J3eA, Journal sur l'enseignement des sciences et technologies de l'information et des systèmes,
Volume 2, Hors-Série 1, 1 (2003)
DOI : http://dx.doi.org/10.1051/bib-j3ea:2003501

Problèmes d'ordonnancement dans les systèmes de production de type Flow-Shop Hybride en contexte déterministe

M. Gourgand, N. Grangeon et S. Norre

Université Blaise Pascal (Clermont-Ferrand II)
Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes
LIMOS CNRS UMR 6158
BP 10125
F-63173 Aubière Cedex, France

gourgand@isima.fr
grangeon@iris.univ-bpclermont.fr
norre@moniut.univ-bpclermont.fr

Problèmes d'ordonnancement dans les systèmes de production de type Flow-Shop Hybride en contexte déterministe

Michel Gourgand, Nathalie Grangeon, Sylvie Norre

Université Blaise Pascal - Clermont Ferrand II
Laboratoire d'Informatique, de Modélisation et d'Optimisation de Systèmes
LIMOS CNRS UMR 6158
BP 10125
63173 Aubière Cedex
FRANCE
gourgand@isima.fr - grangeon@iris.univ-bpclermont.fr
norre@moniut.univ-bpclermont.fr

1 Introduction

Jusqu'à la fin des années 70, la durée de vie des systèmes de production était suffisamment longue pour permettre aux experts d'en acquérir une connaissance approfondie. Actuellement, les experts sont de plus en plus souvent confrontés à des problèmes de productivité et de réactivité de leurs systèmes de production. Les systèmes de production auxquels nous nous intéressons sont des systèmes à événements discrets et à partage de ressources.

Les experts ont besoin d'outils et de méthodes pour :

- résoudre des problèmes de dimensionnement a priori ou a posteriori,
- appréhender de manière satisfaisante le fonctionnement de leur système,
- assurer le suivi et la gestion de la production,
- aider à la planification des opérations de production,
- obtenir un fonctionnement de leur système robuste aux événements aléatoires.

Ainsi, on peut dire qu'un système de production pose, aussi bien lors de sa conception que durant son exploitation, un certain nombre de problèmes : dimensionnement, compréhension du fonctionnement interne, prise en compte des phénomènes aléatoires, évaluation des performances, optimisation... Certains de ces problèmes peuvent acquérir une importance plus ou moins grande en fonction du contexte (sans ou avec événements aléatoires) dans lequel évolue le système.

Pour résoudre ces problèmes, des hypothèses simplificatrices permettent souvent de formaliser le problème industriel réel sous forme d'un modèle théorique. Ceci permet d'appréhender la complexité des problèmes traités, puisque les problèmes théoriques sont en général bien connus et leur complexité a été étudiée. Les modèles théoriques que nous avons étudiés sont le modèle du Flow-Shop et le modèle du Flow-Shop Hybride. Ces deux modèles sont des modèles génériques qui peuvent être utilisés pour modéliser des systèmes de production composés d'ateliers en série.

Nous considérons ces modèles aussi bien dans un contexte sans événement aléatoire qu'avec événements aléatoires. Sur des problèmes de petite taille, des méthodes exactes permettent d'obtenir une solution optimale. Ces méthodes sont abandonnées au profit de méthodes approchées, telles que les métaheuristiques, pour des problèmes de grande taille. L'utilisation des méthodes approchées sur des problèmes de petite taille n'est justifiée que pour leur validation (solutions obtenues de même qualité).

Les systèmes de production complexes présentent en général une double complexité représentée par la figure 1 :

- une complexité algorithmique. Par abus de langage, nous appellerons complexité algorithmique la complexité des problèmes d'optimisation combinatoire. Cette complexité se traduit par la

recherche d'une solution qui minimise un ou plusieurs critères de performance.
- une complexité structurelle et fonctionnelle du système. Cette complexité se traduit par la difficulté d'évaluer de manière simple le ou les critères de performance du système.

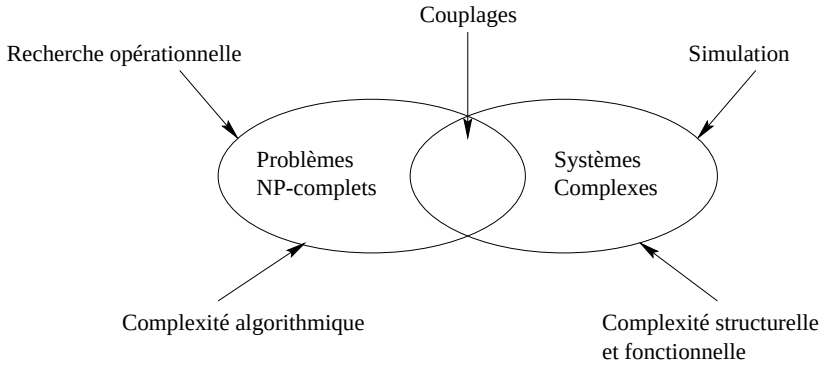


FIG. 1 – La double complexité

Certains problèmes peuvent ne présenter qu'une seule complexité. Le problème d'ordonnancement dans un Flow-Shop de permutation avec des stocks de capacité illimitée, par exemple, présente une complexité algorithmique : le problème d'ordonnancement est NP-complet alors que les critères de performance tels que la date de fin de traitement de l'ensemble des produits peuvent se calculer par une simple formule mathématique.

La comparaison de règles de gestion dans un système complexe, par exemple un Flow-Shop Hybride, présente une complexité structurelle et fonctionnelle. Une méthodologie de modélisation est alors nécessaire pour la construction d'un modèle du système et l'élaboration d'un modèle de simulation, d'un modèle markovien, ... pour l'évaluation des performances.

D'autres problèmes présentent la double complexité. Ce sont des systèmes complexes, de par leur structure et leur fonctionnenemnt pour lesquels on a par exemple à la fois un problème d'ordonnancement et un problème d'évaluation des performances. Un couplage entre une méthode de recherche opérationnelle et un modèle pour l'évaluation des performances est alors utilisé.

Pour résoudre les problèmes d'ordonnancement, nous disposons d'un large éventail de méthodes de recherche opérationnelle regroupées en deux grandes catégories : les méthodes exactes et les méthodes approchées. Pour les modèles théoriques étudiés, nous proposons d'utiliser des heuristiques et des méthodes approchées à base de recuit simulé : l'algorithme de la descente stochastique, l'algorithme du recuit simulé et l'algorithme du kangourou. Ces méthodes sont des métaheuristiques qui consistent, à partir d'une solution initiale (générée aléatoirement ou avec une méthode de construction) à faire évoluer par un processus itératif cette solution à l'aide d'un système de voisinage. Le choix d'une solution dans un voisinage et le critère d'acceptation ou de refus d'une solution voisine varient selon la méthode considérée.

2 Présentation de la problématique

2.1 Définitions

Un système de production pose, aussi bien lors de sa conception que durant son exploitation, un certain nombre de problèmes. Les principaux problèmes rencontrés sont : des problèmes de dimensionnement, de planification, de compréhension du fonctionnement interne, de spécification des règles de gestion, de prise en compte des phénomènes aléatoires et d'évaluation des performances.

Pour une charge donnée (lots de produits à traiter, ...), le dimensionnement consiste à déterminer la quantité et le type des ressources nécessaires ainsi qu'une topologie acceptable afin d'assurer le bon fonctionnement du système de production. Par exemple, on cherche à déterminer la taille des stocks, le nombre de machines pouvant effectuer le même type d'opération, le nombre de ressources de transport,

le type des ressources, etc. On peut réaliser une étude pour le dimensionnement dans les deux cas suivants :

- a priori : le système réel n'existe pas encore, il est en cours de conception. L'objectif est de dimensionner le système afin de permettre un écoulement au mieux de la production, tout en minimisant l'investissement.
- a posteriori : le système réel existe. L'objectif est de redimensionner le système, sans toutefois remettre en cause le système complet, de manière à augmenter la production ou à en améliorer la qualité.

Dans la littérature, les définitions suivantes sont données :

Définition 1 *[35]*
Ordonnancer (ou planifier) le fonctionnement d'un système industriel de production consiste à gérer l'allocation (ou l'accès) à des ressources au cours du temps, tout en satisfaisant au mieux un ensemble de critères.

Définition 2 *[6]*
Ordonnancer, c'est programmer l'exécution d'une réalisation en attribuant des ressources aux tâches et en fixant leurs dates d'exécution. Les problèmes d'ordonnancement apparaissent dans tous les domaines de l'économie : l'informatique (tâches = jobs, produits, pièces, ... et ressources = processeurs, mémoire, machines, ...), la construction (suivi de projet), l'industrie (problèmes d'ateliers, gestion de production), l'administration (emplois du temps). [...]. Les tâches sont le dénominateur commun des problèmes d'ordonnancement, leur définition n'est ni toujours immédiate, ni toujours triviale [...]. Enfin, il faut programmer les tâches de façon à optimiser un certain objectif qui sera, suivant le cas, la minimisation de la durée totale (c'est le critère le plus fréquemment employé) ou le respect des dates de commande ou le lissage des courbes de main d'oeuvre ou encore la minimisation d'un coût. D'une manière générale, trois types d'objectifs sont essentiels dans la résolution des problèmes d'ordonnancement : l'utilisation efficace des ressources, un délai d'exécution des tâches aussi faible que possible, le respect des dates d'achèvement prescrites à l'avance. Bien entendu, il sera souvent plus réaliste, dans la pratique, de considérer plusieurs critères.

Définition 3 *[11]*
Une tâche est un travail élémentaire nécessitant un certain nombre d'unités de temps et de ressources. Ordonnancer un ensemble de tâches, c'est programmer leur exécution en leur allouant les ressources requises et en fixant leur date de début.

Un ordonnancement fournit l'ordre de traitement des différents produits au cours du temps. Dans un problème d'ordonnancement, on cherche en général à minimiser un ou plusieurs critères de performance (tel que le temps total de production). D'autres critères seront présentés dans la suite.

Un problème d'ordonnancement consiste à :

- déterminer les dates d'entrée des produits dans le système.
- trouver un ordre de traitement admissible (qui respecte l'ensemble des contraintes). Le problème peut être surcontraint et donc l'ensemble des solutions admissibles vide.
- constituer des campagnes de production, c'est à dire une stratégie pour la minimisation des temps de montage ou des coûts de changement d'outils. Une campagne est un ensemble de produits préordonnancés.
- déterminer une planification robuste aux événements aléatoires : dans un contexte avec des événements aléatoires, certaines contraintes peuvent entraîner des blocages ou des conflits d'utilisation d'une ressource critique, on cherche donc à minimiser l'impact des événements aléatoires sur le système.

2.2 Prise en compte des événements aléatoires

Les systèmes de production réels sont sujets à de nombreux phénomènes souvent vus comme aléatoires et modélisés comme tels, qui peuvent perturber le fonctionnement du système, la production

et nécessiter rapidement une nouvelle planification. Les raisons liées à l'occurrence des événements aléatoires sont [40] :

- les erreurs humaines (négligence, indisponibilité de la ressource humaine),
- l'insuffisance de ressources auxiliaires (eau, gaz, électricité, ...),
- la panne d'un des équipements (machine, moyen de transport, ...),
- l'usure et les frottements qui peuvent avoir une influence sur les temps de transport.

Parmi les événements aléatoires, on trouve :

- les pannes des machines et des ressources de transport,
- les temps de traitement (ils ne peuvent pas être connus à l'avance),
- l'arrivée inattendue d'une commande prioritaire,
- les dates d'arrivée des produits inconnues.

Pour des systèmes industriels soumis à des événements aléatoires, il s'agit de :

- mesurer l'impact de ces phénomènes sur le fonctionnement du système,
- réagir aux événements,
- concevoir des règles de gestion robustes, permettant de réagir aux aléas.

