Ecole Nationale Supérieure des Mines
SAINT-ETIENNE

N° d'ordre : 454 GI

THÈSE

présentée par

Olga Guschinskaya

Pour obtenir le grade de Docteur
de l'École Nationale Supérieure des Mines de Saint-Étienne

Spécialité : Génie Industriel

Outils d'aide à la décision pour la conception en avant-projet des systèmes d'usinage à boîtiers multibroches

Soutenue à Saint-Étienne le 27 novembre 2007

Membres du jury

Président :	Pierre Villon	Professeur/Université de Technologie de Compiègne
Rapporteurs :	Jean-Charles Billaut	Professeur/Université François-Rabelais
	Gerd Finke	Professeur/Université Joseph Fourier
	Michel Minoux	Professeur/Université Paris-6
Examinateurs :	Daniel Brissaud	Professeur/Université Joseph Fourier
	Pierre Lopez	Chargé de recherche/LAAS-CNRS
	Patrick Martin	Professeur/ENSAM
Directeur de thèse :	Alexandre Dolgui	Professeur/ENSM-SE
Invité :	Damien Poyard	Directeur Commercial/PCI-SCEMM, Saint-Etienne

● **Spécialités doctorales :** **Responsables :**

SCIENCES ET GENIE DES MATERIAUX	**J. DRIVER** Directeur de recherche – Centre SMS
MECANIQUE ET INGENIERIE	**A. VAUTRIN** Professeur – Centre SMS
GENIE DES PROCEDES	**G. THOMAS** Professeur – Centre SPIN
SCIENCES DE LA TERRE	**B. GUY** Maitre de recherche – Centre SPIN
SCIENCES ET GENIE DE L'ENVIRONNEMENT	**J. BOURGOIS** Professeur – Centre SITE
MATHEMATIQUES APPLIQUEES	**E. TOUBOUL** Ingénieur – Centre G2I
INFORMATIQUE	**O. BOISSIER** Professeur – Centre G2I
IMAGE, VISION, SIGNAL	**JC. PINOLI** Professeur – Centre CIS
GENIE INDUSTRIEL	**P. BURLAT** Professeur – Centre G2I
MICROELECTRONIQUE	**Ph. COLLOT** Professeur – Centre CMP

● **Enseignants-chercheurs et chercheurs autorisés à diriger des thèses de doctorat** (titulaires d'un doctorat d'Etat ou d'une HDR)

BATTON-HUBERT	Mireille	MA	Sciences & Génie de l'Environnement	SITE
BENABEN	Patrick	PR 2	Sciences & Génie des Matériaux	SMS
BERNACHE-ASSOLANT	Didier	PR 1	Génie des Procédés	CIS
BIGOT	Jean-Pierre	MR	Génie des Procédés	SPIN
BILAL	Essaïd	DR	Sciences de la Terre	SPIN
BOISSIER	Olivier	PR 2	Informatique	G2I
BOUDAREL	Marie-Reine	MA	Sciences de l'inform. & com.	DF
BOURGOIS	Jacques	PR 1	Sciences & Génie de l'Environnement	SITE
BRODHAG	Christian	MR	Sciences & Génie de l'Environnement	SITE
BURLAT	Patrick	PR 2	Génie industriel	G2I
CARRARO	Laurent	PR 1	Mathématiques Appliquées	G2I
COLLOT	Philippe	PR 1	Microélectronique	CMP
COURNIL	Michel	PR 1	Génie des Procédés	SPIN
DAUZERE-PERES	Stéphane	PR 1	Génie industriel	CMP
DARRIEULAT	Michel	ICM	Sciences & Génie des Matériaux	SMS
DECHOMETS	Roland	PR 1	Sciences & Génie de l'Environnement	SITE
DESRAYAUD	Christophe	MA	Mécanique & Ingénierie	SMS
DELAFOSSE	David	PR 2	Sciences & Génie des Matériaux	SMS
DOLGUI	Alexandre	PR 1	Génie Industriel	G2I
DRAPIER	Sylvain	PR 2	Mécanique & Ingénierie	CIS
DRIVER	Julian	DR	Sciences & Génie des Matériaux	SMS
FOREST	Bernard	PR 1	Sciences & Génie des Matériaux	CIS
FORMISYN	Pascal	PR 1	Sciences & Génie de l'Environnement	SITE
FORTUNIER	Roland	PR 1	Sciences & Génie des Matériaux	CMP
FRACZKIEWICZ	Anna	MR	Sciences & Génie des Matériaux	SMS
GARCIA	Daniel	CR	Génie des Procédés	SPIN
GIRARDOT	Jean-Jacques	MR	Informatique	G2I
GOEURIOT	Dominique	MR	Sciences & Génie des Matériaux	SMS
GOEURIOT	Patrice	MR	Sciences & Génie des Matériaux	SMS
GRAILLOT	Didier	DR	Sciences & Génie de l'Environnement	SITE
GROSSEAU	Philippe	MR	Génie des Procédés	SPIN
GRUY	Frédéric	MR	Génie des Procédés	SPIN
GUILHOT	Bernard	DR	Génie des Procédés	CIS
GUY	Bernard	MR	Sciences de la Terre	SPIN
GUYONNET	René	DR	Génie des Procédés	SPIN
HERRI	Jean-Michel	PR 2	Génie des Procédés	SPIN
KLÖCKER	Helmut	MR	Sciences & Génie des Matériaux	SMS
LAFOREST	Valérie	CR	Sciences & Génie de l'Environnement	SITE
LE COZE	Jean	PR 1	Sciences & Génie des Matériaux	SMS
LI	Jean-Michel	EC (CCI MP)	Microélectronique	CMP
LONDICHE	Henry	MR	Sciences & Génie de l'Environnement	SITE
MOLIMARD	Jérôme	MA	Sciences & Génie des Matériaux	SMS
MONTHEILLET	Frank	DR 1 CNRS	Sciences & Génie des Matériaux	SMS
PERIER-CAMBY	Laurent	PR1	Génie des Procédés	SPIN
PIJOLAT	Christophe	PR 1	Génie des Procédés	SPIN
PIJOLAT	Michèle	PR 1	Génie des Procédés	SPIN
PINOLI	Jean-Charles	PR 1	Image, Vision, Signal	CIS
STOLARZ	Jacques	CR	Sciences & Génie des Matériaux	SMS
SZAFNICKI	Konrad	CR	Sciences de la Terre	SITE
THOMAS	Gérard	PR 1	Génie des Procédés	SPIN
VALDIVIESO	Françoise	CR	Génie des Procédés	SPIN
VALDIVIESO	François	MA	Sciences & Génie des Matériaux	SMS
VAUTRIN	Alain	PR 1	Mécanique & Ingénierie	SMS
VIRICELLE	Jean-Paul	MR	Génie des procédés	SPIN
WOLSKI	Krzysztof	CR	Sciences & Génie des Matériaux	SMS
XIE	Xiaolan	PR 1	Génie industriel	CIS

Glossaire :

PR 1 Professeur 1ère catégorie
PR 2 Professeur 2ème catégorie
MA(MDC) Maître assistant
DR (DR1) Directeur de recherche
Ing. Ingénieur
MR(DR2) Maître de recherche
CR Chargé de recherche
EC Enseignant-chercheur
ICM Ingénieur en chef des mines

Centres :

SMS Sciences des Matériaux et des Structures
SPIN Sciences des Processus Industriels et Naturels
SITE Sciences Information et Technologies pour l'Environnement
G2I Génie Industriel et Informatique
CMP Centre de Microélectronique de Provence
CIS Centre Ingénierie et Santé

Ça c'est la caisse. Le mouton que tu veux est dedans.

« *Le petit prince* », *Antoine de Saint-Exupéry*

.

Remerciements

Je tiens, tout d'abord, à remercier chaleureusement Monsieur Alexandre Dolgui, mon directeur de thèse, de m'avoir accordé sa confiance au cours de ces années et d'avoir encadré ce travail de thèse, avec beaucoup de compétence et d'enthousiasme.

J'exprime ma profonde gratitude aux membres du jury, qui ont accepté d'évaluer mon travail de thèse. Merci à Monsieur Jean-Charles Billaut, Professeur à l'Université François-Rabelais, Monsieur Gerd Finke, Professeur à l'Université Joseph Fourier et Monsieur Michel Minoux, Professeur à l'Université Paris-6, d'avoir accepté d'être rapporteurs de ce manuscrit. Merci à Monsieur Pierre Villon, Professeur à l'Université de Technologie de Compiègne d'avoir accepté de présider le jury de cette thèse. Merci également à Monsieur Daniel Brissaud, Professeur/Université Joseph Fourier, Monsieur Pierre Lopez, Chargé de recherche au LAAS-CNRS et Monsieur Patrick Martin, Professeur à l'ENSAM, pour avoir accepté d'examiner mon mémoire et de faire partie de mon jury de thèse. Je suis également reconnaissante à Monsieur Damien Poyard, Directeur Commercial à PCI-SCEMM, Saint-Etienne, d'avoir accepté représenter notre partenaire industriel lors de cette soutenance de thèse.

Je remercie tous les membres du centre G2I et, plus particulièrement, l'équipe MSGI pour leur accueil et leur soutien. Je tiens également à exprimer ma gratitude à Jean-François, Marie Line, Liliane et Gabrielle pour leur gentillesse et leur aide administrative, logistique et technique.

Finalement, j'adresse un grand merci à toute ma famille qui a toujours été présente lorsque j'en ai eu besoin et à tous mes amis qui m'ont fait découvrir la beauté de la vie.

Sommaire

Liste des figures

Liste des tableaux

Liste des algorithmes

Liste des notations

$B(k)$	l'intervalle des indices des blocs réservés pour le poste k
C_1	le coût relatif d'un poste de travail
C_2	le coût relatif d'un bloc
D_i	le diamètre de l'outil pour l'opération i
E^p	la collection de sous-ensembles $e \subset \mathbf{N}$ d'opérations, tels qu'au moins une opération de chaque ensemble e ne doit pas être affectée au même poste que les autres
E^b	la collection de sous-ensembles $e \subset \mathbf{N}$ d'opérations, tels qu'au moins une opération de chaque ensemble e ne doit pas être affectée au même bloc que les autres
$[f_{n1}(i), f_{n2}(i)]$	l'intervalle des valeurs admissibles de l'avance par tour pour l'opération i
$G^{pr} = (\mathbf{N}, D^{pr})$	un graphe orienté de précédence, dans lequel un arc $\{i, j\} \in D$ si l'opération i ne peut pas être exécutée après l'exécution de l'opération j
I^p	une collection de sous-ensembles $e \subset \mathbf{N}$ d'opérations, tels que toutes les opérations appartenant à l'ensemble e doivent être impérativement affectées au même poste de travail
k	l'indice pour les postes de travail, $k = 1, 2, ..., m$
$K(j) = [k_j^-, k_j^+]$	l'intervalle des indices k des postes où l'opération j peut être affectée
$K_E(j)$	l'ensemble des indices des postes où les opérations liées par des contraintes d'exclusion avec l'opération j peuvent être affectées
$L(N)$	la longueur de la course du boîtier effectuant l'ensemble d'opérations N
l_i	la longueur de la course d'outil, nécessaire pour l'opération i
m	le nombre de postes de travail d'une solution
m_0	le nombre maximum autorisé de postes de travail dans une solution

$\mathbf{N}$	l'ensemble des opérations d'usinage à effectuer, où $i \in \mathbf{N}$ correspond à l'opération ayant le numéro i
N_k	l'ensemble des opérations affectées au poste de travail avec l'indice k
N_{kr}	l'ensemble des opérations affectées au bloc r du poste de travail k
$n_{\|\|}$	le nombre maximum autorisé de boîtiers parallèles sur un poste de travail
$n_{\exists}$	le nombre maximum autorisé de boîtiers en séquence sur un poste de travail
n_0	le nombre maximum autorisé de boîtiers sur un poste de travail
$O_b(q)$	l'ensemble des opérations qui peuvent être affectées au bloc q
$O_E(j)$	l'ensemble des opérations liées avec l'opération $j \in \mathbf{N}$ par une contrainte d'exclusion soit au niveau des blocs, soit au niveau des postes de travail
$O_{EB}(j)$	l'ensemble des opérations liées par une contrainte d'exclusion au niveau des blocs avec l'opération j
$O_{EB*}(j)$	l'ensemble des opérations liées par une contrainte d'exclusion au niveau des blocs avec l'opération j, or la contrainte correspondante contient d'autres opérations que les opérations i et j, par conséquent, il n'est pas possible de savoir d'avance si les opérations i et j seront affectées au même bloc dans la solution optimale ou non
$O_{EB2}(j)$	l'ensemble des opérations avec lesquelles l'opération j ne peut pas être affectée au même bloc
$O_{EP}(j)$	l'ensemble des opérations liées par une contrainte d'exclusion au niveau des postes de travail avec l'opération j
$O_{EP*}(j)$	l'ensemble des opérations liées par une contrainte d'exclusion au niveau des postes avec l'opération j, or la contrainte correspondante contient d'autres opérations que les opérations i et j, par conséquent, il n'est pas possible de savoir d'avance si les opérations i et j seront affectées au même poste de travail dans la solution optimale ou non
$O_{EP2}(j)$	l'ensemble des opérations avec lesquelles l'opération j ne peut pas être affectée au même poste
$O_p(k)$	l'ensemble des opérations qui peuvent être affectées au poste k
$OMP(j)$	l'ensemble des opérations qui doivent être affectées au même poste que l'opération j
OS	« Order Strength », ce paramètre est utilisé comme mesure de la densité du graphe de précédence

P	la collection des données déterminant un problème d'optimisation de la configuration d'un système d'usinage à boîtiers multibroches
$PredD(j)$	l'ensemble des prédécesseurs directs de l'opération $j \in \mathbf{N}$
$PredT(j)$	l'ensemble de tous les prédécesseurs de l'opération $j \in \mathbf{N}$
$Q(j) = [q_j^-, q_j^+]$	l'ensemble des indices de blocs où l'opération j peut être affectée
$Q_E(j)$	l'ensemble des indices des blocs où les opérations liées par des contraintes d'exclusion avec l'opération j peuvent être affectées
r	l'indice pour les blocs d'un poste de travail, pour le poste $k : r = 1, 2, ..., r_k$
r_k	le nombre de blocs du poste de travail k
S	une solution du problème P
$SuccD(j)$	l'ensemble des successeurs directs de l'opération $j \in \mathbf{N}$
$SuccT(j)$	l'ensemble de tous les successeurs de l'opération $j \in \mathbf{N}$
T_0	le temps de cycle d'usinage
$t^b(N)$	le temps d'usinage du boîtier effectuant l'ensemble d'opérations N
$t^p(N)$	le temps d'usinage sur le poste de travail effectuant l'ensemble d'opérations N
$[v_{c1}(i), v_{c2}(i)]$	l'intervalle des valeurs admissibles de la vitesse de coupe pour l'opération i
$\mathbf{V}(N) = [V_1(N), V_2(N)]$	l'intervalle des valeurs admissibles de la vitesse d'avance du boîtier réalisant l'ensemble d'opérations N
$v_f(N)$	la valeur de la vitesse d'avance du boîtier réalisant l'ensemble d'opérations N
τ^p	le temps auxiliaire au niveau d'un poste de travail
τ^b	le temps auxiliaire au niveau d'un bloc

Introduction générale

Dans un contexte économique mondial avec une concurrence exacerbée, les entreprises fabriquant des systèmes d'usinage se doivent de concevoir leurs produits plus rapidement, plus efficacement et moins cher que les concurrents tout en proposant des solutions innovantes à leurs clients. Pour ces équipementiers, la phase cruciale de la conception se trouve, sans doute, à l'étape d'avant-projet, car les devis qui en sont issus sont transmis aux clients et ont un impact prépondérant sur la décision d'attribution du contrat. Afin de proposer à son client une solution pertinente, efficace et conforme à ses attentes, il est nécessaire d'examiner un large éventail de solutions possibles. Le nombre d'éléments à prendre en considération rend cette phase d'analyse extrêmement complexe. D'autant que l'examen d'une solution prometteuse doit être suffisamment précis afin de vérifier sa cohérence, sa fiabilité, et éviter une découverte fortuite de fautes de conception à l'étape de fabrication.

Dans ces circonstances, il devient impératif pour les équipementiers de recourir à des outils d'aide à la décision permettant d'utiliser efficacement le temps du concepteur et de le guider vers des solutions optimales pour l'ensemble des paramètres connus à l'étape d'avant-projet. Les travaux dont les résultats sont présentés dans ce mémoire s'inscrivent dans la démarche de développement de tels outils. Nous nous sommes intéressés plus particulièrement à la problématique de la conception en avant-projet des systèmes d'usinage à boîtiers multi-broches, utilisés pour la production en grande série. Ce choix a été motivé par un réel besoin industriel. À l'heure actuelle, la fabrication des systèmes d'usinage de ce type représente 40% de la production de notre partenaire industriel PSI-SCEMM (Saint-Étienne), considéré comme leader sur le marché français des systèmes d'usinage. Il est à constater également que la globalisation et la standardisation de la production actuelles appellent à penser que les systèmes destinés à la production en grande série auront toujours une place importante dans l'industrie. Or, malgré leur importance économique, les systèmes d'usinage à boîtiers multi-broches ont bénéficié de très peu d'attention dans les études académiques et, par conséquent, jusqu'à maintenant leur conception s'appuie essentiellement sur le savoir-faire des ingénieurs.

Cette thèse a été réalisée dans le cadre de développement d'une méthodologie apte à constituer la charpente d'un outil d'aide à la décision pour la conception en avant-projet des systèmes d'usinage à boîtiers multibroches. C'est en analysant les méthodes existantes et utilisées à différentes étapes de la conception que nous avons pu constater l'absence de méthodes de résolution adaptées au problème d'optimisation de la configuration des systèmes d'usinage à boîtiers multibroches. Ceci représentait un manque très important, car le coût et l'efficacité des systèmes d'usinage de de type dépendent fortement de la qualité des résultats obtenus à cette étape de conception. Par conséquent, le but principal de ce mémoire est

de présenter des modèles mathématiques et des méthodes de résolution dédiés au problème d'optimisation de la configuration des systèmes d'usinage à boîtiers multibroches.

Nous pouvons formuler ce problème comme suit : pour une pièce donnée, trouver une configuration optimale du système d'usinage correspondant en satisfaisant des contraintes technologiques, techniques et économiques et en visant à minimiser le coût total du système. Plus particulièrement, nous nous intéressons aux problèmes d'optimisation issus de la configuration des trois types de systèmes d'usinage : machines de transfert, machines à table mobile et machines à table circulaire pivotante. Malgré les similitudes de ces problèmes avec le problème connu dans la littérature de la recherche opérationnelle sous le nom Assembly Line Balancing Problem (ALBP), les problèmes soulevés possèdent des propriétés plus complexes et des contraintes propres aux systèmes d'usinage. Pour résoudre ces problèmes de manière efficace, nous proposons un certain nombre de modèles et de techniques d'optimisation, notamment des algorithmes de calcul des bornes inférieures, des procédures de pré-traitement, des méthodes de résolution exacte et des algorithmes approchées (heuristiques et métaheuristiques). Pour présenter les études effectuées et les principaux résultats obtenus, nous avons structuré ce manuscrit en 7 chapitres. Dans ce qui suit, nous donnons un aperçu de leur contenu.

Le Chapitre 1 a pour but de familiariser le lecteur avec le domaine d'usinage et la configuration des systèmes d'usinage à boîtiers multibroches. Sans aller dans les détails techniques, nous introduisons les éléments de base des systèmes d'usinage et les notions qui seront utilisées dans la suite du manuscrit.

Le Chapitre 2 aborde la problématique de la conception en avant-projet. Nous faisons ressortir les différentes étapes mises en œuvre durant cette phase et les diverses activités décisionnelles qui en découlent. Nous nous servons également de cette analyse pour pouvoir positionner le problème d'optimisation qui nous intéresse, ainsi que pour déterminer les décisions qui doivent être prises en amont et en aval de celui-ci. À cette fin, nous exposons notamment des méthodes élaborées dans la littérature pour la modélisation de pièces mécaniques et la détermination des processus d'usinage. Ensuite, nous formulons, de manière mathématique, le problème d'optimisation de la configuration d'un système d'usinage à boîtiers multibroches en proposant un modèle générique. Par la suite, ce modèle sera utilisé comme une base pour le développement de modèles mathématiques dédiés à chaque type de systèmes d'usinage considéré.

Le Chapitre 3 offre une synthèse des principales méthodes développées dans la littérature pour la résolution des problèmes d'optimisation proches, à savoir : ceux de la conception et de l'équilibrage des chaînes d'assemblage. Nous proposons une classification des problèmes traités et mettons en évidence l'impossibilité d'appliquer, au moins directement, les méthodes existantes pour le cas de la conception des systèmes d'usinage à boîtiers multibroches. Ceci justifie notamment la nécessité du développement de nouvelles méthodes de résolution adaptées aux particularités de ces systèmes et du domaine d'usinage.

Les Chapitres 4 et 5 sont dédiés à la résolution du problème d'optimisation de la configuration des machines de transfert. Dans le Chapitre 4, nous expliquons le mode de fonctionnement de ce type de systèmes d'usinage. Ensuite, nous formulons le problème d'optimisation. Ce problème étant NP-difficile, le temps de calcul nécessaire pour sa résolution exacte croît de manière exponentielle avec l'augmentation de sa taille. Par conséquent, il est impossible

d'obtenir des solutions optimales pour les instances de grande taille en temps raisonnable. Nous utilisons deux stratégies différentes, exposées dans les Chapitres 4 et 5, pour pallier à cet inconvénient et réduire le temps de calcul.

Dans le Chapitre 4, nous développons des procédures de pré-traitement en vue d'améliorer les performances des méthodes exactes. À travers une série de tests numériques, nous mettons en évidence la réduction du nombre de variables et de contraintes par le biais de l'application de ces procédures avant la phase de résolution. En réduisant ainsi la taille du problème à résoudre, nous pouvons obtenir des solutions optimales pour une gamme de problèmes plus large et avec des temps de calcul plus courts.

Dans le Chapitre 5, nous élaborons des méthodes approchées. Ce développement se fait en plusieurs étapes. Tout d'abord, nous proposons plusieurs variantes d'une heuristique basée sur l'approche Monte Carlo. Ensuite, nous dotons la meilleure variante de cette heuristique d'une phase complémentaire qui permet d'améliorer davantage les solutions. Pour ce faire, nous mettons en place une décomposition dynamique des solutions et faisons appel à une résolution exacte des sous-problèmes ainsi obtenus. Enfin, nous élaborons une métaheuristique de type GRASP. Toutes ces approches sont évaluées sur un échantillon de problèmes, dont la structure est équivalente à celle des problèmes réels existant sur le terrain industriel.

Dans le Chapitre 6, nous présentons nos contributions pour la résolution du problème d'optimisation de la configuration des machines à table mobile et des machines à table circulaire pivotante. Nous exposons le mode de fonctionnement de ces systèmes d'usinage et leurs différences par rapport aux machines de transfert. Ensuite, nous développons des modèles mathématiques pour les problèmes d'optimisation correspondants et proposons des méthodes exactes pour leur résolution.

Dans le Chapitre 7, nous présentons un prototype de logiciel d'aide à la décision supportant les différentes étapes de conception en avant-projet des systèmes d'usinage à boîtiers multibroches. Au cœur de ce logiciel se trouvent les modèles et les outils mathématiques développés dans ce mémoire. Nous illustrons les avantages de son utilisation dans le cadre de conception d'une machine de transfert pour la fabrication d'un carter de moteur. Les données pour cet exemple nous ont été fournies par notre partenaire industriel, PSI-SCEMM (Saint-Étienne). Au cours de cette étude, nous avons validé différents aspects de la méthodologie développée et nous avons confronté les algorithmes proposés à un problème de taille réelle.

Enfin, nous terminons ce mémoire par les conclusions générales sur le travail effectué et nous présentons les perspectives de notre recherche.

Chapitre 1

Systèmes d'usinage à boîtiers multibroches

Lorsque nous regardons un objet, nous sommes habitués à utiliser nos yeux en nous souvenant uniquement de ce que toutes les autres personnes qui nous ont précédé ont pensé à son sujet.

Guy De Maupassant

Dans ce chapitre, nous décrivons les systèmes d'usinage et leurs éléments de base. En particulier, nous introduisons les termes et concepts issus du domaine d'usinage, nécessaires pour la bonne compréhension des chapitres suivants. Nous commençons par introduire les différentes opérations d'usinage et leurs paramètres. Ensuite, nous présentons les types de systèmes d'usinage les plus utilisés et leurs éléments de base. Enfin, nous mettons l'accent sur les caractéristiques des systèmes d'usinage concernés par notre étude.

1.1 Opérations d'usinage et leurs paramètres

1.1.1 Définition de l'opération d'usinage

L'usinage des métaux, procédé de fabrication dont l'origine est très ancienne, a connu une grande expansion avec la mécanisation de plus en plus poussée des ateliers de fabrication. Malgré l'apparition de nouvelles techniques de fabrication au cours des dernières années, il reste à l'heure actuelle l'un des procédés les plus utilisés. Par exemple, le volume des dépenses faites aux U.S.A. pour cette seule technique de fabrication représente 5% du produit national brut [Cetim, 2000].

L'usinage d'une pièce consiste à passer de son état brut (*pièce brute*) à son état fini (*pièce usinée* ou *finie*). Ce passage est assuré par une succession d'actions appelée *processus d'usinage*. Ces actions sont réalisées en utilisant différents moyens du système d'usinage, notamment les *outils de coupe*. Un outil de coupe permet d'enlever de la matière à une pièce

dans le but de générer une nouvelle surface. La Figure 1.1 [www1, Octobre, 2007] montre un exemple d'une pièce avant et après l'usinage.

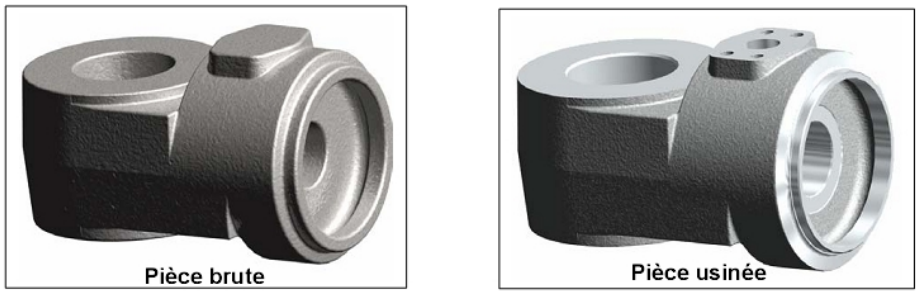


FIG. 1.1 – Deux états de la pièce : pièce brute et pièce usinée

Une action au cours de laquelle un outil produit une forme géométrique est appelée une *opération d'usinage*. Pour l'effectuer, deux configurations sont possibles : fixer la pièce et déplacer l'outil ou faire tourner la pièce alors que l'outil reste figé (ce dernier type de configuration est utilisé dans les tours et centres de tournage). Dans ce mémoire, nous nous limitons au cas de la première configuration.

1.1.2 Types d'opérations d'usinage

Dans le cas d'usinage où la pièce est fixée, l'opération d'usinage consiste à imposer à l'outil une loi de déplacement précise par rapport à la pièce à usiner. Le processus d'usinage d'une pièce est généralement composé de trois phases : ébauche, demi-finition et finition ; chacune de ces phases est caractérisée par différents types d'opérations. L'ébauche permet d'enlever un maximum de matière en un minimum de temps. Pour l'effectuer, on emploie d'importants efforts de coupe et des outils pouvant y résister et ayant, par conséquent, des diamètres élevés. La finition est l'usinage final d'une surface, et on cherche le plus souvent une bonne qualité de surface : dimensions, forme et rugosité dans les tolérances données. Les efforts sont alors plus faibles que pour une ébauche, les outils utilisés sont donc plus minces. La phase de demi-finition englobe les opérations intermédiaires entre l'ébauche et la finition.

De façon générale, toute opération d'usinage peut être décrite par les paramètres suivants : l'outil de coupe, sa trajectoire et les conditions de coupe. Les opérations d'usinage peuvent être classées en fonction de la forme géométrique produite par l'outil. Dans ce mémoire, nous considérons deux classes d'opérations : les opérations de fraisage et de perçage. Ce sont les opérations de ces deux classes qui sont, généralement, effectuées par les systèmes d'usinage à boîtiers multibroches. Nous présentons de manière synthétique les différentes formes géométriques et les outils qui leur sont associés tout en mettant l'accent sur les paramètres d'usinage. Ce sont ces derniers qui jouent un rôle important lors de la conception des boîtiers multibroches.

1.1.3 Opérations de perçage

Cette classe d'opérations regroupe l'ensemble des opérations liées à l'usinage des différents trous, telles que pointage, perçage, alésage, taraudage, etc. (voir Figure 1.2 [Ruby et Maranzana, 2007]). Habituellement, les opérations de perçage sont effectuées par une coupe orthogonale caractérisée par un seul arrêt d'usinage et des trajectoires d'usinage simples.

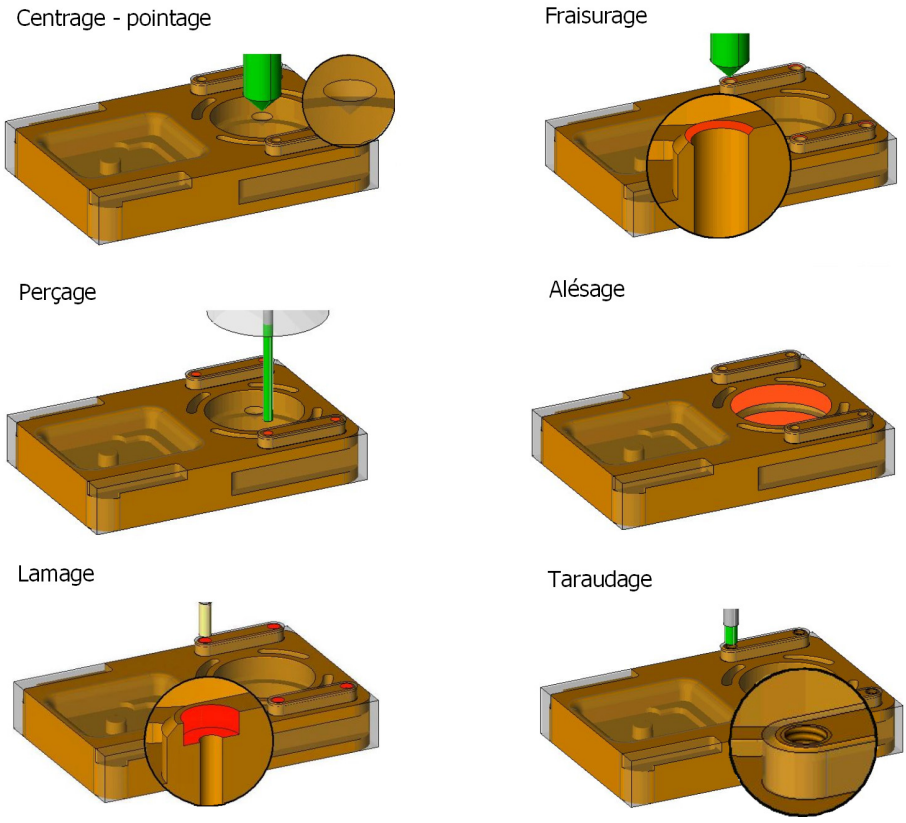


FIG. 1.2 – Types d'opérations de perçage

Une opération de perçage combine deux mouvements d'outil : une rotation et un déplacement linéaire (Figure 1.3 [Coromant Sandvik®, 2006]).

Dans ce qui suit, nous recensons les différents paramètres utilisés pour décrire ces deux mouvements.

La vitesse de broche n (en tr/min) est la vitesse à laquelle s'effectue le mouvement de rotation de l'outil.

La vitesse de coupe v_c (en m/min) est déterminée, en perçage, par la vitesse périphérique de l'élément en rotation et peut être facilement calculée à partir du nombre de tours que la broche effectue par minute. À chaque tour, la périphérie du foret décrit un cercle de circonférence πD, où D est le diamètre de l'outil :

$$v_c = \pi D n \tag{1.1}$$

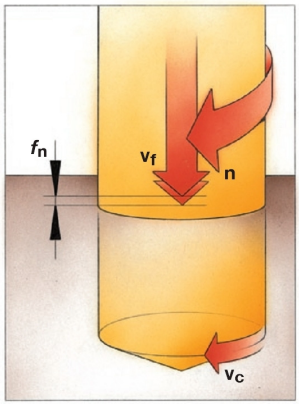


FIG. 1.3 – Paramètres de l'opération de perçage

L'avance par tour f_n (en mm/tr) exprime le déplacement linéaire effectué par l'outil à chaque tour de rotation et sert à calculer la vitesse d'avance v_f.

La vitesse d'avance ou **de pénétration** v_f (en mm/min) correspond au déplacement de l'outil par rapport à la pièce, exprimé en longueur par unité de temps. Elle est également désignée sous le nom d'**avance**, tout simplement, ou d'**avance de table** :

$$v_f = f_n n \tag{1.2}$$

Pour réaliser une forme avec une qualité souhaitée, on définit, pour chaque opération, les *conditions de coupe* qui déterminent les mouvements de l'outil enlevant de la matière. La définition des *conditions de coupe* consiste à sélectionner les valeurs appropriées pour la vitesse de coupe v_c et l'avance par tour f_n. Ces valeurs sont déterminées à partir des tableaux de caractéristiques de coupe. Il y a plusieurs critères pour effectuer ce choix, notamment l'appartenance à une phase d'usinage (ébauche, demi-finition, finition), la matière usinée (acier, aluminium), la matière de l'outil (ARS, carbure), le type de l'opération (perçage, alésage, taraudage).

En disposant de ces valeurs, l'avance v_f est calculée comme suit :

$$v_f = \frac{f_n v_c}{\pi D} \tag{1.3}$$

Hormis les conditions de coupe, une opération de perçage est caractérisée par un autre paramètre, la *longueur de la course d'outil* l qui décrit la longueur de la trajectoire de l'outil nécessaire pour l'achèvement de l'opération. La valeur de l est égale à la profondeur du trou plus la longueur de pointe du foret (pour les trous débouchants) plus la distance d'approchement de l'outil (Figure 1.4).

En spécifiant la longueur de la course d'outil l et en la divisant par la vitesse d'avance v_f, on obtient le temps de perçage t.

$$t = \frac{l}{v_f} \tag{1.4}$$

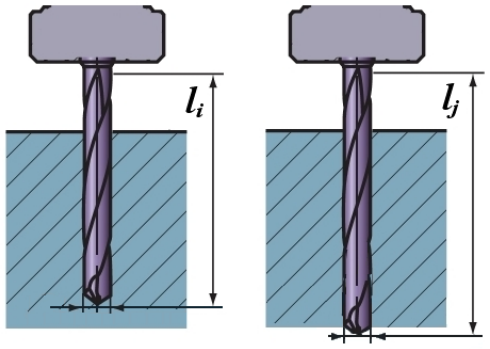


FIG. 1.4 – Longueur de la course d'outil l

Les *outils* effectuant les opérations de perçage peuvent avoir de multiples formes, quelques exemples sont présentés dans la Figure 1.5 [Toumine, 2007].

FIG. 1.5 – Outils de perçage

Généralement, les outils de type « foret » effectuent les opérations de perçage proprement dites, les forets à pointer sont utilisés pour positionner un perçage, les alésoirs sont employés pour la finition de trous, enfin, les fraises à lamer effectuent les opérations de lamage. En outre, il existe des outils spéciaux qui permettent l'exécution simultanée de plusieurs opérations, on les appelle *outils étagés*. La Figure 1.6 [Seco CrownLoc®, 2004] montre comment on peut obtenir, à partir d'un foret simple, un outil étagé effectuant un perçage et un alésage à la fois. La Figure 1.7 [Seco CrownLoc®, 2004] montre quelques configurations des outils étagés destinés à effectuer avec le perçage d'un trou : A1 - un chanfrein, A2 - un bossage (face), A3 - un chanfrein et un bossage, B1 - un chambrage, B2 - un chambrage et un chanfrein.

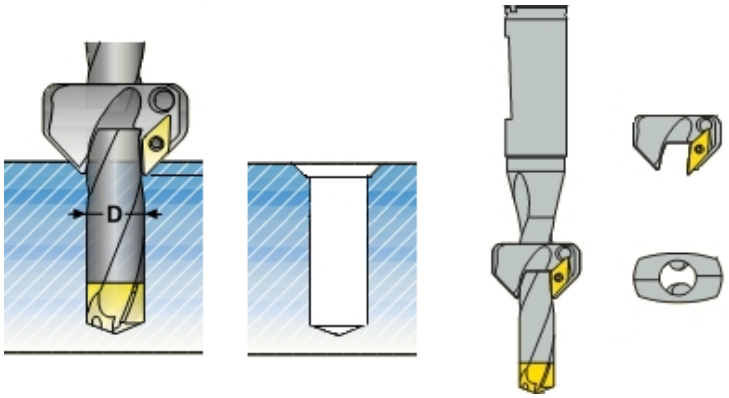


FIG. 1.6 – Usinage avec un outil étagé

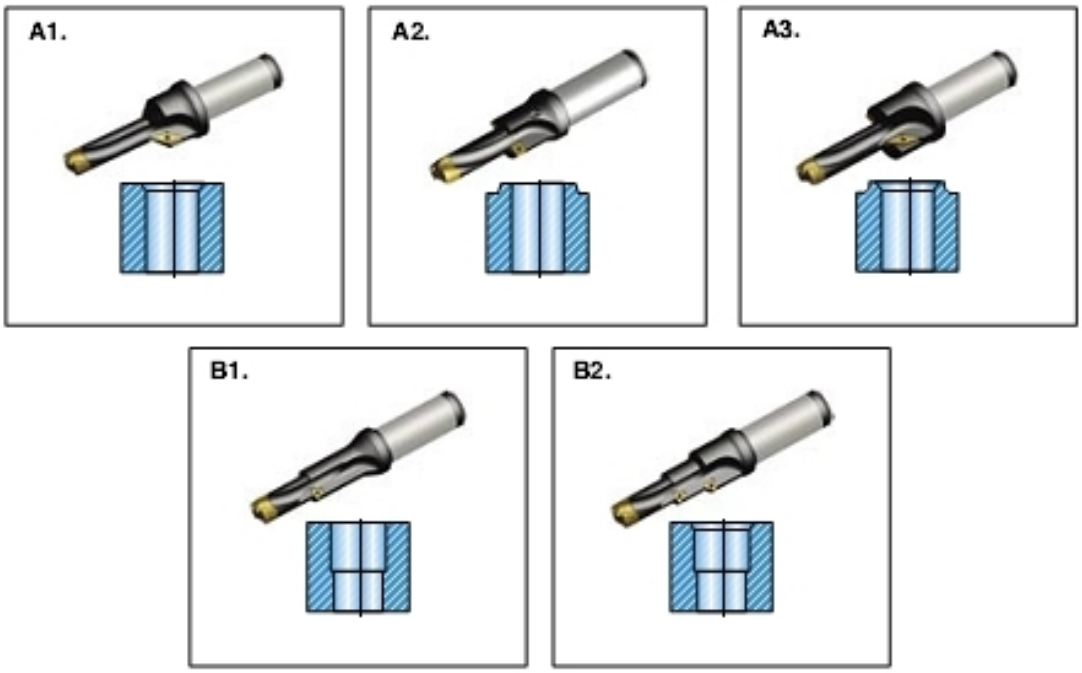


FIG. 1.7 – Outils étagés

1.1.4 Opérations de fraisage

Les opérations de fraisage sont utilisées pour la création de surfaces diverses, comme le montre la Figure 1.8 [Ruby et Maranzana, 2007]. Une opération de fraisage emploie un ou plusieurs mouvements d'outil : radial, périphérique et axial, qui sont décrits en utilisant les mêmes paramètres qu'une opération de perçage, à savoir la vitesse de broche n, la vitesse de coupe v_c, l'avance par tour f_n et la vitesse d'avance v_f. Leurs valeurs sont calculées en utilisant les formules (1.1) - (1.3). La seule différence apparaît lors du calcul de f_n si la fraise a plusieurs dents comme le montre la Figure 1.9 [Coromant Sandvik®, 2006].

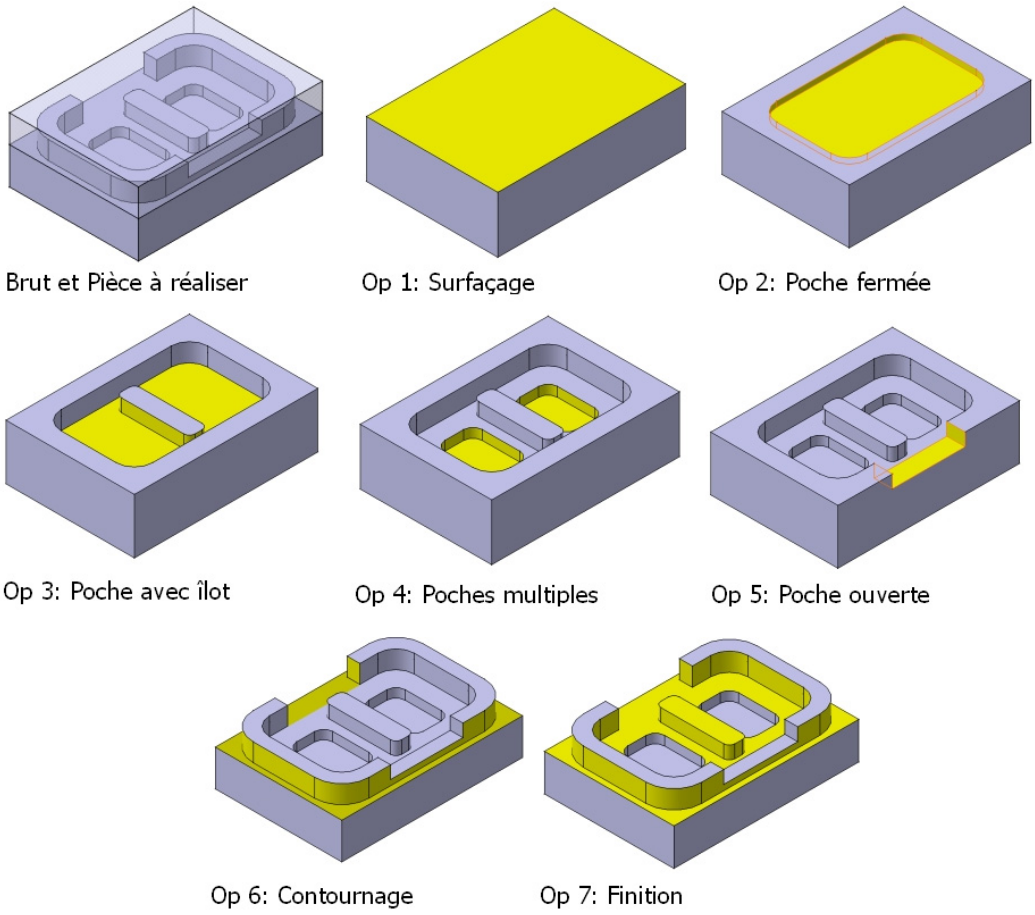


FIG. 1.8 – Différents types d'opérations de fraisage

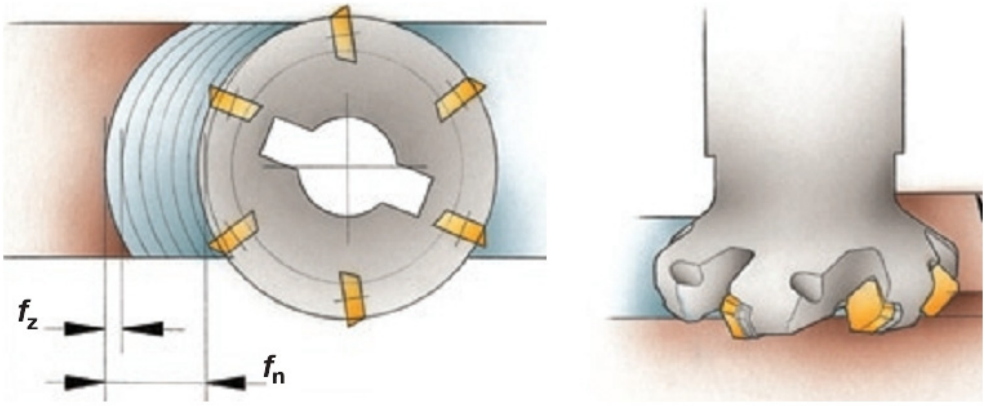


FIG. 1.9 – Fraise à plusieurs dents

Le nombre de dents est désigné par le paramètre z. Dans ce cas, on détermine, à partir des tableaux de caractéristiques de coupe, la valeur de f_z désignant l'**avance par tour par dent** qui correspond à la distance que la dent va parcourir à chaque tour de la fraise. L'avance par tour de la fraise f_n est alors calculée comme suit :

$$f_n = f_z z \tag{1.5}$$

Contrairement aux opérations de perçage, les opérations de fraisage sont caractérisées tantôt par des trajectoires simples, s'il s'agit de fraisage de trous, tantôt par des trajectoires plus complexes, comme le montre la Figure 1.10 [Hamza *et al.*, 2005].

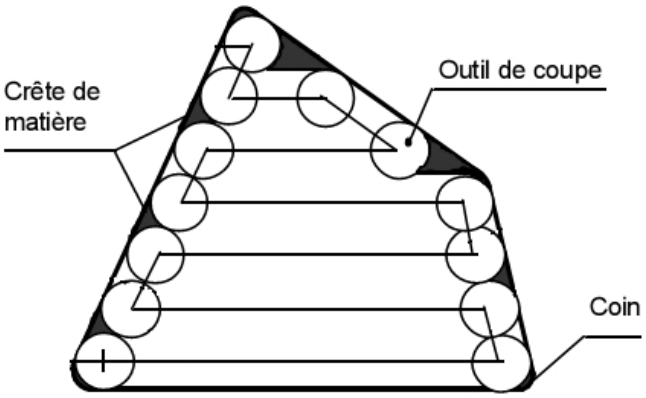


FIG. 1.10 – Trajectoire de l'évidement d'une poche

Une fois la longueur à fraiser l déterminée, le temps de fraisage est calculé, comme pour une opération de perçage, en divisant l par la vitesse d'avance v_f.

La Figure 1.11 [Coromant Sandvik®, 2007] montre différents outils utilisés pour effectuer des opérations de fraisage. Il est à noter que pour la réalisation de formes complexes, comme des poches, par exemple, il est souvent nécessaire d'utiliser plusieurs outils de diamètres différents afin d'obtenir la forme géométrique conforme. Ainsi la réalisation des formes complexes nécessite non seulement le choix optimisé des outils, mais la recherche de la meilleure trajectoire d'usinage et des conditions de coupe correspondantes. Ce processus est largement évolué ces dernières années avec l'utilisation de nouvelles techniques comme la maquette numérique, une modélisation complète du produit et de son procédé dans un système de CFAO. Ces logiciels offrent actuellement un large éventail de typologies d'opérations. Ils sont associés à des générateurs de trajectoires d'usinage de plus en plus performants [Pateloup *et al.*, 2004]. Actuellement, plusieurs approches existent pour réaliser un choix optimisé des outils, de leurs trajectoires et des différents paramètres [Cus, 2000; Dereli *et al.*, 2001; Wang *et al.*, 2002; Vijayakumar *et al.*, 2003; Amiolemhen et Ibhadode, 2004].

Comme nous l'avons déjà indiqué, les opérations d'usinage sont exécutées en employant les moyens d'un système d'usinage. Dans ce qui suit, nous présentons en détail les éléments les plus utilisés et les configurations possibles des systèmes d'usinage.

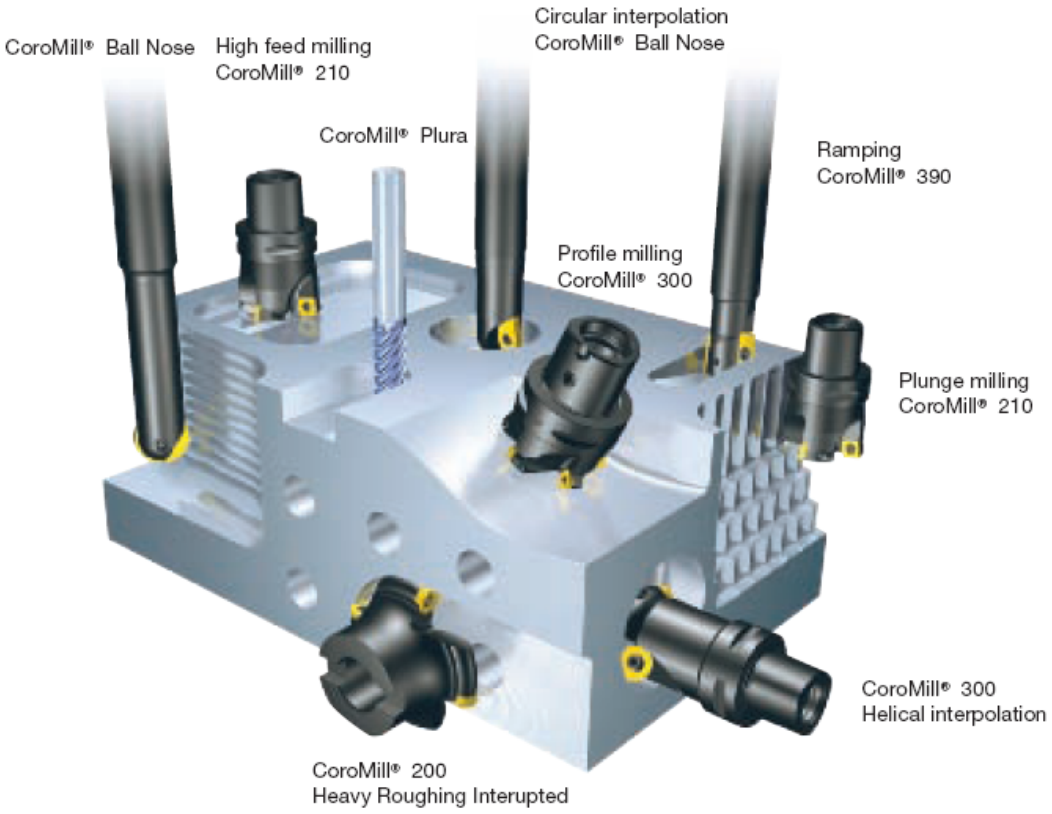


FIG. 1.11 – Outils de fraisage

1.2 Configuration des systèmes d'usinage

1.2.1 Éléments de base des systèmes d'usinage

Comme nous l'avons précédemment mentionné, nous considérons le cas d'usinage où la pièce est fixée. Pour être fonctionnel, un système d'usinage doit donc comporter au moins un dispositif portant la pièce et un autre dispositif mettant en mouvement l'outil. Plus précisément, la pièce est fixée sur une table ou sur une palette porte-pièce à l'aide d'un montage. Une bonne fixation de la pièce permet d'éviter des déformations statiques (défauts de forme et de dimension des surfaces usinées) ou dynamiques (défauts d'état de surface et des vibrations, ...). Quant à l'outil, généralement, il est fixé par l'intermédiaire d'un *porte-outil* dans une *broche*. La broche est connectée à un moteur et imprime un mouvement à l'outil. Le dispositif portant la broche et pilotant son comportement est appelé *unité d'usinage*.

Les avancées du génie mécanique et de l'automatique ont permis de créer des systèmes d'usinage ayant des architectures très diverses, depuis les systèmes très simples jusqu'aux systèmes complètement automatisés intégrant des techniques d'intelligence artificielle. Bien évidemment, la classification de tels systèmes n'est pas unique et peut se faire selon différents critères. Dans ce mémoire, par exemple, nous distinguons les *types de système d'usinage* en

fonction des types d'équipements qui sont employés dans le système d'usinage, des règles de transfert de pièce et de l'organisation du processus d'usinage ainsi que des paramètres permettant de décrire la structure du système d'usinage. L'ensemble des valeurs des paramètres utilisés pour décrire le « type » auquel un système d'usinage appartient définit la *configuration* de ce système. Ainsi deux systèmes d'usinage du même type sont décrits par le même ensemble des paramètres, tandis que les valeurs de ces derniers sont propres à chaque système d'usinage.

Dans cette optique, les systèmes d'usinage « reconfigurables » [Koren *et al.*, 1999] peuvent être définis comme les systèmes dont la configuration peut être facilement changée au cours de leur exploitation sans leur reconception (les changements nécessaires sont anticipés lors de la conception de tels systèmes). Pour plus de détails, voir, par exemple, [Koren *et al.*, 1999; Mehrabi *et al.*, 2000; Son, 2000; Mehrabi *et al.*, 2002; Dashchenko, 2006]. Néanmoins, ce concept est nouveau et il n'existe pas encore de développement industriel à grande échelle. Dans le souci de ne pas disperser l'attention du lecteur, nous axons la suite de la présentation sur les systèmes d'usinage que nous avons observés chez nos partenaires industriels au cours de notre étude. Le lecteur souhaitant en savoir davantage sur d'autres systèmes d'usinage trouvera une littérature riche, par exemple, [Pruvot, 1993; Bernard, 2003; Dashchenko, 2003].

Dans ce qui suit, nous présentons un élément discriminant des systèmes d'usinage : les unités d'usinage.

1.2.2 Unités d'usinage

Les types d'unités d'usinage principalement utilisés dans l'industrie sont présentés dans les Figures 1.12, 1.13, 1.14 [Licon^{MT}®, 2007]. Quelle est la différence entre ces trois types d'unités d'usinage et dans quelles conditions convient-t-il d'employer l'un ou l'autre ?

Le premier type, présenté dans la Figure 1.12, a une seule broche. Il s'agit d'une unité d'usinage *monobroche*. Une telle unité d'usinage ne peut effectuer qu'une seule opération d'usinage à la fois ; celle-ci est exécutée par l'outil de coupe dont la broche est équipée. En fonction de l'opération à effectuer, l'outil peut être remplacé par un autre, stocké dans un magasin d'outils de type revolver. Pour effectuer un changement d'outil au cours de l'usinage d'une pièce, il faut arrêter la broche, la remettre en position d'échange, enlever l'outil désormais inutile, le monter dans le magasin d'outils, y trouver l'outil nécessaire pour l'opération suivante, le faire sortir du magasin et le fixer dans la broche, puis remettre la broche en position d'usinage en augmentant sa vitesse jusqu'à celle prévue pour l'opération d'usinage à effectuer. Bien que pour les machines à commande numérique, les changements d'outils soient effectués de manière automatique, en suivant un ordre programmé, une telle séquence d'actions non productives augmente le temps nécessaire pour l'usinage d'une pièce. La durée totale de tous les changements peut atteindre jusqu'à 40% du temps total d'usinage lorsque les changements d'outils sont fréquents.

Toutefois, malgré ce handicap lié aux pertes de temps, l'emploi de ce type d'unités d'usinage permet aux systèmes d'usinage d'avoir une certaine flexibilité par rapport aux pièces à usiner et aux opérations d'usinage. Ceci représente un atout pour la fabrication de pièces ayant des cycles de vie courts. En effet, le passage d'une pièce à une autre est aisé, car il suffit de recharger le magasin d'outils et de mettre à jour le programme d'usinage.

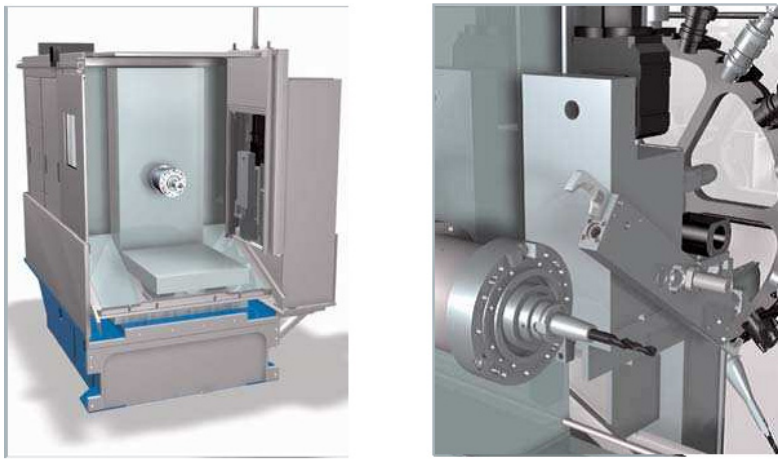


FIG. 1.12 – Unité d'usinage monobroche

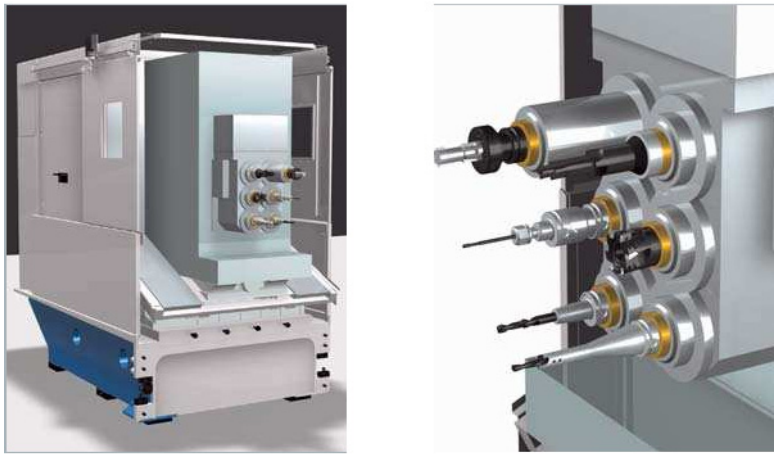


FIG. 1.13 – Boîtier multibroche

FIG. 1.14 – Tourelle

La Figure 1.13 présente un *boîtier multibroche*, qui est le deuxième type d'unité d'usinage. Un tel boîtier peut porter plusieurs broches, chacune munie d'un outil de coupe. Ceci permet d'effectuer simultanément un ensemble d'opérations. Par conséquent, le temps total d'usinage d'une pièce est diminué, d'une part, par l'exécution simultanée de plusieurs opérations, mais aussi par la disparition des changements d'outils. Dans ce qui suit, nous appelons l'ensemble des opérations exécutées par un boîtier multibroche *bloc d'opérations*. Il est à noter que seules les opérations à effectuer sur la même face de la pièce peuvent être groupées dans un bloc. D'autres contraintes doivent être également prises en compte lorsque des boîtiers multibroches sont utilisés.

