;

$$k=\begin{cases}4.5+R, si R\le 0.5\\ 6-2R, si R>0.5\end{cases} \tag{2.5}$$

$$R=\frac{(d_{\max}-d_{\min})}{C_{\max}} \tag{2.6}$$

d_{max}: date d'échéance au plus tard ;

d_{min}: date d'échéance au plus tôt.

$$C_{\max}=r_j+\sum_{j=1}^{n}p_j \tag{2.7}$$

$$\overline{p}=\frac{\sum_{j=1}^{n}p_j}{n} \tag{2.8}$$

Exemple 2.9 :

Nous supposons dans notre exemple que toutes les tâches sont disponibles, alors $r_j = 0$, et les données sont représentées dans le tableau 7.

Tableau 7

Temps de traitement, délai de livraison ,
poids et ratio de chaque tâche (j)

Tâches (j)	1	2	3	4
d_j	10	8	7	20
p_j	8	4	8	15
w_j	4	4	2	5
$\frac{w_j}{p_j}$	0,50	1,00	0,25	0,33

Itération 1 : Début de l'itération à $t=0$

Étape 1: *Calculer le C_{max} avec l'équation (2.7), pour n = 4 tâches.*

$C_{max} = 35$

Étape 2: *Calculer le facteur R avec l'équation (2.6), avec $d_{max} = 20$ et $d_{min} = 7$*

$R = 0.37$

Étape 3: *Calculer le paramètre k avec l'équation (2.5).*

puisque $R < 0.5$, alors $k = 4.5 - R$

$k = 4.87$

Étape 4: *Calculer le temps de traitement moyen avec l'équation (2.8), pour n=4*

$\bar{p}=8.75$

Étape 5: *Évaluer I_j pour chaque tâche j en appliquant l'équation (2.4).*

$$I_1=0.5\exp(-\frac{\min(10-8-0,0)}{4.87*8.75})=0.5$$

$$I_2=1\exp(-\frac{\min(8-4-0,0)}{4.87*8.75})=1$$

$$I_3=0.25\exp(-\frac{\min(7-8-0,0)}{4.87*8.75})=0.24$$

$$I_4=0.3\exp(-\frac{\min(20-15-0,0)}{4.87*8.75})=0.3$$

Alors la tâche 2 est sélectionnée pour aller en premier sur la machine.

Itération 2: Calculer t, $t = p_2 - t_0 = 4$

Et retourner à l'étape 5

La séquence optimale est obtenue après 4 itérations : 2,1,4,3

2.3.1.10 Coût apparenté au retard et temps de mise en course ("CARTMC")

Cette heuristique a été utilisée pour résoudre le problème d'ordonnancement dans le secteur de la microélectronique où l'atelier est composé d'une ligne de fabrication complètement automatisée pour la fixation de la puce sur le substrat. La contrainte est la variété du mix-produit qui arrive dans le système. Ce dernier exige souvent le réglage des équipements qui entraîne des temps de mise en course considérables. Alors l'objectif d'optimisation est le suivant:

1. minimiser la somme des retards, $\sum_{j=1}^{n} T_j$
2. minimiser la somme des temps de mise en course,
3. minimiser la somme pondérée du temps d'achèvement des tâches. $\sum_{j=1}^{n} w_j * C_j$

La séquence optimale est obtenue en passant par les étapes suivantes :

Étape 1: *Calculer l'indice de rangement I_j de chaque tâche j.*

Étape 2: *Classer les tâches dans l'ordre décroissant des indices de rangement calculés dans l'étape précédente.*

L'indice de rangement est calculé par l'équation suivante :

$$I_j(l,t)=\frac{w_j}{p_j}\exp\left(-\frac{(d_j-p_j-t,0)}{k_1\bar{p}}\right)\exp\left(-\frac{s_{jl}}{k_2\bar{s}}\right) \qquad (2.9)$$

l représente les tâches restantes qui ne sont pas encore ordonnancée,

t représente l'instant où la machine est libérée.

$$t=p_j+s_{jl} \qquad (2.10)$$

s_{jl} est le temps de mise en course quant la tâche l suit la tâche j.

k_1 et R sont des facteurs d'ajustement et sont calculés avec les équations (2.5) et (2.6).

Makespan est:

$$C_{max}=\sum_{j=1}^{n} p_j+n^*\bar{s} \tag{2.11}$$

Le temps de mise en course moyen est:

$$\bar{s}=\frac{\sum_{j=1}^{n} s_{0j}}{n} \tag{2.12}$$

s_{0j} est le temps de réglage de la machine pour traiter chaque tâche j à l'instant t_0.

Le délai de livraison moyen des tâches est:

$$\bar{d}=\frac{\sum_{j=1}^{n} d_j}{n} \tag{2.13}$$

$$\eta=\frac{\bar{s}}{\bar{p}} \tag{2.14}$$

$$\tau=1-\frac{\bar{d}}{C_{max}} \tag{2.15}$$

$k_2=\frac{\tau}{2\sqrt{\eta}}$ est le facteur d'ajustement (2.16)

Exemple 2.10 :

Les données de notre problème sont résumées dans les 3 tableaux suivants:

Tableau 8

Temps de traitement, date de livraison et facteur d'importance

Tâches (j)	1	2	3	4
p_j	13	9	13	10
d_j	12	37	21	22
w_j	2	4	2	5

Tableau 9

Temps de mise en course de la première tâche dans la séquence

Temps de mise en course à t_0	Tâches (j)			
	1	2	3	4
s_{0j}	1	1	3	4

Tableau 10

Temps de mise en course de la séquence dépendante des tâches

Tâches (j)	1	2	3	4
s_{1j}	_	4	1	3
s_{2j}	0	_	1	0
s_{3j}	1	2	_	3
S_{4j}	4	3	1	_

En utilisant la règle CARTMC, le temps de traitement moyen $\bar{p}$ et le temps moyen de réglage des machines $\bar{s}$ sont respectivement 11 et 2.

En appliquant les équations (2.11), (2.5), (2.14), (2.15), (2.16), (2.6) nous obtenons les quantités suivantes:

- Le C_{max} estimé est de : 45 + 4*2 = 53
- $R = \frac{25}{53} = 0.47$
- $\tau = 1 - \frac{25}{53} = 0.57$
- $\eta = \frac{2}{11} = 0.18$
- $k_1 = 5$ et $k_2 = 0.7$

Et pour déterminer, laquelle des tâches vient en premier dans la séquence, on calcule $I_j(0,0)$ pour toutes les tâches j = 1.... ,4.

$$I_1(0.0)=\frac{2}{13}\exp(-\frac{\min(12-13-0.0)}{55})\exp(\frac{1}{1.4})=0.075$$

$$I_2(0.0)=\frac{4}{9}\exp(-\frac{\min(37-9-0.0)}{55})\exp(\frac{1}{1.4})=0.131^{***}$$

$$I_3(0.0)=\frac{2}{13}\exp(-\frac{\min(21-13-0.0)}{55})\exp(\frac{1}{1.4})=0.016$$

$$I_4(0.0)=\frac{5}{10}\exp(-\frac{\min(22-10-0.0)}{55})\exp(\frac{1}{1.4})=0.016$$

Alors, la tâche 2 a la plus grande priorité. Après les 4 itérations, nous obtenons la séquence optimale suivante : 2,4,3,1

2.3.2 Algorithme de résolution du problème de deux machines

La règle d'ordonnancement la plus populaire pour la résolution du problème de deux machines est la méthode de Johnson [2], [3]. L'objectif de cette approche, est la minimisation du temps de passage des tâches dans l'atelier, du commencement de la première tâche jusqu'à la fin de la dernière.

2.3.2.1 Algorithme de Johnson

La règle est constituée par les étapes suivantes:

Étape 1: *Lister le temps d'opération de chaque tâche sur les deux machines;*

- p_{1j} le temps de traitement des tâches j sur la machine 1,
- p_{2j} le temps de traitement des tâches j sur la machine 2.

Étape 2: *Diviser l'ensemble des tâches en deux sous-ensembles E1 et E2 ;*

- E1 contient les tâches avec le temps de traitement sur la première machine p_{1j} inférieur à celui de la deuxième machine p_{2j}.
- E2 contient les tâches avec le temps de traitement sur la première machine p_{1j} supérieur à celui de la deuxième machine p_{2j}.

Étape 3: *Appliquer les règles des priorités suivantes :*

- TTPC(1) sur le sous-ensemble des tâches E1 ;
  - Classer les tâches de E1 dans l'ordre croissant des p_{1j}.
- TTPL(2) sur le sous-ensemble des tâches E2 ;
  - Classer les tâches de E2 dans l'ordre décroissant des p_{2j}.

Étape 4: *Concaténer la semi-séquence générée par TTPC(1) avec celle de TTPL(2).*

Nous illustrons cette procédure dans l'exemple ci-dessous par l'ordonnancement de 4 tâches sur 2 machines.

Exemple 2.11 :

Étape 1 : *Lister les temps de traitement t (tableau 11)*

Tableau 11

Temps de traitement de la tâche (j) sur la machine (i)

Tâches (j)	Temps de traitement des tâches	
	Machine 1	Machine 2
	p_{1j}	p_{2j}
1	3	2
2	6	6
3	5	8
4	7	4

Étape 2 : *Diviser l'ensemble des tâches en deux sous-ensembles E1 et E2;*

- E1 contient $\{p_{12} < p_{22}, p_{13} < p_{23}\}$;
- E2 contient $\{p_{11} > p_{21}, p_{14} > p_{24}\}$

Étape 3 : *Appliquer les règles des priorités suivantes :*

- TTPC(1) sur E1, donne la semi-séquence 3,2
- TTPL(2) sur E2, donne la semi-séquence 4,1

Étape 4 : *Concaténer la semi-séquence générée par TTPC(1) avec celle de TTPL(2).*

- La séquence optimale est : 3,2,4,1

2.3.3 Algorithmes de résolution des problèmes de trois machines et plus

Les environnements "job shop" complexes sont caractérisés par une multitude de machines (m) qui traitent une variété de différentes tâches (n) qui arrivent d'une façon intermittente devant les machines. Pour résoudre ce type de problème, nous utilisons des approches basées sur des heuristiques qui donnent des résultats satisfaisants. Dans de tels genres de problème, il n'existe pas des algorithmes qui fournissent des cédules de production optimales.

L'heuristique de Palmer est souvent utilisée pour la résolution du problème de 3 machines. Elle est basée sur le calcul de l'indice de la pente.

2.3.3.1 Heuristique de Palmer

Les tâches sont classées dans l'ordre décroissant de l'indice de la pente S qui est déterminé comme suit,[2] :

$$S_j=\sum_{i=1}^{m}[m-(2i-1)] \qquad (2.17)$$

avec, i=1...m et j=1..n

Exemple 3.2 :

m = 3 machines et n = 4 tâches

Les temps de traitement p_{ij} des tâches (j) sur les machines (i) sont donnés dans le

Tableau 12

Temps de traitement de la tâche (j) sur la machine (i)

Tâches (j)	Temps de traitement		
	Machine1	Machine2	Machine3
	P_{1j}	p_{2j}	p_{3j}
1	1	2	2
2	2	6	6
3	6	8	8
4	3	4	4

Étape 1: *Calculer l'indice de la pente pour chaque tâche (j) en utilisant l'équation (2.17).*

S_1= -2*1+0*8+2*4 = 6

S_2= -2*2+0*4+2*5 = 6

S_3= -2*6+0*2+2*8 = 4

S_4= -2*3+0*9+2*2 = -2

Étape 2: *Classer les tâches dans l'ordre décroissant des indices de la pente.*

$S_1 = S_2 > S_3 > S_4$

Alors les séquences optimales sont: 1,2,3,4 et 2,1,3,4

Quand le nombre de machine dépasse 4, nous opérons par la construction des temps de traitement pour arriver à un problème de 2 machines et nous appliquons après l'algorithme de Johnson. Les heuristiques utilisée pour résoudre un tel problème sont:

2.3.3.2 Heuristique de Campbell, Dudek et Smith

La procédure de cette heuristique est basée sur deux étapes essentielles [2]:

Étape 1: *La réduction du problème à deux machines en construisant les temps de traitement pour les machines 1 et 2.*

- Les temps de traitement des tâches (j) construits sur la machine 1 sont:

$$a_j^k = \sum_{j=1}^{k} p_{ij} \tag{2.18}$$

- les temps de traitements des tâches (j) construits sur la machine 2 sont :

$$b_j^k = \sum_{i=m-k+1}^{k} p_{ij} \tag{2.19}$$

avec $k = 1,2,....,(m-1)$, $j=1,2,....,n$ *et* $i=1,2,....,m$

Étape 2: *Application de l'algorithme de Johnson décrit au point 2.3.2.1 sur ces deux machines.*

Exemple 2.12 :

Soient les données illustrées dans le tableau 12, page 39 en appliquant l'étape 1 de la procédure citée ci-dessus, nous obtenons les temps de traitement construits avec les équations 2.18 et 2.19, et ces valeurs sont récapitulées dans le tableau 13.suivant .

Tableau 13

Construction des temps de traitement

Tâches (j)	Problème de m=3 et n=4	
	Construction du temps de traitement	
	$a_j = p_{1j} + p_{2j}$	$b_j = p_{2j} + p_{3j}$
1	3	4
2	8	12
3	14	16
4	7	8

Donc, on se rapporte à un problème à 2 machines et dans ce cas, l'algorithme de Johnson fourni une séquence optimale. Alors l'étape2 consiste à appliquer la démarche citée dans le point 2.3.2.1.

2.3.3.3 Heuristique de Dannenbring

Cette heuristique [2] combine les avantages de celles de Palmer et de Campbell, Dudek, Smith. Mais, l'idée est toujours basée sur la construction des temps de traitement pour arriver à un problème à deux machines et à partir de là, on utilise l'algorithme de Johnson décrit au point 2.3.2.1.

Le temps de traitement sur la machine 1 est:

$$a_j = \sum_{i=1}^{m} (m-i-1) p_{ij} \qquad (2.20)$$

et celui sur la machine 2 :

$$b_j = \sum_{i=1}^{n} p_{ij} \qquad (2.21)$$

Exemple 2.13 :

Nous considérons les mêmes données du tableau 12, page 39 et nous construisons les temps de traitement pour les machines 1 et 2. En appliquant les équations 2.20 et 2.21, nous obtenons les temps représentés dans le tableau 14.

Pour $m = 3$ et $n = 2$, les équations 2.19 et 2.20 s'écrivent comme suit :

$$a_j = 3p_{1j} + 2p_{2j} + p_{3j} \quad \text{et} \quad b_j = p_{1j} + 2p_{2j} + 3p_{3j}$$

Tableau 14

Construction des temps de traitement (Dannenbring)

Tâches (j)	Problème de m=3 et n=4	
	Construction du temps de traitement	
	$a_j = 3p_{1j} - 2p_{2j} - p_{3j}$	$b_j = p_{1j} - 2p_{2j} - 3p_{3j}$
1	9	11
2	24	32
3	42	46
4	29	23

La seconde étape consiste à appliquer l'algorithme de Johnson, décrit au paragraphe 2.3.2.1, pour obtenir la séquence optimale pour un problème de trois machines réduit à un problème de deux machines.

2.4 Conclusion

Dans ce chapitre nous avons mis en relief les caractéristiques qui définissent un environnement "job shop" comme un système de production, et essayé de comprendre sa complexité lorsqu'il y a eu lieu d'introduire des espèces de nouvelles technologies pour améliorer la productivité sur le plancher de l'usine.

Nous avons défini par la suite le problème d'ordonnancement dans l'environnement " job shop", et tiré comme conclusion que la fonction de l'ordonnancement de la production est devenue aujourd'hui le centre de décision pour l'élaboration des plans de production optimaux dans un système de fabrication. La diversité des produits et la complexité du système ont rendu la gestion des taches dans l'atelier très difficile, ce qui peut entraîner la détérioration des coûts de fabrication reliés aux pertes de temps dues au réglage de machine après chaque traitement de tache, des retards dans l'exécution des programmes de production.

Pour faire face à ces problèmes, nous avons pensé développer un système informatisé pratique et flexible pour aider les décideurs à résoudre des problèmes d'ordonnancement dans un environnement "job shop". Ce système sera doté d'une base de connaissances et des règles d'ordonnancement définies dans la section 2.3 qui serviront comme outil d'optimisation.

Le chapitre suivant traitera de l'architecture de ce système, des paramètres de l'ordonnancement et l'identification de nos critères de performance que l'usager utilisera dans son processus d'aide à la décision.

CHAPITRE 3

SYSTÈME DE BASE DE CONNAISSANCES POUR L'ORDONNANCEMENT DYNAMIQUE (SBC)

3.1 Introduction

Aujourd'hui, les besoins en systèmes d'ordonnancement ont augmenté à travers les demandes dans les industries qui font face constamment aux problèmes de coûts de fabrication générés par la mauvaise planification de la production. Une solution économique est sans doute exigée et un outil pratique et flexible est recommandé pour l'investigation et le « design » de plans de production qui optimisent la productivité sur les lignes de fabrication et réduisent les coûts de production.

Il y a deux types d'approches utilisées pour la solution d'optimisation des problèmes de production:

1. simulation interactive assistée par ordinateur, et
2. l'utilisation de la connaissance et de l'expérience des experts.

Depuis et dans beaucoup de cas, lorsqu'on est devant un problème complexe d'ordonnancement, aucune des deux approches n'a été complètement satisfaisante. Une nouvelle approche, appelée système de base de connaissances (SBC) a été développée[3]. Cette dernière est une combinaison des deux approches citées auparavant, qui peut être utilisée pour la résolution des problèmes complexes de production.

Les systèmes à base de connaissances (SBC) sont les produits de l'intelligence artificielle qui se sont développés vers la fin des années 60. Il y a lieu de traduire en langage de programmation les tâches qui ne pouvaient être réalisées avant , que par des experts humains, pour pouvoir les exploiter d'une manière informatisée.

Les systèmes d'exploitation ont été créés dans le but de simuler, voire de remplacer le raisonnement humain. Un résultat probable qui s'appuie non seulement sur un raisonnement

logique mais aussi sur des connaissances en perpétuelle évolution, dont l'ensemble est appelé expérience.

