RAIRO. RECHERCHE OPÉRATIONNELLE

C. CAUX
G. FLEURY
M. GOURGAND
P. KELLERT

Couplage méthodes d'ordonnancement-simulation pour l'ordonnancement de systèmes industriels de traitement de surface

RAIRO. Recherche opérationnelle, tome 29, n° 4 (1995), p. 391-413

<http://www.numdam.org/item?id=RO_1995__29_4_391_0>

Recherche opérationnelle/Operations Research
(vol. 29, n° 4, 1995, p. 391 à 413)

COUPLAGE MÉTHODES D'ORDONNANCEMENT – SIMULATION POUR L'ORDONNANCEMENT DE SYSTÈMES INDUSTRIELS DE TRAITEMENT DE SURFACE (*)

par C. CAUX ([1]), G. FLEURY ([1]), M. GOURGAND ([1]) et P. KELLERT ([1])

Communiqué par Philippe CHRÉTIENNE

Résumé. – *Cet article traite du problème de la détermination des dates d'introduction des pièces dans un système de traitement de surfaces avec, pour objectif, le traitement d'un maximum de pièces dans une période donnée. Comme la fonction objectif n'est pas aisément connue analytiquement, nous avons réalisé un couplage de méthodes d'ordonnancement avec un simulateur à événements discrets. Les méthodes d'ordonnancement sont une heuristique que nous proposons et l'algorithme de descente stochastique. La simulation déterministe permet d'évaluer l'ordonnancement proposé par l'une des méthodes précédentes. Bien que le problème traité soit NP-difficile, nous déterminons des ordonnancements satisfaisants en quelques minutes de calcul. Nous comparons ensuite l'efficacité des méthodes d'ordonnancement choisies. Enfin, les contraintes de transport étant explicitement prises en compte, l'intégration de couplages dans le système de contrôle/commande de l'atelier est envisagée pour l'aide à la planification d'une part et l'aide au pilotage d'autre part, lorsque le système est soumis à des événements perturbants.*

Mots clés : Traitement de surface, ordonnancement planifié, heuristique, descente stochastique, simulation à événements discrets.

Abstract. – *This paper deals with the entry date computation of incoming components in a surface treatment plant. The aim is to increase the number of components that are treated during a given time. Because the objective function is not easily analytically known, we carry out a scheduling method-discrete event simulation coupling. The used methods are a specific heuristic we propose and the iterative improvement algorithm. These methods provide us with schedules that are evaluated using deterministic simulation. Although the scheduling problem is NP-hard, acceptable schedules are computed in a few minutes. The efficiency of the scheduling methods is then compared. Finally, the transport constraints having been considered, we plan to integrate coupled methods in the plant control system for both the piloting aid and the planning aid when unexpected events occur.*

Keywords: Hoist scheduling, off line scheduling, heuristic, stochastic descent, discrete event simulation.

(*) Reçu en octobre 1992.

([1]) Laboratoire d'informatique, Université Blaise Pascal, Clermont-Ferrand II, 63177 Aubière Cedex, France.

Recherche opérationnelle/Operations Research, 0399-0559/95/04/$ 4.00

1. INTRODUCTION

La productivité de certains systèmes de production dépend essentiellement des dates d'introduction des pièces dans le système, c'est-à-dire de l'ordonnancement des pièces en entrée. Si l'énoncé d'un problème d'ordonnancement est généralement simple à formuler, sa résolution ne l'est pas forcément, comme le prouvent les nombreux travaux de recherche effectués dans ce domaine depuis plusieurs années [GOTHA93].

Il convient de distinguer d'emblée deux types de problèmes d'ordonnancement : le pilotage temps réel et l'ordonnancement planifié. Le pilotage temps réel consiste à ordonnancer l'introduction des pièces dans le système au fur et à mesure où les opérations sont réalisées et en tenant compte du nombre de ressources disponibles et de l'état du système. Son inconvénient est que les règles activées sont « myopes », c'est-à-dire qu'elles ne savent pas appréhender les conséquences des décisions prises [Bel, 91]. L'ordonnancement planifié consiste à affecter, *a priori*, une date de début à chaque opération. Il est donc nécessaire de connaître les caractéristiques du système et la production à réaliser sur une période donnée. Cette technique permet de considérer globalement l'ensemble des contraintes et donc de mesurer les conséquences des décisions prises. Le principal inconvénient de l'ordonnancement planifié est que le temps pour générer un « bon » ordonnancement est souvent prohibitif par rapport au temps dont on dispose. D'autre part, si durant son fonctionnement, le système subit des événements perturbants (panne d'une machine, rupture d'un stock...), l'ordonnancement prévu n'est plus forcément réalisable.

Dans cet article, nous nous intéressons à la détermination d'ordonnancements planifiés en entrée d'un système de production à flux discrets dans le but de minimiser la durée totale d'exécution d'un ensemble de traitements. Après avoir présenté l'atelier de traitement de surfaces étudié, nous montrons qu'il peut être judicieux, voire indispensable, de réaliser un couplage méthode d'ordonnancement-simulation à événements discrets pour résoudre ce problème. Des méthodes d'ordonnancement sont proposées, en particulier une heuristique nommée SESF (Smallest Earliest Start First). La crédibilité des résultats obtenus avec un tel couplage dépend, bien sûr, de la qualité du modèle de simulation. Après avoir comparé l'efficacité des méthodes d'ordonnancement choisies (qualité de l'ordonnancement, temps de calcul), nous montrons qu'il est opportun d'intégrer des méthodes d'ordonnancement dans le système de contrôle/commande de l'atelier, en

particulier pour proposer un nouvel ordonnancement lorsque le système est perturbé par des aléas.

2. LE SYSTÈME ÉTUDIÉ : UN ATELIER DE TRAITEMENT DE SURFACES

Cette étude a été réalisée en collaboration avec EDF Sud-Ingénierie. L'atelier étudié permet de décaper des pièces par immersions successives dans des bains de produits nettoyants. L'objectif est d'écouler la charge journalière prévue pour cet atelier, si possible en moins de vingt-quatre heures, sans aucune modification de l'atelier. L'atelier est constitué de trois lignes parallèles. Chaque ligne est composée de quinze cuves de traitement. Les trois lignes sont desservies par un convoyeur principal mono-directionnel à rouleaux, comme le montre la figure 1.

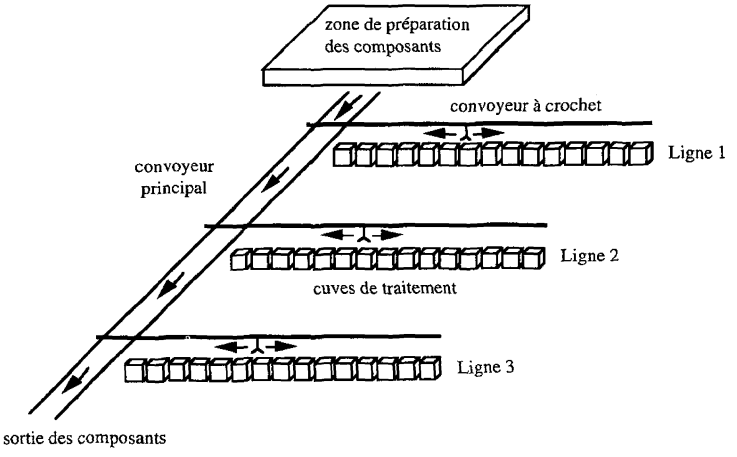

Figure 1. – Topologie de l'atelier.