Néanmoins, on gagne en productivité au détriment de la flexibilité : une fois les outils fixés dans un boîtier multibroche, il est difficile, voire impossible, de le reconfigurer afin de passer à l'usinage d'un autre type de pièce. Par conséquent, seules des pièces « proches » peuvent être usinées par le même boîtier multibroche.

Enfin, le troisième type, présenté dans la Figure 1.14, combine les avantages et les inconvénients des deux types précédents. Comme dans le premier cas, les outils de coupe sont activés séquentiellement. Par contre, leur fixation dans une tourelle permet de diminuer le temps de changements. Ce type d'unités d'usinage convient à l'usinage des pièces nécessitant des opérations d'usinage assez similaires, car le nombre d'outils qui peuvent être fixés dans une tourelle est sensiblement inférieur à la capacité d'un magasin d'outils utilisé avec des unités d'usinage du premier type.

Même si le choix d'unité d'usinage s'appuie sur une analyse des points forts et faibles de chaque type, il est à souligner que ce choix n'est pas toujours évident. Dans le souci d'organiser le processus d'usinage de manière optimale, ce choix implique la considération de plusieurs facteurs, notamment du processus d'usinage de la pièce et des objectifs économiques à atteindre. Souvent l'évaluation d'une solution par rapport à une autre ne peut pas être effectuée au préalable, sans développement détaillé de chacune d'elles.

Du point de vue de la conception, les systèmes comportant des unités d'usinage du premier et du troisième type nécessitent la résolution des problèmes décisionnels proches, puisqu'il s'agit, dans les deux cas, de choisir l'ordre de l'activation des outils et leur positionnement soit dans un magasin d'outils, soit dans une tourelle. De tels problèmes ont été traités, par exemple, dans [Sarma et Wright, 1996; Szadkowski, 1997; Tzur et Altman, 2004]. Par contre, l'utilisation du deuxième type d'unité d'usinage dégage des problèmes de conception très différents et très peu étudiés dans la littérature : il s'agit de regrouper les opérations en blocs afin de les exécuter par des boîtiers multibroches. En présence d'un nombre important d'opérations, ce problème devient fortement combinatoire. Une autre particularité de ce problème réside dans la dépendance entre le temps d'exécution des opérations et le contenu des blocs auxquels elles appartiennent. Dans la section suivante, nous montrerons comment les paramètres d'usinage de boîtiers multibroches dépendent des paramètres des opérations qui leur sont attribuées.

1.2.3 Paramètres d'usinage de boîtier multibroche

Les outils de coupe d'un boîtier multibroche sont déplacés par la force du même moteur. Par conséquent, tous les outils sont activés et désactivés en même temps. Ainsi les

deux paramètres suivants sont communs pour toutes les opérations exécutées par un boîtier multibroche (regroupées dans un bloc) : la *longueur de la course* et la *vitesse d'avance* de boîtier. Les valeurs de ces deux paramètres sont calculées pour chaque boîtier multibroche en fonction des valeurs admissibles des paramètres des opérations qu'il effectue.

Longueur de la course de boîtier

La longueur de la course de boîtier correspond au déplacement du boîtier en phase d'usinage. Afin d'achever chaque opération correctement, la course du boîtier doit être supérieure ou égale à la course de l'outil nécessaire pour chaque opération.

Soit N l'ensemble des opérations à effectuer par un boîtier. Chaque opération i de l'ensemble N est caractérisée par l_i qui représente la longueur de la course de l'outil correspondant. Alors, la longueur de la course du boîtier effectuant l'ensemble d'opérations N est calculée de la manière suivante :

$$L(N) = max\{l_i |\, i \in N\} \tag{1.6}$$

Ainsi, si $L(N) > l_i$ pour une opération $i \in N$, cela est corrigé par l'utilisation des porte-outils. Un exemple d'un tel ajustement est présenté dans la Figure 1.15.

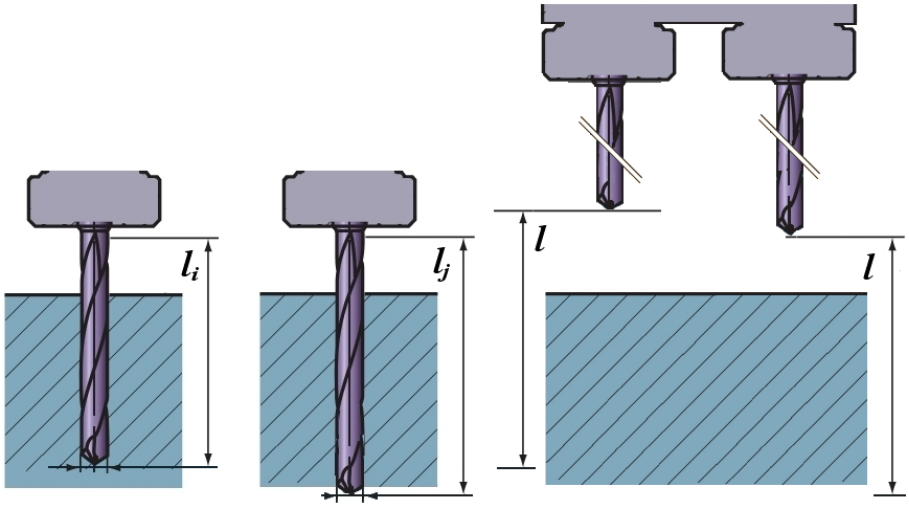


FIG. 1.15 – Fixation d'outils dans un boîtier multibroche

Vitesse d'avance de boîtier

Nous supposons que pour chaque opération $i \in N$, nous connaissons l'outil opportun pour son exécution. Il est caractérisé par les paramètres suivants :

- son diamètre D_i,
- l'intervalle des valeurs admissibles de l'avance par tour $[f_{n1}(i), f_{n2}(i)]$,
- l'intervalle des valeurs admissibles de la vitesse de coupe $[v_{c1}(i), v_{c2}(i)]$.

Les valeurs minimum possibles de l'avance par tour et de la vitesse de coupe sont désignées par $f_{n1}(i)$ et $v_{c1}(i)$, respectivement, tandis que $f_{n2}(i)$ et $v_{c2}(i)$ donnent les valeurs maximum possibles pour ces paramètres. À partir de ces valeurs, nous pouvons obtenir, pour chaque opération, l'intervalle des valeurs admissibles de la vitesse d'avance $[v_{f1}(i),v_{f2}(i)]$, où :

$$v_{f1}(i) = \frac{f_{n1}(i)v_{c1}(i)}{\pi D_i} \tag{1.7}$$

$$v_{f2}(i) = \frac{f_{n2}(i)v_{c2}(i)}{\pi D_i} \tag{1.8}$$

Nous pouvons obtenir ensuite $\mathbf{V}(N)$, l'intervalle des valeurs admissibles de la vitesse d'avance du boîtier réalisant l'ensemble d'opérations N :

$$V_1(N) = max\{v_{f1}(i)|\, i \in N\} \tag{1.9}$$

$$V_2(N) = min\{v_{f2}(i)|\, i \in N\} \tag{1.10}$$

$$\mathbf{V}(N) = [V_1(N), V_2(N)] \tag{1.11}$$

Si l'ensemble $\mathbf{V}(N)$ est vide, alors les opérations appartenant à l'ensemble N ne peuvent pas toutes être effectuées par un seul boîtier multibroche et, par conséquent, ne peuvent pas être groupées dans un seul bloc. Par ailleurs, il n'existe pas de méthode unique pour choisir la valeur $v_f(N)$ de la vitesse d'avance pour un boîtier à partir de l'intervalle $\mathbf{V}(N)$. En pratique, plusieurs techniques peuvent être utilisées. Dans ce mémoire, nous utilisons la technique suivante :

$$v_f(N) = V_2(N) \tag{1.12}$$

Calcul du temps d'usinage de boîtier

Le temps d'usinage de boîtier correspond au temps de déplacement de boîtier en phase d'usinage. Comme pour une simple opération, ce temps est calculé en utilisant (1.4), mais en remplaçant les paramètres d'opération par les paramètres de boîtier.

$$t^b(N) = \frac{L(N)}{v_f(N)} \tag{1.13}$$

Toutes les opérations de l'ensemble N seront exécutées à la fois. Ainsi l'utilisation des boîtiers multibroches permet de diminuer le temps total d'usinage d'un ensemble d'opérations. Cependant, le temps d'exécution d'un bloc d'opérations est toujours supérieur ou égal au temps d'exécution de chaque opération appartenant à ce bloc, c'est-à-dire :

$$\forall\, i \in N,\ t^b(N) \geq \frac{l_i}{v_{f2}(i)} \tag{1.14}$$

1.2.4 Postes de travail

Nous rappelons qu'au cours de l'usinage, la pièce est fixée sur une table ou sur une palette porte-pièce. Nous appelons *position de travail* l'ensemble des dispositifs assurant le positionnement de la pièce. Nous définissons un *poste de travail* comme l'ensemble des dispositifs associés à une position de travail et mettant en œuvre l'usinage de la pièce s'y trouvant. Il est à noter qu'en général, plusieurs posages de la pièce peuvent être réalisés sur la même position de travail à condition que le poste de travail soit équipé de dispositifs pour faire tourner la pièce. Plusieurs boîtiers multibroches peuvent être installés sur un poste de travail, agissant sur la pièce de façon séquentielle, parallèle ou mixte :

Mode parallèle : quand les gabarits de la pièce le permettent, plusieurs boîtiers peuvent agir simultanément sur la pièce. Ainsi le traitement simultané de plusieurs faces de la pièce devient possible et, par conséquent, le temps total d'usinage est réduit. Toutefois, en mettant en place une telle activation de boîtiers, il est important d'étudier des impacts éventuels entraînés par le mouvement simultané de plusieurs broches, afin d'éviter des déformations et des écarts de forme et de qualité d'usinage. Ceci explique le fait qu'il est rarement possible d'installer plus de 3 boîtiers en parallèle. La Figure 1.16 fournit un schéma du mode parallèle d'activation de deux unités d'usinage.

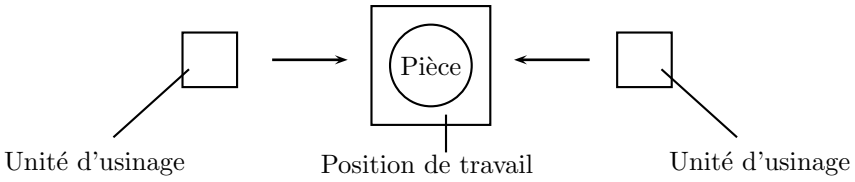


FIG. 1.16 – Schéma du mode parallèle

Mode séquentiel : une seule unité d'usinage agit sur la pièce à la fois. Par contre, il est possible de la remplacer par une autre, installée sur le même poste de travail, soit par le déplacement de la pièce vers l'unité suivante, soit par le changement de l'unité active à l'aide d'un dispositif spécial. Comme dans le cas des unités d'usinage monobroches, un tel changement entraîne forcément des temps non productifs, mais ici plusieurs outils sont changés simultanément, car il s'agit des boîtiers multibroches. Si ce mode est utilisé, il convient de déterminer, à l'étape de conception, l'ordre d'activation des unités d'usinage et de les disposer de manière appropriée. La Figure 1.17 présente un schéma de poste de travail ayant des unités d'usinage activées séquentiellement.

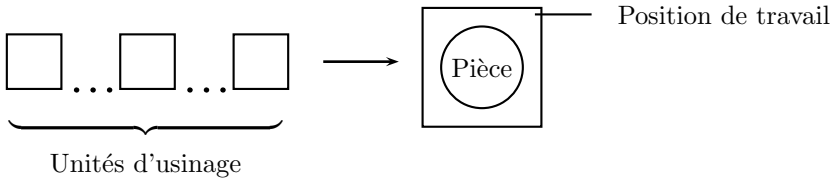


FIG. 1.17 – Schématique du mode séquentiel

Mode mixte : comme son nom l'indique, il s'agit du mode conjuguant l'activation simultanée de plusieurs boîtiers avec l'aptitude à changer des boîtiers actifs à l'aide des dispositifs spéciaux. Cette configuration est délicate à employer, le cas le plus répandu consiste en l'installation de deux ensembles de boîtiers en vis-à-vis ; un seul boîtier de chaque ensemble est actif à la fois (ce qui fait que deux boîtiers agissent en simultané sur la pièce), par contre, chacun de ces deux boîtiers peut être remplacé par un autre boîtier de l'ensemble correspondant (voir Figure 1.18).

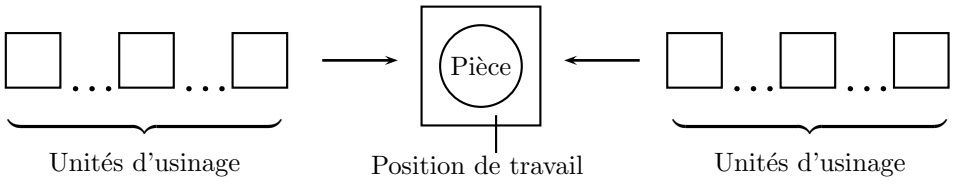


FIG. 1.18 – Présentation schématique du mode mixte

Si un seul poste d'usinage ne suffit pas, il convient d'en installer plusieurs tout en les liant par un dispositif de transfert de la pièce. Ce dispositif assure le passage de la pièce d'un poste de travail à un autre. Le transfert peut être réalisé de manière angulaire, par exemple, pour des machines à plateau ou tambour, ou linéaire pour des machines à transfert linéaire. Pour la facilité du pilotage, généralement, les postes de travail constituant un système d'usinage ont le même mode d'activation d'unités d'usinage.

1.2.5 Cycle d'usinage

Quand le contexte de production recommande l'emploi de systèmes d'usinage à boîtiers multibroches, il impose également une cadence de production. La notion de cadence est liée directement au *temps de cycle d'usinage*, le temps compris entre la production de deux pièces consécutives par le système d'usinage. Plus ce temps est grand, plus la productivité du système d'usinage est faible. En pratique, la productivité requise est estimée sur la base du volume annuel à atteindre en tenant compte des éventuelles interruptions qui pourraient perturber la production. Afin d'assurer la cadence de production objectif, le temps de cycle ne doit pas dépasser une certaine valeur limite, désignée par le paramètre T_0.

Afin d'assurer le déplacement synchrone de pièces et d'éviter l'utilisation de stocks tampons entre les postes, le temps d'usinage sur chaque poste de travail doit être inférieur à ce temps de cycle, désigné par T_0. Si cette condition est satisfaite, toutes les pièces en cours sont simultanément déplacées au poste de travail suivant à intervalle de temps régulier, égal au temps de cycle. Un nouveau brut est chargé sur le premier poste de travail et une pièce finie est déchargée sur le dernier poste.

Pour atteindre le volume objectif de la productivité, on essaie, dans un premier temps, de réduire au maximum le temps total d'usinage de la pièce, notamment en utilisant des boîtiers multibroches. Si cela n'est pas suffisant, on augmente le nombre de postes de travail (Figure 1.19). Dans ce cas, plus d'unités de la pièce sont traitées en même temps dans chaque

cycle, moins d'opérations sont faites par poste de travail, et donc le temps de cycle est plus petit.

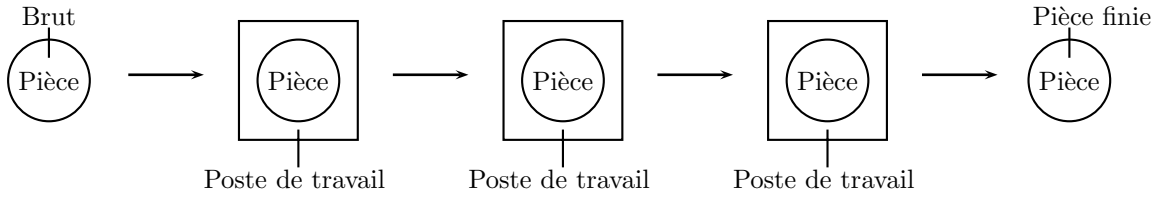


FIG. 1.19 – Schéma d'un système d'usinage avec transfert linéaire de la pièce

En même temps, puisque l'installation de chaque poste et chaque boîtier complémentaire induit un coût, le nombre de boîtiers et de postes de travail à installer est à minimiser tout en tenant compte des contraintes techniques et celle de temps de cycle.

1.2.6 Types de systèmes d'usinage concernés par l'étude

Les systèmes d'usinage que nous considérons dans ce mémoire ont les caractéristiques suivantes :

- leur configuration est dédiée à la fabrication efficace d'une famille de pièces proches ;
- ils doivent assurer une cadence de production importante ;
- ils comportent plusieurs postes de travail par lesquels la pièce doit passer en suivant leur ordre ;
- chaque poste de travail est muni d'un ou de plusieurs boîtiers multibroches.

Parmi tous les systèmes d'usinage qui correspondent à ces critères, nous avons choisi trois types dont la conception sera étudiée dans ce mémoire, à savoir :

1. Machines à transfert linéaire avec boîtiers multibroches activés de manière séquentielle (Chapitre 4) ;
2. Machines à transfert linéaire ou circulaire avec boîtiers multibroches activés de manière parallèle (Chapitre 6) ;
3. Machines à table mobile, qui peuvent être considérées comme systèmes d'usinage ayant le mode mixte d'activation des boîtiers (Chapitre 6).

Comme nous pouvons le constater, la différence entre ces types de systèmes réside dans le mode d'activation de boîtiers sur un poste de travail et le mode de transfert de la pièce d'un poste à un autre.

Toutes ces machines sont appelées dans l'industrie *machines dédiées* ou *machines spéciales*, car chaque machine conçue est « spéciale » et dédiée à la fabrication d'une famille de pièces proches, voire un type de pièce. De façon générale, de tels systèmes d'usinage sont utilisés lorsqu'une grande cadence de production est indispensable ou lorsque la pièce à produire est très spécifique (par exemple, à cause de ses gabarits importants).

1.2.7 Configuration des systèmes d'usinage étudiés

Afin d'être efficaces et rentables, les machines dédiées, qui font l'objet de notre étude, doivent être conçues avec une analyse approfondie de la pièce et de leurs configurations possibles. Nous pouvons recenser les principaux paramètres communs utilisés pour décrire la configuration de tels systèmes :

- le nombre de postes de travail,
- le nombre de boîtiers à chaque poste de travail,
- l'ensemble des opérations exécutées par chaque boîtier,
- les conditions de coupe imposées à chaque boîtier,
- l'ordre d'activation des boîtiers installés sur chaque poste de travail (seulement si le mode d'activation est séquentiel ou mixte).

La difficulté principale pour trouver la configuration optimale pour un tel système d'usinage résulte du caractère combinatoire du problème de la répartition des opérations d'usinage à des boîtiers communs et de ces derniers à des postes de travail.

1.3 Conclusion

Ce chapitre a été consacré à la description des systèmes d'usinage et à l'introduction de leurs principaux paramètres. D'abord, nous avons présenté les types d'opérations d'usinage et leurs caractéristiques. Ensuite, nous avons décrit différents types de systèmes d'usinage. Puis, nous nous sommes concentrés sur les systèmes d'usinage à boîtiers multibroches qui font l'objet de cette thèse. La structure des boîtiers multibroches et leur mise en place ont été énoncées. Un des points importants développés est la dépendance entre les paramètres d'usinage d'un boîtier multibroche et les paramètres d'usinage des opérations qui lui sont attribuées. Ceci doit être impérativement pris en compte lors de la configuration des systèmes d'usinage les utilisant. La problématique de la configuration de tels systèmes d'usinage sera présentée en détail dans le chapitre suivant.

Chapitre 2

Conception des systèmes d'usinage en avant-projet

On ne pourra bien dessiner le simple qu'après une étude approfondie du complexe.

Gaston Bachelard

Tout d'abord, nous présentons les activités méthodologiques et décisionnelles lors de la conception en avant-projet des systèmes d'usinage. Ensuite, nous proposons une modélisation mathématique du problème d'optimisation de la configuration des systèmes d'usinage à boîtiers multibroches.

2.1 Contexte économique

La filière de l'usinage est fortement hétérogène, tant par la variété de ses métiers que par la diversité de ses entreprises. En France, elle comprend environ 6 500 entreprises de plus de 20 salariés pour un effectif global de plus de 492 000 personnes. Dans ce secteur, 97% des entreprises possèdent un effectif inférieur à 500 salariés. Les industries mécaniques françaises représentent environ 12% de la production nationale (51 milliards d'euros de chiffre d'affaires). À l'international, elles occupent la 5ème place en production et la 6ème place mondiale à l'exportation avec un volume de 39,2 milliards d'euros d'exportation (données de la Fédération des Industries Mécaniques, FIM, 2005).

Comme la plupart des secteurs manufacturières, les industries mécaniques sont en mutation dans un contexte technico-économique de plus en plus exigeant : mondialisation des marchés et des capitaux, accroissement des impératifs réglementaires, émergence du développement durable, exigences accrues des clients au niveau notamment des propriétés d'usage des produits et des prestations de services associées. Par conséquent, les industries mécaniques se trouvent aujourd'hui confrontées à de nouveaux défis et dans un marché extrêmement compétitif. Afin de se donner les moyens de produire au plus juste coût, en maîtrisant qualité et délais, il est impératif d'effectuer de manière rationnelle les choix

stratégiques. De nombreuses méthodes de la gestion de production ont été élaborées au cours de ces dernières années afin de fournir aux industriels des outils d'aide à la décision efficaces. Dans ce mémoire, nous avons opté pour étudier un aspect particulier de la gestion de production auquel les entreprises doivent faire face : il s'agit du problème du choix de la configuration du système d'usinage lors de la mise en fabrication d'un nouveau produit. Dans la pratique, plusieurs démarches pour la recherche de la configuration optimale du système d'usinage sont possibles, dont les deux suivantes sont le plus souvent utilisées :

- concevoir une gamme d'usinage en utilisant les ressources de production à la disposition de l'entreprise (déjà acquises),
- lancer un appel d'offre aux entreprises fabriquant des systèmes d'usinage pour acheter une solution clés en main.

Il est à noter que la première problématique a été intensivement étudiée en génie mécanique. Concevoir une gamme d'usinage d'une pièce consiste à proposer la suite des actions à entreprendre pour passer de la pièce brute à la pièce finie, à partir de la définition de la pièce, de la technique d'obtention de la pièce brute, du contexte économique et des moyens de production connus et disponibles. L'idée de la génération automatique de gammes en recourant à des outils informatiques est apparue en 1965, initiée par Niebel [Niebel, 1965]. Cependant, cette démarche s'est avérée complexe et aucun modèle général n'a encore été proposé du fait de la diversité des types de production, mais aussi parce que chaque secteur de production détient un savoir-faire spécifique, jalousement gardé [Villeneuve, 2003]. Toutefois, plusieurs approches ont été développées et de nombreux logiciels ont été dédiés à la génération automatique de gammes dans des contextes particuliers. L'accent a été principalement mis sur l'étude de la planification d'usinage (« la fabricabilité » [Lossent, 1997] ou « l'usinabilité» [Anwer, 2000]) d'une pièce dans un atelier.

Quant à la deuxième problématique, qui concerne le processus de conception des systèmes d'usinage par les entreprises les fabriquant, elle a bénéficié de très peu d'attention dans la littérature. Les travaux étudiant cette problématique restent limités [Jacobe, 1992; Garro, 1992; Martin *et al.*, 2001; Zhang *et al.*, 2002; Dashchenko, 2003].

La différence entre les deux démarches réside dans la définition des ressources. Dans le cadre de la planification d'usinage dans un atelier, le choix des ressources est limité par les machines-outils déjà existantes. Dans le cas de la conception des systèmes d'usinage, les entreprises les fabriquant se trouvent face à un problème différent. Au moment de la conception, les ressources n'existent pas encore et doivent être conçues et organisées dans un système. En conséquence, ce dernier problème s'avère encore plus complexe que le précédent.

Dans ce mémoire, nous étudions la problématique de la conception des systèmes d'usinage dans le cadre d'un appel d'offre ou la conception en avant-projet. Dans ce qui suit, nous décrivons les activités méthodologiques et décisionnelles qui sont mises en œuvre durant cette phase de conception.

2.2 Problématique

Pour répondre à un appel d'offre, l'entreprise fabriquant des solutions clés en main pour les systèmes d'usinage doit proposer au client un devis de fabrication d'un système d'usinage.

Pour rester compétitif, l'équipementier doit concevoir les systèmes d'usinage plus rapidement, plus efficacement et moins cher que les concurrents tout en répondant au mieux aux besoins du client. Le devis qui est transmis au client a un impact prépondérant sur la décision de l'attribution du contrat. Alors, il est capital de dégager des solutions profitables tant au client émetteur de l'appel d'offre qu'à l'entreprise équipementière, notamment en minimisant le coût du système d'usinage développé. La mesure du succès réside dans l'aptitude à examiner un large éventail de solutions possibles et en choisir la meilleure.

Le processus d'élaboration d'un devis commence par l'analyse de la pièce (pièces) à produire et des objectifs économiques fixés par le client. Ensuite, il est primordial de choisir le type de système d'usinage qui répond au mieux aux attentes du client. Plusieurs solutions sont envisageables, telles que des machines dédiées, une ligne Kaizen, une ligne flexible à base de centres d'usinage (CU), etc. Pour trouver la solution optimale, divers aspects économiques et techniques doivent être étudiés : tout d'abord, la productivité requise et l'enveloppe d'investissement, mais aussi la montée en cadence progressive, les paramètres liés au processus d'usinage de la pièce, etc. Cette décision est difficile sachant que souvent l'évaluation d'une solution par rapport à une autre ne peut être effectuée sans développement détaillé de chacune d'elles. Dans la pratique, dans un premier temps, un ensemble des types de systèmes d'usinage susceptibles d'être efficaces est choisi en tenant compte des données connues. Souvent, il est nécessaire de limiter la taille de cet ensemble afin de rendre envisageable l'exploration des solutions les plus prometteuses.

L'exploration d'une solution consiste à trouver, pour un type de système d'usinage fixé, sa configuration optimale. Le nombre d'éléments à prendre en considération rend cette recherche extrêmement complexe. D'autant plus que l'examen d'une configuration possible doit être suffisamment précis afin de vérifier sa cohérence, sa faisabilité, et éviter une découverte tardive des fautes de conception à l'étape de fabrication.

La comparaison des configurations optimales trouvées pour les différents types de systèmes d'usinage permet d'établir la meilleure proposition à faire au client. Éventuellement, le devis élaboré peut contenir des solutions alternatives en laissant le choix de la solution à mettre en place au client. La Figure 2.1 récapitule les activités de conception en avant-projet.

Ainsi cette phase de conception mène à l'exploitation d'un ensemble de scénarii de taille importante et implique des calculs lourds. Le caractère combinatoire du problème de conception pour chaque scénario appelle à envisager le développement d'outils d'aide à la décision permettant d'utiliser efficacement le temps du concepteur et lui fournir des solutions optimales du point de vue de certains critères. La qualité des résultats établis par les outils d'aide à la décision dépendra fortement de la pertinence des méthodes employées à chaque étape, mais aussi de leur cohérence. Ainsi notre premier objectif était de développer une méthodologie cohérente, apte à constituer la charpente d'un outil d'aide à la décision efficace.

L'analyse des approches existantes dans la littérature pour la conception des systèmes d'usinage en avant-projet, notamment [Jacobe, 1992; Martin *et al.*, 2001], fait constater l'absence des méthodes d'optimisation efficaces à appliquer à l'étape « Choix d'une configuration du système d'usinage ». Ceci représente une lacune très importante, car le coût et l'efficacité du système d'usinage conçu dépendent fortement de la qualité des résultats obtenus à cette étape.

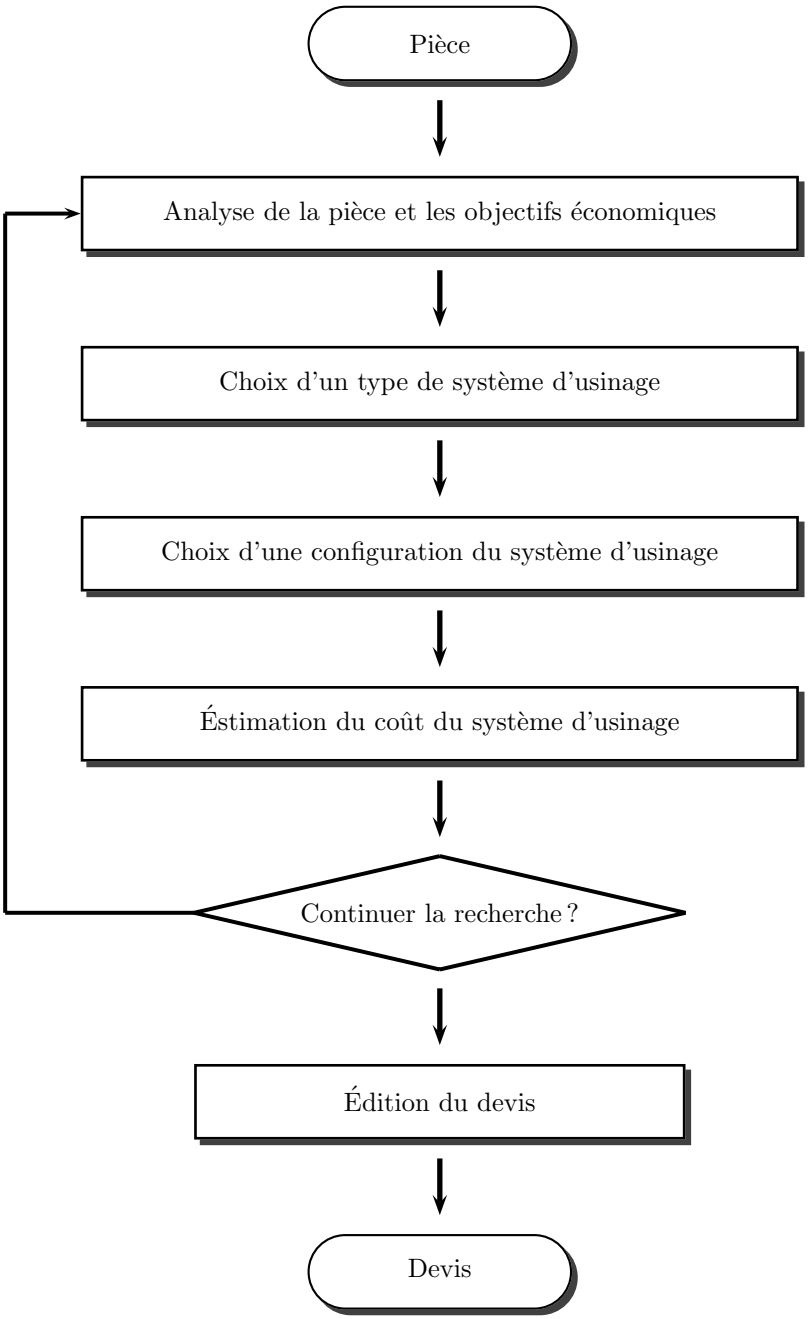


FIG. 2.1 – Conception des systèmes d'usinage en avant-projet

De plus, il faut constater que chaque type de système d'usinage amène des contraintes propres à prendre en compte, pour lesquelles des modèles mathématiques différents sont nécessaires. En conséquence, il semble peu probable de trouver un modèle mathématique et une méthode de résolution convenables à tout système d'usinage. La voie de l'élaboration des méthodes dédiées à chaque type de système d'usinage paraît plus raisonnable. Alors, nous optons pour le développement de modèles mathématiques et de méthodes de résolution pour

le problème d'optimisation de la configuration des systèmes d'usinage à boîtiers multibroches. Ce choix s'explique, d'une part, par un réel besoin industriel car jusqu'à maintenant la conception de ces systèmes s'appuie essentiellement sur le savoir-faire des experts ; et d'autre part, par la curiosité scientifique, engendrée, comme nous le démontrerons par la suite, par l'absence de modèles mathématiques applicables à la conception de ce type de systèmes.

En analysant le problème d'optimisation de la configuration des systèmes d'usinage à boîtiers multibroches, nous avons pu constater qu'à cette étape il est impératif de répondre aux questions suivantes :

- Combien de postes de travail convient-il d'installer ?
- Combien de boîtiers mettre à chaque poste de travail ?
- Quelles opérations exécuter par chaque boîtier ?
- Quelles conditions de coupe imposer à chaque boîtier ?
- Dans quel ordre activer les boîtiers installés sur chaque poste de travail (si le mode d'activation est séquentiel ou mixte) ?

Les réponses à ces questions déterminent non seulement la productivité du système d'usinage, mais aussi son coût. Par conséquent, une mauvaise décision retenue à cette étape se répercute sur la qualité des résultats d'avant-projet et peut amener à une non signature du contrat avec le client ou à l'impossibilité de respecter les spécifications du contrat à la phase de fabrication du système. Afin de trouver une solution efficace, il convient de considérer toutes les solutions possibles d'affectation des opérations (des outils) à des boîtiers et des boîtiers à des postes de travail tout en respectant les règles technologiques et techniques. Malheureusement, le nombre de solutions croît avec le nombre d'opérations de manière exponentielle. De plus, lors de la construction des solutions il convient de tenir compte d'un nombre important des contraintes liées d'une part au processus de fabrication et d'autre part aux caractéristiques des équipements. En outre, le fait que le temps opératoire de chaque bloc dépend de l'ensemble des opérations qui y sont groupées et la nécessité de respecter la cadence requise ne font que compliquer les calculs.

Notre démarche consiste à formuler ce problème comme un problème d'optimisation combinatoire et à utiliser des méthodes de la recherche opérationnelle pour sa résolution. À cette fin, nous nous sommes fixés les trois objectifs suivants :

- déterminer et modéliser de façon formelle les données d'entrée pour ce problème d'optimisation tout en s'assurant que toutes les contraintes technologiques et techniques soient prises en compte,
- formuler mathématiquement le problème d'optimisation en fixant un critère d'optimisation approprié et la structure des résultats nécessaires,
- proposer des méthodes de résolution efficaces pour la résolution du problème formulé.

Dans ce chapitre, nous nous attachons à l'analyse des étapes de la conception en avant-projet des systèmes d'usinage, présentés dans la Figure 2.1, afin d'atteindre les deux premiers objectifs. Il est à souligner que le développement des méthodes pour les étapes autres que le « Choix d'une configuration du système d'usinage » ne sera pas abordé : nous donnons tout simplement un aperçu des méthodes existantes dans la littérature. Quant au dernier objectif concernant la résolution du problème d'optimisation de la configuration du système d'usinage, les prochains chapitres lui seront consacrés.

Dans le cadre d'un appel d'offre, l'entreprise fabriquant des systèmes d'usinage reçoit un dessin de définition d'une pièce. La constitution de l'ensemble des opérations nécessaires pour l'usinage de cette pièce représente la première tâche décisionnelle dans la démarche de conception du système d'usinage correspondant. Dans la section suivante, nous survolons les méthodes existantes pour l'accomplissement de cette tâche.

2.3 Détermination des opérations d'usinage

La spécification de la pièce peut être fournie sous des formes différentes :

- Dans le meilleur des cas, la conception de la pièce a été réalisée à l'aide d'un logiciel CAO (pour Conception Assistée par Ordinateur) dont le format est reconnu par le logiciel CAO employé par l'entreprise.
- Dans d'autres cas, le dessin de définition de la pièce est fourni sur papier ou dans un format électronique non compatible avec le logiciel CAO employé par l'entreprise (image, CAO spécifique, etc.).

Le but est donc de définir les opérations d'usinage nécessaires afin de passer de la pièce brute à la pièce finie tout en obtenant la qualité requise. Le cas échéant, l'ensemble des opérations d'usinage peut être déterminé directement à partir du dessin de définition de la pièce en appliquant des règles « métier ». Néanmoins, la plupart des méthodologies développées optent pour une modélisation de la pièce par entités. Le terme « entité », initialement nommé « caractéristique » en France, correspond au terme anglais « feature ». Ce concept a émergé dans la fin des années 80, engendré par la volonté d'associer avec un objet sémantique un ensemble des caractéristiques utiles de point de vue de la conception et de la fabrication afin de permettre la communication et la coopération entre les experts de deux métiers.

2.3.1 Modélisation des pièces par entités

La modélisation des pièces mécaniques par entités présente de nombreux avantages :

- Tout d'abord, une telle modélisation permet de structurer les connaissances et de capitaliser le savoir-faire en conception des gammes d'usinage. Ceci permet de prendre en compte plus facilement les différentes contraintes technologiques et rend possible la mise en œuvre d'un mécanisme d'établissement quasi-automatique du processus d'usinage de la pièce.
- Ensuite, l'approche intégrée de l'élaboration des gammes d'usinage s'appuyant sur le modèle de la pièce en entités facilite la gestion de solutions alternatives.
- Enfin, l'intégration des connaissances liées au produit et au processus de fabrication contribue à l'augmentation du patrimoine technologique de l'entreprise et à la maîtrise de la conduite de l'activité de conception lors de nouveaux projets.

Même si ce concept semble être unanimement adopté, la notion d'« entité » reste implicite et varie d'un auteur à l'autre. Par exemple, un trou taraudé peut être représenté, selon la granularité choisie, soit par une entité « trou taraudé », soit par deux entités : un « perçage » et un « taraudage ». Toutefois, en se basant sur les définitions développées par [Shah et

Mantyla, 1995] et [Groupe GAMA, 1990], nous retenons la définition suivante : une entité est une collection d'*éléments géométriques* et un ensemble de *spécifications* liés, auxquels correspond une méthode ou un processus de fabrication particulier. Ce processus est quasi indépendant des processus d'autres entités.

Il faut constater qu'il n'existe pas une seule et unique taxinomie des entités valable quelle que soit l'activité de fabrication considérée, puisque l'identification d'une entité repose sur des conditions définies implicitement : une condition d'existence d'un processus d'usinage qui peut lui être associé, et une condition de quasi-indépendance entre les processus d'usinage des différentes entités de la pièce. Par conséquent, la construction du modèle de pièce en entités n'est pas triviale, elle est en forte dépendance du contexte de fabrication considéré et nécessite l'application d'un savoir-faire approprié. Pour cette raison, la plupart des logiciels CAO actuels ne proposent pas à l'utilisateur la possibilité de modéliser les pièces en utilisant des entités. Pourtant, dans la littérature, les trois voies suivantes ont été proposées pour mettre en œuvre une modélisation de pièces en entités plus ou moins automatisée :

1. **La conception par entités** : l'identification d'entités et leur création dans le modèle repose sur le concepteur. Pour faciliter le processus de conception, une bibliothèque de types d'entités prédéfinis est mise à disposition. Alors, il convient de choisir le type d'entité conforme à la forme géométrique du modèle de la pièce et de spécifier les paramètres de dimensions et de positionnement ainsi qu'un certain nombre d'autres caractéristiques intrinsèques et extrinsèques de l'entité. Il faut souligner que cette voie est la seule possible si le dessin de définition de la pièce est fourni dans un format non compatible avec le logiciel CAO employé par l'entreprise. Nous citons quelques exemples de systèmes employant cette technique : [Cutkosky *et al.*, 1988; Dolgui *et al.*, 2005*b*].
2. **La définition interactive d'entités** : l'identification d'entités et de relations entre elles reste à la charge du concepteur, tandis que leur création géométrique et sémantique devient automatisée. Alors, il convient de sélectionner une portion du modèle géométrique de la pièce qui correspond à une entité et le type d'entité y résidant. La création de l'entité sera réalisée automatiquement par un algorithme de transformation des informations géométriques en paramètres de dimensions et de positionnement. Nous pouvons citer FRBOM (Feature and Relation Based Oriented Modeling) [Salomons, 1995; Geelink, 1996] comme un exemple de système employant cette technique.
3. **La reconnaissance automatique d'entités** : l'identification et la création d'entités sont faites sans intervention du concepteur, par un algorithme sophistiqué qui compare des portions du modèle géométrique à des entités génériques prédéfinies. L'avantage de cette méthode est son indépendance par rapport au concepteur. Cependant, son application est limitée aux cas prévus par l'algorithme utilisé : cette limitation concerne les formats de données et l'ensemble des entités dont la reconnaissance est possible par l'algorithme. D'autres inconvénients sont : le temps de résolution important pour des cas difficiles ainsi que des fautes possibles de reconnaissance en présence d'entités similaires. Toutefois, les avantages potentiels de cette technique ont suscité l'intérêt de plusieurs chercheurs et de nombreuses approches ont été développées ; nous ne citons que quelques publications sur ce sujet [Tseng et Joshi, 1994; Trabelsi *et al.*, 1995; Xu

et Hinduja, 1998; Little *et al.*, 1998; Bezdek *et al.*, 1999; Shah *et al.*, 2001; Miao *et al.*, 2002; Derigent, 2005].

Nous pouvons constater que les méthodes de modélisation de pièces par entités sont nombreuses et le choix de la méthode à employer dépend du contexte mais aussi du format de la définition de la pièce.

2.3.2 Détermination des processus d'usinage des entités

L'entité possède la particularité de pouvoir être associée de manière subjective à un ou plusieurs processus d'usinage possibles. Nous sous-entendons par « processus d'usinage » d'une entité l'ensemble des opérations nécessaires pour l'usinage de cette entité, leurs paramètres et les contraintes entre elles. La modélisation d'une pièce sous la forme d'entités se fait essentiellement en vue d'obtenir de manière quasi-automatique l'ensemble des processus d'usinage correspondants.

Néanmoins, afin de parvenir à cette génération quasi-automatique, il convient de modéliser dans un système d'information le processus de raisonnement d'un expert. Ceci est loin d'être facile. En effet, pour chaque type d'entité, il existe un ensemble des processus possibles. Le choix du processus à appliquer se fait en fonction des critères de différents types, appelés *critères contextuels*. Cette appellation englobe tous les critères qui peuvent influencer le choix d'un processus par rapport à un autre, notamment :

- critères géométriques : tolérances, dimensions, défaut de forme, volume de matière à enlever...
- critères topologiques : intersections et relations entre entités...
- critères économiques et temporels : productivité exigée, coût de fabrication...
- critères technologiques : états de surface, matériaux, outillages...

Alors, le premier problème consiste à formaliser toutes les informations qui peuvent être utilisées par un critère contextuel. Ce problème est résolu en partie par l'attribution de ces informations aux entités sous forme des paramètres intrinsèques et extrinsèques. Néanmoins, une partie des informations reste imprécise à cette étape à cause des décisions qui seront prises ultérieurement, comme, par exemple, le choix des ressources. Face à ce problème, deux solutions sont envisageables :

- soit faire intervenir des règles « métier » afin de choisir le processus d'usinage qui convient le mieux pour chaque entité ; au cas où ce choix s'avérerait inefficace lors des étapes postérieures, il faudrait revenir à l'étape de la détermination des opérations pour effectuer un autre choix afin de reconstruire ensuite une nouvelle gamme d'usinage, etc. ;
- soit garder toutes les variantes possibles jusqu'à la fin de l'élaboration de la gamme, ce qui augmente considérablement la taille du problème à traiter et sa difficulté, mais garantit la prise en compte de toutes les solutions possibles.

Dans cette étude, nous nous limitons au premier cas et nous supposons que, pour chaque entité, un processus d'usinage le plus approprié est établi en appliquant des règles « métier » issues de la pratique d'utilisation des systèmes d'usinage à boîtiers multibroches. La formalisation de ces règles « métier » dans un système d'information fait appel à des méthodes de représentation formelle des connaissances. Dans ce domaine, plusieurs techniques ont été

élaborées et la plupart d'entre elles ont déjà été employées dans des systèmes de génération automatique de gammes d'usinage. Des études bibliographiques menées, par exemple dans [Derras, 1998; Derigent, 2005], soulignent la diversité des approches proposées et mises en œuvre. Nous pouvons conclure, hélas, qu'il n'existe pas d'approche universelle mais une pléthore de travaux répondant à des aspects très précis dans des contextes différents [Lombard, 2006].

Pour optimiser la configuration d'un système d'usinage, il faut tenir compte des processus d'usinage choisis pour les entités utilisées pour modéliser la pièce à fabriquer. Dans notre modèle mathématique, ces données sont représentées par les opérations d'usinage et leurs paramètres ainsi que par les contraintes entre ces opérations. Ces contraintes seront utilisées à l'étape de l'optimisation afin d'éviter les configurations du système d'usinage irréalisables en pratique. Dans ce qui suit, nous présentons d'abord la modélisation des opérations d'usinage et ensuite la modélisation des contraintes.

2.3.3 Modélisation des opérations d'usinage

En parcourant l'ensemble des processus d'usinage choisis pour l'ensemble d'entités, nous pouvons constituer l'ensemble des opérations d'usinage à exécuter pour fabriquer la pièce. Nous désignons cet ensemble par $\mathbf{N}$, où $i \in \mathbf{N}$ correspond à l'opération ayant le numéro i.

Dans le Chapitre 1, nous avons expliqué le lien entre le temps d'usinage d'un ensemble d'opérations par un boîtier multibroche et les paramètres des opérations appartenant à cet ensemble. Afin de pouvoir en tenir compte lors de la résolution de configuration du système d'usinage, il faut que les valeurs des paramètres suivants soient déterminées pour chaque opération $\forall\, i \in \mathbf{N}$:

l_i	est la longueur de la course d'outil pour l'opération i ;
$[v_{f1}(i), v_{f2}(i)]$	est l'intervalle des valeurs admissibles de la vitesse d'avance d'outil pour l'opération i.

Lors de la répartition des opérations, deux catégories de contraintes doivent être prises en compte : celles qui dépendent du type de système d'usinage à boîtiers multibroches à concevoir et celles qui en sont indépendantes. Dans ce qui suit, nous ne présentons que les contraintes de la deuxième catégorie. Les contraintes de la première catégorie sont décrites dans la Section 2.4.2.

2.3.4 Modélisation des contraintes entre les opérations d'usinage

Les règles « métier » employées à la phase de la détermination des processus d'usinage pour les entités induisent plusieurs types de contraintes entre les opérations. Dans ce mémoire, nous n'abordons que la modélisation de ces contraintes. Le lecteur intéressé par la problématique d'élaboration de ces contraintes est invité à consulter les publications suivantes : [Jacobe, 1992; Derras, 1998; Martin *et al.*, 2001; Lefebvre *et al.*, 2001; Renaud *et al.*, 2001; Deneux, 2002].

Plusieurs formalismes ont été proposés dans la littérature pour la modélisation des contraintes entre opérations. Par exemple, il est possible d'utiliser une logique temporelle

représentant des relations entre les opérations comme cela a été fait dans [Martin *et al.*, 2001], ou la logique floue comme dans [Derras, 1998]. Dans ce mémoire, pour tenir compte des contraintes entre les opérations d'usinage, nous introduisons les modèles suivants.

Contraintes de précédence

Les contraintes de précédence expriment l'ordre du déroulement des opérations, imposé par la technologie d'usinage utilisée. Par exemple, il n'est pas possible de tarauder un trou sans l'avoir percer auparavant. Dans ce cas, on dit que l'opération de perçage d'un trou doit « précéder » l'opération de taraudage du même trou. De manière générale, les contraintes de précédence tiennent compte des séquences fixes d'usinage des faces, des stratégies d'usinage des intersections d'entités, de l'existence de l'usinage en ébauche, demi-finition et finition, etc. L'ordre dans lequel vont être réalisées les opérations d'usinage a une influence sur la qualité de la pièce, il faut donc en tenir compte lors de la création de contraintes de précédence.

Ce type de contraintes peut être modélisé par un graphe orienté G^{pr}=($\mathbf{N}, D^{pr}$), dans lequel un arc $\{i, j\} \in D^{pr}$ si l'opération i ne peut pas être exécutée après l'exécution de l'opération j. Dans notre modèle, qui permet l'exécution d'opérations en parallèle, les contraintes de précédence sont « non strictes », cela signifie que si un arc $\{i, j\} \in G^{pr}$, alors les opérations i et j peuvent être affectées au même bloc, c'est-à-dire peuvent être exécutées **simultanément** par le même boîtier. Une telle modélisation permet de tenir compte de la possibilité éventuelle de l'utilisation des outils étagés, introduits dans la Section 1.1.3. Seule est interdite l'exécution de l'opération j dans un bloc précédant celui où l'opération i est faite.

À partir du graphe G^{pr}, nous pouvons construire les ensembles suivants que nous utiliserons dans la suite du manuscrit.

$PredD(j)$	l'ensemble des prédécesseurs directs de l'opération $j \in \mathbf{N}$;
$PredT(j)$	l'ensemble de tous les prédécesseurs de l'opération $j \in \mathbf{N}$;
$SuccD(j)$	l'ensemble des successeurs directs de l'opération $j \in \mathbf{N}$;
$SuccT(j)$	l'ensemble de tous les successeurs de l'opération $j \in \mathbf{N}$.

Contraintes d'exclusion au niveau des blocs

Les contraintes d'exclusion au niveau des blocs viennent de l'impossibilité d'exécuter certaines opérations par le même boîtier. Par exemple, les opérations correspondantes à deux faces différentes de la pièce ne peuvent pas être réalisées par le même boîtier. Des contraintes de ce type sont aussi engendrées par l'impossibilité de la fixation de certains outils dans un boîtier commun, l'inexistence d'un outil étagé conforme, l'influence réciproque nuisible des opérations, l'inaccessibilité de certaines zones de travail par les outils, etc.

Ce type de contraintes peut être représenté par une collection E^b de sous-ensembles $e \subset \mathbf{N}$ d'opérations, tels qu'au moins une opération de chaque ensemble e ne doit pas être affectée au même bloc que les autres.

Notons que s'il est nécessaire d'imposer une contrainte de précédence « stricte » entre deux opérations i et j pour interdire l'affectation des opérations i et j au même bloc, il

convient d'introduire une *contrainte d'exclusion au niveau des blocs* entre les opérations i et j.

Contraintes d'inclusion au niveau des blocs

Les contraintes d'inclusion au niveau des blocs traduisent la nécessité d'exécuter certaines opérations par le même boîtier, notamment afin de respecter la tolérance de position requise.

Les opérations liées par ce type de contraintes peuvent être replacées par des macro-opérations en utilisant l'Algorithme 5.6. Dans la suite du manuscrit nous supposons que toutes les transformations de ce type ont déjà été effectuées.

2.4 Choix du type de système d'usinage

2.4.1 Modélisation des paramètres des systèmes d'usinage

Étant donné que nous nous sommes limités au cas des systèmes d'usinage à boîtiers multibroches, le choix du système à concevoir consiste à déterminer le mode d'activation de boîtiers, qui peut être parallèle, séquentiel ou mixte (pour plus de détails voir la section 1.2.4 du manuscrit), mais aussi les paramètres suivants :

- m_0 : le nombre maximum autorisé de postes de travail du système ;
- $n_{||}$: le nombre maximum autorisé de boîtiers parallèles par poste ;
- $n_{\exists}$: le nombre maximum autorisé de boîtiers qu'il est possible de mettre en séquence sur un poste de travail ;
- n_0 : le nombre total maximum autorisé de boîtiers qu'il est possible de mettre sur un poste de travail :

$$n_0 = \begin{cases} n_{||} & \text{si le mode d'activation de boîtiers est parallèle,} \\ n_{\exists} & \text{si ce mode est séquentiel,} \\ n_{||} \cdot n_{\exists} & \text{si le mode est mixte} \end{cases}$$

- τ^p : le temps auxiliaire au niveau d'un poste de travail ;
- τ^b : le temps auxiliaire au niveau d'un bloc ;
- T_0 : le temps de cycle d'usinage à ne pas dépasser (calculé sur la base de la productivité requise et du rendement du système d'usinage).

Les valeurs des six premiers paramètres sont déterminées en tenant compte du type du système d'usinage à concevoir ainsi que des dispositifs choisis pour le transfert de pièce et l'avancement de boîtiers.

Les valeurs des paramètres τ^p et τ^b sont définies par une estimation des temps non productifs avant et après la phase d'usinage aussi bien au niveau des postes de travail qu'au niveau des blocs. Nous les considérons fixes et les mêmes pour tous les postes et les blocs. Ces temps ne comprennent pas le temps d'usinage, qui sera calculé en fonction des opérations affectées à chaque boîtier et à chaque poste de travail.

Enfin, rappelons que la productivité requise par le client définit le temps de cycle du système d'usinage T_0 (pour plus de détails voir la Section 1.2.5 du manuscrit).

2.4.2 Modélisation des contraintes au niveau des postes de travail

Après la sélection d'un mode d'activation de boîtiers, nous devons établir des contraintes d'exclusion et d'inclusion au niveau des postes de travail. La modélisation de ces contraintes est similaire à la modélisation des contraintes d'exclusion et d'inclusion au niveau des blocs, présentée dans la Section 2.3.4.

Contraintes d'exclusion au niveau des postes de travail

Les contraintes d'exclusion au niveau des postes de travail viennent de l'impossibilité d'exécuter certaines opérations sur le même poste de travail. Par exemple, si le mode d'activation de boîtiers est parallèle, il est impossible de mettre sur le même poste deux boîtiers agissant sur la même face de la pièce.

Une contrainte de ce type peut être représentée par une collection E^p de sous-ensembles $e \subset \mathbf{N}$ d'opérations, tels qu'au moins une opération de chaque ensemble e ne doit pas être affectée au même poste de travail que les autres opérations de cet ensemble.

Contraintes d'inclusion au niveau des postes de travail

Les contraintes d'inclusion au niveau des postes de travail traduisent la nécessité d'exécuter certaines opérations sur le même poste de travail, notamment afin de respecter la tolérance de position requise.

Ce type de contraintes peut être représenté par une collection I^p de sous-ensembles $e \subset \mathbf{N}$ d'opérations, tels que toutes les opérations appartenant à l'ensemble e doivent être impérativement affectées au même poste de travail.

Dans la suite du manuscrit, nous utiliserons l'appellation « contraintes de compatibilité » qui englobe les contraintes d'exclusion et d'inclusion aussi bien au niveau des blocs qu'au niveau des postes de travail.

Notons qu'à ce stade il est également possible d'indiquer les posages faisables de la pièce. La considération des posages réalisables permet d'enrichir les contraintes d'exclusion et d'inclusion au niveau des postes de travail ainsi que les contraintes de précédence pour les opérations. Nous nous référons à [Paris, 1995] pour l'étude des prises de pièce et nous ne traitons pas cet aspect en détails ici.

Le problème de la configuration des systèmes d'usinage à boîtiers multibroches considéré dans ce mémoire consiste donc à trouver une répartition des opérations à des boîtiers et à des postes de travail satisfaisant toutes les contraintes. Dans la section suivante, nous présentons un modèle générique pour ce problème.

2.5 Formulation mathématique du problème d'optimisation

Si les contraintes sont utilisées pour distinguer les solutions admissibles des solutions irréalisables, un critère objectif, lui, permet de comparer les solutions admissibles et en

choisir la meilleure. Dans ce qui suit nous présentons le critère que nous avons choisi pour notre problème d'optimisation.

2.5.1 Formulation du critère

En génie industriel, les trois types de critères sont unanimement utilisés afin d'évaluer la qualité de solutions, à savoir : coût, qualité et délais. Nous supposons que la qualité d'usinage sera assurée par les contraintes interdisant les affectations d'opérations qui ne respectent pas la qualité requise. Le critère de délais dans notre cas impose la minimisation du temps de conception des systèmes d'usinage, qui sera réduit par le fait de recourir à l'approche d'aide à la décision que nous développons. Ainsi, nous nous limitons au critère de coût. Nous pouvons intégrer les coûts liés aux ressources mises en place des deux façons suivantes :

- soit en opérant des coûts précis et en les calculant pour chaque affectation des opérations, un peu comme le calcul du temps d'usinage, ce qui compliquerait encore plus la procédure de conception,
- soit en se limitant aux coûts les plus importants en s'assurant qu'en aucun cas que la solution retenue à la phase d'avant-projet ne sera pas remise en question ultérieurement, par le calcul des coûts exacts.

La deuxième voie semble plus réaliste puisqu'elle permet d'alléger les calculs qui sont déjà chargés par le nombre de contraintes à considérer. D'autant plus qu'à l'étape de la conception en avant-projet, il est difficile d'avoir les valeurs exactes de certains coûts. Après avoir analysé les éléments qui constituent le coût d'un système d'usinage à boîtiers multibroches, nous avons pu constater que les coûts *structurants* sont définis par le nombre de postes de travail et le nombre de boîtiers multibroches. Un autre élément important, qui est le coût des outils coupants, peut être supposé indépendant de l'affectation des opérations puisqu'à l'étape du choix de la configuration du système d'usinage nous admettons que les opérations à effectuer sont déjà choisies, ce qui signifie implicitement que les outils de coupe sont également prédéterminés, puisque l'outil est associé directement à l'opération.

Ainsi les deux paramètres suivants sont utilisés pour introduire les coûts d'équipement dans le modèle mathématique :

C_1 le coût relatif d'un poste de travail ;

C_2 le coût relatif d'un bloc.

La valeur C_2 comprend tous les coûts liés aux parties porteuses, parties opératives, actionneurs, dispositifs d'injection de liquide, etc. Quant à la valeur de C_1, elle tient compte des coûts qui ne dépendent pas du nombre de boîtiers installés sur un poste de travail, comme par exemple le coût de dispositifs de transfert de pièce, d'activation des boîtiers, etc.

Les paramètres C_1 et C_2 peuvent aussi être utilisés pour traduire tout simplement la préférence du concepteur dans la situation où il est possible soit d'ajouter un bloc complémentaire à un poste de travail déjà existant, soit d'ajouter un poste de travail complémentaire. Dans ce cas, c'est le rapport le C_2/C_1 qui joue un rôle prépondérant. Pour le choisir, le concepteur peut se référer aux coûts de la partie fixe d'un boîtier et d'un poste d'usinage respectivement (qui ne dépend pas des opérations affectées) et/ou se baser sur son expérience dans le domaine. Dans la pratique, le rapport C_2/C_1 varie entre 0,2 et 0,6 [Finel, 2004].

Dans ce qui suit, nous récapitulons l'ensemble des hypothèses que nous admettons pour la formulation du problème d'optimisation.