En effet, dans les SBC, on trouve à la fois de l'expertise qui se trouve dans la base de connaissances et une mise en œuvre de ce codage grâce au moteur d'inférence. Toutefois, nous n'intéressons pas ici à l'aspect proprement "Intelligence artificielle", mais à fournir à la fois un modèle descriptif et opératoire de notre système d'ordonnancement dynamique à base de connaissances.

3.2 Principaux composants d'un SBC

Les principaux composants d'un SBC sont en générale en nombre de trois et sont organisés comme l'illustre la figure 3.1, page 47

3.2.1 L'interface homme-machine

C'est l'élément qui joue le rôle de liaison entre le système et l'utilisateur. Il se compose en fait de deux modules.

- Un module d'acquisition des connaissances qui sera utilisé par le cogniticien pour développer le SBC. Ce module est constitué par un interpréteur de règles spécifique à un langage de codage de modules de savoir. Ce langage est composé d'une partie prémisse et d'une partie conclusion, et tient le rôle de déclencheur des règles.

- Une interface de communication avec l'utilisateur final du SBC, qui évolue de plus en plus vers une interface sous forme de dialogue en langage naturel.

3.2.2 La base de connaissances

Les deux éléments connus de la base de connaissances sont la base de règles et la base de faits.

- La base de règles représente les connaissances spécifiques d'un domaine d'expertise indiquant quelles conséquences doit-on tirer, ou quelles actions doit-on accomplir lorsque telle situation est établie ou est à établir.
- La base de faits contient les situations que l'on considère établies, tels que par exemple la situation sur le plancher de l'usine, l'état des machines etc.

3.2.3 Le moteur d'inférence

Ce composant essentiel du SBC met en œuvre des mécanismes pour résoudre les problèmes décrits par les données contenues dans la base des faits, et cela en sélectionnant, validant, et déclenchant les règles contenues dans la base de connaissances.

On distingue deux phases de fonctionnement d'un moteur d'inférence:

- La phase d'évaluation:
  Dans cette première partie, le moteur va sélectionner dans la base de connaissances les règles qui seront déclenchées, et les faits avec lesquels elles le seront.

- La phase d'exécution
  Dans cette deuxième partie du cycle du moteur d'inférence, on va exécuter les règles obtenues à la fin de la phase d'évaluation.

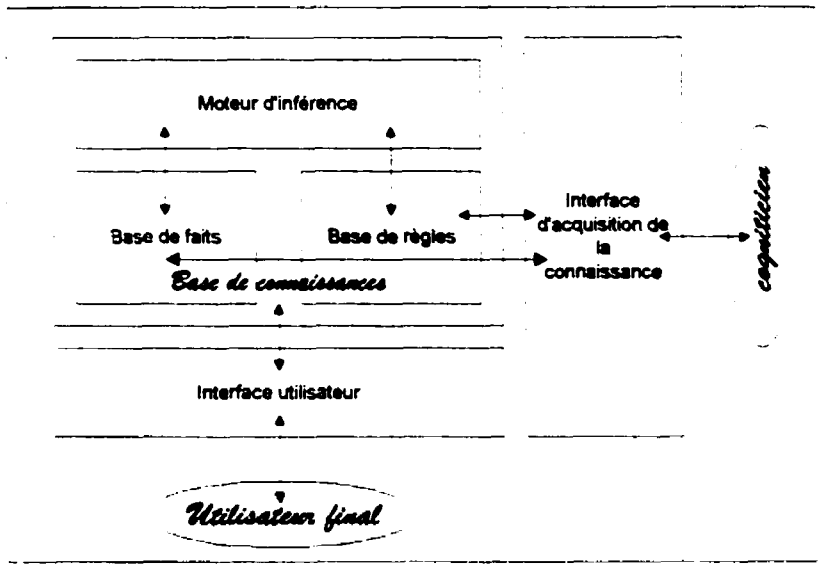


Figure 3.1: Organisation d'un système de base de connaissances

3.3 Principales étapes de conception d'un SBC

Les principales étapes qui constituent un projet de conception d'un SBC sont énumérées ci-après:

1. Étude d'orientation (durée 1 à 3 mois)

- Clarifier la demande du client en analysant le contexte organisationnel auquel s'applique la demande afin de déterminer les caractéristiques du projet envisagé.

2. Recommandations sur La SBC

- Identification des intervenants (ingénieurs de connaissance, experts, usagers) ;
- Le choix d'un sous ensemble restreint et significatif de l'application qui sera implanté avec la maquette ;
- La démarche à suivre et un échéancier.

3. Extraction sommaire des connaissances relatives au sous ensemble de l'application choisie.

4. Élaboration d'une esquisse du noyau de la SBC.

- Préciser le modèle de représentation de connaissances (règles de production, cadres) et les principaux mécanismes d'inférence requis par l'application.

5. Recommandations sur le choix du matériel et de l'environnement de conception de SBC à utiliser au cours de la phase d'exploration.

6. Phase d'exploration (durée de 6 à 10 mois).

- Élaboration de la maquette en fonction des orientations et des décisions du client pour la phase d'exploration ;
- Extraction de la connaissance et mise au point de la maquette.

7. Évaluation de la maquette (1 mois)

- Permet de mesurer la viabilité du projet et établir les recommandations pour le développement du SBC.
- Évaluation et révision des choix concernant:
  - Les modèles de représentation des connaissances,
  - Le matériel et logiciel.

8. Développement du SBC

Après avoir évalué la maquette et choisit le matériel, nous passons au développement du système proprement dit. Cette étape est la plus importante dans notre projet.

9. Intégration

- Couplage entre le SBC et les bases de données ;
- Installation du système dans l'environnement des usagers.

3.6 Développement du SBC

Le développement du SBC met en présence trois catégories d'intervenants:

- Les usagers auxquels on destine le SBC ;
- Les concepteurs ou «ingénieurs de la connaissance » appelés aussi cogniticiens qui prennent en charge les aspects méthodologiques et techniques du projet. Le métier de cogniticien consiste à recueillir des connaissances auprès d'experts et à les coder de manière utilisable dans une base de données, par le moteur d'inférence.
- Les experts du domaine, spécialistes qui détiennent la connaissance qu'on veut formaliser et emmagasiner dans le SBC.

L'acquisition des connaissances est l'une des phases la plus critique de la conception d'un SBC, puisque l'efficacité du comportement du système dépend de la qualité et de la cohérence qu'il contient. Au cours de cette phase, les ingénieurs de la connaissance doivent aider les experts à identifier, représenter et structurer des connaissances.

Le choix de la méthode de la représentation des connaissances est déterminant: elle doit être adaptée au domaine et au raisonnement des experts travaillant sur ce sujet. Il existe différentes techniques de représentation de connaissances dans le domaine de l'intelligence artificielle. La plus familière est celle des règles, dites de production. Ce mode est utilisé par de nombreux systèmes experts et il est représenté de la façon suivante :

SI (prémisses),

ALORS (conclusions)

Le membre du haut décrit les conditions dans lesquelles peuvent se déduire les conclusions du membre du bas de la règle.

Exemple 3.1 :

Si le système manufacturier est composé de 2 machines

Alors utiliser l'algorithme de Johnson pour résoudre le problème d'ordonnancement des tâches

Les autres techniques de représentation des connaissances sont:

- Les réseaux sémantiques: Ils se représentent sous la forme de réseaux comme son nom l'indique dont les nœuds figurent les concepts, et les arcs reliant ces nœuds, les relations entre les concepts. La figure 3.2 illustre un exemple de réseau sémantique.

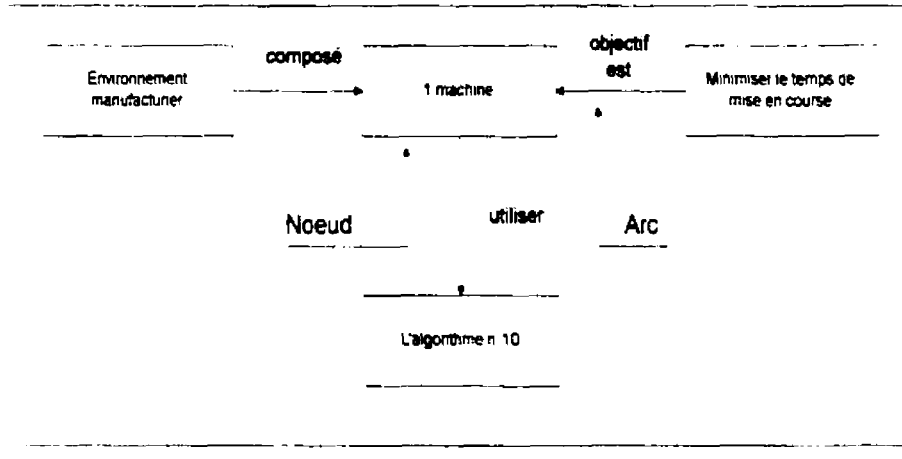


Figure 3.2: Déduction d'une conclusion grâce à un réseau sémantique

- Les frames ou cadres: un frame est une structure qui permet de décrire un concept selon un modèle ou prototype. La figure 3.3 illustre un exemple de frame utilisé dans la représentation de la connaissance.

prototype (personne)	
propriétés de toute entité de type personne	
nom:	chaîne de caractères
prénom:	chaîne de caractères
sexe :	élément de [M, F]
Année de naissance :	entier

Figure 3.3: Cadre définissant un concept

- Les scripts ou schémas: décrivent des situations stéréotypées sous la forme d'une succession d'actions et comprennent aussi une liste des acteurs et des objets nécessaires à la description de la situation. La figure 3.4 illustre un exemple de script.

Acteurs: clients, ouvreuse..
Objets: caisse, tickets, place, film...
Évènements:
Le client entre dans le cinéma.
Le client va à la caisse.
Le client paie le ticket
L'ouvreuse déchire le ticket

Figure 3.4: Exemple de Script

Dans notre système BC proposé, les connaissances sont générées par les règles de production, comme par exemple:

Si *l'environnement manufacturier est composé d'une seule machine et l'objectif est la minimisation du temps total de l'ordonnancement et la règle de priorité premier arrivé, premier servi (PAPS) est utilisée pour l'ordonnancement;*

Alors *utiliser l'algorithme n° 1 pour résoudre le problème.*

3.6 L'architecture proposée du SBC

Le SBC est une combinaison de la simulation et les connaissances des experts. C'est une structure dynamique qui génère des solutions par les algorithmes d'ordonnancement développés dans le chapitre précédent.

Si la solution générée s'approche de l'objectif d'ordonnancement, alors la solution optimale est atteinte et envoyée au système de fabrication. Autrement, le système de base de connaissance générera un nouveau ensemble de données pour une nouvelle simulation et le processus est répété.

Le système d'ordonnancement à base de connaissances est développé pour atteindre la séquence optimale d'un groupe de problème d'ordonnancement C'est un système multi-modèle La figure 3.5 illustre le modèle de la structure et l'interaction de notre SBC.

Plusieurs règles et algorithmes d'ordonnancement sont utilisés pour la simulation des séquences de traitement des tâches. C'est un système d'optimisation dynamique qui a les tâches suivantes:

- Mettre à jour des besoins et données de fabrication du planché de production;
- Introduire et mettre à jour les paramètres intervenant dans le processus de simulation pour générer les différentes solutions possibles du groupe d'ordonnancement;
- Comparer les différentes solutions générées par le système et sélectionner une solution appropriée;

- Valider la solution sélectionnée et comparer à l'objectif de l'ordonnancement pour identifier la solution optimale, qui satisfait les contraintes de l'environnement manufacturier;
- Générer un nouvel ensemble de données de fabrication pour une nouvelle simulation dans le cas où la séquence optimale ne serait pas atteinte.

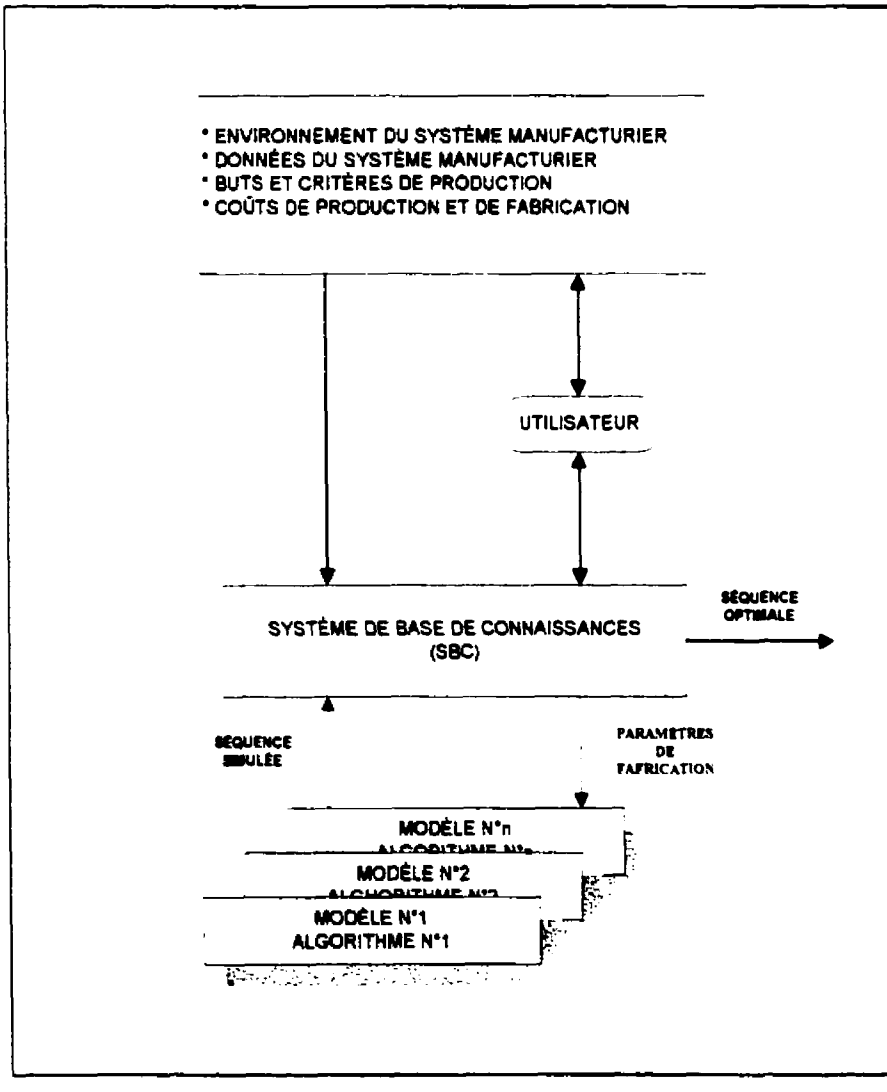


Figure 3.5: Architecture tandem d'un système de base de connaissances pour l'ordonnancement dynamique.

3.6 Structure proposée du SBC

Notre système à base de connaissances (SBC) est défini comme un système informatisé, développé avec l'expertise humaine et les connaissances environnementales pour la résolution du problème d'ordonnancement spécifié. Il est constitué de trois principaux composants A, B et C, comme illustrés dans la figure 3.6 et qui sont définis dans la figure 3.7, page 55.

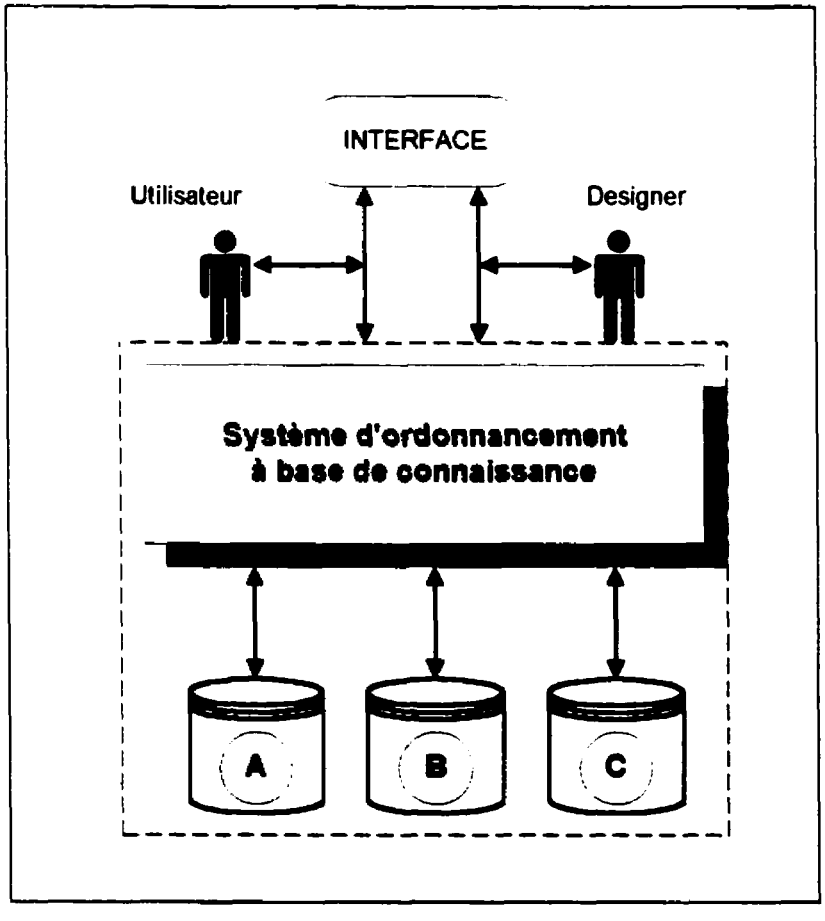


Figure 3.6: Les composants du SBC

Dans le SBC que nous proposons, le composant le plus important de la structure de notre système est la base de connaissances représentée par l'élément C tel qu'illustré dans les figures 3.6 et 3.7.