2.3 Evaluation des performances

La résolution des problèmes énoncés précédemment nécessite le plus souvent le calcul ou l'évaluation de critères de performance (quantitatifs ou qualitatifs). Lorsque les hypothèses sur le système le permettent (stocks de capacité illimitée, pas d'événement aléatoire, ...), certains critères de performance peuvent être calculés par un modèle mathématique ou un simple algorithme. Mais lorsque les contraintes deviennent plus fortes (stocks de capacité limitée, dates d'arrivée des produits aléatoires,...), il devient nécessaire de construire un modèle plus ou moins complexe du système étudié.

Soient :

n	nombre de produits traités
p	nombre de gammes
m	nombre de ressources
n_g	nombre de produits de gamme g. $\sum_{g=1}^{p} n_g = n$
C_i	fin de traitement du produit numéro i
$t_{i,j}$	temps de traitement du produit numéro i par la ressource numéro j
r_i	date de disponibilité du produit numéro i
d_i	date de fin souhaitée du produit numéro i
w_i	poids du produit numéro i
S_i	temps de montage ou de changement d'outils du produit numéro i Ce temps peut dépendre du produit précédent sur la ressource.
O_i	$= 1$ s'il y a un changement d'outils pour débuter la fabrication du produit numéro i, 0 sinon. Cette valeur dépend des caractéristiques du produit i et du produit précédent sur la ressource.
$F_i = C_i - r_i$	temps passé dans le système par le produit numéro i (*flow time*)
$W_i = C_i - r_i - \sum_{j=1}^{m} t_{i,j}$	temps d'attente dans le système du produit numéro i (*waiting time*)
$L_i = C_i - d_i$	décalage temporel du produit numéro i par rapport à la date de fin souhaitée (*lateness*)
$T_i = max\{0,\ C_i - d_i\}$	retard du produit numéro i par rapport à la date de fin souhaitée (*tardiness*)
$E_i = max\{0,\ d_i - C_i\}$	avance du produit numéro i par rapport à la date de fin souhaitée (*earliness*)
$G_i(C_i)$	coût de fin de traitement du produit numéro i à la date C_i
$U_t(i)$	$= 1$ si le produit i est en retard ($C_i > d_i$), 0 sinon
$U_e(i)$	$= 1$ si le produit i est en avance ($C_i < d_i$), 0 sinon

Critère	Maximum	Somme	Somme pondérée
Fin de traitement	$C_{max} = \underset{i=1,n}{max}\ C_i$	$\bar{C} = \frac{1}{n}\sum_{i=1}^{n} C_i$	$\bar{C}_w = \sum_{i=1}^{n} w_i C_i$
Décalage	$L_{max} = \underset{i=1,n}{max}\ L_i$	$\bar{L} = \frac{1}{n}\sum_{i=1}^{n} L_i$	$\bar{L}_w = \sum_{i=1}^{n} w_i L_i$
Retard	$T_{max} = \underset{i=1,n}{max}\ T_i$	$\bar{T} = \frac{1}{n}\sum_{i=1}^{n} T_i$	$\bar{T}_w = \sum_{i=1}^{n} w_i T_i$
Avance	$E_{max} = \underset{i=1,n}{max}\ E_i$	$\bar{E} = \frac{1}{n}\sum_{i=1}^{n} E_i$	$\bar{E}_w = \sum_{i=1}^{n} w_i E_i$
Durée de résidence	$F_{max} = \underset{i=1,n}{max}\ F_i$	$\bar{F} = \frac{1}{n}\sum_{i=1}^{n} F_i$	$\bar{F}_w = \sum_{i=1}^{n} w_i F_i$
Durée de montage et démontage	$S_{max} = \underset{i=1,n}{max}\ S_i$	$\bar{S} = \frac{1}{n}\sum_{i=1}^{n} S_i$	$\bar{S}_w = \sum_{i=1}^{n} w_i S_i$
Coût	$G_{max} = \underset{i=1,n}{max}\ G_i(C_i)$	$\bar{G} = \frac{1}{n}\sum_{i=1}^{n} G_i(C_i)$	$\bar{G}_w = \sum_{i=1}^{n} w_i G_i(C_i)$
Temps d'attente	$W_{max} = \underset{i=1,n}{max}\ W_i$	$\bar{W} = \frac{1}{n}\sum_{i=1}^{n} W_i$	$\bar{W}_w = \sum_{i=1}^{n} w_i W_i$
Nombre de retards		$\bar{U} = \frac{1}{n}\sum_{i=1}^{n} U_t(i)$	$\bar{U}_w = \sum_{i=1}^{n} w_i U_t(i)$
Nombre de décalages		$\sum_{i=1}^{n}(U_t(i) + U_e(i))$	$\sum_{i=1}^{n} w_i(U_t(i) + U_e(i))$
Nombre de changements d'outils		$\bar{O} = (\sum_{i=1}^{n} O_i)$	$\bar{O}_w = (\sum_{i=1}^{n} w_i O_i)$

TAB. 1 – Critères de performance

3 Les modèles du Flow-Shop Hybride

Le modèle du Flow-Shop Hybride est une extension du modèle du Flow-Shop. Il constitue un modèle théorique pour l'étude d'un grand nombre de systèmes de production : industrie du verre [7], [30] ; industrie métallurgique [27], [26] ; industrie du papier [37] ; industrie chimique [10], [34] ; industrie textile [1], [34] ; industrie du bois [34] ; industrie aérospatiale [24].

3.1 Présentation

Un Flow-Shop Hybride (figure 2) est composé de m étages. L'étage j $(j = 1, m)$ est composé de M_j machines parallèles identiques, proportionnelles ou non identiques. Si $M_j = 1,\ \forall j,\ j = 1, m$, on retrouve le modèle du Flow-Shop.

Les produits doivent être traités sur une seule machine de chaque étage. L'ordre des étages est le même pour tous les produits (étage 1, étage 2, ..., étage m). A chaque produit i est associé un temps t_{ijk} de traitement sur la machine k $(k = 1, M_j)$ de l'étage j $(j = 1, m)$.

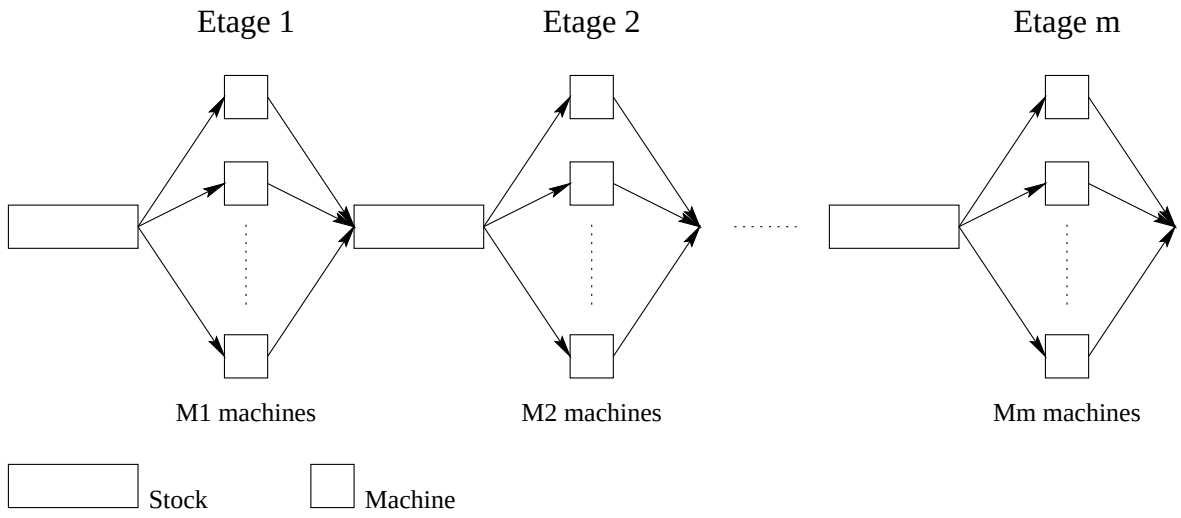


FIG. 2 – Un Flow-Shop Hybride

Les hypothèses classiques pour le Flow-Shop Hybride dans un contexte statique et déterministe sont les suivantes :

H1 La date de disponibilité des produits est connue,

H2 Les machines sont toujours disponibles,

H3 Les temps de traitement sont déterministes et indépendants,

H4 Les temps de montage et de démontage sont inclus dans les temps de traitement,

H5 Les temps de transport sont négligeables,

H6 Il n'y a pas de fragmentation de lots,

H7 Une machine ne peut pas traiter plus d'un produit à un instant donné,

H8 Un produit est traité par au plus une machine à un instant donné,

H9 Les produits peuvent attendre dans des stocks de capacité illimitée entre les étages.

H10 Un produit est traité par une seule machine de chaque étage.

H11 Les machines d'un étage peuvent être identiques, proportionnelles ou non identiques.

Si on considère les événements aléatoires, les trois premières hypothèses peuvent être remplacées par les hypothèses **H'1**, **H'2** et **H'3** présentées pour le Flow-Shop stochastique.

3.2 Notation

La notation proposée par [19], n'étant pas suffisante pour prendre en considération le problème du Flow-Shop Hybride a été enrichie par [39]. L'extension porte essentiellement sur le champ α qui correspond à la description physique du modèle.

Le champ α se compose de quatre paramètres de la manière suivante :

$$\alpha = \alpha_1\alpha_2((\alpha_3\alpha_4^{(j)})_{j=1}^{j=\alpha_2})$$

- $\alpha_1 = FH$ indique qu'il s'agit d'un Flow-Shop Hybride,
- $\alpha_2 = m$ indique le nombre d'étages,
- $\alpha_3 \in \{\emptyset, P, Q, R\}$ indique le type de machines de l'étage j,
- $\alpha_4 = M_j$, indique le nombre de machines de l'étage j.

Les événements aléatoires peuvent être pris en compte par la notation proposée dans la partie précédente.

3.3 Extension de la notation pour la prise en compte des stocks

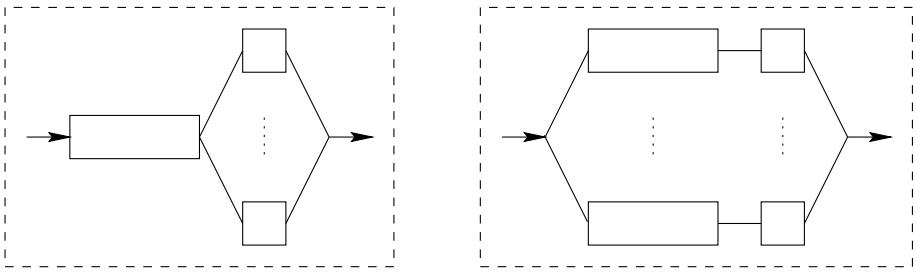
Nous proposons une extension de la notation du Flow Shop Hybride pour prendre en compte la position des stocks (devant un étage et/ou devant une machine). Nous considérons les trois cas représentés par la figure 3.

Cas 1 Les machines constituant l'étage se partagent un seul et même stock.

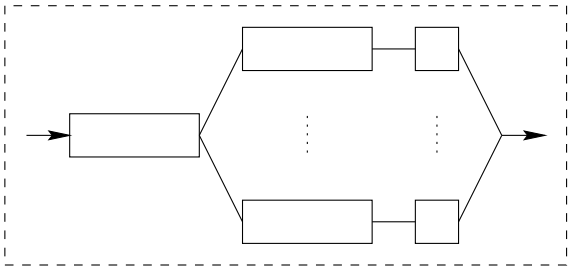
Cas 2 Chaque machine dispose de son propre stock.

Cas 3 Chaque machine dispose de son propre stock et il y a un stock devant l'étage.

Les cas 1 et 2 sont les cas classiques que l'on rencontre dans la litterature. Le cas 3 se retrouve dans des problèmes industriels et dans les systèmes multiprogammés à mémoire virtuelle [5] modélisés par les réseaux de files d'attente.