2.5.2 Hypothèses de base

1. Le système d'usinage à concevoir consiste en une séquence de postes de travail sans stock intermédiaire. Chaque poste est muni d'un ou de plusieurs boîtiers multibroches. La pièce ne peut être déplacée vers le poste de travail suivant que si toutes les opérations d'usinage sont achevées sur le poste courant.
2. Tous les postes de travail sont identiques en terme du mode d'activation de boîtiers et du nombre maximum autorisé de boîtiers.
3. Le nombre maximum de postes de travail est limité par l'architecture du système d'usinage.
4. Le coût de base d'un poste de travail ne dépend pas du nombre de boîtiers qui y sont installés, il est le même pour tous les postes de travail mis en place.
5. Tous les boîtiers ont le même coût qui ne dépend ni du poste de travail où ils sont installés, ni de l'ordre de leur activation, ni du nombre des opérations qui leur seront affectées. A priori chaque boîtier peut être monté à n'importe quel poste de travail s'il n'y a pas de conflit avec d'autres boîtiers de ce poste de travail.
6. L'ensemble des opérations à entreprendre est connu ; chaque opération doit être effectuée une seule fois, ce qui se traduit par son affectation à un et un seul boîtier d'un et d'un seul poste de travail.
7. Il y a un temps de cycle objectif à respecter. Le calcul du temps de cycle réel du système d'usinage dépend du mode de transfert de la pièce, du mode d'activation de boîtiers et de la règle de calcul des paramètres d'usinage pour les boîtiers.
8. Des contraintes de précédence imposent un ordre partiel du déroulement d'opérations qu'il faut respecter.
9. Des contraintes d'inclusion et d'exclusion pour les opérations existent aussi bien au niveau des postes de travail qu'au niveau des blocs, et doivent être respectées.
10. L'objectif de l'optimisation est de minimiser le coût de la configuration du système d'usinage à boîtiers multibroches qui est estimé par le coût total de tous les postes de travail mis en place et tous les boîtiers qui y sont installés.

L'ensemble des données suivant, représenté par une collection P, doit être alors connu pour démarrer l'optimisation de la configuration du système d'usinage à boîtiers multibroches :

$P = <\mathbf{N}, \{ l_i \mid i \in \mathbf{N}\}, \{ v_{f1}(i) \mid i \in \mathbf{N}\}, \{ v_{f2}(i) \mid i \in \mathbf{N}\}, m_0, n_0, G^{pr}, E^p, E^b, I^p, T_0, \tau^p, \tau^b, C_1, C_2 >$.

Avant de présenter la formulation mathématique du problème d'optimisation, nous proposons quelques procédures permettant de vérifier la cohérence de ces données.

2.5.3 Cohérence des contraintes

Il peut advenir que les données de départ soient inconsistantes, c'est-à-dire qu'il n'existe pas de solution satisfaisant l'ensemble de contraintes. Cela peut être le résultat soit d'une mauvaise saisie de données, soit de la présence de contraintes contradictoires [Guschinskaya *et al.*, 2007*b*]. Il est fort souhaitable de détecter de tels cas à l'étape de pré-traitement afin d'éviter une perte de temps en essayant de résoudre un problème qui n'a pas de solution. Dans ce qui suit, nous proposons quelques règles permettant de repérer des contradictions dans les contraintes au travers d'une analyse de données de départ.

Il n'existe pas de solution admissible pour un ensemble de données si au moins une des conditions ci-dessous est vérifiée.

Condition 1 Violation de la contrainte de temps de cycle :

$$\exists\, j \in \mathbf{N} \text{ tel que } l_j / v_{f2}(j) > T_0 - \tau^p - \tau^b$$

L'exécution simultanée des opérations, assurée par des boîtiers multibroches, est utilisée afin de diminuer le temps total d'usinage. Cependant, en aucun cas, il n'est possible de diminuer le temps d'une opération. Comme le montre (1.14), le temps d'exécution d'un ensemble d'opérations est supérieur ou égal au temps d'exécution de chaque opération appartenant à cet ensemble. Alors, s'il existe une opération pour laquelle la condition 1 est vérifiée, la productivité requise ne sera pas assurée quelle que soit la répartition des opérations.

Condition 2 Incohérence des contraintes de compatibilité :

$$\exists\, e_1, e_2 \text{ tel que } e_1 \in E^p,\ e_2 \in I^p \text{ et } e_1 \subseteq e_2$$

La contrainte e_2 exige que toutes les opérations appartenant à e_2 soient affectées au même poste de travail, tandis que la contrainte e_1 interdit une telle affectation ; il est évident qu'une solution respectant les deux contraintes contradictoires n'existe pas.

Condition 3 Incohérence entre des contraintes de compatibilité et de précédence :

$\exists\, e_1, e_2$ tels que :

1. $e_1 = \{i_1, j_1\} \in E^p$, $i_1 \in PredT(j_1)$,
2. $e_2 = \{i_2, j_2\} \in I^p$; $i_2 \in PredT(i_1)$ ou $i_1 = i_2$; et $j_1 \in PredT(j_2)$ ou $j_1 = j_2$.

La contrainte d'exclusion e_1 rend impossible l'affectation au même poste de tout prédécesseur de l'opération i_1 avec tout successeur de l'opération j_1, tandis que la contrainte e_2 suscite une telle affectation. Ainsi, les contraintes e_1 et e_2 sont contradictoires et aucune solution les respectant à la fois ne peut exister.

Dans le cas où une incohérence des informations de départ est détectée, il est nécessaire de réviser les données saisies et de corriger celles qui sont à l'origine de la contradiction. Par ailleurs, il peut advenir que l'incohérence ne vienne pas des données mal saisies mais du fait que le type de système d'usinage choisi ne corresponde pas au processus d'usinage nécessaire. Dans ce dernier cas, il convient de changer le type de système à concevoir et reprendre le processus d'optimisation de sa configuration.

2.5.4 Transformation des contraintes

Les propositions ci-dessous permettent de s'affranchir des contraintes redondantes.

Proposition 2.1 *Si* $\exists\, e_1 = \{i_1, j_1\}$, $\exists\, e_2 = \{i_2, j_2\}$ *tels que* $\{i_1, j_1\} \in D^{pr}$, *de plus, soit* $\{i_2, i_1\} \in D^{pr}$, *soit* $i_1 = i_2$; *en même temps, soit* $\{j_1, j_2\} \in D^{pr}$, *soit* $j_1 = j_2$, *et en outre, une des conditions suivantes est satisfaite* :

Cas 1 : $e_1, e_2 \in E^b$

Cas 2 : $e_1, e_2 \in E^p$

Cas 3 : $e_1 \in E^p, e_2 \in E^b$

alors la contrainte e_2 *est redondante et peut être supprimée de l'ensemble* E^b (*Cas* 1, 3) *ou de l'ensemble* E^p (*Cas* 2) *sans modifier l'ensemble des solutions admissibles pour le problème initial.*

Démonstration. La contrainte d'exclusion e_1 rend impossible l'affectation de tout prédécesseur de l'opération i_1 au même poste de travail (Cas 2, 3) ou au même bloc (Cas 1) avec n'importe quel successeur de l'opération j_1. Ainsi, les contraintes entre les prédécesseurs de l'opération i_1 et les successeurs de l'opération j_1 sont redondantes et peuvent être supprimées. Ceci est également vrai si $e_1 \in E^p, e_2 \in E^b$ (Cas 3), où la contrainte d'exclusion au niveau des postes de travail domine celle au niveau des blocs. Il est évident que si les opérations i_2, j_2 ne sont pas affectées au même poste de travail, ce qui découle de la contrainte e_1 et des contraintes de précédence, elles ne sont pas non plus affectées au même bloc, par conséquent, la contrainte e_2 est redondante. □

Proposition 2.2 *Si* $\exists\, e_1, e_2 \in I^p$ *tels que* $\exists\, i, j \in e_1, \exists\, h \in e_2$ *tels que* $\{i, h\} \in D^{pr}$ *ou* $i = h$, *ainsi que* $\{h, j\} \in D^{pr}$ *ou* $j = h$, *alors* e_1 *et* e_2 *peuvent être fusionnés* : $e_1 = e_1 \cup e_2$ *sans modifier l'ensemble des solutions admissibles pour le problème initial.*

Démonstration. La contrainte e_1 exige que les opérations i et j soient affectées au même poste, par exemple au poste k. Si l'opération i est affectée au poste k, alors il ne serait possible d'affecter l'opération h à aucun poste précédant le poste k à cause des contraintes de précédence ($\{i, h\} \in D^{pr}$). En même temps, si l'opération h est affectée à un poste succédant au poste k, alors l'affectation de l'opération j au poste k serait impossible à cause des contraintes de précédence ($\{h, j\} \in D^{pr}$), ce qui contredirait la contrainte e_1. Il en découle qu'afin de respecter toutes les contraintes données, il est impératif d'affecter les opérations i, h, et j au même poste. Ceci peut être traduit par l'union des contraintes e_1 et e_2. □

Proposition 2.3 *Si* $\exists\, e_1$, e_2 *tels que* $e_1 \in I^p$, $e_2 \in E^p$, *et* $|\, e_1 \cap e_2| > 1$, $e_2 \nsubseteq e_1$ *alors l'ensemble* e_2 *peut être modifié de la manière suivante* : $e_2 = e_2 \setminus (e_1 \cap e_2) \cup \{i\}$, *où* i *est une opération quelconque appartenant à* e_1.

Démonstration. La contrainte e_2 exige que toutes les opérations appartenant à l'ensemble e_2 ne soient pas affectées au même poste. Pour la respecter, au moins une opération doit être affectée à un poste différent des autres. Mais si une opération appartenant à l'ensemble e_1 est affectée à un poste, toutes les opérations appartenant à l'ensemble e_1 seront impérativement

affectées au même poste. En conséquence, en ne laissant qu'une seule opération de e_1 (n'importe laquelle) dans l'ensemble e_2, nous ne changerons pas l'ensemble de solutions admissibles pour le problème initial. □

Proposition 2.4 *Si* $\exists\, i, j, h \in e$, $e \in E^p$ *(ou* E^b*) tels que* $i \in PredT(j)$ *et* $j \in PredT(h)$, *alors l'opération* j *peut être supprimée de l'ensemble* $e : e = e \setminus \{j\}$.

Démonstration. Afin de respecter la contrainte e il suffit de ne pas affecter au moins une opération appartenant à l'ensemble e au même poste (bloc) que les autres. Cependant, nous pouvons conclure qu'à cause des contraintes de précédence, en aucun cas, il n'est possible d'affecter les opérations i et h au même poste (bloc) sans également y affecter l'opération j. En conséquence, en la supprimant de l'ensemble e, nous ne changerons pas l'ensemble de solutions admissibles pour le problème initial. □

2.5.5 Modélisation d'une solution

Nous introduisons les notations suivantes afin de pouvoir proposer une modélisation générique du problème d'optimisation de la configuration des systèmes d'usinage à boîtiers multibroches. Nous nous servirons par la suite de ce modèle générique pour le développement des modèles mathématiques dédiés à chaque type des systèmes considérés.

S	une solution du problème générique ;
m	le nombre de postes de travail dans la solution S ;
k	l'indice pour les postes de travail, $k = 1, 2, ..., m$;
N_k	l'ensemble des opérations affectées au poste de travail avec l'indice k ;
r_k	le nombre de blocs au poste de travail k ;
r	l'indice pour les blocs d'un poste de travail, pour le poste k : $r = 1, 2, ..., r_k$;
N_{kr}	l'ensemble des opérations affectées au bloc r du poste de travail k.

Avec ces notations, une solution admissible du problème P peut être représentée par une collection $S(P) = \{\{N_{11}, ..., N_{1\,r_1}\}, ..., \{N_{m1}, ..., N_{mr_m}\}\}$.

Soit $\mathbf{S}(P)$ l'ensemble de toutes les solutions admissibles pour un problème P, c'est-à-dire toutes les solutions respectant l'ensemble des contraintes du problème P. Pour y trouver la meilleure solution, nous utilisons une fonction objectif permettant d'estimer le coût de chaque solution. Ce coût est estimé en utilisant deux paramètres, C_1 et C_2, comme étant les coûts relatifs à l'installation d'un poste de travail et d'un boîtier, respectivement. Ainsi, nous utilisons la fonction objectif $C(S)$ présentée dans (2.1). La collection $S_{opt}(P) \in \mathbf{S}(P)$ telle que $C(S_{opt}) \leq C(S)$, $\forall S(P) \in \mathbf{S}(P)$, correspond à la solution optimale du problème P.

2.5.6 Modèle mathématique du problème d'optimisation

Le problème générique d'optimisation est formulé de la manière suivante :

$$Minimiser\ C(S) = C_1 m + C_2 \sum_{k=1}^{m} r_k \tag{2.1}$$

S. c. :

$$T(S) \leq T_0 \tag{2.2}$$

$$\bigcup_{k=1}^{m} \bigcup_{r=1}^{r_k} N_{kr} = \mathbf{N} \tag{2.3}$$

$$\begin{gathered} N_{k'r'} \cap N_{k''r''} = \emptyset, \\ \forall\, k', k'' = 1, ..., m, r' = 1, ..., r_{k'}, r'' = 1, ..., r_{k''}, \text{ où } (k', r') \neq (k'', r'') \end{gathered} \tag{2.4}$$

$$(k'-1)n_0 + r' \leq (k''-1)n_0 + r'', \forall\, i \in PredD(j), i \in N_{k'r'}, j \in N_{r''k''} \tag{2.5}$$

$$(N_k \cap e) \in \{\emptyset, e\}, \text{ pour } \forall\, e \in I^p \text{ et } k = 1, ..., m \tag{2.6}$$

$$e \not\subset N_{kr}, \text{pour } \forall\, e \in E^b \text{ et } k = 1, ..., m; r = 1, ..., r_k \tag{2.7}$$

$$e \not\subset N_r, \text{pour } \forall\, e \in E^p \text{ et } k = 1, ..., m \tag{2.8}$$

$$m \leq m_0 \tag{2.9}$$

$$r_k \leq n_0, \text{ pour } k = 1, ..., m \tag{2.10}$$

Le problème consiste donc à minimiser la fonction objectif (2.1) représentant une estimation du coût du système d'usinage à l'étape d'avant-projet. La contrainte (2.2) garantit que le temps de cycle $T(S)$ du système d'usinage n'excède pas la valeur T_0, permettant ainsi d'atteindre la productivité voulue. La technique de calcul de $T(S)$ dépend de l'architecture du système d'usinage, du mode d'activation des boîtiers et de la règle de calcul des paramètres d'usinage pour un bloc d'opérations. Des techniques propres à chaque type de systèmes d'usinage seront développées dans les chapitres suivants. L'équation (2.3) assure l'affectation de toutes les opérations de l'ensemble **N**. Les contraintes (2.4) interdisent l'affectation de chaque opération de l'ensemble **N** plus d'une fois. L'équation (2.5) définit les contraintes de précédence entre les opérations. L'équation (2.6) traduit les contraintes d'inclusion concernant les postes de travail. Les équations (2.7) et (2.8) représentent les contraintes d'exclusion relatives aux blocs (2.7) et aux postes de travail (2.8). L'équation (2.9) assure que le nombre de postes de travail ne dépasse pas le nombre maximum autorisé. L'équation (2.10) interdit le montage de plus de n_0 unités d'usinage sur un poste de travail.

La solution du problème (2.1)-(2.10) représente une ossature de la configuration du système d'usinage, elle comporte toutes les informations nécessaires pour effectuer l'agencement final du système d'usinage et établir un devis de sa fabrication.

Il faut souligner que des problèmes proches ont été traités principalement dans le contexte d'installation des lignes d'assemblage. Pour cette raison, nous analysons dans le Chapitre 3, les méthodes qui ont été élaborées pour leur résolution. Ensuite, nous développons des modèles mathématiques et des algorithmes de résolution pour le problème d'optimisation de la configuration de trois types de systèmes d'usinage à boîtiers multibroches : les Chapitres 4 et 5 sont consacrés aux machines de transfert, le Chapitre 6 traite le cas des machines à table circulaire pivotante et celui des machines à table mobile.

2.6 Méthodologie de la conception des systèmes d'usinage à boîtiers multibroches

Dans le Tableau 2.1, nous faisons une synthèse de la méthodologie pour la conception en avant-projet des systèmes d'usinage à boîtiers multibroches que nous avons développée.

TAB. 2.1 – Méthodologie de la conception en avant-projet des systèmes d'usinage à boîtiers multibroches

Actions	Résultats
Phase 1. Analyse de la pièce et le choix du type de système d'usinage	
Étape 1.1 Détermination des opérations d'usinage	
Modélisation de la pièce par entités	Modèle de la pièce en entités
Choix des processus d'usinage pour chaque entité	$\mathbf{N}$, $\{ l_i \mid i \in \mathbf{N}\}$, $\{ [v_{f1}(i), v_{f2}(i)] \mid i \in \mathbf{N}\}$, G^{pr}, E^b
Étape 1.2 Choix des paramètres des ressources	
Choix des paramètres du modèle	m_0, n_0, T_0, τ^p, τ^b, C_1, C_2
Formulation des contraintes entre les opérations et les ressources	E^p, I^p, mise à jour G^{pr}
Phase 2. Optimisation de la configuration du système d'usinage	
Étape 2.1 Recherche de la configuration optimale du système d'usinage	
Affectation des opérations aux boîtiers Définition des régimes d'usinage pour les boîtiers Affectation des boîtiers aux postes de travail	$S_{opt}(P)$
Étape 2.2 Validation de la solution à mettre en place	
Calcul et analyse des caractéristiques techniques du système d'usinage à l'appui de la solution élaborée	Plan d'usinage, Productivité du système d'usinage, etc.
Phase 3. Agencement et choix d'équipements	
Définition des posages de la pièce Analyse et choix d'équipements de montage Analyse et choix d'équipements de la partie opérative Analyse et choix d'équipements de la partie commande Calcul des caractéristiques du système d'usinage Analyse de la qualité d'usinage	Gamme d'usinage et Prototype du système d'usinage
Phase 4. Calcul du coût de revient et estimation du prix du système d'usinage	
Calcul du coût de revient des équipements mis en place Calcul du coût de revient du système d'usinage Analyse de la rentabilité et calcul du prix et des délais de fabrication Établissement de la documentation	Devis de fabrication du système d'usinage

Les Phases 1 et 2 ont été détaillées dans le texte précédent. La Phase 3 consiste à choisir

des équipements de montage, de la partie opérative et de la partie commande du système d'usinage. Lors de l'agencement du système d'usinage, il convient de prendre en compte les informations suivantes :

- les résultats de la résolution du problème 2.1-2.10 ;
- les posages de la pièce ;
- les paramètres admissibles des équipements ;
- les contraintes de compatibilité entre les équipements ;
- les contraintes d'agencement.

Afin d'évaluer la qualité du système conçu, il faut tenir compte des dispersions générées par l'assemblage des outils, des broches, des boîtiers, et par les dispositifs de montage, qui influencent directement la qualité de l'usinage. Souvent la simulation numérique est la seule voie possible pour estimer l'impact du comportement vibratoire des parties mécaniques et pour évaluer la qualité d'usinage fournie par le système conçu. Cette simulation peut être effectuée soit par un simulateur basé sur la modélisation par éléments finis, soit par un outil FAO, qui, lui, est mieux adapté au contexte d'usinage et regroupe des outils informatiques permettant de gérer, de planifier et de contrôler les opérations de fabrication. À l'aide d'un tel simulateur, il est possible de reproduire graphiquement (visualisation volumique) l'action des outils dans la matière première permettant ainsi au concepteur de vérifier ses méthodes d'usinage et d'éviter a priori les collisions d'outils lors de la fabrication de la pièce.

Enfin, la Phase 4 consiste à calculer le coût de revient et à estimer le prix du système d'usinage conçu. En vue d'évaluer le comportement du système sur l'horizon de son exploitation, on a recours à des outils de simulation de flux, permettant de modéliser des pannes, la maintenance et l'usure d'outils. Une telle simulation permet notamment de calculer plus précisément le coût de fonctionnement du système par unité de pièce fabriquée. Des modèles pour la simulation de systèmes d'usinage ont été développés, par exemple, dans [Inman et Leon, 1994; Masood, 2006]. À l'issue de cette étape de simulation, on dispose de toutes les informations nécessaires pour comparer les solutions envisageables en avant-projet et pour juger de la nécessité d'examiner d'autres types de systèmes d'usinage. Bien évidement, lorsqu'une solution finale est choisie il faut procéder à l'étape de l'élaboration du devis et de la documentation pour le client.

2.7 Conclusion

Ce chapitre a été dédié à l'analyse de la problématique de conception en avant-projet des systèmes d'usinage à boîtiers multibroches. Nous avons présenté les différentes étapes de cette phase de conception. Nous avons également donné un aperçu des méthodes développées pour supporter chaque étape. En nous appuyant sur ces résultats, nous avons proposé une procédure de la conception des systèmes d'usinage à boîtiers multibroches en avant-projet. Un autre résultat que nous avons obtenu dans ce chapitre est la formulation mathématique du problème générique d'optimisation de la configuration de ce type de systèmes. Le chapitre suivant est consacré à l'analyse des méthodes de résolution de problèmes d'optimisation proches qui ont été traités dans la littérature.

Chapitre 3

Optimisation de la configuration des chaînes de fabrication : État de l'art

Ordonner une bibliothèque est une façon silencieuse
d'exercer l'art de la critique.

Jorge Luis Borges

Dans ce chapitre, nous étudions les problèmes d'optimisation qui relèvent de la structuration des systèmes de fabrication, notamment dans le cadre d'installation de chaînes d'assemblage. Comme nous l'avons déjà indiqué, ces problèmes se révèlent proches de la problématique de notre étude. Pour cette raison, leur analyse est indispensable afin de mesurer la nécessité du développement de nouvelles méthodes, adaptées au contexte qui est le nôtre.

3.1 Problématique

3.1.1 Chaînes de fabrication

Comme tout système de fabrication, les chaînes de fabrication sont utilisées dans l'industrie en vue d'obtenir des produits finis à partir de produits bruts ou de matière première. À l'origine, de tels systèmes n'étaient utilisés que pour la production de masse, mais aujourd'hui elles s'adaptent même aux besoins de fabrication en petites séries et gagnent de nouveaux terrains industriels [Boysen *et al.*, 2006]. Une comparaison de l'efficacité des chaînes de production par rapport à d'autres types de système de production peut être trouvée dans [Dolgui et Proth, 2006].

Comme son nom l'indique, une chaîne est constituée d'un ensemble de postes de travail disposés de manière séquentielle, en sorte qu'il soit possible de lancer un produit brut au début de la chaîne (le premier poste de travail) pour récupérer un produit fini au bout de la chaîne (le dernier poste de travail). De cette façon, une unité de produit parcourt la chaîne dans une seule direction et visite séquentiellement tous les postes de travail installés.

L'ensemble des opérations réalisées par la chaîne sur une unité de produit constitue le travail nécessaire pour obtenir le produit fini à partir d'un brut. Lorsqu'une unité de produit arrive à un poste de travail, elle subit une suite d'opérations, effectuée par des ouvriers ou par des équipements associés à ce poste de travail. La durée totale des opérations sur un poste de travail est nommée *la charge* du poste de travail. Deux types de chaînes peuvent être distingués par rapport à l'organisation du transfert de la pièce : chaînes synchronisées et non synchronisées. Si la chaîne est synchronisée, alors toutes les pièces se trouvant sur la chaîne sont transportées simultanément. Le fonctionnement synchronisé de la chaîne se base sur le temps de cycle : il est impératif que la charge de chaque poste de travail ne le dépasse pas. Si cette condition est validée, alors il est possible, au bout d'un temps de cycle, de faire avancer les pièces vers le poste suivant.

Il est facile de constater que les chaînes d'assemblage ont plusieurs points en commun avec les systèmes d'usinage à boîtiers multibroches que nous étudions. Dans les deux types de systèmes, la fabrication se déroule sur un ensemble de postes de travail disposés de manière linéaire. Dans les deux cas, il faut organiser le processus de fabrication (répartir les opérations aux postes) de telle façon que le temps de cycle imposé soit respecté. Enfin, dans le cadre de la conception des deux types de systèmes, il est impératif d'optimiser la performance du système en minimisant le coût pour une productivité donnée. À cause de tous ces points communs, nous nous sommes intéressés aux méthodes proposées dans la littérature pour la conception des chaînes de fabrication et, plus particulièrement, des chaînes d'assemblage, en vue de comprendre en quoi les approches connues peuvent être utiles pour le développement de nouvelles méthodes adaptées au contexte d'usinage et où sont leurs limites.

La conception des chaînes d'assemblage est un problème de décision complexe. De nombreuses études dans des contextes différents ont été présentées dans la littérature, comme par exemple, [Chow, 1990; Dagnino, 1994; Nkasu et Leung, 1995; He et Kusiak, 1998; Dolgui et Proth, 2006; Rekiek *et al.*, 2006]. Dans la pratique, il est impossible de confier la conception d'une chaîne à une seule personne - plusieurs services se chargent de la prise de décision. Généralement, ce processus se rapproche conceptuellement de celui que nous avons proposé dans le chapitre précédent. Il faut dire que dans ce processus une étape nous intéresse plus particulièrement, il s'agit de l'étape de l'équilibrage de la chaîne.

Cette étape a comme objectif de répartir les *opérations*, qui sont nécessaires pour la fabrication du produit (ou des produits), à des *postes de travail* tout en respectant un ensemble de *contraintes*. Il y a des *critères* qui sont utilisés pour évaluer la « qualité » de chaque solution. Comportant impérativement ces quatre éléments de base (opérations, postes, contraintes, critères), les formulations des problèmes d'équilibrage diffèrent par le contenu de ces éléments et leurs représentation et modélisation mathématiques.

Nous commençons notre analyse par la présentation de la formulation de base qui est connue sous le nom du SALBP (*Simple Assembly Line Balancing Problem*).

3.1.2 Formulation du problème de base : SALBP

La nécessité d'organiser le processus de production de manière optimale pousse les industriels à recourir à des outils de la recherche opérationnelle et à formuler mathématiquement leurs problèmes de décision afin d'y trouver des réponses efficaces. L'apparition du problème

d'équilibrage n'échappe pas à la règle : à l'origine de sa première formulation mathématique se trouve la volonté de minimiser les pertes lors du fonctionnement des chaînes d'assemblage, engendrées par les temps morts des opérateurs y travaillant. Il s'agissait de l'équilibrage d'une chaîne d'assemblage fabriquant un seul type de produit. Les hypothèses suivantes ont été utilisées dans cette première formulation.

Opérations

- L'ensemble d'opérations $I = \{1, 2, ..., i, ..., |I|\}$ est connu et fixe.
- Chaque opération est caractérisée par une valeur déterministe t_i estimant sa durée.

Postes

- Les postes de travail $M = \{1, 2, ..., k, ..., m\}$ sont identiques en terme de configuration et n'importe quelle opération peut être affectée à n'importe quel poste de travail.
- Les opérations affectées à un poste de travail k, c'est-à-dire l'ensemble I_k, sont exécutées de manière séquentielle. La somme des durées de toutes ces opérations représente la charge du poste de travail $T_k = \sum_{i \in I_k} t_i$.

Contraintes

- Une opération doit être effectuée une seule fois et entièrement sur un poste de travail. Par conséquent, elle doit être affectée à un unique poste de travail parmi ceux installés sur la chaîne.
- Un seul type de produit est assemblé sur la chaîne, c'est-à-dire que pendant chaque cycle, le même ensemble d'opérations est effectué sur un poste de travail.
- La chaîne est synchronisée, c'est-à-dire que l'exécution d'opérations sur tout poste de travail de la chaîne est restreinte par un temps de cycle objectif $T_k \leq T_0 \, , \forall \, k \in M$.
- Un ordre partiel d'exécution d'opérations, imposé par la technologie d'assemblage utilisée, doit être respecté lors de la répartition des opérations. Les contraintes de ce type sont connues sous le nom de « contraintes de précédence ».

Fonction objectif

- L'objectif est de minimiser le nombre de postes de travail nécessaires pour affecter toutes les opérations en respectant l'ensemble de contraintes de précédence et la contrainte de temps de cycle, la valeur du temps de cycle à respecter T_0 étant fixée.

Le premier modèle mathématique pour ce problème a été publié par Salveson en 1955 [Salveson, 1955]. Plus tard, en 1986, cette formulation à été dénommée *Simple Assembly Line Balancing Problem* (SALBP) par Baybars [Baybars, 1986]. Comme cela a été démontré dans [Wee et Magazine, 1986], ce problème est NP-difficile de manière générale. Afin d'évaluer la difficulté d'un problème particulier de ce type, un certain nombre de paramètres ont été proposés, par exemple, dans [Bhattacharjee et Sahu, 1990; Scholl, 1999; Driscoll et Thilakawardana, 2001].

Hormis le modèle de Salveson, trois autres formulations du problème d'équilibrage des chaînes d'assemblage se trouvent dans la classe du SALBP : chacune comporte les hypothèses ci-dessus et une fonction objectif différente. Le Tableau 3.1 présente ces variantes.

TAB. 3.1 – Les variantes du SALBP

m, **nombre de postes de travail**	T_0, **temps de cycle**	
	Fixé	À minimiser
Fixé	SALBP-F	SALBP-2
À minimiser	SALBP-1	SALBP-E

- Le problème de type SALBP-F (le F désignant faisabilité) consiste à répondre à la question suivante : existe-elle une répartition de l'ensemble d'opérations I à m postes de travail satisfaisant toutes les contraintes de précédence entre les opérations et garantissant que la charge de tout poste ne dépasse pas le temps de cycle T_0.
- Le problème de type SALBP-1 a comme objectif de minimiser le nombre de postes de travail nécessaire pour affecter toutes les opérations en respectant l'ensemble de contraintes de précédence et la contrainte de temps de cycle ayant T_0 fixe.
- Quant au SALBP-2, il s'agit de minimiser le temps de cycle, en affectant les opérations à une séquence de postes de travail, dont le nombre est fixé.
- Le problème SALBP-E conjugue les objectifs de SALBP-1 et SALBP-2 et vise à maximiser l'efficacité de la chaîne par la minimisation à la fois du nombre de postes de travail et du temps de cycle, soit à minimiser mT_0.

Dans sa première publication portant sur la problématique de l'équilibrage des chaînes d'assemblage, Salveson avait prédit un avenir conséquent pour ce type de problème d'optimisation. En effet, l'essor de l'industrie et des systèmes d'assemblage entraîna la popularité de ce problème dans le milieu académique et, par conséquent, plusieurs modèles mathématiques et méthodes de résolution lui furent consacrés. L'ensemble abondant des outils développés a été analysé à maintes reprises, nous mentionnons, dans leur ordre chronologique, les études bibliographiques les plus connues : [Ignall, 1965; Mastor, 1970; Buxey *et al.*, 1973; Baybars, 1986; Ghosh et Gagnon, 1989; Erel et Sarin, 1998; Talbot *et al.*, 1986; Ponnambalam *et al.*, 1999; Scholl, 1999; Rekiek *et al.*, 2002*a*; Scholl et Becker, 2006]. Parmi ces analyses bibliographiques, il y a quelques unes entièrement consacrées au SALBP : l'état de l'art des méthodes exactes [Baybars, 1986], des comparaisons des performances des méthodes approchées [Talbot *et al.*, 1986; Boctor, 1995; Ponnambalam *et al.*, 1999], sans oublier la plus récente publication [Scholl et Becker, 2006], qui donne un large aperçu sur les méthodes de résolution dédiées au SALBP, mais aussi sur les algorithmes de calcul des bornes inférieures, les méthodes de pré-traitement et les règles de dominance.

3.1.3 Généralisation du problème de base : GALBP

La formulation du SALBP n'exprime pas toute la réalité industrielle à cause des hypothèses simplificatrices sur lesquelles elle se base [Ghosh et Gagnon, 1989]. Pour faire face

à cette situation, plusieurs chercheurs ont essayé d'introduire des hypothèses plus générales. N'importe quel problème de type ALBP ayant au moins une hypothèse étendue par rapport à la formulation de base (SALBP) a été nommé dans la littérature *Generalized Assembly Line Balancing Problem* (GALBP).

Les articles présentant l'état de l'art sur GALBP [Ghosh et Gagnon, 1989; Rekiek *et al.*, 2002*a*; Becker et Scholl, 2006] permettent d'observer l'évolution des hypothèses et des approches de résolution proposées. Dans [Ghosh et Gagnon, 1989], les auteurs ont comparé 64 travaux concernant les problèmes de type SALBP et de GALBP (publiés pendant la période 1955-1985). Alors que dans [Becker et Scholl, 2006], les auteurs ont recensé 146 travaux traitant seulement les problèmes de type GALBP (publiés pendant la période 1965-2006).

Évidement, on peut toujours trouver des formulations « plus générales » que d'autres, cela rend la définition « non SALBP = GALBP » inefficace. Il est indispensable d'introduire une classification explicite des formulations de problèmes d'équilibrage. D'ailleurs, une première version d'une telle classification a été proposée dans [Boysen *et al.*, 2006]. Toutefois, la classification suggérée ne permet pas de voir clairement pour quels types de problème on dispose de méthodes efficaces. Bien que les aspects industriels soient différents, il est parfois possible de les décrire en recourant aux mêmes modèles mathématiques et, par conséquent, de surmonter les problèmes d'optimisation correspondants par le biais des mêmes méthodes de résolution. Nous proposons une autre version de classification, basée sur l'analyse des quatre éléments de base de chaque ALBP (modélisation des opérations, des postes de travail, des contraintes et du critère), en vue de comparer les formulations déjà traitées dans la littérature avec celles qui font l'objet de notre étude.

3.2 Classification des modèles pour les problèmes de type ALBP

3.2.1 Modélisation des opérations

Chaque opération est caractérisée par au moins un paramètre qui représente sa durée, mais d'autres paramètres peuvent lui être associés comme, par exemple, son coût. Chaque paramètre peut être défini d'une des manières suivantes :

1. $p_j(i) = const$

   La valeur du paramètre j est une constante, c'est-à-dire connue et invariable pour chaque opération i. Par exemple, l'hypothèse que les temps opératoires sont constants est une hypothèse de base du SALBP. Les valeurs d'un paramètre peuvent appartenir :
   - à l'ensemble de nombres réels $p_j(i) \in \mathbb{R}$, par exemple, le temps [Salveson, 1955], le coût d'exécution [Amen, 2000*a*; Bukchin et Rabinowitch, 2006],l'espace nécessaire [Lee et Johnson, 1991; Kim et Park, 1995; Sawik, 2002], etc.
   - à l'ensemble de nombres entiers $p_j(i) \in \mathbb{N}$, par exemple, le temps [Sarker et Shanthikumar, 1983; Wilhelm, 1999], etc.
   - ou à un ensemble fini $p_j(i) \in \mathcal{A}$, par exemple, le côté de la pièce pour l'exécution de l'opération [Buxey, 1974; Lapierre et Ruiz, 2004], etc.

2. $p_j(i) = f(\mu, \sigma^2)$
   La valeur du paramètre j est une variable aléatoire, dont la moyenne et la variance (μ, σ^2) sont connus pour chaque opération i. Les formulations avec des temps opératoires stochastiques ont été étudiées, par exemple, dans [Moodie et Young, 1965; Kao, 1976; Kao, 1979; Sniedovich, 1981; Raouf et Tsui, 1982; Carraway, 1989; Suresh et Sahu, 1994; Sarin *et al.*, 1999; Baykasoglu et Özbakir, 2006; Urban et Chiang, 2006].
3. $p_j(i) = \tilde{\mathrm{p}}$
   La valeur du paramètre j est une variable floue [Tsujimura *et al.*, 1995; Hop, 2006].
4. $p_j(i) = f(t)$
   La valeur du paramètre j change en fonction du temps t de fonctionnement de la ligne. De tels paramètres sont donnés par leurs valeurs initiales et les fonctions décrivant leur évolution dans le temps. Par exemple, les temps opératoires ont tendance à diminuer en fonction du temps de fonctionnement de la ligne d'assemblage manuelle grâce à l'effet d'apprentissage subi par les opérateurs [Boucher, 1987; Chakravarty, 1988; Cohen et Dar-El, 1998]. Plusieurs fonctions ont été proposées pour modéliser l'effet d'apprentissage [Womer, 1979; Yelle, 1979].
5. $p_j(i) = f(I_k)$
   La valeur du paramètre j, associée à l'opération i, dépend de l'ensemble des opérations qui ont déjà été exécutées ou sont affectées à la même station. Par exemple, le temps d'exécution d'une opération peut dépendre des opérations effectuées auparavant [Chen *et al.*, 2002; Capacho et Pastor, 2005; Capacho et Pastor, 2006; Capacho *et al.*, 2006*b*; Capacho *et al.*, 2006*a*; Scholl *et al.*, 2007] ; le temps opératoire nécessaire pour exécuter l'opération i (et/ou le coût d'exécution de l'opération i) peut dépendre de l'opération h qui la suit, dans ce cas, $p_j(i) = f(h)$ [Graves et Lamar, 1983; Wilhelm, 1999].
6. $p_j(i) = f(A_k)$
   La valeur du paramètre j varie en fonction des paramètres du poste où l'opération est affectée. Par exemple, le temps opératoire peut dépendre du type d'équipement installé sur le poste de travail correspondant [Graves et Redfield, 1988; Bukchin et Tzur, 2000; Gadidov et Wilhelm, 2000; Levitin *et al.*, 2006], le coût d'une opération peut dépendre du poste de travail où elle est exécutée [Wei *et al.*, 1997] ou du type d'équipement installé sur ce poste de travail [Graves et Redfield, 1988]. Une autre hypothèse, considérée par [Shtub, 1984; Wilson, 1986], admet que les temps opératoires sont des fonctions linéaires non croissantes de la taille de l'équipe assignée au poste de travail où l'opération est affectée.

Le paramètre incontournable est le temps opératoire, considéré par presque tous les auteurs. Ceci découle du fait que la contrainte sur le temps de cycle est une des caractéristiques principales des chaînes de fabrication. Les modèles mathématiques où les opérations ont plusieurs paramètres sont peu nombreux, encore moins nombreux sont ceux avec des paramètres représentés de façons différentes. Il est à noter que souvent les approches proposées permettent d'intégrer facilement des paramètres complémentaires dans la formulation du problème traité ou de changer la modélisation d'un paramètre. Pourtant peu d'auteurs ont étudié des adaptations éventuelles des algorithmes développés suite à de telles modifica-

tions. Nous pouvons tout de même citer quelques travaux dans cette direction : [Sphicas et Silverman, 1976; Johnson, 1983; Kim *et al.*, 2000; Boysen et Fliedner, 2006].

3.2.2 Modélisation des postes de travail

Structure d'un poste de travail

Un poste de travail est une ressource de la ligne à laquelle des opérations doivent être affectées. Mais il peut être nécessaire de lui associer d'autres éléments, comme, par exemple, des opérateurs ou des équipements. Nous proposons de modéliser ces éléments à l'aide des *attributs* des postes de travail. Nous désignons par $A(k)$ l'ensemble des attributs qui déterminent la configuration du poste de travail k. L'existence d'attributs permet de différencier les postes de travail qui ne sont plus identiques comme cela a été admis pour le SALBP. Nous distinguons les *attributs scalaires* $A^s(k)$ et les *attributs vectoriels* $A^v(k)$.

Les attributs scalaires caractérisent un poste de travail comme un objet entier. Par exemple, le nombre d'équipements à installer en parallèle sur un poste de travail est un attribut scalaire si toutes les unités d'équipement effectuent le même ensemble d'opérations (qui a été affecté au poste de travail). En revanche, un attribut vectoriel est utilisé quand une répartition des opérations entre plusieurs éléments de même genre et du même poste de travail est nécessaire. Par exemple, la répartition des opérations entre les boîtiers multi-broches installés sur un poste de travail peut être représentée par un attribut vectoriel a_j^v. Dans ce cas, chaque élément h de l'attribut vectoriel a_j^v du poste de travail k est caractérisé par son propre ensemble d'opérations $I(a_{jh}^v(k))$ qui est un sous-ensemble de l'ensemble I_k.

La nécessite de déterminer les attributs des postes de travail à l'étape de la structuration de la ligne amène à résoudre simultanément les deux problèmes d'optimisation suivants : le problème d'équilibrage et de structuration de la ligne. Il est à noter que l'existence des attributs vectoriels augmente la complexité du problème à résoudre, car il convient de répartir des opérations « à l'intérieur » de chaque poste de travail.

Selon les attributs pris en compte, les problèmes d'équilibrage se déclinent en trois catégories :

1. $A(k) = \emptyset$

   Tous les postes de travail sont identiques et les opérations représentent le seul élément qui leur est affecté.

2. $A^s(k) \neq \emptyset$

   Hormis l'affectation d'opérations au poste de travail, il convient de choisir la configuration de chaque poste de travail en déterminant les valeurs des attributs scalaires de ce poste (voir la section suivante).

   Exemples : décider si la duplication du poste travail en question est nécessaire [Pinto *et al.*, 1981], choisir un type d'équipement à installer sur ce poste de travail [Bukchin et Tzur, 2000; Gadidov et Wilhelm, 2000], déterminer le nombre de machines identiques à mettre en parallèle [Bukchin et Rubinovitz, 2002] ou la taille de l'équipe d'opérateurs qui travaillera sur ce poste [Shtub, 1984; Wilson, 1986; Johnson, 1991].

3. $A^v(k) \neq \emptyset$

   La configuration de chaque poste de travail comporte un ensemble d'éléments dont chacun possède un sous-ensemble des opérations qui lui sont associées. La répartition des opérations est représentée par un attribut vectoriel et nécessite d'être déterminée.

   Exemples : concevoir des lignes où il est possible de mettre des équipements des deux côtés du convoyeur [Bartholdi, 1993; Lee *et al.*, 2001], affecter plusieurs opérateurs aux mêmes postes de travail en répartissant la charge entre eux [Dimitriadis, 2006].

   En théorie, chaque élément d'un attribut vectoriel peut à son tour être composé d'un ensemble de sous-éléments et ainsi de suite, mais vu la complexité de tels problèmes, ils n'ont pas encore été formulés dans la littérature.

Attributs scalaires

Les attributs scalaires sont utilisés généralement pour déterminer la configuration de postes de travail. Le choix d'une valeur pour l'attribut scalaire j peut être effectué en fonction des opérations affectées au poste k, c'est-à-dire $a_j^s(k) = f(I(k))$, ou en tenant compte d'autres attributs de ce poste de travail $a_j^s(k) = f(A(k))$ ou des attributs d'autres postes de travail $a_j^s(k) = f(A(M))$. Il est également possible de combiner ces trois fonctions.

1. Par rapport aux opérations affectées à un poste, les valeurs des attributs de celui-ci peuvent être :
   - déterminées par les paramètres des opérations (souvent de type $p_h(i) \in \mathcal{A}$) qui sont affectées au poste. Dans ce cas, l'affectation de la première opération détermine la valeur de l'attribut. Il en découle que les valeurs de l'attribut $a_j^s(k)$ appartiennent au même ensemble fini $\mathcal{A}$ que les valeurs du paramètre $p_h(i)$. Par exemple, s'il existe des opérations qui ne se font pas à la même hauteur, certaines en haut, d'autres en bas. Alors, une fois une opération affectée au poste de travail, la hauteur de ce poste doit correspondre à la hauteur de cette opération. Donc, toutes les opérations se faisant à une hauteur différente ne pourront pas être affectées à ce poste pour éviter que l'opérateur ne descende et monte tout le temps.
   - calculées à partir des paramètres des opérations affectées au poste de travail. Prenons l'exemple d'installation d'équipements en parallèle sur un poste de travail. Cette configuration permet notamment d'exécuter des opérations dont le temps opératoire dépasse le temps de cycle. Plusieurs règles pour le choix du nombre d'équipements à dupliquer ont été proposées dans la littérature. Nous présentons ici la plus simple. Soit w_k le nombre d'équipements parallèles du poste de travail, alors la valeur de w_k peut être calculé comme suit :

     $$w_k = \left\lceil \frac{T(k)}{T_0} \right\rceil$$

   - calculées à partir des paramètres des opérations et à leur tour ayant un impact sur les valeurs des paramètres des opérations de type $p(i) = f(A^s(k))$. Par exemple, si les temps opératoires sont des fonctions linéaires non croissantes de la taille de l'équipe assignée au poste de travail où l'opération est affectée [Shtub, 1984; Wilson, 1986].

2. Par rapport aux autres attributs du même poste, ils peuvent être :
   - indépendants, s'il n'existe pas de lien entre les attributs ;
   - dépendants, s'il faut tenir compte d'impossibles combinaisons des attributs au même poste de travail, par exemple, s'il existe des contraintes d'incompatibilité entre les équipements disponibles [Rekiek *et al.*, 2002*b*; Dolgui *et al.*, 2006*e*; Belmokhtar *et al.*, 2006*b*].
3. Par rapport aux autres postes de la chaîne, les attributs peuvent être :
   - indépendants : par exemple, s'il est considéré qu'il y a un ensemble de types d'équipements et le nombre d'équipements disponibles de chaque type est infini [Graves et Redfield, 1988; Bukchin et Tzur, 2000; Gadidov et Wilhelm, 2000; Bukchin et Rubinovitz, 2002] ;
   - dépendants, si la configuration d'un poste de travail dépend des configurations d'autres postes de la même ligne. Par exemple, s'il y a un ensemble d'équipements de taille finie, alors si un équipement est affecté à un poste de travail, il sera impossible de l'affecter à un autre poste de travail [Aĝpak et Gökçen, 2005; Belmokhtar *et al.*, 2006*b*]. La même hypothèse est valable pour l'affectation des opérateurs ayant différents niveaux, si leur nombre est fixe [Chan *et al.*, 1998]. Un autre facteur faisant que la configuration d'un poste dépend des postes voisins est l'existence de zones de travail : tous les postes appartenant à la même zone de travail ont forcément des attributs communs, comme, par exemple, la hauteur du convoyeur dans [Lapierre et Ruiz, 2004].

Attributs vectoriels

Les attributs vectoriels sont utilisés pour décrire les relations entre les ressources installées sur un poste de travail et les opérations qui lui sont affectées. Comme nous l'avons déjà évoqué, l'introduction d'attributs vectoriels augmente la complexité du problème à résoudre. Pour cette raison, peut être, les formulations les utilisant ne sont pas nombreuses.

Un attribut vectoriel représente la configuration de poste de travail qui comporte un ensemble d'éléments dont chacun possède un sous-ensemble des opérations qui lui sont associées. Il est généralement nécessaire : premièrement, de déterminer l'ensemble de tels éléments associés au poste de travail (dont la cardinalité nous désignons par $\dim a_j^v$) ; deuxièmement, de répartir les opérations affectées au poste de travail entre ces éléments. Alors, cette répartition peut être :

- **immédiate**, si parmi les éléments de l'attribut vectoriel, il n'y a en qu'un seul qui convient à l'affectation de chaque opération. Par exemple, dans [Kim et Park, 1995], les auteurs considèrent le cas de la conception des lignes robotiques, où plusieurs types d'outils peuvent être installés sur chaque poste de travail, chaque opération pouvant être exécutée par un seul type d'outil.
- **à optimiser**, s'il existe plusieurs possibilités d'affectation de chaque opération à l'intérieur de chaque poste de travail et le choix de son affectation n'est pas évident. Prenons l'exemple des lignes où il est possible de mettre des opérateurs ou des équipements des deux côtés du convoyeur [Bartholdi, 1993; Lee *et al.*, 2001]. Dans ce cas, l'affectation des opérations qui peuvent être exécutées de côté gauche comme de côté droit n'est pas

certaine et doit être optimisée. De plus, lorsque la repartition des opérations n'est pas évidente, il peut être nécessaire de tenir compte d'impossibles combinaisons d'affectation, par exemple, s'il existe des contraintes d'incompatibilité entre les équipements disponibles [Rekiek *et al.*, 2002*b*; Dolgui *et al.*, 2006*e*; Belmokhtar *et al.*, 2006*b*].

Caractéristiques calculées

Les attributs de postes de travail décrivent leur configuration et la détermination de valeurs de ces attributs, généralement, fait partie de la solution. Pour évaluer les solutions possibles, les différentes caractéristiques de postes de travail sont également calculées, notamment en vue de vérifier certaines contraintes ou de calculer la valeur de la fonction objectif utilisée. Nous désignons l'ensemble des caractéristiques du poste de travail k par $C(k)$. Il est à noter que les valeurs de certaines caractéristiques peuvent être calculées en tenant compte de la répartition d'opérations aux éléments constituant des attributs vectoriels.

Les fonctions de calcul des valeurs des caractéristiques de postes de travail peuvent être très diverses, les plus simples ont les formes suivantes :

- $c_j(k) = \sum_{i \in I_k} p_j(i)$

  Exemples : la charge du poste de travail k dans [Salveson, 1955] ; l'espace nécessaire pour l'exécution des opérations sur le poste de travail k dans [Sawik, 2002].

- $c_j(k) = max\{p_j(i), i \in I_k\}$

  Exemples : le coût d'exécution d'opérations sur le poste de travail k dans [Amen, 2000*a*; Amen, 2000*b*; Amen, 2001; Amen, 2006] ; le temps d'exécution de l'ensemble d'opérations affecté à un boîtier multibroche [Finel, 2004].

- $c_j(k) = a_j^s(k) \cdot b_h$, où b_h est une constante

  Exemples : le coût salarial des opérateurs affectés au poste k, dans ce cas $a_j^s(k)$ est le nombre d'opérateurs travaillant sur le poste de travail k et b_h représente le montant du salaire d'un opérateur [Sarker et Shanthikumar, 1983; Bard, 1989; McMullen et Frazier, 1998] ; le coût des boîtiers multibroches installés sur un poste de travail [Finel *et al.*, 2006; Dolgui *et al.*, 2006*a*; Guschinskaya *et al.*, 2008*a*].

- $c_j(k) = \sum_{h=1}^{\dim a_j^v} a_{jh}^v(k) \cdot b_{a_{jh}^v(k)}$,
  où $b_{a_{jh}^v(k)}$ est une constante associée à l'élément $a_{jh}^v(k)$.

  Exemples : le coût des équipements installés sur un poste de travail, dans ce cas $a_{jh}^v(k)$ donne le type d'équipement et $b_{a_{jh}^v(k)}$ le coût correspondant [Rekiek, 2001; Belmokhtar *et al.*, 2006*b*].

Nous rapportons également quelques caractéristiques calculées dans le cadre d'installation de ressources identiques et parallèles sur un poste de travail lorsque la charge d'un poste de travail T_k dépasse le temps de cycle T_0.

Puisque l'attribut w_k, déjà introduit, ne peut être qu'un nombre entier, la caractéristique

suivante est ensuite calculée :

$$u_k = \frac{T_k}{T_0 \cdot w_k}$$

u_k mesure le taux d'utilisation des équipements ou des opérateurs installés en parallèle sur le poste k.

De plus, si les temps opératoires sont représentés par des variables aléatoires suivant la loi de Fauss, alors pr_k, la probabilité que la charge du poste de travail k ne dépassera pas le temps de cycle T_0 est calculée comme suit :

$$pr_k = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{Y} e^{-\frac{Z^2}{2}} dZ, \text{ où } Y = \frac{T_0 \cdot w_k - T_k}{\sqrt{\sum_{i \in I(k)} \sigma_i^2}}$$

La caractéristique la plus calculée est, sans doute, la charge de postes de travail, qui est utilisée pour vérifier la contrainte de temps de cycle. Nous rappelons que la charge d'un poste de travail est définie comme le temps durant lequel une unité de pièce séjourne sur ce poste de travail. Il existe différents modèles pour la calculer (représentant des réalités différentes dans les chaînes de production), dans ce qui suit, nous en présentons quelques uns les plus utilisés.

Calcul de la charge des postes de travail

Dès que l'ensemble d'opérations affectées à un poste de travail est connu, il est possible d'estimer le temps nécessaire pour exécuter cet ensemble en vue de vérifier la contrainte de temps de cycle. À ce niveau, il existe trois principales hypothèses :

1. Les opérations s'exécutent séquentiellement
   - et l'ordre d'exécution d'opérations n'a aucune importance pour le calcul de la charge des postes de travail (l'hypothèse de la formulation de base). Dans ce cas, $T_k = \sum_{i \in I_k} t_i$.

     Ce modèle est utilisé lorsque tous les temps auxiliaires sont soit négligeables, soit déjà inclus dans les temps opératoires. Parfois, un incrément de temps est également associé à chaque poste de travail [Boysen *et al.*, 2006]. Il peut représenter, par exemple, le temps de chargement/déchargement de la pièce.
   - et le temps d'exécution des opérations dépend de la séquence dans laquelle elles se font, et ce pour plusieurs raisons :
     – les temps de déplacement d'outil, de changement d'outil et de rotation de la pièce ne sont pas négligeables [Arcus, 1966; Graves et Redfield, 1988; Wilhelm, 1999; Bautista et Pereira, 2002; Guschinskaya et Dolgui, 2007*d*]. Dans ce cas, $T_k = \sum_{i \in I_k} (t_i + \Delta t_{ij})$, où $j \in I(k)$ est une opération qui est exécutée juste après l'exécution de l'opération i sur le même poste de travail.
     – l'ordre d'exécution des opérations a un impact sur leurs durées. Par exemple, l'exécution de l'opération i avant l'opération j peut rendre l'exécution de l'opération j plus longue, car l'endroit de l'exécution devient plus difficilement accessible

[Capacho *et al.*, 2006*b*; Capacho *et al.*, 2006*a*; Capacho *et al.*, 2007; Scholl *et al.*, 2007]. Dans ce cas $T_k = \sum f_i(I_k)$.

- dans le cas de lignes où il est possible de mettre des opérateurs ou des équipements des deux côtés du convoyeur, il peut avoir des délais entre l'exécution de deux opérations qui sont affectées à deux côtés différents. Si l'une précède directement l'autre, mais le début de la deuxième opération a été prévu avant la fin de la première, dans ce cas, la deuxième opération ne peut pas être entamée avant la fin de la première et un temps mort apparaît [Kim *et al.*, 2000; Lee *et al.*, 2001].

2. Les opérations s'exécutent en parallèle, voir par exemple [Belmokhtar *et al.*, 2006*b*; Dolgui *et al.*, 2006*e*; Dolgui *et al.*, 2007*b*].
3. Le mode mixte est utilisé pour exécuter les opérations affectées à un poste de travail, voir par exemple [Belmokhtar *et al.*, 2007; Guschinskaya *et al.*, 2007*c*].

3.2.3 Contraintes

Les contraintes sont utilisées pour définir l'ensemble de solutions admissibles. Les origines des contraintes peuvent être de nature technologique, économique ou logique. Dans la littérature, nous trouvons les types suivants de contraintes.

Contraintes d'affectation

- Toutes les opérations de l'ensemble I doivent être affectées pour être exécutées : soit pendant un temps de cycle (cas mono-produit), soit pendant plusieurs temps de cycle (cas multi-produit).
- Les **contraintes de précédence** entre les opérations doivent être respectées (ce sont les contraintes les plus utilisées) [Salveson, 1955; Baybars, 1986].
- Les **contraintes d'inclusion (inc)** sont introduites pour imposer que certains sous-ensembles d'opérations soient réalisés sur le même poste de travail [Scholl, 1999]. Il est à noter que dans le cas de la conception d'une ligne fabriquant un seul produit et ayant des postes identiques, les opérations liées par ce type de contraintes peuvent être unies en des macro-opérations [Buxey, 1974; Deckro, 1989], puisque l'affectation d'une opération définit explicitement l'affectation des autres.
- Les **contraintes d'exclusion (exc)** sont utilisées pour interdire l'exécution de certains sous-ensembles d'opérations sur le même poste de travail [Buxey, 1974; Raouf et Tsui, 1982] ou par le même type d'équipement [Dolgui *et al.*, 2006*e*; Dolgui *et al.*, 2008; Guschinskaya *et al.*, 2008*a*; Guschinskaya et Dolgui, 2007*b*]. Ce type de contraintes permet d'éviter l'affectation au même poste soit des opérations nécessitant des équipements aux conditions d'exploitation incompatibles, soit tellement différentes qu'une telle affectation conduit à un coût important.
- Les **contraintes sur les zones de travail** imposant des distances entre les opérations (minimum ou maximum). Ces distances sont définies soit en terme de nombre de stations, soit en terme du temps entre la fin d'une opération et le début de l'autre [Park *et al.*, 1996; Pastor et Corominas, 2000].

Contraintes sur les attributs de postes

- les contraintes entre les attributs scalaires d'un poste de travail et les paramètres des opérations qui lui sont affectées (nous notons ce type de contraintes comme $a_j^s(k) \sim p_j(i)$). L'introduction de ce type de contraintes peut être une autre façon d'exprimer l'incompatibilité entre les opérations. Dans ce cas, l'opération i peut être affectée au poste de travail k si et seulement si elle a la même valeur du paramètre $p_j(i)$ que l'attribut scalaire $a_j^s(k)$ associé au poste k [Johnson, 1983] ;
- les contraintes entre les différents attributs d'un poste de travail ;
- les contraintes entre les attributs de différents postes de travail, par exemple, $\sum_{k=1}^{m} a_j^s(k) \leq b$ ou $\sum_{k=1}^{m} \dim a_j^v(k) \leq b$ si par exemple, le nombre total d'ouvriers ou d'équipements à affecter est limité [Aĝpak et Gökçen, 2005].

Contraintes sur les caractéristiques de postes

Ces contraintes peuvent être exprimées des façons suivantes :

- $c_j(k) \leq b$

  Exemples : la charge du poste de travail k ne doit pas dépasser le temps de cycle $b = T_0$; l'espace nécessaire pour exécuter les opérations affectées au poste k doit être inférieur à l'espace disponible b ;

- $Pr[c_j(k) \leq b] \geq \alpha$

  Exemples : la probabilité que la somme des temps opératoires ne dépasse pas le temps de cycle fixe $b = T_0$ doit être supérieur ou égal à α [Kao, 1976; Kao, 1979; Sniedovich, 1981; Carraway, 1989; Baykasoglu et Özbakir, 2006] ;

- $c_j(k) \leq a_h^s(k)$ par exemple : s'il faut admettre un certain déséquilibrage de la charge des postes de travail, alors la valeur $a_h^s(k)$ représente le seuil du temps alloué au poste de travail k [Johnson, 1983; Erel et Gökcen, 1999].