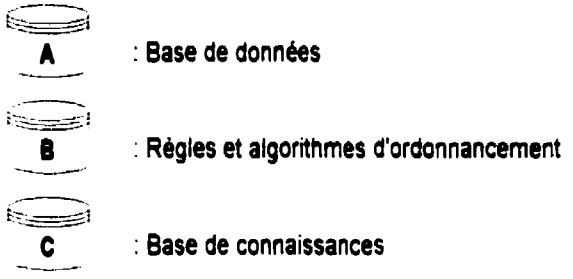


Figure 3.7: Éléments du SBC proposé

3.6.1 Base de données du SBC proposé

La base de données (élément A), appelée base de faits, est un des trois composants du SBC. Elle est constituée par un ensemble de données organisées en fonction d'une structure généralement manipulées par l'intermédiaire d'un logiciel spécialisé appelé système de gestion de bases de données (SGBD).

Cette base de données (BD) contient la description suivante:

- de l'atelier (stations de travail, types de pièces...);
- du problème d'ordonnancement (besoins de productions, les dates de livraison, la disponibilité des machines...);
- et l'état actuel de l'usine.

Toutefois, elle contient de la connaissance dite déclarative qui est représentée par des informations caractérisant chaque tâche candidate à l'ordonnancement, telles que:

- numéro de la tâche,
- numéro de la pièce,
- taille du lot,
- le facteur d'importance w_j, qui représente généralement un coût;
- la date due d_j,

- la date de disponibilité r_j,
- le temps de traitement p_j .
- vitesse de base des équipements v_i..

3.6.2 Moteur d'inférence du SBC proposé

Le moteur d'inférence (élément B) est un logiciel spécialisé qui construit automatiquement le raisonnement du système en sélectionnant et en activant les règles pertinentes de la base de connaissances, en fonction de l'état de la base de faits. Ce dernier sert à générer des séquences de production. Il comprend les règles de priorité et les algorithmes d'optimisation qui ont été recensés dans le tableau 15, page 60. Les séquences générées par le système sont évaluées suivant leurs critères de performance résumés dans le tableau 16; page 61.

3.6.3 Base de connaissances du SBC proposé

La base de connaissances (BC) contient des faits au sujet d'un domaine de connaissance et des heuristiques, c'est à dire des règles à appliquer aux faits pour la résolution du problème posé.

L'organisation et la représentation de la BC ont un impact sur la performance du système d'ordonnancement. La base de connaissance de notre système d'ordonnancement est développée avec les règles de production suivantes:

Règle 01

Si *l'environnement manufacturier est composé d'une seule machine, et l'objectif est la minimisation du temps total de l'ordonnancement, et la règle de priorité premier arrivé, premier servi (PAPS) est utilisée pour l'ordonnancement;*

Alors *utiliser l'algorithme n° 1 pour résoudre le problème.*

Règle 02

Si *l'ordonnancement du groupe de tâches considéré concerne un système composé d'une seule machine, et l'objectif est la minimisation du temps d'achèvement des tâches;*

Alors *résoudre le problème en utilisant les algorithmes du n°2 et n°3.*

Règle 03

Si *l'environnement manufacturier est composé d'une seule machine, et l'objectif est la minimisation de la somme pondérée des temps d'achèvement des tâches;*

Alors *utiliser les algorithmes n°4 et 5 pour résoudre le problème.*

Règle 04

Si *le système de production est composé d'une machine, et l'objectif est de rencontrer les délais de livraison client.*

Alors *utiliser l'algorithme n°6 pour résoudre le problème.*

Règle 05

Si *l'environnement manufacturier est composé d'une machine, et l'objectif est la minimisation des retard;*

Alors *résoudre le problème avec les algorithmes n° 7 à 10.*

Règle 06

Si *l'environnement manufacturier est composé d'une machine, et l'objectif est la minimisation du coût apparenté au retard;*

Alors *résoudre le problème avec l'algorithme n° 11.*

Règle 07

Si *l'environnement manufacturier concerne une machine, et l'objectif recherché est la minimisation du temps total de mise en course des équipement et la somme pondérée des retard;*

Alors *résoudre le problème en utilisant l'algorithme n° 12.*

Règle 08

Si *l'environnement manufacturier est composé de 2 machines;*

Alors *utiliser les algorithmes n° 13 pour résoudre le problème.*

Règle 09

Si *l'environnement manufacturier est composé de 3 machines;*

Alors *résoudre le problème avec les algorithmes de n° 14 au n°15.*

Règle 10

Si l'environnement manufacturier est composé de plus de 3 machines,

Alors *résoudre le problème avec les algorithmes de n° 16.*

3.6.4 Principe de fonctionnement du SBC proposé

Le SBC choisit les algorithmes d'optimisation en fonction de l'environnement manufacturier désiré. Dans notre système d'ordonnancement, nous avons prévu 3 différents types de situation, et pour chacune d'elles correspondent des algorithmes de résolution de problème d'ordonnancement. Ces 3 situations sont définies comme suit:

1. Devant un problème d'ordonnancement d'une cellule composée d'une seule machine (m=1), les algorithmes n° 1 à n° 12 peuvent être utilisés pour simuler différentes possibilités de séquences de tâches.

2. Dans le cas d'une cellule composée de deux machines (m=2), l'algorithme n° 13 (algorithme de Johnson) est considéré comme la règle la plus puissante pour la résolution de tel problème.

3. Et dans un environnement composé de trois machines ou plus ($m \geq 3$), nous utilisons les méthodes par construction des temps de traitement des tâches, pour ramener le problème dans un ordonnancement standard de (n) tâches assignées à une cellule de fabrication

composée de deux machines, et les algorithmes n° 14, 15 et 16 sont utilisés pour résoudre ce problème.

Cependant, en fonction de l'objectif de l'ordonnancement, le SBC sélectionne un groupe de règles de priorité appropriées (tableau 15, page 60) pour résoudre le problème et le système génère des différentes séquences et ainsi il calcule pour chacune, les différents critères de performance (tableau 16, page 61) . L'usager utilise ces critères pour choisir la séquence optimale des tâches en fonction de l'objectif visé.

Tableau 15

Règles et algorithmes sélectionnés pour la résolution des problèmes d'ordonnancement

N°de règles d'ordonnancement	Description des règles	
1	Premier arrivé, premier servi (PAPS)	$Min(r_j)$
2	Temps de traitement le plus court (TTPC)	$Min(p_j)$
3	Temps de traitement le plus long (TTPL)	$Max(p_j)$
4	Temps de traitement pondéré le plus court (TTPPC)	$Max(\frac{w_j}{p_j})$
5	Temps de traitement pondéré le plus long (TTPPL)	$Min(\frac{w_j}{p_j})$
6	Date de livraison la plus rapprochée (DLPR)	$Min(d_j)$
7	Marge dynamique minimale (MDM)	$Min(d_j - p_j)$
8,9,10	Ratio critique minimal de la marge dynamique 1,2,3 (RCMMD)	$RC1=\frac{(d_j-p_j)}{p_j}$ $RC2=\frac{(d_j-p_j)}{d_j}$ $RC3=\frac{d_j}{p_j}$
11	Coût apparenté au retard (CAR)	
12	Coût apparenté au retard et au temps de mise en course (CARTMC)	
13	Algorithme de Johnson	
14	Algorithme de Palmer	
15	Algorithme de Campbell, Dudek et Smith	
16	Algorithme de Dannembring	
	d_j : date de livraison — r_j : date de disponibilité p_j : temps de traitement — w_j : facteur d'importance	

Tableau 16

Critères de performance

N° du critère	Description des critères de performance	Équations
1	Temps total de production ou « Makespan »	$C_{max}=\max(C_j)$
2	Somme des retards	$\sum_{j=1}^{n} T_j = \sum_{j=1}^{n} [(C_j - d_j) > 0]$
3	Retard moyen des tâches	$\frac{\sum T}{n}$
4	Temps moyen des retards	$\frac{\sum T}{\sum U}$
5	Nombre de tâches en retard	$\sum_{j=1}^{n} U_j = \begin{cases} 1, si C_j - d_j > 0 \\ 0, autrement \end{cases}$
6	Somme des temps d'achèvement des tâches	$\sum_{j=1}^{n} C_j = \sum_{j=1}^{n} p_j + t$
7	Temps d'achèvement moyen des tâches	$\frac{\sum_{j=1}^{n} C_j}{n}$
8	Temps de passage total des tâches dans l'atelier	$\sum_{j=1}^{n} (C_j - r_j)$
9	Temps de passage moyen des tâches dans l'atelier	$\frac{\sum_{j=1}^{n} (C_j - r_j)}{n}$
10	Nombre moyen des tâches dans le système	$\frac{\sum_{j=1}^{n} C_j}{C_{max}}$
11	Coût total de l'ordonnancement	$\sum_{j=1}^{n} w_j * C_j$
12	Coût total des retards	$\sum_{j=1}^{n} w_j * T_j$
13	Coût moyen des retards	$\frac{\sum_{j=1}^{n} w_j T_j}{\sum_{j=1}^{n} U}$
14	Temps libre sur les machines	$C_{max} - \sum_{j=1}^{n} p_j$

3.7 Module du SBC sur la plate forme Matlab

Nous avons développé notre système d'ordonnancement en utilisant le logiciel « Matlab » que nous estimons qui est puissant comme outil d'optimisation, et il offre une grande flexibilité ainsi qu'une boite d'outils performants. L'architecture en tandem, comme le montre la figure 3.5 à la page 53, utilise le système de base de connaissances (méthode déclarative) ainsi que les algorithmes de calcul (méthode procédurale). Le SBC choisit, en fonction de certains critères, les algorithmes qui doivent résoudre le problème. Après traitement informatique, la solution générée par le système est évaluée par l'utilisateur, si elle est satisfaisante, elle est envoyée au système de production. La figure3.8, page 65 montre l'ordinogramme du SBC proposé. Les étapes du fonctionnement de ce système sont les suivantes:

Étape 1: *Début de l'ordonnancement des tâches*

Pour commencer l'ordonnancement des tâches, nous faisons appel au système en tapant la commande "systeme_expert".

Étape 2: *Mise à jour de la base de données*

Nous devons mettre à jour la base de données chaque fois qu'une nouvelle tâche est introduite dans le système.

Étape 3: *Introduire le nombre de machine*

Nous devons définir le nombre de machines que constitue notre système de production. Le SBC va nous poser la question suivant:

Quel est le nombre de machines?

Il y a lieu de taper les chiffres 1,2,3... suivant le nombre de machine que constitue le système de production.

Étape 4: *Introduire les tâches et leurs caractéristiques*

Les tâches et leurs caractéristiques sont introduites dans le système dans l'ordre suivant:

1. Le numéro de la tâche;
2. le temps de traitement de la tâche;
3. la date due;
4. le poids de la tâche;
5. et le date de disponibilité de la tâche.

Étape 5: *Choix de l'objectif à optimiser*

Cette étape consiste à choisir l'objectif que nous voulons atteindre. Les objectifs sont représentés par des chiffre. Le système nous demande d'entrer le chiffre correspondant à l'objectif.

Étape 6: *Appel des algorithmes d'ordonnancement*

Une fois que les étapes précédentes ont été réalisées, le système choisira automatiquement le ou les algorithmes (tableau 15, page 60) appropriés pour résoudre le problème.

Étape 7: *Générer les séquences des tâches*

La séquence des tâches est générée par le système et elle est affichée sur l'écran de l'ordinateur de l'usager.

Étape 8: *Calcul des mesures de performance*

Le système calcule les mesures de performances (tableau 16, page 61) suivant la séquence générée par ce dernier. Le résultat de la simulation est affiché sur l'écran de l'ordinateur de l'usager.

Étape 9: *Planification des ressources*

Une fois les plans de production optimaux ont été trouvés, l'usager doit déterminer les ressources humaines et matérielles nécessaires pour la réalisation de ces derniers d'une façon rationnelle.

Étape 10: *Tracer le diagramme de Gantt des tâches*

Le système trace le diagramme de la séquence optimale des tâches. Il nous permet de visualiser la façon dans laquelle les tâches sont traitées sur les machines dans le temps.

Étape 11: *Fin de l'ordonnancement*

L'ordonnancement s'arrête une fois que la cédule optimale est trouvée et elle est envoyée à la fabrication.

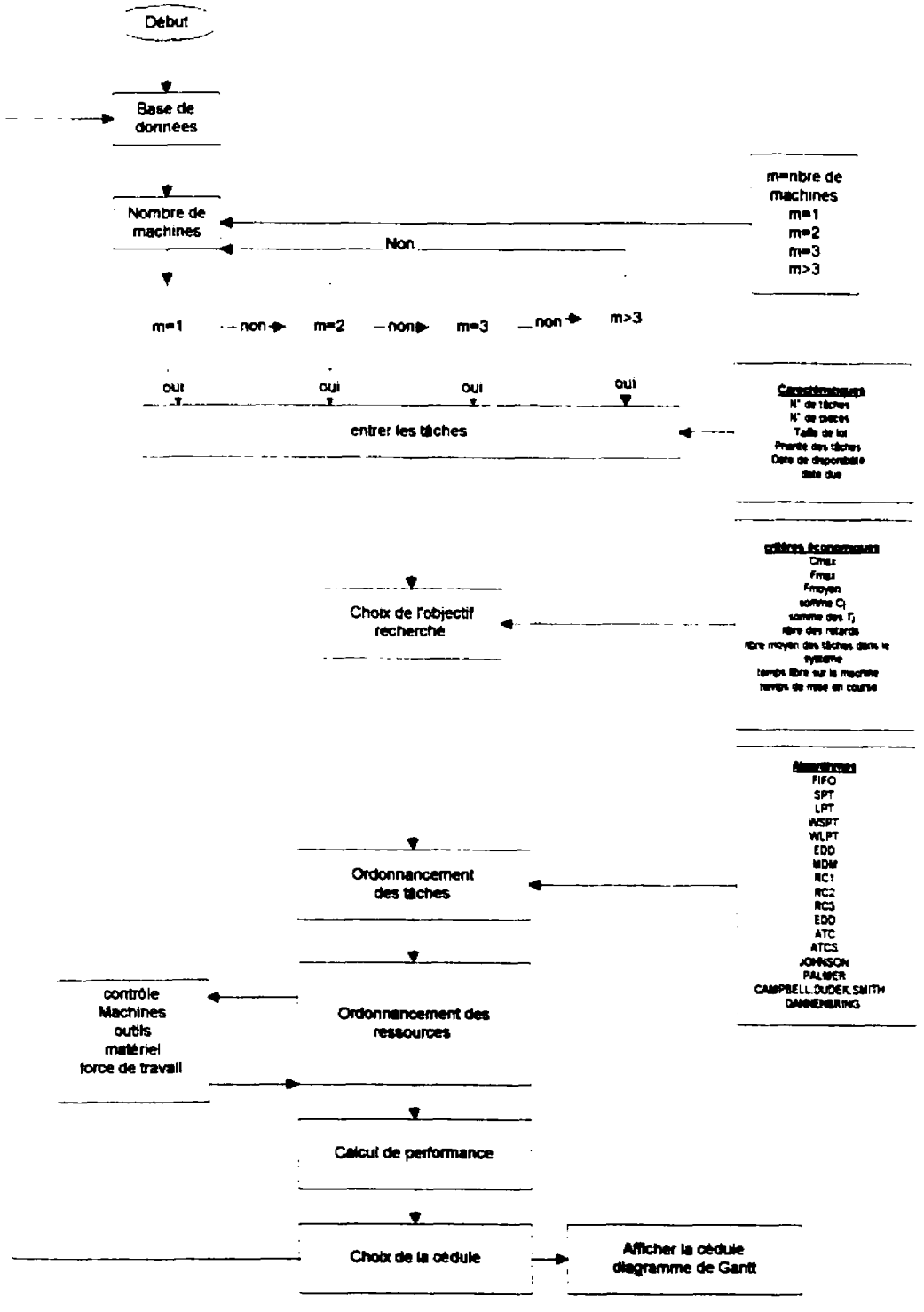


Figure 3.8: Ordinogramme pour le fonctionnement du SBC

3.7.1 Module base de connaissances

Le module principal de notre système d'ordonnancement comprend la base de connaissances (BC), qui est représentée par des règles de production développées dans la section 3.3.3 et que nous avons programmées dans un fichier appelé fonction « système expert » de Matlab comme indiqué ci dessous.

