Titre: Title:	Gestion d'un atelier de type "Flow Shop généralisé"
Auteur: Author:	Corinne Ohayon
Date:	1989
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Ohayon, C. (1989). Gestion d'un atelier de type "Flow Shop généralisé" [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/58266/

UNIVERSITÉ DE MONTRÉAL

GESTION D'UN ATELIER DE TYPE "FLOW SHOP GÉNÉRALISÉ"

par

Corinne OHAYON

DÉPARTEMENT DE MATHÉMATIQUES APPLIQUÉES

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION

DU GRADE DE MAITRE ES SCIENCE APPLIQUEES (M.Sc.A.)

(MATHÉMATIQUES APPLIQUÉES)

Juillet 1989

ISBN 0-315-52705-6

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE

Ce mémoire intitulé:

GESTION D'UN ATELIER DE TYPE "FLOW SHOP GÉNÉRALISÉ"

présenté par: Corinne Ohayon

en vue de l'obtention du grade de: MAITRE ES SCIENCES APPLIQUEES (M.Sc.A)

a été dûment accepté par le jury d'examen constitué de:

M. Benjamin T. Smith ..... , Ph.D., président

M. François Soumis ..... Ph.D.

M. Martin Desrochers ..... Ph.D.

M. Chelliah Srikandarajah ..... Ph.D.

Sommaire

Ce mémoire a pour but de mettre au point une méthode permettant de produire un horaire réalisable pour une chaîne d'assemblage fonctionnant sous forme de "FLOW SHOP GÉNÉRALISÉ", i.e., nous avons des tâches à effectuer sur une série de machines dans un ordre pré-établi. Par contre, les tâches n'ont pas à être traitées par toutes les machines. L'étude sera réalisée sur un atelier fabriquant des produits de réseaux de télécommunication.

En tenant compte des deux problèmes majeurs dans notre atelier, la congestion de certaines machines et la sous-utilisation d'autres, nous avons choisi une fonction objectif. Elle consiste en la minimisation de la somme pondérée des temps d'attente des tâches. Les poids utilisés correspondent aux priorités de production associées à chacune des tâches.

Ensuite, nous modélisons l'atelier pour résoudre par une méthode heuristique basée sur la programmation dynamique. Une exploration partielle de l'espace des états nous permet d'obtenir de "bonnes" solutions pour les horaires de production.

Abstract

The subject of this thesis is to develop a method which will produce a feasible schedule for an assembly line of a "GENERALIZED FLOW SHOP" type, i.e., we have jobs to process on a set of machines with precedence constraints. These jobs are not necessarily processed by every machine in the set. This particular study is based on an assembly line where products for telecommunication networks are manufactured.

Two major problems have been considered in the definition of the objective function: the back-up of jobs at some machines and the under-utilization of other machines. The objective is to minimize the sum of the weighted waiting times of each job. The weights used in the summation correspond to production priorities assigned to each job.

We next model the shop to solve by a heuristic method based on dynamic programming. We have been able to obtain feasible production schedules by partially searching the state space.

Remerciements

Je voudrais exprimer ma profonde reconnaissance à mon directeur de recherche, François Soumis, ainsi qu'à mon co-directeur, Martin Desrochers pour leurs judicieux conseils et leur soutien moral qui m'ont permis de compléter ce mémoire.

Je remercie également "Northern Telecom" de m'avoir fourni les données nécessaires pour valider mon sujet.

Pour le support financier, je remercie François Soumis et Martin Desrochers.

Aussi, pour les corrections finales du texte ainsi que pour l'encouragement qu'elle m'a témoigné, je remercie Anne-Marie Goulet.

Finalement, je tiens à remercier mes parents et mon fiancé Jordan pour leur patience et leur encouragement tout au long de mes études.

Table des Matières

Figures

Tableaux

Chapitre 1

Introduction

Un des plus importants problèmes actuels de l'industrie manufacturière est celui de l'ordonnancement de la production dans un atelier. Quoique la recherche dans ce domaine soit intensive, il existe tellement de cas particuliers qu'elle n'est jamais épuisée. Dans ce mémoire, nous étudions un de ces cas particuliers.

Des composantes de réseaux de télécommunication sont fabriquées dans notre atelier. Chaque produit est composé de différentes cartes électroniques. Selon les désirs du client, un produit peut être modifié en ajustant la combinaison régulière des cartes électroniques du produit. Notre atelier comprend trois lignes d'assemblage et on peut fabriquer deux ou trois produits différents sur chaque ligne. Une chaîne d'assemblage est formée de postes de travail ou machines et chaque carte électronique a un cheminement pré-établi sur cette chaîne.

Plus précisément, l'ordonnancement de la production dans un atelier consiste en l'allocation des tâches aux ressources en réalisant un horaire assurant leur exécution. Ce mémoire comprend la fabrication d'un tel horaire pour un type d'atelier particulier. Un problème d'ordonnancement est bien défini quand quatre facteurs sont identifiés: les ressources, les tâches, les contraintes, et les objectifs. Les ressources correspondent aux stations de travail qui sont chacune formées d'une machine et/ou un opérateur. Une tâche comprend un nombre fixe d'éléments d'un produit que l'on désigne sous le nom de *lot*. Une tâche comporte une série d'opérations à effectuer sur différentes stations de travail. Ces opérations sont effectuées en ordre prédéterminé et cet ordre est le même pour toutes les tâches traitant le même élément. Un temps d'exécution est associé à chaque opération. Chacune de ces opérations doit être terminée avant une date limite (la date au plus tard) et elle ne peut être exécutée avant le temps de disponibilité des matériaux.

Il faut d'abord classer le type d'atelier qui nous concerne. Un atelier comprend m machines où il faut ordonnancer n taches chacune composée d'un nombre fixe d'opérations. Un problème d'atelier peut se trouver dans trois différentes catégories: un "Open Shop", un "Job Shop", ou un "Flow Shop". Le premier type considère un atelier où les opérations d'une tâche sont indépendantes, donc elles peuvent être exécutées dans n'importe quel ordre sur les stations de travail. Dans un "Job Shop" l'ordre des opérations est défini par des contraintes de précédence et l'ordre peut être différent pour chaque tâche. Les tâches dans un "Flow Shop" sont définies par des ensembles ordonnés d'opérations devant passer par toutes les stations. Donc, s'il existe m stations dans l'atelier, chaque tâche comprend m opérations chacune à exécuter sur une machine différente, et l'ordre des opérations pour chaque tâche est le même.

Le problème que nous considérons dans ce mémoire est classé dans la troisième catégorie mais un autre facteur entre en cause. Dans notre cas, les opérations d'une tâche ne doivent pas obligatoirement être traitées par toutes les stations. Nous appelons ce type de problème un "FLOW SHOP GÉNÉRALISÉ". Les propriétés qui distinguent notre atelier d'un "FLOW SHOP" simple sont les suivantes :

- une tâche peut comprendre moins de m opérations
- les opérations n'ont pas à utiliser toutes les stations dans la numérotation de 1 à m
- les opérations initiales et finales ne sont pas obligatoirement aux stations 1 et m, respectivement.

La définition d'un problème de ce type n'est pas complète sans l'étude de ses

contraintes et objectifs. Nous allons donc les définir pour pouvoir les incorporer dans la fonction économique utilisée dans notre modèle mathématique. En analysant les problèmes actuels et les contraintes imposées dans notre atelier, nous avons vu qu'il existe quatre facteurs à considérer en formant notre objectif :

- éviter la congestion aux stations de travail
- éviter des temps morts aux stations de travail
- respecter les dates au plus tard des opérations
- respecter les dates de disponibilité des matériaux

Au cours des dernières années, plusieurs études sur les différents objectifs ont été faites. Parmi elles, Mellor (1966) a présenté 27 objectifs qui pourraient être considérés. Dans ce mémoire, nous en verrons quelques-uns et ensuite nous présenterons celui qui est utilisé dans notre modèle.

En utilisant ces objectifs, nous pouvons appliquer une méthode de résolution. Nous avons décidé d'utiliser une méthode d'énumérations implicite. La programmation dynamique nous permet d'explorer toutes les solutions valables et, à partir des critères imposés par l'objectif, nous pouvons déterminer quel est le "bon" horaire. Nous devons premièremnt énoncer les différentes mesures de performance qui peuvent être appliquées au problème en tant qu'objectifs et ensuite modéliser notre "FLOW SHOP GÉNÉRALISÉ" en un problème de programmation dynamique.

L'organisation du mémoire se présente de la façon suivante. Le chapitre 2 décrit notre problème en détail et énumère les objectifs qui peuvent être appliqués à notre

problème. Le modèle formulé pour la programmation dynamique et son objectif sont exposés au chapitre 3. Les chapitres 4, 5, et 6 énoncent l'algorithme détaillé de résolution et son analyse est traitée au chapitre 7.

Chapitre 2

Description du problème et des objectifs

Le problème posé consiste en la fabrication d'un horaire de production pour un atelier fonctionnant sous forme d'un "FLOW SHOP GÉNÉRALISÉ". Cet atelier comprend trois lignes d'assemblage qui fabriquent des circuits électroniques pour des réseaux de télécommunication. Les opérations dans ces lignes comprennent, entre autres, la préparation de plaques pour cartes électroniques, l'insertion de pièces différentes pour commandes spéciales, l'inspection visuelle et automatique, ainsi que l'assemblage manuel du produit fini. Selon les produits commandés, il y a un certain nombre d'opérations à effectuer sur une série de machines dans un ordre pré-établi et unidirectionnel; et ce, sans devoir utiliser toutes les stations de travail.

Le modèle ainsi que l'algorithme que nous allons développer va s'occuper d'ordonnancer les produits sur une seule chaîne d'assemblage à la fois. Donc, nous traitons le problème comme s'il y avait une seule chaîne de production et les deux autres peuvent être considérées séparément.

Notre ligne d'assemblage fonctionne vingt-quatre heures sur vingt-quatre pendant cinq jours par semaine. On ne peut pas considérer un horaire de production sur les cinq jours car le problème devient trop grand. Puisqu'il y a un changement d'opérateurs aux machines toutes les huit heures, nous avons décidé de produire l'horaire pour une tranche de huit heures à la fois. En tenant compte de ces facteurs nous pouvons procéder. Donc, ce chapitre est consacré à la description des facteurs composant notre modèle.

2.1 Les machines

Les machines sont en réalité des stations de travail. Une station de travail est un poste où s'effectue une des opérations déjà mentionnées ou de type plus détaillé, i.e., préparation d'une plaque pour le soudage. Une opération peut comprendre soit l'utilisation d'une machine (station automatique), celle d'une machine et d'un opérateur (station semi-automatique), ou seulement d'un opérateur (station manuelle).

Nous avons donc un certain nombre de stations distinctes en série. Chacune de ces stations peut effectuer un ensemble d'opérations. L'opération à exécuter est déterminée selon le produit à fabriquer. Des stations en série n'effectuent jamais la même opération.

Il existe également des stations en parallèle à certains postes. À ces postes, on peut trouver plus d'une machine (et/ou opérateur) du même type. Ces machines en parallèle peuvent appartenir à plus d'une ligne de production. L'opération à effectuer est affectée à la première machine libre de la ligne actuelle. Si aucune machine n'est disponible ou fonctionnelle sur cette ligne et s'il s'agit d'une opération urgente, cette dernière peut être affectée à une machine du même type sur une autre ligne d'assemblage. Cela permet donc à la machine inopérante d'être remplacée par une autre disponible, à priori dans la ligne en question, ou dans le cas contraire et lorsqu'il s'agit d'une urgence, solliciter celle d'une autre ligne. Selon la disponibilité des machines (et/ou opérateurs) à une station, une opération devant passer par ce poste, choisirait parmi plusieurs cheminements possibles. Ce choix est déterminé en fonction des objectifs que nous définirons plus loin.

2.2 Les produits

Une tâche est associée à un produit et est définie par un ensemble ordonné d'opérations. Plusieurs produits peuvent être assemblés dans l'atelier. Il existe une base de données contenant les informations pertinentes sur ces produits.

Premièrement, nous définissons avec précision les tâches. Un *produit* est caractérisé par la taille de ses lots et chaque lot est considéré comme étant une tâche à effectuer dans l'atelier. Un *lot* ou une *tâche* est un groupe d'éléments d'un produit qui doit passer par des stations de travail (machines) dans un ordre prévu. A chaque station, différentes *opérations* sont effectuées étape par étape afin de compléter le produit. Un lot n'avancera vers la prochaine station qu'après que tous ses éléments aient été traités.

2.2.1 Temps d'exécution

Pour déterminer un horaire de production, il est nécessaire de connaître le temps d'exécution de toutes les opérations possibles. À chacun des produits est associée une série d'opérations sur des stations de travail dans un ordre pré-établi. Pour chaque opération correspondant à sa station, il existe un temps requis pour l'effectuer. Ce temps inclut le temps de mise au point de la station (si nécessaire) et le temps actuel d'exécution de l'opération.

Le temps de mise au point est le temps requis pour préparer une station de travail pour l'exécution d'une opération. Etant donné que ce temps de mise au point est indépendant de la séquence des opérations et qu'il est de durée négligeable

par rapport au temps d'exécution, il est inclus dans le temps total pour effectuer l'opération. Ce temps sera connu comme le temps d'exécution d'une opération (d'une tâche) à une station dans l'atelier.

Il existe aussi un temps de délai possible après que l'opération soit complétée à la station de travail. Ce temps de délai tient compte par exemple du temps pour laisser sécher la soudure d'une carte électronique. Or, même si l'opération est terminée et la tâche ne peut pas passer à la prochaine station avant la fin du temps de délai, la station précédemment occupée est libre.

Le temps d'exécution est différent entre les stations en parallèle car toutes les machines (et/ou opérateurs) du même type ne fonctionnent pas nécessairement avec le même taux de production. Donc, pour un type de station, il peut y avoir plusieurs temps d'exécution associés à l'opération en question.

2.2.2 Dates au plus tard et dates de disponibilité

À part le temps d'exécution, il existe d'autres informations importantes associées à un produit. Les délais de production pour un ou plusieurs lots d'un produit doivent être respectés. Ceci est déterminé par la demande des clients pour ce produit. Le nombre de lots (ou de tâches) associé à un produit qui doit être traité est prévu à partir de la demande des vingt prochains jours de production ainsi que la disponibilité des matériaux requis pour la fabrication de ce produit.

Nous avons donc deux contraintes à considérer pour permettre la production d'un lot. Premièrement, nous devons respecter les dates au plus tard ("due date")

pour chaque produit. C'est-à-dire, il existe des dates de livraison à la clientèle et nous devons produire ce lot avant sa date au plus tard. La deuxième contrainte, qui est plus forte que la première, est le fait qu'il faut avoir assez de matériel pour pouvoir mettre ce produit en production. Nous devons respecter la date de disponibilité des matériaux pour le produit ("release date") pour nous permettre de produire un ou plusieurs lots du produit.

2.2.3 Les priorités

Ces informations ne sont pas explicitement dans notre base de données. A partir des dates au plus tard et celles de disponibilité des matériaux pour un produit, l'usager du logiciel affectera une priorité de production au produit ainsi que la quantité de produit à laquelle on appliquera cette priorité. Puisque la production dans l'atelier est prévue sur un horizon de vingt jours de travail, les priorités sont des entiers entre 1 et 20. Plus la priorité est élevée, le plus tôt le produit devrait être fabriqué. Donc, chaque produit est caractérisé par une demande et une priorité entre 1 et 20.

2.3 Les objectifs

Puisque nous avons un problème qui dépend du temps d'exécution et de dates, nous devons trouver un horaire de production de façon rapide et efficace en tenant compte de certains objectifs. Parce que notre atelier fonctionne continuellement (vingt-quatre heures sur vingt-quatre) et que les opérateurs changent aux huit

heures, nous devons prévoir l'horaire pour une tranche de huit heures.

L'ordonnancement pour une tranche de huit heures doit tenir compte de l'état du système à la fin de la dernière tranche, i.e., de ce qu'il reste à produire dans l'atelier, et des nouveaux produits introduits au cours de cette dernière tranche de temps. Puisque le problème est continuel, nous avons constamment des informations sur chacun de ces facteurs sur un horizon de 20 jours du mois actuel. Il y a toujours des modifications au cours des journées, donc l'horaire est déterminé selon les informations présentes au début d'une tranche de huit heures.

L'usager a aussi la possibilité de déterminer l'horaire sur des tranches de temps inférieures à huit heures. S'il y a eu trop de modifications au cours d'une tranche de huit heures et qu'il existe de nouvelles tâches à effectuer d'urgence, il est possible de reprendre l'horaire au point où les tâches sont urgentes. Le choix de la durée du nouvel horaire est optionnel.

Pour trouver un horaire raisonnable pour à notre atelier, il faudrait déterminer nos objectifs. Dans notre atelier, il existe deux problèmes majeurs à considérer dans l'objectif en réalisant l'horaire de production. Le premier est celui de la congestion des stations de travail. Ce problème est présent dans l'atelier parce que le système actuel utilisé pour créer l'horaire ne minimise pas le temps d'attente des opérations aux stations. Cette lacune cause aussi le deuxième problème de la sous-utilisation des machines. Il y a souvent des temps morts à certaines stations en attendant que d'autres stations effectuent les opérations qui sont en attente.

Or, en définissant notre objectif, nous devons essayer de bien "balancer" la chaîne d'assemblage afin de diminuer l'engorgement à certaines machines et d'aug-

menter la productivité à d'autres. Pour atteindre ce but, il existe plusieurs mesures de performance que nous appliquerons à notre problème et ensuite nous en analyserons les résultats.

Pour évaluer une solution (ou un horaire), il faut pouvoir en mesurer certains attributs et les comparer aux autres solutions. Ceci nous permettrait de déterminer la meilleure solution. Un certain nombre d'objectifs ont déjà été explorés. Les prochaines sections serviront à décrire ces objectifs et à explorer leur usage, pour ensuite définir l'objectif s'appliquant à notre problème.

2.3.1 Description des données et des résultats possibles

Un problème d'ordonnancement de la production exige un minimum d'informations en entrée ainsi qu'un minimum de résultats à la sortie d'une solution. Les trois éléments de base requis en entrée sont les suivants :

- temps d'exécution ($p_{\ell j}$): temps requis pour effectuer l'opération ℓ de la tâche j
- date de disponibilité ($r_{\ell j}$): date ou temps de disponibilité du matériel pour effectuer l'opération ℓ de la tâche j
- date au plus tard ($d_{\ell j}$): date ou temps quand l'opération ℓ de la tâche j doit être terminée

Ces paramètres d'entrée contribuent à la description des tâches et sont utilisées pour déterminer un horaire de production.