Sur chaque ligne, un convoyeur à crochet assure d'une part, le déplacement des pièces entre le convoyeur principal et les cuves, et, d'autre part, le déplacement des pièces de cuve à cuve. Le convoyeur à crochet se déplace parallèlement à la ligne de cuves et peut aussi descendre ou monter pour immerger ou sortir une pièce d'une cuve. Dans cette étude, la topologie est imposée et aucune zone de stockage n'est prévue ni envisageable.

L'atelier peut traiter vingt-quatre types de pièces. À chaque type de pièce est associée une gamme de routage (ou traitement) qui spécifie la liste ordonnée des bains que doit subir une pièce, ainsi que les temps d'immersion associés à chaque bain. Une pièce sur quatre visite deux lignes pour son

traitement. Les autres pièces sont traitées en totalité sur une seule ligne. La figure 2 donne un exemple de traitement dans lequel les temps minimaux sont mentionnés.

étape 1 :	bain n° C21	20 minutes
étape 2 :	bain n° C210	37,5 minutes
étape 3 :	bain n° C23	7,5 minutes
étape 4 :	bain n° C31	10 minutes
étape 5 :	bain n° C312	7,5 minutes
étape 6 :	bain n° C37	10 minutes

Figure 2. – Exemple de traitement tiré de l'annexe A-1 (pièces de type 3A).

La charge journalière de l'atelier est de seize pièces. Puisque certains traitements peuvent durer jusqu'à quatre heures, il est nécessaire de gérer plusieurs pièces simultanément dans une même ligne, tout en garantissant le bon fonctionnement de l'atelier. Pour cela, il faut respecter les règles suivantes :

a) Un convoyeur à crochet ne peut transporter qu'une pièce à la fois et il n'existe pas de zone de stockage intermédiaire. Une cuve ne peut contenir qu'une pièce. Des interblocages peuvent donc survenir au niveau de chaque ligne. On distingue deux types d'interblocage :

– une pièce résidant dans une cuve i demande à accéder à une cuve j occupée ($i \neq j$ et $1 \leq i, j \leq 15$) ;

– une pièce résidant dans une cuve i demande à accéder à une cuve j occupée et inversement la pièce occupant la cuve j demande à accéder à la cuve i ($i \neq j$ et $1 \leq i, j \leq 15$).

b) Certaines cuves contiennent des produits corrosifs. Dans ce cas, la durée d'immersion d'une pièce ne doit pas excéder une certaine tolérance spécifiée dans la gamme.

L'introduction d'une nouvelle pièce dans l'atelier ne doit pas induire, par la suite, de perturbations. Il est donc nécessaire de déterminer, pour un lot de pièces donné, les dates d'introduction des pièces à traiter de telle sorte qu'aucun blocage ne survienne et que les durées d'immersion soient respectées tout en réalisant ce traitement dans le temps le plus court possible.

La politique actuelle de gestion de l'atelier consiste à n'autoriser qu'une seule pièce par ligne à un instant donné. Les problèmes de blocages sont alors évités. Lorsque l'atelier est vide, une pièce est affectée à chaque ligne. Ensuite, chaque fois qu'une pièce quitte l'atelier, une pièce, susceptible d'être traitée sur la ligne qui vient d'être libérée, est admise dans le lot de

pièces. Cette méthode très simple permet évidemment d'éviter les problèmes d'interblocage mais elle n'est pas efficace car trop pénalisante en temps.

Le problème d'ordonnancement à résoudre est un problème d'optimisation NP-difficile. En effet, supposons qu'il existe un algorithme de complexité polynomiale qui résolve le problème de décision correspondant à ce problème d'optimisation. On sait alors résoudre le problème pour une seule ligne de l'atelier : il suffit pour cela de ne considérer que des traitements qui sont exécutés sur cette ligne et de durée constante pour chaque cuve et pour chaque pièce. On sait aussi résoudre le problème dans le cas particulier où les durées de transport sont nulles. Ordonnancer les pièces en entrée d'une ligne de l'atelier avec des durées de transport nulles est identique à ordonnancer les pièces en entrée d'un atelier de type job-shop sans temps d'attente (no wait job shop). Or, ce dernier est connu pour être NP-complet au sens fort dès que le nombre de machines est supérieur ou égal à 2 [Garey, 79].

Ce problème se retrouve dans de nombreux ateliers. La résolution que nous proposons prend en compte explicitement les transports et les ressources non consommables dans un job shop.

3. CHOIX ET DESCRIPTION DU MODÈLE

La qualité d'un ordonnancement est mesurée par une fonction objectif que l'on sait ou non calculer analytiquement. Ce peut être, par exemple, le coût total de l'exécution des tâches ou encore la durée totale nommée makespan (différence entre la date de sortie de la dernière pièce du système et la date d'entrée de la première pièce dans ce même système). En ce qui concerne notre système, il existe au moins deux formalisations pour le problème du calcul de la fonction objectif pour les dates d'entrée fixes :

– un ensemble de contraintes exprimées sous forme mathématique à l'aide d'opérateurs de type min et max [Fleury, 93] ;

– un modèle de simulation à événements discrets.

La première formalisation est limitée, pour des raisons de concision, à une seule ligne. Considérons J pièces à ordonnancer à L lieux, chaque lieu étant soit une cuve, soit l'entrée dans la ligne, soit la sortie de la ligne. Chaque pièce j, $1 \leq j \leq J$, est caractérisée par un ensemble de données :

– L_j, $1 \leq j \leq J$: le nombre d'opérations que doit subir la pièce j ;

– $t_{j,k}$, $1 \leq j \leq J$, $0 \leq k \leq L_j$: la durée minimum de l'opération de rang k que doit subir la pièce j (avec la convention $t_{j,0} = 0$) ;

– $f_{j,k}$, $1 \leq j \leq J$, $1 \leq k \leq L_j$: la durée maximum de l'opération de rang k que doit subir la pièce j ;

– $m_{j,k}$, $1 \leq j \leq J$, $0 \leq k \leq L_j + 1$: le lieu où la pièce j va subir l'opération de rang k avec les conventions : $m_{j,0}$ est l'entrée de la ligne et $m_{j,L_{j+1}}$ est la sortie de la ligne.

et un ensemble de variables :

– $d_{j,k}$, $1 \leq j \leq J$, $0 \leq k \leq L_{j+1}$: la date d'entrée de la pièce j pour subir l'opération de rang k ;

– $s_{j,k}$, $1 \leq j \leq J$, $0 \leq k \leq L_j$: la date de sortie de la pièce j ayant subi l'opération de rang k, (avec la convention $d_{j,0} = s_{j,0} = 0$).

Soient u et v deux lieux, on note :

– $N(u\,;\,v) \geq 0$: la durée de déplacement du crochet en charge entre u et v ;

– $M(u\,;\,v) \geq 0$: la durée de déplacement du crochet à vide entre u et v.