Étape 1: *Début de l'ordonnancement des tâches*

```
Choisir le nombre de machines composant le système de production
Load variables_projet
nb_machine=input('Combien y a-t-il de machine? ');
If    nb_machine==1
disp(' ')
```

Étape 2: *Introduire la matrice des données du problème*

```
donnee_u=input('Entrer la matrice de donnee : ');
If isempty(donnee_u)==1
Else donnee=donnee_u;
End
disp(' ')
```

Étape 3: *Choisir l'objectif d'optimisation*

```
disp('Quel est l'objectif d'optimisation recherché')
disp(' '),disp(' ')
disp('Minimisation du temps total d'ordonnancement')
disp('et la discipline premier arrivé premier servi          (1)')
disp(' ')
disp('Minimisation du temps d'achèvement de tâche            (2)')
disp(' ')
```

```
disp('Rencontrer les délais clients                    (3)')
disp(' ')
disp('Minimisation du coût lié au retard               (4)')
disp(' ')
disp('Minimisation du retard total                     (5)')
disp(' ')
disp('Minimisation du temps de mise en course')
disp('et le retard total pondéré                       (6)')
disp(' '),disp( ' ')
objectif=input('Entrer le chiffre correspondant à l'objectif:')
clc
```

Étape 4: *Appel aux algorithmes de résolution du problème*

```
switch (objectif)
case 1, regle1(donnee);
case 2, regle2(donnee); regle3(donnee);
case 3, regle4(donnee);
case 4, regle11(donnee);
case5,regle5(donnee);regle6(donnee);regle7(donnee);regle8(donnee);
regle9(donnee); regle10(donnee););
End
```

Étape 5: *Si l'objectif est la minimisation du temps de mise en course, nous devons introduire le nombre 6*

```
If objectif==6
disp(' ')
```

Étape 6: *On introduit dans le système la matrice du temps de mise en course du début de l'ordonnancement*

```
sto_u=input('Entrer la matrice de setup initial : ');
If isempty(sto_u)==1
Else    sto=sto_u;
End
```

Étape 7: *Après on introduit les temps de mise en course entre les tâches qui se suivent.*

```
disp(' ')
combinaison_u=input('Entrer la matrice de setup combiné: ');
If isempty(combinaison_u)==1
Else    combinaison=combinaison_u;
End
regle12(donnee_longue,sto,combinaison);
End
```

Étape 8: *Dans le cas où l'environnement est composé de 2 machines*

```
Elseif nb_machine==2
disp(' ')
```

Étape 9: *On introduit dans le système la matrice des données (le numéro de la tâche, le délai de livraison, le facteur d'importance et le temps de disponibilité).*

```
donnee2_u=input('Entrer la matrice de donnee2 : ');
If isempty(donnee2_u)==1
Else    donnee2=donnee2_u;
End
disp(' ')
```

Étape 10: *On introduit les temps de traitement des tâches sur chaque machine*
1ere colonne = machine1 et 2eme colonne = machine 2.

```
machine2_u=input('Entrer la matrice des pj : ');
```

```
If isempty(machine2_u)==1
Else machine2=machine2_u;
End
clc
```

Étape 11: *Appel à l'algorithme de résolution du problème*

```
regle13(machine2,nb_machine,donnee2,[]);
Else
disp(' ')
```

Étape 12: *Dans le cas où l'environnement manufacturier est composé de 3 machines et plus, on introduit la matrice des données qui sont le numéro de la tâche, le délai de livraison, le facteur d'importance et le temps de disponibilité.*

```
donneex_u=input('Entrer la matrice de donneex : ');
If isempty(donneex_u)==1
Else donneex=donneex_u;
End
disp(' ')
```

Étape13: *On introduit les temps de traitement des tâches sur machine*

```
machinex_u=input('Entrer la matrice des pj : ');
If isempty(machinex_u)==1
Else machinex=machinex_u;
End
```

Étape 14: *Appel aux algorithmes de résolution du problème*

```
regle14(machinex,nb_machine,donneex);
regle15(machinex,nb_machine,donneex);
regle16(machinex,nb_machine,donneex);
```

End

Étape 15: *Afficher les données*

donnee;donneex;machinex;donnee2;machine2;

Étape 16: *Enregistrer les données*

save variables_projet
clear

Étape 17: *Charger les données*

load variables_projet
End

3.7.2 Module comprenant les algorithmes d'optimisation

Le deuxième module comprend le programme sur "matlab" des heuristiques et algorithmes d'ordonnancement représentés dans le tableau 15, page 60.

ALGORITHME N° 1

Premier arrivé, premier servi « PAPS »

Étape 1: *Appel de fonction*

function [vecteur]=regle1(vecteur)
disp(' ')
disp('Règle 1')
End

Étape 2: *Définir les dimensions de la matrice*

[lignes,colonnes]=size(vecteur);

Étape 3: *Calcul des indicateurs*

```
vecteur=calcul_indicateur(vecteur);
```

Étape 4: *Trier les tâches dans l'ordre croissant des r_j*

```
For  i=1:lignes-1
     For  j=i+1:lignes
          If   vecteur(i,5)>vecteur(j,5)
               temp=vecteur(j,:);
               vecteur(j,:)=vecteur(i,:);
               vecteur(i,:)=temp;
          End
     End
End
```

Étape 5: *Calcul des critères de performance*

```
vecteur=calcul_de_performance(vecteur);
```

Étape 6: *Afficher les résultats*

```
vecteur=resultats(vecteur);
```

ALGORITHME N° 2

Le temps de traitement le plus court « TTPC »

Étape 1: *Appel de fonction*

```
function regle2(vecteur)
```

Étape 2: *Définir les dimensions de la matrice*

```
[lignes,colonnes]=size(vecteur);
```

Étape 3: *Calcul des indicateurs*

```
vecteur=calcul_indicateur(vecteur);
```

Étape 4: *Trier les tâches dans l'ordre croissant des p_j*

```
For  i=1:lignes-1
```

```
For  j=i+1:lignes
     If   vecteur(i,2)>vecteur(j,2)
          temp=vecteur(j,:);
          vecteur(j,:)=vecteur(i,:);
          vecteur(i,:)=temp;
     End
End
End
```

Étape 5: *Calcul des critères de performances*

```
vecteur=calcul_de_performance(vecteur);
```

Étape 6: *Afficher les résultats*

```
vecteur=resultats(vecteur);
```

ALGORITHME N° 3

Le temps de traitement le plus long « TTPL »

Étape 1: *Appel de la fonction*

```
function regle3(vecteur)
```

Étape 2: *Déterminer les dimensions de la matrice*

```
[lignes,colonnes]=size(vecteur);
```

Étape 3: *Calcul des indicateurs*

```
vecteur=calcul_indicateur(vecteur);
```

Étape 4: *Trier les tâches dans l'ordre décroissant des p_j*

```
For  i=1:lignes-1
     For  j=i+1:lignes
          If   vecteur(i,2)<vecteur(j,2)
               temp=vecteur(j,:);
```

```
                    vecteur(j,:)=vecteur(i,:);
                    vecteur(i,:)=temp;
                End
            End
        End
```

Étape 5: *Calcul des critères de performances*

```
vecteur=calcul_de_performance(vecteur);
```

Étape 6: *Afficher les résultats*

```
vecteur=resultats(vecteur);
```

ALGORITHME N° 4

Délai de livraison le plus rapproché « DLPR »

Étape 1: *Appel de la fonction*

```
function regle4(vecteur)
```

Étape 2: *Déterminer les dimensions de la matrice*

```
[lignes,colonnes]=size(vecteur);
```

Étape 3: *Calcul des indicateurs*

```
vecteur=calcul_indicateur(vecteur);
```

Étape 4: *Trier les tâches dans l'ordre croissant des délais de livraison*

```
For  i=1:lignes-1
    For  j=i+1:lignes
        If    vecteur(i,3)>vecteur(j,3)
            temp=vecteur(j,:);
            vecteur(j,:)=vecteur(i,:);
            vecteur(i,:)=temp;
        End
```

```
        End
    End
```

Étape 5: *Calcul des critères de performances*

```
vecteur=calcul_de_performance(vecteur);
```

Étape 6: *Afficher les résultats*

```
vecteur=resultats(vecteur);
```

ALGORITHME N° 5

Marge Dynamique Minimale « MDM »

Étape 1: *Appel de la fonction*

```
function regle5(vecteur)
```

Étape 2: *Déterminer les dimensions de la matrice*

```
[lignes,colonnes]=size(vecteur);
```

Étape 3: Calcul des indicateurs

```
vecteur=calcul_indicateur(vecteur);
```

Étape 4: *Trier les tâches dans l'ordre croissant des* (d_j-p_j-r_j)

```
For  i=1:lignes-1
     For  j=i+1:lignes
          If   vecteur(i,9)>vecteur(j,9)
               temp=vecteur(j,:);
               vecteur(j,:)=vecteur(i,:);
               vecteur(i,:)=temp;
          End
     End
End
```

Étape 5: *Calcul des critères de performances*

vecteur=calcul_de_performance(vecteur);

Étape 6: *Afficher les résultats*

vecteur=resultats(vecteur);

ALGORITHME N° 6

Ratio Critique de la Marge Dynamique 1 « RC1 »

Étape 1: *Appel de la fonction*

function regle6(vecteur)

Étape 2: *Déterminer les dimensions de la matrice*

[lignes,colonnes]=size(vecteur);

vecteur=calcul_indicateur(vecteur);

Étape 3: *Trier les tâches dans l'ordre croissant des* $(d_j-p_j-r_j)$ d_j

```
For  i=1:lignes-1
    For  j=i+1:lignes
        If   vecteur(i,10)>vecteur(j,10)
             temp=vecteur(j,:);
             vecteur(j,:)=vecteur(i,:);
             vecteur(i,:)=temp;
        End
    End
End
```

Étape 5: *Calcul des critères de performances*

vecteur=calcul_de_performance(vecteur);

Étape 6: *Afficher les résultats*

```
vecteur=resultats(vecteur);
```

ALGORITHME N° 7

Ratio Critique de la Marge Dynamique 2 « RC2 »

Étape 1: *Appel de la fonction*

```
function regle7(vecteur)
```

Étape 2: *Déterminer les dimensions de la matrice*

```
[lignes,colonnes]=size(vecteur);
```

Étape 3: *Calcul des indicateurs*

```
vecteur=calcul_indicateur(vecteur);
```

Étape 4: *Trier les tâches dans l'ordre croissant des* $(d_j - p_j - r_j)\ p_j$

```
For  i=1:lignes-1
     For  j=i+1:lignes
          If   vecteur(i,11)>vecteur(j,11)
               temp=vecteur(j,:);
               vecteur(j,:)=vecteur(i,:);
               vecteur(i,:)=temp;
          End
     End
End
```

Étape 5: *Calcul des critères de performances*

```
vecteur=calcul_de_performance(vecteur);
```

Étape 6: *Afficher les résultats*

```
vecteur=resultats(vecteur);
```

ALGORITHME N° 8

Ratio Critique de la Marge Dynamique 3 « RCMD3 »

Étape 1: *Appel de la fonction*

```
function regle8(vecteur)
```

Étape 2: *Déterminer les dimensions de la matrice*

```
[lignes,colonnes]=size(vecteur);
```

Étape 3: *Calcul des indicateurs*

```
vecteur=calcul_indicateur(vecteur);
```

Étape 4: *Trier les tâches dans l'ordre croissant des d_j p_j*

```
For  i=1:lignes-1
     For  j=i+1:lignes
          If   vecteur(i,12)>vecteur(j,12)
               temp=vecteur(j,:);
               vecteur(j,:)=vecteur(i,:);
               vecteur(i,:)=temp;
          End
     End
End
```

Étape 5: *Calcul des critères de performances*

```
vecteur=calcul_de_performance(vecteur);
```

Étape 6: *Afficher les résultats*

```
[vecteur,table]=resultats(vecteur);
```

ALGORITHME N° 9

Le temps de traitement pondéré le plus court « TTPPC »

Étape 1: *Appel de la fonction*

function regle9(vecteur)

Étape 2: *Déterminer les dimensions de la matrice*

```
[lignes,colonnes]=size(vecteur);
```

Étape 3: *Calcul des indicateurs*

```
vecteur=calcul_indicateur(vecteur);
```

Étape 4: *Trier les tâches dans l' ordre croissant des $w_j p_j$*

```
For  i=1:lignes-1
     For  j=i+1:lignes
          If   vecteur(i,13)>vecteur(j,13)
               temp=vecteur(j,:);
               vecteur(j,:)=vecteur(i,:);
               vecteur(i,:)=temp;
          End
     End
End
```

Étape 5: *Calcul des critères de performances*

```
vecteur=calcul_de_performance(vecteur);
```

Étape 6: *Afficher les résultats*

```
vecteur=resultats(vecteur);
```

ALGORITHME N° 10

Le temps de traitement pondéré le plus long "TTPPL"

Étape 1: *Appel de la fonction*

```
function regle10(vecteur)
```

Étape 2: *Déterminer les dimensions de la matrice*

```
[lignes,colonnes]=size(vecteur);
```

Étape 3: *Calcul des indicateurs*

```
vecteur=calcul_indicateur(vecteur);
```

Étape 4: *Trier les tâches dans l'ordre décroissant des wj pj*

```
For  i =1:lignes-1
     For  j=i+1:lignes
          If   vecteur(i,13)<vecteur(j,13)
               temp=vecteur(j,:);
               vecteur(j,:)=vecteur(i,:);
               vecteur(i,:)=temp;
          End
     End
End
```

Étape 5: *Calcul des critères de performances*

```
vecteur=calcul_de_performance(vecteur);
```

Étape 6: *Afficher les résultats*

```
(vecteur)=resultats(vecteur);
```

ALGORITHME N° 11

Tardiveté Apparente avec le coût "ATC"

Étape 1: *Appel de la fonction*

```
function regle11(vecteur)
```

Étape 2: *Déterminer les dimensions de la matrice*

```
[lignes,colonnes]=size(vecteur);
```

Étape 3: *Calcul des indicateurs*

```
vecteur=calcul_indicateur(vecteur);
```

Étape 4: *Calcul des paramètres*

```
pmoyen=mean(vecteur(:,2));
Cmax=sum(vecteur(:,2));
dmax=max(vecteur(:,3));
dmin=min(vecteur(:,3));
R=(dmax-dmin)/cmax;
If    R<= 0.5
      k=4.5+R;
Else  k=6-2*R;
End
```

Étape5: *Calcul de l'indice de rangement*

```
Ij=(wj/pj)exp( -min(dj-pj-t,0) / kpmoyen )
t=0;
For   compteur=lignes:-1:1
      Ij=(vecteur(:,4)./vecteur(:,2)).*exp(-min((vecteur(:,3)-vecteur(:,2)-
      t),0)./k./pmoyen);
```

Étape 6: *Trier les tâches dans l'ordre décroissant des I_j*

```
For   i =1:compteur-1
      For   j = i+1:compteur
            If    Ij(i) < Ij(j)
                  temp=vecteur(j,:);
                  vecteur(j,:)=vecteur(i,:);
                  vecteur(i,:)=temp;
                  temp2=Ij(j);
                  Ij(j)=Ij(i);
                  Ij(i)=temp2;
            End
      End
End
```

Étape 7: *Calcul du temps d'achèvement (Cj) de la tâche j*

```
Cj=vecteur(compteur,2)+t;
t = Cj;
End
```

Étape 8: *Trier les tâches dans l' ordre décroissant des l_j*

```
[lignes,colonnes]=size(vecteur);
For  i=1:lignes-1
     For  j=i+1:lignes
          If  lj(i,1)<lj(j,1)
              temp=vecteur(j,:);
              vecteur(j,:)=vecteur(i,:);
              vecteur(i,:)=temp;
              temp2=lj(j);
              lj(j)=lj(i);
              lj(i)=temp2;
          End
     End
End
```

Étape 9: *Calcul des critères de performances*

```
vecteur=calcul_de_performance(vecteur)
```

Étape 10: *Afficher les résultats*

```
vecteur=résultats(vecteur);
```

ALGORITHME N° 12

Tardiveté apparente du coût et de l'installation « ATCS »

Étape 1: *Appel de la fonction*

```
function regle12(vecteur,sto,combinaison)
```

```
t=0;
```

Étape 2: *Déterminer les dimensions de la matrice*

```
[lignes,colonnes]=size(vecteur);
```

Étape 3: *Calcul des indicateurs*

```
vecteur=calcul_indicateur(vecteur);
```

Étape 4: *Calcul des paramètres*

```
Smoyen=mean (sto(:,2));
Cmax=sum(vecteur(:,2),1)+lignes*smoyen;
dmax=max(vecteur(:,3));
dmin=min(vecteur(:,3));
dmoyen=mean(vecteur(:,3));
r =(dmax-dmin)/cmax;
If    r<=0.5
      k1=4.5+r;
Else  k1=6-2*r;
End
Tau=1-dmoyen/Cmax;
pmoyen=mean(vecteur(:,2));
Éta=smoyen/pmoyen;
k2=Tau/(2*sqrt(eta));
setup_total=0;
```

Étape 5: *Calcul de l'indice de rangement pour chaque tâche j*

```
For   count=1:3
      Ij(t)=wj/pj exp(-min(dj-pj-t,0)/k1pmoyen)exp(-sjk/k2smoyen)
      If    count==1
            indice=vecteur(:,4)./vecteur(:,2).*exp((-min(vecteur(:,3)-
            vecteur(:,2)-t,0))./(k1*pmoyen)).*exp(-sto(:,2)./(k2.*smoyen)));
```

```
Else  combine=combinaison';
      indice=vecteur(:,4)./vecteur(:,2).*exp((-min(vecteur(:,3)-
      vecteur(:,2)-t,0))./(k1*pmoyen)).*exp(-(combine(:,count-
      1)./(k2.*smoyen)));
End
```

Étape 6: *Trier les tâches dans l'ordre décroissant des indices de rangement*

```
For  i=count: lignes-1
     For  j=i+1:lignes
          If   indice(i)<indice(j)
temp=vecteur(j,:);  temp2=sto(j,:);  temp5=combinaison(j,:);
vecteur(j,:)=vecteur(i,:);  sto(j,:)=sto(i,:);  combinaison(j,:)=combinaison(i,:);
vecteur(i,:)=temp;  sto(i,:)=temp2;  combinaison(i,:)=temp5;
          End
     End
End
For  i=count: lignes-1
     For  j=i+1:lignes
          If   indice(i)<indice(j)
               temp5=combinaison(:,j);
               combinaison(:,j)=combinaison(:,i);
               combinaison(:,i)=temp5;
          End
     End
End
```

Étape 7: *Calcul du temps d'achèvement de la tâche et du temps de mise en course*

```
If   count==1
     t=vecteur(1,2)+vecteur(1,5)+sto(1,2)+t;
     setup_total=setup_total+sto(1,2);
```

```
        Else  t=vecteur (count, 2)+combinaison (count-1, count)+t;
              setup_total=setup_total+combinaison(count-1,count);
        End
    End
    t=t+vecteur(lignes,2)+combinaison(lignes-1,lignes);
    setup_total=setup_total+combinaison(lignes-1,lignes);
```

Étape 8: *Calcul des critères de performances*

```
vecteur=calcul_de_performance_combine(vecteur,sto,combinaison);
vecteur
```

Étape 9: *Afficher les résultats*

```
vecteur=resultats_combine(vecteur,setup_total);
```

ALGORITHME N° 13

Algorithme de Johnson pour 2 machines

Étape 1: *Appel de la fonction*

```
function regle13(machine2,nb_machine,donnee2,machinex)
If    nb_machine==2
      disp(' ')
      disp('Règle 13')
End
```

Étape 2: *Déterminer les dimensions des matrices*

```
[lignes,colonnes]=size(donnee2);
matrice_spt=[];
matrice_lpt=[];
```

Étape 3: *Concaténer la matrice de donnée avec celle des pj*

```
donnee2=[donnee2(:,1),machine2,donnee2(:,2:4)];
```