3.2.4 Fonction objectif

Comme nous l'avons déjà évoqué, la fonction objectif est utilisée afin de distinguer la(les) meilleure(s) solution(s) parmi les solutions admissibles d'un problème d'optimisation.

Fonctions objectif de base

Nous avons déjà introduit trois fonctions objectif suivantes :

- m, le nombre de postes de travail à installer doit être minimisé (le plus souvent utilisée, voir par exemple [Scholl, 1999]) ;
- T_0, le temps de cycle de la ligne est à minimiser [Rubinovitz et Levitin, 1995; Park *et al.*, 1996; Sawik, 2002; Levitin *et al.*, 2006] ;

- mT_0, le temps de séjour de la pièce dans la ligne doit être minimisé [Macaskill, 1972; Chakravarty, 1988; Scholl et Klein, 1999*b*].

Certains auteurs ont étudié l'impact de l'utilisation de ces trois fonctions objectif sur la qualité des solutions, voir, par exemple, [Nkasu et Leung, 1995; Bukchin, 1998].

Fonctions objectif utilisant les caractéristiques de postes de travail

- Le critère qui est souvent utilisé afin d'évaluer la qualité « d'équilibrage » de la ligne concerne l'égalité des charges des postes de travail, c'est-à-dire « smoothness » (le terme anglo-saxon). Un certain nombre de caractéristiques peuvent être utilisées afin d'évaluer la qualité « d'équilibrage », par exemple, l'écart moyen absolu SSL_{abs} ou l'écart type SSL_{ecrt}. Ces caractéristiques sont à minimiser. Si $\tilde{T} = \dfrac{1}{m}\sum_{i\in I} t_i$ est la charge « optimale » des postes de travail de la ligne (parfois $\tilde{T} = T_0$), alors ces caractéristiques peuvent être calculées comme suit :

$$SSL_{abs} = \frac{1}{m}\sum_{k=1}^{m}|T_k - \tilde{T}|, \quad SSL_{ecrt} = \sqrt{\sum_{k=1}^{m}(T_k - \tilde{T})^2}$$

  la dernière caractéristique est parfois évaluée, par exemple, [Rekiek, 2001] par :

$$SSL_{ecrt2} = \sum_{k=1}^{m}(T_k - \tilde{T})^2$$

  Dans la littérature, deux types de smoothness sont parfois distingués, à savoir : « vertical » et « horizontal » [Becker et Scholl, 2006; Boysen *et al.*, 2006]. Dans le premier cas, il s'agit d'équilibrer les charges de différents postes de travail, c'est-à-dire l'écart « vertical » entre la charge moyenne et la charge de chaque poste [Moodie et Young, 1965; Rachamadugu et Talbot, 1991; Kim *et al.*, 1998; Pastor et Corominas, 2000; Pinnoi et Wilhelm, 1997]. Par contre, pour les lignes fabriquant des produits différents, la charge du même poste de travail peut varier d'un cycle à l'autre, c'est-à-dire d'un produit à l'autre, il convient alors de minimiser la déviation « horizontale » entre les charges du même poste de travail. Une telle fonction objectif a été utilisée, par exemple, dans [Agnetis *et al.*, 1995]. Ces deux objectifs de l'équilibrage « vertical » et « horizontal » peuvent être considérés ensemble comme dans [Miltenburg, 1998; Merengo *et al.*, 1999; Matanachai et Yano, 2001; Pastor *et al.*, 2002; Vilarinho et Simaria, 2002].
- Un critère de « work relatedness » est utilisé s'il est souhaitable que les opérations relatives aux mêmes fonctions du produit soient affectées au même poste de travail [Agrawal, 1985]. Pour le mesurer, le paramètre « index of work relatedness » (IWR) est introduit, dont la valeur est calculée comme suit et doit être maximiser :

$$IWR = \frac{m}{\sum_{k=1}^{m} SN_k}$$

où SN_k est le nombre de sous-ensembles d'opérations affectés au poste de travail k tels qu'il n'existe aucun lien de précédence entre les opérations appartenant à des sous-ensembles différents.

- La fonction objectif suivante, dont la valeur est à maximiser, est souvent utilisée lorsque les temps opératoires sont représentés par des variables aléatoires :

$$Pr = \prod_{k=1}^{m} Pr[c_j(k) \leq T_0]$$

Il s'agit de maximiser la probabilité Pr que la charge de tous les postes de travail ne dépassera pas le temps de cycle T_0, sachant que $Pr[c_j(k) \leq T_0]$ représente la probabilité que la charge du poste de travail k ne dépassera pas le temps de cycle T_0 [McMullen et Frazier, 1997; Bukchin *et al.*, 2002].

- Il existe une classe de fonctions objectif (à maximiser ou à minimiser) qui sont basées sur les coûts liés aux attributs de postes de travail et/ou des coûts engendrés par les opérations affectées aux postes de travail. Ces derniers sont généralement exprimés par les caractéristiques de postes de travail. Ces fonctions peuvent utiliser des opérateurs de somme, de multiplication, de division, et elles ne sont pas toujours linéaires. On peut présenter la forme générale de telles fonctions objectif comme suit :

$$\sum_{k=1}^{m} f_1(A(k)) + \sum_{k=1}^{m} f_2(C(k))$$

où le premier terme représente la part engendrée par les attributs $A(k)$ et le deuxième donne la part relative aux caractéristiques des postes $C(k)$. Dans la suite du développement, nous dénotons les fonctions de ce type par $\sum A(k) + \sum C(k)$.

Les fonctions de ce type sont souvent utilisées lorsqu'on vise à optimiser à la fois les coûts liés à l'installation et à l'utilisation de la ligne. Cette classe de problèmes est connue dans la littérature anglo-saxonne sous le nom *Cost oriented Assembly Line Balancing Problem* (CALBP). Un état de l'art des travaux consacrés au CALBP et une analyse des coûts possibles ont été présentés dans [Amen, 2000*a*; Amen, 2000*b*; Amen, 2006]. Les coûts peuvent être engendrés par l'investissement (liés aux achats des équipements, à l'installation des stations de travail, etc.), ou par son fonctionnement (salaires des opérateurs). Ces coûts peuvent être intégrés dans la fonction objectif soit par le biais des paramètres associés aux postes de travail, comme le coût d'équipement qui ne dépend pas des opérations effectuées, soit par les caractéristiques calculées de postes de travail qui permettent d'estimer les coûts liés à l'exécution des opérations et/ou des salaires des opérateurs.

Problèmes d'optimisation multiobjectif

Un problème d'optimisation peut comporter non pas une seule, mais un ensemble de fonctions objectif. Dans ce cas, il s'agit d'un problème d'optimisation multiobjectif. Il existe plusieurs façons de résoudre un tel problème. Nous ne citerons que celles qui ont été appliquées pour la résolution des problèmes de configuration des lignes de fabrication, à savoir :

- **Agrégation des objectifs** : dans ce cas, les différents objectifs sont agrégés en un seul. Pour cela, il faut que tous les objectifs représentent des grandeurs d'unités comparables. De plus, il est nécessaire de déterminer des poids respectifs pour chacun de ces objectifs. En faisant la somme pondérée des différents objectifs, il est possible d'obtenir un problème d'optimisation mono-objectif. Cette façon de résoudre les problèmes multiobjectif a été utilisée, par exemple, dans [McMullen et Tarasewich, 2003].
- **Résolution lexicographique** : cette méthode consiste à considérer un ordre de priorité (dit lexicographique) entre les objectifs. Cette technique pour résoudre les problèmes multiobjectif a été utilisée, par exemple, dans [Lee *et al.*, 2001; Vilarinho et Simaria, 2002].
- **Goal programming** : cette méthode revient à s'approcher au maximum de valeurs cibles (aussi appelées niveaux d'aspiration) sur chacun des objectifs. Pour cela, des poids sont affectés à chaque objectif, et la somme pondérée des écarts (en excès comme en défaut) par rapport aux valeurs cibles doit être minimisée. Cette méthode a été appliquée, par exemple, dans [Gökçen et Agpak, 2006].
- **Méthode de surclassement** : cette méthode est utilisée pour comparer un ensemble de solutions. Par exemple, dans [Rekiek, 2001; Rekiek *et al.*, 2006] la méthode PROMETHEE II [Brans et Marechal, 1994] a été appliquée pour comparer les individus d'une génération créée par un algorithme génétique. Le principe de la méthode PROMETHEE II consiste à établir un processus de comparaison numérique de chaque solution par rapport à toutes les autres solutions. Ainsi il est possible de calculer le plus (mérite) ou le moins (démérite) de chaque solution par rapport à toutes les autres. Le résultat de cette comparaison permet un classement ordonné des solutions.

3.3 Résolution des problèmes d'équilibrage

Deux classes de méthodes d'optimisation combinatoire sont utilisées dans la littérature pour la résolution des problèmes de structuration des lignes de fabrication, à savoir : les méthodes exactes et les méthodes approchées (ou heuristiques). Comme leurs noms l'indiquent, les méthodes de la première classe visent à trouver « exactement » la meilleure solution. Malheureusement, la qualité des solutions qu'elles fournissent est souvent payée par un temps de résolution très important. Les méthodes approchées se contentent de la recherche de solutions de bonne qualité et elles ne garantissent pas l'optimalité de la meilleure solution trouvée. Elles sont utilisées lorsque le temps nécessaire pour la résolution exacte n'est pas compatible avec le contexte de la prise de décision. En utilisant un algorithme heuristique, on gagne du temps de résolution souvent au détriment de la qualité de la solution finale. Il est toutefois possible d'obtenir des solutions très proches de l'optimum dans des limites acceptables en faisant appel à des heuristiques élaborées.

3.3.1 Méthodes exactes

Les approches exactes utilisées dans la littérature pour la résolution des problèmes de configuration des lignes de fabrication se déclinent en deux catégories :

1. Les approches ayant recours à une bibliothèques d'optimisation, comme celles des logiciels ILOG Cplex, ILOG Solver, Dash Xpress MP, Lindo, GAMS, OSL, etc. Pour appliquer une telle approche, il faut d'abord étudier le problème minutieusement en vue de le modéliser de telle manière que la bibliothèque utilisée soit capable de le résoudre de façon efficace. Ici, nous n'abordons pas la modélisation mathématique des problèmes d'optimisation en général, pour cet aspect, voir par exemple [Minoux, 1983; Williams, 1993]. En fonction de la structure du problème à résoudre et de sa modélisation mathématique proposée, diverses approches de résolution s'appuyant sur une bibliothèque d'optimisation sont envisageables, comme par exemple :
   - Procédure par Séparation et Évaluation (PSE) ou « branch and bound » en utilisant ILOG Cplex [Andres *et al.*, 2006; Belmokhtar *et al.*, 2006*b*; Belmokhtar *et al.*, 2006*a*; Corominas *et al.*, 2007; Miralles *et al.*, 2007] ;
   - Procédure par Séparation et Coupe (PSC) ou « branch and cut » à l'aide d'OSL (IBM optimisation subroutine library) [Pinnoi et Wilhelm, 1997; Gadidov et Wilhelm, 2000] ;
   - génération de colonnes en utilisant OSL [Wilhelm, 1999] ;
   - programmation par contraintes à l'aide d'ILOG Solver [Belmokhtar, 2006].

   Bien que les bibliothèques d'optimisation soient développées afin de permettre la résolution efficace d'un large spectre de problèmes d'optimisation, généralement, l'utilisateur peut y ajouter du code supplémentaire pour rendre la résolution encore plus efficace.
2. Toutes ces approches de résolution peuvent être implémentées sans recourir à une bibliothèque d'optimisation. Dans ce cas, il convient non seulement de modéliser le problème, mais aussi de développer et d'implémenter l'algorithme de résolution. Généralement, de telles méthodes sont plus difficiles à mettre en place, mais elles peuvent être mieux adaptées à un problème particulier et tenir compte plus facilement de ses propriétés. Les méthodes de cette catégorie qui ont été mises en place pour la résolution des problèmes de configuration des lignes de fabrication se basent le plus souvent sur :
   - les procédures par séparation et évaluation, parmi les algorithmes les plus connus nous pouvons évoquer FABLE [Johnson, 1988], EUREKA [Hoffman, 1992; Hoffman, 1993], SALOME [Scholl et Klein, 1997; Scholl et Klein, 1999*a*] ; nous pouvons également citer quelques publications plus récentes comme [Bukchin et Rabinowitch, 2006; Dolgui et Ihnatsenka, 2007] ;
   - la programmation dynamique, par exemple, [Jackson, 1956; Held *et al.*, 1963; Bard, 1989], souvent basée sur la recherche du plus court chemin sous contraintes, par exemple, [Gutjahr et Nemhauser, 1964; Easton *et al.*, 1989; Erel et Gökcen, 1999; Dolgui *et al.*, 2006*e*; Dolgui *et al.*, 2008].

3.3.2 Méthodes approchées

Les heuristiques ou méthodes approchées ne garantissent pas que la solution qu'elles fournissent soit optimale, mais de manière générale, elles donnent de bons résultats, souvent plus ou moins proches de l'optimum. Elles sont utilisées pour des problèmes de grande taille ou lorsque le temps de calcul alloué est limité. Plusieurs classifications des méthodes approchées pour la résolution des problèmes d'équilibrage des lignes d'assemblage ont été

proposées dans la littérature, notamment [Talbot *et al.*, 1986; Boctor, 1995; Scholl et Becker, 2006]. Nous classons ces méthodes en trois catégories suivantes :

1. Les heuristiques s'appuyant sur des règles de priorité. Il en existe un certain nombre dans la littérature : elles se différencient par la/les règles employées. Nous plaçons dans cette catégorie :
   - les approches gloutonnes : ces algorithmes effectuent une seule tentative d'affectation des opérations en s'appuyant sur une règle intuitive de choix du meilleur candidat à chaque fois qu'une opération doit être affectée [Raouf et Tsui, 1982].
   - les approches semi-gloutonnes : ces algorithmes effectuent plusieurs itérations de l'affectation d'opérations pour pouvoir mieux exploiter l'ensemble de solutions admissibles et y trouver une solution de bonne qualité. Pour que cela soit possible, la méthode de construction de la liste de candidats et/ou de la sélection de l'opération à affecter doit permettre d'obtenir une solution différente à chaque itération. Souvent cela est atteint en recourant au facteur « hasard » par l'emploi de valeurs aléatoires [Arcus, 1966; Lapierre et Ruiz, 2004; Dolgui *et al.*, 2005*a*]. Un tel algorithme multi-passage procède jusqu'à la satisfaction d'un critère d'arrêt qui peut être exprimé en nombre maximum d'itérations, fixé au préalable, et/ou par le temps alloué, etc. À la fin de l'algorithme, le meilleur résultat de toutes les itérations est fourni comme la solution finale.
2. Métaheuristiques, cette appellation englobe les stratégies de résolution applicables à une large gamme de problèmes d'optimisation combinatoire (voire tous les problèmes). Une métaheuristique est définie par un squelette, constitué des opérateurs de base, qui représente son schéma général, indépendant de la nature du problème à résoudre. Leur application à un problème particulier consiste en l'adaptation des opérateurs de base aux particularités du problème. Parmi les métaheuristiques qui ont été employées pour la résolution de problèmes d'équilibrage se trouvent :
   - les méthodes par voisinage comme la recherche Tabou [Chiang, 1998; Scholl et Voβ, 1996; Lapierre *et al.*, 2006], méthode Kangourou [Mînzu et Henrioud, 1998], GRASP [Bautista *et al.*, 2000; Andres *et al.*, 2006] ou le recuit simulé [McMullen et Frazier, 1998; Vilarinho et Simaria, 2002] ;
   - les méthodes évolutionnistes comme les algorithmes génétiques [Anderson et Ferris, 1994; Leu *et al.*, 1994; Rubinovitz et Levitin, 1995; Kim *et al.*, 1996; Kim *et al.*, 1998; Kim *et al.*, 2000; Ponnambalam *et al.*, 2000; Sabuncuoglu *et al.*, 2000; Baykasoglu et Özbakir, 2006; Levitin *et al.*, 2006].
   - les méthodes constructives comme les colonies de fourmis [McMullen et Tarasewich, 2003; Bautista et Pereira, 2007]
3. Enfin, nous classons dans la troisième catégorie les méthodes effectuant une énumération incomplète de solutions, parmi lesquelles se trouvent :
   - les méthodes qui exploitent une seule branche dans l'arbre d'énumération, par exemple, l'heuristique proposée par Hoffmann pour le SALBP [Hoffman, 1963], ses modifications [Gehrlein et Patterson, 1978; Fleszar et Hindi, 2003] et ses adaptations pour d'autres types de problèmes [Dimitriadis, 2006] ;
   - les approches qui dérivent des méthodes exactes lorsque la résolution exacte est

restreinte par le temps disponible, par exemple une PSE tronquée (PSE H) qui s'arrête quand le temps alloué est révolu [Talbot *et al.*, 1986] ou qui n'exploite pas tous les nœuds [Bukchin et Tzur, 2000].

Puisque les méthodes appartenant à la première catégorie permettent de voir mieux les particularités des problèmes d'équilibrage, nous les présentons dans la section suivante de manière plus détaillée.

Heuristiques basées sur des règles de priorité

Selon l'ordre du traitement des opérations, les « approches orientées opérations » et les « approches orientées postes de travail » sont distinguées. Dans le premier cas, les opérations sont traitées dans l'ordre de leur priorité, qui est calculé selon un critère choisi. Lorsqu'une opération est sélectionnée, elle est affectée au poste le plus en amont où elle peut être accueillie sans enfreindre aucune contrainte du problème.

Cependant, il paraît que cette stratégie d'affectation est moins performante que celle orientée postes de travail [Scholl et Becker, 2006], même si aucune démonstration n'a été encore faite. L'approche orientée postes de travail construit la liste CL des opérations-candidates, pouvant être affectées au poste courant. En utilisant une règle de priorité, une opération est ensuite sélectionnée dans cette liste et ajoutée dans la solution au poste courant. Son affectation entraîne la modification de la liste CL. Si $CL = \emptyset$ et il reste des opérations non affectées, alors un nouveau poste est ouvert et la liste CL est reconstruite. L'algorithme s'arrête lorsque toutes les opérations sont affectées.

Par rapport à la règle utilisée pour le choix d'une opération de la liste CL, il existe, comme nous l'avons déjà mentionné, des approches gloutonnes et des approches semi-gloutonnes. Il y a aussi des approches qui utilisent des règles composites, c'est-à-dire que plusieurs règles de priorité sont appliquées dans un ordre lexicographique, voir par exemple [Raouf et Tsui, 1982; Boctor, 1995]. D'autres approches choisissent une règle à appliquer à l'itération courante de façon aléatoire.

La plupart des règles utilisent les éléments suivants pour calculer la priorité des opérations : le temps de l'opération t_i, le nombre de successeurs directs $SuccD(i)$, le nombre total de successeurs $SuccT(i)$, le poste au plus tôt (E_i) et le poste au plus tard (L_i) où l'opération i peut être affectée. Pour SALBP, la façon la plus simple pour calculer les valeurs des deux derniers paramètres est la suivante :

$$E_i = \left\lceil (t_i + \sum_{j \in PredT(i)} t_j)/T_0 \right\rceil,\ L_i = m_0 + 1 - \left\lceil \sum_{j \in SuccT(i)} t_j/T_0 \right\rceil \tag{3.1}$$

où m_0 est une borne supérieure sur le nombre de postes de travail.

Dans le Tableau 3.2, nous recensons les règles de priorité les plus utilisées dans la littérature pour les problèmes de type SALBP. La fleche $\uparrow$ signifie que l'opération avec la valeur maximale du paramètre correspondant est prioritaire, respectivement, la fleche $\downarrow$ signifie que l'opération avec la valeur minimale du paramètre correspondant est prioritaire.

TAB. 3.2: Règles de priorité

Règle de priorité	Formulation	Référence
RPW : Poids d'opération (Ranked Positional Weight pw)	$\uparrow t_i + \sum_{j \in SuccT(i)} t_j$	[Helgeson et Birnie, 1961]
CRPW : Poids cumulé (Cumulated Positional Weight)	$\uparrow t_i + \sum_{j \in SuccT(i)} pw_j$	[Scholl et Becker, 2006]
ARPW : Poids moyen (Average Ranked Positional Weight)	$\uparrow \frac{pw_j}{\lvert SuccT(i)\rvert + 1}$	[Talbot *et al.*, 1986]
Temps opératoire	$\uparrow t_i$	[Moodie et Young, 1965]
	$\downarrow t_i$	[Boctor, 1995]
Numéro d'opération	$\downarrow i$	[Arcus, 1966]
Poste au plus tôt	$\downarrow E_i$	[Talbot *et al.*, 1986]
Poste au plus tard	$\downarrow L_i$	[Talbot *et al.*, 1986]
Marge d'affectation	$\downarrow L_i - E_i$	[Talbot *et al.*, 1986]
Temps opératoire divisé par le poste au plus tard	$\uparrow t_i / L_i$	[Talbot *et al.*, 1986]
Nombre de successeurs directs	$\uparrow \lvert SuccD(i)\rvert$	[Tonge, 1960]
Nombre total de successeurs	$\uparrow \lvert SuccT(i)\rvert$	[Talbot *et al.*, 1986]
Nombre total de successeurs divisé par la marge d'affectation	$\uparrow \frac{\lvert SuccT(i)\rvert}{L_i - E_i}$	[Talbot *et al.*, 1986]
Poste au plus tard divisé par le nombre total de successeurs	$\downarrow \frac{L_i}{\lvert SuccT(i)\rvert + 1}$	[Talbot *et al.*, 1986]
Nombre d'opérations affectables au poste après l'affectation de l'opération	$\uparrow C$	[Boctor, 1995]

3.4 Analyse des problèmes étudiés dans la littérature

Dans cette section, nous présentons une synthèse des travaux sur l'équilibrage des lignes publiés au cours des dix dernières années qui tiennent compte de la structure des postes de travail, c'est-à-dire $A(k) \cup C(k) \neq \emptyset$. Dans ce cas, les deux problèmes d'optimisation sont résolus simultanément : le **choix de la structure** de chaque poste de travail et l'**équilibrage de la ligne** avec la structure choisie. Pour la présentation, nous utilisons les notations et la classification des paramètres introduites dans la Section 3.2. Nous utilisons également les caractères $\downarrow$ et $\uparrow$ pour dénoter les fonctions objectif à minimiser et à maximiser, respectivement. Il est à noter que hormis quelques exceptions, tous les problèmes analysés possèdent des contraintes de précédence et la contrainte sur le temps de cycle. Pour cette raison, nous ne mentionnons que le cas contraire, c'est-à-dire l'absence de ces contraintes.

TAB. 3.3: Récapitulatif des problèmes traités dans la littérature

Référence	t_i	$P(i)$	$A(k)$	$C(k)$	Contraintes	Objectifs	Méthode
[Askin et Zhou, 1997]	$\in \mathbb{R}$	$p_2 \in \mathcal{A}$	$a_1^s(k) \in \mathcal{A}$	-	$a_1^s(k) \sim p_2(i)$	$\downarrow \sum A(k)$	Heur. règle de priorité
[McMullen et Frazier, 1997] mix	$f(\mu, \sigma^2)$	-	-	$\frac{w_k}{pr_k}$, u_k,	-	$\uparrow U, \uparrow Pr,$ $\uparrow U \cdot P_r$	7 Heur, règles de priorité
[McMullen et Frazier, 1998] mix	$f(\mu, \sigma^2)$	-	-	$\frac{w_k}{pr_k}$, u_k,	-	agrégation des objectifs	Recuit simulé
[Pinnoi et Wilhelm, 1998]	$f(a_1^s(k))$	$p_2 = f(a_1^s(k))$	$a_1^s(k) \in \mathcal{A}$	-	-	$\downarrow \sum A(k)$ $+\sum C(k)$	B&C avec OSL
[Wilhelm, 1999]	$f(a_1^s(k),\ a_2^s(k))$	$p_2 = f(a_1^s(k),\ a_2^s(k)),$ $p_{3,4} = f(a_1^s(k), a_2^s(k), j)$	$a_{1,2}^s(k) \in \mathcal{A}$	-	-	$\downarrow \sum A(k)$ $+\sum C(k)$	GdC avec OSL + CCC
[Bukchin et Tzur, 2000]	$f(a_1^s(k))$	-	$a_1^s(k) \in \mathcal{A}$	-	-	$\downarrow \sum C(k)$	PSE, PSE H
[Bukchin et Rubinovitz, 2002]	$\in \mathbb{R} \setminus f(a_1^s(k))$	-	$a_{1,2}^s(k) \in \mathcal{A}$	-	-	$\downarrow \sum C(k)$	PSE
[Gadidov et Wilhelm, 2000] P1	$f(a_1^s(k))$	$p_2 = f(a_1^s(k))$	$a_1^s(k) \in \mathcal{A}$	-	-	$\downarrow \sum A(k)$ $+\sum C(k)$	B&C avec OSL
[Gadidov et Wilhelm, 2000] P2	$f(a_1^s(k))$	$p_2 = f(a_1^s(k))$	$a_{1,2}^s(k) \in \mathcal{A}$	-	-	$\downarrow \sum A(k)$ $+\sum C(k)$	B&C avec OSL
[Gadidov et Wilhelm, 2000] P3	$f(a_1^s(k))$	$p_2 = f(a_1^s(k)),$ $p_3(i) \in \mathcal{A}$	$a_1^s(k) \in \mathcal{A}$	-	exc	$\downarrow \sum A(k)$ $+\sum C(k)$	B&C avec OSL
[Kim *et al.*, 2000]	$\in \mathbb{R}$	$p_2(i) \in \mathcal{A}$	$a_1^s(k) \in \mathcal{A}$	-	$a_1^s(k) \sim p_2(i)$	$\downarrow m, \downarrow T_0$	Algorithmes génétiques
[Kimms, 2000]	-	$p_2(i) = f(a_1^s(k))$	-	-	non T_0, $a_1^s(k) \sim p_2(i)$	$\downarrow \sum a_1^s(k)$	CGA avec Cplex, Heur. avec CGA
[Lee *et al.*, 2001]	$\in \mathbb{R}$	$p_2(i) \in \mathcal{A}$	$a_1^s(k) \in \mathcal{A}$	-	$a_1^s(k) \sim p_2(i)$	$\uparrow IWR,$ $\uparrow IWS$	Heur. règle de priorité
[Rekiek *et al.*, 2001], [Rekiek, 2001], [Rekiek *et al.*, 2006]	$f(a_1^s(k))$	$p_2(i) \in \mathcal{A},$ $p_3 \in \mathbb{R}$	$a_{1,2,3}^s(k) \in \mathcal{A}$	-	inc, exc, $a_3^s(k) \sim p_2(i)$	Promethee $\Pi : \uparrow SSL,$ $\uparrow \sum C(k)$	Algorithmes génétiques

Référence	t_i	$P(i)$	$A(k)$	$C(k)$	Contraintes	Objectifs	Méthode
[Bukchin et Rubinovitz, 2002]	$\in \mathbb{R} \setminus f(A_k)$	-	$a^s_{1,2}(k) \in \mathcal{A}$	-	-	$\downarrow \sum C(k)$	PSE
[Nicosia *et al.*, 2002]	$f(a_1(k))$	-	$a^s_1(k) \in \mathcal{A}$	-	-	$\downarrow \sum a^s_1(k)$	Progr-tion dyna-mique et PSE
[Vilarinho et Simaria, 2002]	$\in \mathbb{R}$	-	-	w_k	inc, exc	$\downarrow m + \alpha SSL^{ver}_{ecar}$ $+\beta SSL^{hor}_{ecar}$	Recuit simulé
[McMullen et Tarasewich, 2003] mix	$f(\mu, \sigma^2)$	-	-	w_k, u_k, pr_k, $\sum a^s_1(k) \cdot b_1$	-	$\uparrow U, \uparrow Pr$, $\uparrow U \cdot Pr$, $\downarrow \sum a^s_1(k) \cdot b$	Colonies de fourmis
[Lapierre et Ruiz, 2004]	$\in \mathbb{R}$	$p_{2,3,4}(i) \in \mathcal{A}$, $p_5(i) \in \mathbb{R}$	$a^s_{1,2}(k) \in \mathcal{A}$	-	exc, inc, $a^s_1(k) \sim p_2(i)$, $a^s_2(k) \sim p_3(i)$	$\downarrow m$	4 Heur. règles de priorité avec MS Accesss 97
[Agpak et Gökçen, 2005]	$\in \mathbb{R}$	$p_2(i) \in \mathcal{A}$	$a^s_1(k) \in \mathcal{A}$	-	$a^s_1(k) \sim p_2(i)$	$\downarrow \sum a^s_1(k)$	BLP avec Gams-Cplex
[Dimitriadis, 2006]	$\in \mathbb{R}$	-	$a^v_1(k)$	-	-	$\downarrow \sum \dim a^v_1(k)$	Heuristique de Hoffmann
[Lapierre *et al.*, 2006]	$\in \mathbb{R}$	$p_{2,3}(i) \in \mathcal{A}$	$a^s_{1,2}(k) \in \mathcal{A}$	-	exc, inc, $a^s_1(k) \sim p_2(i)$, $a^s_2(k) \sim p_3(i)$	$\downarrow m$	Recherche Tabou
[Levitin *et al.*, 2006]	$f(a^s_1(k))$	-	$a^s_1(k) \in \mathcal{A}$	-	-	$\downarrow T_0$	Algorithmes génétiques
[Corominas *et al.*, 2007]	$f(a^s_1(k))$	$p_{2,3}(i) \in \mathcal{A}$	$a^s_1(k) \in \mathbb{N}$	-	inc	$\downarrow \sum a^s_1(k)$	BLP avec ILOG Cplex 9.0
[Miralles *et al.*, 2007]	$f(a^s_1(k))$	-	$a^s_1(k) \in \mathcal{A}$	-	$\sum a^s_1(k) \leq m$	$\downarrow T_0$	BLP avec ILOG Cplex 9.0

L'analyse des formulations des problèmes déjà traités dans la littérature nous permet de souligner les principales particularités du problème que nous étudions :

- Une fonction spécifique pour calculer la charge des postes de travail : d'une part, l'utilisation des boîtiers multibroches fait que certaines opérations sont exécutées simultanément, d'autres de manière séquentielle ; d'autre part, le temps d'usinage de boîtier dépend des paramètres des opérations qu'il exécute et il ne peut pas être connu à l'avance.
- L'utilisation d'outils étagés permet l'exécution simultanée des opérations liées par une contrainte de précédence, d'où la nécessite d'une modélisation adaptée des contraintes de précédence.
- La densité des contraintes de précédence pour l'usinage est plus petite que pour le processus d'assemblage. En contrepartie, un nombre important de contraintes d'inclusion et d'exclusion apparaît lors de la conception des systèmes d'usinage à boîtiers multibroches.
- La répartition d'opérations est également nécessaire à l'intérieur de chaque poste de travail aux boîtiers qui lui sont attribués. Pour cette raison, le calcul de la charge d'un poste de travail est plus complexe. De même, il faut introduire des contraintes d'exclusion à deux niveaux : pour les postes de travail et pour les blocs.

3.5 Conclusion

Dans ce chapitre, nous avons présenté en détails les problèmes d'optimisation de la configuration des systèmes de fabrication étudiés dans la littérature. Il s'agit le plus souvent des problèmes d'équilibrage des chaînes d'assemblage. Comme dans le cas que nous étudions, la résolution de ces problèmes demande une répartition d'opérations entre un ensemble de postes de travail disposés de manière linéaire. Néanmoins, comte tenu des spécificités du problème que nous avons soulevé, nous pouvons conclure que les approches adaptées aux problèmes d'équilibrage des lignes d'assemblage peuvent être peu efficaces pour la résolution du problème que nous traitons. Cette conclusion nous amène vers la nécessité de développer des méthodes adaptées au contexte qui est le nôtre. Dans le chapitre suivant, nous présentons nos contributions pour la résolution du problème d'optimisation de la configuration des machines de transfert.

Chapitre 4

Optimisation de la configuration des machines de transfert : procédures de pré-traitement

J′ai beaucoup mieux à faire que de m'inquiéter de l'avenir. J'ai à le préparer.

Félix-Antoine Savard

Ce chapitre est consacré à la formulation mathématique du problème d'optimisation de la configuration des machines de transfert et à l'analyse de la structure de ce problème. Dans un premier temps, nous présentons le fonctionnement des machines de transfert. Puis, nous développons un modèle mathématique du problème d'optimisation de leur configuration. Les approches antérieurement proposées pour la résolution de ce problème sont ensuite analysées dans le but de souligner leurs points faibles et de fixer des améliorations envisageables. Enfin, nous développons de nouveaux algorithmes de pré-traitement des données en vue d'améliorer les performances des méthodes d'optimisation et de rendre la résolution exacte possible pour une gamme de problèmes plus large.

4.1 Machines de transfert

Une machine de transfert est constituée d'une séquence de postes de travail reliés par un dispositif de transfert de pièce. Ainsi, chaque pièce chargée sur la machine passe par tous les postes de travail installés, dans l'ordre de leur disposition. La pièce brute est chargée sur le premier poste. Ensuite, elle traverse la machine et subit des opérations d'usinage sur chaque poste de travail pour être transformée en une pièce finie qui sera déchargée du dernier poste de travail. À intervalle de temps régulier, égal au temps de cycle, toutes les pièces se trouvant sur la machine sont simultanément déplacées vers le poste de travail suivant. Au même moment, un nouveau brut est chargé sur le premier poste de travail et une pièce finie est déchargée du dernier poste. Afin d'assurer le déplacement synchrone de pièces et d'éviter

l'utilisation de stocks tampons entre les postes, le temps d'usinage sur chaque poste de travail doit être inférieur au temps de cycle objectif, désigné par T_0.

Chaque poste de travail est muni d'au moins un boîtier. Les boîtiers peuvent comporter un ou plusieurs outils (boîtiers multibroches). Les boîtiers de chaque poste sont activés séquentiellement dans un ordre fixe. Le changement du boîtier actif est effectué pendant un temps τ_b. Le cas échéant, pendant ce temps la pièce peut également être repositionnée. Le changement de boîtiers actifs n'est pas synchronisé entre les différents postes de travail. L'ordre d'activation de boîtiers et l'affectation d'outils à chaque boîtier sont à définir lors de la conception de la machine.

Le dessin d'une machine de transfert (à droite) et un de ses postes de travail (à gauche) sont présentés dans la Figure 4.1 ([Dashchenko, 2005]).

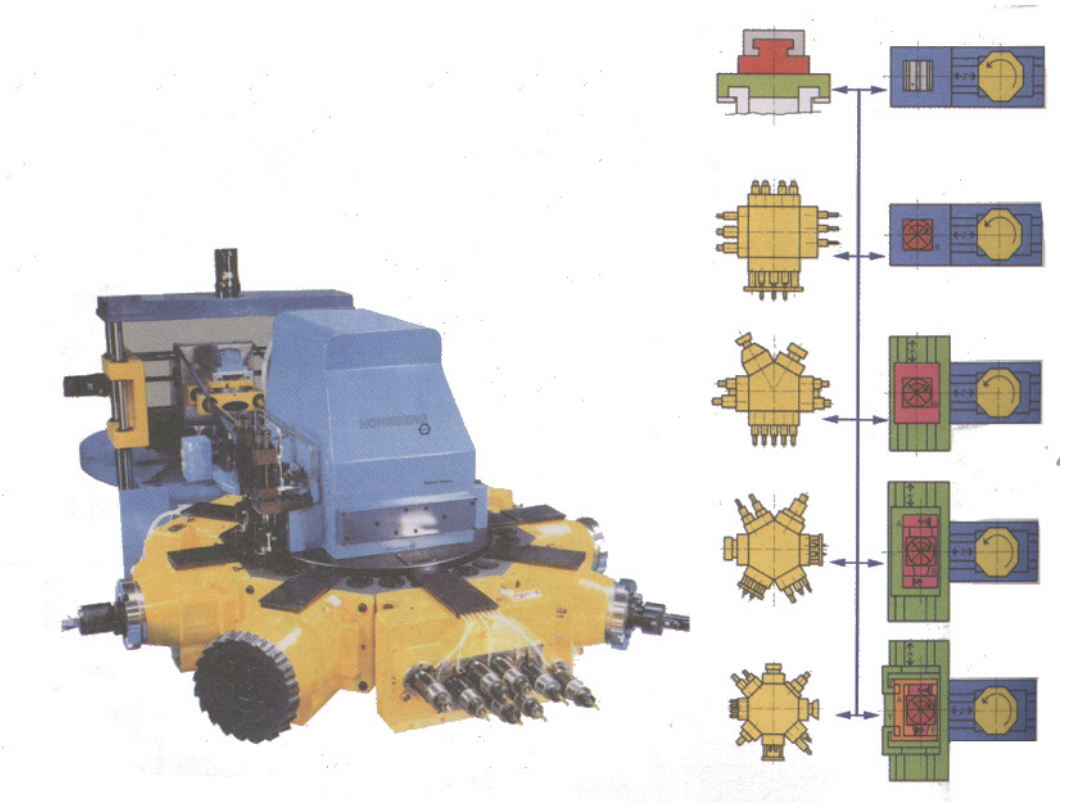


FIG. 4.1 – Machine de transfert

Les machines de transfert appartiennent à la classe des machines spéciales, c'est-à-dire qu'une machine de transfert est généralement dédiée à la fabrication d'une famille de pièces proches, voire un type de pièce. Par conséquent, le choix de la configuration d'une telle machine est très important et doit être fait en analysant minutieusement la pièce à fabriquer et les variantes possibles de configuration, car une erreur à ce stade peut coûter très cher à l'étape de l'exploitation de la machine.

La configuration de telles machines est déterminée par le nombre de postes de travail, par la composition de chaque poste en termes de boîtiers (leur nombre et l'ordre d'activation)

et par les blocs d'opérations affectés à chaque boîtier. Le problème d'optimisation de la configuration de telles machines consiste alors à répartir les opérations d'usinage à des boîtiers d'usinage et à des postes de travail de telle sorte que toutes les contraintes techniques et technologiques soient respectées et le coût de la machine soit le plus petit possible. Rappelons qu'à l'étape d'avant-projet, considérée dans cette thèse, le coût de la machine est estimé par le coût total des postes de travail et des boîtiers utilisés. Dans la section qui suit, nous présentons ce problème d'optimisation de manière formelle en reprenant des notations déjà introduites dans les chapitres précédents.

4.2 Travaux antérieurs et objectifs de l'étude

Le problème d'optimisation de la configuration des machines de transfert a été introduit comme problème d'équilibrage des lignes d'usinage dans [Dolgui *et al.*, 1999]. Dans cette section, nous présentons un aperçu des travaux qui ont déjà été entrepris pour la résolution de ce type de problème.

Dans un premier temps, l'intérêt a été porté sur le développement de méthodes exactes. La première approche qui a été développée consistait en la transformation du problème initial en un problème de recherche du plus court chemin sous contraintes dans un graphe spécialement construit [Dolgui *et al.*, 1999]. Dans cette approche, la difficulté principale réside dans la construction de ce graphe [Dolgui *et al.*, 2006*e*]. Nous présentons cette approche de manière plus détaillée dans la Section 5.2.5.

Ensuite, le problème a été présenté sous forme d'un programme linéaire en variables mixtes (MIP pour Mixed Integer Programming) en vue de le résoudre à l'aide du logiciel d'optimisation Cplex d'ILOG [Finel, 2004]. Les principes qui ont été mis en place pour cette modélisation sont partiellement repris dans la section suivante lors de la présentation de notre modèle mathématique.

L'étude des performances de ces deux méthodes exactes a montré qu'elles sont complémentaires [Guschinskaya et Dolgui, 2007*b*]. Pour les problèmes ayant beaucoup de contraintes, le graphe de décisions utilisé par la première méthode a une taille permettant d'atteindre une solution optimale en temps de calcul raisonnable. Au contraire, les problèmes peu contraints se prêtent bien à une résolution plus rapide par la deuxième méthode. Néanmoins, aucune de ces méthodes n'est pas capable de résoudre les problèmes de grande taille (plus de 100 opérations) sans imposer un temps de calcul trop important. Il est à noter que la deuxième approche s'est avérée tout de même plus sensible à l'augmentation du nombre d'opérations.

Ces résultats ne sont pas inattendus. Les problèmes de ce type sont NP-difficiles [Dolgui *et al.*, 2006*e*], il est donc illusoire de penser que les méthodes exactes peuvent traiter tous les problèmes, quelle que soit leur taille. Il est nécessaire alors de recourir à des méthodes approchées pour la résolution des instances de grande taille.

Quelques heuristiques ont été proposées dans [Finel, 2004; Dolgui *et al.*, 2005*a*; Dolgui *et al.*, 2006*a*; Dolgui *et al.*, 2006*b*], dont la plus performante à ce jour, selon notre étude présentée dans [Guschinskaya et Dolgui, 2007*b*], est l'heuristique FSIC (First Satisfy Inclusion Constraints). L'algorithme de cette heuristique a la même structure que celui des heuristiques « orientées postes de travail » et basées sur une règle de priorité, développées

pour la résolution du SALBP (voir Section 3.3.2). Cependant, l'heuristique FSIC choisit une opération de la liste de candidats non pas en fonction d'une priorité, mais de manière aléatoire. De plus, les procédures de la construction de la liste de candidats et de l'affectation de l'opération sélectionnée tiennent compte des contraintes d'exclusion et d'inclusion (ignorées par la formulation du SALBP), ainsi que de la nécessité d'affecter les opérations non seulement aux postes de travail, mais aussi de les regrouper en blocs.

En analysant les méthodes existantes, nous avons dégagé les trois axes de recherche suivants :

- améliorer les performances des méthodes exactes afin de rendre la résolution exacte possible pour une gamme de problèmes plus large ;
- mettre en place des méthodes approchées plus performantes ;
- développer des stratégies de choix d'une méthode pour la résolution d'un problème particulier.

La suite de ce chapitre est consacrée au développement du premier axe. Nous proposons des procédures de pré-traitements permettant de réduire la taille du problème initial et, par conséquent, de réduire le temps nécessaire pour sa résolution par une méthode exacte.

Dans le Chapitre 5, nous nous intéresserons à la recherche des méthodes approchées efficaces. Quant aux développement des stratégies de choix d'une méthode pour la résolution d'un problème particulier, ce sujet a été abordé dans nos publications, voir [Guschinskaya et Dolgui, 2006; Guschinskaya et Dolgui, 2007*b*].

4.3 Modélisation mathématique

Nous reprenons le modèle mathématique générique introduit dans la section 2.5.6. Notre but est maintenant d'ajouter dans ce modèle la contrainte sur le temps de cycle. La modélisation de cette contrainte doit tenir compte du fait que pour les machines de transfert, le mode séquentiel d'activation de boîtiers est utilisé.

Calcul du temps de cycle de la machine

Comme les postes de travail sont disposés de façon linéaire et sont activés simultanément, le poste le plus lent définit **le temps de cycle de la machine** T. Étant donné que les blocs d'un poste sont activés en séquence, le temps de cycle d'un poste de travail est égal à la somme des temps de ses blocs. Alors, pour calculer T, nous devons passer par le calcul du temps de chaque bloc et, ensuite, de chaque poste de travail en tenant compte des temps auxiliaires existant à chaque niveau.

Comme nous l'avons énoncé dans la section 1.2.3, afin de calculer le temps d'usinage t^b d'un boîtier multibroche, il faut disposer des paramètres d'usinage pour chaque opération qui est affectée à ce boîtier. Ainsi le temps d'usinage t^b du $r$ème boîtier multibroche du $k$ème poste de travail est calculé de la manière suivante :

$$t^b(N_{kr}) = \frac{\max\{\, l_i \mid i \in N_{kr}\}}{V_2(N_{kr})} + \tau^b \tag{4.1}$$

Dès que les temps d'usinage de tous les boîtiers installés sur le $k$ème poste de travail sont déterminés, il est possible de calculer le temps d'usinage sur ce poste de travail t^p de la manière suivante :

$$t^p(N_k) = \sum_{r=1}^{n_k} t^b(N_{kr}) + \tau^p \tag{4.2}$$

Afin de respecter la cadence désirée, il faut que le temps de cycle de chaque poste de travail ne dépasse pas le temps de cycle objectif, c'est-à-dire que la condition suivante soit valide :

$$t^p(N_k) \leq T_0 \text{ pour } k = 1, 2, ..., m \tag{4.3}$$

Dans ce qui suit, nous présentons un modèle linéaire en variables mixtes du problème d'optimisation de la configuration des machines de transfert.

Notations et définitions

Pour la présentation du modèle mathématique du problème d'optimisation de la configuration des machines de transfert, nous utilisons des variables de décision mixtes (réelles et binaires). Un tel modèle peut être facilement implémenté et résolu en recourant à un logiciel muni d'une bibliothèque d'optimisation.

Puisque la modélisation MIP des problèmes de conception des systèmes d'usinage à boîtiers multibroches a déjà été étudiée dans [Finel, 2004; Dolgui *et al.*, 2006*b*], nous reprenons partiellement les principes qui ont été mis en place dans ces publications.

Prenant en compte l'exécution séquentiel des blocs sur un poste de travail, nous pouvons numéroter tous les blocs de la machine selon l'ordre de leur application à la pièce, du premier au dernier. Alors, pour toute solution admissible $S \in \mathbf{S}$, l'ensemble de blocs numérotés b_q peut être représenté par leur séquence $Q(S) = b_1, b_2, ..., b_q, ..., b_{q_0(S)}$, où q est l'indice de bloc cumulé et $q_0(S)$ est la valeur maximale pour q, égale au nombre total de blocs dans la solution S. Étant donné qu'une solution admissible ne peut pas comporter plus de m_0 postes de travail et que chaque poste à son tour ne peut pas contenir plus de n_0 blocs, il en découle que $q_0(S) \leq m_0 n_0 \ , \forall S \in \mathbf{S}$. Toutefois, même sans résoudre le problème d'optimisation, en analysant tout simplement les contraintes pour une instance donnée, il est possible d'obtenir une borne supérieure pour $q_0(S)$ telle que $q_0(S) < m_0 n_0, \forall S \in \mathbf{S}$. Nous présentons les procédures pour le calcul de cette borne dans la Section 4.4.2.

Nous utilisons les variables binaires X_{jq} pour indiquer l'affectation des opérations ($j \in \mathbf{N}$) aux blocs de la séquence $Q(S)$:

$$X_{jq} = \begin{cases} 1 & \text{si l'opération } j \text{ est affectée au bloc } q \\ 0 & \text{sinon} \end{cases}$$

Dans le but de réduire le nombre de variables utilisées, les intervalles $Q(j)$ des indices q de blocs auxquels chaque opération $j \in \mathbf{N}$ peut être assignée sont calculés. Le calcul des intervalles $Q(j)$ est basé sur l'analyse des contraintes du problème et est abordé dans la Section 4.4.2. L'intervalle $Q(j) = [q_j^-, q_j^+]$ est caractérisé par son extrémité gauche q_j^-, qui

donne le numéro du bloc avant lequel l'opération j ne peut pas être affectée (le bloc au plus tôt), et son extrémité droite q_j^+, qui donne le numéro du bloc après lequel l'opération j ne peut pas être affectée (le bloc au plus tard). Ainsi une variable X_{jq} est utilisée si $j \in \mathbf{N}$ et $q \in Q(j)$.

À partir des intervalles $Q(j)$, nous pouvons construire les ensembles $O_b(q)$ qui comportent toutes les opérations qui peuvent être affectées au bloc q : $O_b(q) = \{j \mid q \in Q(j)\}$.

Les valeurs de q auxquels aucune opération n'est affectée dans la solution du problème correspondent aux blocs qui étaient disponibles, mais qui n'ont pas été utilisés (rappelons que l'objectif de l'optimisation est de minimiser le coût total des blocs et des postes).

Pour calculer le coût lié aux blocs (le coût des boîtiers multibroches), nous utilisons les variables binaires Y_q :

$$Y_q = \begin{cases} 1 & \text{si le bloc } q \text{ existe} \\ 0 & \text{sinon} \end{cases}$$

En utilisant les variables Y_q, nous pouvons également obtenir le nombre de postes de travail ouverts. Prenons la séquence Q qui couvre toutes les séquences $Q(S)$ de toutes les solutions admissibles S. La séquence Q peut être scindée en sous-séquences qui correspondent aux postes de travail de la manière suivante : soit $B(k)$ l'intervalle des indices des blocs réservés pour le poste k, cet intervalle est caractérisé par son extrémité gauche b_k^-, qui donne le numéro du premier bloc réservé pour le poste de travail k, et son extrémité droite b_k^+, qui donne le numéro du dernier bloc réservé pour le poste de travail k. Ainsi nous supposons que le poste de travail k est ouvert s'il y a des opérations affectées au bloc b_k^-. Bien évidement, chaque numéro de bloc $q \in Q$ est réservé exactement à un poste de travail et la quantité totale de tous les blocs réservés pour tous les postes de travail est égale à q_0. La façon la plus simple pour définir les intervalles $B(k)$ est de supposer qu'il y a exactement n_0 blocs disponibles sur un poste de travail (ces blocs peuvent être utilisés ou non), ainsi $b_k^- = (k-1)n_0 + 1$, $b_k^+ = kn_0$ et $q_0 = m_0 n_0$. Dans la Section 4.4.2, nous montrons que les intervalles $B(k)$ et, par conséquent, la valeur de q_0 peuvent être réduits en analysant les contraintes du problème.

Comme pour les blocs, il est possible de calculer les intervalles $K(j), j \in \mathbf{N}$ des indices k des postes auxquels l'opération j peut être assignée. Selon notre supposition qu'il y a exactement n_0 blocs réservés par poste de travail, ces intervalles sont calculés à partir des intervalles $Q(j)$ de la manière suivante :

$$k_j^- = \left\lceil \frac{q_j^-}{n_0} \right\rceil \text{ et } k_j^+ = \left\lceil \frac{q_j^+}{n_0} \right\rceil$$

De la même manière, nous pouvons obtenir les ensembles des opérations qui peuvent être affectées au poste k : $O_p(k) = \bigcup_{q \in B(k)} O_b(q)$.

Fonction objectif

La fonction objectif (4.4) représente le coût de la machine, ce coût est constitué de deux composants : le premier correspond au coût des postes de travail, le deuxième au coût de

tous les blocs de la machine.

$$Minimiser\ C(S) = C_1 \sum_{k=1}^{m_0} Y_{b_k^-} + C_2 \sum_{q=1}^{q_0} Y_q \tag{4.4}$$

Dans ce qui suit, nous présentons l'ensemble de contraintes pour notre modèle MIP du problème d'optimisation de la configuration des machines de transfert.

Relations entre les variables de décision

L'équation (4.5) vérifie que le bloc q n'existe que s'il y a des opérations qui lui sont assignées.

$$Y_q \geq X_{jq}; j \in \mathbf{N}; q \in Q(j) \tag{4.5}$$

L'équation (4.6) interdit la création du bloc q, réservé pour le poste k, si le bloc $q-1$, réservé pour le même poste, n'existe pas.

$$Y_{q-1} - Y_q \geq 0; q \in [b_k^- + 1, b_k^+]; k = 1, ..., m_0 \tag{4.6}$$

L'équation (4.7) assure que le poste k ne peut être créé que si le poste $k-1$ existe déjà.

$$Y_{b_k^-} - Y_{b_{k-1}^-} \geq 0; k = 2, 3, ..., m_0 \tag{4.7}$$

Contraintes d'occurrence

L'équation (4.8) impose que chaque opération $j \in \mathbf{N}$ soit affectée à un et un seul bloc.

$$\sum_{q \in Q(j)} X_{jq} = 1; j \in \mathbf{N} \tag{4.8}$$

Contraintes de précédence

Les contraintes de précédence peuvent être exprimées de façons différentes :

$$\sum_{i \in PredD(j)} \sum_{q' \leq q,\ q' \in Q(j)} X_{iq'} \geq X_{jq} |PredD(j)|; j \in \mathbf{N}; q \in Q(i) \tag{4.9}$$

$$\sum_{q' \leq q,\ q' \in Q(i)} X_{iq'} \geq X_{jq}; i \in PredD(j); j \in \mathbf{N}; q \in Q(j) \tag{4.10}$$

$$\sum_{q' \in Q(i)} q' X_{iq'} \geq \sum_{q \in Q(j)} q X_{jq}; i \in PredD(j); j \in \mathbf{N} \tag{4.11}$$

Pour tenir compte des contraintes de précédence, il suffit d'utiliser au moins une représentation parmi (4.9)-(4.11), mais leurs combinaisons peuvent être également utilisées.

Contraintes de compatibilité entre les opérations

L'équation (4.12) introduit les contraintes d'inclusion.

$$\sum_{q \in B(k) \cap Q(j)} X_{jq} = \sum_{q \in B(k) \cap Q(i)} X_{iq}; j, i \in e; e \in I^p; k \in K(j) \tag{4.12}$$

L'équation (4.13) traduit les contraintes d'exclusion au niveau des blocs.

$$\sum_{j \in e} X_{jq} \leq |e| - 1; e \in E^b; q \in \bigcap_{j \in e} Q(j) \tag{4.13}$$

L'équation (4.14) exprime les contraintes d'exclusion au niveau des postes de travail.

$$\sum_{j \in e} \sum_{q \in B(k) \cap Q(j)} X_{jq} \leq |e| - 1; e \in E^p; k \in \bigcap_{j \in e} K(j) \tag{4.14}$$

Contrainte sur le temps de cycle de la machine

Le calcul du temps de cycle de la machine passe par le calcul du temps d'usinage de chaque bloc d'opérations. Or, le calcul du temps d'usinage d'un bloc avec (4.1) a recours à des fonctions non linéaires comme min, max et la division, par conséquent, l'introduction de cette formule dans le modèle (4.4) - (4.14) enfreindra la linéarité du modèle. Toutefois, il est à remarquer que seuls les paramètres de deux (au plus) opérations sont utilisés pour le calcul du temps d'usinage d'un bloc : ceux qui déterminent les valeurs de $L(N)$ et de $v_f(N)$. Ainsi, il devient possible d'estimer le temps de blocs sans perdre la linéarité du modèle mathématique. Pour cela, nous introduisons les paramètres suivants dont les valeurs sont calculées avant la phase d'optimisation à partir des données du problème :

- Pour chaque opération, nous calculons le paramètre t_j comme suit :

$$t_j = \frac{l_j}{v_{f2}(j)} + \tau^b \tag{4.15}$$

  Dans ce cas, t_j représente le temps du bloc q si $X_{jq} = 1$ et $X_{iq} = 0 \ \forall \ i \in \mathbf{N}$ tels que $q \in Q(i)$ et $i \neq j$, c'est-à-dire qu'il y a une seule opération affectée au bloc q (une unité d'usinage monobroche est utilisée pour effectuer l'opération j).
- Pour chaque paire d'opérations i et j telles que $Q(i) \cap Q(j) \neq \emptyset$ et $\{i, j\} \notin E^b$ (rappelons que si $v_{f2}(i) \leq v_{f1}(j)$, alors $\{i, j\} \in E^b$), nous calculons le paramètre t_{ij} de manière suivante :

$$t_{ij} = \frac{\max\{l_i, l_j\}}{\min\{v_{f2}(i), v_{f2}(j)\}} + \tau^b, \tag{4.16}$$

  où t_{ij} représente le temps d'exécution des opérations $i, j \in \mathbf{N}$ quand elles forment à elles seules un bloc.

Nous introduisons une variable réelle auxiliaire $F_q \in [0, T_0 - \tau^p]$ qui détermine le temps du bloc q, $F_q = t^b(N)$, où $N = \{i \,|\, X_{iq} = 1\}$ est l'ensemble des opérations affectées au bloc q.

Puisque $\forall\, i \in N, F_q = t^b(N) \geq t_i$ et $\forall\, i, j \in N, F_q = t^b(N) \geq t_{ij}$ (dont les démonstrations sont immédiates), nous introduisons la contrainte sur le temps de cycle de la machine de transfert sous la forme des équations (4.17)-(4.19), en utilisant les valeurs t_j, t_{ij} et les variables F_q.

$$F_q \geq t_i X_{iq}; i \in \mathbf{N}; q \in Q(i) \tag{4.17}$$

$$F_q \geq t_{ij}(X_{iq} + X_{jq} - 1); i, j \in \mathbf{N}; i < j; q \in Q(i) \cap Q(j) \tag{4.18}$$

$$\tau^p + \sum_{q \in B(k)} F_q \leq T_0; k = 1, ..., m_0 \tag{4.19}$$

Si un bloc est constitué d'une seule opération, son temps est estimé par (4.17), sinon son temps est estimé par (4.18). L'équation (4.19) vérifie que la charge d'aucun poste de travail ne dépasse T_0.

4.4 Procédures de pré-traitement

Dans [Guschinskaya et Dolgui, 2007*b*], nous avons montré que l'efficacité de la résolution du problème (4.4) - (4.19) dépend fortement du nombre de variables et de contraintes du modèle MIP : plus ce nombre est important plus le temps nécessaire pour résoudre le problème est grand. Dans ce qui suit, nous proposons des procédures de pré-traitement qui permettent de diminuer le nombre des contraintes et des variables utilisées. L'algorithme 4.1 présente le schéma général de la phase de pré-traitement. Cette phase a lieu avant la phase de résolution.

Algorithme 4.1 : Schéma de la phase de pré-traitement
Étape 1. Transformer l'ensemble des contraintes (**Propositions 4.1, 4.2**)
Étape 2. Calculer les intervalles $Q(j)$ avec les Algorithmes 4.3 et 4.4
Étape 3. Transformer l'ensemble des contraintes (**Propositions 4.3**, **4.4**)
Étape 4. Modifier l'ensemble des opérations (**Propositions 4.5 - 4.13**)

À l'étape 1, nous remplaçons certaines contraintes par des contraintes plus fortes et fusionnons certaines autres contraintes. Ensuite, à l'étape 2, nous calculons plus finement les intervalles $Q(j)$ en diminuant ainsi le nombre de variables et de contraintes dans le modèle (4.4) - (4.19). À l'étape 3, nous supprimons les contraintes redondantes (elles sont implicitement exprimées par d'autres contraintes du problème). Enfin, à l'étape 4, nous excluons certaines opérations de l'ensemble $\mathbf{N}$ avant la résolution du problème pour les ajouter à la solution optimale lorsqu'elle est trouvée. La diminution de l'ensemble d'opérations permet de réduire la taille du problème à résoudre et, par conséquent, le temps de résolution.