Avec ce formalisme, ce problème peut être exprimé à l'aide de cinq contraintes qui sont :

1) La durée de traitement de l'opération de rang k pour la pièce j doit vérifier :

$$t_{j,k} \leq s_{j,k} - d_{j,k} \leq f_{j,k}, \qquad 1 \leq j \leq J, \quad 1 \leq k \leq L_j.$$

2) La date d'entrée de la pièce j pour l'opération de rang $(k+1)$ est donnée par la date de sortie de la pièce j ayant subi l'opération de rang k (supposée déterminée) augmentée de la durée de déplacement en charge du crochet. En d'autres termes, deux opérations successives pour une pièce j donnée sont obligatoirement séparées par une phase de transport, c'est-à dire :

$$d_{j,k+1} = s_{j,k} + N(m_{j,k}\,;\,m_{j,k+1}), \qquad 1 \leq j \leq J, \quad 0 \leq k \leq L_j.$$

3) La troisième contrainte exprime qu'une machine ne supporte au plus qu'une seule pièce. Si deux pièces i_1 et i_2, $1 \leq i_1 \leq J$, $1 \leq i_2 \leq J$, $i_1 \neq i_2$, doivent accéder à la même machine pour subir leur opération de rang k_1 et k_2 respectivement, c'est-à-dire si $m_{i_1,k_1} = m_{i_2,k_2}$, il faut que :

$$[d_{i_1,k_1}\,;\,s_{i_1,k_1}[\,\cap [d_{i_2,k_2}\,;\,s_{i_2,k_2}[\,= \varnothing, \qquad 1 \leq k_1 \leq L_{i_1}, \quad 1 \leq k_2 \leq L_{i_2}.$$

4) La quatrième contrainte exprime le fait que le crochet, après son dernier dépôt de pièce (à la date $d_{j,p}$), arrive à temps pour retirer la prochaine pièce qui a terminé une opération. C'est-à dire, si α et β, $1 \leq \alpha \leq J$, $0 \leq \beta \leq L_\alpha$ vérifient :

$$d_{\alpha,\beta} + t_{\alpha,\beta} = \mathrm{Min}\,(d_{i,k} + t_{i,k} \,|\, d_{i,k} + t_{i,k} \geq d_{j,p},\ 1 \leq i \leq J,\ 0 \leq k \leq L_i)$$

alors on doit avoir :

$$s_{\alpha,\beta} - d_{j,p} \geq M\,(m_{j,p};\ m_{\alpha,\beta})$$

5) La dernière contrainte exprime que le crochet de transport est une ressource critique, d'où :

$$\forall\,(i_1,\ k_1),\ (i_2,\ k_2),\ i_1 \neq i_2,$$
$$1 \leq i_1,\ i_2 \leq J, \quad 0 \leq k_1 \leq L_{i_1}, \quad 0 \leq k_2 \leq L_{i_2}$$

on a :

$$[s_{i_1,k_1};\ d_{i_1,k_1+1}[\ \cap\ [s_{i_2,k_2};\ d_{i_2,k_2+1}[= \varnothing$$

Cette solution est dédiée, difficilement paramétrable, et toute modification du système impose une nouvelle formalisation.

La deuxième formalisation, contrairement à la première, permet la validation du modèle par l'utilisateur (par exemple, à l'aide d'outils d'animation graphique) et est beaucoup plus facile à maintenir. Elle offre, de plus, une très grande flexibilité (modification de la topologie du système, des règles de fonctionnement...) et permet la prise en compte d'un fonctionnement dégradé et autorise d'autres utilisations (évaluations de critères de performance, aide au pilotage...).

Le principe du couplage « méthodes d'ordonnancement-simulation du système » est le suivant. Pour un lot de pièces donné, une méthode d'ordonnancement propose un ordonnancement dont on ne retient que la suite des dates d'introduction de chaque pièce dans l'atelier. Une simulation du fonctionnement du système est réalisée avec, comme donnée d'entrée, l'ordonnancement proposé. Elle retourne à la méthode d'ordonnancement, la valeur correspondante de la fonction objectif. Ce processus est itéré autant de fois que le requiert la méthode d'ordonnancement. La figure 3 résume le couplage méthode d'ordonnancement – simulation. De telles démarches ont déjà été mises en œuvre avec succès ([Manz, 89] ; [Ferrara, 90] ; [Barnichon, 92]).

Un modèle de simulation de l'atelier réel a été construit et validé, non seulement dans le but d'ordonnancer l'atelier, mais encore pour tester diverses politiques de gestion. Son paramétrage permet de simuler aisément d'autres ateliers. Le langage de simulation utilisé est le logiciel QNAP2 (Queueing Network Analysis Package) [Simulog, 91]. Ce modèle de simulation a été soigneusement validé. Dans le cas d'un système traitant plusieurs types de pièces, un premier type de test consiste à étudier les résultats de la simulation du modèle lorsque les pièces en entrée de l'atelier sont toutes de même type. Seuls les moyens physiques nécessaires pour traiter ce type de pièce doivent

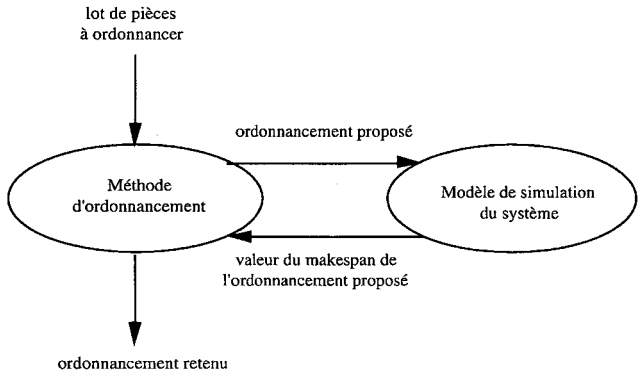

Figure 3. – Couplage méthode d'ordonnancement – modèle de simulation.

être sollicités. Un deuxième type de test est réalisé en ajoutant au modèle des tests d'incohérence, par exemple le passage d'une machine à un autre sans utilisation d'un moyen de transport, dans le cas d'un système de production. Ce modèle est alors simulé avec de nombreux jeux d'essai générés en fonction des fréquences d'apparition de chacun des types de pièces.

4. LES MÉTHODES D'ORDONNANCEMENT

4.1. Les méthodes existantes

Les méthodes d'ordonnancement applicables à un problème dépendent de la complexité de celui-ci. Les problèmes NP-difficiles nécessitent la mise au point de méthodes de traitement adaptées. On peut citer par exemple, la programmation sous contraintes, les systèmes experts, les heuristiques et les méthodes stochastiques.

La *programmation sous contraintes* et les méthodes de *séparation et d'évaluation* (ou Branch and Bound) consistent à interpréter l'ensemble des solutions réalisables au moyen d'une arborescence. L'efficacité de telles méthodes repose sur la puissance de la fonction d'élagage. Dans le pire des cas, ces méthodes énumèrent toutes les solutions possibles.

Les *systèmes experts* peuvent apporter des solutions dans le cas où la connaissance concernant la conception d'ordonnancements peut être formalisée. Cette connaissance est généralement difficile à synthétiser sous forme de règles, soit parce qu'elle provient de l'expérience humaine, soit parce que le problème est complexe.

Les *heuristiques* sont des règles, des principes ou des méthodes permettant de déterminer pour plusieurs lignes de conduite, celle qui promet d'être la plus efficace pour atteindre un but. Les solutions qu'elles fournissent n'ont aucune garantie d'être les meilleures mais l'expérience montre qu'elles sont en général acceptables et obtenues avec un faible temps de calcul.

Ces trois types de méthodes ont en commun d'être très spécifiques du problème étudié.