Étape 4: *Séparer les deux ensembles de matrice*

si pj(machine1)<>pj(machine2)

```
i_spt=1;
i_lpt=1;
For compteur=1:lignes
    If  donnee2(compteur,2) <= donnee2(compteur,3)
        matrice_spt(i_spt,:) = donnee2(compteur,:);
        i_spt=i_spt+1;
    Else matrice_lpt(i_lpt,:) = donnee2(compteur,:);
        i_lpt=i_lpt+1;
    End
End
```

Étape 5: *Trier les tâches dans l' ordre croissant des p_j (machine 1)*

```
If  isempty(matrice_spt)==0
    [lignes,colonnes]=size(matrice_spt);
    For i=1:lignes-1
        For j=i+1:lignes
            If  matrice_spt(i,2) >= matrice_spt(j,2)
                temp=matrice_spt(j,:);
                matrice_spt(j,:)=matrice_spt(i,:);
                matrice_spt(i,:)=temp;
            End
        End
    End
End
```

Étape 6: *Trier les tâches dans l'ordre décroissant des p_j (machine 2)*

```
If  isempty(matrice_lpt)==0
    [lignes,colonnes]=size(matrice_lpt);
```

```
For  i=1:lignes-1
     For  j=i+1:lignes
          If   matrice_lpt(i,3) < matrice_lpt(j,3)
               temp=matrice_lpt(j,:);
               matrice_lpt(j,:)=matrice_lpt(i,:);
               matrice_lpt(i,:)=temp;
          End
     End
End
End
```

Étape 8 : *Concaténer les deux matrices triées*

```
[lignes,colonnes]=size(donnee2);
If    isempty(matrice_spt)==1
      donnee2=matrice_lpt;
Elseif isempty(matrice_lpt)==1
       donnee2=matrice_spt;
Else  donnee2=[matrice_spt;matrice_lpt];
End
```

Étape 9: *Convertir la matrice pour être compatible aux fonctions de calcul*

```
Jobs=donnee2 (:, 1);
If    nb_machine==2
      les_pj=donnee2(:,2:3);
Else  les_pj=machinex;
End
```

Étape 10: *Calcul de la somme des temps de traitement (p_j)*

```
Somme_pj=donnee2(:,2)+donnee2(:,3);
donnee2=[jobs,somme_pj,donnee2(:,4:6)];
```

Étape 11: *Calcul des indicateurs*

```
donnee2=calcul_indicateur(donnee2);
```

Étape 12: *Calcul des critères de performances*

```
donnee2=calcul_de_performance(donnee2);
[jobs,les_pj,donnee2(:,3:16)]
```

Étape 10: *Afficher les résultats*

```
donnee2=resultats(donnee2);
```

ALGORITHME N° 14

Heuristique de Palmer pour 3 machines et plus

Étape 1: *Appel de la fonction*

```
function regle14(machinex,nb_machine,donneex)
disp(' ')
disp('Règle 14')
```

Étape 2: *Déterminer les dimensions des matrices*

```
[lignes,colonnes]=size(donneex);
nb_machine=size(machinex,2);
```

Étape 3: *Concaténer la matrice de donnée avec celle des p_j*

```
donneex=[donneex(:,1),machinex,donneex(:,2:4)];
```

Étape 4: *Calcul de l'indice de la pente pour chaque tâche*

```
For   j=1:lignes
      S (j, 1)=0;
      For  i=1:nb_machine
           rep=(nb_machine-(2*i-1)) * donneex(j,1+i);
           S (j, 1) = S (j, 1) + rep;
      End
```

```
End
S = -S;
```

Étape 5: *Trier les tâches dans l'ordre décroissant de l'indice de la pente*

```
For   i=1:lignes-1
      For   j=i+1:lignes
            If    S (i) < S (j)
                  temp=S(j);
                  S (j)=S (i);
                  S (i)=temp;
                  temp=donneex(j,:);
                  donneex(j,:)=donneex(i,:);
                  donneex(i,:)=temp;
            End
      End
End
```

Étape 6: *Convertir la matrice pour qu'elle soit compatible aux les fonctions de calcul*

```
jobs=donneex(:,1);
```

Étape 7: *Calcul de la somme des temps de traitement (p_j)*

```
somme_pj=sum(donneex(:,2:nb_machine+1),2);
les_pj=donneex(:,2:nb_machine+1);
donneex=[jobs,somme_pj,donneex(:,nb_machine+2:nb_machine+4)];
```

Étape 8: *Calcul des indicateurs*

```
donneex=calcul_indicateur(donneex);
```

Étape 9: *Calcul des critères de performances*

```
donneex=calcul_de_performance(donneex);
[lignes,colonnes]=size(donneex);
```

```
[jobs,les_pj,donneex(:,3:16)]
```

Étape 6: *Afficher les résultats*

```
donneex=resultats(donneex);
```

ALGORITHME N° 15

Heuristique de Campbell, Dudek, Smith

Étape 1: *Appel de la fonction*

```
function regle15(machinex,nb_machine,donneex)
clc
disp(' ')
disp('Règle 15')
```

Étape 2: *Déterminer les dimensions des matrices*

```
[lignes,colonnes]=size(donneex);
nb_machine=size(machinex,2);
```

Étape 3: *Ramener la matrice des p_j en une matrice à 2 colonnes*

```
Somme ai = sum(machinex(:,1:nb_machine-1),2)
Somme bi = sum(machinex(:,2:nb_machine),2)
machine2=[sum(machinex(:,1:nb_machine),
),sum(machinex(:,2:nb_machine),2)];
```

Étape 4: *Appel de l'algorithme de Johnson pour résoudre le problème*

```
regle13(machine2, nb_machine, donneex, machinex);
```

ALGORITHME N° 16

Heuristique de Dannenbring

Étape 1: *Appel de la fonction*

```
function regle16(machinex,nb_machine,donneex)
clc
```

```
disp( )
disp( ' règle 16 ')
```

Étape 2: *Déterminer les dimensions de la matrice*

```
[lignes,colonnes]=size(machinex);
nb_machine=size(machinex,2);
a(lignes,1)=0;
b(lignes,1)=0;
```

Étape 2: *Ramener la matrice des p_j à une matrice de deux colonnes*

```
For i=1:lignes
    For j=1:colonnes
        rep=(nb_machine-j+1)*machinex(i,j);
        a(i,1)=a(i,1)+rep;
        rep2=j*machinex(i,j);
        b(i,1)=b(i,1)+rep2;
    End
End
machine2=[a,b];
```

Étape 3: *Appel de l'algorithme de Johnson pour la résolution du problème*

```
regle13(machine2,nb_machine,donneex,machinex);
```

3.7.3 Module de calcul de mesures de performance

Le troisième module sert à générer les mesures de performance. Il est composé de critères répertoriés dans le tableau 16, page 61. Ce module fait appel à 4 fonctions programmées dans Matlab:

1. Fonction de calcul des indicateurs:

Cette fonction calcule les indicateurs sans prendre en compte le facteur temps de mise en course entre deux tâches successives.

Étape 1: *Appel de la fonction*

function [vecteur]=calcul des indicateurs(vecteur)

Étape 2: *Déterminer les dimensions de la matrice*

[lignes,colonnes]=size(vecteur);

Étape 3: *Calcul du temps d'achèvement C_j pour chaque tâche j: 1,2...n*

$C_j = p_j + r_j$

format short

vecteur(1,6)=vecteur(1,2)+vecteur(1,5);

If vecteur(i,5)<vecteur(i-1,6)

vecteur(i,6)=vecteur(i,2)+vecteur(i-1,6);

Else vecteur(i,6)=vecteur(i,2)+vecteur(i,5);

End

End

Étape 4: *Calcul des retards, $T_j = max(C_j-d_j, 0)$*

La tâche est en avance si $C_j-d_j < 0$, $T_j = 0$

La tâche est en retard si $C_j-d_j > 0$, $T_j = C_j-d_j$

For i=1:lignes

If vecteur(i,6)-vecteur(i,3)<= 0

vecteur(i,7)=0;

Étape 5: *Calcul du nombre de tâches en retard U*

$U=1$, si $C_j-d_j > 0$

$U=0$, si $C_j-d_j < 0$

vecteur(i,8)=0; pas de tâche en retard

Else vecteur(i,7)=vecteur(i,6)-vecteur(i,3);

vecteur(i,8)=1; la tâche est en retard

End

End

Étape 6: *Calcul du coût d'ordonnancement*

Coût = w_j * C

vecteur(:,14) = vecteur(:,4) .* vecteur(:,6);

Étape 7: *Calcul du coût du retard*

w_j * (C_j-d_j); *Si* C_j-d_j = > 0

vecteur(:,15) = vecteur(:,4) .* vecteur(:,7);

Étape 8: *Calcul du temps de passage des tâches dans l'atelier*

$F_j = C_j$-r_j

vecteur(:,16)=vecteur(:,6)-vecteur(:,5);

2. Fonction de calcul des indicateurs combinés

Elle calcule les indicateurs en prenant en compte le facteur temps de mise en course entre deux tâches successives.

Étape 1: *Appel de la fonction*

function [vecteur,setup_total]=calcul_des_indicateurs_combines(vecteur, sto, combinaison)

Étape 2: *Déterminer les dimensions de la matrice*

[lignes,colonnes]=size(vecteur);

Étape 3: *Calcul du temps d'achèvement (C_j) pour chaque tâche j:1,2...n*

$C_j = p_j + r_j$

format short

setup_total=0;

vecteur(1,6)=vecteur(1,2)+vecteur(1,5)+sto(1,2);

setup_total=sto(1,2);

For i=2:lignes

```
        If  vecteur(i,5)<vecteur(i-1,6)
                vecteur(i,6)=vecteur(i,2)+vecteur(i-1,6)+combinaison(i-1,i);
        Else    vecteur(i,6)=vecteur(i,2)+vecteur(i,5)+combinaison(i-1,i);
        End
                setup_total=setup_total+combinaison(i-1,i);
    End
```

Étape 4: *Calcul des retards.*

$T_j = max(C_j - d_j, 0)$

La tâche est en avance $T_j = 0$ *si* $C_j - d_j \leq 0$

La tâche est en retard $T_j = cj - dj$ *si* $C_j - d_j > 0$

```
For  i=1:lignes
     If  vecteur(i,6)-vecteur(i,3)<= 0
         vecteur(i,7)=0;
```

Étape 5: *Calcul du nombre de tâches en retard U*

$U = 1$, *si* $C_j - d_j > 0$

$U = 0$, *si* $C_j - d_j < 0$

```
vecteur(i,8)=0;  pas de tâche en retard
     Else   vecteur(i,7)=vecteur(i,6)-vecteur(i,3);
            vecteur(i,8)=1;  la tâche est en retard
     End
End
```

Étape 6: *Calcul du coût d'ordonnancement* $w_j * C_j$

```
vecteur(:,14) = vecteur(:,4) .* vecteur(:,6);
```

Étape 7: *Calculer le coût de retard*

$wj * (C_j - d_j)$; *Si* $C_j - d_j => 0$

```
vecteur(:,15) = vecteur(:,4) .* vecteur(:,7);
```

Étape 8: *Calcul du temps de passage des tâches dans l'atelier*

$F_j = C_j - r_j$

```
vecteur(:,16)=vecteur(:,6)-vecteur(:,5);
setup_total;
```

3. Fonction de calcul des ratios:

Elle calcule les ratios pour les algorithmes n° 5, 6, 7, 8, 9 et 10

Étape 1: *Appel de la fonction*

```
function [vecteur]=calcul_des_ratios (vecteur)
format short
```

Étape 2: *Déterminer les dimensions de la matrice*

```
[lignes,colonnes]=size(vecteur);
```

Étape 3: *Calcul de la marge dynamique* $(d_j - p_j - r_{ij})$

```
vecteur(:,9)=vecteur(:,3)-vecteur(:,2)-vecteur(:,5);
```

Étape 4: *Calcul du ratio critique de la marge dynamique RCDM1* $(d_j - p_j - r_{ij})/d_j$

```
vecteur(:,10)=(vecteur(:,3)-vecteur(:,2)-vecteur(:,5))./vecteur(:,3);
```

Étape 5: *Calcul du ratio critique de la marge dynamique RCDM2* $(d_j - p_j - r_{ij}))\ p_j$

```
vecteur(:,11)=(vecteur(:,3)-vecteur(:,2)-vecteur(:,5))./vecteur(:,2);
```

Étape 6: *Calcul du ratio critique de la marge dynamique RCDM3* $(d_j\ p_j)$

```
vecteur(:,12)=vecteur(:,3)./vecteur(:,2);
```

Étape 7: *Calcul des* $w_j\ p_j$

```
vecteur(:,13)=vecteur(:,4)./vecteur(:,2);
```

Étape 8: *Calcul des* $w_j * C_j$

```
vecteur(:,14)=vecteur(:,5).*vecteur(:,4);
```

Étape 9: *Calcul (w_j*T_j)*

vecteur(:,15)=vecteur(:,4).*vecteur(:,6);

4. Fonction résultats:

Cette fonction calcule les critères de performance (critères économiques cités dans le tableau 16,page 61) pour la séquence générée par le SBC et les affiche sur l'écran de l'ordinateur de l'usager.

Étape 1: *Appel de la fonction*

function [vecteur]= resultats(vecteur)

Étape 2: *Déterminer les dimensions de la matrice*

[lignes,colonnes]=size(vecteur);

Étape 3: *Calcul du coût total de l'ordonnancement*

sum(vecteur(:,14),1);

Étape 4: *Calcul du coût moyen de l'ordonnancement*

real(cout_total_ordonnancement/lignes);

Étape 5: *Calcul du coût du retard*

coût du retard= w_j*T_j

Coût_total_des_retards=sum(vecteur(:,15),1);

Coût_moyen_des_retards=real(cout_total_des_retards/lignes);

Étape 6: *Calcul de la somme des p_j*

p_j=vecteur(:,2);

Étape 7: *Calcul de temps total de l'ordonnancement*

Makespan= max*(C_j)*

make_span=max(vecteur(:,6));

Étape 8: *Calcul de la somme des retards*

```
retards=vecteur(:,7);
somme_des_retards=sum(retards,1);
```

Étape 9: *Calcul du retard moyen des tâches*

```
retard_moyen_des_taches=real(somme_des_retards/lignes);
```

Étape 10: *Calcul du nombre de tâches en retard*

```
nb_retard=vecteur(:,8);
nombre_de_tâches_en_retard=sum(nb_retard,1);
```

Étape 11: *Calcul de la somme totale des temps d'achèvement des tâches " Completion time"*

```
Nb_achevements=vecteur(:,6);
 Total_completion_time=sum(nb_achevements,1);
 Completion_time_moyen=real(total_completion_time/lignes);
```

Étape 12: *Calcul du temps moyen des retards*

somme des retards nombre de lots en retards

```
If  nombre_de_jobs_en_retard == 0
    temps_moyen_des_retards=0;
Else
temps_moyen_des_retards=real(somme_des_retards/nombre_de_jobs_en    retard);
End
```

Étape 13: *Calcul du temps total de passage des tâches dans l'atelier « Flow time total »*

```
flow_time_total=sum(vecteur(:,16));
```

Étape 14: *Calcul du temps moyen de passage des tâches dans l'atelier*

```
flow_time_moyen=real(flow_time_total/lignes);
```

Étape 15: *Calcul du nombre moyen de tâches dans le système*

```
nombre_moyen_de_job_dans_le_systeme=
real(flow_time_total/sum(vecteur(:,2),1));
```

Étape 16: *Calcul du temps libre de la machine*

```
idle_time=make_span-sum(vecteur(:,2));
```

Étape 17: *Affichage des mesures de performances*

```
fprintf('\n')
sequence=vecteur(:,1)'
fprintf('make_span = %4.3g\n\n',make_span)
fprintf('somme_des_retards = %4.3g\n\n',somme_des_retards)
fprintf('retard_moyen_des_taches=%4.3g\n\n',retard_moyen_des_taches)
fprintf('nombre_jobs_en_retard =4.3g\n\n',nombre_de_jobs_en_retard)
fprintf('temps_moyen_des_retards=%4.3g\n\n',temps_moyen_des_retards)
fprintf('total_completion_time = %4.3g\n\n',total_completion_time)
fprintf('completion_time_moyen = %4.3g\n\n',completion_time_moyen)
fprintf('flow_time_total = %4.3g\n\n',flow_time_total)
fprintf('flow_time_moyen = %4.3g\n\n',flow_time_moyen)
fprintf('nombre_moyen_de_job_dans_le_systeme=
        %4.3g\n\n',nombre_moyen_de_job_dans_le_systeme)
fprintf('cout_total_ordonnancement= %4.3g\n\n',cout_total_ordonnancement)
fprintf('cout_moyen_ordonnancement=
        %4.3g\n\n',cout_moyen_ordonnancement)
fprintf('cout_total_des_retards= %4.3g\n\n',cout_total_des_retards)
fprintf('cout_moyen_des_retards= %4.3g\n\n',cout_moyen_des_retards)
fprintf('idle_time= %4.3g\n\n',idle_time)
fprintf('\n\n')
```

3.8 Marche à suivre pour l'utilisation du logiciel

La procédure de l'utilisation du système est très simple et passe par les étape suivantes :

- La première étape: cette étape consiste à lancer le logiciel en tapant ,

  » ***systeme_expert***

- La deuxième étape : l'interface du système pose la question concernant le nombre de machines constituant le système de production, il faut répondre en tapant le chiffre 1,2,3...

  » ***Combien y a-t-il de machine? 1***

- La troisième étape: l'usager doit introduire les données concernant les tâches candidate pour l'ordonnancement tels que:

  - Colonne 1: le numéro de la tâche,
  - Colonne 2: son temps de traitement,
  - Colonne 3: sa date d'échéance,
  - Colonne 3: son facteur d'importance,
  - Colonne 4: sa date de disponibilité.

  » ***Entrer la matrice de donnée*** : [numéro de la tâche, son temps de traitement, sa date d'échéance, son facteur d'importance et sa date de disponibilité].

- La quatrième étape: l'usager doit choisir l'objectif à optimiser en tapant les chiffres de 1 à 6.