Les résultats générés à partir d'une solution servent comme paramètres dans les différents objectifs que nous pouvons imposer à notre problème. Parmi ces résultats possibles, nous avons choisi les plus appropriés à notre problème :

- C_j: temps d'achèvement de la tâche j

  i.e., le T_j final pour cette tâche où T_j est le temps de fin de la dernière opération de la tâche j.

- L_j: retard de la tâche j par rapport à sa date au plus tard

$$L_j = \max(0, C_j - d_{n_j,j})$$

  où n_j est le nombre d'opérations de la tâche j.

- F_j: temps passé dans l'atelier par la tâche j

$$F_j = C_j - r_{1j}$$

La première quantité, C_j, est la donnée la plus pertinente pour l'évaluation d'une solution car elle nous permet aussi de générer d'autres résultats. Le temps de retard de la tâche j, L_j, évalue la possibilité de l'horaire de respecter la date au plus tard de la tâche. Si la tâche se termine tôt, $C_j - d_{n_j,j}$ est une valeur négative. Dans ce cas, il n'y a aucun bénéfice, donc une valeur de zéro est affectée à L_j. Quand la valeur de $C_j - d_{n_j,j}$ est positive, la tâche est en retard. On peut lui imposer une pénalité pour essayer d'éviter trop de retard. F_j, le temps passé par la tâche j dans l'atelier, est une quantité qui représente l'intervalle de temps (entre son arrivée dans l'atelier et son départ) qu'une tâche attend avant d'être achevée.

2.3.2 Critères d'évaluation

Une solution est évaluée par une quantité qui incorpore de l'information au sujet de toutes les tâches. Cette quantité est généralement une fonction des C_j et on la désigne sous le nom de *critère d'évaluation.* En minimisant ou maximisant cette mesure, nous obtenons une fonction objectif. Les sections suivantes sont destinées à l'énumération des critères pouvant être utilisés pour notre problème.

Durée totale de l'horaire ("makespan")

Un des objectifs les mieux connus et le plus utilisé pour ce type de problème est celui de minimiser la durée totale de l'horaire. Cet objectif implique l'utilisation des C_j directement; c'est la minimisation du temps de fin ou d'achèvement de la tâche la plus tardive:

$$\text{minimiser} \quad \max\{C_j\}, \quad 1 \leq j \leq n.$$

Cet objectif est convenable pour un problème qui n'est pas cyclique dans le temps. Car pour optimiser l'horaire avec cet objectif, il faut appliquer la programmation dynamique jusqu'à ce que toutes les tâches soient terminées. Dans le cas où le problème n'est pas continuel, cet objectif résulte en l'utilisation efficace des ressources.

Notre modèle, étant un problème cyclique, ne pourrait pas utiliser cet objectif efficacement. Pour chaque tranche de temps à laquelle nous appliquerons ce critère, l'objectif s'occuperait plutôt de la fin des dernières tâches au lieu de la tranche entière. Puisque le problème est continuel et qu'il y a constamment des nouvelles

tâches ajoutées au système, nous ne pouvons pas utiliser cet objectif à sa juste valeur.

Respect des dates au plus tard ("due date")

Souvent un horaire est évalué par sa divergence des dates au plus tard (voir Holloway and Nelson (1973)). Ceci peut être mesuré par plusieurs critères d'évaluation entre autres:

- minimiser le plus grand retard:

$$\min \quad \max L_j, 1 \leq j \leq n,$$

- minimiser la somme des retards:

$$\min \sum_j L_j$$

- minimiser le retard moyen:

$$1/n \sum_j L_j$$

- minimiser le nombre de tâches en retard:

$$\min \sum_j U_j$$

où

$$U_j = \begin{cases} 1 & \text{si tâche } j \text{ est en retard} \\ 0 & \text{sinon} \end{cases}$$

Les quatre objectifs çi-dessus seraient convenables dans une situation où la base de données contiendrait les dates au plus tard. Les trois premiers critères pourraient s'appliquer à un problème où la pénalité est imposée sur l'amplitude des retards tandis que le dernier serait approprié à un problème où le nombre de retards est une mesure importante. Ce dernier pénalise le fait d'avoir un retard (quelque soit l'amplitude du retard) car il existe des situations où un retard n'est pas du tout acceptable. Notre problème se rapproche plutôt des trois premières mesures. Un retard devrait être pénalisé mais pas interdit.

Temps passé dans l'atelier ("flow time")

Un autre critère d'évaluation connu implique l'utilisation des $r_{\ell j}$, le temps de disponibilité de l'opération ℓ appartenant à la tâche j. Les objectifs suivants incorporent le paramètre F_j (le temps passé dans l'atelier par la tâche j):

- minimiser le maximum des temps passés dans l'atelier:

$$\min \max F_j, 1 \leq j \leq n$$

- minimiser la somme des temps passés dans l'atelier:

$$\min \sum_j F_j$$

- minimiser la moyenne des temps passés dans l'atelier:

$$\min 1/n \sum_j F_j$$

De façon similaire aux objectifs ayant L_j comme paramètres, les critères de cette section nécessitent les $r_{\ell j}$ explicitement dans la base de données. Chacun de ces derniers est affecté par différents facteurs. Le coût de l'horaire du premier, $\min \max F_j$, est directement affecté par la tâche de plus longue durée. Par contre, le deuxième critère dépend de la durée totale de toutes les tâches. Finalement, le coût du troisième est affecté par le temps moyen pour effectuer une tâche. Etant donné que la base de données n'inclut pas les $r_{\ell j}$, ces critères ne peuvent pas être utilisés pour notre problème.

2.4 Adaptation des critères d'évaluation à notre problème

Établir une fonction objectif pour un problème nécessite d'abord l'analyse de l'atelier et de ses besoins. Tel que présenté dans la description de notre système, il existe deux problèmes importants à résoudre. L'un est celui de la congestion des machines et l'autre est celui de la sous-utilisation des machines. Nous avons également les contraintes de dates au plus tard ($d_{\ell j}$) et de dates de disponibilité ($r_{\ell j}$) à respecter.

Nous savons déjà que les $d_{\ell j}$ et $r_{\ell j}$ ne se trouvent pas dans la base de données. Cependant, l'usager du système, à partir d'autres sources d'informations, va allouer une priorité à chaque tâche. Cette priorité sera basée sur les $d_{\ell j}$ et $r_{\ell j}$ provenant de l'autre source. Par exemple, supposons que la tache j ne sera disponible qu'à partir du jour 3 du mois et qu'elle doit être terminée avant le jour 8 du mois. Puisqu'il y a 20 jours de travail dans un mois, la priorité de cette tâche pourrait

être entre 13 et 18. Donc, plus c'est urgent, plus la priorité se rapproche de 20. Cette priorité sera utilisée comme poids ou pénalité dans notre fonction objectif.

Par conséquent, nous devons trouver comment appliquer cette pénalité. En analysant nos problèmes dans l'atelier, la conséquence évidente de la congestion des machines ainsi que celle de leur sous-utilisation, est le temps d'attente que les opérations doivent subir. Une opération pourrait attendre indéfiniment à une station de travail si elle n'est pas forcée à passer par le système. Si on calcule le temps d'attente de toutes les tâches, il est possible de définir une pénalité. En effet, si on prend la somme pondérée par les priorités des temps d'attente de toutes les tâches, nous obtenons un objectif convenant à notre problème:

$$\min \sum_j pr_j \times t_j$$

où pr_j, la priorité affectée à la tâche j

t_j, le temps d'attente total de la tâche j

Si les priorités sont assez élevées, nous traiterons les opérations appartenant aux tâches les plus urgentes. Si c'est le temps d'attente qui est assez élevé, les opérations appartenant à ces tâches en attente seraient traitées. Par contre, s'il n'y a pas de tâches urgentes à effectuer, en affectant des grandes priorités aux opérations qui sont en attente aux machines engorg'ees, cette fonction objectif nous permettrait de traiter les opérations en suspens.

Chapitre 3

Modélisation du "FLOW SHOP GÉNÉRALISÉ" par la programmation dynamique

Le présent chapitre a pour but de décrire les notations qui seront utilisées dans les chapitres suivants, de poser le modèle employé pour la résolution du problème et d'établir l'objectif.

3.1 Définition des termes pour la programmation dynamique

Avant de présenter le modèle, nous commençons par définir les termes qui seront utilisés dans le contexte de la programmation dynamique.

La programmation dynamique est une méthode d'énumération présentée par Bellman (1957) qui sert à explorer implicitement toutes les solutions possibles d'un problème et, à partir d'un critère ou d'un objectif, permet de déterminer la meilleure solution. Pour modéliser un problème dans le but de le résoudre par programmation dynamique, il faut le répartir en plusieurs niveaux. A chaque niveau, il existe deux facteurs qui le définissent:

- l'espace des états
- la transition.

Donc, à chaque niveau, il existe plusieurs états. À partir d'un état donné, on a des transitions vers d'autres états (appartenant au niveau subséquent).

Nous procédons maintenant à l'application des principes de la programmation dynamique pour définir notre modèle. Pour simplifier la présentation du modèle,

nous considérons qu'il n'y a pas de machines en parallèle et nous omettons également le concept des priorités. Nous introduirons ces deux facteurs par la suite.

3.2 Notations pour la formulation du modèle

Les notations qui suivent seront utilisées pour présenter le modèle:

- n, nombre de tâches à ordonnancer,
- m, nombre de stations de travail en série,
- j, indice d'une tâche,
- ℓ, indice d'une opération,
- i, indice d'une station de travail (machine),
- t, indice de l'étape ou niveau de la programmation dynamique.

En tenant compte de ces notations, nous passons à la formulation du modèle.

3.3 Le Modèle

3.3.1 Description

Il y a n tâches j chacune comprenant n_j opérations, devant être traitées sur m machines distinctes. La $\ell^{\text{ième}}$ opération de la tâche j ($o_{\ell j}$) doit être effectuée sur la machine $m_{\ell j}$. En conformité avec la définition d'un "FLOW SHOP GÉNÉRALISÉ", une tâche ne passe pas nécessairement par toutes les machines mais son passage reste toujours ordonné par rapport à leur numérotation. $p_{\ell j}$ est le temps

d'exécution de l'opération o_{ij}. Il est possible d'avoir un temps de délai minimum entre deux opérations de la même tâche. Le délai existant après l'opération o_{ij} est dénoté par dy_{ij}.

3.3.2 Formulation pour la programmation dynamique

Maintenant que toutes nos notations du problème ainsi que tous nos termes pour la programmation dynamique sont établis, nous pouvons procéder à l'exposé du modèle. Donc, nous décrivons précisément les éléments d'un niveau où le $t^{ième}$ niveau est l'ensemble des états où il reste t opérations à ordonnancer. Donc, il y aura autant de niveaux qu'il y a d'opérations.

État au niveau t

La définition d'un état au niveau t est la suivante:

- S_t l'ensemble des opérations qu'il reste à ordonnancer
  où $t = \sum_{j=1}^{n} n_j - |S_t|$ ($|S_t|$ représente la cardinalité de l'ensemble de S_t)
- P_i est le temps de fin de la dernière opération déjà
  ordonnancée sur la machine i ou m_{ij}.
- T_j est le temps de fin de la dernière opération déjà
  ordonnancée de la tâche j.

Ainsi, un état au niveau t se présente sous la forme (S_t, P, T), où P et T sont des vecteurs de dimensions m et n, respectivement.

Transition

Une transition qui permettra à l'état en question de générer un état du prochain niveau consiste à choisir une prochaine opération $o_{\ell j}$, appartenant à l'ensemble S_t, des opérations à ordonnancer. Cette transition (ou transformation) entre deux états (S_t, P, T) et (S_{t+1}, P', T') survient en ordonnancant l'opération $o_{\ell j} \in S_t$. Une opération $o_{\ell j} \in S_t$ peut être selectionnée pour ordonnancement sur la machine $m_{\ell j}$ si l'opération qui la précède, $o_{\ell-1,j}$, n'appartient pas à l'ensemble S_t.

Or, pour définir la transition entre les deux états nous posons α, le temps de fin de l'opération $o_{\ell j}$; i.e.,

$$\alpha = \max\{T_j + dy_{\ell-1,j}, P_i\} + p_{\ell j} \quad \text{où} \quad i = m_{\ell j}.$$

Donc, si α provient de l'expression $T_j + dy_{\ell j} + p_{\ell j}$, le temps de début de l'opération a été limité par le temps de fin de l'opération précédente de la tâche j. Si α est égal à $P_i + p_{\ell j}$, le temps de début de l'opération a été restreint par le temps de fin de la dernière opération effectuée sur la machine $m_{\ell j}$. Autrement dit, l'opération $o_{\ell j}$ ne peut pas être traitée avant que l'opération qui la précède, $o_{\ell-1,j}$, ne soit complétée *et* que la machine qu'elle exige ne soit libre. Par conséquent, nous prenons le maximum de ces deux temps de fin pour déterminer le temps de début de l'opération en question à ordonnancer.

État au niveau $t+1$

La transition décrite ci-dessus crée le nouvel état (paramètres de sortie) suivant:

$$(S_{t+1}, P', T') = (S_t - \{o_{\ell j}\}, P', T')$$

où

$$P'_i = \begin{cases} P_i & \text{si} \quad i \neq m_{\ell j} \\ \alpha & \text{si} \quad i = m_{\ell j} \end{cases} \quad i = 1, \ldots, m$$

et

$$T'_k = \begin{cases} T_k & \text{si} \quad k \neq j \\ \alpha & \text{si} \quad k = j \end{cases} \quad j = 1, \ldots, n.$$

Remarques concernant le niveau

À un niveau, il est possible de choisir parmi plusieurs transitions, c'est-à-dire qu'il existe plus d'une opération candidate pour l'ordonnancement. Par conséquent, ceci nous ramène à la programmation dynamique qui, dans ce cas, sert à explorer tous les états possibles au niveau en question.

Il faudrait noter que l'état initial (le seul état au niveau 0) est égal à $(N, 0, 0)$, où N est l'ensemble de toutes les opérations à ordonnancer; et l'état final (au niveau $|N|$, cardinalité de l'ensemble N) est égal à (ϕ, P', T').

3.4 Calcul de l'objectif

Ce nouvel objectif introduit des modifications à notre modèle. Puisque les priorités tiennent compte des dates au plus tard et des dates de disponibilité, il n'est plus nécessaire de les avoir dans notre modèle. Donc, nous revenons au premier modèle

présenté pour lui introduire le concept des priorités. Il existe une seule modification à notre premier modèle; l'état considère maintenant un coût; (S_t, P, T, Z). Z est définie par les expressions suivantes:

$$Z' = Z + pr_j \times t_j$$

où

- Z est le coût de l'état prédécésseur,
- t_j est le temps d'attente de la tâche à laquelle est associée l'opération choisie $t_j = \alpha - p_{ij} - T_j$
- pr_j est la priorité de la tâche à laquelle est associée l'opération choisie et
- Z' est le coût du nouvel état.

Ainsi une transition entre deux états est la suivante :

$$(S_{t+1}, P', T', Z') = (S_t - \{o_{ij}\}, P', T', Z + pr_j \times t_j).$$

La définition des P' et T' reste la même que celle du premier modèle. Or, la fonction objectif:

$$F(\phi, P, T, Z) = \min Z$$

où le minimum est trouvé parmi tous les états finaux.

3.5 Traitement des stations parallèlles

Jusqu'à date, notre modèle tient compte de tous nos objectifs et nos contraintes de temps. Pour compléter ce modèle, il existe un dernier facteur à considérer. Il faut incorporer le fait qu'à chaque station il peut y avoir des stations en parallèle. Donc, nous modifions le vecteur P.

A chaque station i, il peut y avoir m_i stations en parallèle. Ainsi, notre vecteur P est modifié de la façon suivante:

P_{ki} est le temps de fin de la dernière opération déjà ordonnancée à la $k^{ième}$ machine de la station i, $k = 1, \ldots m_i$.

Ceci nous pousse à faire un choix entre les machines en parallèle. Nous choisissons la règle de "première terminée première choisie" (FIFO). Donc, pour ordonnancer l'opération $o_{\ell j}$, on choisit la machine avec le plus petit P_{ki} $(i = m_{\ell j})$, parmi les $k = 1, \ldots, m_i$.

3.6 Modèle avec date au plus tard et date de disponibilité ($d_{\ell j}$ et $r_{\ell j}$)

Pour pouvoir appliquer les différents critères que nous avons vu au chapitre précédant, il faut que notre modèle tienne compte des dates au plus tard ainsi que des dates de disponibilité du matériel. En utilisant les mêmes notations qu'auparavant, nous exposons le nouveau modèle.

3.6.1 Description du nouveau modèle

Comme le premier modèle, nous avons n tâches chacune ayant n_j opérations, à être effectuées sur m machines différentes. De la même façon, notre définition du "FLOW SHOP GÉNÉRALISÉ" est respectée. La description de temps d'exécution, $p_{\ell j}$, et celle du temps de délai, $dy_{\ell j}$, sont également les mêmes. L'introduction des $r_{\ell j}$ et $d_{\ell j}$ marque la différence entre ce modèle et le premier. Pour chaque opération, il existe une date de disponibilité, $r_{\ell j}$. C'est-à-dire que nous pouvons considérer l'opération à partir du temps $r_{\ell j}$. La date au plus tard, $d_{\ell j}$, de l'opération $o_{\ell j}$ nous impose une contrainte sur l'heure de fin de cette opération.

La définition d'un état reste le même pour ce modèle. Un état est formé de l'ensemble S_t, ainsi que des vecteurs P et T. Donc, l'état au niveau t se présente sous la même forme: (S_t, P, T).

3.6.2 Transition

Une transition dans ce modèle reste la même que celle du premier mais la valeur de α, de P, et de T prennent les nouveaux paramètres en considération.