4.4.1 Transformation de l'ensemble de contraintes

La proposition suivante est une généralisation de la coupe *ES* développée dans [Finel, 2004], p. 101.

Proposition 4.1 *Si* $\exists\, e = \{i, j\} \in E^b$ *tel que* $t_i + t_j > T_0 - \tau^p$, *alors si* $\exists\, e' \notin I^p$ *tel que* $\{i, j\} \subseteq e'$, *le sous-ensemble e peut être supprimé de la collection* E^b, *puis rajouté dans la collection* E^p : $E^b = E^b \setminus \{e\}$, $E^p = E^p \cup e$.

Démonstration. La contrainte e exige l'affectation des opérations i et j à deux blocs différents. Cette contrainte sera respectée même si ces deux blocs appartiennent au même poste. Cependant, il est impossible d'affecter les opérations i et j à deux blocs du même poste sans violer la contrainte sur le temps de cycle, car $t_i + t_j > T_0 - \tau^p$. Donc, il est impératif d'affecter les opérations i et j à deux postes distincts, ce qui peut être exprimé par une contrainte $e = \{i, j\}$ appartenant à E^p. Par contre, si l'affectation des opérations i et j à deux postes distincts est interdite par une contrainte d'inclusion, c'est-à-dire que $\exists\, e' \in I^p$ tel que $\{i, j\} \subseteq e'$, alors le problème n'a pas de solution admissible. □

Proposition 4.2 *Si* $\exists\, e = \{i, j\} \in I^p$ *tel que* $t_i + t_j > T_0 - \tau^p$, *alors les opérations i et j doivent forcément être affectées au même bloc afin de respecter toutes les contraintes du problème. Par conséquent, si* $\{i, j\} \notin E^b$, *les opérations i et j peuvent être remplacées par une macro-opération, sinon le problème n'a pas de solution admissible.*

Démonstration. La contrainte e exige que les opérations i et j soient affectées au même poste, soit au poste k. Or, il est impossible d'affecter les opérations i et j à deux blocs différents du même poste sans violer la contrainte sur le temps de cycle, car $t_i + t_i > T_0 - \tau^p$. La seule possibilité de respecter la contrainte e et la contrainte du temps de cycle, c'est d'affecter les opérations i et j au même bloc. Ceci est possible uniquement si $\{i, j\} \notin E^b$. Puisque dans ce cas l'affectation d'une des opérations i ou j définit explicitement l'affectation de l'autre, elles peuvent être remplacées par une macro-opération. Par contre, si $\{i, j\} \in E^b$, alors le problème n'a pas de solution admissible. □

4.4.2 Calcul des intervalles $Q(j)$

Pour le calcul des intervalles $Q(j)$, il suffit de projeter le graphe de précédence sur la séquence des blocs de taille maximale, c'est-à-dire ayant $q_0 = m_0 n_0$. Cette projection doit tenir compte des contraintes de compatibilité. Rappelons que l'intervalle $Q(j) = [q_j^-, q_j^+]$ est caractérisé par son extrémité gauche q_j^- qui donne le numéro du bloc avant lequel l'opération j ne peut pas être affectée et son extrémité droite q_j^+ qui donne le numéro du bloc après lequel l'opération j ne peut pas être affectée.

Les valeurs q_j^- et q_j^+ ont les propriétés suivantes : $\forall\, i, j \in \mathbf{N}$: **si** $(j, i) \in D^{pr}$, **alors** $q_i^- \geq q_j^-$ et $q_i^+ \geq q_j^+$, **si de plus**, $\{j, i\} \in E^b$ (ou $\{j, i\} \in E^p$), **alors** $q_i^- > q_j^-$ et $q_i^+ > q_j^+$.

Nous pouvons construire les intervalles de même type pour les postes de travail $K(j) = [k_j^-, k_j^+]$. Les valeurs k_j^- et k_j^+ ont les propriétés suivantes :

1. $\forall\, i, j \in \mathbf{N}$: **si** $(j, i) \in D^{pr}$, **alors** $k_i^- \geq k_j^-$ et $k_i^+ \geq k_j^+$, **si de plus**, $\{j, i\} \in E^b$ (ou $\{j, i\} \in E^p$), **alors** $k_i^- > k_j^-$ et $k_i^+ > k_j^+$;
2. $\forall\, i, j \in e \in I^p : k_i^- = k_j^-$ et $k_i^+ = k_j^+$.

Nous commençons par la présentation de la méthode de calcul de ces intervalles proposée dans [Finel, 2004], ensuite nous montrerons les améliorations que nous avons apportées.

Calcul des intervalles $Q(j)$ et $K(j)$: méthode de base

Tout d'abord, pour tout $j \in \mathbf{N}$, la valeur q_j^- est calculée en supposant que les opérations de $\mathbf{N}$ sont classées dans l'ordre non décroissant de leur rang dans G^{pr}. La méthode de calcul emploie une procédure pas à pas qui se base sur la notion de distance $d[i,j]$ entre deux opérations i et j qui est définie de la façon suivante :

$$d[i,j] = \begin{cases} 2 & \text{si } \{i,j\} \in E^p, \\ 1 & \text{si } \{i,j\} \in E^b, \\ 0 & \text{si } \{i,j\} \notin E^p \cup E^b \end{cases} \tag{4.20}$$

La valeur q_j^- est calculée en tenant compte des contraintes de précédence et d'exclusion :

$$q_j^- \geq \max\{q_i^- \oplus d[i,j] \mid i \in PredD(j)\} \tag{4.21}$$

où $q \oplus d$ est défini comme suit :

$$q \oplus d = \begin{cases} \lceil q/n_0 \rceil n_0 + 1 & \text{si } d = 2, \\ q + d & \text{si } d \in \{0,1\} \end{cases} \tag{4.22}$$

Les contraintes d'inclusion sont également prises en compte en vérifiant que :

$$q_j^- \geq \max\{(k_i^- - 1)n_0 + 1 \mid \{i,j\} \subseteq e, e \in I^p\}\} \tag{4.23}$$

L'algorithme 4.2 récapitule le calcul des valeurs q_j^-. La variable j_{cur} est utilisée pour savoir si une nouvelle itération est nécessaire ou non. Rappelons que les opérations de $\mathbf{N}$ sont classées dans l'ordre non décroissant de leur rang dans G^{pr}.

Algorithme 4.2 : Algorithme de calcul des intervalles

Initialiser $j_{cur} = 1$
pour $j = 1, ..., |\mathbf{N}|$ **faire**
  $q_j^- = 1$
fin
répéter
  Mettre $j_{min} = j_{cur}$ et $j_{cur} = |\mathbf{N}|$
  pour $j = j_{min} + 1, ..., |\mathbf{N}|$ **faire**
    $q_j^- = \max\{q_i^- \oplus d[i,j] \mid i \in PredD(j)\}$
  fin
  pour chaque $e \in I^p$ **faire**
    Calculer $M(e) = \max\{(\lceil q_j^- / n_0 \rceil - 1)n_0 + 1 \mid j \in e\}$
    si $q_j^- < M(e)$ **pour** $j \in e$ **alors**
      mettre $q_j^- = M(e)$ et $j_{cur} = \min\{j_{cur}, j\}$
    fin
  fin
jusqu'à $j_{cur} = |\mathbf{N}|$

Pour déterminer les valeurs q_j^+, la technique suivante est utilisée :

- le graphe de précédence $\overline{G^{pr}}$ est créé en inversant les arcs du graphe G^{pr} ;
- pour tout $i \in \mathbf{N}$, l'extrémité gauche q_i^- de l'intervalle $Q(i)$ est calculée en appliquant l'Algorithme 4.2 pour $\overline{G^{pr}}$. L'extrémité droite de l'intervalle $Q(i)$ pour le graphe initial G^{pr} est alors égale à : $q_i^+ = m_0 n_0 - \overline{q_i^-} + 1$

Pour le calcul des intervalles $K(j)$, il est supposé que chaque poste contient exactement n_0 blocs. Alors, les valeurs k_j^- et k_j^+ pour tout $j \in \mathbf{N}$, sont déterminées de la manière suivante : $k_j^- = \lceil q_j^- / n_0 \rceil$ et $k_j^+ = \lceil q_j^+ / n_0 \rceil$.

Calcul des intervalles $Q(j)$: méthode améliorée

L'algorithme 4.2 peut être amélioré. En fait, cet algorithme ne considère que les contraintes d'incompatibilité entre les opérations qui sont liées directement par un arc de précédence dans le graphe G^{pr}. L'analyse d'autres contraintes d'incompatibilité permet d'affiner les intervalles. Pour montrer cela, nous prenons les deux exemples suivants.

Exemple 1. Considérons un problème simple avec $\mathbf{N} = \{1, 2, 3, 4, 5\}$, $m_0 = 2$, $n_0 = 2$, $I^p = E^p = \emptyset, E^b = \{1, 5\}$. Le graphe de précédence est présenté dans la Figure 4.2 (les autres données ne sont pas prises en compte dans l'Algorithme 4.2).

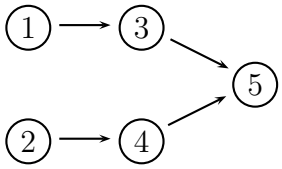


FIG. 4.2 – Exemple 1 : graphe de précédence

Si nous calculons l'intervalle $Q(j)$ avec l'Algorithme 4.2, alors nous obtenons $q_j^- = 1$, $\forall j \in \mathbf{N}$. Pourtant nous pouvons facilement constater que l'opération 5 ne peut en aucun cas être affectée au premier bloc, car les opérations 1 et 5 ne peuvent pas être affectées au même bloc. La contrainte $E^b = \{1, 5\}$ n'est pas prise en compte par l'Algorithme 4.2 parce que l'opération 1 n'est pas un prédécesseur direct de l'opération 5.

Exemple 2. Modifions légèrement le problème de l'exemple précédent. Toutes les données restent les mêmes sauf que la contrainte $E^b = \{3, 4\}$ remplace la contrainte $E^b = \{1, 5\}$.

Le calcul de l'intervalle $Q(j)$ par l'Algorithme 4.2 donne le même résultat : $q_j^- = 1$, $\forall j \in \mathbf{N}$. Pourtant, ici aussi, il est évident qu'au minimum 2 blocs sont nécessaires afin de réaliser les opérations 1, 2, 3, et 4 et, par conséquent, l'opération 5 ne peut en aucun cas être affectée au premier bloc. Alors, nous pouvons en déduire que les intervalles peuvent être calculés plus précisément en considérant les contraintes d'exclusion entre les opérations qui ne sont pas liées directement par un arc dans le graphe G^{pr}.

Les contraintes d'exclusion de type suivant : $e = \{i, j\} \in E^b$ (ou E^p) où $i \in PredT(j)$ peuvent être facilement intégrées dans l'Algorithme 4.2, il suffit juste d'y remplacer $PredD(j)$ par $PredT(j)$, ainsi nous obtenons la formule suivante :

$$q_j^- \geq \max\{q_i^- \oplus d[i, j] \mid i \in PredT(j)\} \tag{4.24}$$

Il est plus difficile de tenir compte des contraintes de type : $e = \{i, h\} \in E^b$ (ou E^p) où $i, h \in PredD(j), i \notin PredD(h), h \notin PredD(i)$. En fait, même sans prendre en compte les contraintes de temps de cycle et d'inclusion, dans le cas général, pour déterminer le nombre exact de blocs nécessaires pour affecter tous les prédécesseurs directs d'une opération en respectant les contraintes de précédence et d'exclusion, il faut résoudre un problème similaire au problème de coloration de graphe mixte [Sotskov *et al.*, 2001]. Toutefois, tenant compte du fait que $|PredD(j)| \ll |\mathbf{N}|$, nous pouvons considérer toutes les paires des opérations $i, h \in PredD(j)$ telles que $\{i, h\} \in E^b(E^p)$ lors du calcul de la valeur de q_j^-. Dans ce cas :

$$q_j^- \geq \max\{\min\{q_i^-, q_h^-\} \oplus d[i, h] \mid i, h \in PredD(j)\} \tag{4.25}$$

Nous pouvons également tenir compte des contraintes de type : $e = \{i, h\} \in E^b$ (ou E^p) où $i \in PredD(j), h \notin PredT(j)$ mais seulement si l'opération h est affectée forcément avant l'opération j, soit $q_j^- \geq q_h^+$. Dans ce cas, à condition que les valeurs q_k^+ soient déjà connues :

$$q_j^- \geq \max\{\min\{q_i^-, q_h^-\} \oplus d[i, h] \mid i \in PredD(j), h \notin PredT(j), q_j^- \geq q_h^+\} \tag{4.26}$$

En réunissant les équations (4.24) - (4.26), nous obtenons la formule suivante :

$$q_j^- = \max \left\{ \begin{array}{l} q_i^- \oplus d[i, j] \mid i \in PredT(j), \\ \min\{q_i^-, q_h^-\} \oplus d[i, h] \mid i, h \in PredD(j), \\ \min\{q_i^-, q_h^-\} \oplus d[i, h] \mid i \in PredD(j), h \notin PredT(j), q_j^- \geq q_h^+ \end{array} \right\} \tag{4.27}$$

L'algorithme 4.3 présente la méthode améliorée pour le calcul des intervalles $Q(j)$.

Algorithme 4.3 : Un nouvel algorithme de calcul des intervalles

```
Initialiser j_cur = 1
pour j = 1, ..., |N| faire
    q_j^- = 1
fin
répéter
    Mettre j_min = j_cur et j_cur = |N|
    pour j = j_min + 1, ..., |N| faire
        Calculer q_j^- avec (4.27)
    fin
    pour chaque e ∈ I^p faire
        Calculer M(e) = max{(⌈q_j^-/n_0⌉ − 1)n_0 + 1| j ∈ e}
        si q_j^- < M(e) pour j ∈ e alors
            mettre q_j^- = M(e) et j_cur = min{j_cur, j}
        fin
    fin
jusqu'à j_cur = |N|
```

Quand les intervalles $Q(j)$ sont déterminés, en utilisant l'Algorithme 4.4, nous pouvons calculer les ensembles $O_b(q)$ et utiliser la contrainte de temps de cycle pour affiner les intervalles $Q(j)$. Pour cela, nous classons les opérations dans l'ordre non décroissant de q_j^-, si deux opérations ont la même valeur de q_j^-, alors elles sont classées dans l'ordre de leur rang dans le graphe de précédence G^{pr}.

Algorithme 4.4 : Algorithme d'affinement des intervalles

Initialiser $j_{cur} = 1$, $q_{cur} = |\mathbf{N}|$
répéter
  Mettre $j_{min} = j_{cur}$ et $j_{cur} = |\mathbf{N}|$
  pour $j = j_{min} + 1, ..., |\mathbf{N}|$ **faire**
    si $q_j^- > q_{cur}$ **et** $PredD(j) \neq \emptyset$ **alors**
      $O_b(q_j^-) = O_b(q_j^-) \setminus j$
      $q_j^- =$
      $$\max \left\{ \begin{array}{l} q_j^- \\ q_i^- \oplus d[i,j] \mid i \in PredT(j), \\ \min\{q_i^-, q_h^-\} \oplus d[i,h] \mid i,h \in PredD(j), \\ \min\{q_i^-, q_h^-\} \oplus d[i,h] \mid i \in PredD(j), h \notin PredT(j), q_j^- \geq q_h^+ \end{array} \right\}$$
      $O_b(q_j^-) = O_b(q_j^-) \cup j$
    fin
    si $q_j^- \neq (k_j^- - 1)n_0 + 1$ **alors**
      si $\sum_{q=b_{k_j^-}^-}^{q_j^- - 1} \min\{t_i \mid i \in O_b(q)\} > T_0 - \tau^p - t_j$ **alors**
        si $\exists e \in I^p$ **tel que** $j \in e$ **alors**
          $AL = e$
        sinon
          $AL = \{j\}$
        fin
        pour chaque $i \in AL$ **faire**
          si $q_i^- < q_{cur}$ **alors**
            $q_{cur} = q_i^-$, $j_{cur} = i$
          fin
          $O_b(q_i^-) = O_b(q_i^-) \setminus j$, $q_i^- = k_i^- n_0 + 1$, $O_b(q_i^-) = O_b(q_i^-) \cup i$
        fin
      fin
    fin
  fin
jusqu'à $j_{cur} = |\mathbf{N}|$

4.4.3 Suppression de contraintes redondantes

Une contrainte est jugée redondante si après sa suppression, le nombre de solutions admissibles n'augmente pas. Afin de pouvoir éliminer une telle contrainte, il suffit de démontrer qu'elle est implicitement imposée par les autres contraintes du problème.

Proposition 4.3 *Si pour un poste de travail k :*

$$\sum_{q \in B(k)} \max\{t_{ij} \mid i,j \in O_b(q)\} \leq T_0 - \tau^p \tag{4.28}$$

alors la contrainte sur le temps de cycle est redondante pour ce poste de travail.

Démonstration. Pour une machine de transfert, la charge d'un poste de travail est égale à la somme des temps de ses blocs. Sachant que $\max\{t_{ij} \mid i,j \in O_b(q)\}$ donne une borne supérieure pour le temps du bloc q, la partie gauche de l'équation (4.28) donne une borne supérieure sur la charge du poste de travail k, que nous désignons par $UB(t^p(N_k))$. Par définition, $UB(t^p(N_k)) \geq t^p(N_k)$. Selon l'équation (4.28), $UB(t^p(N_k)) \leq T_0$, par conséquent, $t^p(N_k) \leq T_0$ pour toute affectation d'opérations au poste k respectant l'ensemble d'autres contraintes du problème. La contrainte de temps de cycle pour le poste k est donc redondante et elle peut être supprimée du modèle, c'est-à-dire l'équation (4.19) aura la forme suivante :

$$\tau^p + \sum_{q \in B(p)} F_q \leq T_0, p = 1, ..., k-1, k+1, ..., m_0$$

□

Remarque 1 Si la condition (4.28) est vérifiée pour tout $k = 1, 2, \ldots, m_0$, alors la contrainte sur le temps de cycle de la machine peut être carrément supprimée. Si besoin est, il est également possible de diminuer le temps de cycle T_0 pour concevoir une machine avec une productivité plus grande.

Proposition 4.4 *Si pour un poste de travail k :*

$$\sum_{q \in B(k)} \min\{t_{ij}, t_i, t_j \mid i,j \in O_b(q)\} > T_0 - \tau^p \tag{4.29}$$

alors la contrainte sur le nombre de blocs est redondante pour le poste p et l'ensemble $B(k)$ peut être réduit.

Démonstration. Sachant que $\min\{t_{ij}, t_i, t_j \mid i,j \in O_b(q)\}$ donne une borne inférieure pour le temps du bloc q, la partie gauche de l'équation (4.29) donne une borne inférieure sur la charge du poste de travail k, que nous désignons par $LB(t^p(N_k))$. Par définition, $LB(t^p(N_k)) \leq t^p(N_k)$. Selon l'équation (4.29), $LB(t^p(N_k)) > T_0$, par conséquent, $t^p(N_k) > T_0$ si tous les blocs $q \in B(k)$ existent. Il en découle que pour respecter la contrainte de temps de cycle, le poste k ne pourra pas contenir $|B(k)|$ blocs, alors $B(k)$ peut être réduit de sorte que la condition (4.29) ne soit plus valide. □

Remarque 2 Soit $q^* \in B(k)$ le bloc le plus en amont dont la suppression est nécessaire pour que la condition (4.29) ne soit plus valide. L'ensemble $B(k)$ peut être réduit d'une des deux façons suivantes : soit mettre à 0 toutes les variables relatives aux blocs $q \in [q^*, b_k^+]$ à supprimer, c'est-à-dire à mettre $X_{jq} = 0,\ \forall j \in O_b(q); q \in [q^*, b_k^+]$; soit mettre $b_k^+ = q^* - 1$, cela implique la modification de q_0 et de tous les ensembles $Q(j)$ pour lesquels $q^* \leq q_j^+$ et tous les ensembles $B(p)$ pour tout $p > k$.

4.4.4 Calcul de bornes inférieures sur le nombre de blocs et sur le nombre de postes de travail

Les algorithmes de calcul des bornes inférieures, développés pour des problèmes d'équilibrage et, d'ailleurs, essentiellement pour le SALBP, sont basés sur l'hypothèse que les temps opératoires sont cumulables lors du calcul de la charge de poste de travail. Cette hypothèse n'est plus valable pour le problème que nous étudions, car dans notre cas, certaines opérations sont exécutées en parallèle et, par conséquent, certains temps opératoires sont masqués. Alors, aucun algorithme existant ne peut être utilisé en tant que tel. Par contre, nous pouvons adapter aux particularités du problème étudié l'approche proposée dans [Patterson et Albracht, 1975] qui est basée sur le calcul des postes de travail au plus tôt et au plus tard. D'abord, nous calculons les intervalles $Q(j)$ et $K(j)$ comme nous l'avons présenté ci-dessus. Puis, nous mettons $n_0 = 1$ et calculons les intervalles $Q^*(j)$ en utilisant l'Algorithme 4.3 mais sans tenir compte des contraintes d'inclusion.

Les intervalles $Q^*(j)$ ainsi calculés sont ensuite utilisés pour le calcul de la borne inférieure sur le nombre de blocs, désignée comme LB^b :

$$LB^b = \max\{q_j^{*-} \mid j \in \mathbf{N}\} \tag{4.30}$$

Quant à la borne inférieure sur le nombre de postes de travail, notée LB^p, elle est calculée de la manière suivante :

$$LB^p = \max\{\lceil LB^b / n_0 \rceil, \max\{k_j^- \mid j \in \mathbf{N}\}\} \tag{4.31}$$

4.4.5 Modification de l'ensemble des opérations

Nous avons remarqué que pour les instances du problème étudié, il existe, souvent, plusieurs solutions optimales. Ces solutions ont le même coût (le même nombre de blocs et de postes de travail) et se distinguent uniquement par une permutation de certaines opérations. En partant de cette observation, nous avons développé des règles qui permettent de détecter les opérations dont l'affectation n'est pas unique lorsque toutes les autres opérations sont affectées de manière optimale. S'il est possible en plus de démontrer (sans résoudre le problème) qu'en affectant a posteriori de telles opérations nous ne changerions ni la faisabilité ni le coût de la solution optimale, ces opérations peuvent être extraites du modèle. Dans les propositions qui suivent, une opération j peut être extraite de l'ensemble $\mathbf{N}$ s'il existe au moins une opération i ($i \neq j$) telle que l'affectation de l'opération j au bloc contenant l'opération i dans une solution admissible quelconque S' du problème $P' : \mathbf{N}' = \mathbf{N} \setminus j$ ne change ni la faisabilité

ni le coût de la solution S'. Ainsi la solution optimale du problème initial $P(j \in \mathbf{N})$ peut être obtenue à partir d'une solution optimale du problème P' $(j \notin \mathbf{N}')$ de taille plus petite.

Afin de trouver les opérations qui peuvent être ainsi extraites, il convient d'analyser les contraintes du problème. Pour structurer cette analyse, nous avons classé les opérations susceptibles d'être extraites en trois catégories :

- les opérations ayant des contraintes de compatibilité mais n'ayant pas de contrainte de précédence : pour insérer une telle opération dans la solution optimale S', il suffit d'y trouver un bloc où l'opération peut être affectée en respectant les contraintes de compatibilité et du temps de cycle ;
- les opérations n'ayant pas de contrainte d'inclusion : puisque l'affectation de telles opérations est restreinte par les contraintes de précédence, une telle opération ne peut être affectée dans la solution S' qu'à un bloc entre celui de son dernier prédécesseur et celui de son premier successeur ;
- les opérations ayant des contraintes de précédence et d'inclusion : l'affectation d'une telle opération dans la solution S' n'est possible qu'à un bloc entre celui de son dernier prédécesseur et celui de son premier successeur et appartenant au même poste de travail que les opérations avec lesquelles elle est liée par des contraintes d'inclusion.

Nous allons développer des conditions à vérifier pour les opérations de chaque catégorie.

Quelques notations

$O_{EB2}(j) = \bigcup_{e \in E^b,\, \lvert e\rvert=2,\, j \in e} e \backslash \{j\}$	est l'ensemble des opérations avec lesquelles l'opération $j \in \mathbf{N}$ ne peut pas être affectée au même bloc ;
$O_{EB*}(j) = \bigcup_{e \in E^b,\, \lvert e\rvert>2,\, j \in e} e \backslash \{j\}$	est l'ensemble des opérations liées par une contrainte d'exclusion au niveau des blocs avec l'opération j, or la contrainte correspondante contient autres opérations que les opérations i et j, par conséquent, il n'est pas possible de savoir d'avance si les opérations i et j seront affectées au même bloc dans la solution optimale ou non ;
$O_{EB}(j) = O_{EB2}(j) \cup O_{EB*}(j)$	est l'ensemble des opérations liées par une contrainte d'exclusion au niveau des blocs avec l'opération j ;
$O_{EP2}(j) = \bigcup_{e \in E^p,\, \lvert e\rvert=2,\, j \in e} e \backslash \{j\}$	est l'ensemble des opérations incompatibles avec l'opération $j \in \mathbf{N}$ au niveau des postes de travail ;
$O_{EP*}(j) = \bigcup_{e \in E^p,\, \lvert e\rvert>2,\, j \in e} e \backslash \{j\}$	est l'ensemble des opérations liées par une contrainte d'exclusion au niveau des postes avec l'opération j, or la contrainte correspondante contient autres opérations que les opérations i et j, par conséquent, il n'est pas possible de savoir d'avance si les opérations i et j seront affectées au même poste de travail dans la solution optimale ou non ;
$O_{EP}(j) = O_{EP2}(j) \cup O_{EP*}(j)$	est l'ensemble des opérations avec lesquelles l'opération $j \in \mathbf{N}$ ne peut pas être affectée au même poste ;

$O_E(j) = O_{EB}(j) \cup O_{EP}(j)$	est l'ensemble des opérations liées avec l'opération $j \in \mathbf{N}$ par une contrainte d'exclusion soit au niveau des blocs, soit au niveau des postes de travail ;
$Q_E(j) = \bigcup_{i \in O_E(j)} Q(i)$	est l'ensemble des indices de blocs où les opérations liées par des contraintes d'exclusion avec l'opération j peuvent être affectées ;
$K_E(j) = \bigcup_{i \in O_E(j)} K(i)$	est l'ensemble des indices des postes où les opérations liées par des contraintes d'exclusion avec l'opération j peuvent être affectées ;
$OMP(j)$	est l'ensemble des opérations qui doivent être affectées au même poste que l'opération j.

Afin de simplifier la présentation qui suit, nous introduisons les définitions suivantes.

Définition 4.1 Nous écrivons $i \succ j$ si les conditions suivantes sont vérifiées pour les opérations i et j :

$$l_i \geq l_j, v_{f2}(i) \leq v_{f2}(j), v_{f2}(i) \geq v_{f1}(j)$$

Lemme 4.1 *Si $i \succ j$ et N_{kr} est le bloc dans une solution admissible S', où l'opération i est affectée ($i \in N_{kr}$, j ne fait pas partie de S'), alors l'affectation de l'opération j à N_{kr} ne changera pas le temps d'usinage de N_{kr}, c'est-à-dire $t^b(N_{kr}) = t^b(N_{kr} \cup j)$.*

Démonstration. La démonstration est triviale si on tient compte de la définition $i \succ j$ et de la Formule (4.1) utilisée pour le calcul du temps de bloc. □

Définition 4.2 Nous écrivons $O_{EB2}(i) \succcurlyeq O_{EB*}(j)$ si la condition suivante est vérifiée pour les opérations i et j :

$$\forall e \in E^b \text{ tel que } |e| > 2 \text{ et } j \in e,\ \exists e_1 \in E^b \text{ tel que } e_1 = \{i, h\} \text{ où } h \in e, h \neq j$$

Définition 4.3 Nous écrivons $O_{EB}(i) \succcurlyeq O_{EB}(j)$, si les deux conditions suivantes sont vérifiées pour les opérations i et j :

- $O_{EB2}(i) \succcurlyeq O_{EB*}(j)$
- $O_{EB2}(j) \subseteq O_{EB2}(i)$

Si la première condition est vérifiée, alors $\{j, i\} \not\subset e,\ \forall e \in E^b$ tel que $|e| > 2$, et si la deuxième condition est vérifiée, alors $\{j, i\} \notin E^b$. Notons que si $\forall e \in E^b$, $j \notin e$, alors $\forall i \in \mathbf{N}, i \neq j$: $O_{EB}(i) \succcurlyeq O_{EB}(j)$.

Définition 4.4 Nous écrivons $O_{EP2}(i) \succcurlyeq O_{EP*}(j)$ si la condition suivante est vérifiée pour les opérations i et j :

$$\forall e \in E^p \text{ tel que } |e| > 2 \text{ et } j \in e,\ \exists e_1 \in E^p \text{ tel que } e_1 = \{i, h\} \text{ où } h \in e, h \neq j$$

Définition 4.5 Nous écrivons $O_{EP}(i) \succcurlyeq O_{EP}(j)$, si les deux conditions suivantes sont vérifiées pour les opérations i et j :

- $O_{EP2}(i) \succcurlyeq O_{EP*}(j)$
- $O_{EP2}(j) \subseteq O_{EP2}(i)$

Si la première condition de la Définition 4.5 est vérifiée, alors $\{j, i\} \not\subset e$, $\forall e \in E^p$ tel que $|\, e| > 2$, et si la deuxième condition est vérifiée, alors $\{j, i\} \notin E^p$.

Lemme 4.2 *Si $O_{EB}(i) \succcurlyeq O_{EB}(j)$ et N_{kr} est le bloc dans une solution admissible S' (j ne fait pas partie de S') du problème P', où l'opération i est affectée ($i \in N_{kr}$), alors l'affectation de l'opération j à N_{kr} ne violera aucune contrainte d'exclusion au niveau des blocs.*

Démonstration. Si S' est une solution admissible, alors les contraintes d'exclusion au niveau des blocs sont respectées pour le bloc N_{kr}. La vérification de la condition $O_{EB}(i) \succcurlyeq O_{EB}(j)$ garantit que les contraintes d'exclusion au niveau des blocs ne seront pas violées après l'affectation de l'opération j au bloc N_{kr}. D'une part, le fait que $O_{EB2}(j) \subseteq O_{EB2}(i)$ garantit que $\forall e \in E^b$ tel que $|\, e| = 2$ et $j \in e$: $e \cap N_{kr} \cup j \;\neq\; e$. D'autre part, le fait que $O_{EB2}(i) \succcurlyeq O_{EB*}(j)$ signifie que le bloc N_{kr} ne contient pas au moins une opération appartenant à chaque sous-ensemble $e \in E^b$ tel que $j \in e$ et $|\, e| > 2$. Par conséquent, si l'opération j est affectée au bloc N_{kr}, aucune contrainte exprimée sous forme $e \in E^b$ telle que $|\, e| > 2$ et $j \in e$ ne sera violée, car une contrainte de ce type est satisfaite si au moins une opération du sous-ensemble correspondant n'est pas affectée au même bloc que les autres. □

Lemme 4.3 *Si $O_{EP2}(i) \succcurlyeq O_{EP*}(j)$ et k est le poste dans une solution admissible S' (j ne fait pas partie de S'), où l'opération i est affectée ($i \in N_k$), alors l'affectation de l'opération j à N_k ne violera aucune contrainte d'exclusion au niveau des postes de travail.*

Démonstration. Par analogie avec la démonstration du Lemme 4.2, où il faut seulement remplacer E^b par E^p, N_{kr} par N_k et « bloc » par « poste ». □

Extraction des opérations sans contraintes de précédence

Dans cette catégorie se trouvent les opérations n'ayant ni prédécesseurs ni successeurs ($PredD(j) = \emptyset, SuccD(j) = \emptyset$) et qui ne sont pas liées par des contraintes d'inclusion avec d'autres opérations ($\forall e \in I^p, j \notin e$, c'est-à-dire $OMP(j) = \emptyset$). Rappelons qu'afin de pouvoir extraire une opération j de l'ensemble $\mathbf{N}$, il est nécessaire de démontrer qu'il existe au moins une opération i ($i \neq j$) telle que l'affectation de l'opération j au bloc contenant l'opération i dans une solution admissible quelconque S' du problème $P' : \mathbf{N}' = \mathbf{N} \setminus j$ ne change ni la faisabilité ni le coût de la solution S'. Le coût de la solution demeure le même si l'affectation de l'opération j ne demande d'ajouter ni des postes ni des blocs supplémentaires. Pour vérifier la faisabilité de la solution après l'affectation de l'opération j, il convient de vérifier le respect des contraintes du problème P.

Proposition 4.5 *Si $\exists j \in \mathbf{N}$ tel que $PredD(j) = \emptyset$, $SuccD(j) = \emptyset$ et $OMP(j) = \emptyset$, et si en même temps $\exists i \in \mathbf{N}$, $i \neq j$ tel que $i \succ j$ et $O_{EB}(i) \succcurlyeq O_{EB}(j)$, $O_{EP}(i) \succcurlyeq O_{EP}(j)$, alors les opérations j et i peuvent être agrégées en une macro-opération.*

Démonstration. Soit r le bloc du poste k dans la solution S' où l'opération i est affectée. Quel que soit ce bloc, si nous y ajoutons l'opération j, les contraintes de précédence et d'inclusion demeureront respectées, car a priori $PredD(j) = \emptyset$, $SuccD(j) = \emptyset$ et $OMP(j) = \emptyset$. Nous avons $i \succ j$, par conséquent, selon le Lemme 4.1, $t^b(N_{kr}) = t^b(N_{kr} \cup j)$. Ainsi après l'affectation de l'opération j au bloc N_{kr}, la contrainte de temps de cycle pour le poste k demeura satisfaite. Les contraintes E^b et E^p seront respectées selon les Lemmes 4.2 et 4.3, car $O_{EB}(i) \gtrdot O_{EB}(j)$ et $O_{EP}(i) \gtrdot O_{EP}(j)$. Les contraintes sur le nombre de blocs et sur le nombre de postes resteront satisfaites, car aucun poste ni bloc supplémentaire ne sera créé. Pour la même raison, le coût de la solution demeura le même. Toutes les contraintes du problème P seront donc respectées et, par conséquent, la solution ainsi obtenue sera une solution admissible du problème P. En conclusion, quel que soit le bloc d'affectation de l'opération i, il est possible d'y ajouter l'opération j sans changer ni coût ni de la faisabilité de la solution, par conséquent, les opérations i et j peuvent être agrégées en une macro-opération. □

Remarque 3 Si $PredD(j) = \emptyset$, $SuccD(j) = \emptyset$, $O_{EP}(j) = \emptyset$, $O_{EB}(j) = \emptyset$ et $OMP(j) = \emptyset$, l'opération j peut être agrégée avec n'importe quelle opération i respectant la condition $i \succ j$.

Proposition 4.6 *Si $PredD(j) = \emptyset$, $SuccD(j) = \emptyset$ et $OMP(j) = \emptyset$, alors l'opération j peut être extraite de l'ensemble* **N**, *si $\exists k \leq LB^p$ tel que $k \notin K_E(j)$ et $\forall i \in O_p(k) : i \succ j$.*

Démonstration. Le poste k fait partie de toute solution admissible S puisque $k \leq LB^p$. Au moins une opération $i \in O_p(k)$ sera affectée au poste k dans la solution S. Soit r le bloc auquel elle est affectée. Quel que soit ce bloc, si nous y ajoutons l'opération j, les contraintes de précédence et d'inclusion demeureront respectées, car $PredD(j) = \emptyset$, $SuccD(j) = \emptyset$ et $OMP(j) = \emptyset$. Puisque $i \succ j$, $t^b(N_{kr}) = t^b(N_{kr} \cup j)$ selon le Lemme 4.1. Les contraintes d'exclusion seront respectées parce que $k \notin K_E(j)$. Les contraintes sur le nombre de blocs et sur le nombre de postes resteront satisfaites, car aucun poste ni bloc supplémentaire ne sera créé. Pour la même raison, le coût de la solution demeura le même. Toutes les contraintes du problème P seront donc respectées et, par conséquent, la solution ainsi obtenue sera une solution admissible du problème P. En conclusion, quel que soit le bloc d'affectation de l'opération i, il est possible d'y ajouter l'opération j sans changer ni coût ni de la faisabilité de la solution, par conséquent, les opérations i et j peuvent être agrégées en une macro-opération. □

Remarque 4 Il suffit de trouver au moins une opération i respectant les conditions de la Proposition 4.6 pour l'agréger avec l'opération j. S'il en existe plusieurs, alors l'opération j peut être agrégée avec n'importe quelle de ces opérations.

Extraction des opérations sans contraintes d'inclusion

Proposition 4.7 *Si $\exists j \in$* **N** *tel que $OMP(j) = \emptyset$, $PredD(j) \neq \emptyset$ et pour $\forall i \in PredD(j)$ toutes les conditions suivantes sont validées :*

- $i \succ j$;
- $O_{EP}(i) \gtrdot O_{EP}(j)$;

– $O_{EB}(i) \succcurlyeq O_{EB}(j)$;

alors nous pouvons extraire l'opération j de l'ensemble $\mathbf{N}$ *en effectuant les modifications suivantes* :

– *des arcs de précédence sont à créer entre tous les prédécesseurs immédiats de j et tous les successeurs immédiats de j dans le graphe de précédence G* ;
– *si $O_{EP}(j) \neq \emptyset$, tous les sous-ensembles e tels que $j \in e$ doivent être supprimés de E^p* ;
– *si $O_{EB}(j) \neq \emptyset$, tous les sous-ensembles e tels que $j \in e$ doivent être supprimés de E^b.*

Démonstration. Soit $q' = \max\{q \,|\, X_{iq} = 1,\, i \in PredD(j)\}$ le dernier bloc N_{kr} (le plus en aval) contenant un des prédécesseurs de j. En y affectant l'opération j, nous respecterons les contraintes de précédence, car d'une part, il n'y aura pas de prédécesseur de j affecté à un bloc $q'' > q'$, étant donné que $q' = \max\{q \,|\, X_{iq} = 1,\, i \in PredD(j)\}$; d'autre part, il n'y aura pas de successeur de j affecté à un bloc $q'' < q'$ puisqu'aucun successeur de j ne pourra être affecté avant tout prédécesseur de j. Cela sera assuré par les arcs de précédence créés entre tous les prédécesseurs immédiats de j et tous les successeurs immédiats de j dans le graphe de précédence G. Les contraintes E^b et E^p seront respectées selon les Lemmes 4.2 et 4.3, car $O_{EB}(i) \succcurlyeq O_{EB}(j)$ et $O_{EP}(i) \succcurlyeq O_{EP}(j)$. Les contraintes d'inclusion demeureront respectées, car $OMP(j) = \emptyset$. Les contraintes sur le nombre de blocs et sur le nombre de postes resteront satisfaites, car aucun poste ni bloc supplémentaire ne sera créé. Pour la même raison, le coût de la solution demeura le même. Toutes les contraintes du problème P sont donc respectées et, par conséquent, la solution obtenue après l'affectation de l'opération j au bloc q' sera une solution admissible du problème P. En conclusion, il est toujours possible d'affecter l'opération j a posteriori sans changer ni coût ni la faisabilité de la solution, par conséquent, cette opération peut être extraite de l'ensemble $\mathbf{N}$. □

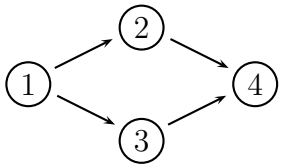
Remarque 5 Si l'opération j a un seul prédécesseur, c'est-à-dire $PredD(j) = \{i\}$, les opérations i et j peuvent être agrégées en une macro-opération. Dans tous les autres cas, l'opération j n'est agrégée avec aucune autre opération, elle est simplement extraite de l'ensemble $\mathbf{N}$ avant la résolution du problème. Nous ne pouvons pas agréger cette opération avec une autre, parce que nous ne connaissons pas a priori avec lequel de ses prédécesseurs directs l'opération j sera affectée dans le même bloc dans la solution optimale. Or, si les conditions de la **Proposition 4.7** sont vérifiées, nous pouvons garantir qu'en affectant l'opération j au bloc $q' = \max\{q \,|\, X_{iq} = 1,\, i \in PredD(j)\}$ de la solution optimale du problème P' nous obtiendrons une solution optimale du problème P.

Remarque 6 Soit $q_1 = \max\{q \,|\, X_{iq} = 1,\, i \in PredD(j)\}$ le bloc le plus en aval contenant un des prédécesseurs de j et q_2 le bloc le plus en amont contenant un des successeurs de j, c'est-à-dire $q_2 = \min\{q \,|\, X_{iq} = 1,\, i \in SuccD(j)\}$. Si $q_1 = q_2$, alors l'opération j doit être affectée au bloc $q = q_1 = q_2$ pour respecter les contraintes de précédence. Si $q_1 < q_2$, alors l'opération j peut être affectée au bloc q tel que $q_1 \leq q \leq q_2$ à condition qu'une telle affectation ne viole aucune contrainte du problème P.

Pour illustrer la **Proposition 4.7**, nous présentons l'exemple suivant.

Exemple 1 Considérons le problème P suivant : $\mathbf{N} = \{1, 2, 3, 4\}$, $\forall j \in \mathbf{N}$: $l_j = 10$, $v_{f1}(j) = 10$, $v_{f2}(j) = 20$, $n_0 = 2$, $m_0 = 3$. Le graphe de précédence et les contraintes de compatibilité

sont présentés dans la Figure 4.3. Il est à noter que les autres données n'ont pas d'importance pour les procédures de modification de l'ensemble d'opérations que nous voulons illustrer.



$$I^p = (\{1,3\}),\ E^b = (\{1,3\}),\ E^p = (\{3,4\})$$

FIG. 4.3 – Exemple 1 : graphe de précédence et contraintes de compatibilité

Pour ce problème, il existe 3 différentes solutions optimales, toutes ayant 2 postes de travail et 3 blocs au total : $S_1 = <(\{1,2\},\ \{3\}),\ (\{4\})>$, $S_2 = <(\{1\},\ \{2,\ 3\}),\ (\{4\})>$, $S_3 = <(\{1\},\ \{3\}),\ (\{2,4\})>$. Ces solutions se différencient par la position de l'opération 2. Puisque les conditions de la **Proposition 4.7** sont vérifiées pour l'opération 2 et cette opération a un seul prédécesseur, l'opération 1, nous agrégeons ces deux opérations en une macro-opération. Ainsi nous obtenons un nouveau problème P' suivant : $\mathbf{N} = \{1,3,4\}$, $\forall j \in \mathbf{N}: l_j = 10$, $v_{f2}(j) = 20$. Le graphe de précédence et les contraintes de compatibilité pour ce nouveau problème sont présentés dans la Figure 4.4.

$$I^p = (\{1,3\}),\ E^b = (\{1,3\}),\ E^p = (\{3,4\})$$

FIG. 4.4 – Exemple 1 : ensemble des contraintes pour le problème réduit

La solution optimale du problème P' est comme suit : $S' = <(\{1\},\ \{3\}),\ (\{4\})>$. Nous appliquons la **Remarque 6** afin de construire les solutions optimales pour le problème P et nous obtenons les solutions S_1, S_2, S_3 en affectant l'opération 2 à des blocs différents. Ces solutions ont le même coût que la solution S' et respectent les contraintes du problème P.

Proposition 4.8 *Si* $\exists j \in \mathbf{N}$ *tel que* $OMP(j) = \emptyset$, $SuccD(j) \neq \emptyset$ *et* $\forall i \in SuccD(j)$ *toutes les conditions suivantes sont validées* :

- $i \succ j$;
- $O_{EP}(i) \succcurlyeq O_{EP}(j)$;
- $O_{EB}(i) \succcurlyeq O_{EB}(j)$;

alors nous pouvons retirer l'opération j *de l'ensemble* $\mathbf{N}$ *en effectuant les modifications suivantes* :

- *des arcs de précédence sont à créer dans le graphe de précédence* G^{pr} *entre tous les prédécesseurs immédiats de* j *et tous les successeurs immédiats de* j ;
- *si* $O_{EP}(j) \neq \emptyset$, *alors il convient de supprimer de* E^p *tous les sous-ensembles* e *tels que* $j \in e$;
- *si* $O_{EB}(j) \neq \emptyset$, *alors il convient de supprimer de* E^b *tous les sous-ensembles* e *tels que* $j \in e$.

Démonstration. La démonstration peut être faite par analogie avec la démonstration de la proposition précédente, sauf qu'ici $q' = \min\{q \mid X_{iq} = 1,\ i \in SuccD(j)\}$ le premier bloc N_{kr} (le plus en amont) contenant un des successeurs de j. □

Afin d'augmenter le nombre d'opérations pour lesquelles les conditions de la **Proposition 4.7** sont valides, nous effectuons la modification de l'ensemble de contraintes selon la **Proposition 4.9**.

Proposition 4.9 *Si* $\exists j \in \mathbf{N}$ *tel que* $OMP(j) = \emptyset$, $PredD(j) \neq \emptyset$, $SuccD(j) \neq \emptyset$ *et toutes les conditions suivantes sont valides* :
- $\forall i \in PredD(j)$, $i \succ j$;
- $O_{EP2}(i) \gtrdot O_{EP*}(j)$;
- $O_{EB2}(i) \gtrdot O_{EB*}(j)$,
- $\forall e \in E^b$ *tel que* $e = \{h, j\}$ *soit* $h \in SuccT(j)$, *soit* $\forall i \in PredD(j), h \in O_{EB2}(i)$;
- $\forall e \in E^p$ *tel que* $e = \{h, j\}$ *soit* $h \in SuccT(j)$, *soit* $\forall i \in PredD(j), h \in O_{EP2}(i)$;

alors chaque contrainte $e' = \{h, j\} \in E^b$*(ou* E^p*) peut être remplacée par un ensemble de contraintes* $\{e''\}$ *où une contrainte* $e'' = \{h, i\} \in E^b$*(ou* E^p*) est créée pour chaque prédécesseur direct* i *de l'opération* j.

Démonstration. S'il existe une contrainte $e' = \{j, g\} \in E^b(E^p)$ et $g \in SuccT(j)$, alors aucun prédécesseur de l'opération j ne peut être affecté au même bloc (poste de travail) que l'opération g en respectant les contraintes de précédence. La création des contraintes $\{e''\} \in E^b(E^p)$ pour chaque prédécesseur de j et la suppression de la contrainte e' nous ramènent aux conditions de la **Proposition 4.7** pour l'opération j. □

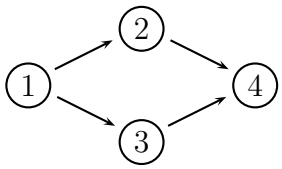
Proposition 4.10 *Si* $\exists j \in \mathbf{N}$ *tel que* $OMP(j) = \emptyset$, $PredD(j) \neq \emptyset$, $SuccD(j) \neq \emptyset$ *et toutes les conditions suivantes sont valides* :
- $\forall i \in SuccD(j)$, $i \succ j$;
- $O_{EP2}(i) \gtrdot O_{EP*}(j)$;
- $O_{EB2}(i) \gtrdot O_{EB*}(j)$;
- $\forall e \in E^b$ *tel que* $e = \{h, j\}$ *soit* $h \in PredT(j)$, *soit* $\forall i \in SuccD(j)$, $h \in O_{EB2}(i)$;
- $\forall e \in E^p$ *tel que* $e = \{h, j\}$ *soit* $h \in PredT(j)$, *soit* $\forall i \in SuccD(j)$, $h \in O_{EP2}(i)$;

alors chaque contrainte $e' = \{h, j\} \in E^b$*(ou* E^p*) peut être remplacée par un ensemble de contraintes* $\{e''\}$ *où une contrainte* $e'' = \{h, i\} \in E^b$*(ou* E^p*) est créée pour chaque successeur direct* i *de l'opération* j.

Démonstration. S'il existe une contrainte $e' = \{j, g\} \in E^b$(ou E^p) où $g \in PredT(j)$, alors aucun successeur de l'opération j ne peut être affecté au même bloc (poste de travail) que l'opération g en respectant les contraintes de précédence. La création des contraintes $\{e''\}$ et la suppression de la contrainte e' nous ramènent aux conditions de la **Proposition 4.8** pour l'opération j. □

Exemple 2 Considérons le problème de l'exemple 1 mais avec les contraintes de compatibilité différentes, elles sont présentées dans la Figure 4.5. Pour cet exemple, les conditions de la Proposition 4.7 ne sont plus respectées pour l'opération 2, puisqu'elle est désormais liée par une contrainte d'exclusion avec l'opération 4, tandis que son prédécesseur, l'opération

1, est compatible avec l'opération 4. Pourtant si l'opération 2 ne peut pas être affectée au même bloc que l'opération 4, alors l'opération 1 ne pourra pas être affectée au même bloc que l'opération 4, puisque l'opération 1 doit être affectée avant l'opération 2 pour respecter les contraintes de précédence.



$$I^p = \emptyset, E^p = (\{1,3\}), E^b = (\{2,4\})$$

FIG. 4.5 – Exemple 2 : graphe de précédence et contraintes de compatibilité

Nous pouvons constater que les conditions de la **Proposition 4.9** sont valides pour l'opération 2 et les conditions de la **Proposition 4.10** sont vérifiées pour l'opération 3. Nous pouvons donc transformer les contraintes d'exclusion selon les **Propositions 4.9-4.10**. Ces transformations aboutissent aux contraintes d'exclusion suivantes : $E^b = (\{1,4\}), E^p = (\{1,4\})$. Puisque la première contrainte est redondante, nous obtenons $E^b = \emptyset, E^p = (\{1,4\})$. Compte tenu de ces contraintes, les conditions de la **Proposition 4.7** deviennent valides pour l'opération 2 et les conditions de la **Proposition 4.8** sont vérifiées pour l'opération 3. Par conséquent, ces opérations peuvent être extraites de l'ensemble **N**.

Pour le problème ainsi réduit, la solution optimale est comme suit : $S' = <(\{1\}), (\{4\})>$. Afin de construire la solution optimale du problème initial, il suffit d'affecter l'opération 2 au même bloc que son dernier prédécesseur, en l'occurrence, l'opération 1 ; l'opération 3 doit être affectée au même bloc que son premier successeur, l'opération 4. Ainsi nous obtenons la solution $S = <(\{1,2\}), (\{3,4\})>$ qui représente la solution optimale pour le problème initial.

Extraction des opérations liées par des contraintes d'inclusion

Maintenant considérons les opérations j dont l'affectation est restreinte par une contrainte d'inclusion ($\exists e \in I^p$ tel que $j \in e$). Il est facile de voir qu'aucune opération appartenant à l'ensemble $OMP(j)$ ne peut être affectée au même poste que toute opération $i \in O_{EP2}(j)$.

Proposition 4.11 *Si $\exists j \in \mathbf{N}$ tel que $OMP(j) \neq \emptyset$ et $SuccD(j) \subseteq OMP(j)$, et si les conditions suivantes sont valides $\forall i \in SuccD(j)$:*

- *$i \succ j$;*
- *$O_{EB}(i) \succcurlyeq O_{EB}(j)$,*

*alors nous pouvons extraire l'opération j de l'ensemble **N** en effectuant les modifications suivantes :*

- *si $|OMP(j)| = 2$, alors la contrainte $e' \in I^p$ tel que $j \in e'$ est supprimée de la collection I^p : $I^p = I^p \backslash e'$, sinon la contrainte e' est remplacée par e'' : $e'' = e' \backslash j$ et $I^p = I^p \backslash e' \cup e''$;*
- *dans le graphe de précédence G^{pr}, des arcs sont à créer entre tous les prédécesseurs immédiats de j et tous les successeurs immédiats de j ;*

- *l'opération j est remplacée par h, qui est un de ses prédécesseurs immédiats, dans tous les $e^{EP} \in E^p$ tels que $j \in e^{EP}$: $e^{EP} = (e^{EP} \backslash j) \cup h$;*
- *si $O_{EB}(j) \neq \emptyset$, alors tous les $e^{EB} \in E^b$ tels que $j \in e^{EB}$ sont supprimés de E^b.*

Démonstration. Soit $q' = \min\{q \,|\, X_{iq} = 1,\, i \in SuccD(j)\}$ le bloc « au plus tôt » dans la séquence Q contenant un des successeurs de j. Si l'opération j est affectée à ce bloc, les contraintes de précédence ne seront pas violées, car d'une part, il n'y aura pas de successeur de j affecté à un bloc $q'' < q'$, étant donné que $q' = \min\{q \,|\, X_{iq} = 1,\, i \in SuccD(j)\}$; d'autre part, il n'y aura pas de prédécesseur de j affecté à un bloc $q'' > q'$ puisqu'aucun prédécesseur de j ne pourra être affecté avant tout successeur de j à cause des arcs de précédence créés entre tous les prédécesseurs immédiats de j et tous les successeurs immédiats de j dans le graphe de précédence G^{pr}. Les contraintes E^p seront respectées grâce aux contraintes e^{EP} modifiées. Les contraintes E^b seront respectées selon le Lemme 4.2, car $O_{EB}(i) \succcurlyeq O_{EB}(j)$. La contrainte d'inclusion initiale e' sera validée puisque toutes les opérations $h \in OMP(j)$ seront affectées au même poste de travail grâce à la nouvelle contrainte e'', le bloc q' où nous affecterons l'opération j appartient également à ce poste de travail. □

Proposition 4.12 *Si $\exists j \in \mathbf{N}$ tel que $OMP(j) \neq \emptyset$ et $SuccD(j) \subseteq OMP(j)$, et si toutes les conditions suivantes sont valides :*

- *$\forall i \in PredD(j)$, $i \succ j$;*
- *$O_{EB}(i) \succcurlyeq O_{EB}(j)$,*

alors nous pouvons extraire l'opération j de l'ensemble $\mathbf{N}$ en effectuant les modifications suivantes de l'ensemble de contraintes :

- *si $|OMP(j)| = 2$, alors la contrainte $e' \in I^p$ tel que $j \in e'$ est supprimée de la collection I^p : $I^p = I^p \backslash e'$, sinon la contrainte e' est remplacée par $e'' : e'' = e' \backslash j$ et $I^p = I^p \backslash e' \cup e''$;*
- *des arcs de précédence sont créés entre tous les prédécesseurs immédiats de j et tous les successeurs immédiats de j dans le graphe de précédence G^{pr} ;*
- *l'opération j est remplacée par h, qui est un de ses successeurs immédiats, dans tous les $e^{EP} \in E^p$ tels que $j \in e^{EP}$: $e^{EP} = (e^{EP} \backslash j) \cup h$;*
- *si $O_{EB}(j) \neq \emptyset$, alors tous les $e^{EB} \in E^b$ tels que $j \in e^{EB}$ sont supprimés de E^b.*

Démonstration. Par l'analogie avec la démonstration de la proposition précédente, mais en prenant $q' = \max\{q \,|\, X_{iq} = 1,\, i \in PredD(j)\}$, c'est-à-dire le bloc « au plus tard » dans la séquence Q contenant un des prédécesseurs de j. □

Proposition 4.13 *Si $\exists j \in \mathbf{N}$ tel que $PredD(j) = \emptyset$, $SuccD(j) = \emptyset$, $j \in e'$, $e' \in I^p$ et $\exists i \in OMP(j)$ tel que $i \succ j$ et $O_{EB}(i) \succcurlyeq O_{EB}(j)$, alors les opérations j et i peuvent être agrégées en une macro-opération. L'opération j est alors extraite de l'ensemble $\mathbf{N}$ en modifiant les données du problème de la manière suivante :*

- *en remplaçant la contrainte e' par $e'' : e'' = e' \backslash j, I^p = (I^p \backslash e') \cup e''$;*
- *en remplaçant j par i dans tous les $e^{EP} \in E^p$ tels que $j \in e^{EP}$: $e^{EP} = (e^{EP} \backslash j) \cup i$;*
- *si $O_{EB}(j) \neq \emptyset$ en supprimant tous les $e^{EB} \in E^b$ tels que $j \in e^{EB}$.*

Démonstration. Par l'analogie avec la **Proposition 4.11**, mais ici q' est le bloc dans la séquence Q où l'opération i est affectée. □

Remarque 7 Il est à noter que les conditions des **Propositions 4.5 - 4.13** ne dépendent pas des valeurs m_0, n_0, T_0, τ^p, τ^b, C_1, C_2.

4.4.6 Expérimentations numériques

Afin d'étudier l'impact de la phase de pré-traitement sur le temps nécessaire pour la résolution de problèmes d'optimisation de la configuration des machines de transfert, nous avons créé des séries de tests ayant différentes densités de contraintes. Les paramètres de ces séries sont récapitulés dans le Tableau 4.1. Chaque série comporte 50 instances distinctes qui ont été générées de manière aléatoire. Les instances de la même série ont la même densité de contraintes de précédence et de compatibilité, les contraintes sont distribuées entre les opérations de façon aléatoire. Afin d'évaluer le niveau de la densité de contraintes d'un problème, nous utilisons les paramètres suivants :

- OS est une mesure de la densité du graphe de précédence [Scholl, 1999] qui peut être calculée de la manière suivante :

$$OS = \frac{2 \sum_{j \in \mathbf{N}} |PredT(j)|}{|\mathbf{N}|(|\mathbf{N}| - 1)} \tag{4.32}$$

  OS=0 si il n'y a aucune contrainte de précédence, OS=1 si le graphe de précédence représente une chaîne d'opérations, c'est-à-dire que les opérations sont complètement ordonnées.
- le nombre de contraintes d'exclusion au niveau des blocs $|E^b|$ et au niveau des postes de travail $|E^p|$, dont les valeurs minimum, maximum et moyennes pour chaque série sont présentées dans le Tableau 4.1.