Cas 1 Cas 2



Cas 3

FIG. 3 – Position des stocks dans un Flow-Shop Hybride

Nous proposons d'ajouter dans le champ α :

- S () pour indiquer que les machines se partagent le même stock.

– (S) pour indiquer que chaque machine dispose de son propre stock.
– S (S) pour indiquer que chaque machine dispose de son propre stock et qu'il y a un stock devant l'étage.

La capacité des stocks est donnée dans le champ β par :
– b_j si la capacité du stock devant l'étage j est limitée,
– $b_{j,k}$ si la capacité du stock devant la machine k de l'étage j est limitée.

Si un étage est composé d'une seule machine, elle sera notée $S1$.

Exemple 1 – *Le Flow-Shop Hybride de la figure 4 se note :* $FH2((SP2), S1)|b_2 = 3$.
– *Le Flow-Shop Hybride de la figure 5 se note :* $FH2(S(P2), (SP2))|b_{2,2} = 0$.
– *Le Flow-Shop Hybride de la figure 6 se note :* $FH2(S(SP2), S1)|b_1 = 2, b_{1,1} = 2, b_{1,2} = 2$.

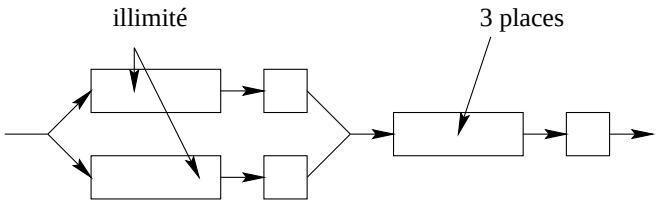


FIG. 4 – $FH2((SP2), S1)|b_2 = 3$

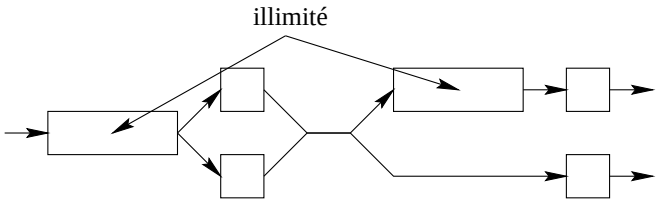


FIG. 5 – $FH2(S(P2), (SP2))|b_{2,2} = 0$

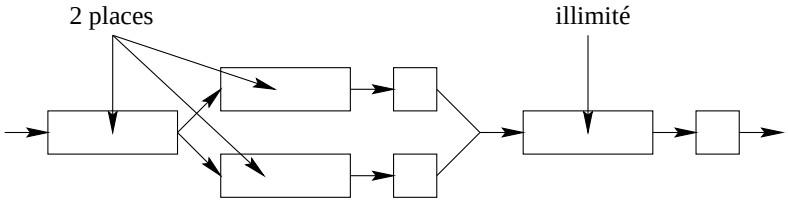


FIG. 6 – $FH2(S(SP2)), S1)|b_1 = 2, b_{1,1} = 2, b_{1,2} = 2$

Un état de l'art complet est donné dans [13].

4 Les systèmes industriels étudiés

On rencontre fréquemment des systèmes industriels modélisables par des Flow-Shop, des Flow-Shop Hybrides. Ces systèmes ont fait l'objet d'un grand nombre de publications et de thèses ([3], [34], [22], [31], ...). Nous présentons dans ce paragraphe deux problèmes industriels, l'un en liaison avec un constructeur automobile et l'autre avec un atelier de production de tôles laminées.

4.1 Fabrication des fils électriques

Le système étudié est un atelier de confection de faisceaux électriques. La production s'effectue selon deux ou trois grandes étapes : la fabrication des fils électriques, la préparation et éventuellement la constitution des faiseaux (figure 7).

Cette étape doit, à partir d'un ensemble de matières premières, fabriquer des lots de fils électriques. Les fils constituant un lot ont les mêmes caractéristiques :

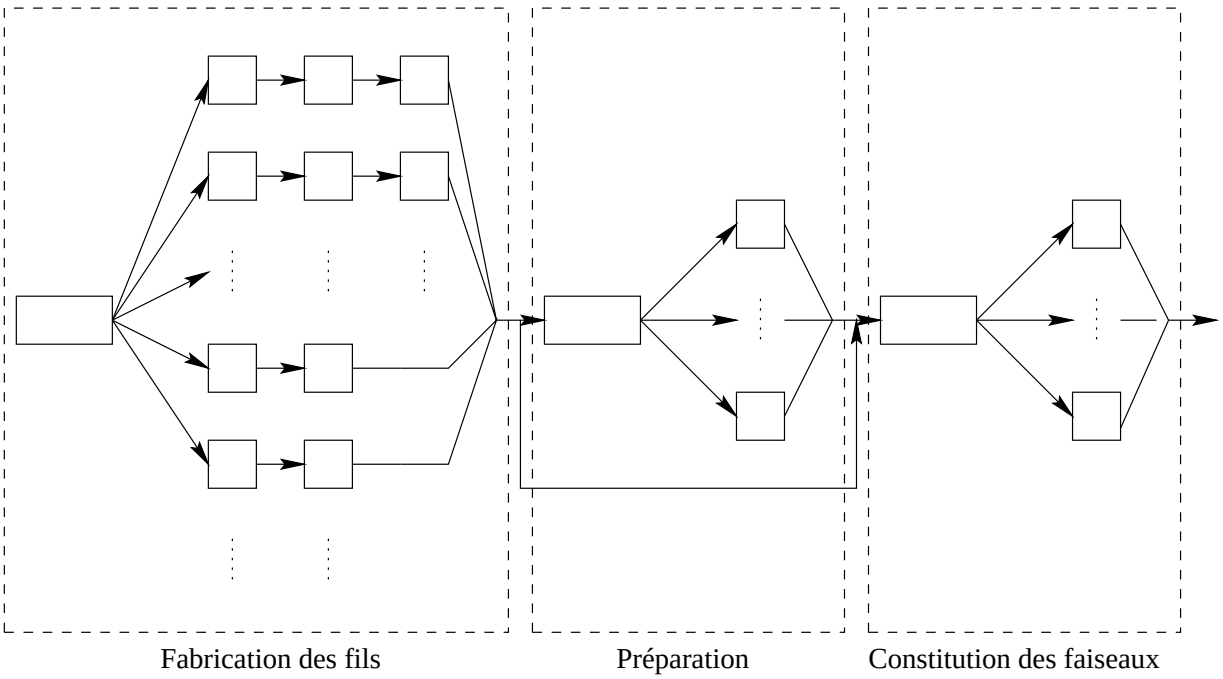


FIG. 7 – Schéma de l'atelier

- couleur,
- longueur,
- longueur de dénudage (une pour chaque extrémité),
- section (diamètre du fil),
- présence ou non d'un joint aux extrémités.

Les opérations effectuées sur les fils sont des opérations de découpe, de dénudage et de pose éventuelle de joint. Elles sont réalisées par une machine divisée en deux ou trois parties :

- découpe et dénudage d'une extrémité,
- dénudage de l'autre extrémité et marquage,
- éventuellement pose de joint.
- Les machines sont non identiques au sens des temps de traitement,
- Les machines sont non identiques du point de vue des opérations effectuées : certaines peuvent poser des joints alors que d'autres ne peuvent pas.
- Les machines sont non identiques du point de vue des caractéristiques : les caractéristiques sont
  - longueur maximum de fil (bornée par la capacité de la machine),
  - section maximum de fil.

  Ainsi, les caractéristiques des machines et des fils font qu'un lot de fils ne peut pas être traité par n'importe quelle machine.

Les machines nécessitent un temps d'équipement et de réglage pour pouvoir passer d'un lot à un autre. Ces temps sont des temps de montage dépendants de la séquence non négligeables : les temps de montage d'un lot dépendent de ses caractéristiques et de celles du lot précédent. Si deux lots qui se suivent ont une caractéristique en commun, le réglage effectué pour le premier des deux lots n'a pas besoin d'être réeffectué pour le lot suivant.

Certaines machines doivent être utilisées en priorité. Si à un instant donné, deux machines sont disponibles pour fabriquer un lot, la machine qui a la priorité la plus forte doit être utilisée.

4.1.1 Préparation et constitution des faisceaux électriques

Un faisceau électrique est constitué d'un grand nombre de fils de différents types auxquels sont ajoutés des connecteurs, des boitiers, etc. Un faisceau peut être vu comme un lot de fils qui doivent

être conditionnés. A la fin de la première étape, les lots de fils de mêmes caractéristiques sont fragmentés pour former d'autres lots de fils avec des caractéristiques différentes.

L'étape de constitution des faisceaux est précédée d'une étape facultative de préparation qui consiste à monter des sous-ensembles de fils en faisceaux. Ce montage s'effectue sur des planches, plusieurs planches sont disponibles en parallèle. Il existe différents types de sous-ensembles et chaque planche est dédiée à une famille de sous-ensembles : une famille regroupe des sous-ensembles similaires, les différences se situent au niveau des connecteurs. La fabrication de chaque sous-ensemble nécessite un temps de montage spécifique dépendant de la séquence.

De même, la constitution des faisceaux s'effectue sur des planches. Plusieurs planches sont disponibles en parallèle. Chaque planche est dédiée à une famille de faiseaux. La fabrication de chaque faisceau nécessite un temps de montage spécifique dépendant de la séquence.

Sur ces machines, la reconfiguration pour la fabrication des lots différents s'avèrent souvent coûteux en temps et nécessitent parfois des ressources humaines importantes. Le problème s'énonce comme le problème de constitution de lots, compte-tenu d'une demande connue et en prenant en compte l'ensemble des contraintes (fils incompatibles, fils devant être incorporés dans les faisceaux par paire,...). Le prix de la matière première (fils électriques, connecteurs, ...) est faible comparé au coût de reconfiguration et de conditionnement.

Le premier objectif de ce problème est de satisfaire la demande tout en minimisant les temps de reconfiguration. Ainsi, si on diminue le nombre de types de faisceaux, on peut diminuer les temps de reconfiguration nécessaires.

Le deuxième objectif est de déterminer le nombre de planches nécessaires pour satisfaire la demande de faisceaux.

4.1.2 Le problème à résoudre

Le système étudié est modélisé par un Flow-Shop Hybride H à trois étages. Le premier étage est constitué de $L_1 = k_1 + k_2$ Flow Shop en parallèle, k_1 Flow-Shop à trois machines et k_2 Flow-Shop à deux machines. Le deuxième étage est composé de L_2 machines en parallèle. Le troisième étage est composé de L_3 machines en parallèle.

La journée de travail est divisée en trois tournées : deux tournées de jour et une tournée de nuit. A chaque tournée est associé un programme de fabrication, c'est-à-dire une liste de lots de fils et de faisceaux électriques à produire. Le problème consiste à déterminer un ordonnancement et une affectation des lots aux Flow-Shop dans le premier étage et aux machines dans le deuxième et le troisième étage, qui respectent les contraintes sur le système, qui permettent de produire tout le programme de fabrication en une tournée et qui optimisent un ou plusieurs critères de performance.

Le comportement du système est déterministe : les machines ne tombent pas en panne et les temps de traitement sont constants.

En utilisant la notation proposée précédemment, le problème se note :

$$
\begin{array}{l}
FHH3(S(Rl_{1,1}k_1, Rl_{1,2}k_2), S(Rl_2L_2), S(Rl_3L_3))|S_{nsd}|H \\
l_{1,1} = F2|b_j = 0 \\
l_{1,2} = F3|b_j = 0 \\
l_2 = F1|b_1 = 0 \\
l_3 = F1|b_1 = 0
\end{array}
$$

4.2 Optimisation d'un atelier

La figure 8 donne une vision synthétique des ateliers de production de tôles laminées. La matière première est issue soit de la fonderie de l'usine soit de fonderies extérieures. Elle passe ensuite éventuellement par une scalpeuse avant de commencer son traitement dans un des deux ateliers : atelier tolerie ou atelier tôles fortes.