Ainsi, on définit le temps de fin de l'opération $o_{\ell j}$ par

$$\alpha = \max\{T_j + dy_{\ell-1,j}, P_i, r_{\ell j}\} + p_{\ell j} \quad \text{où} \quad i = m_{\ell j}.$$

Ici, l'opération $o_{\ell j}$ ne peut pas être traitée avant que l'opération qui la précède, $o_{\ell-1,j}$, ne soit terminée, avant que la machine qu'elle exige ne soit libre, *et* avant

qu'elle même ne soit disponible à être traitée. Donc, nous prenons le maximum de ces trois valeurs pour assurer que les trois conditions soient respectées.

3.6.3 État au niveau $t+1$

En considérant les dates au plus tard dans le modèle, la transition entre deux états nous donne le nouvel état suivant :

$$(S_{t+1}, P', T') = (S_t - \{o_{\ell j}\}, P', T')$$

où

$$P'_i = \begin{cases} P_i & \text{si} \quad i \neq m_{\ell j} \\ \alpha & \text{si} \quad i = m_{\ell j} \quad \text{et} \quad \alpha \leq d_{\ell j} \\ \infty & \text{si} \quad i = m_{\ell j} \quad \text{et} \quad \alpha > d_{\ell j} \end{cases}$$

et

$$T'_k = \begin{cases} T_k & \text{si} \quad k \neq j \\ \alpha & \text{si} \quad k = j \quad \text{et} \quad \alpha \leq d_{\ell j} \\ \infty & \text{si} \quad k = j \quad \text{et} \quad \alpha > d_{\ell j} \end{cases}$$

En affectant une valeur ∞ aux P_i et T_k quand le temps de fin dépasse la date au plus tard, on pénalise ce retard. On pourrait aussi garder la valeur à α, et dans le cas où $\alpha > d_{\ell j}$, imposer une pénalité à la différence pour créer un objectif minimisant la somme des retards.

Chapitre 4

Méthode de résolution

Le modèle et les objectifs établis, nous pouvons maintenant exposer notre algorithme. Ce chapitre a précisément pour but de présenter l'approche de résolution du problème d'ordonnancement.

Il existe déjà plusieurs méthodes pour résoudre les problèmes d'ordonnancement (voir Carlier (1988), French (1986), Germain (1986), Muhlemann, Lockett, and Farn (1982)), entre autres, la programmation dynamique. Le choix de la méthode de résolution dépend de la taille du problème, du temps disponible pour la résolution, ainsi que les ressources à notre disposition pour sa réalisation. Nous allons utiliser comme ressource informatique le système Sun avec le système d'exploitation UNIX. Donc, l'espace mémoire n'est pas vraiment limité parce que le système fonctionne à mémoire virtuelle. Pour une description du système, voir Horspool, (1986). Comme langage de programmation, le C a été choisi pour sa flexibilité et sa puissance (voir le Van Wyk, (1988)). Il suffit donc de s'occuper du temps informatique qui dépend de la taille du problème et de la méthode de résolution.

Un problème d'atelier est un problème de très grande taille qui dépend du nombre d'opérations à effectuer ainsi que du nombre de stations requises. Pour une problème de petite taille, la solution optimale peut être obtenue avec la programmation dynamique ou une méthode arborescente. Par contre, étant donné que notre problème est de très grande taille, on se contente d'une "bonne" solution (pas nécessairement optimale). Donc, nous optons pour une approche heuristique utilisant la programmation dynamique.

Le problème est continuel vingt-quatre heures sur vingt-quatre. Nous avons donc un système *dynamique* (où l'ensemble des tâches évolue avec le temps).

Par contre, nous le traitons comme un système *statique* (l'ensemble des tâches ne change pas avec le temps) sur un horizon de huit heures.

Globalement, le processus de résolution est constitué de trois parties. Ceci est illustré par le schéma du processus global à la Figure 4.1.

Ce chapitre servira à décrire le détail du sytème de résolution ce qui apparaitra comme *BOÎTE NOIRE* à l'usager. qui est le coeur de notre système de résolution. Les deux chapitres suivants sont destinés à préciser davantage un module à l'intérieur de notre *BOÎTE NOIRE* ainsi que de tracer les conditions requises pour les parties *ENTRÉES* et *SORTIES*.

4.1 Algorithme de résolution

Jusqu'à présent, nous avons présenté des modèles et des objectifs qui recherchent une solution optimale. En réalité, nous voulons trouver une "bonne" solution. Étant donné que notre problème est continuel, il est presque impossible de terminer le travail durant la tranche de temps considéré, i.e., de se rendre à un état final, (ϕ, P, T, Z), tel qu'énoncé dans notre modèle définitif. Nous sommes donc obligés d'aborder le but de trouver une bonne solution différemment.

La répartition d'une journée en tranches de huit heures est l'approche que nous avons choisie. Une tranche de huit heures nous permet de limiter la taille du problème ainsi que d'établir une condition d'arrêt pour l'algorithme. La solution choisie à la fin de ces huit heures ne peut pas être considérée optimale mais, "bonne". Il faut noter qu'il n'existe aucun autre outil spécifique à notre problème.

Donc, étant donné la taille et la complexité du problème à résoudre, une solution réalisable est déjà intéressante.

Les prochaines sections examinent les différents modules de notre *BOÎTE NOIRE.* Le détail de cette dernière est présenté à l'intérieur de la *BOÎTE NOIRE.* à la figure 4.1.

4.2 Création d'un sous-problème

Le premier élément de notre processus de résolution est celui de la *CRÉATION D'UN SOUS-PROBLÈME.* À partir de notre base de données, ce module extrait les informations nécessaires pour décrire le problème pour une tranche de huit heures. C'est un outil qui sert à réduire la taille de notre problème et à transformer les informations de la base de données pour les adapter à notre algorithme et à ses structures de données.

Pour effectuer cette partie, nous avons besoin de deux sources d'informations. La première est le fichier contenant les informations sur les produits et l'autre est le fichier contenant l'état du système à la fin de la dernière tranche de temps. Comme nous verrons en détail plus loin, le fichier des produits contient pour chacun des produits, entre autres, l'ordre d'exécution sur les machines, le temps d'exécution par élément sur chacune des machines, la taille standard d'un lot, la quantité commandée, la priorité, et le nombre de lots associés à cette priorité. L'état du système garde les mêmes informations et indique aussi quels produits se trouvent dans l'atelier et les stations de travail où ils résident.

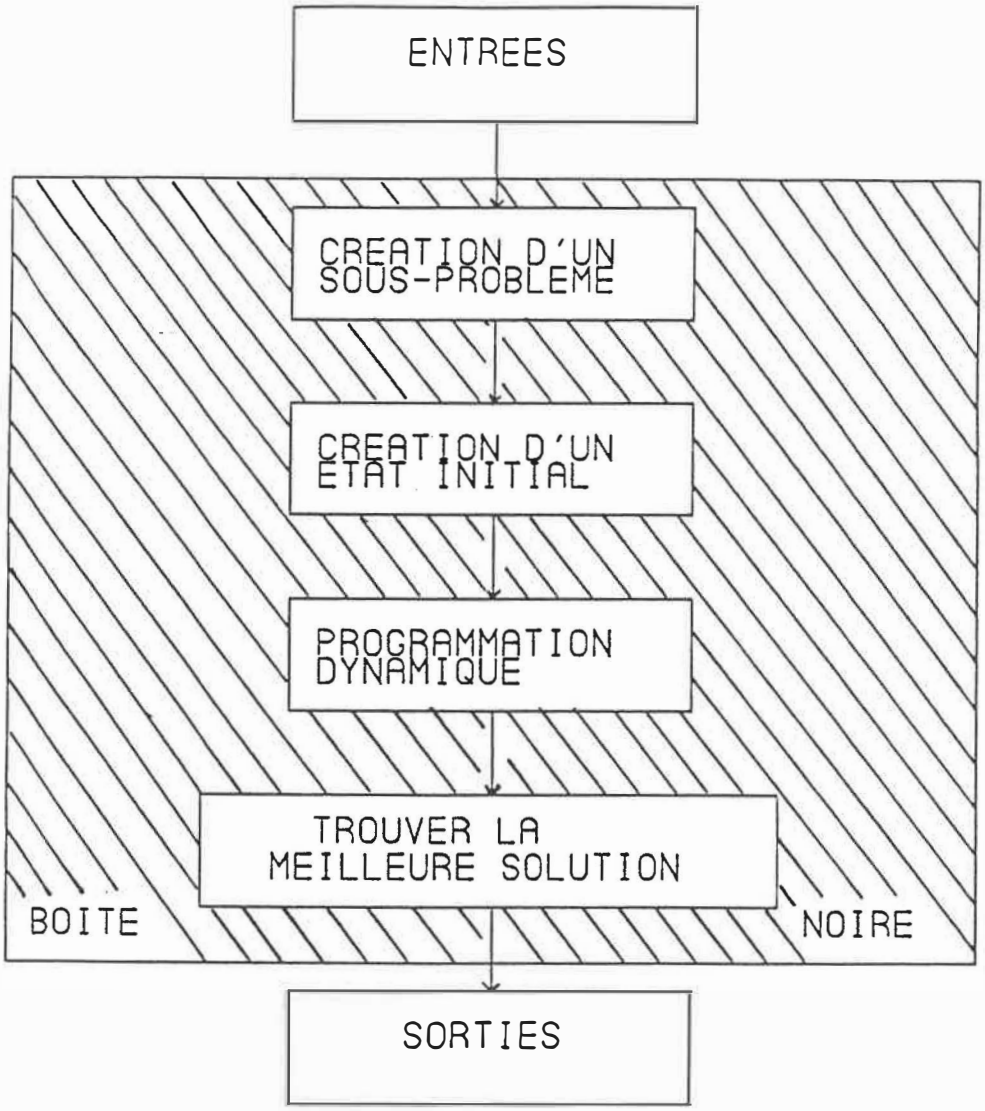

Figure 4.1: Méthode de résolution

À partir de ces informations, ce module créé un fichier contenant les opérations à exécuter au cours des prochaines huit heures. Pour déterminer les tâches à inclure dans ce fichier, il faut considérer leurs priorités et le temps requis pour les effectuer. Les tâches sont choisies en fonction de leurs priorités. On extrait les tâches du fichier *PRODUITS* jusqu'à ce que les huit heures de temps sur chacune des machines soient épuisées.

Donc, nous avons un processus itératif qui permet de créer un fichier contenant l'ensemble des tâches à effectuer pendant cette période de temps. La réalisation de cette partie s'énonce de la façon suivante :

Étape 0: Initialisation des variables

La priorité de recherche = 20.
Temps d'exécution accumulé sur chaque machine = (0,...,0).

Étape 1: Vérification de fin

Si toutes les machines ont un temps accumulé supérieur ou égal à huit heures, l'algorithme se termine; sinon on passe à l'étape 2.

Étape 2: Traitement des produits ayant la priorité désirée

Pour chacun des produit avec la priorité désirée, faire

- créer le nombre de lots (ou de tâches) à effectuer
  i.e., quantité en priorité / taille d'un lot
- pour chaque lot, accumuler le temps requis sur les machines
- chaque lot devient un enregistrement avec toutes les informations nécessaires pour l'exécuter

Diminuer la priorité de recherche par 1.
Retour à l'étape 1.

Dans ce module, l'état du système n'est pas encore considéré. Nous tenons compte de ce dernier à la *CRÉATION DE L'ÉTAT INITIAL*. Quant à la limite des huit heures, nous savons que la solution n'incluera pas toutes les tâches de notre fichier car il existe aussi des tâches déjà sur le plancher à effectuer. Cependant, en ayant un surplus de travail, nous nous assurons qu'il y en ait suffisament pour ces huit heures.

4.3 Création d'un état initial

Pour pouvoir appliquer la programmation dynamique à notre problème, il est nécessaire de créer l'état initial tel que présenté dans notre modèle. Cette section décrira la construction de chaque composante de l'état initial.

4.3.1 Formation de l'ensemble S

Les éléments de l'ensemble S, soit, toutes les opérations à ordonnancer, sont tirés des deux fichiers mentionnés précédemment. Le premier fichier utilisé est celui du sous-problème, créé par le module de la section 4.2. Les opérations, o_{ij}, chacune des tâches du fichier constituent les nouveaux éléments de S. Mais, ce fichier ne génère pas tous les éléments de l'ensemble S. Il manque les opérations associées aux tâches qui sont déjà sur le plancher. C'est le fichier *ÉTAT DU SYSTÈME* qui fournit ces éléments. Ainsi, toutes les opérations qui n'ont pas encore été effectuées venant des tâches sur le plancher, formeront le reste de l'ensemble S.

4.3.2 Les vecteurs P et T

En rappel, P_{ki} est le temps de fin de la dernière opération déjà ordonnancée sur la $k^{ième}$ machine en parallèle de la station i. Or, pour créer le vecteur P, il faut connaître la disponibilité des machines à chaque station pendant la tranche de temps. Dans la base de données, il existe un fichier *MACHINES* qui, pour chacune des stations, indique le nombre de machines disponibles en parallèle. À partir de ceci, nous pouvons créer le vecteur P avec ses dimensions nécessaires.

Une fois créé, il faut initialiser le vecteur P. Le fichier *ÉTAT DU SYSTÈME* permet d'obtenir cette information. Le vecteur P de la dernière tranche de temps a été conservé dans ce fichier. Donc, le nouveau P est initialisé à la valeur de l'ancien vecteur en excluant les machines qui ne sont plus disponibles.

Le fichier contenant le sous-problème ainsi que celui de l'état du système nous

aideront à initialiser le vecteur T. Pour faire un autre rappel, T_j est le temps de fin de la dernière opération, appartenant à la tâche j, déjà ordonnancée. La dimension de T est définie par le total des tâches à effectuer tirées des deux fichiers mentionnés. Puisque nous commençons une nouvelle tranche de temps, les valeurs de tous les T_j sont les mêmes: le temps de début de la tranche de huit heures.

4.3.3 Le coût Z

Étant donné que nous commençons une nouvelle tranche de temps, nous la considérons étant un nouveau problème. Les coûts survenus au cours de la dernière tranche de temps et associés au tâches encore résidentes sur le plancher sont donc éliminés. Nous recommençons avec $Z = 0$.

4.4 Programmation dynamique

Nous avons construit notre état initial qui, en termes de programmation dynamique, est considéré comme étant le premier niveau de génération d'états. Notre modèle indique que si on doit trouver la solution optimale, nous avons à effectuer $|N|$ niveaux de génération d'états pour arriver à l'état final (ϕ, P, T, Z). Dans la section précédente, nous avons éliminé cette possibilité car notre problème est trop gros. Donc, la condition d'arrêt sera basée sur la contrainte de temps. Notons que s'il manque de travail, il suffit d'aller en extraire du fichier *PRODUITS*.

L'algorithme de programmation dynamique se présente comme suit:

Étape 0: Initialisation des temps

Définir le début et la fin de la tranche de temps.

Étape 1: Vérification de fin de la tranche de temps

Si pour tous les états du niveau actuel, le temps de fin de la dernière opération ordonnancée sur chacune des stations de travail est supérieur ou égal à la fin de la tranche de temps, trouver la meilleure solution; sinon passer à l'étape 2.

Étape 2: Vérification des états disponibles

S'il n'y a plus d'états à traiter à ce niveau de programmation dynamique:

- Passer au prochain niveau de la programmation dynamique
- Retourner à l'étape 1.

Sinon passer à l'étape 3.

Étape 3: Traitement d'un état

Pour chaque opération prête à être ordonnancée

- Créer un état successeur.
- Retourner à l'étape 2.

À la première étape, nous vérifions si nous avons ordonnancé suffisament d'opérations pour occuper les machines pendant toute la période de la tranche. Si c'est le cas, la programmation dynamique est terminée et nous pouvons trouver la meilleure solution. Nous verrons comment la déterminer à la prochaine section.

Si la tranche de temps n'est pas encore comblée, nous générons les états successeurs pour le prochain niveau de programmation dynamique. Ceci est effectué pour chacun des états du présent niveau. Quand tous les états possibles sont générés, on retourne à l'étape précédante. Le détail du traitement d'un état sera vu au chapitre suivant.

Donc, pour mieux comprendre le résultat de notre algorithme, à chaque niveau, il existe un certain nombre d'états. Pour chaque état, tous les états successeurs sont générés. Nous construisons donc un arbre qui s'élargit avec la profondeur du niveau. Le niveau final est atteint quand le temps de fin de l'opération la plus hâtive dépasse la fin de la tranche de temps. Il ne nous reste alors qu'à trouver la meilleure solution.

4.5 Trouver la meilleure solution

Au cours de la programmation dynamique, à chaque niveau nous conservons un coût accumulé, Z, jusqu'à date. Ce coût est égale à la somme pondérée des temps d'attentes des tâches qui ont été oronnancées. Ceci n'inclut pas le temps d'attentes des tâches qui n'ont pas encore été ordonnancées. Donc, pour considérer ces dernières tâches, nous conservons un autre coût Z_g qui représente le coût glo-

bal incluant toutes les tâches de l'ensemble S. Bien entendu, à chaque niveau, les états à conserver sont déterminés à partir du coût Z. Entre temps, pendant tout le processus, le niveau et l'état ayant le meilleur coût global Z_g est retenu. Quand la programmation dynamique est terminée à cause du critère d'arrêt, à partir de l'état retenu ayant le meilleur Z_g, nous pouvons chercher la meilleure solution. En partant de cet état, on effectue un retour en arrière (ou "backtracking") en déterminant son prédécesseur appartenant au niveau précédent. Cette opération est répétée jusqu'au moment où l'on rencontre l'état initial. Ainsi, nous avons construit la meilleure solution. À la Figure 4.2, nous exposons le détail des deux derniers modules.