Les *méthodes stochastiques* comprennent, entre autres, l'algorithme du recuit simulé et ses variantes. Cet algorithme s'appuie sur des principes de physique statistique pour trouver des solutions approchées à des problèmes complexes [Kirkpatrick, 83]. Le recuit simulé est applicable à la résolution de problèmes d'optimisation dans lesquels de nombreuses solutions sont envisageables.

Nous explicitons, dans ce qui suit, les méthodes que nous avons conçues pour résoudre le problème d'ordonnancement.

4.2. L'heuristique SESF (Smallest Earliest Start First)

Cette heuristique permet d'ordonnancer des pièces en entrée d'un système telles que des pièces dans un atelier, des jobs dans un système informatique... Son principe est d'envoyer les pièces le plus tôt possible dans le système afin de minimiser le makespan.

On considère un lot de n pièces à ordonnancer. L'heuristique, selon une méthode gloutonne, bâtit alors progressivement un ordonnancement partiel de la façon suivante. Une première pièce est choisie arbitrairement et sa date d'introduction est fixée à 0. Si i dates d'entrée de pièces sont connues, pour chacune des $(n - i)$ restantes, $1 \leq i \leq n - 1$, une date d'introduction au plus tôt est calculée par essais successifs. Pour cela, l'heuristique suppose d'abord que la pièce candidate est introduite en même temps que la pièce précédente. Tant que le système ne peut pas traiter cet ordonnancement partiel (conflit de ressource, blocage...), l'heuristique incrémente la date d'introduction. La pièce ayant la plus petite date d'introduction est alors ajoutée à l'ordonnancement partiel. Ce processus est itéré jusqu'à ce que toutes les pièces soient ordonnancées.

La complexité temporelle de l'heuristique peut être évaluée par le nombre d'appels à la simulation et l'encombrement mémoire est représenté par le nombre de pièces à ordonnancer. L'heuristique calcule la date au plus tôt de n pièces lors de la première étape, de $n - 1$ pièces lors de la deuxième étape... Pour chaque calcul de la date au plus tôt d'une pièce, le nombre

d'appels à la simulation est borné par la durée $d_{\max}$ de la plus longue des gammes. En effet, dans le pire des cas, une pièce pourra être introduite dans l'atelier lorsque la précédente sera sortie. Le nombre d'appels à la simulation est donc majoré par :

$$d_{\max}\,(n + (n-1) + \ldots + 1) \quad \text{soit } d_{\max}\, n\,(n+1)/2$$

La complexité temporelle de l'heuristique est $O\,(d_{\max}\, n^2)$.

La description précédente donne le fonctionnement de l'heuristique dans le cas général. Il est possible d'améliorer son efficacité en l'adaptant au problème traité. Cette heuristique donne des résultats rapidement et il est possible de calculer une borne supérieure du nombre d'itérations. En effet, cette heuristique trouvera toujours une date au plus tôt pour introduire une pièce : il suffit d'introduire la pièce lorsque toutes les pièces précédentes ont quitté le système. Les modifications consistent à utiliser au mieux le parallélisme de l'atelier (pour améliorer le makespan) et à tenir compte des types de pièces à traiter (pour améliorer la vitesse d'exécution).

On considère un lot initial de pièces à ordonnancer (ici de l'ordre de 16). L'heuristique construit progressivement un ordonnancement de ces pièces de la façon suivante :

Étape 1 : Initialisation

• *Choisir 3 pièces dans le lot initial telles que la première pièce soit traitée sur la première ligne, la deuxième sur la deuxième ligne et la troisième sur la troisième ligne (ces 3 pièces peuvent être traitées en parallèle).*

• *Fixer les dates d'introduction de la première pièce (resp. deuxième, troisième) à 0 (resp. 5, 10) : on suppose que le délai minimum entre deux pièces est de 5 minutes.*

• $k := 4$. *($k-1$ pièces ont déjà été ordonnancées)*

Étape 2 : Calcul des dates au plus tôt des pièces candidates à la k-ième place de l'ordonnancement

• *Choisir comme k-ième pièce de l'ordonnancement, une pièce non déjà choisie dans le lot initial.*

• *Initialiser la date d'introduction de cette k-ième pièce à la date d'introduction de la pièce précédente augmentée de 5 minutes.*

• *Évaluer par simulation le makespan de cet ordonnancement partiel.*

• *Tant que le makespan est infini (blocage) alors*

incrémenter la date d'introduction de la k-ième pièce de 1 minute,

évaluer par simulation le makespan de cet ordonnancement partiel.

• *Répéter cette étape pour toutes les pièces du lot initial non déjà choisies.*

Étape 3 : Choix de la k-ième pièce de l'ordonnancement

• *Retenir comme k-ième pièce de l'ordonnancement la pièce donnant la plus petite date d'introduction.*

Étape 4 :

• $k := k + 1$.

• *Répéter depuis l'étape 2 jusqu'à* $k = n + 1$.

Deux modifications évidentes peuvent être apportées à l'étape 2 de l'heuristique afin de diminuer le nombre d'appels à la simulation : d'une part, mémoriser les types de pièces qui ont été traitées et, d'autre part, mémoriser la date au plus tôt minimale.

4.3. La descente stochastique

Le recuit simulé est applicable à des problèmes de grande combinatoire. Il permet, à partir d'une solution initiale et en générant des solutions voisines, de trouver la solution optimale à un problème combinatoire sous certaines conditions. La grande combinatoire des problèmes étudiés et les limitations en temps de calcul nous ont amenés à utiliser un algorithme stochastique. Comme il n'est pas possible d'appliquer un algorithme de type recuit simulé convergent puisque le makespan d'un ordonnancement provoquant un blocage du convoyeur à crochet est infini [Fleury, 93], nous nous sommes restreints à un algorithme de descente stochastique :

Étape 1 :

Générer un premier ordonnancement courant OC et évaluer par simulation le makespan de cet ordonnancement.

Étape 2 :

Générer un ordonnancement voisin OV à partir de OC et évaluer par simulation le makespan de cet ordonnancement.

Étape 3 :

Si le makespan de OV est inférieur ou égal au makespan de OC, l'ordonnancement OV devient l'ordonnancement courant OC.

Étape 4 :

Répéter depuis l'étape 2 jusqu'à ce que le nombre d'itérations soit suffisant ou le résultat convenable.

Le point le plus délicat lors de l'utilisation de ce type d'algorithme est la définition du voisinage d'une solution. Lors de l'ordonnancement d'atelier de type flow-shop, les méthodes classiques de génération de voisins sont les

insertions et les permutations de pièces à ordonnancer [Fleury, 91]. Nous utilisons ici des techniques similaires et proposons la mise en œuvre de deux variantes de génération de voisins.

La première variante consiste à choisir aléatoirement une pièce dans l'ordonnancement courant et à lui assigner une date d'introduction choisie aléatoirement entre 0 et 1 440 minutes (la pièce est aléatoirement placée dans la période de 24 heures). Les autres dates d'introduction ne sont pas modifiées. Cette technique nécessite une suppression suivie d'une insertion.

La deuxième variante consiste à choisir des pièces plutôt vers la fin de l'ordonnancement (celles dont la date d'introduction est élevée). Ce choix est effectué en choisissant les pièces selon une loi géométrique. La pièce choisie reçoit une date d'introduction aléatoirement comprise entre 0 et sa date d'introduction. Les autres dates d'introduction ne sont pas modifiées. Cette variante a pour but de « tasser » l'ordonnancement en ramenant les pièces en fin d'ordonnancement vers le début.