  Minimisation du temps total d'ordonnancement et la discipline premier arrivé premier servi (1)

  Minimisation du temps d'achèvement de tâche (2)

  Rencontrer les délais clients (3)

  Minimisation du coût lié au retard (4)

  Minimisation du retard total (5)

  Minimisation du temps de mise en course et le retard total pondéré (6)

  » ***Entrer le chiffre correspondant à l'objectif : 1***

Une fois que nous avons procédé à toutes les étapes citées ci haut, le système nous génèrera la séquence avec ces critères de performance (critères économiques) que l'usager les utilisera pour le choix de sa séquence (plan de production) optimale.

3.9 Conclusion

Le système de base de connaissances pour l'ordonnancement dynamique est un outil qui peut condenser les informations rattachées aux conditions préalables pour l'implantation effective de certaines approches, méthodologies. Le choix d'une telle approche a été motivé par les nombreuses possibilités que peut offrir un tel outil, à savoir; la flexibilité, la simplicité dans son utilisation, et sa performance dans la résolution des problèmes complexes d'ordonnancement. Ce système demeure plus efficace qu'un système conventionnel d'aide à la décision. C'est dans ce contexte que nous avons opté à utiliser un système à base de connaissances dans un processus de développement d'un programme d'ordonnancement.

Le modèle prend l'avantage de certaines caractéristiques des approches heuristiques traditionnelles dans le but de résoudre des problèmes d'ordonnancement avec moins de temps de calcul. Il prend aussi l'avantage des compétences humaines d'une façon interactive. Le but n'est pas de remplacer le planificateur, mais mettre ensemble l'expertise humaine et la capacité de l'outil pour le choix d'une séquence optimale à travers le système à base de connaissances. L'intérêt d'un tel modèle est la facilité de modifier, de supprimer ou d'ajouter de la connaissance dans le système sans avoir recours à rebâtir complètement ce dernier. Mais, la connaissance doit être présentée de façon qu'il serait plus facile à l'utilisateur à comprendre avec peu ou aucune connaissance de programmation.

Les avantages de notre outil sont:

- Le nombre de tâches à traiter est illimité;
- Le nombre de machine est aussi illimité;
- Le temps d'exécution est très faible.

Par contre l'inconvénient est:

- L'outil génère des connaissances et c'est à l'usager de choisir un compromis entre les critères pour déterminer la séquence optimale.

Nous présentons dans le chapitre qui suit un exemple d'illustration quant à l'utilisation de l'outil pour la résolution de problème d'ordonnancement dans différentes situations.

CHAPITRE 4

UTILISATION DE L'OUTIL

4.1 Étude de cas

Nous allons voir dans ce chapitre comment nous pouvons utiliser l'outil pour répondre à certaines questions qui se posent au niveau du plancher de l'usine quand nous sommes devant un problème complexe d'ordonnancement. À travers cet exemple, nous allons exposer quelques cas d'ordonnancement possibles qui ont été traités dans les paragraphes (2.3.1, 2.3.2, 2.3.3) du chapitre 2, pour montrer la puissance et la flexibilité qu'offre un tel système. Les résultats générés par le SBC seront compilés dans le tableau 21, page 117, pour permettre à l'usager d'interpréter facilement ces derniers.

4.2 Énoncé du problème

Nous considérons un problème d'ordonnancement d'un environnement manufacturier de fabrication de microprocesseurs. Le nombre de commandes de pièces s'élève à 10 lots. Le problème est de trouver les plans de production optimaux dans chacune des situations suivantes:

- Cas 1: Atelier composé d'une seule machine pour le marquage des pièces ;
- Cas 2: Atelier composé de 2 machines pour la fixation de la puce sur le substrat;
- Cas 3: Atelier composé de 3 machines pour la fixation de boules de contact.

Les temps de traitement, la date de disponibilité, la date de livraison et le facteur d'importance (poids) des tâches (lots) sont donnés dans le tableau 17, page 102.

Les temps et les autres informations de fabrication concernant ces tâches tel que le statut des machines sont introduites dans le système et dépendamment de l'objectif recherché, le système de base de connaissances (SBC) sélectionne le groupe de règles d'ordonnancement

appropriées (tableau 15, page 60) et différentes séquences sont générées. Les différents critères économiques (tableau 16, page 61) sont évalués pour le choix de la séquence optimale.

4.3 Résolution des problèmes en utilisant le SBC

4.3.1 Cas 1: Atelier composé d'une machine pour le marquage des pièces

Nous supposons que notre système de production est composé d'une seule machine, et les caractéristiques des tâches (lots) sont regroupées dans le tableau 17.

Tableau 17

Matrice des caractéristiques des commandes (Temps en heures)

Nombre de machines (m=1)							
N° de tâches (Lots)	Famille de produit	N° de pièce	Qté (pces)	Temps de traitement (p_j)	Délai de livraison (d_j)	type de tâche (w_j)	Date de disponibilité (r_j)
1	32 CCGA	L3091	433	10	40	1	0
2	42 630FP	K4320	800	25	55	2	5
3	32 C MAC	C0120	450	12	40	3	2
4	32 COBR	K4320	420	10	40	1	3
5	25 LONE	H6351	470	15	40	3	5
6	32 CCGA	L7769	750	22	60	2	0
7	32 CLASP	L7967	480	13	45	4	1
8	32X42 C	K3130	600	20	60	1	5
9	25 PYTHON	K1867	420	10	40	2	2
10	40 CCGA	L3091	475	15	45	5	3

La base de connaissances choisira les algorithmes appropriés pour la résolution du problème d'une seule machine en fonction de l'objectif à atteindre. Dans notre cas, nous essayons de

générer toutes les séquences possibles et l'usager choisira celle qui répond au critère d'optimisation. Alors l'utilisation de l'outil passe par les étapes décrites au paragraphe 3.8, soit la procédure suivante:

- La première étape: lancer le logiciel en tapant "systeme_expert",
  » systeme_expert

- La deuxième étape : introduire le nombre de machines qui est de 1.
  » Combien y a-t-il de machine? 1

- La troisième étape: Introduire les données comme suit:
  - Colonne 1: le numéro de la tâche,
  - Colonne 2: son temps de traitement,
  - Colonne 3: sa date d'échéance,
  - Colonne 3: son facteur d'importance,
  - Colonne 4: sa date de disponibilité.

  » Entrer la matrice de donnée :

  [1,10,40,1,0;
  2,25,55,2,5;
  3,12,40,3,2;
  4,10,40,1,3;
  5,15,40,3,5;
  6,22,60,2,0;
  7,13,45,4,1;
  8,20,60,1,5;
  9,10,40,2,2;
  10,15,45,5,3]

- La quatrième étape: choisir l'objectif à optimiser en tapant les chiffres de 1 à 6.

» Quel est l'objectif d'optimisation recherché?

Minimisation du temps total d'ordonnancement et la discipline premier arrivé premier servi *(1)*

Minimisation du temps d'achèvement de tâche *(2)*

Rencontrer les délais clients *(3)*

Minimisation du coût lié au retard *(4)*

Minimisation du retard total *(5)*

Minimisation du temps de mise en course et le retard total pondéré *(6)*

Une fois que nous avons procédé à toutes les étapes citées ci haut, le système génère la séquence avec ces critères de performance (critères économiques) que l'usager les utilise pour le choix de sa séquence (plan de production).

» systeme_expert

» Combien y a-t-il de machine? 1

» Entrer le chiffre correspondant à l'objectif : 1

Résultats de l'ordonnancement générés par le SBC en utilisant la règle n°1

Algorithme N°		1
séquence	=	39410825
make_span	=	152.00
somme_des_retards	=	374.00
retard_moyen_des_taches	=	37.40
nombre_de_jobs_en_retard	=	7.00
temps_moyen_des_retards	=	53.40
total_completion_time	=	781.00
completion_time_moyen	=	78.10
flow_time_total	=	755.00

flow_time_moyen	=	75.50
nombre_moyen_de_job dans_le_systeme	=	4.97
cout_total_ordonnancement	=	1940.00
cout_moyen_ordonnancement	=	194.00
cout_total_des_retards	=	929.00
cout_moyen_des_retards	=	92.90
idle_time	=	0.00

» ***systeme_expert***

» ***Combien y a-t-il de machine ? 1***

» ***Entrer le chiffre correspondant à l'objectif : 2***

Résultats de l'ordonnancement générés par le SBC en utilisant la règle n°2:

Algorithme N°		2	3
sequence	=	14937510862	26851073491
make_span	=	152.00	157.00
somme_des_retards	=	291.00	594.00
retard_moyen_des_taches	=	29.10	59.40
nombre_de_jobs_en_retard	=	7.00	8.00
temps_moyen_des_retards	=	41.60	74.20
total_completion_time	=	696.00	1030.00
completion_time_moyen	=	69.60	103.00
flow_time_total	=	670.00	1000.00
flow_time_moyen	=	67.00	100.00
nombre_moyen_de_job_ dans_le_systeme	=	4.41	6.58
cout_total_ordonnancement	=	1730.00	2440.00
cout_moyen_ordonnancement	=	173.00	244.00

cout_total_des_retards	=	709.00	1410.00
cout_moyen_des_retards	=	70.90	141.00
idle_time	=	0.00	5.00

» systeme_expert

» Combien y a-t-il de machine? 1

» Entrer le chiffre correspondant à l'objectif : 3

Résultats de l'ordonnancement générés par le SBC en utilisant la règle n°3:

Algorithme N°		4
sequence		=13459710268
make_span	=	152.00
somme_des_retards	=	308.00
retard_moyen_des_taches	=	30.80
nombre_de_jobs_en_retard	=	7.00
temps_moyen_des_retards	=	44.00
total_completion_time	=	717.00
completion_time_moyen	=	71.70
flow_time_total	=	691.00
flow_time_moyen	=	69.10
nombre_moyen_de_job_		
dans_le_systeme	=	4.55
cout_total_ordonnancement	=	1700.00
cout_moyen_ordonnancement	=	170.00
cout_total_des_retards	=	701.00
cout_moyen_des_retards	=	70.10
idle_time	=	0.00

» systeme_expert

»Combien y a-t-il de machine à 1

»Entrer le chiffre correspondant à l'objectif : 4

Résultats de l'ordonnancement générés par le SBC en utilisant la règle n°4:

Algorithme N°	11
sequence	=10735914628
make_span	= 155.00
somme_des_retards	= 360.00
retard_moyen_des_taches	= 36.00
nombre_de_jobs_en_retard	= 8.00
temps_moyen_des_retards	= 45.00
total_completion_time	= 784.00
completion_time_moyen	= 78.40
flow_time_total	= 758.00
flow_time_moyen	= 75.80
nombre_moyen_de_job_	
dans_le_systeme	= 4.99
cout_total_ordonnancement	= 1460.00
cout_moyen_ordonnancement	= 146.00
cout_total_des_retards	= 560.00
cout_moyen_des_retards	= 56.00
idle_time	= 3.00

» systeme_expert

» Combien y a-t-il de machine? 1

» Entrer le chiffre correspondant à l'objectif: 5

Résultats de l'ordonnancement générés par le SBC en utilisant la règle n°5:

Algorithmes	5	6	7	8	9	10
séquences	52341091786	25810634791	25681037491	25681037491	82614593710	10739514628
makespan	157.00	157.00	157.00	157.00	157.00	157.00
Somme des retards	437.00	561.00	573.00	573.00	548.00	355.00
Retard moyen des tâches	43.70	56.10	57.30	57.30	54.80	35.50
Nbre de tâches en retard	8.00	9.00	9.00	9.00	8.00	8.00
Temps moyen des retards	54.60	62.30	63.70	63.70	68.50	44.40
Total completion time	872.00	1000.00	1010.00	1010.00	973.00	779.00
Completion time moyen	87.20	100.00	101.00	101.00	97.30	77.90
Flow time total	846.00	975.00	987.00	987.00	947.00	753.00
Flow time moyen	84.60	97.50	98.70	98.70	94.70	75.30
Nbre moyen de tâches dans le	5.57	6.41	6.49	6.49	6.23	4.95
Coût total de l'ordonnancement	1990.00	2330.00	2360.00	2360.00	2740.00	1460.00
Coût moyen de l'ordonnancement	199.00	233.00	236.00	236.00	274.00	146.00
Coût total des retards	978.00	1280.00	1320.00	1320.00	1690.00	560.00
Coût moyen des retards	97.80	128.00	132.00	132.00	169.00	56.00
Idle time	5.00	5.00	5.00	5.00	5.00	3.00

Nous pouvons tiré comme conclusion que la séquence optimale générée par cette la règle N°5 est: 10739514628

Dans le cas où un temps de mise en course entre deux tâches successives de nature différente est exigé, alors pour minimiser ce dernier , la base de connaissance fait appel à l'algorithme N°12. Pour résoudre un tel problème, nous faisons appel aux étapes suivantes:

Étape 1 : nous faisons appel au SBC en tapant "system_expert"

» systeme_expert

Étape 2: nous répondons à la question ci dessous en tapant le chiffre correspondant au nombre de machine constituant le système de production.

» Combien y a-t-il de machine? 1

Étape 3: nous introduisons la matrice des caractéristiques des tâches.

» Entrer la matrice de donnée :

[1,10,40,1,0;
2,25,55,2,5;
3,12,40,3,2;
4,10,40,1,3;
5,15,40,3,5;
6,22,60,2,0;
7,13,45,4,1;
8,20,60,1,5;
9,10,40,2,2;
10,15,45,5,3]

Étape 4: nous introduisons le chiffre correspondant à l'objectif.

» Entrer le chiffre correspondant à l'objectif : 6

Étape 5: nous introduisons la matrice des temps de mise en course à l'instant t_0.

» Entrer la matrice de setup initial: [1,2;2,2;3,1;4,2;5,0;6,2;7,1;8,0;9,2;10,1]

Étape 6: nous introduisons la matrice des temps de mise en course des séquences dépendantes:

» ***Entrer la matrice de setup combiné :***

[0,2,1,3,5,2,4,3,0,1;
2,0,3,2,4,5,1,0,1,2;
4,3,0,1,2,4,0,5,1,1;
2,1,3,0,2,3,2,1,0,4;
1,2,3,4,0,2,4,2,3,1;
2,4,3,1,2,0,1,2,1,0;
2,3,1,4,2,1,0,1,2,0;
2,2,2,1,3,3,0,0,1,1;
3,1,2,1,2,1,2,2,0,0;
1,2,3,4,1,2,1,2,1,0]

Résultats de l'ordonnancement générés par le SBC en utilisant la règle n°6:

Algorithme N°	12
sequence	=10739628514
make_span	= 168.00
somme_des_retards	= 682.00
retard_moyen_des taches	= 68.20
nombre_de_jobs_en retard	= 9.00
temps_moyen_des_retards	= 75.80
total_completion_time	= 919.00
completion_time_moyen	= 91.90
flow_time_total	= 919.00
flow_time_moyen	= 91.90
nombre_moyen_de_job dans_le_systeme	= 6.05

cout_total_ordonnancement = 1680.00
cout_moyen_ordonnancement = 168.00
cout_total_des_retards = 1140.00
cout_moyen_des_retards = 114.00
idle_time = 16.00
setup_total = 6.00

4.3.2 Cas 2: Atelier composé de 2 machines pour la fixation de la puce

Dans le cas d'un environnement de 2 machines, le système choisit l'algorithme N°13 (Johnson) pour résoudre le problème. Nous avons besoin des 2 matrices de données illustrées dans le tableau 18 et 19 suivants.

Tableau 18

Matrice des caractéristiques des commandes

Nombre de machines (m=1)						
N° de tâches	Famille de produit	N° de pièce	Qté (pces)	Délais de livraison (d_j)	type de tâche (w_j)	Date de disponibilité (r_j)
1	32 CCGA	L3091	433	40	1	0
2	42 630FP	K4320	800	55	2	5
3	32 C MAC	C0120	450	40	3	2
4	32 COBR	K4320	420	40	1	3
5	25 LONE	H6351	470	40	3	5
6	32 CCGA	L7769	750	60	2	0
7	32 CLASP	L7967	480	45	4	1
8	32X42 C	K3130	600	60	1	5
9	25 PYTHON	K1867	420	40	2	2
10	40 CCGA	L3091	475	45	5	3

Tableau 19

Matrice des temps de traitement des tâches sur 2 machines

Tâches (j)	Nombre de machines = 2	
	p_{1j}	p_{2j}
1	5	5
2	10	15
3	6	6
4	5	5
5	7	8
6	10	12
7	6	7
8	10	10
9	5	5
10	8	7

Étape 1: Taper "systeme_expert"

» systeme_expert

Étape 2: Introduire le chiffre 2 correspondant au nombre de machines composant notre système de production.

» Combien y a-t-il de machine? 2

Étape 3: Introduire la matrice des données (n° de tâches, date de livraison, le facteur d'importance, et la date de disponibilité).

» Entrer la matrice de donnee2 :

[1,40,1,0;

2,55,2,5;

3,40,3,2;

4,40,1,3;

5,40,3,5;

6,60,2,0;

7,45,4,1;

8,60,1,5;

9,40,2,2;

10,45,5,3].

Étape 4: Introduire la matrice des temps de traitement des tâches sur chaque machine.

»Entrer la matrice des p_j : [5,5;10,15;6,6;5,5;7,8;10,12;6,7;10,10;5,5;8,7]

Résultats de l'ordonnancement pour un problème de 2 machines:

Algorithme N°	13
sequence	=94173586210
make_span	= 154.00
somme_des_retards	= 328.00
retard_moyen_des_taches	= 32.80
nombre_de_jobs_en_retard	= 6.00
temps_moyen_des_retards	= 54.70
total_completion_time	= 739.00
completion_time_moyen	= 73.90
flow_time_total	= 713.00
flow_time_moyen	= 71.30
nombre_moyen_de_job_dans le_systeme	= 4.69
cout_total_ordonnancement	= 2010.00
cout_moyen_ordonnancement	= 201.00
cout_total_des_retards	= 1000.00

cout_moyen_des_retards = 100.00

idle_time = 2.00

4.3.3 Cas 3 : Atelier de fixation boules de contact composé de 3 machines

Le cas d'un environnement composé de 3 machines et plus, le système (SBC) fait appel aux algorithmes N°14, 15 et 16 pour résoudre le problème. Nous supposons pour titre de comparaison des résultats générés par le système pour différentes situations que nous avons énumérées dans l'énoncé du problème. Donc nous avons besoins des données du tableau 18, page 111 et la matrice des temps de traitement sur chaque machines (tableau 20). A titre d'exemple nous considérons le problème à 3 machines.