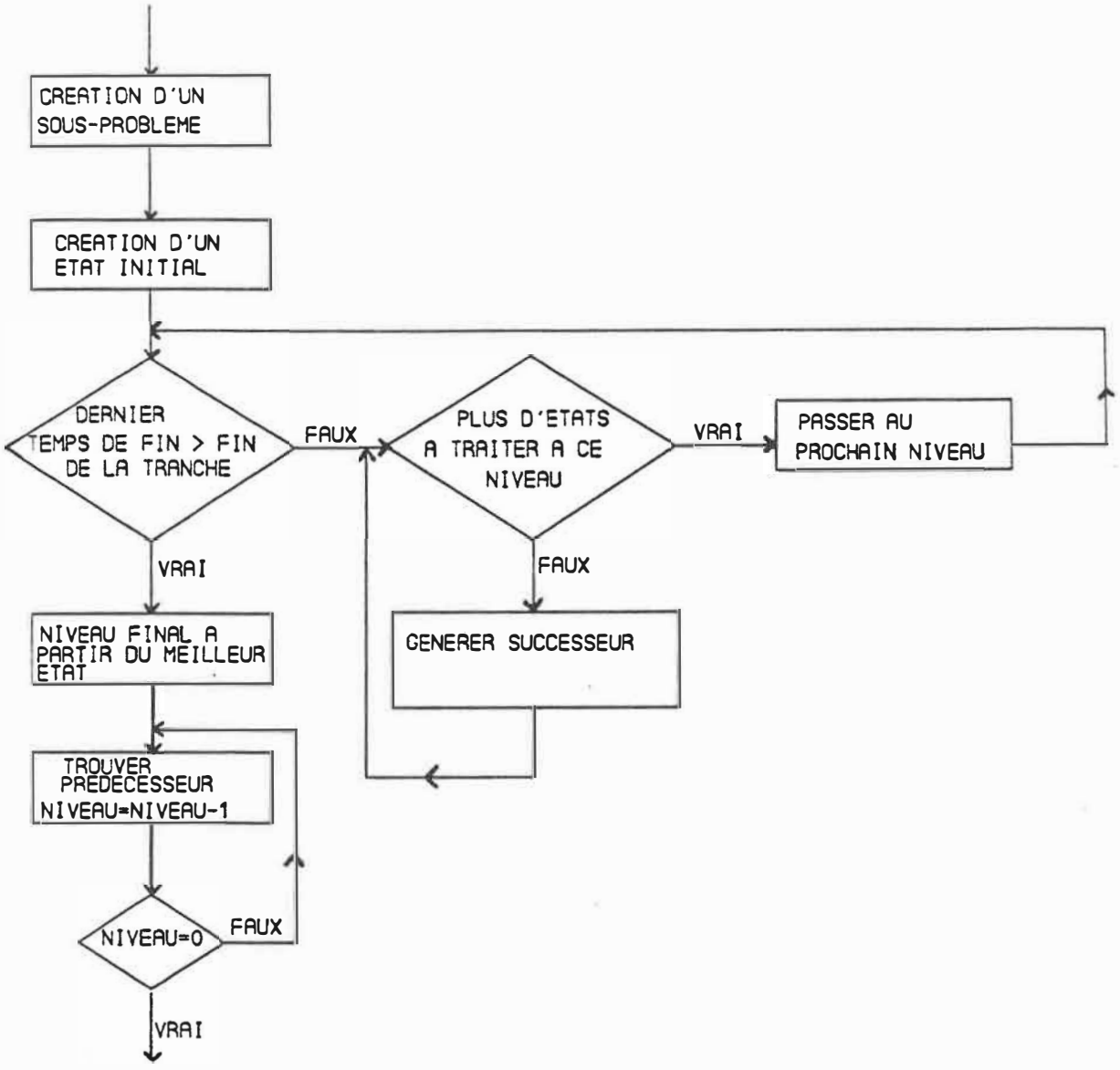

Figure 4.2: Détail des deux derniers modules

Chapitre 5

Traitement d'un état et élimination d'états

Traiter un état consiste en la génération de ses successeurs. Comme nous l'avons vu, la taille du problème augmente rapidement avec le nombre de niveaux de programmation dynamique et aussi avec le nombre d'opérations à effectuer. Il devient donc essentiel de restreindre le nombre de successeurs conservés à chaque niveau. Par conséquent, il faut introduire des méthodes exactes ainsi qu'heuristiques pour réduire la taille des problèmes à traiter par la programmation dynamique. Ce chapitre passera en revue les méthodes envisagées pour résoudre ce type de problème.

5.1 Sous-ensembles d'opérations

Avant de réduire le nombre d'états à traiter, commençons par déterminer le nombre de sous-ensembles d'opérations S à considérer à chaque niveau de la programmation dynamique. Pour un problème de "FLOW SHOP GÉNÉRALISÉ", nous avons un total de $|N| = \sum_{j=1}^{n} n_j$ opérations, où n_j est le nombre d'opérations à effectuer pour la tâche j, et n est le nombre de tâches totales à exécuter. Ainsi, il peut y avoir $2^{|N|} = \Pi_{j=1}^{n} 2^{n_j}$ sous-ensembles d'opérations. Par contre, pour une tâche j, il n'y a que quelques-uns des 2^{n_j} sous-ensembles qui sont réalisables à cause des contraintes de précédence d'opérations. Pour clarifier cette remarque, les sous-ensembles possibles peuvent être énumérés explicitement comme suit:

$$\{o_{1j}, \ldots, o_{n_j,j}\}, \{o_{2j}, \ldots, o_{n_j,j}\}, \ldots, \{o_{n_j-1,j}, o_{n_j,j}\}, \{o_{n_j,j}\}, \emptyset$$

Pour chaque tâche nous avons des sous-ensembles, chacun contenant le reste des opérations à ordonnancer. Il y a donc pour chacune des tâches $n_j + 1$ sous-ensembles réalisables. Nous pouvons ainsi réduire la borne supérieure du nombre

de sous-ensembles réalisables d'opérations à $\Pi_{j=1}^{n}(n_j + 1)$.

Même si cette réduction est substantielle, le problème demeure toujours très grand. Par une méthode heuristique que nous verrons plus loin, nous allons essayer de restreindre davantage le nombre d'états.

5.2 Méthode exacte de réduction d'états

Théoriquement, la programmation dynamique est une approche qui produit une solution optimale car toutes les possibilités sont explorées et on peut choisir la meilleure. Malheureusement, si le problème est trop grand, essayer d'atteindre l'optimalité n'est pas réaliste du point de vue temps de calcul. Il faudrait donc trouver un moyen de limiter l'ensemble des états à explorer. Il existe des méthodes exactes et heuristiques. Nous allons profiter de certaines propriétés du problème pour choisir une méthode exacte.

Rappelons d'abord qu'à chaque niveau de la programmation dynamique, un ensemble d'états possibles est généré. Chaque état a la forme (S, P, T, Z). Si on peut trouver un critère pour éliminer certains états, cela permettra de diminuer le nombre d'états qui devront être générés au prochain niveau.

Donc, pour l'ensemble des états à chaque niveau, nous introduisons la notion d'un ensemble dominant d'états. Pour former un ensemble dominant D, appliquons le raisonnement suivant:

1. $D_t = \emptyset$

2. Pour chaque état (S'_t, P', T', Z') au niveau t:

   (a) S'il existe $(S_t, P, T, Z) \in D$ où $S'_t \equiv S_t$

   si $T_j \leq T'_j \ \forall j$ et $Z \leq Z'$

   – éliminer (S'_t, P', T', Z')

   – retour à 2

   sinon

   si $T_j \geq T'_j \ \forall j$ et $Z \geq Z'$

   – $D_t = D_t \backslash \{(S_t, P, T, Z)\}$ (éliminer (S, P, T, Z))

   – $D_t = D_t \cup \{(S'_t, P', T', Z')\}$ (conserver (S'_t, P', T', Z'))

   – retour à 2

   sinon

   – aller à b

   sinon

   aller à b

   (b) $D_t = D_t \cup \{(S'_t, P', T', Z')\}$ (conserver (S'_T, P', T', Z'))

Ainsi, notre ensemble dominant D part du principe que si S, l'ensemble des opérations qu'il reste à ordonnancer est le même pour deux états et que les temps de fin de toutes les tâches déjà ordonnancées pour un état se terminent plus tôt que ceux du deuxième état, nous pouvons dire que l'horaire partiel créé par le premier état est au moins aussi bon que celui associé au deuxième. Donc, il suffit de considérer les états du premier type qui formeront un *ENSEMBLE EFFICACE D'ÉTATS.*

Pour pouvoir appliquer cette notion de dominance, nous introduisons un autre principe, la relation d'ordre partiel "$\leq$" suivante:

"Soit V, un ensemble de k vecteurs de dimension $n+1$. Pour un vecteur $v \in V$, soit $x_i(v)$ la $i^{ème}$ composante de v. Un ordre partiel $\leq$ sur V est défini de la façon suivante: soient $v, u \in V$, $v \leq u$ si $x_i(v) \leq x_i(u)$ $\forall i = 1, \ldots, n+1$. Pour $v \in V$, v est un *ÉLÉMENT MAXIMAL* de V s'il n'existe aucun $u \in V$ tel que $u \geq v$ et u n'est pas égal à v."

Cette définition, tirée de Kung, Luccio et Preparata (1975) s'applique directement à notre problème qui est celui de trouver un ensemble dominant d'états. V correspond à un ensemble d'états ayant les mêmes S que nous voulons réduire au niveau t. Les vecteurs de dimension $n+1$ se présentent dans notre contexte sous forme de (T, Z) où n est la dimension de T et 1 est celle de Z. Donc, si on applique cette définition à l'ensemble V, nous formons un sous-ensemble d'éléments maximaux.

Or, en correspondance avec notre problème, à un niveau t de programmation dynamique, nous formons une série d'ensembles V_r où $r = 1, \cdots, r_t$, et r_t est le nombre d'ensembles, chacun contenant l'ensemble d'états avec le même S_t. Avec la définition d'ordre partiel, pour chacun de ces ensembles, V_r, nous formons un sous-ensemble V_r' contenant les éléments maximaux. Notre ensemble dominant d'états au niveau t est défini par l'union de ces derniers sous-ensembles; $D_t = V_1' \cup V_2' \cup \cdots \cup V_r'$.

Considérons le problème de trouver tous les éléments maximaux d'un ensemble V_r. La complexité de calcul de ce problème a été étudiée par Kung, Luccio, et Preparata (1975). Ils démontrent que la complexité pour un problème où V_r contient k_r vecteurs de dimension $n \geq 4$ a une borne supérieure dans le pire cas de la forme

suivante:

$$C_n \leq O(k_r n(\log k_r)^{n-2})$$

Ils énoncent également l'algorithme considéré qui utilise plusieurs tris d'ensembles et applique le principe de "diviser-pour-régner". C'est-à-dire que pour un ensemble V_r de vecteurs, ils les placent en ordre lexicographique décroissant de la façon suivante:

$$x_1(v_1) > x_1(v_2) > \ldots > x_1(v_{k_r})$$

où p est le nombre de vecteurs dans l'ensemble V_r. V_r est divisé en deux sous-ensembles $Q = \{v_1, \ldots, v_{p/2}\}$ et $R = \{v_{(p/2)+1}, \ldots, v_p\}$. Ensuite, ils trouvent Q' et R', les sous-ensembles d'éléments maximaux de Q et R respectivement. Remarquons que les éléments de Q' sont aussi des éléments maximaux de V_r mais ceux de R' ne le sont pas nécessairement. Un élément de R' est un élément maximal de V_r si et seulement si il est inférieur ou égal à aucun élément de Q'. Donc, un algorithme récursive est appliqué où T', le sous-ensemble d'éléments de R' qui sont inférieurs ou égal à aucun élément de Q', est formé. $Q' \cup T'$ forme le nouvel ensemble d'éléments maximaux V_M et l'algorithme est réappliqué sur V_M jusqu'à que T' devienne un ensemble vide.

Cette procédure devrait être appliquée à chacun des $V_r, r = 1, \ldots, r_t$, pour former l'ensemble dominant d'états. Ceci réduirait le nombre d'états à générer au prochain niveau $t + 1$, donc diminuerait le temps de calcul dû à la programmation dynamique et aussi épargnerait de l'espace mémoire. Par contre, un problème

moyen pour une tranche de huit heures serait capable d'avoir jusqu'à 50 ou 60 tâches à compléter. Ceci résulterait avec des états ayant des vecteurs de dimension entre 51 et 61. Puisque le temps de calcul de l'algorithme de dominance dépend du nombre de vecteurs à un niveau ainsi que de leur dimension, même en appliquant la propriété de dominance à un problème de telle taille, le temps de tri et de comparaison des vecteurs remplacerait le temps de génération des états n'appartenant pas à l'ensemble dominant. L'utilisation de l'algorithme n'est donc pas justifiée par une amélioration. Par conséquent, il faudrait trouver une autre façon de réduire l'ensemble des états à traiter.

5.3 Méthode heuristique de réduction d'états

Cette section servira à exposer la méthode utilisée pour diminuer le nombre d'états à explorer à chaque niveau de la programmation dynamique. Au lieu d'aborder le problème en essayant de trouver une solution optimale, nous optons pour une approche heuristique. Un compromis sera fait en acceptant de trouver une "bonne" solution (qui n'est pas nécessairement optimale) pour pouvoir y arriver plus rapidement en réduisant les états à explorer. Donc, en appliquant l'heuristique à chaque niveau, nous allons garder un nombre fixe d'états générés pour le prochain niveau. Il est possible que dans ce groupe d'états conservé, l'état qui aboutira à la solution optimale n'existe plus mais le groupe contiendra des états qui produiront de "bonnes" solutions. Ceci dit, nous présentons plus en détail le sous-module de la figure 3. L'heuristique est la suivante:

Heuristique au niveau t

Étape 0: Initialisation

K = le nombre maximum d'états à conserver

C_t = $\emptyset$ (ensemble des états à conserver)

nbtraité = 0

Étape 1: Programmation dynamique

- S'il n'y a plus de successeurs à générer, fin de l'heuristique
- Sinon générer un successeur (transition)

Étape 2: Critère d'arrêt de la génération d'états

Si nbtraité $\leq K$

- Insérer l'état dans C_t de telle façon que ses éléments soient en ordre décroissant de "potentiel" (par rapport à Z)
- nbtraité = nbtraité + 1

Sinon

- Comparer son coût à l'état de C_t avec le moins de "potentiel"
- Si le coût de l'état généré est supérieur à celui de dernier de C_t, on ne le conserve pas

- Sinon, on insère le nouvel état à sa position appropriée et on élimine le dernier état de la liste C_t

Retour à l'étape 1.

C_t est la liste des états à examiner

fin de l'heuristique

sinon

retour à l'étape 2.

L'idée de l'heuristique est de conserver un groupe d'états ayant le "potentiel" de produire de "bonnes" solutions. Examinons le fonctionnement de l'heuristique. À la première étape, on initialise le nombre maximum d'états à conserver. Ce nombre est utile dans les derniers niveaux de la programmation dynamique car conserver un petit pourcentage d'un grand nombre d'états évite l'explosion du nombre d'états. Aux premiers niveaux, tous les états sont conservés pour augmenter les chances de trouver une "bonne" solution.

À l'étape suivante, on s'occupe de générer le prochain successeur de notre niveau. S'il n'en reste plus à générer, c'est la fin de l'heuristique. À la dernière étape, nous déterminons si l'état généré va être conservé dans notre liste d'états ayant le plus de "potentiel" pour aboutir à une bonne solution. Ceci dépend du coût Z. Si l'état est parmi les K meilleurs, on élimine le $K + 1^{ème}$ meilleur.

Or, si on applique cette heuristique à chaque état de chaque niveau, plusieurs branches de la programmation dynamique seront éliminées. Ceci résultera en la diminution de temps, et une solution sera obtenue plus rapidement et avec l'utilisation de moins d'espace mémoire. Malgré le fait qu'on risque d'éliminer la solution

optimale, ce compromis nous permet de trouver une "bonne" solution en temps raisonnable.

Chapitre 6

Les entrées et les sorties

Pour compléter l'étude de notre système de résolution, nous allons examiner les deux modules *ENTRÉES* et *SORTIES* de la figure 4.1. Ce chapitre a pour but de présenter le sytème de base de données requis pour notre algorithme et la méthode utilisée pour produire la solution finale.

Avant de procéder, nous devons préciser que notre méthode de résolution ne considère pas des stations en parallèle qui proviennent d'une autre ligne de production. Donc, la base de données est créée en fonction d'une seule ligne d'assemblage et la solution produit un horaire qui n'enverra aucune opération à une station d'une autre ligne. Les recours possibles, si on veut considérer une deuxième ligne dans notre solution, sont discutés à la section des *RECOMMANDATIONS À L'USAGER* (section 6.3).

Ainsi, en tenant compte de ce détail, nous passons à l'exposé de nos deux modules.

6.1 Les entrées

La base de données est répartie en trois fichiers: le fichier *PRODUITS*, le fichier *MACHINES*, et celui de l'*ÉTAT DU SYSTÈME*. Chacun de ces fichiers doit être constamment mis à jour en fonction des demandes qui se développent, des différentes disponibilités des machines, et du travail en cours dans l'atelier. Cette section décrira les trois fichiers et leur utilisation dans notre système.

6.1.1 Le fichier *PRODUITS*

Nous commençons par décrire le fichier qui contient toutes les informations pertinentes sur les produits. Le fichier consiste en une série d'enregistrements chacun correspondant aux données d'un produit. Le fichier contient les données ne s'appliquant qu'à une seule ligne de production. Pour un produit, un enregistrement contient les informations suivantes:

code	:	le numéro du produit
nom	:	le nom descriptif du produit
taille des lots	:	la taille standard des lots pour ce produit
quantité	:	nombre d'unités en demande pour le mois
priorité	:	la priorité (entre 1 et 20) allouée au produit en tenant compte des dates au plus tard et de la disponibilité des matériaux
quantité en priorité	:	le nombre d'unité à fabriquer auquel est associée la priorité
routage du produit	:	l'ordre de passage d'un lot sur les machines.

En utilisant des méthodes d'index avec une clé primaire (le "code"), nous avons accès direct à chacun des produits. Une demande (la "quantité") du produit est commandée au cours d'un mois mais il n'y a qu'une partie de cette quantité ("quantité en priorité") qui puisse être fabriquée à cause des contraintes de dates au plus tard et de disponibilité des matériaux. Donc, la priorité est associée à cette quantité. Pour le routage du produit, la liste d'opérations en ordre de précédence est décrite. Chaque opération est identifiée par son nom, le numéro de la machine sur laquelle elle doit être effectuée, et le temps d'exécution pour compléter un

élément du lot.