TAB. 4.1 – Séries de tests pour l'évaluation des procédures de pré-traitement

	Série 1	Série 2	Série 3	Série 4	Série 5	Série 6
OS	bas		moyen		haut	
OS min	0,12		0,4		0,55	
OS max	0,33		0,77		0,85	
OS moy	0,2		0,54		0,7	
$\|E^b\|$	bas	haut	bas	haut	bas	haut
$\|E^b\|$ min	20	36	10	20	11	19
$\|E^b\|$ max	24	47	24	44	23	40
$\|E^b\|$ av	22	42	19	34	17	29
$\|E^p\|$	bas	haut	bas	haut	bas	haut
$\|E^p\|$ min	21	38	12	19	8	15
$\|E^p\|$ max	24	48	24	44	23	43
$\|E^p\|$ av	22	45	19	36	17	31

Chaque série d'instances a été résolue avec un programme linéaire mixte implémenté sous ILOG Cplex 9.0 : d'abord sans phase de pré-traitement et ensuite avec. Dans les deux cas,

le temps de résolution a été limité à 600 sec. Les résultats de la comparaison des temps de résolution par la méthode d'optimisation sans et avec la phase de pré-traitement sont recensés dans le Tableau 4.2. Pour les présenter, nous utilisons les notations suivantes :

NS	le nombre d'instances résolues **avec** le pré-traitement dans la limite de temps alloué (600 sec) ;
ΔNS	le nombre d'instances non résolues **sans** le pré-traitement, mais qui ont été résolues **avec** le pré-traitement ;
NA	le nombre d'instances dont le temps de résolution a été réduit en appliquant le pré-traitement ;
$T_{\min}$, $T_{\max}$, T_{moy}	les valeurs minimum, maximum et moyennes du temps de résolution **avec** le pré-traitement ;
$\Delta T_{\min}$, $\Delta T_{\max}$, ΔT_{moy}	les valeurs minimum, maximum et moyennes absolues du gain de temps obtenu par l'application du pré-traitement ;
$\Delta T\%_{\min}$, $\Delta T\%_{\max}$, $\Delta T\%_{\text{moy}}$	les valeurs minimum, maximum et moyennes du gain de temps relatif ;
$N^{del}_{\min}$, $N^{del}_{\max}$, N^{del}_{moy}	les valeurs minimum, maximum et moyennes du nombre des opérations qui ont été extraites lors de la phase d'optimisation en utilisant les Propositions 4.5-4.13.

TAB. 4.2 – Résultats de l'évaluation des procédures de pré-traitement

	Série 1	Série 2	Série 3	Série 4	Série 5	Série 6
NS	49	30	45	33	48	35
ΔNS	4	2	1	3	5	1
NA	42	20	32	20	35	28
$T_{\min}$	0,56	1,94	0,45	0,77	0,28	0,49
$T_{\max}$	600	600	600	600	600	600
T_{moy}	40,72	277,48	75,73	265,08	112,8	193,69
$\Delta T_{\min}$	0	0	0	0	0	0
$\Delta T_{\max}$	559,74	453,59	544,36	545,36	276,48	51,91
ΔT_{moy}	40,43	72,97	19,5	70,84	22,75	40,34
$\Delta T\%_{\min}$	0	0	0	0	0	0
$\Delta T\%_{\max}$	93,28	97,31	97,76	91,95	95,35	85,22
$\Delta T\%_{\text{moy}}$	26,01	29,91	24,46	29,71	25,85	30,02
$N^{del}_{\min}$	2	0	1	0	0	0
$N^{del}_{\max}$	12	7	13	9	16	7
N^{del}_{moy}	5,64	2,74	5,16	2,5	5,06	2,3

Avec l'utilisation des procédures de pré-traitement proposées, le temps de résolution a été réduit pour les 192 instances sur 300 testées, ce qui représente 64% d'instances. Si

nous ne considérons pas les instances qui n'ont pas été résolues même avec l'application du pré-traitement, le pourcentage des instances pour lesquelles le gain de temps a été observé augmente jusqu'à 72 % des cas. La Figure 4.6 montre la distribution des gains sur l'ensemble des instances testées.

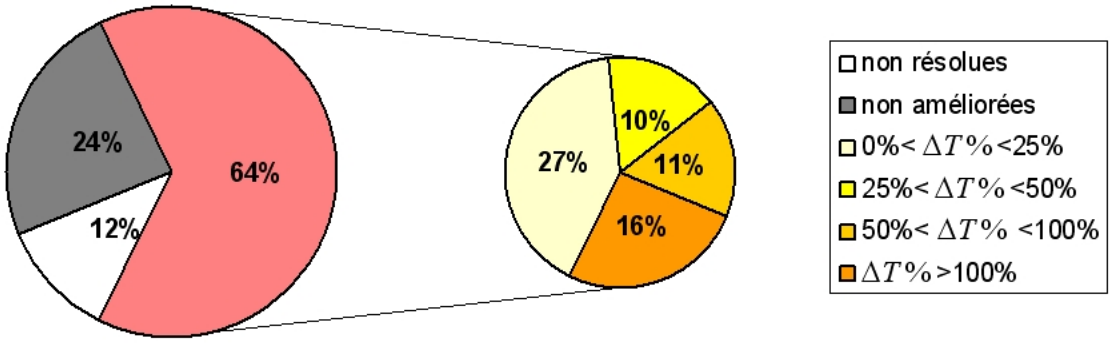


FIG. 4.6 – Les gains pour l'ensemble des instances testées

Dans la Figure 4.7, nous analysons l'impact du niveau de contraintes d'exclusion sur le gain de temps obtenu. Pour les instances ayant un niveau bas de contraintes d'exclusion, le pourcentage des cas où le temps de résolution a été réduit est supérieur par rapport au pourcentage moyen. Cet effet est lié au fait qu'avec l'augmentation du nombre de contraintes d'exclusion, le nombre d'opérations qui peuvent être extraites décroît. Il est à noter que ceci se confirme si nous analysons les valeurs du paramètre N^{del}.

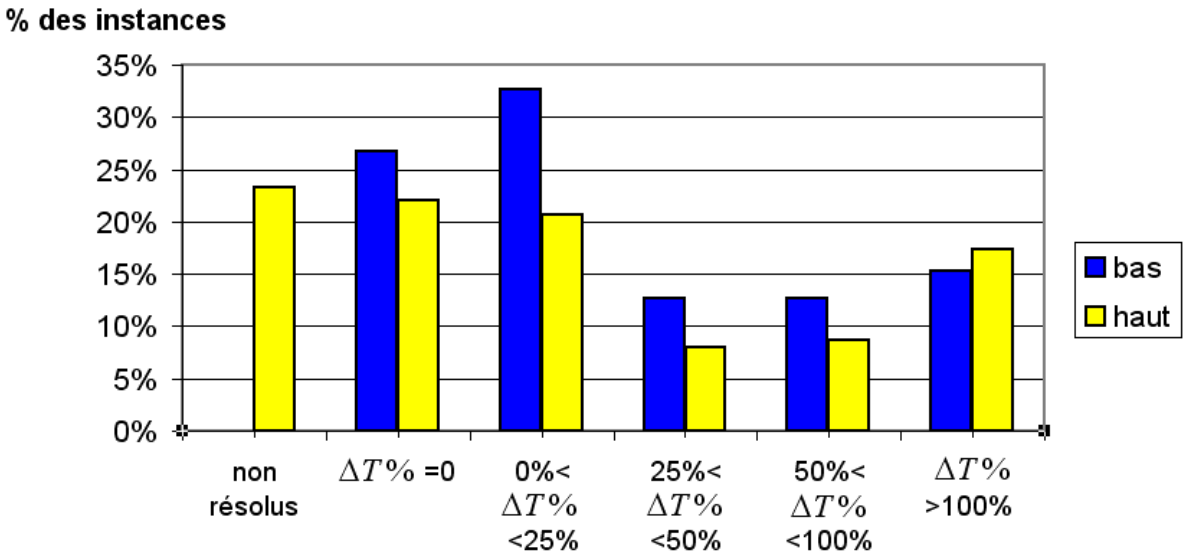


FIG. 4.7 – Impact du niveau de contraintes d'exclusion

Dans la Figure 4.8, nous analysons l'impact du niveau de contraintes de précédence sur le gain de temps obtenu en appliquant la phase de pré-traitement. Rappelons que les instances des Séries 1 et 2 ont un niveau bas de ces contraintes, les instances des Séries 3 et 4 ont un niveau moyen, tandis que les instances des Séries 5 et 6 ont un niveau haut.

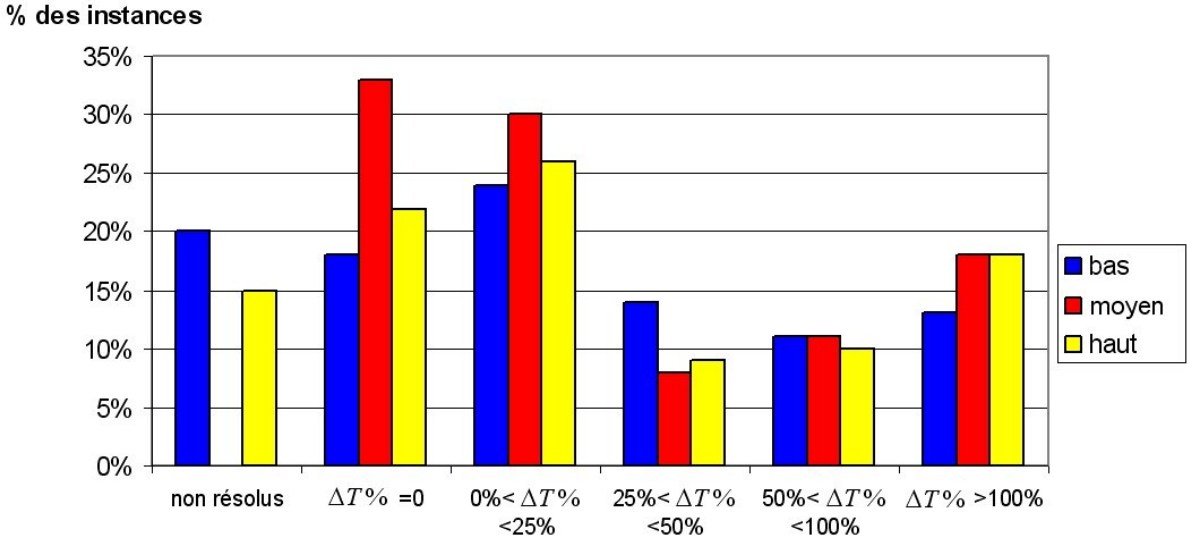


FIG. 4.8 – Impact des contraintes de précédence

Tout d'abord, nous devons remarquer que toutes les instances ayant le niveau moyen de contraintes de précédence ont été résolues dans la limite de temps disponible. Nous pouvons alors constater que ce type d'instances se révèle plus facilement abordable pour la résolution avec ILOG Cplex 9.0. Ensuite, nous pouvons observer que les gains de temps avec l'application de la phase de pré-traitement sont également plus importants lors de la résolution des instances ayant le niveau moyen de contraintes de précédence. Ceci peut être expliqué par le fait que la résolution des instances avec le bas niveau de contraintes de précédence est plus difficilement abordable par la méthode d'optimisation, car il existe plus d'affectations d'opérations possibles. De l'autre côté, lorsque le nombre de contraintes de précédence croît, le nombre des opérations qui peuvent être extraites décroît. Par conséquent, le niveau moyen de contraintes de précédence représente un juste milieu entre les deux extrémités. Pour cette raison, les meilleurs résultats ont été obtenus pour de telles instances.

Pour étudier l'impact du nombre d'opérations, nous avons testé les procédures de pré-traitement sur un ensemble de problèmes de référence présentés dans la littérature, à savoir dans [Dolgui *et al.*, 2006*b*]. Le Tableau 4.3 fournit les résultats de ces expérimentations. Les colonnes $|\mathbf{N}|$, m_0, n_0, OS, $|E^b|$, $|E^p|$ et $|I^p|$ contiennent les caractéristiques des problèmes testés. Ces problèmes ont été d'abord résolus moyennant un programme implémenté sous ILOG Cplex 9.0 sans phase de pré-traitement ; les temps de résolution correspondants sont présentés dans la colonne MIP. La notion « >10h » signifie que la solution optimale du problème n'a pas été trouvée après 10 heures de calcul. Ensuite, nous avons résolu les mêmes problèmes par le biais du même programme mais en faisant appel au préalable aux procédures

de pré-traitement que nous avons développées dans ce chapitre. Les temps de résolution incluant le temps de la phase de pré-traitement sont rapportés dans la colonne PT.

TAB. 4.3 – Résultats pour les problèmes de référence présentés dans la littérature

$\|\mathbf{N}\|$	m_0	n_0	OS	$\|E^b\|$	$\|E^p\|$	$\|I^p\|$	MIP	PT
23	4	4	0,73	4	4	3	431,86″	40,328″
35	4	5	0,68	12	4	5	414,375′	8,687″
38	3	7	0,66	4	6	5	1950″	109,5″
100	3	7	0,58	6	7	5	>10h	501,78″
120	3	7	0,42	14	7	8	>10h	523,86″

Les résultats obtenus nous permettent de constater que pour tous les problèmes testés, le temps de résolution a été divisé au moins par 10 avec l'application de la phase de pré-traitement. Ceci peut être considéré comme un très bon résultat. Il est à noter que l'utilisation des procédures de pré-traitement que nous avons développées a également permis de trouver assez rapidement les solutions optimales pour les problèmes qui n'ont pas été résolus auparavant après 10 heures de calcul. Avec nos procédures de pré-traitement, le temps de résolution de ces problèmes ne dépasse pas 10 minutes. Ceci témoigne de l'efficacité des procédures que nous avons mises en place.

En ce qui concerne l'impact du nombre d'opérations sur l'efficacité des procédures développées, nous pouvons observer le résultat assez prévisible. Lors de la résolution des problèmes de petite taille, le temps de résolution n'est pas très élevé et le gain de temps absolu obtenu par l'application de la phase de pré-traitement est moins important. Avec l'augmentation de la taille des problèmes, le temps de résolution croît mais le gain de temps aussi. Ceci est lié au fait que le nombre d'opérations plus important rend possible l'augmentation du nombre d'opérations extraites. La différence de taille du problème initial et du problème réduit permet d'obtenir un gain de temps de résolution considérable.

4.5 Conclusion

Dans ce chapitre, nous avons étudié le problème d'optimisation de la configuration des machines de transfert. Les machines de ce type appartiennent à la classe des systèmes d'usinage à boîtiers multibroches ayant un mode séquentiel d'activation de boîtiers.

Dans un premier temps, nous avons analysé les méthodes proposées auparavant pour la résolution de ce type de problèmes. Cette analyse nous a permis d'indiquer les points faibles de ces méthodes et de dégager des axes de développement nécessaires. Notamment, nous avons relevé la nécessité d'amélioration des méthodes exactes et l'absence des méthodes heuristiques efficaces.

Dans un deuxième temps, afin de rendre la résolution exacte accessible pour une gamme de problèmes plus large, nous avons mis en œuvre des procédures de pré-traitement. Les procédures développées ont pour objectifs :

1. de transformer l'ensemble des contraintes, notamment de supprimer les contraintes redondantes,
2. de calculer le plus précisément possible les intervalles d'indices des blocs possibles pour chaque opération et les bornes inférieures sur le nombre de postes de travail et de blocs afin de réduire le nombre de variables utilisées,
3. enfin, d'extraire de l'ensemble d'opérations les opérations dont l'affectation peut être effectuée a posteriori, sans modifier pour autant le coût et la faisabilité de la solution retenue lors de la phase d'optimisation.

Pour tester l'efficacité des procédures développées, nous avons utilisé un échantillon de problèmes de complexité différente. Les résultats obtenus montrent des améliorations significatives lors de la résolution de problèmes de grande taille, pour certains problèmes, par exemple, le temps de résolution a été divisé par 10 après l'application des procédures de pré-traitement développées. De plus, nous avons évalué l'impact des différentes contraintes du problème et du nombre d'opérations sur l'efficacité des procédures de pré-traitement mises en place.

Dans le chapitre suivant, nous exploitons une autre piste pour la résolution efficace des problèmes de grande taille, il s'agit de développer des méthodes approchées performantes.

Chapitre 5

Optimisation de la configuration des machines de transfert : méthodes approchées

Le désir fleurit, la possession flétrit toutes choses.

Marcel Proust

Dans ce chapitre, nous présentons trois nouvelles approches heuristiques pour la résolution du problème d'optimisation de la configuration des machines de transfert. D'abord, nous proposons différentes variantes d'une heuristique basée sur l'approche Monte Carlo. Puis, nous dotons la meilleure variante de cette heuristique d'une phase d'amélioration des solutions. Cette phase consiste à former, en utilisant des solutions initialement obtenues, des sous-problèmes et à résoudre ces derniers par une méthode exacte. Puisque cette approche utilise des méthodes exactes ainsi que heuristiques, nous l'appelons approche mixte. Enfin, nous mettons en place une métaheuristique de type GRASP. Toutes les approches proposées sont évaluées sur un échantillon de problèmes dont la structure est similaire à celle des problèmes réels existant sur le terrain industriel. Ceci nous permet de tirer des conclusions pertinentes sur les performances des approches proposées vis-à-vis des problèmes réels.

5.1 Approche heuristique

L'étude effectuée dans [Guschinskaya et Dolgui, 2007*b*] a permis de conclure que la meilleure heuristique proposée pour la résolution du problème d'optimisation de la configuration des machines de transfert est l'heuristique FSIC, publiée dans [Dolgui *et al.*, 2005*a*]. Nous commençons par la présentation de cette heuristique et ensuite nous proposons quelques extensions permettant d'améliorer ses performances.

5.1.1 Schéma général de l'approche

L'heuristique FSIC est basée sur l'approche Monte Carlo. Ainsi lors de la construction d'une solution, un choix aléatoire est effectué s'il existe plusieurs décisions possibles. Le recours au facteur « hasard » peut amener à des solutions de bonne qualité ou même optimales, mais également à des solutions de mauvaise qualité, voire inadmissibles (à cause de la contrainte sur le nombre maximum de postes de travail). Pour cette raison, une itération effectuée par l'heuristique FSIC peut être soit « concluante » si elle donne une solution admissible pour le problème, soit « non concluante » si toutes les opérations n'ont pas été affectées et aucune action n'est possible car le nombre de postes de travail dépasse le nombre maximum autorisé. Au total, l'algorithme effectue TR_{tot} itérations, et plus ce nombre est grand plus il y a des chances de trouver une solution de bonne qualité. Pour essayer d'éviter la répétition des solutions générées, la « graine » utilisée par le générateur de nombres pseudo-aléatoires (désignée comme *seed*) est modifiée au début de chaque itération. Plusieurs conditions d'arrêt de l'algorithme sont utilisées, à savoir :

- l'écoulement du temps disponible ($T_{cur} > T_{res}$, où T_{cur} est le temps consommé pour la résolution, T_{res} est le budget du temps disponible) ;
- l'obtention d'une solution avec un coût C_{min} inférieur à un seuil donné C_{stop} ;
- le dépassement des valeurs maximales autorisées ($TR_{tot^{aut}}$ et $TR_{nimp^{aut}}$) pour le nombre d'itérations effectuées TR_{tot} ou pour le nombre d'itérations sans amélioration TR_{nimp}.

L'Algorithme 5.1 présente le schéma général de l'heuristique FSIC.

Algorithme 5.1 : Schéma général de l'heuristique FSIC

Initialiser $C_{min} = \infty$, $TR_{tot} = 0$, $TR_{nimp} = 0$
répéter
  Attribuer une valeur à *seed*
  Trouver une solution S_{cur} ayant le coût C_{cur} en essayant d'affecter les opérations de l'ensemble **N**
  si *La solution* S_{cur} *est admissible* **et** $C_{cur} < C_{min}$ **alors**
    $C_{min} = C_{cur}, S_{min} = S_{cur}, TR_{nimp} = 0$
  sinon
    $TR_{nimp} = TR_{nimp} + 1$
  fin
  $TR_{tot} = TR_{tot} + 1$
jusqu'à $T_{cur} > T_{res}$ **ou** $TR_{nimp} > TR_{nimp^{aut}}$ **ou** $TR_{tot} > TR_{tot^{aut}}$ **ou** $C_{min} < C_{stop}$

Dans la sous-section suivante, nous détaillons la procédure d'affectation des opérations.

5.1.2 Heuristique FSIC : version de base

L'heuristique FSIC utilise la même approche que celle des heuristiques « orientées postes de travail » avec des règles de priorité, développées pour la résolution du SALBP. Rappelons qu'elles comprennent trois opérateurs de base, à savoir : la construction de la liste d'opérations-candidates *CL* qui regroupe toutes les opérations qui peuvent être affectées au

moment de la décision, la sélection d'une opération de la liste *CL* qui sera effectivement affectée, et la procédure de son affectation qui entraîne la modification de la liste des opérations non-affectées. Dans ce qui suit, nous décrivons la réalisation de ces trois opérateurs dans l'heuristique FSIC, proposée dans [Finel, 2004].

Construction de la liste *CL* de candidats

Soit m le poste de travail courant et n_m le bloc courant de ce poste de travail. Pour construire la liste *CL*, l'ensemble des opérations non affectées est analysé et l'opération i est placée dans la liste *CL*, si les conditions suivantes sont respectées :

1. Tous les prédécesseurs de l'opération i sont déjà affectés, soit si $(j, i) \in D^{pr}$, alors $j \in \bigcup_{k=1}^{m} \bigcup_{r=1}^{n_m} N_{kr}$;
2. t_i est inférieur au temps libre du poste courant (sans compter le bloc courant), c'est-à-dire :
   $t_i \leq T_0 - \sum_{r=1}^{n_m-1} t^b(N_{mr})$;
3. L'affectation de l'opération i au poste courant n'enfreindrait pas les contraintes de type E^p, soit $\forall e \in E^p$ tel que $i \in e : e \cap (N_m \cup \{i\}) \neq e$.

Sélection d'une opération i de la liste *CL*

L'opération i est choisie dans la liste *CL* de manière aléatoire. Pour cela, l'approche Monte-Carlo [Hammersley et Handscomb, 1964; Calos et Whitlock, 1986; Decker, 1991] est utilisée.

Affectation de l'opération i

Afin de diminuer le nombre de solutions inadmissibles, l'algorithme affecte les opérations liées par des contraintes d'inclusion en priorité, comme nous l'expliquons ci-dessous.

Si $\nexists e \in I^p$ tel que $i \in e$, alors l'algorithme essaie d'affecter l'opération i au bloc courant. Si cela est impossible à cause des contraintes de type E^b, alors un nouveau bloc est ouvert pour le poste courant. Si un nouveau bloc ne peut pas être créé à cause de la contrainte sur le temps de cycle ou à cause du nombre de blocs déjà ouverts, et si en même temps il est encore possible de créer un nouveau poste (le nombre de postes est inférieur à m_0), alors un nouveau poste est créé, sinon l'itération en cours est considérée comme non concluante et $C_{cur}(S) = \infty$.

Si $\exists e \in I^p$ tel que $i \in e$, alors l'algorithme crée une liste complémentaire *AL*. Cette liste contiendra toutes les opérations dont l'affectation au poste courant est nécessaire afin que la contrainte e soit respectée. Par conséquence, l'opération i est placée en tête de cette liste ainsi que toutes les opérations j telles que $j \in e$, $j \neq i$. Puis, pour chaque opération j, les contraintes de précédence sont analysées et tous ses prédécesseurs non affectés sont également ajoutés à la liste *AL*. Ensuite, pour chaque nouvelle opération ajoutée, les contraintes d'inclusion et de précédence sont analysées de manière recursive.

Les opérations de la liste *AL* sont affectées dans l'ordre de leur placement dans cette liste. La procédure d'affectation de toute opération de la liste *AL* est la même que l'affectation d'une opération pour laquelle $\nexists e \in I^p$ tel que $i \in e$, à une seule exception : si l'affectation d'une opération de la liste *AL* s'avère impossible, alors l'algorithme annule toutes les affectations d'opérations effectuées à partir de cette liste. Ensuite, l'algorithme ferme les bloc et poste courants. Un nouveau poste est ouvert et devient courant. Une deuxième tentative d'affectation de toutes les opérations de la liste *AL* est effectuée : si elle échoue à nouveau, l'itération en cours est considérée comme non concluante et $C(S) = \infty$.

Soit N^{na} l'ensemble des opérations non affectées. L'algorithme 5.2 décrit la construction des solutions par l'heuristique FSIC. La procédure *ASSIGN*(*AL*) correspond à l'algorithme d'affectation d'une opération i, décrit ci-dessus. Elle retourne la valeur « vrai » si l'affectation est réussie et la valeur « faux », sinon.

Algorithme 5.2 : Construction des solutions par l'heuristique FSIC

```
Set m = 1, n_m = 1, C_cur = C_1 + C_2, N^na = N, assign = vrai
tant que N^na ≠ ∅ faire
    si assign = vrai alors
        Construire la liste CL
    fin
    si CL = ∅ alors
        si m + 1 ≤ m_0 alors
            m = m + 1, n_m = 1, C_cur = C_cur + C_1 + C_2
        sinon
            C_cur = ∞, Quitter
        fin
    sinon
        assign = faux
        Choisir au hasard une opération j dans la liste CL
        si ∃e ∈ I^p tel que i ∈ e alors
            Construire AL
            S_copy = S_cur, C_copy = C_cur
        sinon
            AL = {j}
        fin
        assign = ASSIGN(AL)
        si assign = faux alors
            S_cur = S_copy, C_cur = C_copy
        sinon
            N^na = N^na \ AL
        fin
    fin
fin
```

Dans la sous-section suivante, nous suggérons 5 extensions de cet algorithme. Nous introduisons également les paramètres de contrôle qui permettront de déclencher ou non l'application de certaines parties de l'algorithme modifiées.

5.1.3 Heuristique FSIC : extensions

Extension 1 - prise en compte des paramètres des opérations

La première extension concerne les données du problème. La version de base de l'heuristique FSIC a été développée pour le problème où les temps opératoires étaient connus. Dans notre cas, les opérations sont caractérisées par leurs paramètres, et leurs temps opératoires ne peuvent pas être calculés avant l'affectation. Cela entraîne un petit changement à l'étape de la construction de la liste CL. Rappelons que pour ajouter une opération i à la liste CL, les trois conditions suivantes sont à vérifier :

1. tous ses prédécesseurs doivent être déjà affectés,
2. son affectation au poste courant doit respecter les contraintes d'exclusion au niveau des postes de travail avec les opérations qui y sont déjà affectées,
3. et la contrainte sur le temps de cycle doit également être respectée après son affectation au block courant.

Pour notre formulation du problème d'optimisation, les deux premières conditions à vérifier demeurent les mêmes, par contre, la troisième condition doit tenir compte des paramètres des opérations et peut être formulée comme suit :
$t^p(N_m \setminus N_{mn_m}) + t^b(N_{mn_m} \cup \{i\}) \leq T_0$

Extension 2 - procédure de l'affectation de l'opération i

Dans la version de base, l'affectation de l'opération i commence par le bloc courant, pourtant il est possible qu'il existe un bloc antérieur au bloc courant qui puisse encore l'accueillir. La recherche du bloc le plus en amont pour l'affectation de l'opération i nécessite un nombre complémentaire de comparaisons et, par conséquent, le temps de la construction d'une solution admissible augmente. Pour trouver un compromis entre le gain en qualité et la perte en temps de résolution, nous introduisons la variable « check blocks » pouvant avoir les valeurs suivantes :

1. « check blocks » = 0 : l'affectation commence par le bloc courant ;
2. « check blocks » = 1 : l'affectation commence par le premier bloc du poste courant ;
3. « check blocks » = 2 : l'affectation commence par le premier bloc du premier poste.

Si l'option 2 ou 3 est choisie, l'algorithme essaie d'affecter l'opération i à des blocs précédant le bloc courant. Une telle affectation n'est admissible que si elle respecte toutes les contraintes pour le bloc et le poste de travail correspondants. S'il existe au moins une contrainte qui empêche l'affectation de l'opération i à un bloc, alors l'algorithme essaie de l'affecter l'opération i soit au bloc, soit au poste suivant, et ainsi de suite jusqu'à ce que soit l'opération i soit affectée, soit le bloc courant soit atteint. Dans le dernier cas, l'affectation de l'opération i se poursuit de manière habituelle.

Il est à noter que cette extension a déjà été suggérée dans [Finel, 2004]. Nous la présentons ici afin d'étudier son impact sur les performances de l'heuristique FSIC en combinaison avec les autres extensions que nous proposons.

Extension 3 - procédure de la construction de la liste *CL*

Dans la version de base de l'heuristique FSIC, une opération est ajoutée à la liste *CL*, si son affectation au poste courant n'enfreint pas la contrainte sur le temps de cycle. Néanmoins, si la valeur du paramètre « check blocks » = 2, c'est-à-dire que l'affectation de l'opération commence par le premier bloc du premier poste, alors il est possible que l'opération dont l'affectation au bloc courant enfreint la contrainte de temps de cycle, puisse être affectée à un des postes antérieurs. Cependant, non affectation d'une telle opération à un des postes précédant le poste courant impose la création d'un nouveau poste, car son affectation au poste courant est impossible. Nous introduisons le paramètre de contrôle « check time » dont l'utilisation permet de changer les conditions à vérifier lors de la construction de la liste *CL*. Ce paramètre prend les valeurs suivantes :

1. « check time »= 1, si la contrainte sur le temps de cycle pour le poste courant est vérifiée lors de la construction de la liste *CL*,
2. « check time »= 0, sinon.

Extension 4 - procédure de l'affectation des opérations de la liste *AL*

Dans la version de base de l'heuristique FSIC, la première tentative d'affectation de toutes les opérations de la liste *AL* étant échoue, toutes les affectations déjà effectuées à partir de cette liste sont annulées. Ensuite, un nouveau poste est ouvert et une nouvelle tentative d'affectation de toutes les opérations de la liste *AL* est effectuée. Pourtant il est possible que la liste *CL* contienne encore des opérations qui ne font pas partie de la liste *AL* mais qui peuvent être affectées au poste courant. Nous introduisons le paramètre de contrôle « 2 trials *AL* » dont l'utilisation permet de choisir la procédure de l'affectation des opérations de la liste *AL*. Ce paramètre prend les valeurs suivantes :

1. « 2 trials *AL* » = 1, si deux tentatives d'affectation des opérations de la liste *AL* sont effectuées (au poste courant et, ensuite, si l'échec, au poste suivant) ;
2. « 2 trials *AL* » = 0, si une seule tentative est effectuée, c'est-à-dire qu'un nouveau poste n'est pas créé à la suite d'un échec lors de l'affectation de la liste *AL* et l'affectation des opérations de la liste *CL* continue pour le poste courant.

Extension 5 - tri de la liste *AL*

La liste *AL* contient des opérations liées par une contrainte d'inclusion ainsi que leurs prédécesseurs non affectés. Dans la version de base de l'heuristique FSIC, lors de l'affectation de la liste *AL*, les opérations sont traitées dans l'ordre de leur inscription dans la liste. Nous suggérons de scinder la liste *AL* en deux parties : la première avec les opérations sans contrainte d'inclusion, c'est-à-dire les prédécesseurs non affectés, et la deuxième avec les opérations liées par une contrainte d'inclusion. De cette façon, les opérations de la première partie peuvent être traitées en priorité. S'il n'est pas possible de les affecter toutes au poste courant alors que le paramètre « 2 trials *AL* » = 1, nous pouvons ouvrir un nouveau poste sans annuler les affectations de la liste *AL* déjà effectuées. Nous introduisons le paramètre

de contrôle « divide *AL* » dont l'utilisation permet de choisir comment effectuer l'affectation des opérations sans contrainte d'inclusion. Ce paramètre peut prendre les valeurs suivantes :

1. « divide *AL* » = 1, si la liste *AL* est décomposée en deux parties selon la façon décrite ci-dessus,
2. « divide *AL* » = 0, sinon.

Direction de l'affectation des opérations

Dans la littérature, on trouve souvent des heuristiques développées pour la résolution du SALBP qui mettent en place différentes directions de l'affectation des opérations : directe, inversée, ou directe et inversée à la fois. Nous avons également testé ces différentes politiques d'affectation. Cependant, les tests effectués montrent que l'utilisation de l'affectation directe (comme présentée ci-dessus) donne toujours des meilleurs solutions. Ce résultat s'explique par les particularités du graphe de précédence dans le cadre de l'usinage.

5.1.4 Performances de l'algorithme FSIC sans et avec extensions

Les variantes de l'algorithme FSIC testées

Une variante de l'heuristique FSIC est définie par les valeurs des paramètres de contrôle introduits. Le Tableau 5.1 rappelle ces paramètres et les valeurs qu'ils peuvent prendre.

Tab. 5.1 – Paramètres de contrôle de l'heuristique FSIC

check blocks	check time	2 trials *AL*	divide *AL*
0	0	0	0
1	1	1	1
2	-	-	-

Par exemple, la version de base est désignée comme « 0110 », puisqu'elle a les valeurs des paramètres de contrôle suivantes : « check blocks »= 0, « check time »= 1, « 2 trials *AL* »= 1, « divide *AL* » = 0. Il est facile de voir qu'au total nous pouvons avoir $24 = 3 \cdot 2 \cdot 2 \cdot 2$ variantes de l'heuristique FSIC différentes.

Les jeux de données

Comme nous l'avons évoqué dans la Section 2.3 du Chapitre 2, la fabrication d'une pièce consiste à réaliser un ensemble d'entités. La définition des entités et leurs paramètres varient selon le contexte. Nous avons développée une taxinomie des entités utilisées dans le cadre d'usinage avec des boîtiers multibroches. Cette taxinomie est présentée dans le Chapitre 7. Chaque type d'entités est caractérisé par un ensemble d'opérations d'usinage diverses (fraisage, perçage, alésage, etc.) et par les contraintes entre elles. Afin d'étudier l'impact des extensions de l'heuristique FSIC sur la qualité des solutions finales, nous avons créé

un échantillon de problèmes générés aléatoirement, mais en utilisant des vraies entités lors de la génération des instances. De manière générale, plus la pièce à fabriquer comporte d'entités, plus la taille de l'ensemble **N** est grande [Guschinskaya et Dolgui, 2007*c*]. Alors, afin d'étudier l'impact du nombre d'opérations sur la qualité des solutions obtenues, nous avons créé 4 séries de tests qui se distinguent par le nombre d'entités utilisées lors de la génération des instances.

Chaque série comporte 50 instances distinctes qui ont été générées de manière aléatoire. Donc, 200 instances ont été testées au total. Lors de la génération de toute instance, le type de chaque entité et la face de la pièce à laquelle cette entité appartient ont été choisis au hasard. De cette façon, nous avons obtenu des instances de la même série, ayant le nombre d'opérations et l'ensemble de contraintes différents.

Le Tableau 5.2 fournit les paramètres des séries, à savoir : « Série » représente le numéro de la série, n_{ent} est le nombre d'entités utilisées lors de la génération des instances, $|N_{min}|, |N_{max}|, |N_{moy}|$ sont les valeurs minimum, maximum et moyennes du nombre d'opérations pour les instances de la série correspondante. Afin d'étudier l'impact du temps de calcul alloué sur la qualité des solutions obtenues, les instances de chaque série ont été d'abord résolues avec un temps très limité, désigné par T_{res}^{court}, et ensuite avec un temps 10 fois plus grand, désigné par T_{res}^{long}. Les valeurs de T_{res}^{court} et T_{res}^{long} varient d'une série à l'autre et sont également présentées dans le Tableau 5.2 (elles sont mesurées en secondes).

TAB. 5.2 – Paramètres des séries de tests pour l'évaluation de l'heuristique FSIC

Série	n_{ent}	$\|N_{min}\|$	$\|N_{max}\|$	$\|N_{moy}\|$	T_{res}^{court}	T_{res}^{long}
1	10	29	47	38	1,5	15
2	20	46	92	55	3	30
3	30	80	127	84	4,5	45
4	40	115	158	141	6	60

L'analyse des résultats

Les problèmes de chaque série ont été résolus par les 24 variantes de l'algorithme FSIC. Les résultats fournis par chaque méthode ont été comparés afin de trouver la meilleure solution obtenue. Dans le Tableau 5.3, *NMS* et *PMS* représentent le Nombre et le Pourcentage des Meilleures Solutions obtenues par chaque méthode. Les calculs ont été effectués sur un PC Pentium IV, 3GHz et 512 Mo de RAM.

Nous pouvons constater que si « check blocks » = 2, alors les méthodes ayant « 2 trials *AL*» = 0 sont plus performantes que celles ayant « 2 trials *AL* » = 1. Cependant, pour les méthodes ayant « check blocks » = 0 ou 1 nous pouvons observer le contraire : les méthodes ayant « 2 trials *AL* » = 1 surpassent celles ayant « 2 trials *AL* » = 0.

En analysant les résultats des tests, nous pouvons sélectionner les meilleures méthodes, qui sont 2011 (c'est-à-dire « check blocks » = 2, « check time » = 0, « 2 trials *AL* » = 1, « divide *AL* » = 1), puis 2010 et 2001.

TAB. 5.3 – Résultats de l'évaluation de l'heuristique FSIC

Méthode	Total		T_{res}^{court}	T_{res}^{long}
	NMS	*PMS* %	*PMS* %	*PMS* %
0000	32	16,0	11	21
0001	32	16,0	10	22
0010	35	17,5	12	23
0011	36	18,0	13	23
0100	52	26,0	21	31
0101	55	27,5	22	33
0110	62	31,0	27	35
0111	61	30,5	26	35
1000	29	14,5	8	21
1001	29	14,5	8	21
1010	28	14,0	8	20
1011	30	15	7	23
1100	50	25	18	32
1101	49	24,5	17	32
1110	53	26,5	20	33
1111	55	27,5	22	33
2000	89	44,5	38	51
2001	96	48,0	44	52
2010	106	53,0	52	54
2011	113	56,5	54	59
2100	67	33,5	31	36
2102	72	36,0	34	38
2110	70	35	27	43
2111	71	35,5	27	44

Il est à noter également que pour chaque méthode testée, le nombre de meilleures solutions trouvées avec le temps de résolution court T_{res}^{court} est toujours inférieur à la valeur de ce paramètre pour le temps de résolution prolongé T_{res}^{long}. Le classement des méthodes capables de trouver le maximum de meilleures solutions ne change pas.

L'étude de l'impact du nombre d'opérations (ou nombre d'entités) sur la qualité des solutions obtenues par chaque variante de l'heuristique FSIC a été présentée dans [Guschinskaya et Dolgui, 2007*c*]. Nous avons pu constater qu'avec l'augmentation du nombre d'entités, les méthodes ayant « check blocks » = 2 se démarquent.

5.1.5 Conclusion

Nous pouvons conclure que la plus performante variante de l'heuristique FSIC, désignée par les résultats des expérimentations, est celle qui utilise toutes les extensions proposées. Cette variante de l'algorithme FSIC surpasse largement la version de l'heuristique de base :

136 meilleures solutions ont été obtenues en l'utilisant contre seulement 75 par l'heuristique de base. Cependant, malgré les améliorations observées, nous avons pu constater qu'aucune variante de l'heuristique FSIC n'a pas été capable de fournir la meilleure solution pour tous les problèmes testés.

L'heuristique FSIC s'appuie sur l'approche Monte-Carlo lors du choix de l'opération à affecter. Alors, la probabilité de trouver la solution optimale tend vers 1 seulement si le nombre d'itérations tend vers l'infini. Le développement d'une approche intégrant une phase d'amélioration des solutions obtenues par l'heuristique FSIC fait l'objet de la section suivante.

5.2 Approche mixte

L'approche que nous proposons permet d'intégrer des méthodes exactes et heuristiques dans un algorithme commun. Nous commençons la présentation de cette approche par son schéma général.

5.2.1 Schéma général de l'approche

Cette approche a été mise en œuvre afin d'améliorer les solutions obtenues par une heuristique, en l'occurrence l'heuristique FSIC. Dans un premier temps, l'heuristique est utilisée afin de trouver une solution admissible du problème initial qui consiste en une affectation possible de toutes les opérations de l'ensemble $\mathbf{N}$ aux m_{heur} postes de travail. Cette solution peut être représentée comme une séquence S_{heur}, où chaque élément correspond à un poste de travail. Bien évidemment, toutes les contraintes de précédence et de compatibilité sont respectées pour chaque élément et entre eux. Alors nous utilisons cette solution pour obtenir la décomposition de l'ensemble $\mathbf{N}$ en sous-ensembles indépendants, soit $S_{heur} = \{N^1, ..., N^u, ..., N^w\}$. Ceci est mis en œuvre par un découpage de la séquence S_{heur} en w sous-séquences de façon à respecter les conditions suivantes :

- chaque sous-ensemble N^u contient les opérations affectées à un nombre entier de postes de travail successifs, c'est-à-dire $N^u = \bigcup_{k=k_1}^{k_2} N_k, 1 \leq k_1 \leq k_2 \leq m_{heur}$;
- toutes les opérations de chaque poste de travail sont dans le même sous-ensemble, c'est-à-dire si $N_k \subset N^u$, alors $N_k \not\subset N^i, i \in \{1, ..., w\} \backslash u$;
- l'union de tous les sous-ensembles N^u donne l'ensemble $\mathbf{N}$, c'est-à-dire $\bigcup_{u=1}^{w} N^u = \mathbf{N}$.

Ensuite, à partir de chaque sous-ensemble N^u, obtenu par le découpage de la séquence S_{heur}, nous formons un nouveau problème, dit sous-problème SP_u, qui est de même type que le problème initial mais de taille plus petite. Nous avons élaboré deux techniques de formation de ces sous-problèmes :

- sans intersection (indépendante) : chaque sous-problème peut être résolu indépendamment des autres (les sous-problèmes n'ont pas d'opération en commun), la solution du problème initial est alors formée par la concaténation des solutions des sous-problèmes ;

- avec agrégation (agrégée) : les opérations qui étaient dans les sous-problèmes précédents sont également présentes dans le sous-problème courant, mais sous forme de macro-opérations. La solution du problème initial est alors le résultat de la résolution du dernier sous-problème.

Quelle que soit la technique de décomposition utilisée, chaque sous-problème est résolu par une méthode exacte. Cela nous permet de trouver une solution optimale de chaque sous-problème SP_u. Bien évidement, le coût de cette solution $C_{SP_u} \leq C_u^{heur}$, où C_u^{heur} est le coût de la sous-séquence u dans la solution S_{heur}. La technique d'obtention de la solution finale à partir des solutions partielles dépend de la technique de formation des sous-problèmes, voir la Section 5.2.4.

Après avoir trouvé une solution pour le problème initial suite à l'application de la méthode exacte à tous les sous-problèmes, nous admettons qu'une itération de l'algorithme est terminée. Pour commencer une nouvelle itération, nous revenons à l'heuristique afin de trouver une nouvelle solution approchée qui ensuite sera décomposée et ainsi de suite. L'Algorithme 5.3 décrit le schéma global de l'approche.

Algorithme 5.3 : Schéma général de l'approche mixte

```
1  Initialiser C_min = ∞, TR_tot = 0, TR_nimp = 0
   répéter
2      Choisir une solution heuristique S_heur avec son coût C_heur
3      si C_heur < C_min alors
           C_min = C_heur, S_min = S_heur
       fin
4      Initialiser k = 1, u = 1
       tant que k < m_heur faire
5          Déterminer N^u et k_u
6          Générer un nouveau sous-problème SP_u à partir de N^u
           k = k + k_u, u = u + 1
7          Résoudre le sous-problème SP_u par une méthode exacte
       fin
8      Former la solution finale S_mix avec son coût C_mix à partir de u sous-problèmes
       résolus
       TR_tot = TR_tot + 1
9      si C_mix < C_min alors
           C_min = C_mix, S_min = S_mix, TR_nimp = 0
       sinon
           Incrémenter le compteur TR_nimp = TR_nimp + 1
       fin
   jusqu'à T_cur > T_res || TR_nimp > TR_nimp^aut || TR_tot > TR_tot^aut || C_min < C_stop
```

Notons que l'heuristique que nous utilisons donne, de manière générale, une solution différente à chaque appel, car elle fait des choix aléatoires lors de la construction d'une solution. La nature stochastique de la décomposition elle-même participe également à la génération des solutions différentes à chaque itération de l'algorithme.

Étant donné que le temps alloué est limité, nous avons introduit un critère permettant de décider si une solution heuristique est utilisée pour la décomposition ou s'il faut l'abandonner et chercher une autre solution heuristique (en lançant de nouveau l'algorithme heuristique). Quelques propositions sur la façon d'analyser une solution heuristique afin de savoir si nous pouvons la retenir ou non, c'est-à-dire s'il y a une chance ou non d'obtenir une meilleure solution finale après la phase de la décomposition, sont présentées dans la Section 5.2.2.

Quand le temps de résolution alloué est atteint ($T_{cur} > T_{res}$, où T_{cur} est le temps consommé pour la résolution, T_{res} est le temps alloué), nous choisissons la meilleure solution obtenue. Éventuellement, d'autres conditions d'arrêt peuvent être ajoutées, telles que l'obtention d'une solution avec un coût C_{min} inférieur à un seuil donné C_{stop} ou le dépassement des valeurs maximales autorisées ($TR_{tot^{aut}}$ et $TR_{nimp^{aut}}$) par le nombre d'itérations effectuées TR_{tot} ou par le nombre d'itérations sans amélioration TR_{nimp}.

5.2.2 Sélection d'une solution de départ

Comme nous l'avons mentionné précédemment, une heuristique quelconque peut être utilisée à l'étape de résolution approchée. En l'occurrence, nous employons l'heuristique FSIC, dont les performances ont été étudiées dans la Section 5.1. Les choix aléatoires utilisés par cette heuristique font qu'à chaque lancement, elle fournisse des solutions de différentes qualités. Par conséquent, nous pouvons disposer d'une solution de départ de mauvaise qualité à une des itérations. Même si dans un cas théorique, il est possible d'obtenir une solution de bonne qualité à partir d'une mauvaise solution, cela est très peu probable en utilisant la technique d'amélioration des solutions que nous avons mise en place. En effet, le nombre de réaffectations possibles pour chaque opération sera limité par le sous-ensemble auquel elle appartient. Dans ces conditions, l'optimum global ne peut pas être atteint à partir de n'importe quelle solution de départ. Le gain possible pour la fonction objectif est encore plus limité à cause de nombreuses contraintes et parce que beaucoup de solutions voisines auront la même valeur de la fonction objectif (à cause des permutations possibles), voir [Guschinskaya *et al.*, 2005]. Étant donné qu'il n'est pas envisageable d'exploiter toutes les solutions de départ fournies par l'heuristique à cause du temps de calcul limité, nous avons mis en œuvre deux techniques de sélection des solutions de départ pour ne garder que des solutions prometteuses.

La première compare le coût C_{heur} de chaque solution heuristique obtenue avec le coût de la meilleure solution heuristique connue C_{heur_min}. Si la différence ne dépasse pas un écart autorisé δ, une telle solution heuristique est acceptée comme une solution de départ, sinon elle est rejetée. La deuxième technique lance l'heuristique pendant un certain temps T_{heur} suffisant pour effectuer NIT_{heur} itérations, ensuite la meilleure solution obtenue est comparée avec C_{heur_min} comme dans le premier cas. Alors, l'étape 2 de l'Algorithme 5.3 est donnée par l'Algorithme 5.4. Cet algorithme utilise soit l'étape 1' pour la réalisation de la première technique, soit l'étape 1" pour la deuxième. La deuxième technique doit normalement fournir des solutions de départ avec un coût plus petit, en contrepartie elle consomme plus de temps à cette étape, cela diminue le temps restant pour la résolution des sous-problèmes de manière exacte.

Algorithme 5.4 : Sélection d'une solution heuristique

1' **tant que** *Une solution admissible S_{heur} n'a pas été trouvée* **faire**
  | Générer une nouvelle solution heuristique
  fin
1" **tant que** *Le temps de calcul* $< T_{heur}$ **faire**
  | Générer des solutions heuristiques, mais pas plus de NIT_{heur}
  | La solution S_{heur} avec son coût C_{heur} est la meilleure solution parmi celles qui sont obtenues dans cette boucle
  fin
  si $C_{heur} \geq (1+\delta)C_{heur\_min}$ **alors**
  | Aller à l'étape 1' ou 1"
  fin
  si $C_{heur} < C_{heur\_min}$ **alors**
  | $C_{heur\_min} = C_{heur}, S_{heur\_min} = S_{heur}$
  fin
  Aller à l'étape 3 de l'Algorithme 5.3

5.2.3 Décomposition d'une solution heuristique

L'augmentation exponentielle du temps de calcul pour les méthodes exactes avec l'augmentation de la taille du problème soulève la question suivante : comment découper le problème initial en sous-problèmes pour que la méthode exacte utilisée soit capable de tous les résoudre dans le temps alloué. Nous avons deux cas extrêmes à éviter. De manière générale, plus les sous-ensembles sont petits, plus il est facile d'avoir un temps de calcul raisonnable, mais en même temps nous avons moins de chances d'améliorer la solution de départ et de s'approcher de l'optimum global. Mais si la taille du sous-problème est trop importante, nous risquons de ne jamais pouvoir le résoudre de manière optimale sans dépasser le temps alloué. Puisque la taille du problème est déterminée par la cardinalité de l'ensemble N^u correspondant, nous utilisons le paramètre $MaxOp$ afin de pouvoir contrôler le nombre d'opérations d'un sous-problème. Notons que nous ne pouvons toujours garantir qu'aucun sous-problème ne dépasse $MaxOp$, car si le nombre d'opérations affectées à un poste de travail est supérieur à $MaxOp$, nous sommes obligés de les prendre toutes.

Comme nous ne disposons pas de critère permettant de déterminer a priori le découpage en sous-séquences qu'il faut utiliser pour avoir plus de gain, nous l'effectuons de manière aléatoire. Plus précisément, pour déterminer le nombre de postes k_u dont les opérations constitueront un sous-problème, nous choisissons au hasard un nombre entier dans l'intervalle $[1 \,..\, MaxSt]$, où le paramètre $MaxSt$, comme $MaxOp$, est utilisé pour limiter la taille des sous-problèmes : il donne le nombre maximal de postes qui peuvent être inclus dans un sous-problème.

Le choix des paramètres $MaxOp$ et $MaxSt$ doit être fait à l'étape d'initialisation de l'algorithme en fonction des paramètres suivants : la taille du problème initial, le temps de calcul alloué, la technique de formation de sous-problèmes et la méthode exacte utilisée pour leur résolution. Si lors de l'exécution de l'algorithme, ces deux paramètres sont en contradiction, c'est le paramètre $MaxOp$ qui domine. Par exemple, si $MaxSt = 4$, $MaxOp$

$= 30$, $|N_1| = 10$, $|N_2| = 20$, $|N_3| = 15$, $|N_4| = 20$, $k_1 = random[1..4] = 3$, alors, $k_1 = 2$.

L'étape 5 de l'Algorithme 5.3 est présentée en détail par l'Algorithme 5.5.

Algorithme 5.5 : Formation d'une sous-séquence

```
Initialiser k_u = random[1..MaxSt]
si (k_u + k > m_heur) alors
|  k_u = m_heur − k
fin
N^u = N_k, i = 1.
tant que i ≤ k_u faire
|  si |N^u| + |N_{k+i}| < MaxOp alors
|  |  N^u = N^u ∪ N_{k+i}
|  sinon
|  |  stop
|  fin
|  i = i + 1
fin
k_u = i − 1
```

Le caractère stochastique de la détermination de la taille de chaque sous-séquence permet d'obtenir différents ensembles de sous-séquences en utilisant les même valeurs des paramètres $MaxOp$ et $MaxSt$. Ainsi il est possible d'effectuer plusieurs découpages d'une seule solution de départ et, en conséquence, de résoudre différents sous-problèmes en vue de mieux exploiter le voisinage d'une solution. Néanmoins, les tests préliminaires ont montré que le même effet est atteint grâce à la répétition des solutions heuristiques de départ. De plus, en tandem avec la technique de sélection des solutions de départ, un seul découpage d'une solution permettait d'atteindre des solutions finales de meilleure qualité, car dans ce cas, les mauvaises solutions de départ n'ont été traitées qu'une seule fois. Pour cette raison, le paramètre déterminant le nombre d'essais pour le découpage d'une solution heuristique a été finalement abandonné et l'algorithme effectue un seul découpage de chaque solution heuristique.

5.2.4 Formation de sous-problèmes

Comme cela a été mentionné dans la Section 5.2.1, nous avons mis en œuvre deux techniques de formation de sous-problèmes, à savoir : indépendante et agrégée. Nous détaillons ici leurs principes.

La technique indépendante repose sur la résolution indépendante de sous-problèmes. Ceci est possible du fait que les sous-séquences obtenues après le découpage de la solution heuristique ne contiennent pas d'opérations communes et que les contraintes de précédence et d'incompabilité sont respectées entre les sous-ensembles d'opérations N^1, ..., N^u, ..., N^w. Le sous-problème SP_u consiste ainsi à affecter l'ensemble d'opérations N^u en prenant en compte les contraintes (2.2)-(2.10) modifiées. La modification consiste à remplacer l'ensemble $\mathbf{N}$ par l'ensemble N^u et à supprimer les contraintes contenant des opérations de l'ensemble $\mathbf{N}\backslash N^u$. Nous pouvons également remplacer m_0 pour SP_u par $\lfloor C_u^{heur}/(C_1 + C_2) \rfloor$ où C_u^{heur} est le coût de la sous-séquence u dans la solution heuristique S_{heur} (puisque la solution optimale ne peut

contenir plus de postes de travail qu'une solution heuristique). La solution finale du problème initial est obtenue en remplaçant les sous-séquences de S_{heur} par les solutions exactes des sous-problèmes. Les algorithmes de formation des solutions finales à partir des solutions des sous-problèmes sont présentés dans la section 5.2.5. Le schéma de la technique indépendante de formation des sous-problèmes est présenté dans la Figure 5.1.

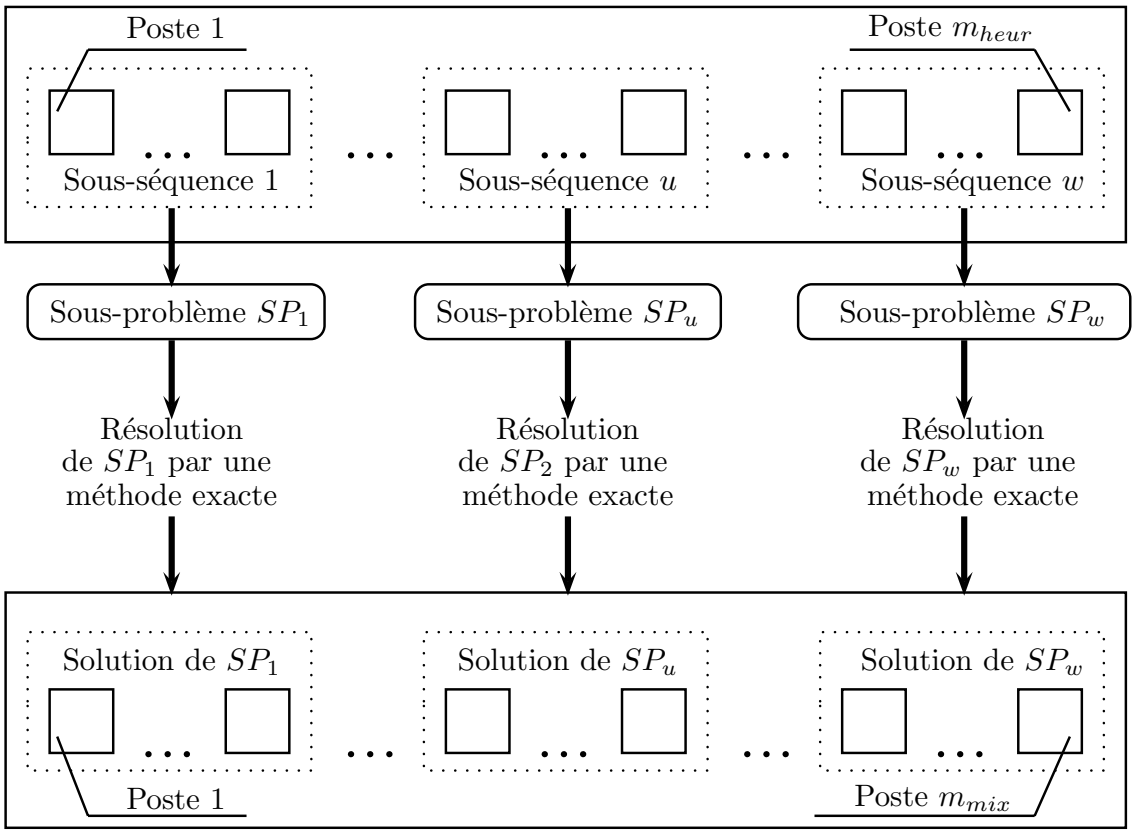


FIG. 5.1 – Formation des sous-problèmes : technique indépendante

La technique agrégée quant à elle, construit une solution finale en augmentant progressivement la taille du sous-problème de telle façon que les opérations qui ont données lieu aux sous-problèmes antérieurs soient également inclues dans le sous-problème courant mais en tant que macro-opérations. Ainsi, la solution du problème initial est le résultat de la résolution du dernier sous-problème.

Afin d'éviter l'explosion du temps de calcul, nous avons besoin des techniques d'agrégation des solutions des sous-problèmes. Nous avons élaboré une technique dont la réalisation est relativement simple : après avoir résolu le sous-problème SP_{u-1}, il suffit de remplacer les blocs obtenus dans la solution S_{u-1} par des macro-opérations. Avec cette stratégie, nous tenons compte de la possibilité d'ajouter dans les postes de travail et blocs déjà construits les opérations qui n'étaient pas traitées au moment de la formation de ces blocs et postes,

ainsi que la possibilité d'assigner les blocs existants aux autres postes de travail que ceux auxquels ils ont été assignés auparavant. Le schéma de cette technique est présenté dans la Figure 5.2.