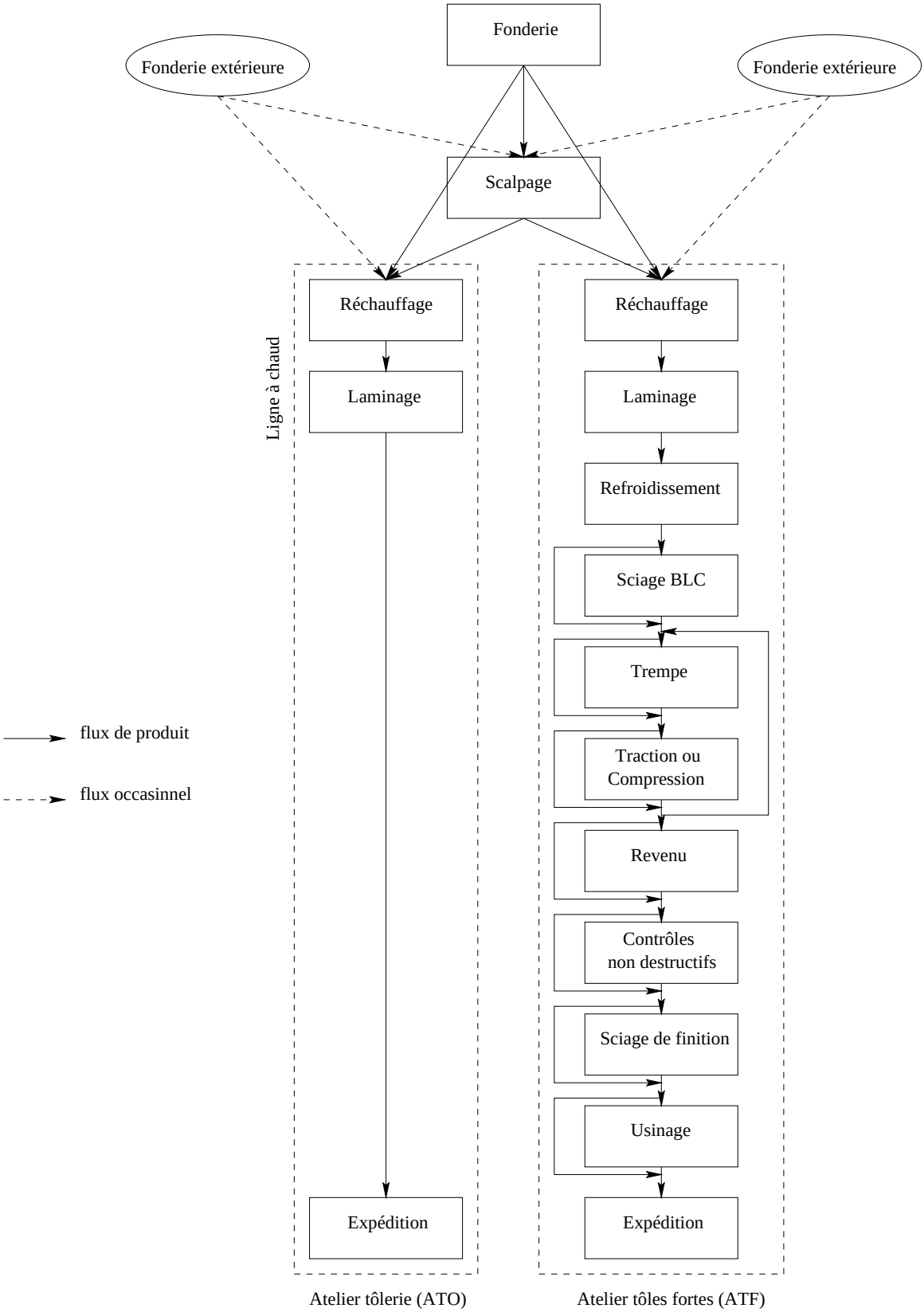


FIG. 8 – Les ateliers de production de tôles laminées

4.2.1 L'atelier tôlerie (ATO)

L'atelier tôlerie d'Issoire produit des tôles laminées de différentes épaisseurs (en général faible épaisseur). La matière première (des plaques de différents alliages) issue de la fonderie est réchauffée et maintenue à température dans des fours PITS puis laminée à chaud avant d'être acheminée vers l'expédition. Le laminage consiste à faire subir à un produit métallurgique porté à bonne température une transformation à chaud par le passage entre des cylindres d'axes parallèles et tournant en sens inverse pour obtenir des produits commercialisables de différentes dimensions.

Un produit est caractérisé, entre autres, par une date de disponibilité, une date de fin souhaitée, un code de réchauffage, un temps de laminage, un temps minimum de séjour dans les fours PITS (dépend du poids). Le code de réchauffage de chaque produit définit la température du four.

Pour réaliser l'opération de réchauffage, plusieurs fours PITS sont disponibles. Un four peut contenir plusieurs produits. Le temps de séjour d'un produit dans un four PITS doit être compris entre une valeur minimum et une valeur maximum. La valeur maximum peut être dépassée, mais au risque d'endommager le produit.

Les produits n'ayant pas forcément le même code de réchauffage, un temps est nécessaire entre deux lots de produits compatibles pour régler la température. Ce temps dépend de la température du lot courant et du lot suivant.

Une seule ressource de transport peut vider un four à un instant donné. Une fois le produit sorti du four, il doit être laminé immédiatement. Cette contrainte peut entraîner des dépassements des durées maximum de séjour.

4.2.2 L'atelier tôles fortes (ATF)

L'atelier tôles fortes produit des tôles laminées d'épaisseur de 5mm à 135mm.

La matière première issue de la fonderie suit un processus de traitement pour ensuite être acheminée vers l'expédition. Les différents traitements possibles d'un produit (i.e. une tôle) sont les suivants :

1. Réchauffage dans un four PITS,
2. Laminage à chaud,
3. Refroidissement,
4. Sciage BLC (Brut de Laminage à chaud),
5. Trempe (Mise en solution et bac de trempe),
6. Traction ou Compression,
7. Revenu,
8. Contrôles non destructifs,
9. Sciage de finition,
10. Usinage,
11. Contrôle, Emballage et Expédition.

Chaque produit est caractérisé entre autres par une gamme, une date de disponibilité de la plaque, une date de fin souhaitée (expédition), un code de réchauffage, un code de trempe et un code de revenu. Toutes ces données ne dépendent pas de la gamme. Les codes de réchauffage, de trempe et de revenu définissent principalement la température du four nécessaire.

Pour réaliser chacune de ces étapes, plusieurs ressources (fours, bancs, scies, ...) sont disponibles. Ces ressources sont soumises à un grande nombre de contraintes. Par exemple, les fours PITS, de trempe et de revenu sont soumis à des contraintes dimensionnelles qui limitent l'épaisseur et la longueur des produits qui peuvent être traités. Un four de revenu ne fonctionne que si un certain tonnage de produits de code de revenu compatible est chargé.

Il existe des possibilités de stockage entre chaque étape. Si les stocks sont trop importants, l'étape précédente est interrompue. Actuellement, le laminoir est régulièrement arrêté. Le temps de stockage entre les fours de trempe et les bancs de traction peut être borné supérieurement pour certains produits.

Les ressources nécessaires pour réaliser chaque étape du processus ne sont pas regroupées au même endroit dans l'atelier. Des moyens de manutention (pont, navette) sont donc nécessaires. Certains moyens de manutention sont propres à chaque ressource.

4.2.3 Le problème à résoudre

Les problèmes suivants correspondent à une vision globale de l'ATO et de l'ATF.
- Maximiser l'utilisation des ressources,
- Minimiser les stocks : les produits en attente sont empilés et sont déstockés selon la règle LIFO ce qui peut entraîner des problèmes de manutention et des retards.

Mais il existe également des problèmes locaux tels que :
- Maximiser le taux d'occupation des fours PITS,
- Déterminer l'affectation des produits aux fours de trempe de manière à maximiser le nombre de produits traités par les fours pendant une durée donnée,
- Minimiser le nombre de lots en attente devant les fours de revenu,
- Minimiser les périodes d'inactivité des fours de revenu.

Les critères de performance peuvent être :
- le retard par rapport à la date de livraison au client,
- le temps de séjour des produits dans l'ATF,
- le temps d'attente des produits dans les stocks,
- etc.

5 Métaheuristiques pour le problème du Flow-Shop Hybride déterministe

Le problème d'ordonnancement dans un Flow-Shop Hybride est un problème NP-complet, sauf dans quelques cas particuliers, ce qui limite l'utilisation de méthodes exactes telles que les procédures par séparation et évaluation ou la programmation linéaire à la résolution de problèmes de faible taille. Les heuristiques et les métaheuristiques sont des méthodes approchées qui permettent de fournir une bonne solution au problème dans un temps court.

Les métaheuristiques auxquelles nous nous intéressons sont des méthodes de recherche locale à base de recuit simulé. Nous avons vu dans le deuxième chapitre que les métaheuristiques, basées sur le recuit simulé permettent l'étude de problèmes à forte combinatoire [38], [9], [17]. Nous avons également vu, dans le troisième chapitre, la nécessité de construire un modèle de simulation pour l'évaluation des performances. L'exécution de ce modèle de simulation peut prendre un temps plus ou moins grand en fonction de la taille du problème étudié. Contrairement à la recherche tabou et aux algorithmes évolutionnistes, les métaheuristiques à base de recuit n'ont besoin que d'une seule évaluation du critère de performance à chaque itération, ce qui est intéressant du point de vue du temps de calcul.

Nous proposons donc d'utiliser
- la descente stochastique,
- le recuit simulé homogène,
- le recuit simulé inhomogène,
- l'algorithme du kangourou

pour la résolution de problèmes d'ordonnancement en contexte déterministe dans la classe des Flow-Shop Hybrides et des Flow-Shop Hybrides Hiérarchisés.

Les hypothèses pour le problème du Flow-Shop Hybride sont les suivantes :

H1 La date de disponibilité des produits est connue,

H2 Les machines sont toujours disponibles,

H3 Les temps de traitement sont déterministes,

H4 Les temps de montage et de démontage sont dépendants ou non dépendants de la séquence,

H5 Les temps de transport sont négligeables,

H6 Il n'y a pas de fragmentation de lots,

H7 Une machine ne peut pas traiter plus d'un produit à un instant donné,

H8 Un produit est traité par au plus une machine à un instant donné,

H9 Les produits peuvent attendre dans des stocks de capacité illimitée, limitée ou nulle devant les machines ou entre les étages,

H10 Un produit est traité par une seule machine de chaque étage,

H11 Les machines d'un étage peuvent être identiques, proportionnelles ou non identiques.

Les hypothèses pour le problème du Flow-Shop Hybride H sont les suivantes :

H1 à H9 Identiques à celles du Flow-Shop Hybride,

H10 Un produit est traité par une seule ligne de chaque étage,

H11 Les lignes d'un étage peuvent être identiques ou non identiques,

H12 Les lignes d'un étage sont des Flow-Shop.

5.1 Notation

Le problème du Flow-Shop Hybride peut être formalisé comme un problème d'optimisation combinatoire :

$$\text{Trouver } x^* \in \Omega / H(x^*) = \min_{x \in \Omega} H(x)$$

où Ω est l'ensemble des solutions admissibles et H le critère à minimiser.

Nous proposons [12] de représenter une solution admissible x par la concaténation de $n+1$ vecteurs :

$$x = (\sigma, v^{(1)}, v^{(2)}, ..., v^{(n)})$$

Le premier vecteur correspond à l'ordonnancement des produits en entrée et les n suivants représentent l'affectation des produits :

- $\sigma = \sigma_1, \sigma_2, ..., \sigma_n$ est une permutation de $\{1, n\}$ pour l'ordonnancement des n produits en entrée du Flow-Shop Hybride,
- $v^{(i)}$ est un vecteur de longueur m pour l'affectation dans chaque étage du $i$ème $(i = 1, n)$ produit dans l'ordonnancement.
  - $v_j^{(i)}$, $1 \leq v_j^{(i)} \leq M_j$ est la machine de l'étage j, $j = 1, m$ affectée au produit σ_i, $i = 1, n$.