Donc, à partir du fichier *PRODUITS*, un problème est formé en tenant compte de la tranche de huit heures et comprenant les produits et leurs lots avec priorités élevées. L'information sera alors prête pour la *CRÉATION DE L'ÉTAT INITIAL.*

6.1.2 Le fichier *MACHINES*

Nous avons aussi un fichier contenant des enregistrements, chacun définissant une station de travail distincte sur la ligne d'assemblage. Chacun des enregistrements est consititué des champs suivants:

code	:	le numéro du type de station
nom	:	nom descriptif de la station
numéro	:	le numéro alloué à la station pour faciliter son identité dans le "FLOW SHOP GENERALISE"
disponibilité	:	pour chaque tranche de huit heures, le nombre de stations en parallèle disponibles

Par le même principe d'index utilisé pour les produits, l'accès aux enregistrements machines est direct avec la clé primaire (le "code"). Pour faciliter la numérotation des stations et satisfaire notre modèle de "FLOW SHOP GÉNÉRALISÉ", nous numérotons les stations en série. Enfin, selon la tranche de huit heures, les stations de travail ont de différentes disponibilités. Il y en a trois par jour et pour chacune des tranches, dépendant de la disponibilité des opérateurs et des machines, le nombre de stations disponibles est affecté. Avant de passer à l'algorithme, pour la tranche de temps en question, il faut marquer la disponibilité

des stations. Donc, à partir de cette information, le vecteur P peut être créé pour notre état initial.

6.1.3 Le fichier *ÉTAT DU SYSTÈME*

Quand la solution à la fin d'une tranche de huit heures est déterminée, l'ensemble S n'est pas nécessairement vide. C'est-à-dire que le sous-problème créé pour la tranche de huit heures contenait plus d'opérations à effectuer qu'il n'en fallait pour la remplir. Pour conserver cet excès d'opérations, nous avons le fichier *ÉTAT DU SYSTÈME*. Ce fichier contient des enregistrements, chacun identifiant une tâche. Chaque tâche est une série d'opérations qu'il reste à effectuer pour compléter la tâche originale de la dernière tranche de huit heures. Un enregistrement est de la forme suivante:

code : la clé identifiant le produit

priorité : la priorité allouée à ce lot

routage : liste des opérations qu'il reste à effectuer

Ce fichier est aussi utilisé pour créer le nouveau sous-problème. Il est possible de modifier la priorité de chacune des tâches dans ce fichier. Donc, si on veut que ces tâches soient complétées le plus tôt possible, il suffit d'augmenter leurs priorités. S'il existe des tâches plus urgentes venant du fichier *PRODUITS*, les priorités peuvent être diminuées. Quand tous les enregistrements sont ajoutés au fichier du sous-problème, ce fichier d'*ÉTAT DU SYSTÈME* devient vide.

6.2 Les sorties

Une fois qu'une "bonne" solution est trouvée, le module des *SORTIES* peut être séparé en deux sous-modules: impression de la solution et la mise à jour de la base de données.

L'impression de la solution devient simple une fois que le retour en arrière dans le module *TROUVER LA MEILLEURE SOLUTION* a été effectué. La liste de l'ordre des opérations à chaque station est créée et il suffit que de l'imprimer. Donc, pour chaque machine, l'opération est identifiée par le numéro de la tâche, le code du produit, l'heure de début et l'heure de fin de l'opération. Nous verrons des résultats sous cette forme au chapitre 7.

Au niveau de la mise à jour de la base de données, il faut premièrement décider si la solution obtenue va être utilisée. Si la solution est satisfaisante, les fichiers *PRODUITS* et *ÉTATS DU SYSTÈME* sont mis à jour. La quantité de chaque produit qui a été choisie pour le sous-problème est soustraite de la "quantité" pour le mois. Chaque tâche qui n'a pas été complétée devient un nouvel enregistrement dans le fichier pour pouvoir créer l'état initial de la prochaine tranche de huit heures.

Si la solution obtenue n'est pas acceptable, il faut recommencer le processus. C'est-à-dire, il faut modifier des priorités et des quantités en priorité pour ajuster les entrées. Une autre modification possible serait de changer le paramètre pour la programmation dynamique: le nombre maximum d'états à traiter K. Si l'usager est prêt à attendre davantage en temps de calcul, en augmentant cette valeur, il est possible de se rapprocher de la solution optimale. Ce processus pourrait être

répété jusqu'à ce qu'une solution satisfaisante soit trouvée. Quand l'usager décide que la solution est définitive, la mise à jour des fichiers est effectuée et l'horaire est appliqué à la production dans l'atelier. La base de données est donc prête pour la prochaine tranche de huit heures. Evidemment, s'il y a des modifications à faire entre temps, l'usager doit les faire.

6.3 Recommandation à l'usager

Cette section a pour but de décrire le déroulement général du système de résolution, les fichiers qu'il exige et ceux qu'il génère. Ensuite, nous présenterons les recours à prendre dans le cas où une machine en parallèle appartenant à une autre ligne d'assemblage serait nécessaire. Finalement, nous terminons par quelques suggestions pour améliorer l'algorithme de résolution.

6.3.1 Fonctionnement général du système

Le premier fichier utilisé par le système est celui des produits. À partir de ce fichier, un second est généré contenant les nouvelles tâches possibles à considérer pour la tranche de temps en question. Nous avons ensuite le fichier décrivant l'état du système contenat l'information nécessaire pour compléter l'ensemble des tâches à considérer dans l'horaire. Le fichier d'entrée final, celui des machines, établit la disponibilité des stations de travail. Donc, à partir de ces données, l'état initial est créé et la programmation dynamique heuristique est appliquée. Pour éviter l'engorgement de l'espace mémoire, à chaque niveau, les états à traiter sont

conservés dans un fichier en ordre décroissant de potentiel. Donc, il y a autant de fichiers contenant les états que d'opérations traitées. Une fois que le critère d'arrêt est atteint (la tranche de temps est terminée), en utilisant les fichiers générés à chaque niveau, un retour en arrière ("backtracking") est effectué et l'horaire est conservé dans un fichier de sortie. Pour compléter le système, il faudrait générer un dernier fichier: le nouvel état du système après la solution. A partir de ce fichier, et en créant le prochain fichier de nouvelles tâches à considérer, la tranche de temps suivante est prête à planifier.

Si l'horaire généré n'est pas satisfaisant, en réutilisant les fichiers et en modifiant les priorités des tâches, d'autres solutions peuvent être obtenues. L'usager peut donc en produire plusieurs pour choisir la plus acceptable. Quand une solution choisie est effectuée, le fichier *PRODUITS* devrait être mis à jour en fonction de ce qui a été traité.

6.3.2 Machines en parallèle d'une autre ligne

Jusqu'à présent, notre système de résolution considère des stations de travail en parallèle appartenant à une seule ligne de production. Nous avons déjà mentionné qu'une station en parallèle peut aussi appartenir à une seconde ligne d'assemblage. Pour remédier à ce cas, c'est le mode d'utilisation du système qu'il faut considérer.

Supposons qu'une tâche exige une station de travail où les postes en parallèle de la ligne actuelle ne sont pas disponibles. Il faudrait ordonnancer la tâche en partie sur une ligne et le reste sur l'autre. C'est-à-dire, l'usager doit répartir la tâches en trois. Toutes les opérations précédant le poste inopérant sont considérées

sur la première ligne. Une fois que l'horaire est généré sur cette ligne, le temps de disponibilité de la tâche pour la deuxième ligne est obtenu. La tâche est insérée pendant les temps morts de la machine disponible appartenant à la deuxième ligne. Une fois que l'opération est terminée, la tâche est replacée sur la première ligne. Ceci permet l'achèvement d'une tâche urgente.

6.3.3 Suggestions pour l'amélioration du système

Pour que notre système soit le plus utilisable possible, il faudrait qu'il soit facile à utiliser. Pour le moment, il est sous un forme purement utilisable par un informaticien qui comprend le langage C et qui comprend le système à fond. Pour que ce système puisse être utilisé dans l'industrie, une interface usager devrait être créée. Ceci permettrait à un opérateur de l'utiliser sans devoir comprendre la théorie ainsi que tout l'informatique appliqué pour effectuer l'algorithme de résolution. L'interface usager s'occuperait de la base de données à l'entrée et à la sortie de l'algorithme. L'environnement serait tel que la création d'un sous-problème se ferait automatiquement. L'opérateur n'aurait qu'à décider les priorités de chaque produits et d'accepter ou de refuser une solution. S'il refuse la solution, il faudrait qu'il puisse réajuster les paramètres facilement. S'il l'accepte, il pourrait appliquer l'horaire et passer à la prochaine tranche de temps.

Chapitre 7

Les résultats

Dans ce chapitre, nous allons présenter les solutions obtenues à partir de données fictives ainsi que réelles. Pour chacun des problèmes, nous avons fixé le nombre maximum d'états à conserver à 25 et à 50. Les résultats sont présentés par des tableaux. Les deux derniers problèmes traités, étant des problèmes réels, ont aussi été considérés en conservant 37, 63, 75, 87, 100, 150, et 200 états. Les données utilisées pour chacun des problèmes sont regoupées à l'annexe.

Nous commençons par deux problèmes fictifs de petite taille. Ces problèmes ont été construits pour vérifier si le système fonctionne. Le premier, *PROBLÈME 1*, est tiré des trois fichier *PRODUITS*, *ÉTAT DU SYSTEME*, et *MACHINES* (à l'annexe). Il y a 6 tâches et 3 stations en série et aucune en parallèle. Le ***Tableau 7.1*** présente l'horaire généré. La solution est la même avec 25 états conservés qu'avec 50. Le second problème, *PROBLÈME 2*, est le même que le premier à l'exception de la première station de travail où il y a deux stations en parallèle. Le ***Tableau 7.2*** représente la solution pour ce problème. Il est également vrai que la solution avec 25 états conservés est égale à celle avec 50 états conservés. Dans les deux problèmes, il n'y a pas assez de travail pour remplir une tranche de huit heures. Donc, on ne peut pas juger l'efficacité du modèle à partir de ces derniers. Nous pouvons seulement constater que le système fonctionne. Dans une ligne d'un tableau, nous avons le numéro de la station en série, le numéro de la station en parallèle, la tâche, le début et la fin de l'opération. En comparant les deux solutions, nous remarquons qu'en ajoutant une station en parallèle, le coût de l'horaire diminue car les tâches passent moins de temps d'attente.

PROBLÈME 3 est une série de données tirées d'un atelier réel qui fonctionne sous forme de "FLOW SHOP GÉNÉRALISÉ". Ce problème réel considère à peu

près 80 tâches pour l'ordonnancement. Il y a un total d'environs 500 opérations à être affectées sur 18 stations de travail en série prenant une tranche de huit heures. Nous avons un problème qui a beaucoup de travail en cours sur le plancher. Nous avons aussi appliqué notre heuristique à ce problème en conservant 25 états et 50 états. Les résultats sont présentés aux *Tableau 7.3* et *Tableau 7.4*. À cause de la taille du problème, la solution étaillée est présentée en annexe. Par contre nos tableaux résument les résultats. Chaque ligne présente le numéro de la machine, le nombre de tâches affectées à la machine, le pourcentage des huit heures utilisé par la machine, et les remarques expliquant le pourcentage d'utilisation. Notons qu'il y a certaines machines qui n'ont pas été utilisées à cause du manque de travail. Donc, l'analyse de ces machines est exclue des tableaux. En ce qui concerne le pourcentage d'utilisation, il y a certaines machines qui n'utilisent pas la tranche au complet à cause du manque de travail, ou parce qu'elles doivent attendre la fin d'opérations précédentes. Ceci est plutôt vrai pour les machines qui se trouvent vers la fin de la chaîne. Une raison pour ce phénomème est la différence des tailles des opérations à certaines machines par rapport aux autres. Nous avons aussi plusieurs grosses opérations qui s'effectuent à la fin de l'atelier.

Pour réduire le problème des grosses opérations nous avons testés un quatrième problème, *PROBLÈME 4*. Nous avons éliminé les cinq dernières machines du *PROBLÉME 3* pour éviter d'ordonnancer de grosses opérations. Donc nous considérons un atelier avec 13 machines plutôt que 18. Les résultats sont présentés au *Tableau 7.5* et *Tableau 7.6*. En examinant les résultats, nous pouvons voir que la réduction du problème a augmenté le pourcentage d'utilisation de certaines machines mais l'a diminué à d'autres. Donc, une réduction du problème n'a pas réussi à améliorer les temps morts de toutes les stations.

Pour avoir un horaire où les machines travaillent continuellement, il faut que l'horaire soit généré pour plusieurs jours pour permettre d'éliminer le travail en retard sur le plancher et d'ajouter du travail en cours à toutes les machines pour que leurs pourcentages d'utilisation se rapprochent de 100%. Une autre façon de remplir le temps aux machines serait de générer la solution et en utilisant notre base de données originale, remplir les trous par des tâches dont les opérations sont assez petites.

Étant donné que notre coût est une valeur qui ne peut pas être évaluée directement, nous avons décidé de faire d'autres essais pour les comparer. Chaque problème a été traité en conservant 25, 37, 50, 63, 75, 87, 100, 150, et 200 états. Pour pouvoir analyser davantage, nous avons aussi conservé le meilleur coût global parmi les 10 derniers niveaux de la programmation dynamique ainsi que pour les 20 derniers. Les *Tableaux 7.7, 7.8, 7.9, et 7.10* présentent ces résultats et les *Figures 7.1, 7.2, 7.3, et 7.4* exposent leurs courbes correspondantes.

La première chose que nous pouvons soulever est le fait que la forme des deux courbes (10 et 20 états conservés) pour chacun des problèmes se ressemblent. Pour chacun des problèmes, il y a un point indiquant le coût minimum. Un nombre d'états à consever pour le *PROBLÈME 3* dans les deux cas, il serait de 150 car ceci est le point du coût global minimum. Pour le *PROBLÈME 4*, ce serait 50 états à conserver.

Nous observons aussi pour les deux problèmes que les coûts globaux diminuent entre la vérification des 10 derniers niveaux et celle des 20 derniers. À partir de ce fait, nous pouvons conclure qu'il n'est pas nécessaire d'aller trop loin dans la programmation dynamique pour s'approcher de la solution optimale.

En regardant les résultats, nous remarquons que contrairement à ce que nous prévoyons, les coûts globaux n'augmentent pas en augmentant le nombre d'états conservés. Ceci est expliqué par le fait que le nombre d'états conservés à chaque niveau n'est pas déterminé en conservant ceux ayant les meilleurs coûts globaux mais plutôt en conservant ceux avec les meilleurs coûts accumulés. Ce n'est que quand la programmation dynamique est terminée que nous évaluons les meilleurs coûts globaux des 10 ou 20 derniers niveaux.

machine	numpar	tâche	début	fin
1	1	2	2:45	3:10
1	1	3	3:10	3:32
1	1	4	3:32	4:19
1	1	6	4:19	5:24
1	1	1	5:24	7:08
2	1	5	2:45	3:04
2	1	2	3:10	4:08
2	1	3	4:08	5:12
2	1	1	7:08	7:39
3	1	2	4:08	5:45
3	1	6	5:45	7:03

Le cout de cet horaire = 446.696.

Tableau 7.1: PROBLÈME 1 (25 et 50 états convervés)

machine	numpar	tâche	début	fin
1	1	2	2:45	3:10
1	1	4	3:10	3:57
1	2	1	2:45	4:24
1	2	3	4:24	4:54
1	1	6	3:57	5:02
2	1	5	2:45	3:04
2	1	2	3:10	4:08
2	1	1	4:24	4:55
2	1	3	4:55	5:58
3	1	2	4:08	5:45
3	1	6	5:45	7:03

Le coût de cet horaire = 342.540.

Tableau 7.2: PROBLÈME 2 (25 et 50 états convervés)

machine	nombres tâches	% d'util. de tranche	remarques
1	25	27.5	pas assez de travail
2	14	99.4	
4	7	75.8	attente des opérations préc.
5	3	22.5	attente des opérations préc.
6	3	8.3	attente des opérations préc.
8	5	66.0	attente entre les opérations
9	5	85.2	attente entre les opérations
10	2	11.5	attente des opérations préc.
11	6	41.3	attente des opérations préc.
12	2	69.2	attente des opérations préc.
14	2	100.0	grosses opérations
15	6	44.6	attente des opérations préc.
16	1	30.8	attente des opérations préc.
17	2	70.6	attente entre les opérations
18	1	41.3	attente des opérations préc.
Le coût de cet horaire = 1 887 824.500			

Tableau 7.3: PROBLÈME 3 (25 états convervés)

machine	nombres tâches	% d'util. de tranche	remarques
1	25	27.5	pas assez de travail
2	19	99.4	
4	4	53.0	attente des opérations préc.
5	4	27.5	attente des opérations préc.
6	3	8.3	attente des opérations préc.
8	5	72.3	attente entre les opérations
9	5	85.2	attente entre les opérations
10	3	11.5	attente des opérations préc.
11	7	48.3	attente des opérations préc.
12	2	69.2	attente des opérations préc.
14	2	100.0	grosses opérations
15	6	44.6	attente des opérations préc.
16	1	30.8	attente des opérations préc.
17	2	70.6	attente entre les opérations
18	1	41.3	attente des opérations préc.
Le coût de cet horaire = 1 849 176.000			

Tableau 7.4: PROBLÈME 3 (50 états convervés)

machine	nombres tâches	% d'util. de tranche	remarques
1	25	27.5	pas assez de travail
2	11	99.4	
4	3	45.8	attente des opérations préc.
5	1	7.5	attente des opérations préc.
6	4	8.7	attente des opérations préc.
8	5	69.8	attente entre les opérations
9	5	85.2	attente entre les opérations
10	3	14.2	attente des opérations préc.
11	7	60.6	attente des opérations préc.
12	2	69.2	attente des opérations préc.
Le coût de cet horaire = 1 456 849.125			

Tableau 7.5: PROBLÈME 4 (25 états convervés)

machine	nombres tâches	% d'util. de tranche	remarques
1	25	27.5	pas assez de travail
2	11	99.4	
4	3	53.0	attente des opérations préc.
5	1	7.5	attente des opérations préc.
6	4	25.2	attente des opérations préc.
8	5	72.3	attente entre les opérations
9	5	85.2	attente entre les opérations
10	3	26.7	attente des opérations préc.
11	7	48.3	attente des opérations préc.
12	2	69.2	attente des opérations préc.
Le coût de cet horaire = 1 433 275.500			

Tableau 7.6: PROBLÈME 4 (50 états convervés)

Nombre d'état conservés	Coût global
25	140 904 672.000
37	141 489 680.000
50	140 821 264.000
63	141 260 960.000
75	131 949 896.000
87	134 360 816.000
100	132 305 616.000
150	130 052 768.000
200	141 260 944.000

Tableau 7.7: PROBLÈME 3 (Meilleur coût global parmi 10 derniers niveaux)