Nous utilisons dans l'algorithme de descente stochastique ces deux variantes : la première avec la probabilité 1/3 et la deuxième avec la probabilité 2/3.

Il est préférable que la solution initiale ne provoque pas de blocage pour que la descente puisse être activée. Pour cela, dans l'ordonnancement initial, les dates d'introduction sont telles qu'il n'y a qu'une seule pièce dans l'atelier à un instant donné. Tous les problèmes de blocage sont donc évités.

4.4. Méthode hybride

Cette méthode est conçue à partir du mécanisme de génération de voisins d'une descente stochastique déterminant l'ordre d'introduction des pièces (problème de permutation) au lieu d'utiliser les dates d'introduction, et du calcul de leurs dates effectives d'entrée au plus tôt utilisé dans l'heuristique.

L'algorithme utilisé est alors le suivant :

Étape 1 :

Générer un premier ordonnancement courant OC (comme dans le cas de la descente stochastique) et initialiser le makespan record avec le makespan de cet ordonnancement (calculé par simulation).

Étape 2 :

Générer un ordonnancement voisin OV à partir de OC par déplacement relatif d'une entrée de pièce. Pour cela, choisir i et j ($i \neq j$, i et j aléatoirement choisis entre 1 et la taille du lot de pièces) et insérer la i-ième pièce à la position j.

Étape 3 :

Recalculer par simulation les dates au plus tôt de l'ordonnancement OV à partir de la pièce $\min(i, j)$.

Étape 4 :

Si le makespan de OV est inférieur ou égal au makespan record, l'ordonnancement OV devient l'ordonnancement courant OC et le makespan record prend la valeur du makespan de OV.

Étape 5 :

Répéter depuis l'étape 2 jusqu'à ce que le nombre d'itérations soit suffisant ou le résultat convenable.

Cette méthode confère au programme de simulation un rôle actif puisqu'il doit recalculer les dates au plus tôt des pièces de l'ordonnancement (étape 3). Ce calcul des dates au plus tôt permet de « tasser » l'ordonnancement obtenu à chaque itération.

5. RÉSULTATS OBTENUS

Les ordonnancements sont obtenus par les méthodes présentées précédemment. L'indépendance de ces méthodes permet de les valider mutuellement. En effet, lorsque les différentes méthodes fournissent des makespans peu différents, pour un même lot de pièces à ordonnancer, on peut considérer que la solution trouvée est « convenable ». Il faut mentionner que, pour un lot de pièces donné, la valeur optimale du makespan est inconnue et qu'aucune information n'indique si la valeur trouvée est proche de l'optimum.

Le tableau de la figure 4 donne les makespans obtenus (en minutes) selon les trois méthodes présentées précédemment, pour 50 jeux d'essai. Ces jeux d'essai sont générés aléatoirement en fonction de l'histogramme donnant le nombre de pièces annuel à traiter pour chaque type de pièce. Les makespans obtenus sont comparés à ceux obtenus par la politique de gestion actuelle de l'atelier. Les différentes méthodes sont mises en œuvre sur une station de travail UNIX équipée d'un processeur 68040 cadencé à 25 MHz. Sur une telle machine, le temps nécessaire pour évaluer par simulation un ordonnancement d'un lot candidat est d'environ 0,4 seconde. Il est donc clair que l'essentiel du temps pour déterminer un ordonnancement satisfaisant est consacré à la simulation.

La tendance observée s'est avérée être la même avec d'autres jeux d'essais. Les données correspondant à ces résultats ne peuvent être reproduites *in extenso*, en raison de leur volume. Nous donnons, en annexe A-4, les

Méthode Temps U.C.	Politique 1"	Heuristique 15 min.	Descente Stochastique 35 min. 5 000 it.	Descente Stochastique 140 min. 20 000 it.	Méthode hybride 80 min. 20 it.	Méthode hybride 160 min. 40 it.
1	2 968	1 850	2 282	2 014	1 950	**1 705**
2	2 172	1 436	1 633	1 439	**1 428**	1 431
3	3 314	2 230	2 848	2 681	2 227	**2 223**
4	2 669	1 850	2 077	1 921	1 798	**1 704**
5	1 868	1 429	1 646	**1 425**	**1 425**	**1 425**
6	2 367	1 796	2 006	2 040	1 794	**1 789**
7	1 830	1 429	1 639	1 432	1 433	**1 423**
8	1 222	**904**	1 052	997	934	**904**
9	1 873	1 430	1 702	1 534	**1 425**	**1 425**
10	2 363	1 796	1 817	2 016	**1 789**	**1 789**
11	1 592	895	1 003	**803**	985	990
12	2 366	1 436	1 751	1 615	**1 213**	1 292
13	3 724	2 644	2 992	2 865	**2 484**	2 491
14	1 464	1 164	1 360	**1 159**	**1 159**	1 164
15	3 459	2 179	2 704	2 279	**1 966**	**1 966**
16	1 527	1 057	1 008	999	1 024	**896**
17	1 507	1 165	1 168	1 265	1 165	**1 160**
18	1 527	1 057	1 028	**900**	1 155	903
19	1 323	**899**	1 023	993	1 112	966
20	1 514	1 165	1 273	1 302	1 219	**1 160**
21	2 625	**1 850**	2 092	1 954	1 948	1 859
22	1 916	1 432	1 627	1 727	1 427	**1 426**
23	2 259	1 439	1 685	1 727	1 437	**1 432**
24	1 851	1 172	1 466	1 172	1 174	**1 166**
25	1 165	**643**	804	711	836	748
26	1 979	**1 322**	1 508	1 416	1 332	1 325
27	1 911	1 431	1 683	1 428	1 432	**1 426**
28	2 259	1 537	1 677	**1 430**	1 595	1 525
29	2 593	1 642	1 875	1 638	**1 547**	**1 547**
30	1 873	1 430	1 631	1 529	1 430	**1 425**
31	2 605	1 962	2 294	2 062	**1 954**	**1 954**
32	3 199	1 729	1 864	1 810	**1 606**	1 661
33	2 562	1 957	2 297	2 301	**1 952**	**1 952**
34	2 236	1 693	1 795	1 802	**1 689**	1 690
35	1 954	1 433	1 676	1 574	1 463	**1 428**
36	2 429	1 352	1 481	1 435	1 440	**1 347**
37	2 302	1 586	1 760	1 535	1 433	**1 428**
38	1 199	**749**	768	**749**	822	772
39	2 282	1 697	1 960	2 026	**1 692**	**1 692**
40	1 611	**747**	867	780	869	846
41	1 830	1 428	1 445	1 534	1 428	**1 423**
42	2 254	1 534	1 838	1 680	1 441	**1 430**
43	3 479	1 915	2 430	2 107	1 844	**1 825**
44	2 732	1 700	1 851	1 791	1 842	**1 687**
45	1 570	1 057	1 039	1 000	910	**909**
46	2 346	1 586	1 750	1 846	1 552	**1 430**
47	2 729	2 060	2 532	2 194	**1 954**	**1 954**
48	1 916	1 432	1 708	1 436	**1 427**	**1 427**
49	1 494	960	**892**	914	932	920
50	1 717	1 268	1 566	1 283	1 393	**1 266**
Meilleur ; pire*		**7** ; 3	**1** ; 31	**6** ; 11	**16** ; 5	**34** ; 0
Moyenne	2 151	1 471	1 678	1 565	1 470	1 435

* Nombre de fois où l'on a obtenu, par la méthode considérée, le meilleur (respectivement : pire) résultat.