Contrairement aux représentations de [29] et de [28], nous considérons uniquement l'ordonnancement des produits en entrée du premier étage. L'ordonnancement en entrée des autres étages ou en entrée des machines ou des lignes de chaque étage est déterminé par l'ordre de sortie des produits de l'étage précédent (règle FIFO) ou en appliquant une règle de gestion. Ainsi, le codage proposé représente une solution admissible, quelles que soient

- les hypothèses sur les machines ou les lignes (identiques ou non identiques)
- la position des stocks (devant une machine ou devant un étage),
- la capacité des stocks (illimitée, limitée, nulle),
- les règles de gestion des stocks.

Les ordonnancements construits à partir de ce codage sont au moins semi-actifs car l'affectation des produits aux machines est imposé et aucun produit ne peut débuter son traitement plus tôt sur la machine à laquelle il est affecté.

Exemple 2 *Soit un Flow-Shop Hybride à deux étages avec des stocks de capacité illimitée et 4 produits en entrée. Le premier étage est composé de 3 machines identiques en parallèle et le second de 2 machines identiques en parallèle. Le tableau 2 illustre l'ordonnancement et l'affectation correspondant au vecteur* $x = ((1342)(11)(11)(31)(22))$. *x représente une solution admissible. Le diagramme de Gantt correspondant est donné par la figure 9. Compte tenu de l'affectation des produits aux machines, les produits sont traités dans le premier étage selon l'ordre 1, 2, 4 et 3 et dans le deuxième étage, selon l'ordre 4, 2, 1 et 3.*

Produit	Etage 1			Etage 2		x	Temps de traitement	
	M1	M2	M3	M1	M2		Etage 1	Etage 2
						$\sigma = (1, 3, 4, 2)$		
1	X			X		$v^{(1)} = (11)$	3	2
3	X			X		$v^{(2)} = (11)$	2	2
4			X	X		$v^{(3)} = (31)$	1	5
2		X			X	$v^{(4)} = (22)$	4	1

TAB. 2 – Exemple

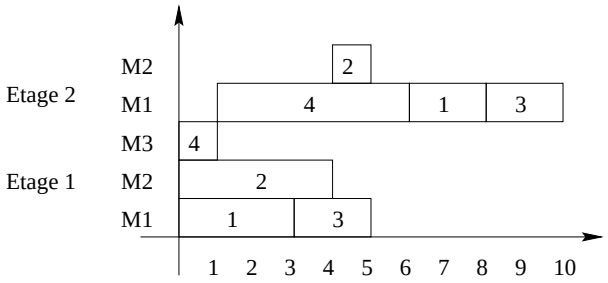


FIG. 9 – Diagramme de Gantt

Lorsque les machines ou les lignes sont non identiques, le nombre de solutions admissibles est :

$$Card(\Omega) = n! \left(\prod_{j=1}^{m} M_j \right)^n \tag{1}$$

Brah et Hunsucker, dans [4], ont montré que lorsque les machines de chaque étage d'un Flow-Shop Hybride sont identiques, le nombre de solutions admissibles est :

$$Card(\Omega) = \prod_{j=1}^{m} \binom{n-1}{M_j - 1} \frac{n!}{M_j!} \tag{2}$$

5.2 Proposition de systèmes de voisinage

Un des points les plus importants dans l'utilisation de métaheuristiques réside dans la définition de systèmes de voisinage. Un bon système de voisinage accélère la convergence de la méthode et permet d'obtenir de bons résultats en un temps de calcul raisonnable. Pour les problèmes de Flow-Shop Hybride, nous proposons [12] un système de voisinage, noté $\mathcal{V}(\mathcal{S}, \mathcal{A})$ où $\mathcal{S}$ est un système de voisinage pour l'ordonnancement en entrée et $\mathcal{A}$ un système de voisinage pour l'affectation.

5.2.1 Système de voisinage $\mathcal{S}$ pour l'ordonnancement

Les systèmes de voisinage classiques pour l'ordonnancement dans un problème de Flow-Shop sont la permutation et l'insertion. Nous proposons d'utiliser ce type de système de voisinage pour l'ordonnancement σ en entrée. Le tableau 3 présente les trois systèmes de voisinage proposés, une illustration est donnée pour le vecteur x de l'exemple 2

Dans les systèmes de voisinage présentés, les produits conservent leur affectation, mais on peut également envisager des systèmes de voisinage dans lesquels les produits ne conservent pas leur affectation. Les produits sont choisis aléatoirement ainsi que la position i_2 dans $\mathcal{I}_{i_1,i_2}$.

Les systèmes de voisinage proposés respectent les propriétés d'accessibilité et de réversibilité présentées dans le deuxième chapitre.

$\mathcal{P}_{i_1,i_1+1}$	permutation du produit σ_{i_1} et du produit σ_{i_1+1}
	$((\mathbf{134}2)(11)(\mathbf{11})(\mathbf{31})(22)) \rightarrow ((\mathbf{143}2)(11)(\mathbf{31})(\mathbf{11})(22))$
$\mathcal{P}_{i_1,i_2}$	permutation du produit σ_{i_1} et du produit σ_{i_2}
	$((\mathbf{1342})(\mathbf{11})(11)(31)(\mathbf{22})) \rightarrow ((\mathbf{2341})(\mathbf{22})(11)(31)(\mathbf{11}))$
$\mathcal{I}_{i_1,i_2}$	insertion du produit σ_{i_1} à la position i_2
	$((134\mathbf{2})(11)(11)(31)(\mathbf{22})) \rightarrow ((\mathbf{1}234)(11)(\mathbf{22})(11)(31))$

TAB. 3 – Systèmes de voisinage $\mathcal{S}$ pour l'ordonnancement

5.2.2 Système de voisinage $\mathcal{A}$ pour l'affectation

Nous proposons deux types de systèmes de voisinage pour l'affectation. Le premier, appelé système de voisinage de base respecte les propriétés d'accessibilité et de résersibilité. Le second, appelé système de voisinage guidé ne respecte pas ces propriétés mais est proposé dans le but d'accélerer la convergence des métaheuristiques.

Système de voisinage de base Les systèmes de voisinage proposés pour l'affectation consistent à modifier pour un ou plusieurs produits, choisis aléatoirement, l'affectation dans un ou plusieurs étages, choisis aléatoirement. Le tableau 4 présente les six systèmes de voisinage proposés, une illustration est donnée pour le vecteur x de l'exemple 2

$\mathcal{A}_1^1$	Pour un produit, modification de son affectation dans un étage
	$((1342)(11)(\mathbf{11})(31)(22)) \rightarrow ((1342)(11)(\mathbf{21})(31)(22))$
$\mathcal{A}_1^n$	Pour les n produits, modification de leur affectation dans un étage
	$((1342)(\mathbf{11})(\mathbf{11})(\mathbf{31})(\mathbf{22})) \rightarrow ((1342)(\mathbf{12})(\mathbf{21})(\mathbf{32})(\mathbf{12}))$
$\mathcal{A}_m^1$	Pour un produit, modification de son affectation dans les m étages
	$((1342)(11)(\mathbf{11})(31)(22)) \rightarrow ((1342)(11)(\mathbf{22})(31)(22))$
$\mathcal{A}_1^r$	Pour $r,\ r \in \{1,n\}$ produits, modification de l'affectation dans un étage
	$((1342)(\mathbf{11})(11)(\mathbf{31})(22)) \rightarrow ((1342)(\mathbf{21})(11)(\mathbf{32})(22))$
$\mathcal{A}_{r'}^1$	Pour un produit, modification de l'affectation dans $r',\ r' \in \{1,n\}$ étages
	$((1342)(11)(\mathbf{11})(31)(22)) \rightarrow ((1342)(11)(\mathbf{32})(31)(22))$
$\mathcal{A}_{r'}^r$	Pour $r,\ r \in \{1,n\}$ produits, modification de l'affectation dans $r',\ r' \in \{1,n\}$ étages
	$((1342)(\mathbf{11})(11)(\mathbf{31})(22)) \rightarrow ((1342)(\mathbf{22})(11)(\mathbf{21})(22))$

TAB. 4 – Systèmes de voisinage $\mathcal{A}$ pour l'affectation

Les produits, les étages et les machines ou les lignes sont choisis aléatoirement. Le système de voisinage ne tient pas compte des étages ne comportant qu'une seule machine ou une seule ligne.

Système de voisinage guidé Le système de voisinage de base construit des ordonnancements au moins semi-actifs. Pour améliorer la qualité des solutions générées, à chaque itération une heuristique peut permettre de construire des ordonnancements actifs ou sans délai. Cette modification correspond à l'utilisation d'un système de voisinage guidé.

Pour augmenter la vitesse de convergence des métaheuristiques, nous proposons un système de voisinage guidé qui consiste à utiliser une heuristique pour l'affectation : à chaque itération, pour un ordonnancement σ donné (obtenu avec le système de voisinage pour l'ordonnancement), une affectation est déterminée par une heuristique.

Cette heuristique (algorithme 1) notée $\mathcal{H}$ correspond à la règle d'affectation ECT et généralise les heuristiques de [15] et de [34]. Son principe est le suivant : Etage par étage, l'heuristique $\mathcal{H}$ affecte les produits à la machine susceptible d'en terminer le traitement le plus tôt.

Un autre système de voisinage guidé consiste à utiliser le couplage entre le module d'évaluation des performances et le module de spécification du fonctionnement interne du couplage triple : le module

Algorithme 1 Heuristique $\mathcal{H}$ pour l'affectation

Soit σ un ordonnancement en entrée
for chaque étage $j = 1, m$ **do**
  if $j > 1$ **then**
    Ordonner les produits de σ selon les dates de sortie décroissantes de l'étage $j-1$.
  end if
  for chaque produit σ_i, $i = 1, n$ **do**
    Affecter le produit σ_i à la machine susceptible d'en terminer le traitement le plus tôt.
    Mettre à jour la date de disponibilité de la machine.
    Calculer les dates de début et de fin de traitement du produit σ_i.
  end for
end for

de spécification du fonctionnement interne détermine, au fur et à mesure de la simulation l'affectation des produits, en fonction de l'état des sous-systèmes physique et logique et de règles de gestion.

5.3 Plan de réglage

Pour le recuit simulé, les paramètres suivants doivent être déterminés :
- Valeur initiale de la température (c_0),
- Valeur finale de la température (c_f) ou critère d'arrêt,
- Longueur L_{c_k} des chaînes de Markov,
- Règle de changement de la valeur courante de la température c_k en c_{k+1}.

Pour l'algorithme du kangourou, la longueur du palier (nombre maximum d'itérations sans amélioration) doit être spécifié.

5.3.1 Recuit simulé homogène

La méthode du recuit simulé (algorithme 2) a été proposée en 1983 par Kirkpatrick [21] pour la résolution de problèmes d'optimisation combinatoire.

La valeur initiale de la température T_k (notée T_0) est déterminée de manière à accepter toutes les transitions i.e. $exp((H(x) - H(y))/T_0) \simeq 1$, $\forall(x, y)$. On pose

$$T_0 = \frac{\overline{\Delta H}^{(+)}}{ln(x_0^{-1})} \tag{3}$$

où
- $x_0 \simeq 0.8$ [20],
- $\overline{\Delta H}^{(+)}$ est l'augmentation moyenne du coût pour un certain nombre de transitions aléatoires.

L'algorithme 3 décrit le calcul de T_0 pour L_{T_0} transitions. Toutes les transitions sont acceptées. Si une transition de x vers y entraîne une augmentation du coût ($H(x) < H(y)$), elle est mémorisée.