Nombre d'état conservés	Coût global
25	112 764 824.000
37	111 591 136.000
50	112 440 712.000
63	111 260 960.000
75	106 246 912.000
87	107 883 688.000
100	106 345 112.000
150	105 087 304.000
200	111 589 224.000

Tableau 7.8: PROBLÈME 3 (Meilleur coût global parmi 20 derniers niveaux)

Nombre d'état conservés	Coût global
25	91 988 528.000
37	92 809 792.000
50	91 572 304.000
63	94 599 624.000
75	92 060 296.000
87	92 941 328.000
100	92 060 296.000
150	97 889 568.000
200	91 926 528.000

Tableau 7.9: PROBLÈME 4 (Meilleur coût global parmi 10 derniers niveaux)

Nombre d'état conservés	Coût global
25	72 720 592.000
37	73 637 400.000
50	71 417 104.000
63	74 599 624.000
75	73 143 272.000
87	73 677 856.000
100	73 321 673.000
150	77 095 800.000
200	75 324 236.000

Tableau 7.10: PROBLÈME 4 (Meilleur coût global parmi 20 derniers niveaux)

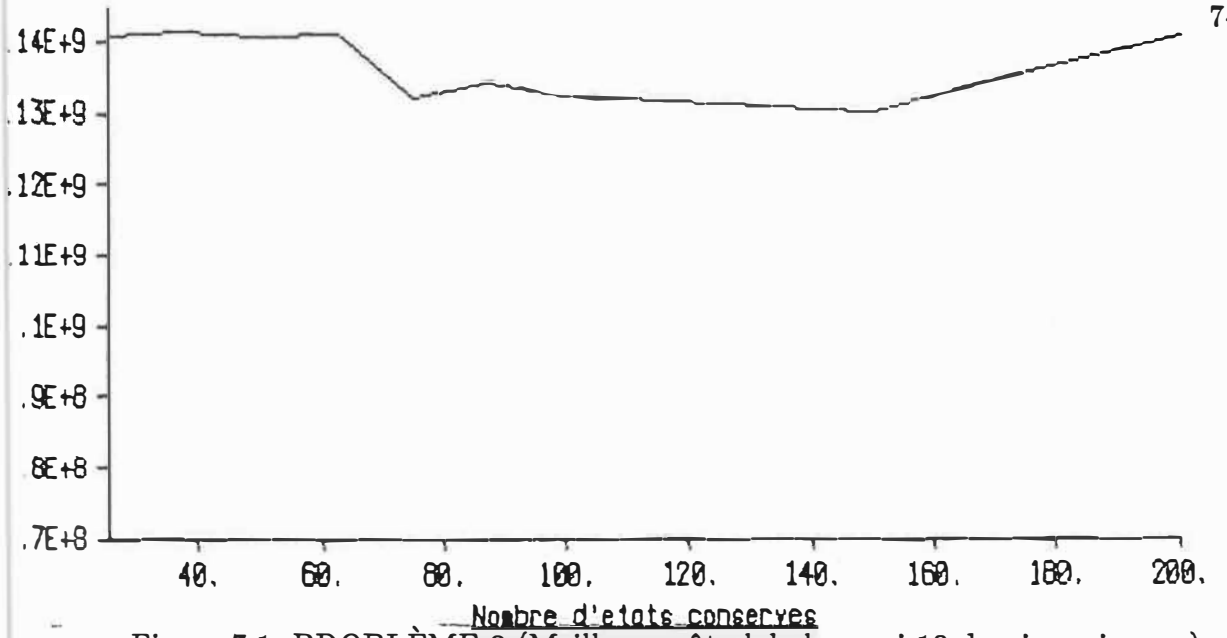

Figure 7.1: PROBLÈME 3 (Meilleur coût global parmi 10 derniers niveaux)

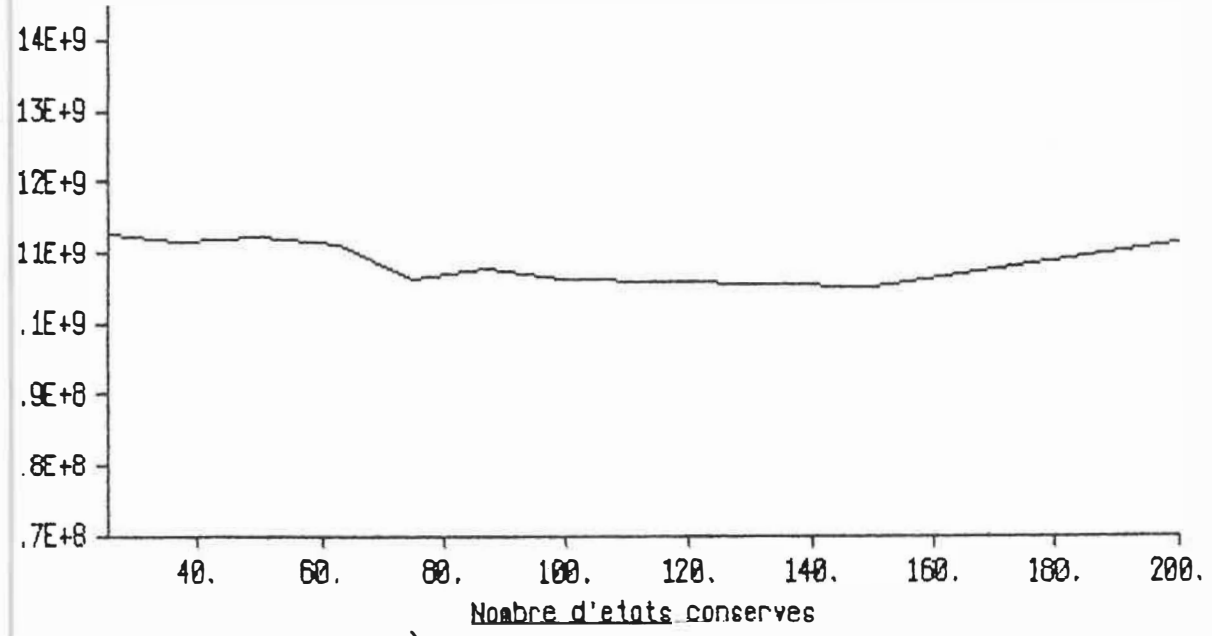

Figure 7.2: PROBLÈME 3 (Meilleur coût global parmi 20 derniers niveaux)

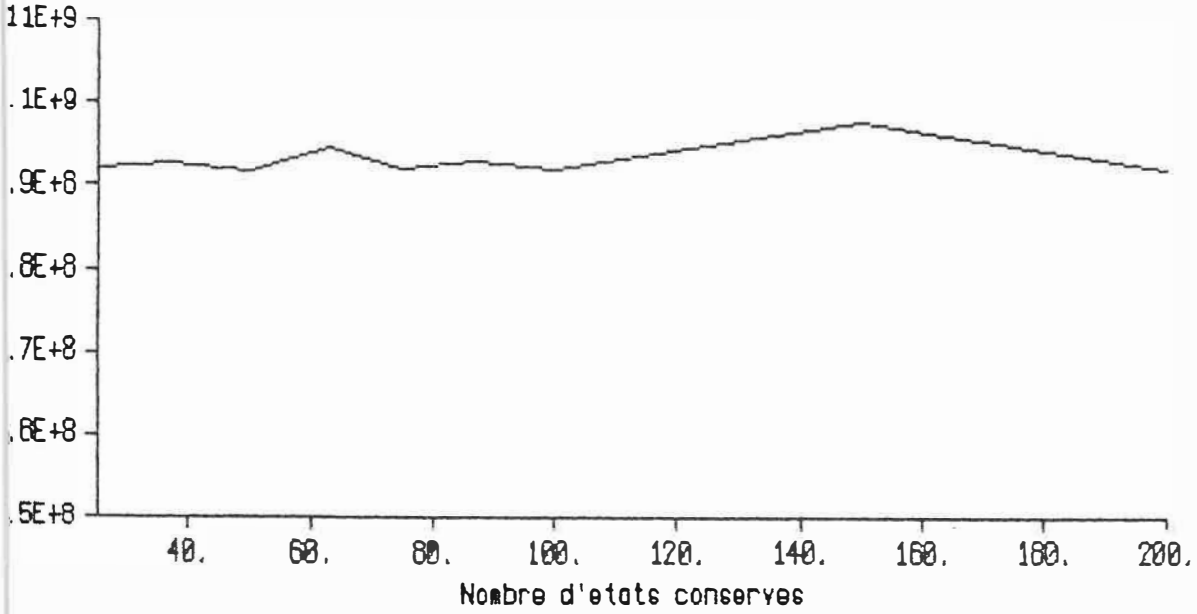

Figure 7.3: PROBLÈME 4 (Meilleur coût global parmi 10 derniers niveaux)

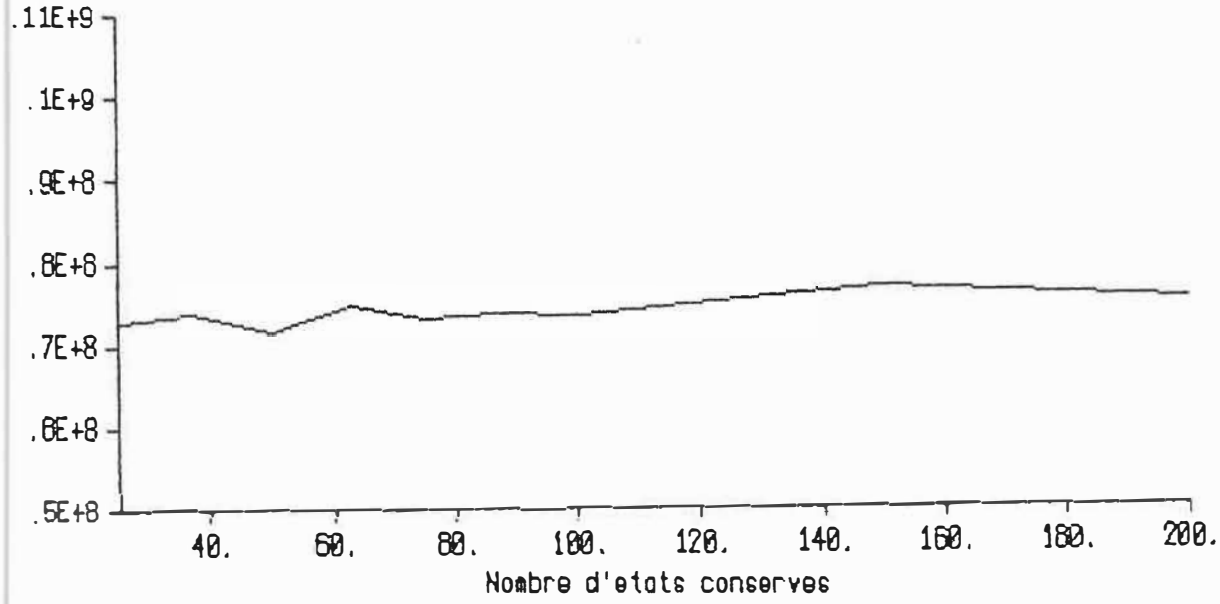

Figure 7.4: PROBLÈME 4 (Meilleur coût global parmi 20 derniers niveaux)

Chapitre 8

Conclusion

Le but de cette étude était de produire un horaire de production pour un atelier de type "FLOW SHOP GÉNÉRALISÉ". Pour atteindre ce but, nous avons implanté un algorithme de programmation dynamique utilisant une heuristique qui permet d'explorer un espace de solutions réalisables. Une "bonne" solution parmi ces dernières est choisie.

Avant d'implanter notre algorithme, nous avons passé en revue les objectifs possibles pour notre problème. Étant donné le manque de deux informations importantes (dates au plus tard et dates de disponibilité des matériaux) dans notre base de données, plusieurs objectifs ne pouvaient être utilisés. Il y avait aussi le fait que notre problème est un problème continuel dans le temps qui nous a empêché d'utiliser la plupart des autres des objectifs présentés. Donc, il a fallu trouver un objectif qui était spécifique à notre problème. Nous avons choisi de pondérer les temps d'attente de chaque tâche par sa priorité de production. En minimisant la somme de ces temps d'attente pondérés, ceci nous a permis d'évaluer une solution en la comaparant à une autre.

Une fois que notre objectif a été établi, nous avons formulé notre modèle pour la programmation dynamique. Le problème étant trop grand, nous avons décidé de réduire l'espace des états à explorer. Nous avons donc présenté une méthode exacte (la dominance). Malheureusement le problème demeure toujours trop grand. Par conséquence, nous avons opté pour une méthode heuristique pour réduire le nombre d'états à traiter. Le principe de la méthode est de conserver les K meilleurs états à chaque niveau de la programmation dynamique. La solution est construite à partir de l'état avec le meilleur coût global.

Sur le plan numérique, les solutions ont été produites à partir de données

fictives et réelles. Nos tests effectués ont réalisés des solutions intéressantes par rapport aux données. Pour mieux évaluer les solutions, il faudrait conserver le coût global parmi tous les niveaux parcourus plutôt que de conserver les meilleurs états à chaque niveaux en fonction des coûts accumulés.

Pour compléter notre système de résolution, il faut envisager la création d'une interface usager. Ceci incluerait tout ce qui concerne les *ENTRÉES* et les *SORTIES*. Il y aurait aussi des améliorations à apporter à l'algorithme pour réduire le temps de calcul.

Bibliographie

1. BAKER, Kenneth R., *Introduction to Sequencing and Scheduling.* John Wiley, New York (1974).

2. BELLMAN, R., *Dynamic Programming.* Princeton University Press (1957).

3. CARLIER, J., CHRETIENNE, P., *Problèmes d'ordonnancement: modélisation/complexité/algorithmes* (1e édition). Masson, Paris (1988).

4. GERMAIN, R., *Conception d'un sytème d'aide à la gestion d'ateliers : étude et comparaison d'algorithmes heuristiques*, Institut national polytechnique de Grenoble (1986).

5. FRENCH, Simon, *Sequencing and Scheduling* (2nd Edition), Ellis Horwood Limited, England (1986).

6. HOLLOWAY, C. A. and NELSON, R.T., "Alternative formulation of the job shop problem with due dates", *Management Science*, 20, 65-75 (1973).

7. HORSPOOL, Nigel R., *Programming in the Berkeley UNIX Environment*, Prentice Hall Canada, Scarborough, Ontario (1986).

8. KUNG, H. T., LUCCIO, F., PREPARATA, F. P., "On Finding the Maxima of a Set of Vectors", *Journal of the Association for Computing Machinery*, 22, 469-476 (octobre 1975).

9. MELLOR, P., "A review of job-shop scheduling", *Opl. Res. Q.*, 17, 161-171 (1966).

10. MUHLEMANN, A.P., LOCKETT, A.G., and FARN, C.K., "Job shop scheduling heuristics and frequency of scheduling", *International Journal of Production Research*, 20, 227-241 (1982).

11. VAN WYK, Christopher J., *Data Structures and C Programs* (1e edition), Addison-Wesley (1988).

Annexe A

Dans chacun des problèmes suivants, nous présentons trois fichiers de données. Les formats de chacun des fichiers sont disposés de la façon suivante:

FICHIER "PRODUITS"

numéro de la tranche de temps

heure de début de la tranche, minute de début de la tranche

nombre de tâches, nombre d'opérations

code du produit # 1, taille du lot, priorité, nombre d'opérations

numéro de la machine, temps d'exécution

⋮

numéro de la machine, temps d'exécution

⋮

code du produit # x, taille du lot, priorité, nombre d'opérations

numéro de la machine, temps d'exécution

⋮

numéro de la machine, temps d'exécution

FICHIER "ÉTAT DU SYSTÉME

nombre de tâches, nombre d'opérations

code du produit # 1, taille du lot, priorité, nombre d'opérations

numéro de la machine, temps d'exécution

⋮

numéro de la machine, temps d'exécution

⋮

code du produit # x, taille du lot, priorité, nombre d'opérations

numéro de la machine, temps d'exécution

⋮

numéro de la machine, temps d'exécution

FICHIER "MACHINES"

nombre de machines

code de la machine #

1, disponibilité à la tranche # 1, dispon # 2, dispon # 3

⋮

code de la machine #

m, disponibilité à la tranche # 1, dispon # 2, dispon # 3

Problème 1

FICHIER "PRODUITS"

```
1
2 45
4 8
a000000001 20 1 2
1 5.231
2 1.543
a000000002 18 2 3
1 1.435
2 3.231
3 5.364
a000000003 15 2 2
1 1.472
2 4.231
a000000004 20 1 1
1 2.326
```

FICHIER "ETAT DU SYSTEME"

```
2 3
a000000004 20 4 1
2 0.987
a000000005 20 1 2
1 3.234
3 3.875
```

FICHIER "MACHINES"

```
3
mach1 1 1 2
mach2 1 2 2
mach3 1 2 2
```

Problème 2

FICHIER "PRODUITS"

```
1
2 45
4 8
a000000001 20 1 2
1 5.231 4.979
2 1.543
a000000002 18 2 3
1 1.435 1.675
2 3.231
3 5.364
a000000003 15 2 2
1 1.472 2.001
2 4.231
a000000004 20 1 1
1 2.326
```