Tableau 20

Matrice des temps de traitement des tâches sur 3 machines

Tâches (j)	Nombre de machines = 3		
	p_{1j}	p_{2j}	p_{3j}
1	2	4	4
2	10	5	10
3	4	4	4
4	2	4	4
5	5	5	5
6	10	6	6
7	3	5	5
8	6	6	8
9	2	4	4
10	5	5	5

Étape 1: Taper "systeme_expert"

» ***systeme_expert***

Étape 2: Introduire le chiffre 3 correspondant au nombre de machines composant notre système de production.

» Combien y a-t-il de machine? 3

Étape 3: Introduire la matrice des données(n° de tâches, date de livraison, le facteur d'importance, et la date de disponibilité) illustrées dans le tableau 18, page 110.

» Entrer la matrice de donneex :

[1,40,1,0;
2,55,2,5;
3,40,3,2;
4,40,1,3;
5,40,3,5;
6,60,2,0;
7,45,4,1;
8,60,1,5;
9,40,2,2;
10,45,5,3]

Étape 4: Introduire la matrice des temps de traitement des tâches sur chaque machine illustrés dans le tableau 20.

» Entrer la matrice des p_j :

[2,4,4;10,5,10;4,4,4;2,4,4;5,5,5;10,6,6;3,5,5;6,6,8;2,4,4;5,5,5]

Résultats de l'ordonnancement pour un problème de 3 machines:

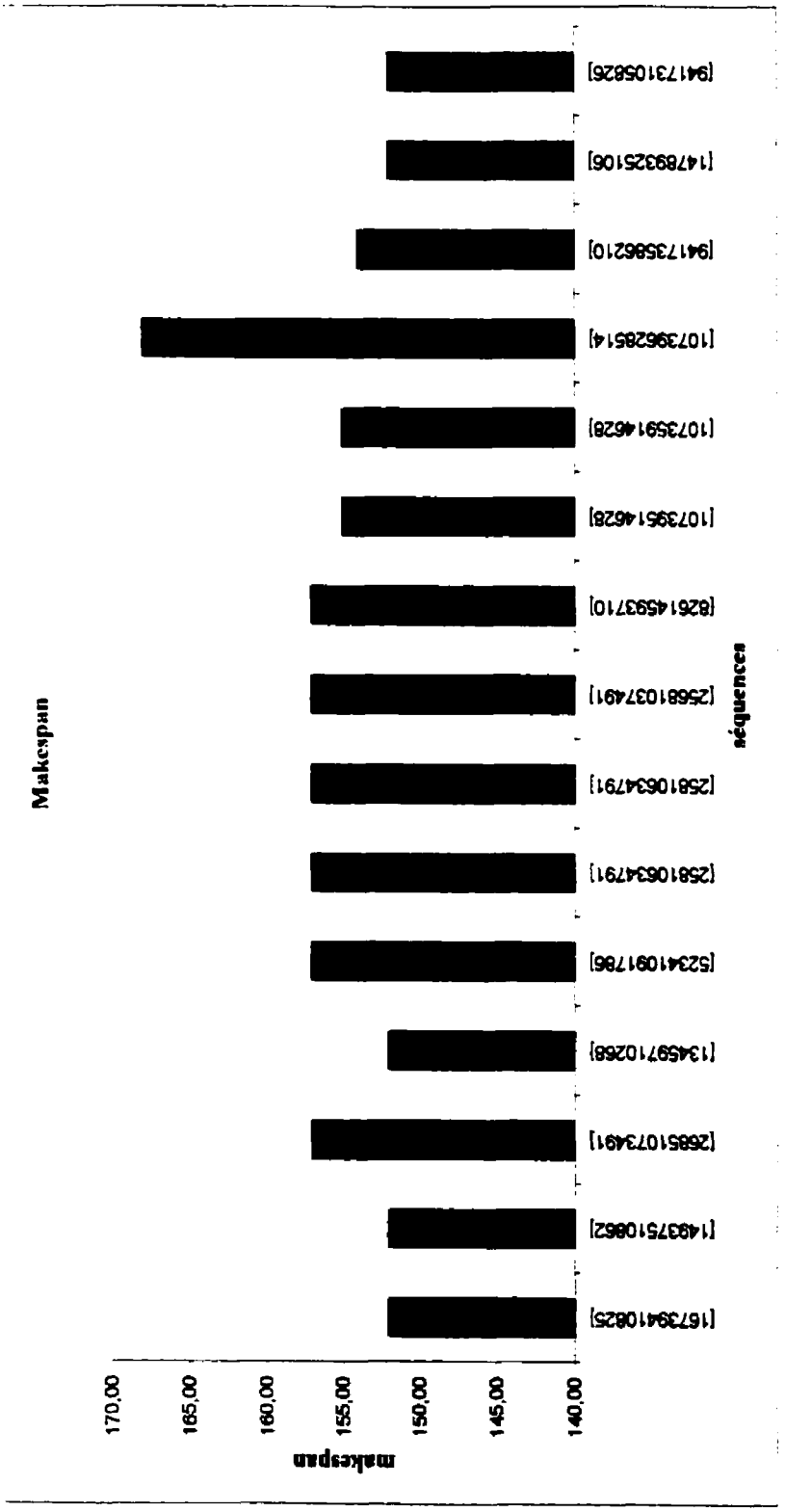
Algorithme N°	14	15
sequence	=14789325106	94173105826
make_span	= 152.00	154.00
somme_des_retards	= 355.00	309.00
retard_moyen_des_taches	= 35.50	30.90

nombre_de_jobs_en_retard	= 6.00	6.00
temps_moyen_des_retards	= 59.20	51.50
total_completion_time	= 751.00	720.00
completion_time_moyen	= 75.10	72.00
flow_time_total	= 725.00	694.00
flow_time_moyen	= 72.50	69.40
nombre_moyen_de_job dans_le_systeme	= 4.77	4.57
cout_total_ordonnancement	=2060.00	1730.00
cout_moyen_ordonnancement	= 206.00	173.00
cout_total_des_retards	=1080.00	716.00
cout_moyen_des_retards	= 108.00	71.60
idle_time	= 0.00	2.00

Pour fin d'analyse, nous avons compilé les résultats générés par le SBC dans le tableau 21, pour faciliter la comparaison des critères de performance. En fonction de ces critères, l'usager choisit sa séquence (son plan de production) pour répondre aux objectifs d'optimisation recherchés.

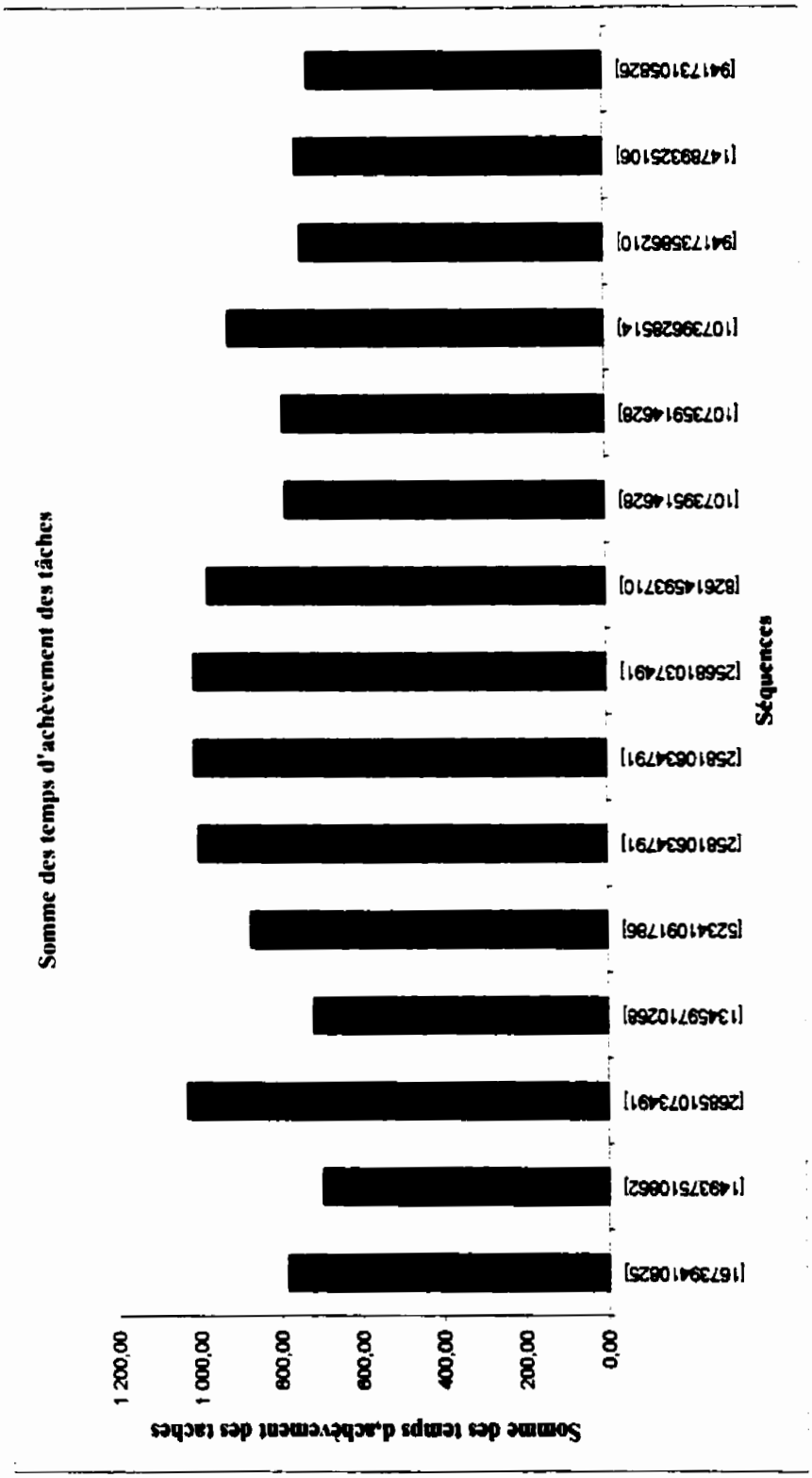
Tableau 21 : Compilation des résultats générés par le SBC

Séquence	Algorithme	Makespan	Somme des retards	Retard moyen	Nbre de tâches en retard	Temps moyen des retards	Somme des temps d'achèvement des tâches	Temps d'achèvement moyen des tâches	Temps de passage total	temps moyen de passage	Nbre moyen des tâches dans le système	Coût total de l'ordonnancement	Coût moyen de l'ordonnancement	Coût total des retards	Coût moyen des retards	Temps libre des machines
16739410825	1	152,00	374,00	37,40	7	53,40	781,00	78,10	755,00	75,50	4,97	1 940,00	194,00	929,00	92,90	0
14937510862	2	152,00	291,00	29,10	7	41,60	696,00	69,60	670,00	67,00	4,41	1 730,00	173,00	709,00	70,90	0
26851073491	3	157,00	594,00	59,40	8	74,20	1 030,00	103,00	1 000,00	100,00	6,58	2 440,00	244,00	1 410,00	141,00	5
13459710268	4	152,00	308,00	30,80	7	44,00	717,00	71,70	691,00	69,10	4,55	1 700,00	170,00	701,00	70,10	0
52341091786	5	157,00	437,00	43,70	8	54,60	872,00	87,20	846,00	84,60	5,57	1 990,00	199,00	978,00	97,80	5
25810634791	6	157,00	561,00	56,10	9	62,30	1 000,00	100,00	975,00	97,50	6,41	2 330,00	233,00	1 280,00	128,00	5
25810634791	7	157,00	573,00	57,30	9	63,70	1 010,00	101,00	987,00	98,70	6,49	2 360,00	236,00	1 320,00	132,00	5
25681037491	8	157,00	573,00	57,30	9	63,70	1 010,00	101,00	987,00	98,70	6,49	2 360,00	236,00	1 320,00	132,00	5
82614593710	9	157,00	548,00	54,80	8	68,50	973,00	97,30	947,00	94,70	6,23	2 740,00	274,00	1 690,00	169,00	5
10739514628	10	155,00	355,00	35,50	8	44,40	779,00	77,90	753,00	75,30	4,95	1 460,00	146,00	560,00	56,00	3
10735914628	11	155,00	360,00	36,00	8	45,00	784,00	78,40	758,00	75,80	4,99	1 460,00	146,00	560,00	56,00	3
10739628514	12	168,00	682,00	68,20	9	75,80	919,00	91,90	919,00	91,90	6,05	1 680,00	168,00	1 140,00	114,00	16
94173586210	13	154,00	328,00	32,80	6	54,70	739,00	73,90	713,00	71,30	4,69	2 010,00	201,00	1 000,00	100,00	2
14789325106	14	152,00	355,00	35,50	6	59,20	751,00	75,10	725,00	72,50	4,77	2 060,00	206,00	1 080,00	108,00	0
94173105826	15	154,00	309,00	30,90	6	51,50	720,00	72,00	694,00	69,40	4,57	1 730,00	173,00	716,00	71,60	2