La décroissance de la température est choisie de manière à ce que les chaînes de Markov rétablissent rapidement l'équilibre après la modification de T_k. La règle suivante est fréquemment utilisée avec α constant et inférieur à 1.

$$T_k = \alpha.T_{k-1} \text{ avec } \alpha \in [0.5, 0.99] \tag{4}$$

Dans [20], Kirkpatrick, Gelatt et Vecchi donnent $\alpha = 0.95$

Algorithme 2 Algorithme de principe du recuit simulé homogène

Soient x une solution initiale, T_0 une température initiale, L_{T_0} une longueur initiale de palier
$k := 0$
repeat
  for $l := 1$ à L_{T_k} **do**
    Choisir uniformément et aléatoirement y dans le voisinage $\mathcal{V}(x)$
    if $H(y) \leq H(x)$ **then**
      x := y
    else
      if $exp\left(\frac{H(x) - H(y)}{T_k}\right) > random[0,1)$ **then**
        x := y
      end if
    end if
  end for
  Déterminer $L_{T_{k+1}}$ en fonction de L_{T_k}
  Déterminer T_{k+1} en fonction de T_k
  k := k + 1
until critère d'arrêt

Algorithme 3 Calcul de $T_0 = \frac{\overline{\Delta H}^{(+)}}{ln(x_0^{-1})}$

Soit x une solution initiale quelconque
$\Delta H := 0$
for $l := 1$ to L_{T_0} **do**
  Choisir uniformément et aléatoirement y dans $\mathcal{V}(x)$
  $\Delta H := \Delta H + max\{0; H(y) - H(x)\}$
  $x := y$
end for
$\overline{\Delta H}^{(+)} := \Delta H \ / \ L_{T_0}$
$x_0 := 0.8$
$T_0 = \frac{\overline{\Delta H}^{(+)}}{ln(x_0^{-1})}$

Le critère d'arrêt est déterminé, soit en fixant le nombre de valeurs de T_k pour lesquelles l'algorithme sera exécuté (de 6 [25] à 50 [2]), soit en terminant l'exécution si il n'y a pas eu d'amélioration pendant un certain nombre d'itérations.

La manière la plus simple de choisir la taille L_{T_k} de la $k$ième chaîne de Markov est de prendre une valeur dépendant polynomialement de la taille du problème. Dans [20], Kirkpatrick, Gelatt et Vecchi donnent $L_{T_k} =$ nombre de variables du problème. D'où, pour le Flow-Shop Hybride et le Flow-Shop Hybride Hiérarchisé,

$$L_{T_k} = n \times (m+1)$$

5.3.2 Recuit simulé inhomogène

Le réglage des paramètres pour le recuit simulé inhomogène est similaire à celui du recuit simulé homogène. Le critère d'arrêt est le même. La valeur initiale de la température T_0 est calculée par l'algorithme 3. Contrairement au recuit simulé homogène, la longueur L_{T_k} des chaînes de Markov est constante et vaut 1. Pour éviter une décroissance trop rapide de la température, nous proposons de choisr α tel que $\alpha \in [0.9, 0.99]$.

5.3.3 Algorithme du kangourou

L'algotihme 4 donne l'algorithme de principe du kangourou.

[9] a proposé la formule suivante pour calculer A, le nombre d'itérations sans amélioration avant un saut :

$$A \approx -card(\mathcal{V}) ln(1-\rho)$$

En pratique, [22] propose $A = 0.7 \times card(\mathcal{V})$. Pour les systèmes de voisinage proposés, $card(\mathcal{V}) = card(\mathcal{S}) \times card(\mathcal{A})$ avec $card(\mathcal{S})$ et $card(\mathcal{A})$ donnés par le tableau 5.

$card(\mathcal{S})$	$card(\mathcal{A})$
$card(\mathcal{P}_{i_1,i_1+1}) = n$	$card(\mathcal{A}_1^1) = n \sum_{j=1}^{m} (M_j - 1)$
$card(\mathcal{P}_{i_1 i_2}) = n(n-1)/2$	$card(\mathcal{A}_1^n) = \left(\sum_{j=1}^{m} (M_j - 1) \right)^n$
$card(\mathcal{I}_{i_1 i_2}) = (n-1)^2$	$card(\mathcal{A}_m^1) = n \prod_{j=1/M_j>1}^{m} (M_j - 1)$
	$card(\mathcal{A}_1^r) = \binom{r}{n} \left(\sum_{j=1}^{m} (M_j - 1) \right)^r$
	$card(\mathcal{A}_{r'}^1) \leq n \binom{r'}{m} M^{r'}$ où $M = \max_{j=1,m} (M_j - 1)$
	$card(\mathcal{A}_{r'}^r) \leq \binom{r}{n} \left(\binom{r'}{m} M^{r'} \right)^r$ où $M = \max_{j=1,m} (M_j - 1)$

TAB. 5 – $card(\mathcal{S})$ et $card(\mathcal{A})$

Par exemple, pour un Flow-Shop Hybride composé de 5 étages avec 5 machines et traitant 100 produits, $card(\mathcal{V}(\mathcal{I}_j, \mathcal{A}_1^1)) = 100 \times 99 \times 100 \times 20 = 19800000$ solutions voisines et $card(\mathcal{V}(\mathcal{I}_j, \mathcal{H})) = 100 \times 99 = 9900$ solutions voisines.

En pratique, nous proposons de considérer que $A = card(\mathcal{S}) + card(\mathcal{A})$

Algorithme 4 Algorithme de principe de l'algorithme du kangourou

Soient $A > 0$ une longueur de palier et x une solution initiale
$k := 0$, $best := x$
repeat
  if $k < A$ **then** {Descente stochastique}
    Choisir uniformément et aléatoirement y dans le voisinage $\mathcal{V}(x)$
    if $H(y) \leq H(x)$ **then**
      if $H(y) < H(x)$ **then**
        $k := 0$
        if $H(y) < H(best)$ **then**
          $best := y$
        end if
      end if
      $x := y$
    else
      $k := k + 1$
    end if
  else {Pas d'amélioration depuis A itérations}
    Choisir uniformément et aléatoirement y dans le voisinage $\mathcal{W}(x)$.
    if $H(y) \neq H(x)$ **then**
      $k := 0$
    end if
    if $H(y) < H(best)$ **then**
      $best := y$
    end if
    $x := y$
  end if
until critère d'arrêt

5.4 Mise en oeuvre et résultats

Dans cette partie, nous proposons de tester les métaheuristiques et les systèmes de voisinage proposés sur des problèmes de la littérature (en général de «petite» taille), puis sur des problèmes que nous avons générés et sur un problème industriel. Le critère considéré est le makespan. Les résultats obtenus sont comparés avec le makespan optimal obtenu par la résolution du modèle mathématique proposé si cette valeur peut être obtenue en un temps raisonnable et avec la borne inférieure de [36] sinon.

Nous utilisons les termes ou notations suivants :

configuration : une configuration de Flow-Shop Hybride définit le nombre n de produits en entrée, le nombre m d'étages et le nombre de machines $M_j,\ j = 1, m$ ou de lignes $L_j,\ j = 1, m$ par étage.

problème : un problème de Flow-Shop Hybride est défini par une configuration (n, m, $M_j,\ j = 1, m$ ou $L_j,\ j = 1, m$) et un tirage des temps de traitement de chaque produit sur chaque machine. Un problème peut être à machines identiques, proportionnelles ou non identiques.

réplication : en raison de la nature probabiliste des métaheuristiques utilisées, chaque métaheuristique sera lancée plusieurs fois pour un problème donné. Un lancement sera appelé réplication.

distance relative : pour comparer les solutions obtenues, nous calculons la *distance relative* entre le critère de la solution obtenue et une *valeur de référence* selon la formule :

$$\text{Distance Relative } = 100 \times (\text{Résultat} - \text{Valeur de Référence})/\text{Valeur de Référence} \quad (5)$$

La *valeur de référence* est soit la valeur optimale du critère si celle-ci peut être obtenue en un temps raisonnable, soit une borne inférieure.

M_I : distance relative moyenne entre la solution initiale et la borne inférieure pour un ensemble P de problèmes.

$$M_I = \frac{1}{|P|}\sum_{p=1}^{|P|} 100 \times (H_0^p - BI_p)/BI_p$$

où H_0^p est le critère de la solution initiale du problème p et BI_p est la borne inférieure du problème p.

M_{BI} : distance relative moyenne entre la solution obtenue et la borne inférieure pour un ensemble P de problèmes et R de réplications d'une méthode.

$$M_{BI} = \frac{1}{|P|}\sum_{p=1}^{|P|}(\frac{1}{R}\sum_{r=1}^{R} 100 \times (H_r^p - BI_p)/BI_p)$$

où H_r^p est le critère de la solution obtenue par la réplication $r,\ r = 1, R$ de la méthode pour le problème $p,\ p = 1, P$.

ET_{BI} : écart type de la distance relative entre la solution obtenue et la borne inférieure pour un ensemble P de problèmes et R de réplications d'une méthode.

$Min; Max$: plus petite et plus grande distance relative obtenue pour un nombre P de problèmes et R de réplications d'une méthode.

M_{OPT} : distance relative moyenne entre la solution obtenue et la solution optimale pour un ensemble P de problèmes et R de réplications d'une méthode.

$$M_{OPT} = \frac{1}{|P|}\frac{1}{R}\sum_{p=1}^{|P|}\sum_{r=1}^{R} 100 \times (H_r^p - OPT_p)/OPT_p$$

où $OPT_p,\ p = 1, P$ est la solution optimale du problème p.

ET_{OPT} : écart type de la distance relative entre la solution obtenue et la solution optimale pour un ensemble P de problèmes et R de réplications d'une méthode.

Nb : pourcentage du nombre de fois où la solution optimale est obtenue pour un ensemble P de problèmes et R de réplications d'une méthode.

5.4.1 Problèmes de la littérature

Notre objectif n'est évidemment pas de traiter des instances qui peuvent facilement être résolus en un temps raisonnable par des méthodes exactes telles que les procédures par séparation et évaluation ou la programmation linéaire, mais de valider les systèmes de voisinage proposés. On recourt, en général aux métaheuristiques lorsque les méthodes exactes échouent.

Pour calculer le makespan dans un Flow-Shop Hybride avec des stocks de capacité illimitée devant les machines et la règle FIFO, nous utilisons l'algorithme donné dans [13]. Cet algorithme est en général plus rapide qu'un modèle de simulation. Il consiste à calculer la longueur du plus long chemin dans un graphe orienté et est basé sur la remarque suivante : un produit σ_i commence dans l'étage j soit à sa sortie de l'étage $j-1$ soit lorsque la machine $v_j^{\sigma(i)}$ qui lui est affectée est libre.

Dans un premier temps, nous avons testé les méthodes et les systèmes de voisinage proposés sur des instances de la littérature avec deux ou trois étages et de 1 à 4 machines identiques par étage. Les méthodes utilisées dans la littérature pour résoudre ces instances sont des méthodes exactes donc la solution optimale est connue. Le tableau 6 donne les résultats pour les réplications de la descente stochastique.

Référence	m	M	n	M_{BI}	ET_{BI}	M_{OPT}	ET_{OPT}
[36]	3	2 2 2	5	0	0	0	0
[4]	2	2 2	4	10	0	0	0
[33]	3	2 2 1	4	2.7	0	0	0
[32]	2	3 2	5	6.5	0	0	0
[8]	2	4 4	8	5.5	0	0	0
[16]	2	1 2	8	0	0	0	0
[23]	2	2 2	5	0	0	0	0
[15]	2	2 1	8	0	0	0	0
[18]	3	1 1 1	4	12.7	0	0	0

TAB. 6 – Résultats de la descente stochastique pour les instances de la littérature

Nous remarquons que nous obtenons la solution optimale pour chaque instance et puisque l'écart type est égal à zéro, la solution optimale est atteinte à chaque réplication de la descente stochastique. Nous obtenons les mêmes résultats pour l'algorithme du kangourou et pour le recuit simulé. Ces résultats ont été obtenus en un faible temps de calcul : au maximum une seconde sur une O2 Silicon Graphics à 195MHz.