FICHIER "ETAT DU SYSTEME"

```
2 3
a000000004 20 4 1
2 0.987
a000000005 20 1 2
1 3.234 3.657
3 3.875
```

FICHIER "MACHINES"

```
3
mach1 2 1 2
mach2 1 2 2
mach3 1 2 2
```

Problème 3

FICHIER "PRODUITS"

```
1
2 00
16 145
A0000001 20 18 10
1 0.218
2 0.578
4 0.889
8 4.918
9 4.442
10 0.945
11 1.670
12 0.490
14 25.991
15 1.174
A0000001 20 18 10
1 0.218
2 0.578
4 0.889
8 4.918
9 4.442
10 0.945
11 1.670
12 0.490
14 25.991
15 1.174
A0000001 20 18 10
1 0.218
2 0.578
4 0.889
8 4.918
9 4.442
10 0.945
11 1.670
12 0.490
14 25.991
15 1.174
A0000001 20 18 10
```

```
1 0.218
2 0.578
4 0.889
8 4.918
9 4.442
10 0.945
11 1.670
12 0.490
14 25.991
15 1.174
A0000001 20 18 10
1 0.218
2 0.578
4 0.889
8 4.918
9 4.442
10 0.945
11 1.670
12 0.490
14 25.991
15 1.174
A0000001 20 18 10
1 0.218
2 0.578
4 0.889
8 4.918
9 4.442
10 0.945
11 1.670
12 0.490
14 25.991
15 1.174
A0000002 20 18 8
1 0.376
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000002 20 18 8
```

```
1 0.376
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000002 20 18 8
1 0.376
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000002 20 18 8
1 0.376
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000002 20 18 8
1 0.376
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000003 20 18 9
1 0.264
2 0.487
5 1.810
9 2.802
10 1.342
11 1.558
```

```
12 0.461
14 0.900
15 1.119
A0000003 20 18 9
1 0.264
2 0.487
5 1.810
9 2.802
10 1.342
11 1.558
12 0.461
14 0.900
15 1.119
A0000003 20 18 9
1 0.264
2 0.487
5 1.810
9 2.802
10 1.342
11 1.558
12 0.461
14 0.900
15 1.119
A0000003 20 18 9
1 0.264
2 0.487
5 1.810
9 2.802
10 1.342
11 1.558
12 0.461
14 0.900
15 1.119
A0000003 20 18 9
1 0.264
2 0.487
5 1.810
9 2.802
10 1.342
11 1.558
12 0.461
14 0.900
```

15 1.119

FICHIER "ETAT DU SYSTEME"

```
84 523
A000000004 20 20 12
2 5.132
6 0.650
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
14 7.357
15 2.677
17 27.650
18 9.944
A000000004 20 10 12
2 5.132
6 0.650
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
14 7.357
15 2.677
17 27.650
18 9.944
A000000005 20 20 13
1 0.164
2 5.132
6 0.650
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
14 7.357
15 2.677
17 26.475
```

```
18 9.944
A000000005 20 20 13
1 0.164
2 5.132
6 0.650
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
14 7.357
15 2.677
17 26.475
18 9.944
A000000005 20 20 13
1 0.164
2 5.132
6 0.650
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
14 7.357
15 2.677
17 26.475
18 9.944
A000000005 20 20 13
1 0.164
2 5.132
6 0.650
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
14 7.357
15 2.677
17 26.475
```

```
18 9.944
A000000005 20 20 13
1 0.164
2 5.132
6 0.650
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
14 7.357
15 2.677
17 26.475
18 9.944
A000000005 20 20 13
1 0.164
2 5.132
6 0.650
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
14 7.357
15 2.677
17 26.475
18 9.944
A000000005 20 20 12
2 5.132
6 0.650
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
14 7.357
15 2.677
17 26.475
18 9.944
```

```
A000000005 20 20 10
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
14 7.357
15 2.677
17 26.475
18 9.944
A000000005 20 20 10
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
14 7.357
15 2.677
17 26.475
18 9.944
A000000005 20 20 10
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
14 7.357
15 2.677
17 26.475
18 9.944
A000000005 20 20 10
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
14 7.357
15 2.677
```

```
17 26.475
18 9.944
A000000005 20 10 10
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
14 7.357
15 2.677
17 26.475
18 9.944
A000000005 20 10 6
12 22.923
13 1.402
14 7.357
15 2.677
17 26.475
18 9.944
A000000005 20 15 3
15 2.677
17 26.475
18 9.944
A000000005 20 25 2
17 26.475
18 9.944
A000000005 20 15 1
18 9.944
A0000006 20 20 5
8 5.582
9 1.048
10 2.095
11 1.074
14 3.000
A0000006 20 20 3
9 2.332
11 1.485
12 10.547
A0000006 20 20 3
9 2.332
11 1.485
```

```
12 10.547
A0000006 20 20 2
11 1.485
12 10.547
A0000006 20 20 2
11 1.485
12 10.547
A0000006 20 20 2
11 1.485
12 10.547
A0000006 20 20 2
11 1.485
12 10.547
A0000006 20 20 1
12 10.547
A0000007 20 20 3
14 2.260
15 1.799
16 5.427
A0000007 20 15 1
16 5.427
A0000008 20 15 7
9 5.909
10 1.465
11 2.183
12 5.909
13 1.351
14 30.001
15 1.428
A0000008 20 15 2
14 30.001
15 1.428
A0000008 20 15 2
14 30.001
15 1.428
A0000008 20 15 1
15 1.428
A0000009 20 20 4
11 1.670
12 0.490
14 25.991
15 1.174
```

```
A0000009 20 20 2
14 25.991
15 1.174
A0000009 20 20 2
14 25.991
15 1.174
A0000009 20 20 2
14 25.991
15 1.174
A0000009 20 20 2
14 25.991
15 1.174
A0000009 20 10 2
14 25.991
15 1.174
A0000009 20 20 1
15 1.174
A0000009 20 10 1
15 1.174
A0000010 20 20 7
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 7
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 7
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
14 3.481
```

```
15 1.491
A0000010 20 20 7
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 7
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 7
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 7
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 7
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 7
```

```
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 7
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 7
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 7
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 7
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 7
2 0.536
4 1.024
```

```
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 7
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 7
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 7
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 7
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 5
9 3.824
11 1.582
12 0.683
14 3.481
```

```
15 1.491
A0000010 20 20 5
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 5
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 5
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 5
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 5
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 5
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 5
9 3.824
11 1.582
12 0.683
14 3.481
```

```
15 1.491
A0000010 20 20 5
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 5
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 5
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 5
9 3.824
11 1.582
12 0.683
14 3.481
15 1.491
A0000010 20 20 2
14 3.481
15 1.491
A0000010 20 10 1
15 1.491
A0000011 20 20 8
1 0.376
2 0.536
4 5.852
9 3.824
11 2.148
12 0.683
14 3.481
15 1.491
A0000011 20 20 8
1 0.376
2 0.536
```

```
4 5.852
9 3.824
11 2.148
12 0.683
14 3.481
15 1.491
A0000011 20 10 8
1 0.376
2 0.536
4 5.852
9 3.824
11 2.148
12 0.683
14 3.481
15 1.491
A0000012 20 20 10
2 0.728
4 1.253
5 0.829
7 0.192
8 5.036
9 3.032
11 1.495
12 1.015
14 3.983
15 1.460
A0000012 20 20 10
2 0.728
4 1.253
5 0.829
7 0.192
8 5.036
9 3.032
11 1.495
12 1.015
14 3.983
15 1.460
A0000012 20 20 10
2 0.728
4 1.253
5 0.829
7 0.192
```

```
8 5.036
9 3.032
11 1.495
12 1.015
14 3.983
15 1.460
A0000012 20 20 10
2 0.728
4 1.253
5 0.829
7 0.192
8 5.036
9 3.032
11 1.495
12 1.015
14 3.983
15 1.460
A0000012 20 20 10
2 0.728
4 1.253
5 0.829
7 0.192
8 5.036
9 3.032
11 1.495
12 1.015
14 3.983
15 1.460
A0000012 20 20 10
2 0.728
4 1.253
5 0.829
7 0.192
8 5.036
9 3.032
11 1.495
12 1.015
14 3.983
15 1.460
A0000012 20 20 10
2 0.728
4 1.253
```

```
5 0.829
7 0.192
8 5.036
9 3.032
11 1.495
12 1.015
14 3.983
15 1.460
A0000012 20 20 10
2 0.728
4 1.253
5 0.829
7 0.192
8 5.036
9 3.032
11 1.495
12 1.015
14 3.983
15 1.460
A0000012 20 10 2
14 3.983
15 1.460
```

FICHIER "MACHINES"

```
18
MACH1 1 1 0
MACH2 1 1 0
MACH3 1 1 0
MACH4 1 1 0
MACH5 1 1 0
MACH6 1 1 0
MACH7 1 1 0
MACH8 1 0 0
MACH9 1 0 0
MACH10 1 0 0
MACH11 1 0 0
MACH12 1 0 0
MACH13 1 0 0
MACH14 1 1 1
MACH15 1 1 1
MACH16 1 1 1
MACH17 1 1 1
MACH18 1 1 1
```

25 états conservés-PROBLÈME 3

MACHINE	NUMPAR	TACHE	DEBUT	FIN
1	1	20	2:00	2:03
1	1	16	2:03	2:08
1	1	9	2:08	2:16
1	1	91	2:16	2:23
1	1	3	2:23	2:27
1	1	2	2:27	2:32
1	1	4	2:32	2:36
1	1	6	2:36	2:41
1	1	5	2:41	2:45
1	1	1	2:45	2:49
1	1	14	2:49	2:55
1	1	12	2:55	3:00
1	1	10	3:00	3:07
1	1	13	3:07	3:13
1	1	15	3:13	3:18
1	1	11	3:18	3:25
1	1	8	3:25	3:33
1	1	7	3:33	3:40
1	1	24	3:40	3:44
1	1	23	3:44	3:47
1	1	22	3:47	3:50
1	1	19	3:50	3:54
1	1	21	3:54	3:57
1	1	89	3:57	4:04
1	1	90	4:04	4:12
2	1	20	2:03	3:45
2	1	91	3:45	3:56
2	1	12	3:56	4:06
2	1	10	4:06	4:17
2	1	13	4:17	4:26
2	1	15	4:26	4:36
2	1	11	4:36	4:47
2	1	8	4:47	4:58
2	1	7	4:58	5:08
2	1	18	5:08	6:51
2	1	90	6:51	7:02

2	1	89	7:02	7:12
2	1	21	7:12	8:55
2	1	19	8:55	10:38
2	1	22	10:38	12:20
2	1	14	12:20	12:30
2	1	1	12:30	12:42
2	1	23	12:42	14:24
2	1	5	14:24	14:36
2	1	6	14:36	14:47
2	1	4	14:47	14:59
2	1	2	14:59	15:10
2	1	3	15:10	15:22
2	1	24	15:22	17:05
2	1	9	17:05	17:15
2	1	16	17:15	17:25
2	1	73	17:25	17:36
2	1	72	17:36	17:47
2	1	71	17:47	17:57
2	1	70	17:57	18:08
2	1	69	18:08	18:19
2	1	74	18:19	18:29
2	1	58	18:29	18:40
2	1	61	18:40	18:51
2	1	64	18:51	19:02
2	1	66	19:02	19:12
2	1	68	19:12	19:23
2	1	60	19:23	19:34
2	1	63	19:34	19:44
2	1	67	19:44	19:55
2	1	59	19:55	20:06
2	1	65	20:06	20:17
2	1	62	20:17	20:27
2	1	57	20:27	20:38
2	1	95	20:38	20:53
2	1	97	20:53	21:07
2	1	92	21:07	21:22
2	1	98	21:22	21:36
2	1	94	21:36	21:51
2	1	93	21:51	22:05
2	1	96	22:05	22:20
2	1	99	22:20	22:35
2	1	17	22:35	24:17

2	1	25	24:17	2:00
4	1	91	3:56	5:53
4	1	11	5:53	6:14
4	1	8	6:14	6:34
4	1	7	6:34	6:55
4	1	10	6:55	7:15
4	1	90	7:15	9:12
4	1	89	9:12	11:09
5	1	12	4:06	4:42
5	1	13	4:42	5:18
5	1	15	5:18	5:54
6	1	20	3:45	3:58
6	1	18	6:51	7:04
6	1	21	8:55	9:08
8	1	28	2:00	3:10
8	1	20	3:58	5:09
8	1	18	7:04	8:14
8	1	30	8:14	9:24
8	1	21	9:24	10:35
8	1	26	10:35	11:45
8	1	29	11:45	12:55
8	1	27	12:55	14:05
8	1	35	14:05	15:57
9	1	45	2:00	3:58
9	1	20	5:09	6:49
9	1	91	6:49	8:06
9	1	8	8:06	9:22
9	1	18	9:22	11:03
9	1	90	11:03	12:20
9	1	30	12:20	14:00
9	1	10	14:00	15:17
9	1	7	15:17	16:33
9	1	11	16:33	17:50
9	1	15	17:50	18:46
9	1	13	18:46	19:42
9	1	12	19:42	20:38
9	1	28	20:38	22:19

9	1	36	22:19	23:05
9	1	37	23:05	23:52
9	1	75	23:52	1:09
9	1	80	1:09	2:25
9	1	82	2:25	3:42
9	1	86	3:42	4:58
9	1	81	4:58	6:14
9	1	85	6:14	7:31
9	1	77	7:31	8:47
9	1	79	8:47	10:04
9	1	76	10:04	11:20
9	1	84	11:20	12:37
9	1	78	12:37	13:53
9	1	83	13:53	15:10
10	1	45	3:58	4:27
10	1	20	6:49	7:15
11	1	38	2:00	2:29
11	1	45	4:27	5:11
11	1	20	7:15	7:59
11	1	91	8:06	8:49
11	1	8	9:22	9:54
11	1	39	9:54	10:24
11	1	41	10:24	10:53
11	1	40	10:53	11:23
11	1	49	11:23	11:57
12	1	38	2:29	6:00
12	1	20	7:59	15:37
12	1	91	15:37	15:51
12	1	8	15:51	16:04
12	1	31	16:04	23:43
12	1	45	23:43	1:41
12	1	42	1:41	5:12
14	1	100	2:00	3:19
14	1	54	3:19	11:59
14	1	46	11:59	21:59
14	1	47	21:59	7:59
14	1	43	7:59	8:44
14	1	87	8:44	9:54

14	1	52	9:54	18:34
14	1	51	18:34	3:13
14	1	50	3:13	11:53
14	1	53	11:53	20:33
15	1	32	2:00	2:53
15	1	100	3:19	3:48
15	1	56	3:48	4:12
15	1	88	4:12	4:42
15	1	48	4:42	5:10
15	1	55	5:10	5:34
16	1	44	2:00	3:48
17	1	32	2:53	11:43
17	1	33	11:43	20:32
18	1	34	2:00	5:18

Le coût de cet horaire = 140904672.000.

50 états convervés-PROBLÈME 3

MACHINE	NUMPAR	TACHE	DEBUT	FIN
1	1	19	2:00	2:03
1	1	15	2:03	2:08
1	1	4	2:08	2:12
1	1	91	2:12	2:20
1	1	5	2:20	2:24
1	1	3	2:24	2:29
1	1	2	2:29	2:33
1	1	6	2:33	2:37
1	1	1	2:37	2:42
1	1	14	2:42	2:47
1	1	12	2:47	2:52
1	1	13	2:52	2:58
1	1	16	2:58	3:03
1	1	10	3:03	3:10
1	1	11	3:10	3:18
1	1	9	3:18	3:25
1	1	8	3:25	3:33
1	1	7	3:33	3:40
1	1	20	3:40	3:44
1	1	22	3:44	3:47
1	1	21	3:47	3:50
1	1	23	3:50	3:54
1	1	24	3:54	3:57
1	1	89	3:57	4:04
1	1	90	4:04	4:12
2	1	19	2:03	3:45
2	1	91	3:45	3:56
2	1	16	3:56	4:06
2	1	18	4:06	5:49
2	1	24	5:49	7:31
2	1	90	7:31	7:42
2	1	7	7:42	7:53
2	1	8	7:53	8:03
2	1	89	8:03	8:14
2	1	13	8:14	8:24
2	1	11	8:24	8:35

2	1	9	8:35	8:45
2	1	12	8:45	8:55
2	1	10	8:55	9:06
2	1	14	9:06	9:15
2	1	1	9:15	9:27
2	1	6	9:27	9:39
2	1	2	9:39	9:50
2	1	23	9:50	11:33
2	1	21	11:33	13:15
2	1	22	13:15	14:58
2	1	20	14:58	16:41
2	1	3	16:41	16:52
2	1	5	16:52	17:04
2	1	4	17:04	17:15
2	1	15	17:15	17:25
2	1	74	17:25	17:36
2	1	73	17:36	17:47
2	1	72	17:47	17:57
2	1	71	17:57	18:08
2	1	70	18:08	18:19
2	1	25	18:19	20:01
2	1	57	20:01	20:12
2	1	59	20:12	20:23
2	1	61	20:23	20:33
2	1	63	20:33	20:44
2	1	65	20:44	20:55
2	1	67	20:55	21:06
2	1	68	21:06	21:16
2	1	66	21:16	21:27
2	1	64	21:27	21:38
2	1	62	21:38	21:49
2	1	98	21:49	22:03
2	1	58	22:03	22:14
2	1	69	22:14	22:25
2	1	60	22:25	22:35
2	1	96	22:35	22:50
2	1	99	22:50	23:04
2	1	92	23:04	23:19
2	1	93	23:19	23:33
2	1	95	23:33	23:48
2	1	94	23:48	24:03
2	1	97	24:03	24:17

2	1	17	24:17	2:00
4	1	91	3:56	5:53
4	1	90	7:42	9:39
4	1	7	9:39	9:59
4	1	8	9:59	10:20
4	1	89	10:20	12:17
4	1	11	12:17	12:37
4	1	9	12:37	12:58
4	1	10	12:58	13:18
4	1	1	13:18	13:36
4	1	6	13:36	13:54
4	1	2	13:54	14:12
5	1	16	4:06	4:42
5	1	13	8:24	9:00
5	1	12	9:00	9:36
5	1	14	9:36	10:12
6	1	19	3:45	3:58
6	1	18	5:49	6:02
6	1	24	7:31	7:44
8	1	28	2:00	3:10
8	1	19	3:58	5:09
8	1	18	6:02	7:12
8	1	24	7:44	8:54
8	1	30	8:54	10:05
8	1	26	10:05	11:15
8	1	29	11:15	12:25
8	1	27	12:25	13:35
8	1	35	13:35	15:27
9	1	45	2:00	3:58
9	1	19	5:09	6:49
9	1	91	6:49	8:06
9	1	18	8:06	9:47
9	1	24	9:47	11:27
9	1	90	11:27	12:44
9	1	7	12:44	14:00
9	1	12	14:00	14:56
9	1	13	14:56	15:53

9	1	16	15:53	16:49
9	1	28	16:49	18:29
9	1	36	18:29	19:16
9	1	84	19:16	20:32
9	1	86	20:32	21:49
9	1	37	21:49	22:36
9	1	77	22:36	23:52
9	1	80	23:52	1:09
9	1	83	1:09	2:25
9	1	85	2:25	3:42
9	1	75	3:42	4:58
9	1	82	4:58	6:14
9	1	78	6:14	7:31
9	1	81	7:31	8:47
9	1	79	8:47	10:04
9	1	76	10:04	11:20
10	1	45	3:58	4:27
10	1	19	6:49	7:15
10	1	18	9:47	10:13
11	1	38	2:00	2:29
11	1	45	4:27	5:11
11	1	19	7:15	7:59
11	1	91	8:06	8:49
11	1	49	8:49	9:22
11	1	41	9:22	9:52
11	1	39	9:52	10:22
11	1	40	10:22	10:51
12	1	38	2:29	6:00
12	1	19	7:59	15:37
12	1	91	15:37	15:51
12	1	49	15:51	16:01
12	1	41	16:01	19:32
12	1	31	19:32	3:10
12	1	45	3:10	5:08
12	1	42	5:08	8:39
14	1	100	2:00	3:19
14	1	54	3:19	11:59
14	1	46	11:59	21:59

14	1	47	21:59	7:59
14	1	43	7:59	8:44
14	1	87	8:44	9:54
14	1	50	9:54	18:34
14	1	51	18:34	3:13
14	1	52	3:13	11:53
14	1	53	11:53	20:33
15	1	32	2:00	2:53
15	1	100	3:19	3:48
15	1	88	3:48	4:18
15	1	56	4:18	4:42
15	1	48	4:42	5:10
15	1	55	5:10	5:34
16	1	44	2:00	3:48
17	1	32	2:53	11:43
17	1	33	11:43	20:32
18	1	34	2:00	5:18

Le coût de cet horaire = 140821264.000.