Figure 4. – Exemple de résultats obtenus pour 24 heures (1 440 minutes).

données correspondant aux cinq premiers essais. Les résultats obtenus avec les méthodes stochastiques dépendent évidemment du générateur pseudo-aléatoire utilisé et de l'initialisation de son germe.

L'heuristique nécessite environ 15 minutes de temps de calcul. C'est la méthode qui donne le meilleur rapport qualité du makespan/temps de calcul.

La méthode hybride fournit de bons résultats mais le temps de calcul est d'environ 80 minutes pour effectuer 20 itérations. Ce nombre d'itérations semble suffisant pour obtenir des résultats satisfaisants. Doubler le nombre d'itérations double le temps de calcul, mais aucune amélioration notable n'est obtenue.

La descente stochastique donne des résultats irréguliers, comprenant parfois la meilleure solution. Son temps d'exécution est de 35 minutes pour effectuer 5 000 itérations. Il est important de remarquer que 70 % des ordonnancements voisins générés ne sont pas réalisables : ils provoquent des blocages de crochet ou des dépassements de la durée d'immersion. En fait, sur 5 000 itérations, seuls 1 500 ordonnancements réalisables sont comparés. Effectuer 20 000 itérations améliore en moyenne les makespans, mais le temps de calcul devient très important (140 minutes). Il peut arriver que le makespan obtenu en 20 000 itérations soit supérieur à celui obtenu en 5 000 itérations. Ce résultat est lié au caractère aléatoire de l'algorithme de descente. De même, les écarts entre les makespans fournis par des descentes stochastiques successives (5 000 itérations), pour un même jeu d'essai, peuvent atteindre 6 heures. Ces résultats montrent l'instabilité de la descente stochastique.

Deux techniques sont généralement utilisées pour améliorer l'algorithme de la descente stochastique : effectuer des descentes successives ou appliquer la méthode du kangourou [Fleury, 93]. La descente stochastique donnant une solution fonction de la solution initiale, une première amélioration consiste à effectuer des descentes successives avec des solutions initiales différentes. Cette solution est très coûteuse en temps de calcul. Pour pallier cet inconvénient, la méthode du kangourou consiste à stopper une descente dès qu'un certain nombre d'itérations a été effectué sans amélioration du makespan. Après cet arrêt, deux solutions sont possibles : accepter la transition défavorable puis continuer, ou repartir d'une nouvelle solution initiale. Dans notre cas, aucune de ces techniques ne peut être utilisée : les essais montrent que même en effectuant 20 000 itérations, la valeur du makespan ne se stabilise pas. Arrêter la descente avant ce nombre d'itérations ne peut donc pas donner des résultats satisfaisants.

Les ordonnancements les meilleurs permettent de gérer simultanément dix pièces dans l'atelier. Rappelons que, de manière empirique, il n'est pas possible de gérer plus de trois pièces dans l'atelier. Le gain moyen de temps par rapport à la politique de gestion mise en œuvre est d'environ 50 %. Cette performance est obtenue sans aucune modification de la topologie.

Les méthodes d'ordonnancement mises en œuvre précédemment cherchent un ordonnancement d'un lot de seize pièces. Étant donné le caractère combinatoire du problème d'ordonnancement, il est intéressant d'étudier l'évolution du temps de calcul des méthodes d'ordonnancement lorsque le nombre de pièces à ordonnancer augmente. Il est toujours possible d'évaluer le temps de calcul des méthodes itératives. En effet, le temps de calcul est borné par le nombre d'itérations. Seul le temps d'évaluation d'un ordonnancement augmente.

L'heuristique, de par son fonctionnement, semble coûteuse en temps de calcul lorsque le nombre de pièces augmente. Cependant, l'atelier ne traite que vingt-quatre types différents de pièces. Le nombre de calculs de dates au plus tôt à effectuer lors d'une itération de l'heuristique ne dépasse donc pas vingt-quatre. Si *dmax* est fixé, le temps de calcul de l'heuristique est borné par un polynôme de degré 2. La figure 5 présente les mesures réalisées avec les données du problème traité.

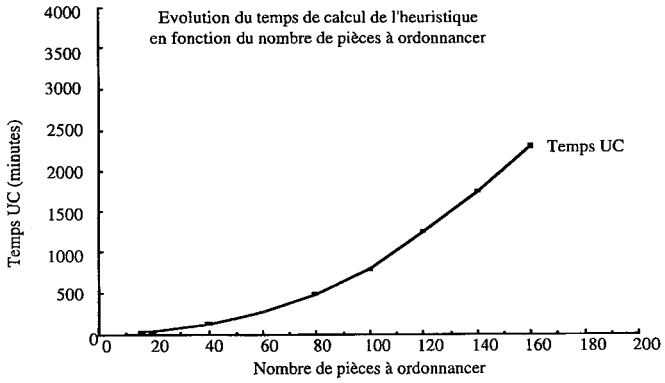

Figure 5. – Mesures de temps de calcul de l'heuristique.

6. UTILISATION DU MODÈLE ET PERSPECTIVES

Il est prévu d'intégrer le modèle et le logiciel de simulation dans le système de contrôle/commande de l'atelier. Cette intégration a pour but d'établir un

flux d'informations bidirectionnel entre le système et son modèle. Pour cela, le modèle et le simulateur doivent être implantés sur un ordinateur de pilotage qui est en liaison avec les automates programmables de l'atelier. Cette intégration permet de travailler à deux niveaux, comme le montre la figure 6 : à long terme pour la planification de l'atelier et à court terme pour le pilotage temps réel.

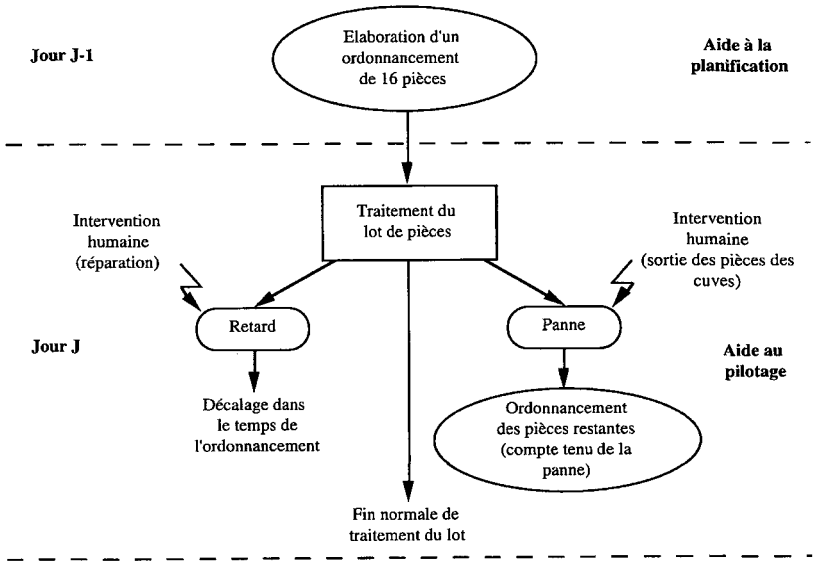

Figure 6. – Intégration du couplage dans le système de contrôle/commande de l'atelier.

La planification consiste à élaborer le jour $J-1$ l'ordonnancement d'un lot de pièces pour le jour J. Cette planification est réalisée par l'exploitation du couplage méthode d'ordonnancement – simulation comme nous l'avons décrit plus haut.