Nous remarquons également que pour certaines instances, la distance relative entre nos résultats et la borne inférieure est égale à zéro. La borne inférieure, pour ces instances correspond donc à la solution optimale. Par contre, pour les autres instances, l'écart entre nos résultats et la borne inférieure est plus grand. Etant donné que nous obtenons la solution optimale, il semble que la borne inférieure peut être très éloignée de la solution optimale (de 2.7% à 12.7%).

À partir de cette remarque, nous avons testé l'efficacité de la borne inférieure et des métaheuristiques sur différentes configurations de Flow-Shop Hybrides. Notre plan d'expérience est celui de [39]. Il permet de tester l'efficacité de méthodes, non seulement en fonction du nombre d'étages et du nombre de produits, mais également en fonction de la position d'un étage goulet dans le système. Dans ce plan d'expérience, un étage goulet est défini comme un étage composé uniquement d'une machine - les autres étages sont composés de trois machines identiques. En général, la définition d'un étage goulet dépend de la capacité moyenne de chaque étage (basée sur les temps de traitement des produits) mais dans notre cas, les temps de traitement sont définis de manière à ce que les capacités de chaque étage ne dépendent que du nombre de machines dans l'étage : les temps de traitement dans chaque étage sont similaires.

Le nombre d'étages varie de 2 à 5 et le nombre de produits de 5 à 15. Les temps de traitement des produits sont distribués selon une loi uniforme de paramètres (8-EC, 8+EC) avec EC=1 (temps

de traitement proches) ou $EC = 7$ (temps de traitement éloignés). Comme dans [39], pour chaque configuration (nombre d'étages, nombre de machines, nombre de produits et EC), trois instances (avec des temps de traitement différents) ont été générées.

Descente stochastique EC = 1												
m	n	Position de l'étage goulet										
		Début		Milieu		Fin		Aucun				
		M_{BI}	ET_{BI}	M_{BI}	ET_{BI}	M_{BI}	ET_{BI}	M_{BI}	ET_{BI}	M_{OPT}	ET_{OPT}	Nb
2	5	0	0	-	-	0	0	12.70	0	0	0	100
2	10	0	0	-	-	0	0	10.59	0.98	0.23	0.04	47
2	15	0	0	-	-	0	0	5.84	1.45	0.87	0.26	10
3	5	0	0	0	0	0	0	6.90	0	0	0	100
3	10	0	0	0	0	0	0	8.37	1.16	0.17	0.02	43
3	15	0	0	0	0	0	0	7.03	1.41	0.56	0.12	10
5	5	0	0	0	0	0	0	6.82	0	0	0	100
5	10	0	0	0	0	0	0	7.24	0.77	0.14	0.02	50
5	15	0	0	0	0	0	0	8.19	1.79	0.37	0.08	17
EC=7												
m	n	Position de l'étage goulet										
		Début		Milieu		Fin		Aucun				
		M_{BI}	ET_{BI}	M_{BI}	ET_{BI}	M_{BI}	ET_{BI}	M_{BI}	ET_{BI}	M_{OPT}	ET_{OPT}	Nb
2	5	0	0	-	-	0	0	5.09	0	0	0	100
2	10	0	0	-	-	0	0	4.13	1.62	2.56	1.14	20
2	15	0	0	-	-	0	0	1.92	1.65	1.92	0.32	33
3	5	0	0	0	0	0.78	0	1.23	0	0	0	100
3	10	0	0	0	0	0	0	8.78	1.71	0.59	0.11	13
3	15	0.26	0	0	0	1.16	0	6.35	1.35	0.60	0.14	13
5	5	0	0	4.92	0	0	0	13.16	0	0	0	100
5	10	0	0	0.97	0	0	0	14.29	1.88	0.29	0.04	10
5	15	0	0	0	0	0	0	16.30	1.46	0.15	0.01	20

TAB. 7 – Résultats pour le plan d'expérience [39] - Descente stochastique

Les tableaux 7 et 8 donnent les résultats pour la descente stochastique et l'algorithme du kangourou. Les résultats obtenus par les recuits simulés sont similaires aux résultats obtenus par l'algorithme du kangourou.

Pour les problèmes de Flow-Shop Hybride avec un étage goulet, nous remarquons que M_{BI} et ET_{BI} sont souvent égaux à zéro, quelle que soit la position de l'étage goulet (premier étage, étage du milieu ou dernier étage) et quel que soit l'algorithme utilisé. Ceci signifie que à chaque réplication de la méthode, nous obtenons la solution optimale et que la borne inférieure correspond à la solution optimale. Nous en avons conclu que lorsqu'un Flow-Shop Hybride comporte un étage goulet, la borne inférieure de [36] est proche ou égale à la solution optimale et que les métaheuristiques utilisées n'ont aucune difficulté pour trouver la solution optimale.

Pour les problèmes de Flow-Shop Hybride sans étage goulet, nous obtenons des résultats différents, M_{BI} et ET_{BI} ne sont pas égaux à zéro : la borne inférieure n'est pas atteinte et pour chaque réplication de la méthode (pour une instance donnée), les résultats obtenus ne sont pas les mêmes. Nous proposons donc de comparer nos résultats avec la solution optimale obtenue en résolvant le modèle mathématique de [14].

La première remarque est que l'algorithme du kangourou obtient de meilleurs résultats (distance relative moyenne et écart type) que la descente stochastique : la descente stochastique converge vers des optima locaux alors que l'algorithme du kangourou permet de quitter les optima locaux. Les résultats

Algorithme du kangourou EC = 1												
m	n	Position de l'étage goulet										
		Début		Milieu		Fin		Aucun				
		M_{BI}	ET_{BI}	M_{BI}	ET_{BI}	M_{BI}	ET_{BI}	M_{BI}	ET_{BI}	M_{OPT}	ET_{OPT}	Nb
2	5	0	0	-	-	0	0	12.70	0	0	0	100
2	10	0	0	-	-	0	0	9.80	0.59	0.14	0.01	67
2	15	0	0	-	-	0	0	4.57	1.05	0.49	0.13	23
3	5	0	0	0	0	0	0	6.90	0	0	0	100
3	10	0	0	0	0	0	0	7.51	0.91	0.06	0.01	80
3	15	0	0	0	0	0	0	5.46	0.90	0.24	0.04	40
5	5	0	0	0	0	0	0	6.82	0	0	0	100
5	10	0	0	0	0	0	0	6.84	0.17	0.08	0.01	70
5	15	0	0	0	0	0	0	7.57	1.37	0.28	0.05	23
EC=7												
m	n	Position de l'étage goulet										
		Début		Milieu		Fin		Aucun				
		M_{BI}	ET_{BI}	M_{BI}	ET_{BI}	M_{BI}	ET_{BI}	M_{BI}	ET_{BI}	M_{OPT}	ET_{OPT}	Nb
2	5	0	0	-	-	0	0	5.09	0	0	0	100
2	10	0	0	-	-	0	0	2.54	0.84	1.46	0.84	57
2	15	0	0	-	-	0	0	0.63	0.21	0.63	0.21	70
3	5	0	0	0	0	0.78	0	1.23	0	0	0	100
3	10	0	0	0	0	0	0	7.09	1.41	0.31	0.06	40
3	15	0.26	0	0	0	1.16	0	4.97	0.94	0.27	0.06	43
5	5	0	0	4.92	0	0	0	13.16	0	0	0	100
5	10	0	0	0.97	0	0	0	13.01	1.48	0.18	0.02	23
5	15	0	0	0	0	0	0	15.63	1.22	0.10	0.01	30

TAB. 8 – Résultats pour le plan d'expérience [39] - Algorithme du kangourou

Problème	m	$M_j, j = 1, m$	n
01	2	1 2	8
02	2	2 1	50
03	3	3 2 3	15
04	3	1 2 2	50
05	3	2 2 2	100
06	5	1 1 2 2 6	50
07	5	3 3 2 3 1	100
08	5	8 3 3 8 7	100
09	10	8 6 6 3 4 8 1 8 9 8	100
10	10	3 3 3 4 4 4 4 2 2 2	200
11	1	10	100
12	10	10 10 10 10 10 10 10 10 10 10	100
13	20	9 8 4 10 9 10 10 4 6 8 10 6 5 3 7 7 3 3 9 9	500
14	3	3 3 3	100
15	5	5 5 5 5 5	100
16	10	5 5 5 5 5 5 5 5 5 5	500
17	2	2 2	50

TAB. 9 – Configurations de Flow-Shop Hybride considérées

obtenus sont bons en moyenne : M_{OPT} est inférieur à 1% et l'écart type est faible : plusieurs réplications de la méthode donnent des résultats similaires.

Pour les deux méthodes, si on compare M_{BI} et M_{OPT}, nous remarquons que M_{BI} est supérieur à M_{OPT}. Pour la descente stochastique, M_{BI} est compris entre 1.23 et 16.30 alors que M_{OPT} est compris entre 0 et 2.56. Pour l'algorithme du kangourou, M_{BI} est compris entre 0.63 et 15.63 alors que M_{OPT} est compris entre 0 et 1.46. Nous pouvons en conclure que lorsque le Flow-Shop Hybride ne comporte pas d'étage goulet, la borne inférieure de [36] peut être très éloignée de la solution optimale.

Pour certaines configurations, le temps de calcul pour la procédure par séparation et évaluation de [39] couplée avec un algorithme génétique est supérieur à deux heures (PC). Les temps de calcul pour la descente stochastique et l'algorithme du kangourou sur une O2 Silicon Graphics sont compris entre moins de 1 seconde et 5 secondes.

Les résultats présentés concernent le système de voisinage $\mathcal{V}(\mathcal{I}_{ij}, \mathcal{A}_1^1)$. Nous obtenons des résultats similaires pour les autres systèmes de voisinage, mais les algorithmes semblent converger plus rapidement avec $\mathcal{V}(\mathcal{I}_{ij}, \mathcal{A}_1^1)$.

5.4.2 Problèmes générés

Dans cette partie, nous présentons des résultats pour des problèmes générés aléatoirement. 17 configurations significatives de Flow-Shop Hybride sont considérées. Elles sont décrites par le tableau 9 et sont composées de 2 à 20 étages, avec de 1 à 10 machines par étage et traitent de 8 à 500 produits. Les temps de traitement sont distribués selon une loi uniforme de paramètres (0,100). Pour chaque configuration, nous avons généré 27 problèmes avec des temps de traitement différents. Dans la suite, P_{xy} représente un problème avec des machines identiques et R_{xy} un problème avec des machines non identiques.

Pour comparer l'efficacité des systèmes de voisinage, nous avons lancé pour chaque configuration et chaque système de voisinage, 25 descentes stochastiques et calculé les distances relatives moyennes par rapport à la borne inférieure. Certains systèmes de voisinage fonctionnent mieux que d'autres. Lorsque la combinatoire n'est pas trop élevée (configurations de 01 à 09, 14, 15 et 17), les systèmes de voisinage $\mathcal{V}(\mathcal{I}_{i_1 i_2}, \mathcal{A}_1^1)$ et $\mathcal{V}(\mathcal{I}_{i_1 i_2}, \mathcal{A}_1^r)$ sont les plus efficaces. Lorsque la combinatoire est plus élevée (configurations 10, 12, 13 et 16), il est plus intéressant d'utiliser le système de voisinage guidé $\mathcal{V}(\mathcal{I}_{i_1 i_2}, \mathcal{H})$. De plus,

nous avons remarqué que certains systèmes de voisinage permettent de faire décroître rapidement le makespan pour ne plus l'améliorer par la suite, alors que d'autres font décroître régulièrement le makespan pour obtenir une meilleure valeur.