Problème 4

FICHIER "PRODUITS"

```
1
2 00
16 113
A0000001 20 18 8
1 0.218
2 0.578
4 0.889
8 4.918
9 4.442
10 0.945
11 1.670
12 0.490
A0000001 20 18 8
1 0.218
2 0.578
4 0.889
8 4.918
9 4.442
10 0.945
11 1.670
12 0.490
A0000001 20 18 8
1 0.218
2 0.578
4 0.889
8 4.918
9 4.442
10 0.945
11 1.670
12 0.490
A0000001 20 18 8
1 0.218
2 0.578
4 0.889
8 4.918
9 4.442
10 0.945
```

```
11 1.670
12 0.490
A0000001 20 18 8
1 0.218
2 0.578
4 0.889
8 4.918
9 4.442
10 0.945
11 1.670
12 0.490
A0000001 20 18 8
1 0.218
2 0.578
4 0.889
8 4.918
9 4.442
10 0.945
11 1.670
12 0.490
A0000002 20 18 6
1 0.376
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
A0000002 20 18 6
1 0.376
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
A0000002 20 18 6
1 0.376
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
A0000002 20 18 6
```

```
1 0.376
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
A0000002 20 18 6
1 0.376
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
A0000003 20 18 7
1 0.264
2 0.487
5 1.810
9 2.802
10 1.342
11 1.558
12 0.461
A0000003 20 18 7
1 0.264
2 0.487
5 1.810
9 2.802
10 1.342
11 1.558
12 0.461
A0000003 20 18 7
1 0.264
2 0.487
5 1.810
9 2.802
10 1.342
11 1.558
12 0.461
A0000003 20 18 7
1 0.264
2 0.487
5 1.810
9 2.802
```

```
10 1.342
11 1.558
12 0.461
A0000003 20 18 7
1 0.264
2 0.487
5 1.810
9 2.802
10 1.342
11 1.558
12 0.461
```

FICHIER "ETAT DU SYSTEME"

```
66 344
A00000004 20 20 8
2 5.132
6 0.650
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
A00000004 20 10 8
2 5.132
6 0.650
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
A00000005 20 20 9
1 0.164
2 5.132
6 0.650
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
A00000005 20 20 9
1 0.164
2 5.132
6 0.650
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
A00000005 20 20 9
```

```
1 0.164
2 5.132
6 0.650
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
A00000005 20 20 9
1 0.164
2 5.132
6 0.650
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
A00000005 20 20 9
1 0.164
2 5.132
6 0.650
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
A00000005 20 20 9
1 0.164
2 5.132
6 0.650
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
A00000005 20 20 8
2 5.132
6 0.650
```

```
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
A00000005 20 20 6
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
A00000005 20 20 6
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
A00000005 20 20 6
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
A00000005 20 20 6
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
A00000005 20 10 6
8 3.512
9 5.039
10 1.298
11 2.162
12 22.923
13 1.402
A00000005 20 10 2
```

```
12 22.923
13 1.402
A000006 20 20 4
8 5.582
9 1.048
10 2.095
11 1.074
A0000006 20 20 3
9 2.332
11 1.485
12 10.547
A0000006 20 20 3
9 2.332
11 1.485
12 10.547
A0000006 20 20 2
11 1.485
12 10.547
A0000006 20 20 2
11 1.485
12 10.547
A0000006 20 20 2
11 1.485
12 10.547
A0000006 20 20 2
11 1.485
12 10.547
A0000006 20 20 1
12 10.547
A0000007 20 15 5
9 5.909
10 1.465
11 2.183
12 5.909
13 1.351
A0000008 20 20 2
11 1.670
12 0.490
A0000009 20 20 5
2 0.536
4 1.024
9 3.824
```

```
11 1.582
12 0.683
A0000009 20 20 5
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
A0000009 20 20 5
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
A0000009 20 20 5
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
A0000009 20 20 5
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
A0000009 20 20 5
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
A0000009 20 20 5
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
A0000009 20 20 5
2 0.536
4 1.024
9 3.824
```

```
11 1.582
12 0.683
A0000009 20 20 5
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
A0000009 20 20 5
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
A0000009 20 20 5
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
A0000009 20 20 5
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
A0000009 20 20 5
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
A0000009 20 20 5
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
A0000009 20 20 5
2 0.536
4 1.024
9 3.824
```

```
11 1.582
12 0.683
A0000009 20 20 5
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
A0000009 20 20 5
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
A0000009 20 20 5
2 0.536
4 1.024
9 3.824
11 1.582
12 0.683
A0000009 20 20 3
9 3.824
11 1.582
12 0.683
A0000009 20 20 3
9 3.824
11 1.582
12 0.683
A0000009 20 20 3
9 3.824
11 1.582
12 0.683
A0000009 20 20 3
9 3.824
11 1.582
12 0.683
A0000009 20 20 3
9 3.824
11 1.582
12 0.683
A0000009 20 20 3
9 3.824
```

```
11 1.582
12 0.683
A0000009 20 20 3
9 3.824
11 1.582
12 0.683
A0000009 20 20 3
9 3.824
11 1.582
12 0.683
A0000009 20 20 3
9 3.824
11 1.582
12 0.683
A0000009 20 20 3
9 3.824
11 1.582
12 0.683
A0000009 20 20 3
9 3.824
11 1.582
12 0.683
A0000009 20 20 3
9 3.824
11 1.582
12 0.683
A0000010 20 20 6
1 0.376
2 0.536
4 5.852
9 3.824
11 2.148
12 0.683
A0000010 20 20 6
1 0.376
2 0.536
4 5.852
9 3.824
11 2.148
12 0.683
A0000010 20 10 6
1 0.376
```

```
2 0.536
4 5.852
9 3.824
11 2.148
12 0.683
A0000011 20 20 8
2 0.728
4 1.253
5 0.829
7 0.192
8 5.036
9 3.032
11 1.495
12 1.015
A0000011 20 20 8
2 0.728
4 1.253
5 0.829
7 0.192
8 5.036
9 3.032
11 1.495
12 1.015
A0000011 20 20 8
2 0.728
4 1.253
5 0.829
7 0.192
8 5.036
9 3.032
11 1.495
12 1.015
A0000011 20 20 8
2 0.728
4 1.253
5 0.829
7 0.192
8 5.036
9 3.032
11 1.495
12 1.015
A0000011 20 20 8
```

```
2 0.728
4 1.253
5 0.829
7 0.192
8 5.036
9 3.032
11 1.495
12 1.015
A0000011 20 20 8
2 0.728
4 1.253
5 0.829
7 0.192
8 5.036
9 3.032
11 1.495
12 1.015
A0000011 20 20 8
2 0.728
4 1.253
5 0.829
7 0.192
8 5.036
9 3.032
11 1.495
12 1.015
A0000011 20 20 8
2 0.728
4 1.253
5 0.829
7 0.192
8 5.036
9 3.032
11 1.495
12 1.015
```

FICHIER "MACHINES"

```
13
MACH1 1 1 0
MACH2 1 1 0
MACH3 1 1 0
MACH4 1 1 0
MACH5 1 1 0
MACH6 1 1 0
MACH7 1 1 0
MACH8 1 0 0
MACH9 1 0 0
MACH10 1 0 0
MACH11 1 0 0
MACH13 1 0 0
MACH14 1 0 0
```

25 états convervés-PROBLÈME 4

MACHINE	NUMPAR	TACHE	DEBUT	FIN
1	1	23	2:00	2:03
1	1	15	2:03	2:08
1	1	4	2:08	2:12
1	1	74	2:12	2:20
1	1	2	2:20	2:24
1	1	1	2:24	2:29
1	1	3	2:29	2:33
1	1	5	2:33	2:37
1	1	6	2:37	2:42
1	1	16	2:42	2:47
1	1	13	2:47	2:52
1	1	12	2:52	2:58
1	1	14	2:58	3:03
1	1	10	3:03	3:10
1	1	9	3:10	3:18
1	1	11	3:18	3:25
1	1	8	3:25	3:33
1	1	7	3:33	3:40
1	1	24	3:40	3:44
1	1	21	3:44	3:47
1	1	19	3:47	3:50
1	1	22	3:50	3:54
1	1	20	3:54	3:57
1	1	72	3:57	4:04
1	1	73	4:04	4:12
2	1	23	2:03	3:45
2	1	74	3:45	3:56
2	1	14	3:56	4:06
2	1	18	4:06	5:49
2	1	20	5:49	7:31
2	1	7	7:31	7:42
2	1	22	7:42	9:25
2	1	73	9:25	9:35
2	1	8	9:35	9:46
2	1	11	9:46	9:57
2	1	72	9:57	10:07

2	1	9	10:07	10:18
2	1	10	10:18	10:29
2	1	19	10:29	12:11
2	1	12	12:11	12:21
2	1	13	12:21	12:31
2	1	21	12:31	14:14
2	1	16	14:14	14:23
2	1	6	14:23	14:35
2	1	5	14:35	14:46
2	1	3	14:46	14:58
2	1	24	14:58	16:41
2	1	1	16:41	16:52
2	1	2	16:52	17:04
2	1	4	17:04	17:15
2	1	15	17:15	17:25
2	1	58	17:25	17:36
2	1	57	17:36	17:47
2	1	56	17:47	17:57
2	1	55	17:57	18:08
2	1	80	18:08	18:23
2	1	44	18:23	18:33
2	1	46	18:33	18:44
2	1	51	18:44	18:55
2	1	25	18:55	20:37
2	1	49	20:37	20:48
2	1	53	20:48	20:59
2	1	52	20:59	21:09
2	1	50	21:09	21:20
2	1	42	21:20	21:31
2	1	47	21:31	21:42
2	1	43	21:42	21:52
2	1	45	21:52	22:03
2	1	54	22:03	22:14
2	1	59	22:14	22:25
2	1	48	22:25	22:35
2	1	76	22:35	22:50
2	1	81	22:50	23:04
2	1	77	23:04	23:19
2	1	82	23:19	23:33
2	1	75	23:33	23:48
2	1	79	23:48	24:03
2	1	78	24:03	24:17

2	1	17	24:17	2:00
4	1	74	3:56	5:53
4	1	7	7:42	8:02
4	1	73	9:35	11:32
4	1	8	11:32	11:53
4	1	11	11:53	12:13
5	1	14	4:06	4:42
6	1	23	3:45	3:58
6	1	18	5:49	6:02
6	1	20	7:31	7:44
6	1	22	9:25	9:38
8	1	29	2:00	3:10
8	1	23	3:58	5:09
8	1	18	6:02	7:12
8	1	20	7:44	8:54
8	1	22	9:38	10:48
8	1	30	10:48	11:58
8	1	26	11:58	13:08
8	1	27	13:08	14:18
8	1	28	14:18	15:29
8	1	32	15:29	17:20
9	1	40	2:00	3:58
9	1	23	5:09	6:49
9	1	74	6:49	8:06
9	1	18	8:06	9:47
9	1	20	9:47	11:27
9	1	7	11:27	12:44
9	1	14	12:44	13:40
9	1	29	13:40	15:21
9	1	34	15:21	16:07
9	1	33	16:07	16:54
9	1	70	16:54	18:11
9	1	65	18:11	19:27
9	1	68	19:27	20:44
9	1	62	20:44	22:00
9	1	66	22:00	23:16
9	1	71	23:16	24:33

9	1	60	24:33	1:49
9	1	64	1:49	3:06
9	1	63	3:06	4:22
9	1	61	4:22	5:39
9	1	67	5:39	6:55
9	1	69	6:55	8:12
10	1	40	3:58	4:27
10	1	23	6:49	7:15
10	1	18	9:47	10:13
11	1	35	2:00	2:29
11	1	40	4:27	5:11
11	1	23	7:15	7:59
11	1	74	8:06	8:49
11	1	41	8:49	9:22
11	1	37	9:22	9:52
11	1	38	9:52	10:22
11	1	36	10:22	10:51
12	1	35	2:29	6:00
12	1	23	7:59	15:37
12	1	74	15:37	15:51
12	1	41	15:51	16:01
12	1	31	16:01	23:39
12	1	37	23:39	3:10
12	1	40	3:10	5:08
12	1	39	5:08	8:39

Le coût de cet horaire = 91988528.000.

50 états convervés-PROBLÈME 4

MACHINE	NUMPAR	TACHE	DEBUT	FIN
1	1	19	2:00	2:03
1	1	15	2:03	2:08
1	1	2	2:08	2:12
1	1	74	2:12	2:20
1	1	5	2:20	2:24
1	1	4	2:24	2:29
1	1	1	2:29	2:33
1	1	3	2:33	2:37
1	1	6	2:37	2:42
1	1	14	2:42	2:47
1	1	12	2:47	2:52
1	1	13	2:52	2:58
1	1	16	2:58	3:03
1	1	11	3:03	3:10
1	1	8	3:10	3:18
1	1	10	3:18	3:25
1	1	9	3:25	3:33
1	1	7	3:33	3:40
1	1	20	3:40	3:44
1	1	22	3:44	3:47
1	1	23	3:47	3:50
1	1	24	3:50	3:54
1	1	21	3:54	3:57
1	1	72	3:57	4:04
1	1	73	4:04	4:12
2	1	19	2:03	3:45
2	1	74	3:45	3:56
2	1	16	3:56	4:06
2	1	18	4:06	5:49
2	1	21	5:49	7:31
2	1	73	7:31	7:42
2	1	7	7:42	7:53
2	1	72	7:53	8:03
2	1	9	8:03	8:14
2	1	24	8:14	9:57

2	1	10	9:57	10:07
2	1	11	10:07	10:18
2	1	8	10:18	10:29
2	1	23	10:29	12:11
2	1	13	12:11	12:21
2	1	12	12:21	12:31
2	1	22	12:31	14:14
2	1	14	14:14	14:23
2	1	6	14:23	14:35
2	1	3	14:35	14:46
2	1	1	14:46	14:58
2	1	20	14:58	16:41
2	1	5	16:41	16:52
2	1	4	16:52	17:04
2	1	2	17:04	17:15
2	1	15	17:15	17:25
2	1	59	17:25	17:36
2	1	58	17:36	17:47
2	1	57	17:47	17:57
2	1	56	17:57	18:08
2	1	55	18:08	18:19
2	1	42	18:19	18:29
2	1	54	18:29	18:40
2	1	43	18:40	18:51
2	1	45	18:51	19:02
2	1	47	19:02	19:12
2	1	49	19:12	19:23
2	1	51	19:23	19:34
2	1	52	19:34	19:44
2	1	50	19:44	19:55
2	1	77	19:55	20:10
2	1	48	20:10	20:20
2	1	44	20:20	20:31
2	1	46	20:31	20:42
2	1	53	20:42	20:53
2	1	78	20:53	21:07
2	1	81	21:07	21:22
2	1	82	21:22	21:36
2	1	80	21:36	21:51
2	1	75	21:51	22:05
2	1	76	22:05	22:20
2	1	79	22:20	22:35

2	1	25	22:35	24:17
2	1	17	24:17	2:00
4	1	74	3:56	5:53
4	1	73	7:42	9:39
4	1	7	9:39	9:59
4	1	72	9:59	11:56
4	1	9	11:56	12:17
5	1	16	4:06	4:42
6	1	19	3:45	3:58
6	1	18	5:49	6:02
6	1	21	7:31	7:44
6	1	24	9:57	10:10
8	1	29	2:00	3:10
8	1	19	3:58	5:09
8	1	18	6:02	7:12
8	1	21	7:44	8:54
8	1	30	8:54	10:05
8	1	27	10:05	11:15
8	1	28	11:15	12:25
8	1	26	12:25	13:35
8	1	32	13:35	15:27
9	1	40	2:00	3:58
9	1	19	5:09	6:49
9	1	74	6:49	8:06
9	1	18	8:06	9:47
9	1	21	9:47	11:27
9	1	73	11:27	12:44
9	1	7	12:44	14:00
9	1	16	14:00	14:56
9	1	29	14:56	16:37
9	1	33	16:37	17:24
9	1	70	17:24	18:40
9	1	69	18:40	19:57
9	1	60	19:57	21:13
9	1	62	21:13	22:30
9	1	34	22:30	23:16
9	1	65	23:16	24:33

9	1	61	24:33	1:49
9	1	66	1:49	3:06
9	1	67	3:06	4:22
9	1	63	4:22	5:39
9	1	64	5:39	6:55
9	1	68	6:55	8:12
9	1	71	8:12	9:28
10	1	40	3:58	4:27
10	1	19	6:49	7:15
10	1	18	9:47	10:13
11	1	37	2:00	2:29
11	1	40	4:27	5:11
11	1	19	7:15	7:59
11	1	74	8:06	8:49
11	1	41	8:49	9:22
11	1	36	9:22	9:52
11	1	35	9:52	10:22
11	1	38	10:22	10:51
12	1	37	2:29	6:00
12	1	19	7:59	15:37
12	1	74	15:37	15:51
12	1	41	15:51	16:01
12	1	36	16:01	19:32
12	1	31	19:32	3:10
12	1	40	3:10	5:08
12	1	39	5:08	8:39

Le coût de cet horaire = 91572304.000.