Comme l'a étudié Bolignano [Bolignano, 84] (sans détailler la prise en compte du transport), le pilotage temps réel consiste à diriger le traitement des pièces. Pour cela, la simulation peut donner une liste datée des mouvements des moyens de transport. Ainsi on peut s'assurer que le fonctionnement de l'atelier sera strictement identique au fonctionnement prévu par la simulation et donc que l'ordonnancement prévu pourra être exécuté.

Lors du traitement, des aléas peuvent survenir. Nous distinguons deux types d'aléas : les retards et les pannes. Les retards sont des problèmes qui

ralentissent un traitement, par exemple lorsqu'un convoyeur à crochet n'arrive pas à crocheter une pièce. Ce genre d'incident n'entraîne aucune destruction dans l'atelier mais nécessite une intervention humaine rapide et transparente. Après réparation, le traitement peut reprendre normalement. Les pannes sont des incidents plus graves qui empêchent la continuation normale du traitement, comme par exemple une fissure dans une cuve. Dans ce cas, une intervention humaine est nécessaire pour enlever les pièces restées dans des cuves contenant des solutions agressives. Ces pièces sont manipulées à l'aide de palans indépendants des moyens de transport de l'atelier. Il est nécessaire de rechercher un ordonnancement pour les pièces non traitées. L'algorithme doit alors tenir compte des dégradations de l'atelier pour ordonnancer les pièces dont les traitements peuvent être réalisés. Dans ce contexte, un modèle de simulation paramétré s'avère particulièrement utile.

Enfin, indépendamment de la planification et du pilotage, le modèle peut servir à tester des hypothèses faites sur le fonctionnement de l'atelier. Dans ce cas la simulation permet de répondre à la question « que se passe-t-il si... ? ». Ce mode d'utilisation de la simulation permet d'étudier l'influence de pannes, de tester l'impact de la suppression de certaines cuves, d'étudier le comportement de l'atelier soumis à une surcharge...

CONCLUSION

Nous avons proposé deux méthodes d'ordonnancement (une heuristique et une méthode hybride) et le couplage de ces méthodes avec un modèle de simulation déterministe à événements discrets pour calculer la fonction objectif. Les résultats obtenus ont été jugés très satisfaisants et ont répondu aux besoins exprimés par les industriels.

L'utilisation d'un langage de simulation présente de nombreux avantages. Il permet d'obtenir rapidement des modèles variés et aisément modifiables (par exemple, modification de la topologie, des caractéristiques, des politiques de gestion...). De plus, la simulation fournit des critères de performance qualitatifs et quantitatifs du système étudié (par exemple, évaluation des taux moyens d'occupation des cuves et des convoyeurs, détection de goulets d'étranglement, comparaison des coûts de diverses politiques de gestion...).

Les méthodes d'ordonnancement ont été mises en œuvre avec le même modèle de simulation. L'intérêt d'utiliser différentes méthodes est de pouvoir les valider mutuellement, surtout lorsque la valeur de l'optimum est inconnue et qu'il n'existe aucun moyen de connaître la qualité d'une solution proposée.

Par ailleurs, si plusieurs méthodes sont utilisées, le rapport qualité/temps de calcul sera variable. Selon le temps imparti pour calculer un ordonnancement, l'une ou l'autre des méthodes sera privilégiée. Le manque de réactivité face aux aléas, reproché en général aux ordonnancements planifiés, est alors pallié en partie en utilisant une méthode rapide.

Un modèle déterministe de l'atelier peut paraître peu réaliste car il ne tient compte d'aucun aléa. Dans notre exemple, un tel modèle est suffisant car, en cas d'aléa, une intervention humaine est nécessaire pour reprendre le traitement. Le problème d'ordonnancement est alors reconsidéré avec les pièces non déjà traitées. Dans un cas plus général, la simulation peut fournir au système une liste datée des mouvements des moyens de transport. Le fonctionnement de l'atelier est alors strictement identique au fonctionnement prévu par la simulation et l'ordonnancement s'exécute correctement. Ce type de pilotage évite une dérive entre le fonctionnement réel de l'atelier et le fonctionnement simulé.

Ce couplage méthode d'ordonnancement – simulation à événements discrets peut s'appliquer à d'autres systèmes discrets à partage de ressources. En effet, les méthodes d'ordonnancement présentées sont générales et peuvent facilement s'adapter à d'autres systèmes. Nos perspectives se tournent donc, d'une part, vers l'étude d'autres problèmes tels que celui présenté dans cet article et, d'autre part, vers le couplage méthode d'ordonnancement-modèle de simulation stochastique ([Bulgak, 88] ; [Fleury, 93]).

RÉFÉRENCES

[Barnichon 92] D. Barnichon, C. Caux, G. Fleury et P. Kellert, Simulated Annealing and Discrete Event Simulation for Scheduling, *Proceedings of the European Simulation Multiconference 92*, 1992, June 1-3, York, United Kingdom, p. 162-166.

[Bel 91] G. Bel, *Rapport introductif*, Colloque bilan sur les Actions de Recherche en Robotique & Productique, 1991, 15-16 janvier, Paris, France, p. 551-562.

[Bolignano 84] D. Bolignano, J. M. Proth et K. Voyiatzis, *Ordonnancement des pièces dans un système de traitement de surfaces*, Rapport de recherche INRIA n° 234, août 1984.

[Bulgak 88] A. A. Bulgak et J. L. Sanders, Integrating a modified simulated annealing algorithm with the simulation of a manufacturing system to optimize buffer sizes in automatic assembly systems, *Proceedings of the 1988 Winter Simulation Conference*, 1988, December 12-14, San Diego California, USA, p. 684-690.

[Ferrara 90] A. FERRARA et R. MINCIARDI, Resource Constrained Scheduling via Simulated Annealing: a Discrete Event Approach, *Proceedings of the European Simulation Symposium*, 1990, November 8-10, Ghent, Belgium, p. 177-181.

[Fleury 91] G. FLEURY, Simulated Annealing and Scheduling in Manufacturing Systems, *Operations Research'91, GMÖOR, Physica Verlag*, p. 244-248.

[Fleury 93] G. FLEURY, Quelques méthodes de résolution de problèmes NP-complets, Thèse d'université, Université Blaise Pascal, Clermont-Ferrand-II, février 1993.

[Garey 79] M. R. GAREY et D. S. JOHNSON, *Computers and Intractability. A Guide to the Theory of NP-Completeness*, Freeman, New York, 1979.

[GOTHA 93] GOTHA, Les problèmes d'ordonnancement, *RAIRO Recherche Opérationnelle*, 1993, *27*, n° 1, p. 77-150.

[Kirkpatrick 83] S. KIRKPATRICK, C. D. GELATT Jr. et M. P. VECCHI, Optimization by Simulated Annealing, *Science*, 1983, *221*, p. 671-680.

[Manz 89] E. M. MANZ, J. HADDOCK et J. MITTENTHAL, Optimization of an Automated Manufacturing System Simulation Model Using Simulated Annealing, *Proceedings of the Winter Simulation Conference*, 1989, Dec. 4-6, Washington, USA, p. 390-395.

[Simulog 93] Société Simulog, QNAP2 version 8, *Manuel de référence*, 1993.

ANNEXE : description de l'atelier étudié

A.1. Description des gammes

La figure 7 précise les notations retenues.