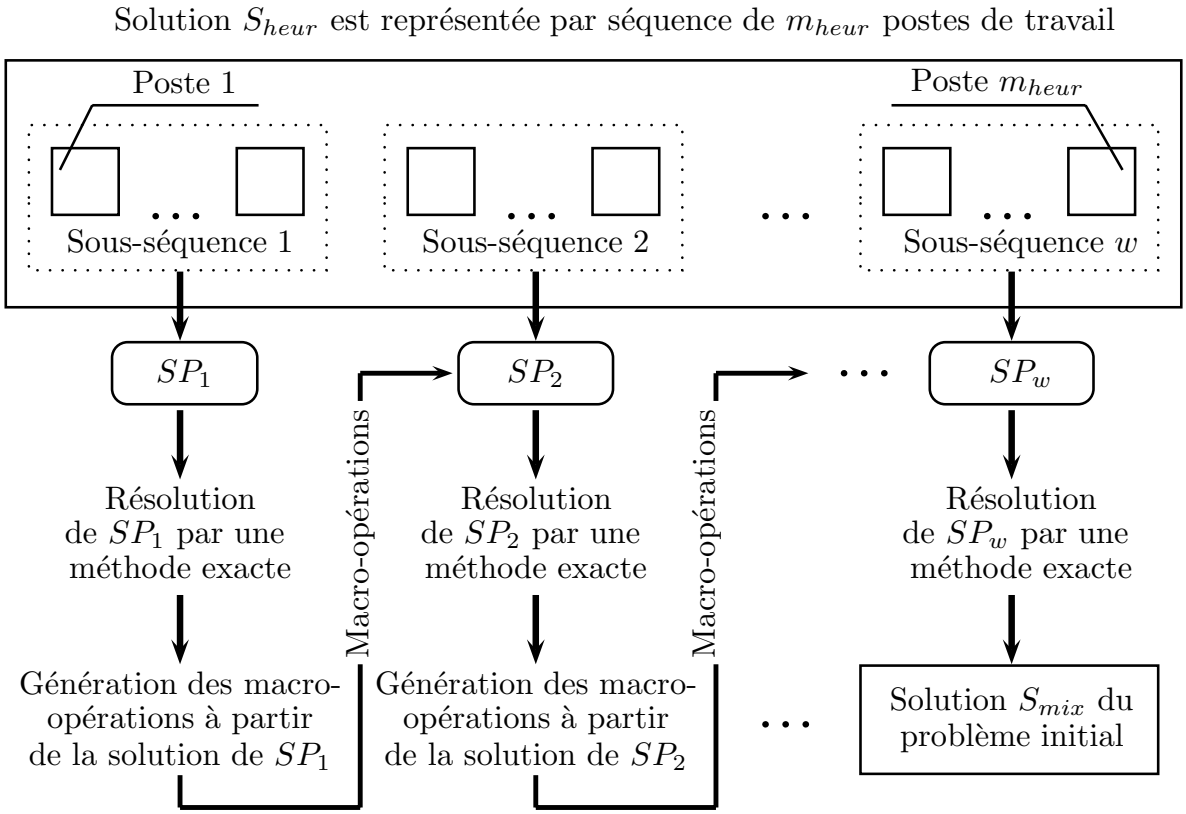


FIG. 5.2 – Formation des sous-problèmes : technique agrégée

Nous remplaçons chaque ensemble d'opérations N_{kr} affecté au r-éme bloc du poste k dans la solution S_{u-1} par une seule macro-opération MO ayant le numéro $num(MO)$. Les paramètres $l_{num(MO)}$, $v_{f1}(num(MO))$, $v_{f2}(num(MO))$ de cette macro-opération sont calculés à partir des valeurs individuelles des opérations de l'ensemble N_{kr}. Après avoir créé toutes les macro-opérations, nous ajustons les contraintes du problème initial de la manière suivante : si N_{kr} est un élément de I^p, alors cet élément peut être supprimé, car la contrainte correspondante est déjà respectée ; autrement, s'il existe des opérations appartenant à la fois à l'ensemble N_{kr} et à des ensembles des collections E^b, E^p ou I^p (mais N_{kr} n'est pas un élément entier de I^p), alors nous les remplaçons par les numéros des macro-opérations correspondantes. La formation des macro-opérations à partir de la solution S_{u-1} du sous-problème SP_{u-1} est récapitulée dans l'Algorithme 5.6.

Les macro-opérations sont ajoutées dans l'ensemble N^u et ensuite le sous-problème SP_u est formé. Le paramètre m_0 prend une nouvelle valeur $m_0 = \lfloor (C_{u-1} + C_u^{heur})/(C_1 + C_2) \rfloor$, où C_{u-1} est le coût de la solution du sous-problème SP_{u-1}. Notons que nous devons garder les informations concernant la composition des macro-opérations, car pour reconstituer la solution du problème initial après avoir obtenu la solution du dernier sous-problème, il faut

remplacer les macro-opérations dans cette solution par les opérations d'origine.

Algorithme 5.6 : Formation des macro-opérations

Initialiser $k = 1, r = 1$
tant que $k \leq m^{u-1}$ **faire**
  tant que $r \leq r_k$ **faire**
    $l_{num(MO)} = \max\{l_i \mid i \in N_{kr}\}$,
    $v_{f1}(num(MO)) = \max\{v_{f1}(i) \mid i \in N_{kr}\}$,
    $v_{f2}(num(MO)) = \min\{v_{f2}(i) \mid i \in N_{kr}\}$,
    $PredD(num(MO)) = \bigcup_{i \in N_{kr}} PredD(i)$
    pour tous les $i \in \mathbf{N} \backslash N_{kr}$ **faire**
      $Cur = PredD(i) \cap N_{kr}$
      si $Cur \neq \emptyset$ **alors**
        $PredD(i) = (PredD(i) \backslash Cur) \cup num(MO)$
      fin
    fin
    pour $E = I^p$, $E = E^b$ *et* $E = E^p$ **faire**
      pour tous les $e \in E$ **faire**
        $Cur = e \cap N_{kr}$
        si $Cur \neq \emptyset$ **alors**
          si $E = I^p$ ***et*** $Cur = N_{kr}$ **alors**
            $E = E \backslash e$
          sinon
            $e = (e \backslash Cur) \cup num(MO)$
          fin
        fin
      fin
    fin
    $\mathbf{N} = \mathbf{N} \backslash N_{kr} \cup num(MO)$
    $r = r + 1$
  fin
  $k = k + 1$
fin

5.2.5 Résolution des sous-problèmes

Suite au découpage aléatoire du problème initial, nous pouvons nous retrouver face à deux situations problématiques lors de la résolution d'un sous-problème :

1. Malgré l'utilisation des paramètres de contrôle de la taille, le sous-problème formé peut s'avérer « trop lourd » pour la méthode exacte utilisée à cause de sa structure particulière. Si nous tentons de résoudre un tel sous-problème, nous risquerons de consommer tout le temps alloué rien que pour ce sous-problème et ne pas laisser le temps pour les autres sous-problèmes. Notons que le terme « trop lourd » dépend de la méthode exacte utilisée.

2. Si la taille d'un sous-problème est « trop petite », alors, la solution heuristique partielle a une forte chance d'être optimale, et par conséquent, la résolution de ce sous-problème de manière exacte ne pourra pas l'améliorer. Par expérience, nous qualifions la taille d'un sous-problème comme « trop petite » si le sous-problème contient un seul poste de travail avec un nombre de blocs inférieur à 3, ou deux postes de travail avec un nombre de blocs inférieur à 2 chacun.

Alors, si à l'étape de résolution d'un sous-problème SP_u, nous nous apercevons que sa taille est « trop petite » ou le problème est « trop lourd », afin d'éviter les pertes du temps, nous n'appliquons pas la méthode exacte, mais nous copions simplement la sous-séquence u dans la solution S_{mix}.

De plus, afin d'éviter la résolution des sous-problèmes pour lesquels les solutions optimales sont déjà connues, nous calculons la borne inférieure du coût pour chaque sous-problème. Si jamais le coût de la solution heuristique est égal à la borne inférieure, alors il est inutile de résoudre ce sous-problème.

Comme nous l'avons déjà évoqué, nous utilisons deux méthodes exactes pour la résolution des sous-problèmes. La première méthode (voir la Section 4.3) consiste à modéliser le problème sous forme d'un programme linéaire en variables mixtes pour qu'il se prête à la résolution par le biais d'un logiciel d'optimisation. La deuxième méthode, que nous allons présenter ci-dessous, consiste en la transformation du problème initial en un problème de recherche du plus court chemin dans un graphe spécialement construit.

Approche par graphe

Cette méthode a été proposée dans [Dolgui *et al.*, 2008]. Soit $\mathbf{S}$ l'ensemble de collections $S = \{\{N_{11},...,N_{1r_1}\}, ..., \{N_{m1}, ..., N_{mr_m}\}\}$, satisfaisant les contraintes (2.2)-(2.10). Soit l'ensemble d'opérations v_{kr} comporte les opérations affectées aux $k-1$ postes et aux r premiers blocs du poste k :

$$v_{kr} = \bigcup_{n=1}^{k-1} \bigcup_{q=1}^{r_n} N_{nq} \cup \bigcup_{q=1}^{r} N_{kq}$$

Cet ensemble peut être considéré comme l'état de la pièce après son usinage sur les $k-1$ premières postes et par r boîtiers de la k-ème poste.

Soit V l'ensemble de tous les états possibles v_{kr} de la pièce de toutes les solutions possibles $S \in \mathbf{S}$, y compris l'état initial (brut) $v_0 = \emptyset$ et l'état final (produit fini) $v_{\mathbf{N}} = \mathbf{N}$.

Pour chaque état $v \in V$ nous définissons un paramètre $\Gamma(v) = 1$ si pour tous les $e \in I^p$ nous avons $e \cap v = \emptyset$ ou $e \subseteq v$, sinon $\Gamma(v) = 0$. La valeur 1 signifie qu'il n'y pas d'opération dans v liée avec les opérations non appartenant à v par une contrainte I^p. Ceci signifie que nous pouvons fermer ce poste dans cet état sans violer les contraintes d'inclusion. Nous construisons un graphe orienté $H = (V, F)$, dans lequel un arc (v', v'') représente l'ensemble des opérations $N'' = v''\backslash v'$ réalisé dans un bloc. Un arc $(v', v'') \in F$ si et seulement si $v' \subset v''$, et pour tous les $e \in E^b \cup E^p$, $e \not\subset v''\backslash v'$. À chaque arc, un temps $t(v', v'') = t^b(v''\backslash v')$, égal au temps d'usinage de l'ensemble N'' dans un bloc, est associé.

À chaque solution $S \in \mathbf{S}$, nous associons le chemin correspondant $x(S) = ((u_0 = v_0, ..., u_{j-1}, u_j, ..., u_{l(x)} = v_{\mathbf{N}}), (\gamma_0 = 1, ..., \gamma_{j-1}, \gamma_j, ..., \gamma_{l(x)} = 1))$ du nœud v_0 au nœud $v_{\mathbf{N}}$

dans le graphe H. Le paramètre γ_j est égal à 1 si le bloc $u_j \backslash u_{j-1}$ est le dernier bloc d'un poste dans la solution S. Soit $\mathbf{X}$ l'ensemble de tous les chemins $x = ((v_0 = u_0, ..., u_{j-1}, u_j, ..., u_{l(x)} = v_\mathbf{N}), (\gamma_0 = 1, ..., \gamma_{j-1}, \gamma_j, ..., \gamma_{l(x)} = 1))$ de v_0 à $v_\mathbf{N}$ dans le graphe H où γ_j est égal à 0 si $\Gamma(u_j) = 0$ et 0 ou 1 sinon. Dans chaque chemin $x \in \mathbf{X}$, il y a une séquence $j_0 = 0, j_1, j_2, ..., j_{m(x)} = l(x)$ des indices j_n avec $\gamma_{j_n} = 1$, $n = 0, ..., m(x)$. Alors un chemin $x \in \mathbf{X}$ correspond à une solution $S(x) = \{\{u_1 \backslash u_0, ..., u_{j_1} \backslash u_{j_1 - 1}\}, \{u_{j_1+1} \backslash u_{j_1}, ..., u_{j_2} \backslash u_{j_2-1}\}, ..., \{u_{j_{m(x)-1}+1} \backslash u_{j_{m(x)-1}}, ..., u_{j_{m(x)}} \backslash u_{j_{m(x)}-1}\}\}$ qui satisfait les contraintes (2.2)-(2.7), mais peut ne pas satisfaire les contraintes (2.8)-(2.10). Alors, le problème initial (2.1)-(2.10) peut être transformé en problème de la recherche du plus court chemin sous contraintes :

$$\text{Minimiser } C(x) = C_1 m(x) + C_2 l(x) \tag{5.1}$$

$$S.\ c.\ :$$

$$x \in \mathbf{X} \tag{5.2}$$

$$\sum_{i=j_{k-1}+1}^{j_k} t^b(u_i \backslash u_{i-1}) \leq T_0 - \tau^p, k = 1, ..., m(x) \tag{5.3}$$

$$e \not\subset (u_{j_k} \backslash u_{j_{k-1}}), e \in E^p, k = 1, ..., m(x) \tag{5.4}$$

$$j_k - j_{k-1} \leq n_0, k = 1, ..., m(x) \tag{5.5}$$

$$m(x) \leq m_0 \tag{5.6}$$

Formation de la solution finale pour la technique de décomposition indépendante

Lorsque la technique de décomposition indépendante est utilisée, il convient de compiler la solution finale à partir des solutions des sous-problèmes. Si l'approche par graphe est utilisée, alors il suffit de connecter les chemins x^u obtenus après la résolution des sous-problèmes $SP_u, u = 1, 2, ..., w$, comme le montre l'Algorithme 5.7 :

Algorithme 5.7 : Construction de la solution finale à partir des solutions partielles obtenues à l'aide de l'approche par graphe

```
Initialiser u = 1, C_mix = 0, j = 0
tant que u ≤ w faire
    j_cur = 0
    tant que j_cur < l(x^u) faire
        u_j = u_{j_cur}, γ_j = γ_{j_cur}
        j_cur = j_cur + 1, j = j + 1
    fin
    C_mix = C_mix + C^u_opt ; u = u + 1
fin
```

Si l'approche par programmation linéaire en variables mixtes (MIP) est utilisée, la formation de la solution finale à partir des solutions partielles, obtenues en utilisant le modèle (4.4)-(4.7), est présentée dans l'algorithme 5.8.

Algorithme 5.8 : Construction de la solution finale à partir des solutions partielles obtenues en utilisant l'approche MIP

Initialiser $u = 1; k = 0; C_{mix} = 0; \forall i \in \mathbf{N},\ \forall q \in Q(i) : X_{iq} = 0$
tant que $u \leq w$ **faire**
  $q_{cur} = 1$
  tant que $q_{cur} \leq q_0^u$ **faire**
    si $Y^u_{q_{cur}} = 1$ **alors**
      $q = kn_0 + q_{cur}$
      pour tous les $i \in N^u$ **faire**
        si $q_{cur} \in Q(i)$ **alors**
          $X_{iq} = X^u_{iq_{cur}}$
        fin
      fin
      $F_q = F^u_{q_{cur}}$
    fin
    $q_{cur} = q_{cur} + 1$
  fin
  $k = \lceil q/n_0 \rceil$; $C_{mix} = C_{mix} + C^u_{opt}$; $u = u + 1$
fin

5.2.6 Les variantes de l'algorithme mixte testées

Notre but est maintenant de tester les différentes variantes de l'algorithme mixte en le comparant avec l'heuristique FSIC utilisée individuellement. Une variante de l'algorithme mixte est définie par le choix des paramètres suivants :

- l'heuristique utilisée pour la construction des solutions de départ ;
- la technique de sélection d'une solution de départ ;
- les valeurs des paramètres $MaxOp$ et $MaxSt$ pilotant la taille des sous-problèmes ;
- la technique de formation des sous-problèmes ;
- la méthode pour la résolution des sous-problèmes.

Dans les expérimentations effectuées, toutes les variantes de l'approche mixte testées utilisent la même variante de l'heuristique FSIC, celle qui a été désignée comme la plus performante dans la section précédente. Cette variante a les paramètres suivants : « check blocks » $= 2$, « check time » $= 0$, « 2 trials AL » $= 1$, « divide AL » $= 1$ (pour plus de détails voir la Section 5.1). Les paramètres $MaxOp$ et $MaxSt$ ont également les mêmes valeurs pour toutes les variantes de l'approche mixte testées, à savoir : $MaxOp = 30$, $MaxSt = 3$. Le choix de ces valeurs a été fait à la base des résultats obtenus dans [Guschinskaya *et al.*, 2008*a*] qui ont montré leur préférence.

Les tests réalisés permettent alors d'étudier l'impact d'autres paramètres (présentés dans le Tableau 5.4) sur les performances de l'approche mixte, à savoir : de la technique de sélection d'une solution de départ (« δ » si seul le paramètre δ est utilisé ou « $\delta\&T$ » si les deux paramètres δ et T_{heur} sont utilisés), de la technique de formation de sous-problèmes (agrégée « A » ou indépendante « I ») et de la méthode exacte utilisée (programmation

linéaire en variables mixtes « MIP » ou approche par graphe « G »).

Tab. 5.4 – Paramètres de contrôle pour l'approche mixte

Paramètre	Choix 1	Choix 2
Technique de sélection d'une solution de départ	δ	$\delta\&T$
Technique de formation de sous-problèmes	I	A
Méthode pour la résolution des sous-problèmes	G	MIP

Pour présenter un jeu de paramètres, nous utilisons une notation constituée de trois champs sous forme x| y| z. Par exemple, la variante δ| I| GR fait appel à la technique de sélection d'une solution de départ utilisant le paramètre δ et à la technique indépendante de formation des sous-problèmes qui sont résolus moyennant l'approche par graphe.

Les jeux de données

Nous reprenons les jeux de données déjà utilisés dans la Section 5.1.4 pour l'évaluation des performances de l'heuristique FSIC. Rappelons qu'il s'agit d'un échantillon des problèmes générés aléatoirement, mais en tenant compte des caractéristiques et de la structure des problèmes industriels. Cet échantillon comporte 4 séries de problèmes, chacune avec 50 instances distinctes.

5.2.7 Résultats numériques

Comme auparavant, les calculs ont été effectués sur un PC Pentium IV, 3GHz avec 512 Mo de RAM. Tous les exemples ont été résolus par les 6 variantes de l'algorithme mixte et par l'heuristique FSIC. Les résultats fournis par ces méthodes ont été comparés afin de trouver la meilleure solution pour chaque problème. Ensuite, les écarts par rapport au meilleur résultat ont été calculés, ils sont représentés dans le Tableau 5.5 où Δ_{min}, Δ_{moy} et Δ_{max} sont respectivement l'écart minimum, moyen et maximum en pourcentage, *PMS* est le Pourcentage des Meilleures Solutions obtenues par chaque variante de l'approche mixte.

En comparant les résultats obtenus, nous pouvons constater que les deux variantes suivantes : δ| A| G et $\delta\&T$| A| G se démarquent, avec une légère dominance de $\delta\&T$| A| G. Elles sont également beaucoup plus performantes que l'heuristique FSIC.

En analysant l'impact de chaque technique sur la qualité des résultats obtenus pour les problèmes testés, nous pouvons tirer les conclusions suivantes :

- les variantes comportant la technique agrégée de formation des sous-problèmes montrent de meilleures performances que celles utilisant la technique indépendante ;
- les variantes utilisant l'approche par graphe pour la résolution des sous-problèmes ont été capables d'obtenir des solutions de meilleure qualité par rapport à celles faisant appel à la programmation linéaire en variables mixtes ;
- aucune technique de sélection des solutions heuristiques ne surpasse l'autre, les deux techniques peuvent conduire à des solutions de bonne qualité.

TAB. 5.5 – Résultats de l'évaluation de l'approche mixte

Série	Caractéristique	FSIC	δ| I| G	δ| A| G	$\delta\&T$| A| G	δ| A| MIP	$\delta\&T$| A| MIP
1	Δ_{min}	0	0	0	0	0	0
	Δ_{max}	5,4	3,7	0	0	15	10,9
	Δ_{moy}	0,49	0,35	0	0	4,86	2,63
	PMS %	90	94	100	100	34	56
2	Δ_{min}	0	0	0	0	0	0
	Δ_{max}	7,7	4,5	3,92	3,92	8,2	5,88
	Δ_{moy}	3,1	1,8	0,58	0,81	3,74	2,44
	PMS %	18	34	78	70	34	22
3	Δ_{min}	0	0	0	0	0	0
	Δ_{max}	7,69	3,5	3,85	1,87	4,22	5,1
	Δ_{moy}	3,87	1,2	0,74	0,13	1,36	1,72
	PMS %	6	42	62	90	34	28
4	Δ_{min}	0	0	0	0	0	0
	Δ_{max}	10,08	4,2	3,74	1,56	3,74	3,74
	Δ_{moy}	3,58	1,2	0,82	0,26	1,32	0,84
	PMS %	14	36	50	76	22	46

Nous pouvons également remarquer que pour la série avec $n_{ent} = 10$ (Série 1), tous les algorithmes utilisant l'approche par graphe ont été capables de trouver des solutions de bonne qualité. En revanche, pour les séries comportant la grande valeur de n_{ent}, l'avantage de la technique agrégée de formation des sous-problèmes sur la technique indépendante est évident.

Pour terminer, notons que nous pouvons envisager une autre technique d'agrégation des solutions des sous-problèmes si la programmation linéaire en variables mixtes est utilisée pour la résolution exacte. Ayant la solution S_{u-1} du sous-problème SP_{u-1}, nous pouvons fixer toutes les opérations appartenant à cette solution aux blocs et aux postes auxquels elles sont affectées, c'est-à-dire que pour toutes les opérations $j \in N^{u-1}$: $Q(j) = [q^*(j), q^*(j)]$ et $K(j) = [\lceil q^*(j)/n_0 \rceil, \lceil q^*(j)/n_0 \rceil]$ où $q^*(j)$ est le numéro du bloc auquel l'opération j est affectée dans la solution S_{u-1}. Dans ce cas, seule l'affectation des opérations de l'ensemble N^u est possible lors de la résolution du sous-problème SP_u. Ceci permet de diminuer la taille des problèmes à résoudre par la méthode exacte.

5.2.8 Conclusion

Nous avons proposé une approche mixte basée sur la décomposition dynamique d'une solution heuristique en sous-problèmes et sur la résolution de ces derniers par une méthode exacte. Plusieurs techniques nouvelles ont été élaborées pour réaliser cette approche, notamment deux techniques de sélection des solutions heuristiques à décomposer, deux techniques de formation des sous-problèmes, ainsi que des algorithmes permettant d'intégrer dans cette

approche deux méthodes exactes (approche par graphe et programmation linéaire).

À travers les expérimentations numériques, nous avons comparé les différentes variantes de cette approche entre elles et avec l'heuristique FSIC sans la phase d'amélioration. Les résultats obtenus montrent que la décomposition dynamique et la résolution exacte des sous-problèmes permettent d'améliorer les solutions fournies par l'heuristique FSIC. L'amélioration dépend des caractéristiques du problème ainsi que des paramètres de l'algorithme et du temps de résolution alloué.

Rappelons que cette approche peut être utilisée avec des heuristiques et des méthodes exactes différentes, tout en sachant que la qualité des résultats obtenus dépendra également de la qualité des méthodes utilisées.

Maintenant nous avons une méthode assez efficace pour l'amélioration des solutions heuristiques, par conséquent, nous pouvons envisager le développement d'une approche métaheuristique qui sera détaillé dans la section suivante.

5.3 Approche GRASP

La métaheuristique GRASP (Greedy Randomized Adaptive Search Procedure) a été proposée initialement par Feo et Resende [Feo et Resende, 1989]. Elle fait partie des méthodes approchées qui sont applicables pour la résolution d'une large gamme de problèmes d'optimisation combinatoire. D'ailleurs, elle a été déjà appliquée avec succès à un grand nombre de problèmes d'optimisation, notamment pour la résolution des problèmes de structuration des lignes d'assemblage [Andres *et al.*, 2006]. Des analyses bibliographiques consacrées à cette méthode ont été publiées dans [Festa et Resende, 2001; Resende et Ribeiro, 2003]. La flexibilité de cette approche réside dans le fait que ses opérateurs de base ne sont pas axés sur un problème particulier, mais peuvent être adaptés à la résolution de problèmes différents. Le développement d'une méthode de type GRASP consiste alors à adapter ses opérateurs de base au contexte du problème étudié.

5.3.1 Schéma général de l'approche

Une méthode de type GRASP est constituée des deux phases suivantes :

1. La construction d'une solution admissible à l'aide d'un algorithme glouton aléatoire, dit semi-glouton,
2. Son amélioration par une méthode de recherche locale.

Dans le schéma de GRASP, initialement proposé dans [Feo et Resende, 1989], ces deux phases sont répétées de manière itérative un grand nombre de fois jusqu'à ce qu'une condition d'arrêt soit satisfaite. Afin de compléter cette approche de base, de nombreux opérateurs d'approfondissement ont été proposés, comme par exemple :

- l'évolution, appelée « Reactive GRASP », qui est basée sur la détermination de manière dynamique de l'importance de la composante aléatoire,

- l'utilisation de techniques à base de mémoire et d'apprentissage, pour profiter de l'information apportée par les solutions générées aux itérations précédentes,

- le couplage avec des méthodes de « path relinking » pour rechercher de nouvelles solutions de bonne qualité en recomposant des chemins entre les solutions trouvées par GRASP.

En outre, de nombreuses hybridations avec d'autres métaheuristiques ont été mises en œuvre, pour plus de détails voir [Resende et Ribeiro, 2003; Delorme, 2003]. Une présentation générale de GRASP a été également faite dans [Pitsoulis et Resende, 2002].

Dans ce mémoire, nous adaptons l'approche de Reactive GRASP au problème d'optimisation de la configuration des machines de transfert. L'approche que nous mettons en œuvre est constituée des deux phases de base, proposées dans [Feo et Resende, 1989], et est dotée d'un seul opérateur d'approfondissement qui permet de piloter l'importance de la composante aléatoire lors de la construction de solutions initiales. La phase de construction est détaillée dans la Section 5.3.2. La phase d'amélioration est considérée dans la Section 5.3.3.

5.3.2 Phase de construction gloutonne aléatoire : heuristique GBL

Commençons par une description générale de la phase de construction. Durant cette phase, une solution admissible doit être construite par une heuristique en utilisant une fonction gloutonne, que nous désignons comme $g(j)$. Un tel algorithme construit la solution élément par élément. À chaque étape de la construction, un ensemble CL des éléments-candidats, pouvant être ajoutés dans la solution, est construit. La valeur de la fonction gloutonne utilisée est calculée pour chaque élément-candidat. Ensuite, les éléments sont classés dans l'ordre des valeurs de la fonction gloutonne (la direction de tri dépend de la fonction gloutonne choisie, qui peut être à minimiser ou à maximiser). L'élément le mieux placé est ajouté dans la solution. Le changement de la solution entraîne le changement du contenu de l'ensemble CL, qui est mis à jour, et le changement des valeurs de la fonction gloutonne pour ses éléments.

Si l'algorithme glouton ne comporte pas de composante aléatoire, la solution sera identique quelle que soit l'itération. Pour introduire le facteur « hasard » et générer de différentes solutions, sans pour autant perdre la manière rigoureuse de construction des solutions, des algorithmes semi-gloutons [Hart et Shogan, 1987; Feo et Resende, 1989] sont utilisés. Un tel algorithme construit une liste complémentaire des éléments qui sont sélectionnés dans la liste CL. Cette nouvelle liste est désignée comme RCL (Restricted Candidate List). Ensuite, un élément à affecter est choisi au hasard dans la liste RCL. Différentes techniques de la construction de cette liste ont été proposées dans la littérature [Hart et Shogan, 1987; Feo et Resende, 1989; Resende *et al.*, 2000]. Nous avons opté pour l'utilisation de la méthode suivante : un élément est placé dans la liste RCL si la valeur de la fonction gloutonne qui lui est attribuée n'est pas très loin de la meilleure valeur obtenue pour les éléments de la liste CL. Par exemple, pour notre cas, nous avons choisi à maximiser la fonction $g(j)$, en conséquence, nous sélectionnons les éléments dans la liste RCL de la manière suivante : $RCL = \{j \in CL |\, g(j) \geq g_{max} - \alpha(g_{max} - g_{min})\}$, où g_{max} et g_{min} donnent respectivement la valeur maximum et minimum de la fonction $g(j)$ pour les éléments de la liste CL. Le paramètre $\alpha \in [0, 1]$ pilote l'importance de la composante aléatoire. Si $\alpha = 0$, alors l'algorithme devient purement glouton, si $\alpha = 1$, alors un élément est choisi de manière complètement aléatoire.

Dans notre cas, les éléments des listes CL et RCL sont des opérations à affecter. Nous

pouvons également remarquer que pour notre problème, un algorithme avec $\alpha = 1$ existe déjà, c'est l'heuristique FSIC, présentée dans la Section 5.1. Cependant, cette heuristique n'intègre pas le paramètre α dans la procédure du choix des opérations. Pour cette raison, nous avons développé une nouvelle heuristique, baptisée GLB (pour Greedy Blocks Loading), qui, à la différence de l'heuristique FSIC, s'appuie sur une règle de priorité lors de la sélection des opérations à affecter. Dans ce qui suit, nous présentons cette nouvelle heuristique. Tout d'abord, l'Algorithme 5.9 décrit son schéma général. Ensuite, chaque procédure de cet algorithme est énoncée, notamment les procédures de la construction des listes *CL* et *RCL*.

L'algorithme commence par l'ouverture d'un poste de travail et d'un seul bloc vide appartenant à ce poste. Soit m l'indice du poste courant et n_m l'indice du bloc courant du poste m. L'algorithme s'arrête soit lorsque toutes les opérations sont affectées, soit lorsqu'il reste encore des opérations non affectées, mais qu'aucune action n'est possible : ni de les affecter aux blocs et postes déjà existants, ni d'ouvrir un nouveau poste, car $m + 1 > m_0$. Dans le deuxième cas, l'itération courante est jugée non concluante et $C_{cur} = \infty$.

Construction de la liste *CL* de candidats

La liste N^{na} des opérations non affectées est analysée, l'opération j est placée dans la liste *CL* si elle peut être affectée au bloc courant, c'est-à-dire si toutes les conditions suivantes sont validées :

1. Tous ses prédécesseurs sont déjà affectés, c'est-à-dire si $(i, j) \in D^{pr}$ alors $i \in \bigcup_{k=1}^{m} \bigcup_{r=1}^{n_m} N_{kr}$;
2. Son affectation au bloc courant n'enfreint pas la contrainte de temps de cycle pour le poste courant, c'est-à-dire :
   $t^p(N_m \setminus N_{mn_m}) + t^b(N_{mn_m} \cup \{j\}) \leq T_0$;
3. L'opération j n'est pas liée par des contraintes E^p avec les opérations déjà affectées au poste courant, c'est-à-dire que $\forall e \in E^p$ tels que $j \in e : e \cap (N_m \cup \{j\}) \neq e$
4. L'opération j n'est pas liée par des contraintes E^b avec les opérations déjà affectées au bloc courant, c'est-à-dire que $\forall e \in E^b$ tels que $j \in e : e \cap (N_{mn_m} \cup \{j\}) \neq e$.

Si $CL = \emptyset$, cela signifie qu'aucune opération parmi celles non encore affectées ne peut être assignée au bloc courant. Dans ce cas, l'algorithme essaie d'ouvrir un nouveau bloc et de reconstruire la liste *CL*. Si l'ouverture d'un nouveau bloc au poste courant n'est plus possible ou si la liste *CL* est de nouveau vide et si, en même temps, le nombre de postes ouverts est inférieur à m_0, alors un nouveau poste est ouvert et la liste *CL* est reconstruite.

Construction de la liste *DAO*(*CL*)

Grâce à l'exécution parallèle d'opérations dans un bloc, certaines opérations se font en temps caché, c'est-à-dire que les paramètres de deux opérations au plus sont utilisés pour le calcul de temps d'usinage, les autres opérations du bloc n'influencent pas ce temps. Il en découle qu'il est possible d'affecter des opérations à un bloc contenant déjà des opérations sans changer son temps. Si nous prenons le bloc courant N_{mn_m}, alors nous pouvons ajouter dans ce bloc, sans changer son temps, toute opération j pour laquelle :

Algorithme 5.9 : Algorithme GLB

Initialiser $m = 1, n_m = 1, C_{cur} = C_1 + C_2$, $N^{na} = \mathbf{N}$, $newst = faux$
tant que $N^{na} \neq \emptyset$ **faire**
  si non *newst* **alors**
    Construire la liste *CL*
  fin
  si $CL = \emptyset$ **ou** *newst* **alors**
    si non *newst* **et** $n_m + 1 \leq n_0$ **alors**
      $n_m = n_m + 1$, $C_{cur} = C_{cur} + C_2$
    sinon
      si $m + 1 \leq m_0$ **alors**
        $m = m + 1$, $n_m = 1$, $C_{cur} = C_{cur} + C_1 + C_2$, $newst = faux$
      sinon
        $C = \infty$, **Quitter**
      fin
    fin
  sinon
    Construire $DAO(CL)$.
    si $DAO(CL) \neq \emptyset$ **alors**
      Affecter tous les $j \in DAO(CL)$
    sinon
      $assign = faux$
      tant que $assign \neq vrai$ **faire**
        Construire *RCL*
        Choisir l'opération i de la liste *RCL*
        Construire *AL*
        si $|AL| > 1$ **alors**
          $S_{copy} = S_{cur}$
        fin
        $assign = ASSIGN(AL)$
        si $assign = vrai$ **alors**
          $N^{na} = N^{na} \setminus AL$
        sinon
          $S_{cur} = S_{copy}$, $RCL = RCL \setminus \{i\}$, $CL = CL \setminus \{i\}$
          si $CL = \emptyset$ **alors**
            si $N_{mn_m} = \emptyset$ **alors**
              $n_m = n_m - 1$, $C_{cur} = C_{cur} - C_2$, $newst = vrai$
            fin
            Quitter
          fin
        fin
      fin
    fin
  fin
fin

$$L(N_{mn_m}) \geq l_j, v_f(N_{mn_m}) \geq v_{f1}(j), v_f(N_{mn_m}) \leq v_{f2}(j) \tag{5.7}$$

Si la condition (5.7) n'est pas valide, nous pouvons vérifier les conditions suivantes :

$$\begin{array}{c} \min\{l_i \mid i \in CL, i \neq j\} \geq l_j, \\ \max\{v_{f1}(i) \mid i \in CL, i \neq j\} \leq v_{f2}(j), \\ \min\{v_{f2}(i) \mid i \in CL, i \neq j\} \geq v_{f2}(j) \end{array} \tag{5.8}$$

Si pour l'opération j la condition (5.7) ou (5.8) est respectée et si en plus cette opération n'est liée avec aucune opération non affectée par une contrainte d'exclusion ou d'inclusion, alors, nous pouvons constater que l'affectation de l'opération j au bloc courant n'exclut aucune opération de la liste CL. Dans le cas où la condition (5.8) est vérifiée, nous pouvons constater que même si le temps mort du poste courant est réduit après l'affectation de l'opération j, toutes les autres opérations restent encore affectables au bloc courant.

Par conséquent, nous pouvons affecter de telles opérations avant les autres sans exclure aucune opération de la liste CL. Soit $DAO(CL)$ (Directly Assigned Operations) la liste de telles opérations dans la liste CL. Si $DAO(CL) \neq \emptyset$, alors toutes les opérations $j \in DAO$ sont affectées au bloc courant et seulement après la liste CL est reconstruite. La même procédure est répétée jusqu'à $DAO(CL) = \emptyset$. Si $DAO(CL) = \emptyset$, alors l'algorithme passe à l'étape suivante.

Évaluation gloutonne des opérations

La différence principale entre les heuristiques FSIC et GLB se situe à cette étape. Si l'heuristique FSIC choisit l'opération j de la liste CL de manière aléatoire, l'heuristique GLB, elle, utilise une règle pour effectuer ce choix.

Cette règle a pour objectif de privilégier les affectations qui mènent à une solution finale de bonne qualité, c'est-à-dire ayant le moindre coût. Puisque toutes les possibilités ne seront pas exploitées, l'optimalité de la solution obtenue évidemment ne peut pas être garantie, mais une solution de bonne qualité devrait être atteinte plus rapidement qu'avec l'heuristique FSIC. La difficulté réside dans le développement de cette règle.

Comme nous l'avons montré dans le Chapitre 3, les règles de priorité [Talbot *et al.*, 1986; Hackman *et al.*, 1989; Scholl, 1999] utilisées pour la résolution du SALBP sont basées sur le cumul des temps opératoires, l'hypothèse qui n'est plus valable pour le problème que nous considérons. En plus, à notre connaissance, aucune règle de choix ne tient compte des contraintes d'inclusion et d'exclusion.

Nous avons proposé et testé les règles qui se basent sur :

1. Le nombre des opérations qui resteront affectables au bloc (ou poste) courant après l'affectation de l'opération j ;
2. La borne inférieure sur le coût de la solution calculée en tenant compte de l'affectation de l'opération j au bloc courant ;
3. La borne inférieure sur le nombre de blocs nécessaires pour affecter tous les successeurs de l'opération j.

Les résultats de tests ont montré la supériorité de la dernière règle, par conséquent, c'est elle qui est utilisée dans notre approche GRASP pour évaluer les opérations de la liste *CL*. Plus la valeur de cette borne est grande, plus tôt il faut affecter l'opération j. Pour calculer cette borne, nous utilisons l'approche développée dans la section 4.4.2. S'il y a plusieurs opérations qui ont la même valeur, nous choisissons en priorité celle qui est liée par plus de contraintes d'exclusion avec les autres. La supériorité de cette règle n'est pas inattendue, car elle tient compte de toutes les contraintes du problème. La première règle défavorise les opérations qui ont beaucoup de contraintes d'exclusion, ces opérations risquent de se retrouver à la fin, toutes ensemble, incompatibles les unes avec les autres. Ceci peut entraîner la création d'un grand nombre de blocs ayant très peu d'opérations. La deuxième et troisième règles s'appuient sur les mêmes propriétés, mais le calcul de la borne inférieure sur le coût donne souvent des valeurs proches, voire les mêmes, pour plusieurs opérations, ne permettant pas de distinguer l'impact de l'affectation de chacune d'entre elles au moment de la décision. De plus, ce calcul consomme plus de temps, puisque l'ensemble des opérations impliquées est d'une taille importante.

Construction de la liste *RCL*

Nous avons opté pour l'utilisation de la méthode suivante pour la construction de la liste *RCL* : un élément y est placé, si la valeur de la fonction gloutonne qui lui est attribuée n'est pas très loin de la meilleure valeur obtenue pour les éléments de la liste *CL* :

$$RCL = \{j \in CL |\, g(j) \geq g_{max} - \alpha(g_{max} - g_{min})\}$$

où g_{max} et g_{min} représentent les valeurs maximum et minimum de la fonction g pour les éléments de la liste *CL*.

Des expérimentations préliminaires ne nous ont pas permis d'identifier une valeur particulière du paramètre α pouvant être considérée comme une recommandation pour la résolution de toutes les instances (ou au moins une grande partie). Cette observation avait déjà été rapportée dans [Delorme, 2003; Resende et Ribeiro, 2003]. En conséquence, nous avons décidé de mettre en place la procédure d'ajustement de la valeur du paramètre α lors de la résolution de chaque problème particulier. De cette manière, la valeur utilisée est à chaque fois adaptée aux particularités du problème à résoudre. La procédure d'ajustement opère avec un ensemble discret des valeurs de α pré-définies (cet ensemble est désigné comme αSet). Chaque valeur de l'ensemble αSet a une probabilité d'être choisie par l'algorithme, désignée par pr_α. Au début de la première itération, toutes les valeurs ont la même probabilité d'être choisies, égale à :

$$pr_\alpha = \frac{1}{|\,\alpha Set|}\,, \forall\, \alpha \in \alpha Set$$

Mais au fur et à mesure des itérations, ces probabilités sont modifiées afin de privilégier le choix des valeurs de α qui amènent à des meilleures solutions finales. Les probabilités pr_α du choix des valeurs pré-définies $\alpha \in \alpha Set$ sont mises à jour selon une périodicité dépendant de la condition *prUpdate*. Les nouvelles probabilités sont calculées à partir du coût moyen des meilleures solutions obtenues en utilisant chaque valeur du paramètre α (ces solutions

sont stockées dans les ensembles S_α) et en tenant compte du meilleur et du pire coût parmi toutes les solutions déjà obtenues, représentées par l'ensemble de solutions $\mathbf{S}_{gr}$.

Tout d'abord, la valeur de val_α est calculée pour chaque $\alpha \in \alpha Set$ de la manière suivante :

$$val_\alpha = \left(\frac{\max\{\ C(S) \mid S \in \mathbf{S}_{gr}\} - \text{moyen}\{\ C(S) \mid S \in S_\alpha\}}{\max\{\ C(S) \mid S \in \mathbf{S}_{gr}\} - \min\{\ C(S) \mid S \in \mathbf{S}_{gr}\}} \right)^\sigma \tag{5.9}$$

où le paramètre σ est un paramètre de contrôle qui est utilisé afin d'atténuer les écarts entre les probabilités des différentes valeurs du paramètre α. La taille maximale des ensembles S_α est également un paramètre de contrôle de l'algorithme. Ensuite, la probabilité du choix de la valeur du paramètre $\alpha \in \alpha Set$ est calculée comme suit :

$$pr_\alpha = \frac{val_\alpha}{\sum_{\alpha \in \alpha Set} val_\alpha} \tag{5.10}$$

La valeur du paramètre α est ensuite choisie en tenant compte de la probabilité pr_α.

Construction de la liste *AL*

Rappelons qu'afin de diminuer le nombre d'itérations non concluantes, nous devons veiller à l'affectation de toutes les opérations liées par une contrainte d'inclusion au même poste de travail. Pour cela, si l'opération j, qui a été choisie pour être affectée, possède ce genre des contraintes ($OMP(j) \neq \emptyset$), alors l'algorithme crée une liste complémentaire *AL* contenant toutes les opérations de $OMP(j)$ non affectées et tous leurs prédécesseurs non affectés. En effet, nous ne pouvons pas, en général, savoir d'avance le nombre de blocs nécessaire pour l'affectation de toutes les opérations de *AL*, et nous ne pouvons pas savoir s'il sera possible de toutes les affecter au poste courant. Pour cette raison, l'état de la solution courant S_{cur} est sauvegardé (désigné par S_{copy}) avant l'affectation de la première opération de la liste *AL*. Ensuite, les opérations comprises dans la liste *AL* sont affectées de la même façon que les opérations de la liste *CL*, à savoir : les valeurs de la fonction gloutonne sont calculées pour ces opérations, puis la liste *RCL* est construite et l'opération à affecter y est sélectionnée. Si aucune opération de la liste *AL* ne peut plus être affectée au bloc courant, une opération peut être choisie dans la liste *CL*, si ses contraintes d'exclusion n'empêcheront pas par la suite l'affectation des opérations de la liste *AL* au poste courant.

Si toutes les opérations de la liste *AL* ont été affectées avec succès au poste courant, alors la liste *CL* est mise à jour et la construction de la solution se poursuit, sinon l'algorithme restaure l'état de la solution avant l'affectation de la première opération de la liste *AL* : $S_{cur} = S_{copy}$. L'opération j est ensuite supprimée des listes *RCL* et *CL*. Puis, une autre opération est sélectionnée de la liste *RCL*.

5.3.3 Phase d'amélioration par recherche locale

La métaheuristique GRASP recoure à une procédure de recherche locale pour améliorer les solutions obtenues à la phase de construction. Compte tenu de bons résultats que nous

avons obtenus en utilisant l'approche de décomposition dynamique des solutions heuristiques dans la section précédente, nous avons décidé d'intégrer cette approche dans notre algorithme GRASP à la phase d'amélioration de solutions. Plus précisément, nous faisons appel à la technique de sélection des solutions de départ utilisant le paramètre δ, à la technique agrégée de formation de sous-problèmes et à l'approche par graphe pour les résoudre (pour leur description, voir la Section 5.2). Le voisinage que nous considérons est donc défini par l'ensemble des recompositions possibles de la solution après son découpage aléatoire en sous-problèmes. Il est intéressant de noter que, du fait du caractère aléatoire du découpage, l'application de cette technique plusieurs fois à la même solution initiale ne conduira pas nécessairement au même optimum local [Guschinskaya et Dolgui, 2007*a*]. L'Algorithme 5.10 récapitule l'approche « Reactive GRASP » que nous proposons pour la résolution du problème d'optimisation de la configuration des machines de transfert.

Algorithme 5.10 : Schéma général de la GRASP réactive

Initialiser $C_{min} = \infty$, $TR_{tot} = 0$, $TR_{nimp} = 0$, $pr_\alpha = \frac{1}{|\alpha Set|}$, $\forall\, \alpha \in \alpha Set$.

répéter

- Choisir α de l'ensemble αSet en tenant compte de pr_α
- Construire une solution admissible S_{cur} ayant le coût C_{cur}
- **si** $C_{cur} < C_{min}$ **alors**
  - $C_{min} = C_{cur}$, $S_{min} = S_{cur}$
- **fin**
- **si** $C_{cur} < \infty$ **alors**
  - Phase d'amélioration
  - **si** $C_{cur} < C_{min}$ **alors**
    - $C_{min} = C_{cur}, S_{min} = S_{cur}$, $TR_{nimp} = 0$
  - **fin**
- **fin**
- **si** $C_{cur} \geq C_{min}$ **alors**
  - $TR_{nimp} = TR_{nimp} + 1$
- **fin**
- **si** $prUpdate$ **alors**
  - $val_{\sum \alpha} = 0$
  - **pour chaque** $\alpha \in \alpha Set$ **faire**
    - $val_\alpha = \left(\frac{\max\{\, C(S) \mid S \in \mathbf{S}_{gr}\} - \text{moyen}\{\, C(S) \mid S \in S_\alpha\}}{\max\{\, C(S) \mid S \in \mathbf{S}_{gr}\} - \min\{\, C(S) \mid S \in \mathbf{S}_{gr}\}} \right)^\delta$
    - $val_{\sum \alpha} = val_{\sum \alpha} + val_\alpha$
  - **fin**
  - **pour chaque** $\alpha \in \alpha Set$ **faire**
    - $pr_\alpha = \frac{val_\alpha}{val_{\sum \alpha}}$
  - **fin**
- **fin**
- $TR_{tot} = TR_{tot} + 1$

jusqu'à $T_{cur} > T_{res}$ **ou** $TR_{nimp} > TR_{nimp\_aut}$ **ou** $TR_{tot} > TR_{tot\_aut}$ **ou** $C_{min} < C_{stop}$

5.3.4 Comparaison des approches mixte et GRASP

Notre but est de comparer les performances des approches mixte et GRASP. Puisque les deux approches utilisent la même méthode pour la phase d'amélioration des solutions heuristiques, la comparaison concerne l'efficacité des deux heuristiques utilisées pour la construction des solutions en tandem avec la phase d'amélioration.

Nous reprenons les jeux de données déjà utilisés dans la Section 5.2. Rappelons qu'il s'agit d'un échantillon de problèmes générés aléatoirement, mais en tenant compte des caractéristiques des problèmes industriels. Nous comparons les résultats déjà obtenus pour l'approche mixte, plus précisément en utilisant ses deux meilleures variantes, à savoir : $\delta|\,\mathrm{A}|\,\mathrm{G}$ et $\delta\&T|\,\mathrm{A}|\,\mathrm{G}$, avec les résultats fournis par l'approche GRASP. Les calculs ont été effectués sur un PC Pentium IV, 3GHz et 512 Mo de RAM. Le Tableau 5.6 fournit les résultats des expérimentations. Nous utilisons les mêmes notations que dans la Section 5.2.

Tab. 5.6 – Résultats de la comparaison des approches mixte et GRASP

Série	Méthode	$\Delta_{min}, \%$	$\Delta_{max}, \%$	$\Delta_{av}, \%$	$PMS, \%$
1	$\delta\|\,\mathrm{A}|\,\mathrm{G}$	0	4,76	0,23	93,3
	$\delta\&T|\,\mathrm{A}|\,\mathrm{G}$	0	4,76	0,3	93,3
	GRASP	0	0	0	100
2	$\delta\|\,\mathrm{A}|\,\mathrm{G}$	0	4	1,38	47,8
	$\delta\&T|\,\mathrm{A}|\,\mathrm{G}$	0	6	1,62	39,13
	GRASP	0	0	0	100
3	$\delta\|\,\mathrm{A}|\,\mathrm{G}$	0	10,2	4,14	14,3
	$\delta\&T|\,\mathrm{A}|\,\mathrm{G}$	0	11,2	3,5	14,3
	GRASP	0	0	0	100
4	$\delta\|\,\mathrm{A}|\,\mathrm{G}$	0,78	14,12	6,02	0
	$\delta\&T|\,\mathrm{A}|\,\mathrm{G}$	0,78	11,76	5,43	0
	GRASP	0	0	0	100

Les résultats obtenus nous permettent de conclure que l'utilisation de l'heuristique GBL à la phase de construction de solutions améliore considérablement la qualité des résultats finaux. Nous pouvons même constater que l'approche GRASP surpasse incontestablement l'approche mixte. Cette supériorité se révèle particulièrement importante lors de la résolution des problèmes de grande taille. Par exemple, pour toute instance de la série 4, l'approche GRASP a donné un résultat meilleur que l'approche mixte. L'écart entre les deux solutions est en moyenne de 6,02% pour la méthode $\delta|\,\mathrm{A}|\,\mathrm{G}$ et de 5,43% pour la méthode $\delta\&T|\,\mathrm{A}|\,\mathrm{G}$. Cette hypothèse est également confirmée par l'analyse des valeurs du paramètre *PMS*. Pour les instances de petite taille, le pourcentage des meilleures solutions trouvées par l'approche mixte est proche de 100% ; ensuite, il diminue avec l'augmentation de la taille d'instances jusqu'à ce qu'il soit carrément nul pour les instances de grande taille. Nous expliquons ce résultat par l'aptitude de l'heuristique GLB à fournir de bonnes solutions de départ. Ceci permet d'obtenir de bonnes solutions finales avec un temps relativement court de résolution.

Pour tester l'approche GRASP sur un échantillon de problèmes plus large, nous avons

généré 6 séries d'instances complémentaires. Chaque série comporte 20 instances différentes. Ces instances ont été générées de manière aléatoire à partir d'un nombre d'opérations et de la densité du graphe de précédence fixés dont les valeurs sont rapportées dans le Tableau 5.7. Chaque série a été résolue par l'approche GRASP et la méthode $\delta\&T|\,\mathrm{A}|\,\mathrm{G}$. Comme, pour toute instance testée, l'approche GRASP a donné un résultat au moins aussi bon que celui fourni par la méthode $\delta\&T|\,\mathrm{A}|\,\mathrm{G}$. Dans le Tableau 5.7, nous recensons seulement les écarts en pourcentage caractérisant la qualité des résultats obtenus avec la méthode $\delta\&T|\,\mathrm{A}|\,\mathrm{G}$ par rapport à ceux fournis par l'approche GRASP.

Tab. 5.7 – Résultats de l'évaluation de l'approche GRASP sur un échantillon de problèmes complémentaire

S	$\lvert\mathbf{N}\rvert$	OS	$\Delta_{min},\%$	$\Delta_{max},\%$	$\Delta_{av},\%$	$PMS,\%$
1	25	0,25	0	26,7	3,1	79
2	25	0,6	0	26,3	1,3	95
3	50	0,25	0	30	9,4	30
4	50	0,6	0	30,8	4,5	55
5	100	0,25	2,2	42	21,5	0
6	100	0,6	0	40,8	6,4	35

Les résultats obtenus pour cet échantillon de tests confirment l'hypothèse que plus la taille du problème à résoudre est grande plus la probabilité que la solution trouvée par l'approche GRASP soit meilleure que celle fournie par l'approche mixte est grande. Mais ici nous pouvons aussi observer une autre tendance. Nous pouvons constater que la méthode $\delta\&T|\,\mathrm{A}|\,\mathrm{G}$ est moins performante pour les problèmes ayant un bas niveau de contraintes de précédence que pour ceux avec un niveau moyen.

Pour récapituler, l'approche GRASP a fourni, pour l'ensemble des instances testées, des meilleurs résultats que l'approche mixte dans 58,6% des cas et des solutions identiques dans 42,4% des cas.

5.4 Conclusion

Dans ce chapitre, nous avons développé et étudié les performances de trois méthodes approchées pour la résolution du problème d'optimisation de la configuration des machines de transfert.

Tout d'abord, nous avons proposé des améliorations pour la meilleure méthode heuristique connue dans la littérature (FSIC) pour la résolution du problème traité. Les résultats de tests numériques nous ont permis de constater l'efficacité de nos améliorations.

Ensuite, nous avons doté l'heuristique FSIC d'une phase d'amélioration de solutions en faisant appel à une décomposition dynamique de solutions heuristiques en sous-problèmes et à une méthode exacte pour la résolution de ces derniers. Nous avons comparé les solutions obtenues par l'heuristique FSIC sans et avec la phase d'amélioration. Les résultats numériques

montrent que la décomposition dynamique et la résolution exacte des sous-problèmes permet d'améliorer les solutions fournies par l'heuristique, surtout pour les problèmes de grande taille. L'amélioration dépend des caractéristiques du problème ainsi que des paramètres de l'algorithme et du temps de résolution alloué.

Enfin, nous avons proposé une métaheuristique de type GRASP. Cette approche utilise une heuristique gloutonne aléatoire, baptisée GLB, pour la construction de solutions. La recherche locale qui suit cette construction est basée, comme dans le cas précédent, sur la décomposition dynamique.

Toutes ces approches ont été évaluées en recourant à des séries de tests, dont la structure a été choisie de sorte d'être similaire aux problèmes existant sur le terrain industriel. Les résultats obtenus permettent de conclure que l'utilisation de la métaheuristique GRASP pour la résolution des problèmes réels de taille importante est efficace. Toutefois, dans les cas où le temps de résolution est extrêmement restreint (de l'ordre de quelques minutes), l'utilisation de l'heuristique GLB sans la phase d'amélioration peut être recommandée.

Nos perspectives à court terme consistent à étudier plus minutieusement les performances des approches développées, notamment à évaluer l'impact de la structure de données du problème à résoudre.

Ainsi nous terminons l'étude du problème d'optimisation de la configuration des machines de transfert et nous considérons dans le chapitre suivant deux autres types de systèmes d'usinage à boîtiers multibroches. Plus précisément, nous formulons et étudions les problèmes d'optimisation de la configuration des machines à table mobile et des machines à table circulaire pivotante.

Chapitre 6

Optimisation de la configuration des machines à table mobile et à table circulaire pivotante : méthodes exactes

Tout ce qui est exact est court.

Joseph Joubert

Dans ce chapitre, nous étudions le problème d'optimisation de la configuration de deux différents systèmes d'usinage à boîtiers multibroches. Nous commençons par les machines à table mobile. Les spécificités de tels systèmes d'usinage vis-à-vis des machines de transfert sont liées aux deux facteurs suivants. D'une part, ils utilisent le mode parallèle d'activation de boîtiers. D'une autre part, le transfert de la pièce d'un poste de travail à un autre est effectué avec une seule table mobile. Par conséquent, une seule pièce est traitée tout au long du temps de cycle. Nous développons d'abord une modélisation mathématique du problème d'optimisation de la configuration des machines à table mobile. Puis, nous proposons une méthode exacte pour la résolution de ce problème. Un exemple industriel est ensuite exposé. À la fin du chapitre, nous présentons les machines à table circulaire pivotante et nous développons une modélisation mathématique du problème d'optimisation de la configuration des machines de ce type.

6.1 Machines à table mobile

6.1.1 Mode de fonctionnement

Les machines à table mobile sont utilisées pour la fabrication des pièces dont les gabarits ne permettent pas l'usinage de plusieurs unités de pièce par le même système d'usinage à boîtiers multibroches. Par exemple, la machine présentée dans la Figure 6.1 est destinée à

l'usinage des pipelines pour le transport du gaz et du pétrole. La photo nous a été fournie par notre partenaire industriel dans le cadre du projet européen INTAS 03-51-5501 (http ://www-leibniz.imag.fr/RO/INTAS/). Cette machine a été fabriquée par l'usine des systèmes d'usinage MZAL située à Minsk (Bélarus).

FIG. 6.1 – Une machine à table mobile

Sur une machine à table mobile, chaque pièce est usinée en passant par m postes de travail successifs. L'avancement de la pièce d'un poste au poste suivant se fait par le déplacement de la table mobile sur laquelle la pièce est fixée. À cause de ce mode de transfert, seule une pièce peut être présente sur la machine tout au long du temps de cycle. En plus, seul un poste de travail peut être actif à la fois (celui où se trouve la pièce), ce qui fait la différence entre ce type de machines et les machines de transfert considérées dans les chapitres précédents.

Une fois la pièce arrivée à un poste de travail, les unités d'usinage du poste avancent vers la table avec la pièce pour effectuer les opérations d'usinage qui leur sont affectées. Les unités d'usinage sont le plus souvent des boîtiers multibroches. Par conséquent, toutes les opérations affectées à un tel boîtier sont exécutées en parallèle. Sur un poste, nous pouvons avoir plusieurs unités d'usinage, ce qui permet de traiter simultanément plusieurs faces de la pièce, car toutes les unités d'usinage du même poste de travail sont activées simultanément. Lorsque toutes les opérations d'usinage affectées à un poste de travail sont finies, s'il reste encore au moins un poste de travail non activé, alors la table mobile avance vers le poste suivant, sinon elle revient au poste initial pour décharger la pièce finie et charger un nouveau brut. Un nouveau cycle commence pour la nouvelle pièce et ainsi de suite.

Pour ne pas gêner le déplacement de la table mobile, seulement deux unités d'usinage peuvent être installées sur chaque poste de travail : une à gauche par rapport à la table et une autre sur sa droite. Une troisième unité d'usinage peut être rajoutée devant la table uniquement sur le dernier poste de travail. Ainsi, sur chaque poste de travail (sauf le dernier), il est possible d'installer soit deux boîtiers horizontaux, soit un boîtier horizontal et un vertical, et sur le dernier poste on peut avoir au plus soit deux boîtiers horizontaux et un vertical, soit trois boîtiers horizontaux. Quant au nombre de postes de travail, ce type de machines en possède rarement plus de 3, pour avoir l'espace occupé par la machine et le temps d'usinage acceptables.