5.4.3 Autres problèmes

Des résultats sont présentés pour les deux problèmes industriels et les problèmes suivants dans [13] :
- $FHm((SPM_j)_{j=1}^m)||C_{max}$,
- $FHm((SRM_j)_{j=1}^m)||C_{max}$,
- $FHm((PM_j)_{j=1}^m)|b_{j,j+1}=0|C_{max}$,
- $FHm((RM_j)_{j=1}^m)|b_{j,j+1}=0|C_{max}$,
- $FHm((SPM_j)_{j=1}^m)|no-wait|C_{max}$,
- $FHm((SRM_j)_{j=1}^m)|no-wait|C_{max}$.

6 Conclusion

Notre démarche repose sur la prise en compte de la double complexité des systèmes de production : complexité structurelle et fonctionnelle et complexité algorithmique. La complexité structurelle et fonctionnelle consiste à résoudre un problème d'analyse et d'évaluation des performances alors que la complexité algorithmique consiste à résoudre un problème d'optimisation combinatoire. La majorité des travaux de la littérature ne s'intéressent, en général, qu'à une seule complexité : certains auteurs considèrent uniquement la complexité structurelle et fonctionnelle alors que d'autres ne traitent que la complexité algorithmique. Nous suivons donc une démarche peu fréquente, qui considére les deux complexités en même temps. Les méthodes de résolution proposées reposent sur la notion de couplage entre un modèle pour l'évaluation des performances et une méthode de recherche opérationnelle pour l'optimisation combinatoire.

La conception d'un modèle pour l'évaluation des performances nécessite la construction d'un modèle de connaissance. La méthodologie ASCI proposé par le LIMOS nous a permis de construire un modèle générique de connaissance pour la classe des modèles théoriques étudiés. Pour construire ce modèle, nous avons suivi une décomposition en trois sous-systèmes : sous-système physique, sous-système logique et sous-système décisionnel. Pour spécifier et structurer le sous-système décisionnel, nous avons utilisé un modèle multi-agents qui permet de concevoir des modèles composés d'agents indépendants pouvant évoluer au cours du temps et capables de réagir aux événements. Le fonctionnement des trois sous-systèmes a été spécifié à l'aide de réseaux de Petri temporisés, interprétés et communicants.

La méthodologie ASCI permet à partir du modèle générique de connaissance d'obtenir des modèles d'action pour l'évaluation des performances. Les réseaux de Petri ont été traduits, de manière quasi automatique dans le langage du logiciel QNAP2 pour la construction d'un modèle de simulation à événements discrets. Ce modèle ainsi que le modèle générique de connaissance prennent en compte un grand nombre d'hypothèses, à savoir : stocks de capacité limitée ou nulle, temps de traitement modélisés par des variables aléatoires, dates d'arrivée des produits modélisées par des variables aléatoires, temps de montage et de démontage dépendant de la séquence, ...

Les méthodes de recherche opérationnelle proposées sont principalement des métaheuristiques à base de recuit simulé : descente stochastique, recuit simulé homogène, recuit simulé inhomogène et algorithme du kangourou. L'intérêt de ces méthodes est qu'elles sont faciles à mettre en oeuvre, elles peuvent être couplées sans difficulté avec un modèle pour l'évaluation des performances et on dispose à tout instant d'une solution réalisable. Pour ces méthodes, nous avons proposé des systèmes de voisinage, des plans de réglage et des mécamismes de comparaison pour le contexte stochastique.

Les systèmes de voisinage proposés sont basés sur la concaténation d'un système de voisinage pour l'ordonnancement des produits et d'un système de voisinage pour l'affectation des produits aux machines ou aux lignes. Les mécanismes de comparaison utilisent des moyennes et des intervalles de confiance des critères de performance.

Les perspectives que nous souhaitons donner à ce travail concernent :

- l'extension des modèles markoviens proposés au modèle du Flow-Shop Hybride à deux étages dans un premier temps puis à m étages,
- la poursuite des travaux sur la robustesse dans un Flow-Shop avec la généralisation de la définition d'une solution robuste à d'autres modèles,
- la prise en compte d'autres événements aléatoires tels que les pannes, les temps de montage et démontage modélisés par des variables aléatoires, ...
- la prise en compte d'autres contraintes telles que des contraintes de précédence, des contraintes de routage, des ressources de transport, ... Cette prise en compte passe par l'extension du modèle générique de connaissance,
- l'extension au problème du Job-Shop et au problème du Job-Shop à machines dupliquées. Elle nécessite la construction d'un modèle générique de connaissance qui étend le modèle construit pour la classe des FS((H)H), l'extension des algorithmes de résolution des agents pour tenir compte du fait que les produits ne suivent pas tous la même gamme, et l'adaptation des systèmes de voisinage pour les métaheuristiques...

Références

[1] E.H. Aghezzaf, A. Artiba, O. Moursli, and C. Tahon. Hybrid flowshop problems, a decomposition based heuristic approach. In *Proceedings of International Conference on Industrial Engineering and Production Management (IEPM'95), Marrakech, FUCAM/IFIP/INRIA*, pages 43–56, 1995.

[2] E. Bonomi and J.L. Lutton. The n-city travelling salesman problem : Statistical mechanics and the metropolis algorithm. *SIAM revue*, 26 :551–568, 1984.

[3] V. Botta-Genoulaz. *Planification et ordonnancement d'une succession d'ateliers avec contraintes - vers un atelier de génie décisionnel pour le pilotage des systèmes de production.* PhD thesis, Université Claude Bernard, Lyon I, 1996.

[4] S.A. Brah and J.L.Hunsucker. Branch and bound algorithm for the flowshop with multiple processors. *European Journal of Operational Research*, 51 :88–99, 1991.

[5] A. Brandwajn. Equivalence et décomposition dans les modèles à files d'attente et leur application à l'évaluation des performances de systèmes d'exploitations. *Thèse d'état, Université de Paris VI*, 1975.

[6] J. Carlier and P. Chrétienne. *Problèmes d'ordonnancement.* Masson, Paris, 1988.

[7] G. Chevalier, J. Barrier, and P. Richard. Production planning in the glass industry. *Revue VERRE*, 2(5) :27–29, 1996.

[8] D.E. Deal and J.L. Hunsucker. The two-stage flow-shop problem with m machines at each stage. *Journal of Information and Optimization Sciences*, 12 :407–417, 1991.

[9] G. Fleury. *Méthodes stochastiques et déterministes pour les problèmes NP-difficiles.* Phd. thesis, Université Blaise Pascal, Clermont-Ferrand II, 1993.

[10] P. Fortemps, C. Ost, M. Pirlot, J. Teghem, and D. Tuyttens. Using metaheuristics for solving a production scheduling problem in a chemical firm. *International Journal of Production Economics*, 46-47(1-3) :13–26, 1996.

[11] GOTHA. Les problèmes d'ordonnancement. *RAIRO-Recherche opérationnelle / Operation research*, 27(1) :77–150, 1993.

[12] M. Gourgand, N. Grangeon, and S. Norre. Metaheuristics for the deterministic hybrid flow shop problem. *RAIRO - APII - JESA*, 34(9) :1107–1135, 2000.

[13] N. Grangeon. *Métaheuristiques et Modèles d'Evaluation pour le Problème du Flow-Shop Hybride Hiérarchisé.* PhD thesis, Université Blaise Pascal, Clermont-Ferrand II, 2001.

[14] A. Guinet, M.M. Solomon, P.K. Kedia, and A. Dussauchoy. A computational study of heuristics for two-stage flexible flow-shops. *International Journal of Production Research*, 34(5) :1399–1416, 1996.

[15] J.N.D. Gupta. Two stage hybrid flow shop scheduling problem. *Journal of the Operations Research Society*, 39(4) :359–364, 1988.

[16] J.N.D. Gupta and E.A Tunc. Schedules for a two stage hybrid flow shop with parallel machines at the second stage. *International Journal of Production Research*, 29(7) :1489–1502, 1991.

[17] J.K. Hao, P. Galinier, and M. Habib. Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes. *Revue d'Intelligence Artificielle*, 13(2) :283–324, 1999.

[18] A. Ignall and L. Schrage. Application of the branch and bound technique to some flow-shop scheduling problems. *Operation Research*, 13(3) :400–412, 1965.

[19] A.H.G. Rinnooy Kan. *Machine scheduling problem : classification, complexity and computations.* Martinus Nijhoff, The Hague, Netherlands, 1976.

[20] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *IBM Research report*, RC 9355, 1982.

[21] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 20 :671–680, 1983.

[22] P. Lacomme. *Optimisation des systèmes de production : méthodes stochastiques et approche multi-agents.* Phd. thesis, Université Blaise Pascal, Clermont-Ferrand II, 1998.

[23] C.Y. Lee and G.L. Vairaktarakis. Minimizing makespan in hybrid flow-shops. *Operations Research Letters*, 16(3) :149–158, 1994.

[24] S. Li. A hybrid two stage flow shop with part family, bath production, major and minor set ups. *European Journal of Operational Research*, 102 :142–156, 1997.

[25] S. Nahar, S. Shani, and E. Shragowitz. Experiments with simulated annealing. *Proc. 22nd Des. Automation Conf., Las Vegas*, pages 748–752, 1985.

[26] S.L. Narasimhan and P.M. Mangiameli. A comparison of sequencing rules for two stage hybrid flow shop. *Decision Sciences*, 18 :250–265, 1987.

[27] S.L. Narasimhan and S.S. Panwalkar. Scheduling in a two stage manufacturing process. *International Journal of Production Research*, 22(4) :555–564, 1984.

[28] E.G. Negenman. Local search algorithms for the multiprocessor flow shop scheduling problem. *European Journal of Operational Research*, 128 :147–158, 2001.

[29] E. Nowicki and C. Smutnicki. The flow shop with parallel machines : a tabu search approach. *European Journal of Operational Research*, 106 :226–253, 1998.

[30] R.J. Paul. A production scheduling problem in the glass container industry. *Operation Research*, 27(2) :290–302, 1979.

[31] A. Quéré. *Gestion et optimisation des flux matières dans les systèmes de production de type flow shop hybride.* Phd. thesis, Université Blaise Pascal, Clermont-Ferrand II, 1998.

[32] C. Rajendran and D. Chaudhuri. A multi-stage parallel-processor flowshop problem with minimum flowtime. *European Journal of Operational Research*, 57 :111–122, 1992.

[33] C. Rajendran and D. Chaudhuri. Scheduling in n-job, m-stage flow-shop with parallel processors to minimize makespan. *International Journal of Production Research*, 27 :137–143, 1992.

[34] F. Riane. *Scheduling hybrid flow shop : Algorithms and Applications.* PhD thesis, Facultés Universitaires Catholiques de Mons, 1998.

[35] F.A. Rodammer and K. Preston White. A recent survey of production scheduling. *IEEE transaction on systems, man and cybernetics*, 6(18), 1988.

[36] D.L. Santos, J.L. Hunsucker, and D.E. Deal. Global lower bounds for flow shop with multiple processors. *European Journal of Operational Research*, 80 :112–120, 1995.

[37] H.D. Sherali, S.C. Sarin, and M.S. Kodialam. Models and algorithm for a a two stage production process. *Production Planning and Control*, 1(1) :27–39, 1990.

[38] P.J.M. van Laarhoven and A.H.L. Aarts. *Simulated Annealing : Theory and Applications*. Kluwer Academic Publishers, 1987.

[39] A. Vignier. *Contribution à la résolution de problèmes d'ordonnancement de type monogamme, multimachines (Flow shop hybride)*. PhD thesis, Université de Tours, 1997.

[40] K.S. Wang, H.W. Hsia, and Z.D. Zhuang. An intelligent decision system for a modern manufacturing system. *International Journal of Computer Integrated Manufacturing*, 5(6) :281–292, 1993.