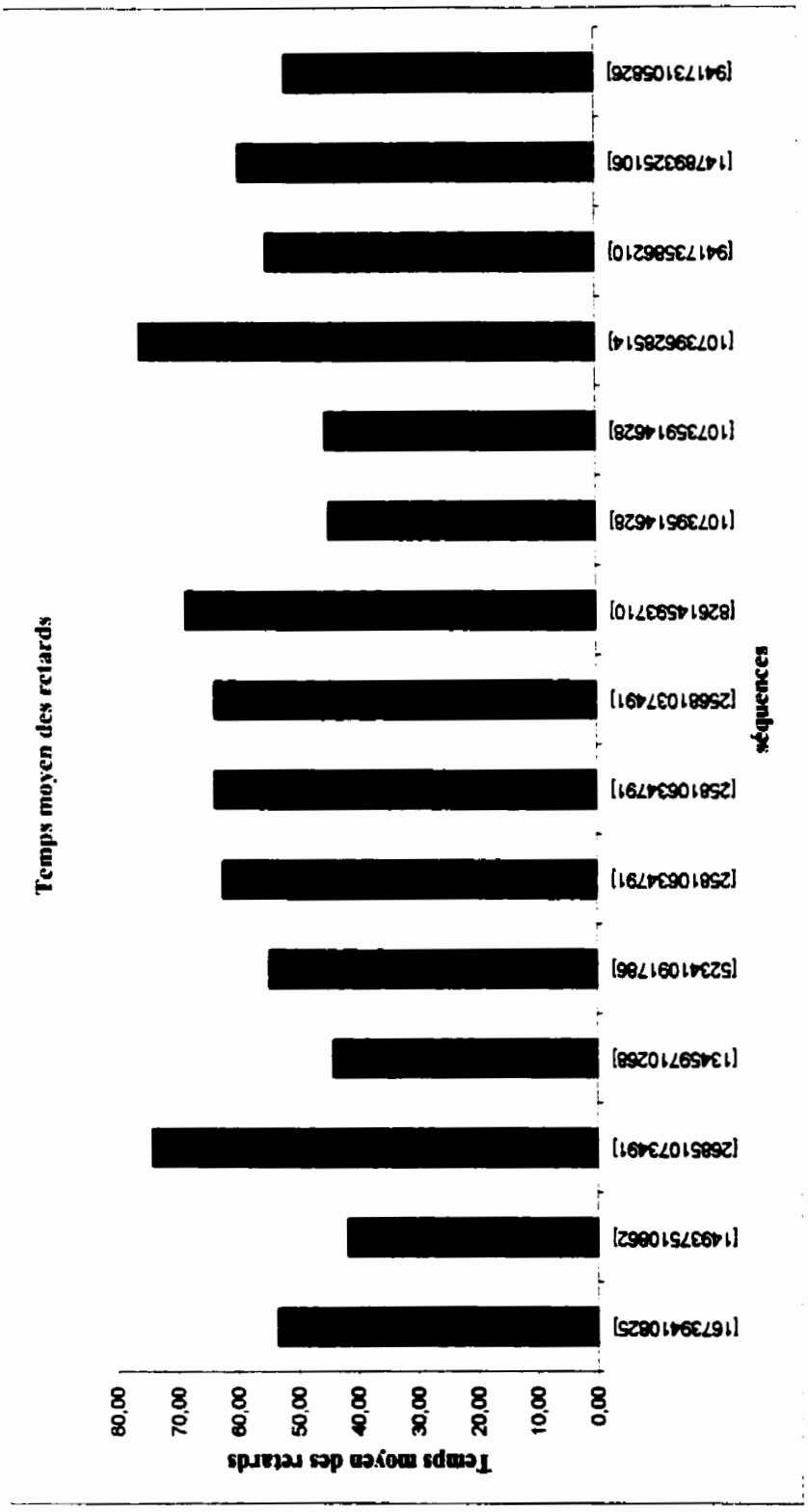


Graphique 2: Makespan vs séquence des tâches

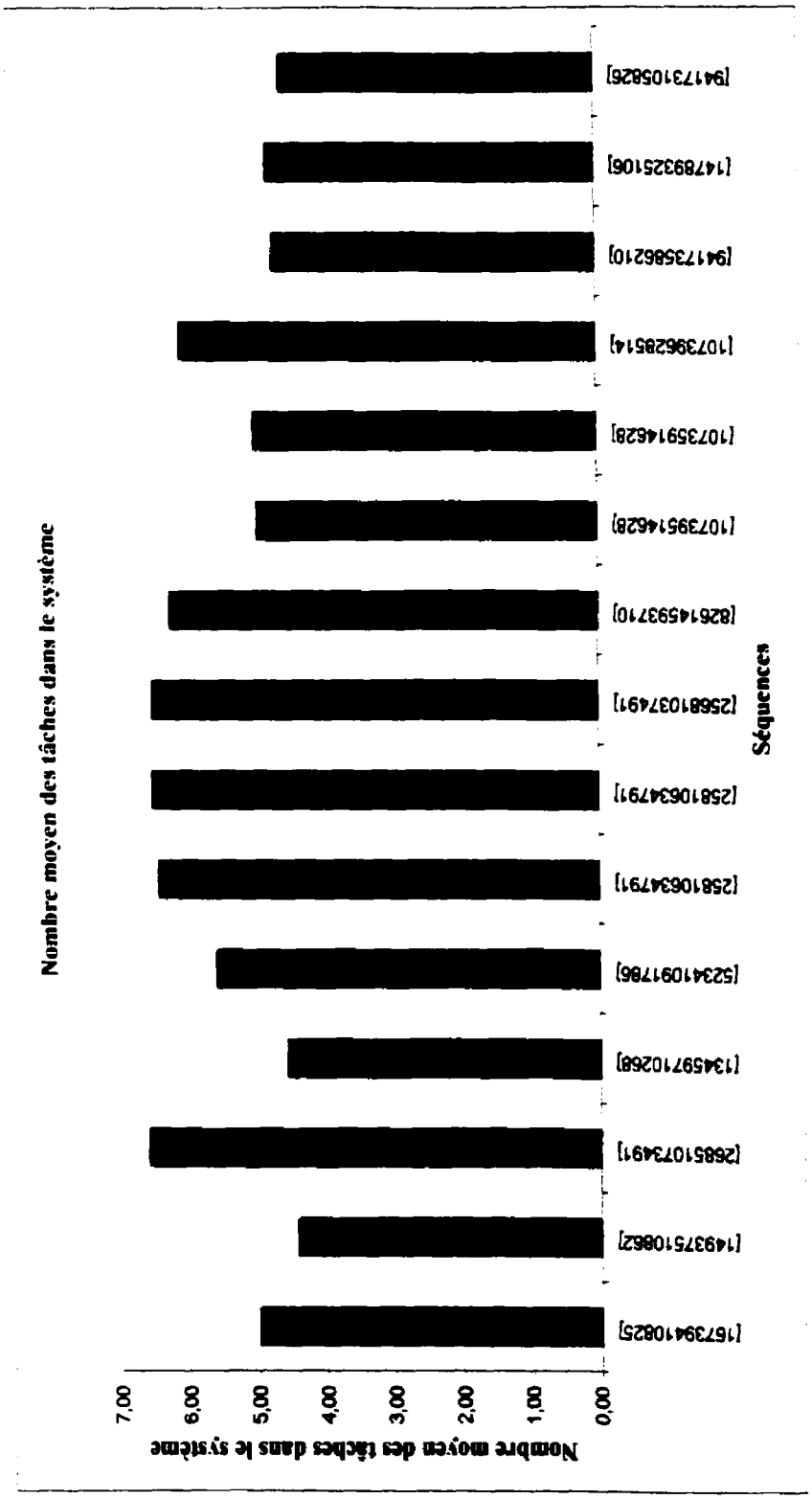
Graphique 3 : Somme des retards vs séquence des tâches



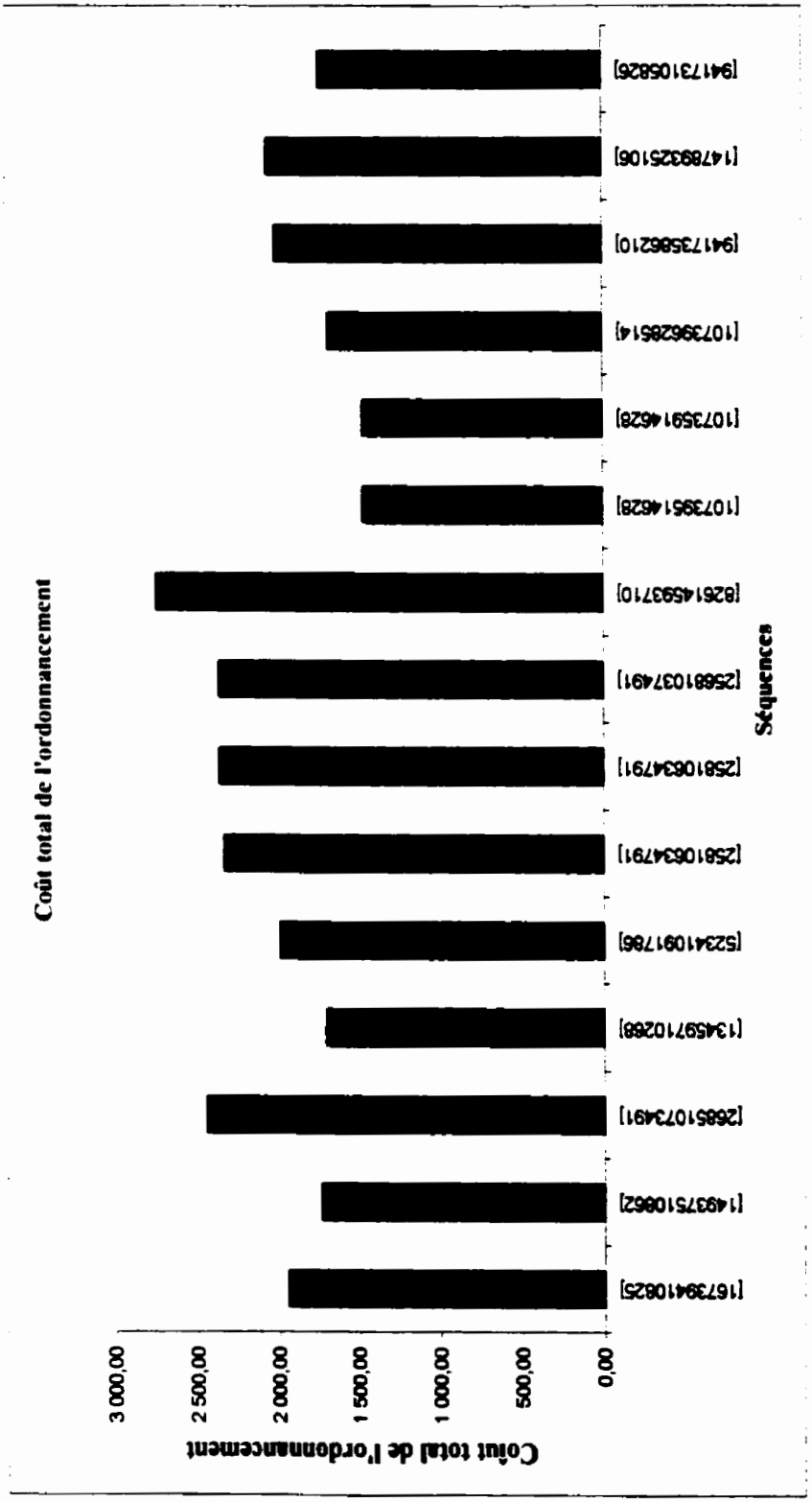
Graphique 4: Somme du temps d'achèvement des tâches vs séquence



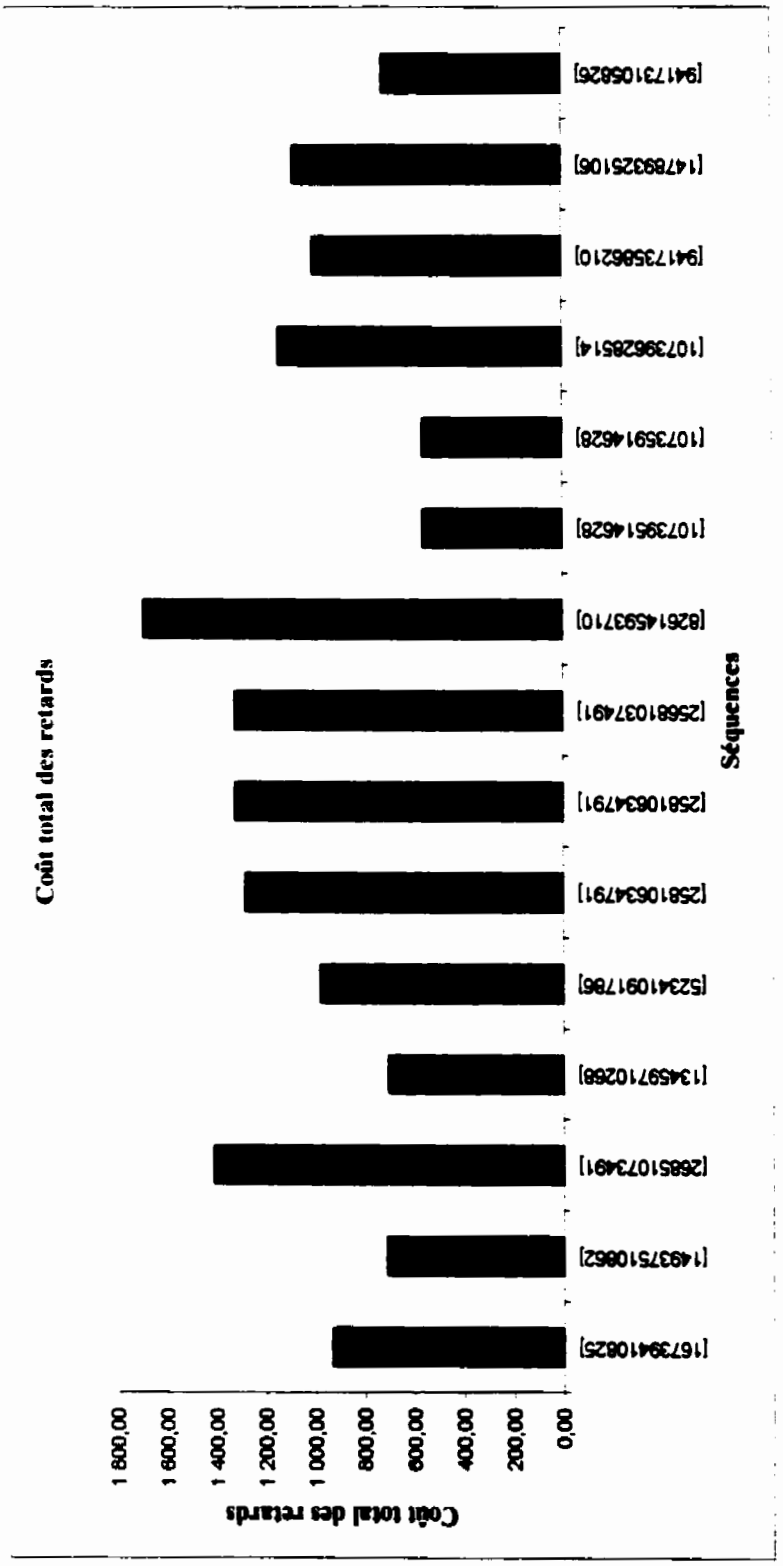
Graphique 5: Temps moyen des tâches vs séquence



Graphique 6: Nombre moyen des tâches dans le système vs séquence



Graphique 7: Coût total de l'ordonnancement vs séquence



Graphique 8: Coût total des retards vs séquence

4.4 Interprétation des résultats générés par le SBC

La décision sur le choix de la séquence de production est déterminée par le résultat du critère économique fixé par le décideur. À partir du tableau 21, page 117 nous pouvons choisir la séquence en fonction de l'objectif d'optimisation que nous souhaitons atteindre. Les graphiques 2, 3, 4, 5, 6, 7 et 8 illustrent les critères de performance en fonction des séquences de tâches fournies par chacun des algorithmes d'ordonnancement bâtis dans le SBC.

- Si nous choisissons comme critère à optimiser le "Makespan", alors les algorithmes 1,2,4 et 14 fournissent les différentes séquences optimales.

- Par contre si notre objectif est de minimiser la somme des retards, comme nous pouvons le constater dans le tableau 21, page 117 et le graphique 2, page 118, la séquence 14937510862 fournie par l'algorithme N°2 est optimale avec la somme des retards de 291,00.

- Un autre exemple , si l'objectif est de minimiser le coût total de l'ordonnancement, alors les séquences optimales générées par notre système sont celles qui sont fournies par les algorithmes N°10 et 11 correspondant aux cédules 10739514628 et 10735914628 avec un coût de 1460,00 respectivement.

- Autre paramètre de performance que nous pouvons fixer comme critère d'aide à la décision dans le choix de la séquence optimale est celui du nombre moyen des tâches dans le système. À ce moment, les séquences 94173586210, 14789325106 et 94173105826 fournies par les algorithmes N°13, 14 et 15 sont choisies, car le résultat en terme de coût des encours de production sera minimal.

CONCLUSION

Par ce travail, nous avons voulu ouvrir la voie à une nouvelle approche dans le domaine de l'ordonnancement des systèmes de production en fournissant au décideur un outil puissant, qui lui permet de résoudre les problèmes d'ordonnancement dans les différents environnements manufacturiers. L'architecture tandem du SBC développé pour l'ordonnancement est construite dans une forme générale. Le système peut être en conséquence utilisé pour les différents types de problèmes d'ordonnancement, explore les différentes solutions possibles et identifie la séquence optimale des taches assignées à un cellule composée d'un groupe de machines.

La nature dynamique du système alloue à l'utilisateur la mise à jour à n'importe quel moment, de tous les paramètres de fabrication et les règles de production de la base de connaissance, permettant au système de répondre à temps réel. Plusieurs critères de performance ont été introduits dans le système. Le SBC proposé, est considéré comme assez flexible pour résoudre un groupe de problèmes d'ordonnancement particuliers avec différents objectifs définis par l'utilisateur.

Le test de l'outil sur l'exemple présenté dans le chapitre 4, permet de tirer les conclusions suivantes:

- L'utilisation du logiciel est d'une simplicité exceptionnelle et ne nécessite aucune connaissance préalable de l'ordonnancement. Par contre la connaissance de système de production est nécessaire pour le choix du modèle de l'environnement à simuler et de l'objectif à atteindre.

- L'outil offre une flexibilité acceptable, car il peut s'adapter à n'importe quel environnement manufacturier.

- Cet approche constitue un départ pour l'utilisation de la base de connaissances dans le domaine de l'ordonnancement et les développements peuvent être poussés pour élargir le types de situations pouvant être résolues.

RECOMMANDATIONS

Il reste en effet beaucoup de choses à améliorer dans notre système de base de connaissances que nous venons de proposer. Nous recommandons d'apporter les améliorations suivantes, à savoir :

- L'élargissement de la base des connaissances pour répondre à toutes les questions qui peuvent être posées devant un problème complexe d'ordonnancement;

- Introduire la génération de diagramme de Gantt pour les séquences générées par l'outil, afin de permettre à l'utilisateur de visualiser la répartition des tâches dans le temps sur les différentes machines composant notre atelier de fabrication.

- Pour faciliter la tâche à l'utilisateur dans le processus de prise de décision quant à la construction de son plan de production, il y a lieu de bâtir un algorithme pour permettre à l'outil de faire le choix automatique de la séquence (plan de production) optimale: c'est ce que nous appelons : "Système Expert".

BIBLIOGRAPHIE

1. Richard B. Chase et Nicholas J. Aquilano, (1992), "Production & Operations management". Irwin, sixth edition, pp, 753-786.

2. French, S., (1982), "Sequencing and scheduling: Introduction to the mathematics of the job shop", John Wiley & Sons.

3. Thien-My D., Michel G., et Gilbert H., (1991), "A knowledge based scheduling system for productivity optimization of the group technology manufacturing", Design Productivity International Conference' Honolulu, Hawai, pp. 299-304.

4. Jeong K.C. & Kim Y.D.,(1997) " A real time scheduling mechanism for flexible manufacturing system: Using simulation and dispatching rules", International Journal of Production Research, vol. pp2611-2625.

5. Mc Pherson & Preston White K., Jr., (1998), "Periodic flow line scheduling", International Journal of Production Research, vol. 36, pp. 51-73.

6. William J. Stevenson, (1996), "Production & Operation Management", Irwin, fifth edition, pp. 710-746.

7. Michael Pinedo, (1995), " Scheduling: Theory, Algorithms and systems", Englewood Cliffs, N.J. Prentice-Hall.

8. M K Khan, C S Wright, and R Whalley, (1996), "Advanced Manufacturing Process, Systems, and Technologies (AMPST 96)" Mechanical Engineering Publications Limited, London and Bury St Edmunds,UK, pp. 201-210.

9. Alain Bonnet, Jean-.Paul Haton, Jean-Michel Truong-Ngoc, (1986), "Systèmes experts : Vers la maîtrise technique", Intereditions.

10. Alain Bonnet, (1984), " L'intelligence artificielle: Promesses et Réalités", Intereditions.

11. Mitchell S. Steffen, (1986), "A Survey of artificial Intelligence-Based Scheduling Systems", Fall of 1986 industrial Engineering Conference proceeding", pp. 395-405.

12. Clive L.Dym, (1985), "Expert systems: New approches to computer-aided Engineering", Engineering with computers1, pp. 9-25.

13. Bensana E., Correge,M. Bel G. And Dubois D., (1986), "An Expert System Approch in Industrial Job-Shop Scheduling", Proceeding of 1986 IEEE International Conference on Robotics and Automation, San Francisco, April 7-10, 1986, pp. 1645-1650.

14. Panwalker, SS. & Iskander, W. (1977), "A survey of scheduling rules", Operations Research, Vol. 25 (1), pp. 45.

15. Blackstone, J.H.Jr., Phillips, D.T. & Hogg, G.L., (1982), "A state-of-the-Art Survey of Dispatching Rules for Manufacturing Job Shop Operations", International Journal of production Research, Vol. 20 (1), pp. 27-45.

16. Gupta, S.K. & Kyparisis, J., (1987), " Single Machine Scheduling Research", OMEGA, International Journal of Management Science, Vol. 15 (3), pp. 207-227.

17. Ramasesh, R. (1990),"Dynamic Job Shop Scheduling: A survey of Simulation Research", OMEGA, International Journal of Management Science, Vol. 18 (1), pp.43-57.

18. MacCarthy, B.L. & Liu, J., (1993), "Addressing the Gap in Scheduling Research: a Review of Optimization and Heuristic Methods in Production Scheduling, International Journal of Production Research, Vol. 31, pp. 331-340.

19. Muhlemann, A.P., Oakland' J.S., & Lockyer, K.G., (1992), Production and Operations Management, (Pitman), 6th Ed. London.