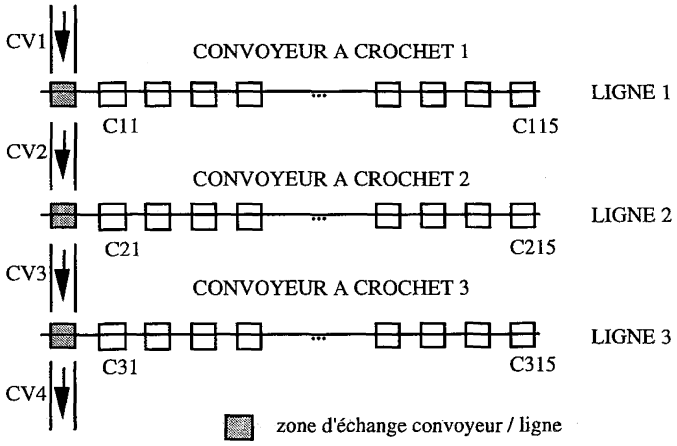

Figure 7. – Topologie de l'atelier (vue de dessus).

Un traitement effectué par l'atelier est représenté par une gamme. Une gamme est une liste ordonnée de cuves et de temps d'immersion associés. Chaque cuve est identifiée par un nombre dont le premier chiffre est le numéro de la ligne où se trouve la cuve et dont le nombre suivant est le numéro de la cuve sur la ligne. Par exemple, la cuve C113 est la cuve numéro 13 de la ligne 1. Les nombres entre parenthèses représentent les temps d'immersion minimum dans une cuve. Le temps d'immersion maximum est fixé à 1.1 fois le temps d'immersion minimum.

Gamme **1 B** : C113(20), C19(10).

Gamme **1 C** : C310(20), C37(10).

Gamme **1 F** : C31(10), C33(7.5), C37(10).

Gamme **1 K** : C310(20), C31(10), C312(7.5), C310(10), C37(10).

Gamme **1 L** : C11(20), C17(10).

Gamme **2 A** : C310(20), C311(75), C312(7.5), C33(7.5), C37(10).

Gamme **3 A** : C21(20), C210(37.5), C23(7.5), C31(10), C312(7.5), C37(10).

Gamme **4 A** : C11(20), C112(360), C13(12.5), C17(10), C19(10).

Gamme **5 A** : C11(20), C12(37.5), C13(7.5), C14(45), C15(7.5), C16(45), C13(7.5), C17(10), C14(45), C15(7.5), C12(37.5), C13(7.5), C14(22.5), C15(7.5), C18(20), C13(7.5), C19(10).

Gamme **5 L** : C21(20), C22(37.5), C13(7.5), C24(45), C25(7.5), C32(45), C33(15), C34(22.5), C35(15), C36(10), C37(10).

Gamme **6 A** : C21(20), C22(37.5), C23(7.5), C31(10), C33(6), C38(3.5), C35(7.5), C37(10), C39(5).

Gamme **6 B** : C21(20), C22(37.5), C23(7.5), C32(45), C33(7.5), C38(37.5), C35(7.5), C37(10), C39(5).

Gamme **6 C** : C21(20), C22(37.5), C23(7.5), C24(22.5), C25(7.5), C32(45), C33(7.5), C38(37.5), C35(7.5), C37(10).

Gamme **6 E** : C21(20), C22(37.5), C23(7.5), C24(12.5), C29(7.5), C22(22.5), C23(7.5), C38(37.5), C35(7.5), C37(10), C39(5).

Gamme **6 G** : C21(20), C22(37.5), C23(7.5), C31(20), C33(7.5), C32(45), C33(10), C38(37.5), C35(7.5), C37(10).

Gamme **7 A** : C11(20), C110(90), C13(12.5), C18(7.5), C13(7.5), C19(10).

Gamme **8 A** : C26(12.5), C214(10).

Gamme **8 B** : C27(12.5), C28(7.5), C213(10), C214(10).

Gamme **8 C** : C27(12.5), C28(7.5), C210(20), C28(7.5), C213(10), C214(10).

Gamme **8 E** : C27(12.5), C28(7.5), C211(37.5), C28(7.5), C212(0.33), C29(7.5), C213(10), C214(10).

Gamme **8 F** : C27(12.5), C28(7.5), C210(22.5), C28(7.5), C211(37.5), C28(7.5), C212(0.33), C29(7.5), C213(10), C214(10).

Gamme **11 B** : C21(20), C24(12.5), C25(7.5), C32(60), C33(22.5), C36(10), C31(10), C33(7.5), C37(10).

Gamme **12 F** : C11(20), C111(180), C13(7.5), C311(75), C33(10), C31(7.5), C33(7.5), C37(10).

Gamme **13 A** : C310(20), C314(5), C33(7.5), C312(7.5), C313(3.5), C35(7.5), C37(10).

A.2. Caractéristiques physiques de l'atelier

Le convoyeur principal est composé des quatre convoyeurs identiques notés CV1, CV2, CV3 et CV4. Un convoyeur ne transporte qu'une seule pièce à un instant donné. La longueur d'un convoyeur est de 4 mètres et sa vitesse est de 6 mètres à la minute.

Les convoyeurs à crochet 1, 2 et 3 sont identiques. Un convoyeur à crochet ne transporte qu'une seule pièce à un instant donné et a une vitesse de 6 mètres à la minute.

La distance entre deux cuves contiguës d'une même ligne est de 2,4 mètres.

Les zones d'échange sont considérées comme des cuves où le temps d'immersion minimal (resp. maximal) est nul (resp. infini).

A.3. Hypothèses de fonctionnement

À l'instant initial, l'atelier est supposé vide et les convoyeurs à crochet de chacune des lignes se trouvent au-dessus des zones d'échange.

Le crochet est requis pour déplacer une pièce immergée dans une cuve dès que le temps minimum d'immersion est écoulé. Si le crochet est occupé, les différentes requêtes sont traitées selon la politique « premier arrivé premier servi ».

Les temps de chargement et de déchargement d'une pièce sur le crochet sont considérés comme nuls.

Les temps de déplacement entre les convoyeurs et les zones d'échange sont considérés comme nuls.

Les vitesses des convoyeurs sont supposées constantes.

A.4. Composition des jeux d'essai

Les jeux d'essai utilisés pour construire le tableau de la figure 4 sont constitués chacun de 16 pièces. Chaque jeu d'essai est généré aléatoirement en fonction des probabilités d'apparition de chacune des gammes. Les jeux d'essai 1 à 5 utilisés dans la figure 4 sont les suivants :

jeu 1 : 4A, 3A, 1C, 5A, 3A, 12F, 6A, 5A, 11B, 5A, 5A, 2A, 3A, 5A, 5A, 5L.

jeu 2 : 8A, 6A, 2A, 8F, 12F, 1C, 6G, 5A, 5A, 1C, 5A, 3A, 5A, 6G, 5A, 8A.

jeu 3 : 8B, 3A, 5A, 5A, 5A, 12F, 1B, 2A, 5A, 5A, 5A, 1C, 5A, 2A, 2A, 5A.

jeu 4 : 3A, 5A, 4A, 1B, 5A, 5A, 5A, 3A, 3A, 1F, 5A, 5A, 1C, 2A, 8F, 6A.

jeu 5 : 5A, 5A, 3A, 1L, 6A, 3A, 11B, 5A, 8C, 5A, 5A, 8C, 2A, 1K, 1C, 5L.