Le problème de configuration d'une machine à table mobile consiste à déterminer le nombre de postes de travail et de boîtiers multibroches à installer sur chaque poste afin d'effectuer toutes les opérations d'usinage tout en respectant les contraintes technologiques et techniques. La différence majeure par rapport au problème d'optimisation étudié dans les chapitres précédents réside dans l'activation des postes de travail. La configuration avec un seul poste de travail actif rend à la fois l'usinage de la pièce plus lent et la contrainte sur le temps de cycle plus sensible aux affectations d'opérations. Dans ce qui suit, nous présentons un modèle mathématique incorporant l'ensemble des contraintes à prendre en compte lors de la configuration d'une telle machine.

6.1.2 Modélisation mathématique

Nous reprenons les notations introduites dans la Section 2.5. Notre but maintenant est d'y ajouter la modélisation de la contrainte sur le temps de cycle en tenant compte du fait qu'une machine à table mobile utilise le mode parallèle d'activation de boîtiers et dispose d'un seul poste de travail actif à la fois.

Calcul du temps de cycle de la machine

Prenant en compte le fait que les postes de travail sont activés de façon séquentielle, la somme des temps des postes de travail définit le temps de cycle T de la machine. Étant donné que les blocs d'un poste sont activés en parallèle, le temps d'un poste de travail est défini par son bloc le plus lent. Alors, pour calculer T, nous devons passer par le calcul du temps de cycle de chaque bloc et, ensuite, de chaque poste de travail. Nous devons tenir compte non seulement des incréments (les temps auxiliaires) existant à chaque niveau, mais aussi du temps pendant lequel la table revient au poste initial pour décharger la pièce.

Comme nous l'avons énoncé dans la section 1.2.3, afin de calculer le temps d'usinage d'un boîtier multibroche t^b, il faut disposer des paramètres d'usinage pour chaque opération qui lui est affectée. Ainsi le temps d'usinage t^b du $r$ème boîtier multibroche du $k$ème poste de travail est calculé de la manière suivante :

$$t^b(N_{kr}) = \frac{\max\{\, l_i \mid i \in N_{kr}\}}{V_2(N_{kr})} + \tau^b \tag{6.1}$$

Le temps d'un poste de travail t^p est calculé de la manière suivante :

$$t^p(N_k) = \max\{t^b(N_{kr}) |\, r \in \{1, ..., r_k\}\} \; + \tau^p \tag{6.2}$$

Afin de respecter la cadence désirée, il faut que la somme des temps d'usinage des postes de travail successifs ne dépasse pas le temps de cycle objectif, soit :

$$T(S) = \tau^a + \sum_{k=1}^{m} t^p(N_k) \leq T_0 \tag{6.3}$$

où m est le nombre de postes de travail ; τ^a est une constante qui tient compte du temps de mouvement de la table.

Activation parallèle d'unités d'usinage

Le nombre de boîtiers installés sur un poste de travail est restreint par le paramètre n_0. De plus, il convient de veiller à ce que les boîtiers installés agissent sur les faces de la pièce de sorte que le posage correspondant de la pièce existe et qu'il n'y ait pas de collision entre les outils lors de l'usinage. Afin de vérifier ce genre de contraintes, nous avons besoin de distinguer les différents types de boîtiers. Le type de boîtier étant défini par sa position par rapport à la table mobile et par la direction de son action. Soit U l'ensemble des types de boîtiers qui peuvent être utilisés pour l'usinage de la pièce. Pour chaque opération, nous pouvons déterminer $u(j) \subseteq U$ l'ensemble des types de boîtiers capables de l'exécuter. De plus, l'installation de boîtiers sur un poste de travail doit respecter les contraintes d'agencement. Pour assurer la prise en compte des contraintes de ce type, nous dénotons l'ensemble des paires de types de boîtiers qui ne peuvent pas être installés sur le même poste de travail, s'il ne s'agit pas du dernier poste de travail, par E^{c2}. De même, nous dénotons l'ensemble des triplets de types de boîtiers qui ne peuvent pas usiner simultanément sur le dernier poste de travail par E^{c3}.

Pour alléger le modèle mathématique, nous inversons le graphe de précédence. De cette façon, nous pouvons considérer qu'il est possible d'installer trois boîtiers sur le premier poste de travail et non pas sur le dernier. Afin d'obtenir la solution du problème initial, il convient d'inverser tout simplement l'ordre des postes de travail constituant la solution pour le problème résolu.

Notations

Nous présentons un modèle mathématique du problème d'optimisation de la configuration des machines à table mobile en utilisant des variables de décision mixtes (réelles et binaires). Un tel modèle peut être facilement implémenté et résolu en recourant à un logiciel muni d'une bibliothèque d'optimisation.

Sachant que les boîtiers sont activés en parallèle sur un poste de travail, nous ne pouvons pas numéroter les blocs d'opérations correspondants selon l'ordre de leur exécution. De plus, nous devons distinguer les faces de la pièce traitées par les boîtiers afin de vérifier le respect des contraintes sur l'affectation des boîtiers au même poste. Alors, nous utilisons deux indices

pour déterminer les affectations d'opérations, à savoir :

- $k = 1, 2, ..., m_0$, est l'indice utilisé pour les postes de travail,
- $r = 1, ..., |U|$, est l'indice utilisé pour les types de boîtiers, le type de boîtier étant défini par sa position par rapport à la table mobile et par la direction de son action.

En utilisant ces indices, nous introduisons les variables de décision suivantes.

Variables de décision

X_{jkr} est une variable de décision binaire, qui est utilisée pour déterminer l'affectation des opérations $j \in \mathbf{N}$:

$$X_{jkr} = \begin{cases} 1 & \text{si l'opération } j \text{ est affectée au poste de travail } k \\ & \text{avec boîtier de type } r \\ 0 & \text{sinon} \end{cases}$$

Bien évidemment, chaque opération $j \in \mathbf{N}$ peut être affectée à un boîtier dont le type est compris dans l'ensemble $u(j)$, ce qui réduit le nombre des variables X_{jkr}.

Y_{kr} est une variable binaire auxiliaire, utilisée pour, d'une part, calculer le nombre de boîtiers installés lors de l'évaluation de la fonction objectif et, d'autre part, garantir que les boîtiers installés sur le poste de travail k sont compatibles :

$$Y_{kr} = \begin{cases} 1 & \text{si le boîtier de type } r \text{ est installé sur le poste de travail } k \\ 0 & \text{sinon} \end{cases}$$

Z_k est une variable binaire auxiliaire, elle est utilisée pour calculer le nombre de postes de travail dans une solution lors de l'évaluation de la fonction objectif :

$$Z_k = \begin{cases} 1 & \text{si le poste de travail } k \text{ existe} \\ 0 & \text{sinon} \end{cases}$$

Fonction objectif

La fonction objectif (6.4) représente le coût de la machine, ce coût est constitué de deux composants : le premier estime le coût propre aux postes de travail, le deuxième le coût de tous les boîtiers installés.

$$Minimiser\ C(S) = C_1 \sum_{k=1}^{m_0} Z_k + C_2 \sum_{k=1}^{m_0} \sum_{r=1}^{|U|} Y_{kr} \tag{6.4}$$

Dans ce qui suit nous présentons l'ensemble des contraintes de ce modèle.

Relations entre les variables de décision

L'équation (6.5) exprime le fait qu'un bloc n'existe que s'il y a des opérations qui lui sont assignées.

$$Y_{kr} \geq X_{jkr}, j \in \mathbf{N}; k = 1, ..., m_0; r \in u(j) \tag{6.5}$$

L'équation (6.6) assure que le poste k n'existe que s'il y a des opérations qui sont affectées à un de ses blocs.

$$Z_k \geq Y_{kr}; k = 1, \ldots, m_0; r = 1, \ldots, |\, U| \tag{6.6}$$

Contraintes d'occurrence

L'équation (6.7) impose que chaque opération de l'ensemble **N** soit affectée à un et un seul bloc.

$$\sum_{k=1}^{m_0} \sum_{r \in u(j)} X_{jkr} = 1, j \in \mathbf{N} \tag{6.7}$$

Contraintes de précédence

Les contraintes de précédence peuvent être exprimées de manière suivante :

$$\sum_{k=1}^{m_0} \sum_{r \in u(i)} kX_{ikr} \leq \sum_{k=1}^{m_0} \sum_{r' \in u(j)} kX_{jkr'}, i \in PredD(j), j \in \mathbf{N} \tag{6.8}$$

Contraintes de compatibilité entre les opérations

L'équation (6.9) introduit les contraintes d'inclusion :

$$\sum_{r \in u(i)} X_{ikr} - \sum_{r' \in u(j)} X_{jkr'} = 0; j, i \in e, e \in I^p; k = 1, 2 \ldots, m_0 \tag{6.9}$$

L'équation (6.10) traduit les contraintes d'exclusion au niveau des blocs :

$$\sum_{j \in e} X_{jkr} \leq |\, e| - 1, e \in E^b; r \in \bigcap_{j \in e} u(j); k = 1, 2 \ldots, m_0 \tag{6.10}$$

L'équation (6.11) exprime les contraintes d'exclusion au niveau des postes de travail :

$$\sum_{j \in e} \sum_{r \in u(j)} X_{jkr} \leq |\, e| - 1; e \in E^p; k = 1, 2 \ldots, m_0 \tag{6.11}$$

Contrainte sur le temps de cycle de la machine

Comme pour le problème présenté dans le chapitre 4, pour chaque opération $j \in \mathbf{N}$, nous calculons le paramètre t_j en utilisant (4.15). De plus, pour chaque paire des opérations i et j De plus, pour chaque paire d'opérations pouvant être affectées au même bloc ($\{i, j\} \notin E^b$), nous calculons le paramètre t_{ij} en utilisant (4.16). En recourant aux paramètres t_j, t_{ij} et en

introduisant une variable réelle auxiliaire F_k qui détermine le temps du poste k, soit $F_k = t^p(N_k)$, nous évitons la perte de la linéarité du modèle. La contrainte sur le temps de cycle de la machine s'exprime donc par les équations (6.12)-(6.14). Plus précisément, les équations (6.12)-(6.13) calculent la durée d'un bloc s'il est constitué d'une seule opération (6.12) ou d'un ensemble d'opérations (6.13). Tandis que l'équation (6.14) calcule la somme des temps de cycle des postes de travail et vérifie que cette somme ne dépasse pas T_0.

$$F_k \geq t_j X_{jkr} + \tau^p; k = 1, 2 \ldots, m_0; r \in u(j), j \in \mathbf{N} \tag{6.12}$$

$$\begin{gathered} F_k \geq t_{ij}(X_{ikr} + X_{jkr} - 1) + \tau^p, \\ i, j \in \mathbf{N}; r \in u(i) \cap u(j), i < j, \{i, j\} \notin (E^p \cup E^b); k = 1, ..., m_0 \end{gathered} \tag{6.13}$$

$$\tau^a + \sum_{k=1}^{m_0} F_k \leq T_0; k = 1, ..., m_0 \tag{6.14}$$

$$F_k \in [\tau^p, T_0 - \tau^a]; k = 1, ..., m_0 \tag{6.15}$$

Contraintes sur l'affectation des blocs

Les équations (6.16)-(6.17) assurent que le nombre de boîtiers affectés à chaque poste de travail ne dépasse pas la valeur maximum autorisée. Rappelons que pour le problème inversé, cette valeur est égale à 3 pour le premier poste de travail, alors que pour tous les autres postes, cette valeur est fixée à 2.

$$\sum_{r=1}^{|U|} Y_{kr} \leq 2; k = 2, ..., m_0 \tag{6.16}$$

$$\sum_{r=1}^{|U|} Y_{kr} \leq 3; k = 1 \tag{6.17}$$

Prenons tous les postes de travail sauf le premier, E^{c2} est l'ensemble des paires de types de boîtiers qui ne peuvent pas être installés sur le même poste de travail. Pour modéliser cette contrainte, nous utilisons les équations suivantes :

$$Y_{kr_1} + Y_{kr_2} \leq 1; k = 2, ..., m_0; \{r_1, r_2\} \in E^{c2} \tag{6.18}$$

De la même manière, sachant que E^{c3} est la collection des ensembles (paires ou triplets) de boîtiers qui ne peuvent pas être installés sur le premier poste de travail, l'équation suivante interdit leur affectation au même poste :

$$\sum_{r \in e} Y_{kr} \leq |\, e| - 1; k = 1; e \in E^{c3} \tag{6.19}$$

Le modèle présenté peut être facilement implémenté et résolu en recourant à un logiciel muni d'une bibliothèque d'optimisation. Cependant, les résultats obtenus dans [Dolgui *et al.*,

2006*d*] montrent que pour ce type de problèmes, il est parfois plus intéressant d'implémenter une approche dédiée, permettant de tenir compte des contraintes du problème de façon plus efficace. Dans ce qui suit, nous proposons une telle approche basée sur la recherche du plus court chemin sous contraintes dans un graphe spécialement construit.

6.1.3 Approche par graphe

Notations et définitions

D'abord, nous rappelons les notations introduites dans la section 2.5.5.

m	le nombre de postes de travail de la solution S
k	l'indice pour les postes de travail, $k = 1, 2, ..., m$
N_k	l'ensemble des opérations affectées au poste avec l'indice k
r_k	le nombre de blocs du poste de travail k
r	l'indice pour les blocs d'un poste de travail, pour le poste k : $r = 1, 2, ..., r_k$
N_{kr}	l'ensemble des opérations affectées au bloc b du poste de travail r

En utilisant les notations ci-dessus, une solution du problème d'optimisation de la configuration des machines à table mobile a la forme suivante : $S = \{\{N_{11}, ..., N_{1r_1}\}, ..., \{N_{m1}, ..., N_{mr_m}\}\}$.

Nous définissons une fonction auxiliaire $O(N)$, dont la valeur est égale au nombre de boîtiers nécessaires pour exécuter l'ensemble d'opérations N sur un poste de travail. Pour les machines étudiées, un ensemble quelconque N est soit décomposable de manière unique en $O(N)$ sous-ensembles correspondant aux boîtiers, soit il est impossible d'accomplir l'ensemble N sur le même poste de travail (incompatibilité de régimes d'usinage, non respect des contraintes...). Cette propriété est liée au fait que chaque opération est associée à une face de la pièce qui peut être traitée par un seul boîtier multibroche à chaque poste de travail. De plus, la faisabilité de l'usinage sur plusieurs faces est vérifiée à l'aide des contraintes E^{c2} et E^{c3}. La fonction $O(N)$ prend une valeur très grande (> 3, c'est-à-dire plus grande que le nombre maximum autorisé de blocs sur un poste de travail pour ce type de machine) lorsque l'ensemble N ne peut pas être affecté à un poste de travail en respectant toutes les contraintes du problème, c'est-à-dire dans le cas où au moins une des conditions suivantes serait vérifiée :

1. $\exists\, e \in E^p$ tel que $e \subseteq N$.
2. $\exists\, e \in I^p$ tel que $e \cap N \neq e$.
3. $\exists\, e \in E^b$ et $\exists\, r \in \{1, ..., r_k\}$ tels que $e \subseteq N_{kr}$.
4. $\exists\, r \in \{1, ..., r_k\}$ tel que $V_1(N_{kr}) > V_2(N_{kr})$, où les valeurs des paramètres $V_1(N_{kr})$ et $V_2(N_{kr})$ sont calculées à l'aide de (1.9) et (1.10), respectivement.
5. Les types de boîtiers nécessaires pour l'exécution des blocs N_{kr} sont incompatibles sur un poste de travail, c'est-à-dire une des contraintes E^{c2} ou E^{c3} est violée.

En utilisant les notations ci-dessus, le problème d'optimisation de la configuration des machines à table mobile, initialement présenté par le modèle en variables mixte (6.4)-(6.19), peut être formulé comme suit.

La fonction objectif (6.20) donne une estimation du coût du système de fabrication (à minimiser).

$$\text{Minimiser } C(S) = C_1 m + C_2 \sum_{k=1}^{m} O(N_k) \tag{6.20}$$

La contrainte (6.21) assure la productivité voulue.

$$T(S) = \tau^a + \sum_{k=1}^{m} t^p(N_k) \leq T_0 \tag{6.21}$$

Les contraintes (6.22)-(6.23) sont nécessaires pour garantir l'affectation de chaque opération à un et un seul poste de travail.

$$\bigcup_{k=1}^{m} N_k = \mathbf{N} \tag{6.22}$$

$$N_{k'} \cap N_{k''} = \emptyset, \forall k', k'' \in \{1, ..., m\} \text{ tel que } k' \neq k'' \tag{6.23}$$

Les équations (6.24)-(6.25) vérifient les contraintes de précédence, d'inclusion et d'exclusion selon la définition de la fonction $O(N)$. L'équation (6.25) ne concerne que le dernier poste de travail. Ici nous n'utilisons pas l'ordre inversé des postes comme dans le modèle précédent, mais l'ordre direct.

$$O(N_k) \leq 2, k = 1, \ldots, m-1 \tag{6.24}$$

$$O(N_m) \leq 3 \tag{6.25}$$

L'équation (6.26) assure que le nombre de postes de travail ne dépasse pas m_0.

$$m = m(S) \leq m_0 \tag{6.26}$$

Génération du graphe

Le problème d'optimisation (6.20)-(6.26) peut être ramené à un problème de recherche du plus court chemin sous contraintes dans le graphe orienté Γ suivant. Soit $\mathcal{S}$ l'ensemble des collections $S^* =< N_1, \ldots, N_k, \ldots, N_m >$ satisfaisant les contraintes (6.22)-(6.25). L'ensemble $v_k = \bigcup_{n=1}^{k-1} N_n$ peut être considéré comme l'état de la pièce après son usinage sur les k premiers postes de travail (l'ensemble des opérations déjà exécutées). Soit $\mathcal{V}$ l'ensemble de tous les états possibles de la pièce y compris l'état initial $v_0 = \emptyset$ (brut) et l'état final $v_{|\mathbf{N}|}=\mathbf{N}$(produit fini). Le graphe orienté $\Gamma = (\mathcal{V}, \mathcal{D})$ est construit de telle manière qu'un arc $d = (v', v'') \in \mathcal{D}$ si et seulement si $v' \subset v''$, $O(N'') \leq 2$ pour $v'' \neq v_{|\mathbf{N}|}$ et $O(N'') \leq 3$ pour $v''= v_{|\mathbf{N}|}$ où $N''=v'' \setminus v'$. L'arc (v', v'') représente l'ensemble des opérations $v'' \setminus v'$ réalisées sur un poste de travail. À chaque arc (v', v'') deux paramètres sont associés :

– un coût $C^p(v', v'') = C_1 + C_2 \cdot O(v'' \setminus v')$;

– un temps $T(v', v'') = t^p(v'' \setminus v')$.

À chaque solution $S \in \mathcal{S}$, nous associons un chemin $z(S) = (v_0 = u_0, \ldots, u_{j-1}, u_j, \ldots, u_m(x) = v_{|\mathbf{N}|})$ de v_0 à $v_{|\mathbf{N}|}$ dans le graphe Γ. Soit $\mathcal{Z}$ l'ensemble de tous les chemins dans Γ de v_0 à $v_{|\mathbf{N}|}$. Alors un chemin $z \in \mathcal{Z}$ correspond à une solution $S(z)$=($u_1 \setminus u_0, \ldots, u_j \setminus u_{j-1}, \ldots, u_{m(x)} \setminus u_{m(x)-1}$) qui satisfait les contraintes (6.22)-(6.25) mais peut ne pas satisfaire les contraintes (6.21) et (6.26). Alors, le problème initial (6.20)-(6.26) peut être transformé en problème de recherche du plus court chemin suivant :

$$\text{Minimiser } C(z) = \sum_{k=1}^{m(z)} C^p(u_k \setminus u_{k-1}) \tag{6.27}$$

$$\sum_{k=1}^{m(z)} t^p(u_k \setminus u_{k-1}) \leq T_0 - \tau^a \tag{6.28}$$

$$z \in \mathcal{Z} \tag{6.29}$$

$$m(z) \leq m_0 \tag{6.30}$$

Algorithme de résolution

Nous proposons l'Algorithme 6.1 qui génère le graphe Γ et résout le problème (6.27)-(6.30) simultanément. Pour sa présentation, nous introduisons les notations suivantes. Les sommets $\mathcal{V}$ sont numérotés dans un ordre non décroissant de leur rang dans le graphe Γ. L'ensemble des sommets $\mathcal{V}$ peut être facilement décomposé en sous-ensembles V_i de telle manière que $v \in V_i$ si $|\, v| = i, i = 0, 1, ..., |\, \mathbf{N}|$. Il est à noter que le graphe Γ ne contient pas d'arc entre les sommets de l'ensemble V_i et les sommets de l'ensemble V_j si $i \geq j$.

Soit $D(v)$ l'ensemble des arcs d sortants du sommet v. Pour construire l'ensemble $D(v)$, la démarche suivante est utilisée. Soit $J(v)$ l'ensemble des opérations qui peuvent être exécutées sur le poste de travail succédant au poste de travail où la pièce a été transformée en l'état v. En analysant les contraintes d'inclusion, nous pouvons construire les sous-ensembles d'opérations $J_i(v)$, i= 1, 2, ..., $I(v)$ de l'ensemble $J(v)$, tels que chaque sous-ensemble $J_i(v)$ ne comporte que les opérations qui doivent être exécutées sur le même poste de travail (un sous-ensemble $J_i(v)$ peut ne contenir qu'une seule opération). La variable $I(v)$ est utilisée pour indiquer le nombre total de sous-ensembles ainsi obtenus. Ensuite, les arcs sortant du sommet v sont créés en regroupant des sous-ensembles $J_i(v)$ compatibles sur un poste de travail. Dans l'algorithme 6.1, la variable $last(d)$ indique le plus grand indice du sous-ensemble $J_i(v)$ déjà inclut dans l'arc $d \in D(v)$. Après la construction de l'ensemble $D(v)$, les propriétés de dominance sont appliquées. Par exemple, un arc d' domine un autre arc d si $d \subset d'$, $O(d') = O(d)$, $L(d') = L(d)$, et $v_f(d') = v_f(d)$. Les arcs dominés sont supprimés de l'ensemble $D(v)$.

Un triplet $(c(v), t(v), m(v))$ correspond au chemin dans le graphe Γ du sommet v_0 jusqu'au sommet v contenant exactement $m(v)$ arcs et ayant le coût $c(v)$ et le temps $t(v)$. L'opération MINMINMIN dénote la recherche de triplets non-dominés (un triplet (a_1, b_1, c_1) domine un autre triplet (a_2, b_2, c_2) si $a_1 \leq a_2$, $b_1 \leq b_2$ et $c_1 \leq c_2$).

Algorithme 6.1 : Génération du graphe Γ et résolution du problème d'optimisation

$V_i \leftarrow \{\emptyset\}$, $i = 0, 1, \ldots, |\,\mathbf{N}|$, $\{(c(v_0), t(v_0), m(v_0))\} = \{(\infty, \infty, \infty)\}$
pour $i = 0, \ldots, |\,\mathbf{N}| - 1$ **faire**
  pour chaque $v \in V_i$ **tel que** $\min\{m(v)\} < m_0$ **faire**
    Construire $J(v)$ et le partitionner en sous-ensembles $J_i(v)$, $i = 1, 2, ..., I(v)$.
    Assigner $D(v) \leftarrow \{\emptyset\}$
    pour $i = 1, 2, ..., I(v)$ **faire**
      $d \leftarrow J_i(v)$
      $last(d) \leftarrow i$
      ajouter d à $D(v)$
    fin
    pour chaque $d \in D(v)$ **faire**
      pour $k = last(d) + 1, 2, \ldots, I(v)$ **faire**
        $d' \leftarrow d \cup J_k(v)$; $last(d') \leftarrow k$
        si $O(d') \leq 2$ **ou** $O(d') \leq 3$ **et** $d' \cup v = \mathbf{N}$ **alors**
          ajouter d' à $D(v)$.
        fin
      fin
    fin
    Exclure les arcs dominés de $D(v)$
    pour chaque *arc* $d \in D(v)$ **faire**
      a) $w \leftarrow v \cup d$
      b) **si** $w \notin V_{|w|}$ **alors**
        ajouter w dans $V_{|w|}$ et $\{(c(w), t(w), m(w))\} \leftarrow \{(\infty, \infty, \infty)\}$
      fin
      c) **si** $v = v_0$ **alors**
        $\{(c(w), t(w), m(w))\} \leftarrow \{(C^p(d), T(d), 1)\}$
      sinon
        $\{(c(w), t(w), m(w))\} \leftarrow$ MINMINMIN$(\{(c(w), t(w), m(w))\} \cup$
        $\{(c(v) + C^p(d), t(v) + T(d), m(v) + 1) |\, t(v) + T(d) \leq T_0\})$
      fin
    fin
  fin
fin
si $V_{|\mathbf{N}|} = \emptyset$ **alors**
  il n'y pas de solution admissible
sinon
  $\min\{C(z) |\, z \in \mathbf{Z}\} \leftarrow \min\{c(v_{|\mathbf{N}|}) |\, m(v_{|\mathbf{N}|}) \leq m_0\}$
fin

En utilisant les triplets $(c(v), t(v), m(v))$, tous les chemins dans le graphe Γ correspondants aux solutions optimales du problème (6.27)-(6.30) peuvent être facilement retrouvés. Afin d'illustrer l'approche proposée, nous présentons un exemple de résolution d'un problème industriel.

6.1.4 Exemple industriel

La pièce qui nous servira d'exemple est présentée dans la Figure 6.2. Cette pièce comporte 5 trous à usiner. Les trois trous appartenant à la face « gauche » (T1, T2, T3) sont borgnes, ils ont une profondeur de 50 mm et des diamètres respectifs de 80 mm, 75 mm, et 70 mm. Le 4ème trou, situé sur la face « droite » (T4), est débouchant, il a un diamètre de 50 mm et une profondeur de 100 mm. Enfin, le 5ème trou, appartenant également à la face « droite » (T5), est borgne, il a un diamètre de 70 mm et une profondeur de 44 mm.

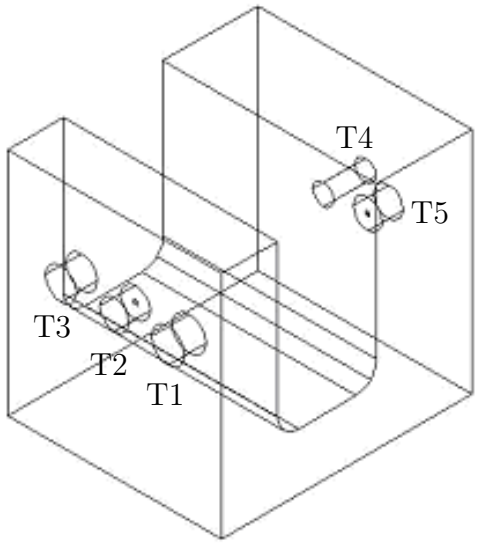


FIG. 6.2 – Pièce à usiner

Le Tableau 6.1 présente l'ensemble des opérations à réaliser.

TAB. 6.1 – Opérations de l'ensemble **N** et leurs paramètres

i	Opération	l_i	$v_{f1}(i)$	$v_{f2}(i)$
1	Percer H1 ∅75, 5	72	16,9	202,8
2	Aléser H1 ∅80	62,3	9,7	28,2
3	Percer H2 ∅70, 5	72	19	220
4	Aléser H2 ∅75	62,3	9,7	28,2
5	Percer H3 ∅65, 5	72	19,5	233,7
6	Aléser H3 ∅70	62,3	9,7	28,2
7	Percer H4 ∅50	114	25,5	306,2
8	Percer H5 ∅65, 5	64	19,5	233,7
9	Aléser H5 ∅70	55,3	12,4	42,3

Les contraintes d'inclusion pour cet exemple n'existent pas : $I^p = \emptyset$. Les autres contraintes sont respectivement recensées dans les tableaux suivants. Le Tableau 6.2 présente l'ensemble des contraintes de précédence.

TAB. 6.2 – Contraintes de précédence

Opération i	$PredD(i)$
1	$\emptyset$
2	1 3 5 7 8
3	$\emptyset$
4	1 3 5 7 8
5	$\emptyset$
6	1 3 5 7 8
7	$\emptyset$
8	$\emptyset$
9	1 3 5 7 8

Le Tableau 6.3 présente les contraintes d'exclusion au niveau des postes de travail.

TAB. 6.3 – Contraintes d'exclusion au niveau des postes de travail

Opération i	Opérations incompatibles sur un poste de travail
7	1
2 3 4 5 6 8	7

Le Tableau 6.1.4 présente l'ensemble des contraintes d'exclusion au niveau des blocs.

TAB. 6.4 – Contraintes d'exclusion au niveau des blocs

Opération i	Opérations incompatibles dans un bloc
7	1 2 3 8 9
8	1 2 3 4 5 6
9	1 2 3 4 5 6

Les paramètres caractérisant la machine à table mobile à concevoir sont les suivants : $T_0 = 4$ min, $\tau^a = 0.2$ min, $\tau^p = 0$, $\tau^b = 0$, $m_0 = 3$. Deux types d'unités d'usinage peuvent être utilisés : les unités horizontales placées soit à gauche (type 1) soit à droite (type 2) par rapport à la table mobile. Ces deux types peuvent être installés aussi bien sur le dernier poste de travail que sur n'importe quel autre. Alors, nous avons les contraintes suivantes $E^{c2} = \emptyset$ et $E^{c3} = \emptyset$.

Nous avons résolu ce problème sur un PC Pentium II 450 en 0,01 sec. En utilisant notre algorithme et les règles de dominance, nous avons obtenu le graphe Γ qui contient 19 nœuds et 51 arcs (voir la Figure 6.3). Notons que si les arcs dominés ne sont pas éliminés de l'ensemble $D(v)$, alors le graphe correspondant contient 47 nœuds et 211 arcs.

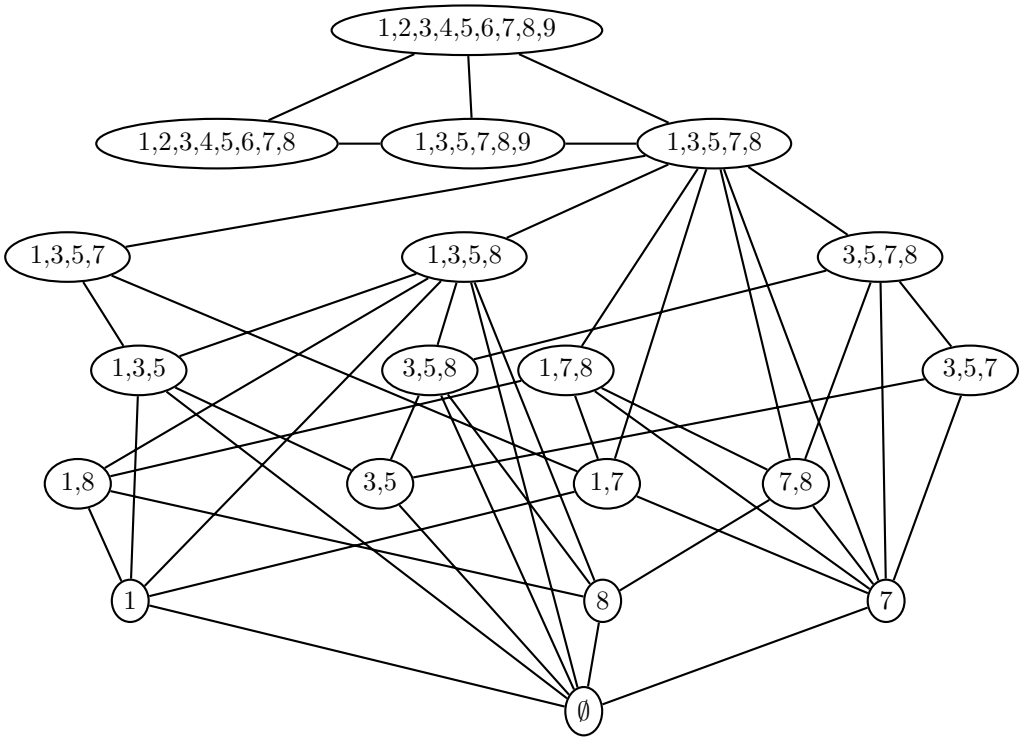


FIG. 6.3 – Graphe de solutions Γ

Le graphe présenté par la Figure 6.3 contient deux chemins optimaux : $\{\emptyset\}$, $\{1,3,5,8\}$, $\{1,3,5,7,8\}$, $\{1,2,3,4,5,6,7,8,9\}$ et $\{\emptyset\}$, $\{7\}$, $\{1,3,5,7,8\}$, $\{1,2,3,4,5,6,7,8,9\}$. La solution optimale correspondant au premier chemin est représentée dans le Tableau 6.5.

TAB. 6.5 – Solution optimale

Poste	Bloc	Opérations	$L(N_{kb})$	$v_f(N_{kb})$	$t^b(N_{kb})$
1	1	1 3 5	72	202,8	0,355
	2	8	64	233,7	0,27
2	1	7	114	306,2	0,37
3	2	2 4 6	62,3	28,2	2,2
	2	9	55,3	42,3	1,3

La configuration optimale de la machine à table mobile pour usiner la pièce de la Figure 6.2 comporte donc trois postes de travail. Sur le premier poste il y a deux unités d'usinage, une pour effectuer les opérations 1, 3, et 5, et une autre, monobroche, pour exécuter l'opération 8. Le deuxième poste est muni d'une seule unité d'usinage effectuant l'opération 7. Enfin, le troisième poste comporte deux unités d'usinage, dont la première est utilisée pour exécuter les opérations 2, 4, et 6 simultanément, et la deuxième réalise l'opération 9. Les paramètres de coupe sont répertoriés dans le Tableau 6.5.

6.1.5 Conclusion

Nous avons étudié le problème d'optimisation de la configuration des machines à table mobile. Étant donné que les problèmes de ce type n'atteignent pas une taille importante dans la pratique (par exemple, souvent $m_0 = 3$), nous avons proposé deux méthodes exactes pour leur résolution. En nous basant sur les résultats que nous avons obtenus dans [Guschinskaya et Dolgui, 2007*b*], nous pouvons supposer que l'approche par graphe aura les meilleures performances lors de la résolution des problèmes ayant un nombre assez important de contraintes d'exclusion et de précédence. Tandis que les problèmes avec un niveau moyen de contraintes seront plus rapidement résolus en utilisant le modèle MIP avec, par exemple, les logiciels d'optimisation ILOG Cplex, XPress MP, etc. Toutefois, une étude complémentaire est nécessaire afin d'évaluer les performances et la robustesse des deux approches. Cette étude fait partie de nos objectifs à court terme.

Dans la section suivante, nous présentons un autre type des systèmes d'usinage ayant également un mode parallèle d'activation de boîtiers, mais permettant de traiter plusieurs unités de pièce en même temps, il s'agit des machines à table circulaire pivotante.

6.2 Machines à table circulaire pivotante

6.2.1 Mode de fonctionnement

Nous présentons un exemple de telle machine en provenance de l'usine des systèmes d'usinage MZAL située à Minsk (Bélarus) dans la Figure 6.4.

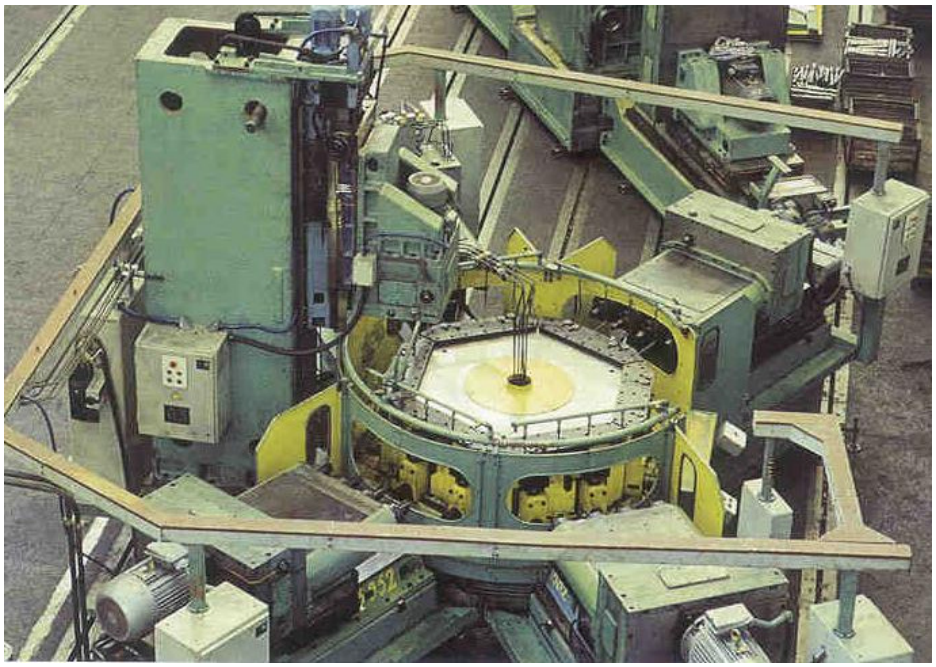


FIG. 6.4 – Une machine à table circulaire pivotante

Un système d'usinage de ce type est muni d'une table circulaire pivotante contenant plusieurs positions de travail. L'architecture de la machine permet de mettre des boîtiers horizontaux autour de la table circulaire et de regrouper tous les boîtiers verticaux dans un support au-dessus de la table. Un poste de travail correspond à une position de la table circulaire pivotante. Le premier poste (la position 0) est toujours utilisé pour les opérations de chargement\déchargement des pièces et aucune opération d'usinage ne lui est attribuée. En revanche, à tout autre poste de travail, nous pouvons affecter au plus deux unités d'usinage, dont une horizontale et une verticale. Si deux boîtiers sont installés sur un poste de travail, ils sont activés en même temps, par conséquent, deux faces de la pièce s'y trouvant peuvent être traitées simultanément, une horizontale et une verticale.

Le fait que le plateau tournant de la table est divisé en plusieurs positions de travail permet de traiter plusieurs unités de la pièce en même temps : une par position. Ainsi, tous les boîtiers de tous les postes de travail sont déclenchés en même temps : ils effectuent les opérations d'usinage sur la pièce présente sur la position de travail où ils sont installés. Même si les temps d'usinage peuvent varier d'un poste de travail à un autre, le déplacement de toutes les pièces se trouvant sur la machine se fait en même temps par la rotation de la table d'un pas. Ainsi, chaque pièce avance au poste de travail suivant. L'intervalle de temps entre deux rotations de la table est égal au cycle d'usinage, car après chaque rotation de la table une pièce est déchargée du dernier poste de travail et un nouveau brut est tout de suite chargé sur le premier poste. Il en découle que le boîtier ayant le temps d'usinage le plus long définit le temps de cycle de la machine.

Le problème d'optimisation de la configuration des machines à table circulaire pivotante consiste à déterminer le nombre de postes de travail et les boîtiers multibroches à installer sur chaque poste de travail pour effectuer toutes les opérations d'usinage, tout en respectant les contraintes technologiques et techniques données.

La différence entre ce problème et le problème considéré dans la section précédente réside dans le fait qu'il est possible de traiter sur une machine à table circulaire pivotante plusieurs unités de pièce en même temps, une par poste de travail. La différence entre ce problème et le problème considéré dans le chapitre précédent, réside dans le mode d'activation de boîtiers installés sur le même poste de travail. Dans le chapitre précédent, nous avons étudié les machines de transfert ayant le mode séquentiel d'activation de boîtiers, les machines à table circulaire pivotante utilisent le mode parallèle. Dans ce qui suit, nous présentons un modèle mathématique incorporant l'ensemble des contraintes à prendre en compte lors de la configuration d'une machine à table circulaire pivotante.

6.2.2 Modélisation mathématique

Prenant en compte le fait que les blocs d'un poste de travail sont activés simultanément, nous reprenons le modèle développé pour le problème d'optimisation de la configuration des machines à table mobile. Notre but maintenant est d'adapter ce modèle au cas des machines à table circulaire pivotante.

Sachant que les postes de travail et les boîtiers d'un poste de travail sont activés simultanément, le bloc qui demande plus de temps définit le temps de cycle T de la machine. Afin de respecter la cadence désirée, il faut que le temps de ce bloc ne dépasse pas le temps de

cycle objectif, c'est-à-dire la condition suivante doit être valide :

$$t(P) = \max\{t^b(N_{kr}) |\, r \in \{1, ..., r_k\}, k \in \{1, ..., m\}\} \; + \tau^p \leq T_0 \tag{6.31}$$

où $t^b(N_{kr})$ est calculé, par exemple, en utilisant (6.1).

Comme les machines à table mobile, les machines à table circulaire pivotante ont le mode parallèle d'activation de boîtiers sur chaque poste de travail. La différence entre les deux types de machines réside dans la disposition de boîtiers. Dans le cas des machines à table circulaire pivotante, sur tout poste de travail, il est possible d'installer au plus deux unités d'usinage, dont une horizontale et une verticale. Ainsi le choix de boîtiers à installer sur un poste de travail est plus restreint. Cette restriction nous permet de simplifier le modèle développé pour le problème d'optimisation de la configuration des machines à table mobile. Afin de vérifier qu'un poste de travail d'une machine à table circulaire pivotante ne comporte pas deux boîtiers ayant le même type, nous attribuons à chaque opération $j \in \mathbf{N}$ le paramètre ς_j qui détermine la face sur laquelle l'opération j doit être exécutée : si $\varsigma_j = 1$, alors l'opération j doit être exécutée par un boîtier vertical, si $\varsigma_j = 2$, alors l'opération j doit être exécutée par un boîtier horizontal. De plus, les ensembles N^r, contentant les opérations à exécuter sur la chaque face $r \in \{1, 2\}$, sont déterminés au préalable.

Notations

Nous présentons le modèle mathématique du problème d'optimisation de la configuration des machines à table circulaire pivotante en utilisant des variables de décision mixtes (réelles et binaires). Un tel modèle peut être facilement implémenté en recourant à un logiciel muni d'une bibliothèque d'optimisation.

Prenant en compte le fait que les boîtiers sont activés en parallèle sur chaque poste de travail, il est impossible de numéroter les blocs d'opérations selon l'ordre de leur exécution en utilisant un seul indice cumulé. De plus, nous devons distinguer les faces de la pièce traitées pour veiller au respect de contraintes de l'affectation des blocs au même poste. Alors, nous utilisons deux indices pour déterminer les affectations d'opérations, à savoir :

- $k = 1, 2, ..., m_0$, est l'indice utilisé pour les postes de travail ; il est à noter que la position 0, qui est uniquement utilisée pour les opérations de chargement\déchargement des pièces, n'est pas comptée.
- $r = 1, 2$, est l'indice utilisé pour les boîtiers de chaque poste de travail : $r = 1$ correspond à un boîtier vertical et $r = 2$ correspond à un boîtier horizontal.

En utilisant ces indices, nous introduisons les variables de décision suivantes.

Variables de décision

X_{jk} est une variable de décision binaire, qui est utilisée pour déterminer l'affectation des opérations $j \in \mathbf{N}$:

$$X_{jk} = \begin{cases} 1 & \text{si l'opération } j \text{ est affectée au poste de travail } k \\ 0 & \text{sinon} \end{cases}$$

Pour chaque opération $j \in \mathbf{N}$, nous connaissons la face de la pièce sur laquelle elle doit être exécutée, par conséquent, son affectation au poste k définit explicitement le bloc dans lequel elle doit être ajoutée.

Y_{kr} est une variable binaire auxiliaire, elle est utilisée pour calculer le nombre de boîtiers dans une solution lors de l'évaluation de la fonction objectif :

$$Y_{kr} = \begin{cases} 1 & \text{si le boîtier } r \text{ est installé sur le poste de travail } k \\ 0 & \text{sinon} \end{cases}$$

Rappelons que $Y_{k1} = 1$ signifie qu'un boîtier vertical est installé sur le poste de travail k, et $Y_{k2} = 1$ représente l'installation d'un boîtier horizontal sur le poste de travail k.

Z_k est une variable binaire auxiliaire, elle est utilisée pour calculer le nombre de postes de travail dans une solution lors de l'évaluation de la fonction objectif :

$$Z_k = \begin{cases} 1 & \text{si le poste de travail } k \text{ existe} \\ 0 & \text{sinon} \end{cases}$$

Fonction objectif

La fonction objectif (6.32) calcule le coût de la machine qui est constitué de deux composants : le premier correspond au coût des postes, le deuxième au coût de tous les blocs.

$$Minimiser\ C(S) = C_1 \sum_{k=1}^{m_0} Z_k + C_2 \sum_{k=1}^{m_0} (Y_{k1} + Y_{k2}) \tag{6.32}$$

Dans ce qui suit, nous présentons l'ensemble de contraintes de ce modèle.

Relations entre les variables de décision

L'équation (6.33) signifie qu'un bloc n'existe que s'il y a des opérations qui lui sont assignées.

$$Y_{kr} \geq X_{jk}; k = 1, 2 \ldots, m_0; r = 1, 2; j \in N^r \tag{6.33}$$

L'équation (6.34) assure que le poste k n'existe que s'il y a des opérations qui sont affectées à un de ses blocs.

$$Z_k \geq Y_{kr}; k = 1, \ldots, m_0; r = 1, 2 \tag{6.34}$$

Contraintes d'occurrence

L'équation (6.35) impose que chaque opération de l'ensemble $\mathbf{N}$ soit affectée à un et un seul poste de travail et, par conséquent, à un et un seul bloc.

$$\sum_{k=1}^{m_0} X_{jk} = 1, j \in \mathbf{N} \tag{6.35}$$

Contraintes de précédence

Les contraintes de précédence sont exprimées de la façon suivante :

$$\sum_{k=1}^{m_0} kX_{ik} \leq \sum_{k=1}^{m_0} kX_{jk}, j \in \mathbf{N}, i \in PredD(j) \tag{6.36}$$

Contraintes de compatibilité entre les opérations

L'équation (6.37) introduit les contraintes d'inclusion :

$$X_{ik} - X_{jk} = 0; j, i \in e, e \in I^p; k = 1, 2 \ldots, m_0 \tag{6.37}$$

L'équation (6.38) traduit les contraintes d'exclusion au niveau des blocs :

$$\sum_{j \in e} X_{jk} \leq |e| - 1; e \in E^b; r = 1, 2; k = 1, 2 \ldots, m_0 \tag{6.38}$$

L'équation (6.39) exprime les contraintes d'exclusion au niveau des postes de travail :

$$\sum_{j \in e} X_{jk} \leq |e| - 1; e \in E^p; k = 1, 2 \ldots, m_0 \tag{6.39}$$

Contrainte sur le temps de cycle de la machine

Comme pour le problème d'optimisation considéré dans le Chapitre 4, pour chaque opération $j \in \mathbf{N}$, nous calculons le paramètre t_j en utilisant (4.15). De plus, pour chaque paire d'opérations pouvant être affectées au même bloc ($\{i, j\} \notin E^b$), nous calculons le paramètre t_{ij} en utilisant (4.16). En recourant aux paramètres t_j, t_{ij}, nous évitons la perte de la linéarité du modèle.

Rappelons que, pour ce type de machines, le temps de cycle est défini par le bloc le plus lent. Dans ce cas, la contrainte sur le temps de cycle de la machine s'exprime par les équations (6.40)-(6.41). Plus précisément, ces équations calculent la durée d'un bloc s'il est constitué d'une seule opération (6.40) ou d'un ensemble d'opérations (6.41) et assurent que ce temps ne dépasse pas le temps de cycle objectif.

$$t_j X_{jk} \leq T_0, k = 1, 2 \ldots, m_0; j \in \mathbf{N} \tag{6.40}$$

$$\begin{gathered} t_{ij}(X_{ik} + X_{jk} - 1) \leq T_0, \\ k = 1, 2 \ldots, m_0; r = 1, 2; i, j \in N^r; i < j; \{i, j\} \notin (E^p \cup E^b) \end{gathered} \tag{6.41}$$

Le modèle présenté peut être facilement implémenté et résolu en recourant à un logiciel muni d'une bibliothèque d'optimisation.

6.2.3 Conclusion

Dans ce chapitre, nous avons étudié le problème d'optimisation de la configuration des systèmes d'usinage à boîtiers multibroches de deux types différents.

En premier lieu, nous nous sommes intéressés aux machines à table mobile. La particularité des machines de ce type réside dans leur mode de transport de pièce, car une seule unité de pièce est traitée sur la machine dans chaque cycle d'usinage. Une autre différence de ces machines par rapport aux machines de transfert, considérées dans les Chapitres 4 et 5, est due au mode d'activation de boîtiers qui est parallèle pour les machines à table mobile et séquentiel pour les machines de transfert.

Nous avons proposé un modèle mathématique en variables mixtes du problème d'optimisation de la configuration des machines à table mobile. Ce problème peut donc être résolu avec un logiciel d'optimisation, par exemple, ILOG Cplex. Cependant, l'étude des propriétés des problèmes industriels, nous a permis de constater que dans la pratique, les problèmes de ce type n'atteignent pas la taille très importante et sont par leur nature riches en contraintes d'exclusion aussi bien au niveau des blocs qu'au niveau des postes de travail. Pour cette raison, nous avons développé une approche dédiée utilisant un modèle de graphe qui est une approche exacte, basée sur la recherche des chemins les plus courts sous contraintes dans un graphe spécialement conçu. Ce graphe est construit en explorant toutes les affectations d'opérations possibles et en tenant compte des contraintes du problème. De manière générale, la taille de ce graphe croît de manière exponentielle avec le nombre d'opérations nécessaires pour l'usinage de la pièce. Cependant, de nombreuses contraintes d'exclusion et de précédence qui caractérisent les problèmes réels de ce type, ainsi que les règles de dominance que nous avons développées, permettent de réduire la taille du graphe et le temps de sa construction. Puisque l'algorithme construit le graphe et résout le problème de plus court chemin en même temps, il est possible d'obtenir toutes les solutions optimales avec un temps de calcul relativement court. Cela a été notamment montré par l'étude d'un exemple industriel.

En deuxième lieu, nous avons étudié le problème d'optimisation de la configuration des machines à table circulaire pivotante et nous avons développé, pour ce problème, un modèle de programmation linéaire en variables mixtes. Ce problème peut donc être résolu avec un logiciel d'optimisation (ILOG Cplex, par exemple).

Les perspectives de l'étude des problèmes d'optimisation présentés dans ce chapitre relèvent de la nécessité d'une analyse plus approfondie des performances des méthodes exactes proposées, notamment leur dépendance de la structure du problème à résoudre. Cette analyse permettrait également d'évaluer la nécessité du développement de méthodes approchées pour ces problèmes.

Chapitre 7

Un prototype de système d'aide à la décision

La connaissance s'acquiert par l'expérience, tout
le reste n'est que de l'information.

Albert Einstein

Dans ce chapitre, nous présentons un prototype de système d'aide à la décision pour la conception de systèmes d'usinage à boîtiers multibroches, dans lequel nous avons intégré les modèles mathématiques et les méthodes d'optimisation présentés dans les chapitres précédents. Ce système a été baptisé MTE (Machine Tools Engineering). Dans un premier temps, la structure générale de ce système est présentée, ensuite le rôle et les fonctions assurées par chaque module le constituant sont énoncés à travers une étude de cas. Parmi les modules exposés, se trouvent ceux qui permettent la modélisation de la pièce à fabriquer, la planification de son processus d'usinage, et la configuration d'un système d'usinage à boîtiers multibroches. Les modules développés pour l'agencement et le calcul du coût des systèmes d'usinage à boîtiers multibroches ne feront pas partie de la présentation. Le lecteur intéressé peut trouver leur description dans [Dolgui *et al.*, 2005*b*; Dolgui *et al.*, 2006*c*; Dolgui *et al.*, 2007*a*].

7.1 Aide à la décision en conception des systèmes de fabrication

La conception des systèmes de fabrication représente un champ d'application très intéressant pour les méthodes d'aide à la décision. Par conséquent, il existe de nombreux systèmes d'aide à la décision développés pour étayer la prise de décisions cohérentes et efficaces durant la conception d'un produit et de son système de fabrication. Néanmoins, des études bibliographiques menées, par exemple dans [Derras, 1998; Derigent, 2005], concluent qu'il n'existe pas d'approche universelle mais une pléthore de travaux répondant à des aspects très précis dans des contextes différents [Lombard, 2006].

De manière générale, les composants suivants doivent être articulés dans un tel outil d'aide à la décision [Brown, 2004; Grieves, 2005; Stark, 2005] :

- des techniques d'aide à la décision,
- des modèles mathématiques et des méthodes d'optimisation développés en tenant compte de la problématique considérée,
- une interface conviviale pour réaliser des échanges d'informations fructueux entre l'utilisateur et le système.

L'intervention de l'utilisateur est obligatoire pour tous les choix importants et pour assurer le lien entre les fonctions du système.

Du côté de fonctionnalité, une bonne coordination de la conception du produit et de la conception du processus de fabrication doit être mise en place. De plus, un tel logiciel doit offrir pour les spécialistes de différents métiers la possibilité de travailler simultanément au sein d'un même environnement et sur le même projet.

Pour satisfaire ces besoins, les outils commerciaux, comme par exemple EM Machining (Technomatix), Prismatic Machining Assistant v2 ou Delmia Process Engineer (DASSAULT Systèmes), s'appuient sur des bases de données puissantes, contenant des informations sur Produit/Processus/Ressources. En même temps, les travaux académiques actuels portent sur le développement de nouveaux concepts pour faire évoluer les outils de représentation de ces informations.

Rappelons que le processus de la conception des systèmes d'usinage à boîtiers multibroches comporte trois phases, à savoir : la détermination des opérations d'usinage, la configuration du système et l'évaluation de sa qualité et de son coût. Ces phases étant présentées en détail dans le Chapitre 2, ici nous nous intéressons plutôt à l'implémentation informatique des méthodes aidant l'utilisateur à prendre des décisions efficaces et cohérentes durant les deux premières phases.

7.2 Structure du système MTE

Pour prendre en compte les particularités de la conception des systèmes d'usinage à boîtiers multibroches, un prototype de système d'aide à la décision a été développé. Ce système a été baptisé MTE (Machine Tools Engineering). Il comporte différents modules qui permettent la modélisation de la pièce à fabriquer, la planification de son processus d'usinage, la configuration du système d'usinage à boîtiers multibroches, son agencement et le calcul de son coût. La structure de ce système est présentée dans la Figure 7.1.

Les éléments essentiels de ce système d'aide à la décision sont les suivants :

- une base de données puissante comportant l'ensemble des informations concernant les entités, leurs processus de fabrication, les paramètres des équipements utilisés, etc ;
- un système expert comprenant un ensemble de règles « métier » employées à différentes étapes de conception ;
- des méthodes de modélisation mathématique et d'optimisation pour assurer la prise de solutions efficaces ;
- une interface conviviale.

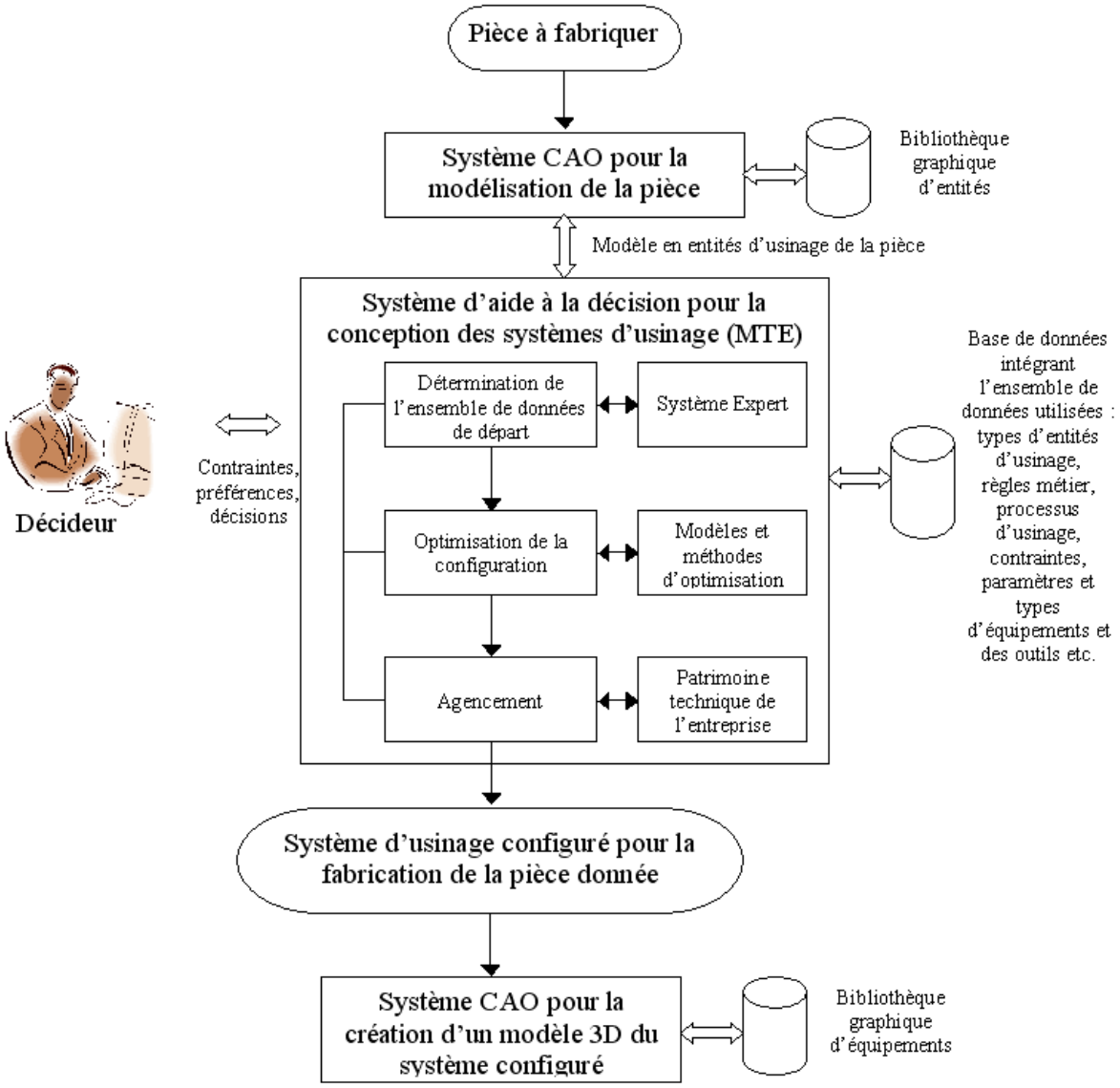


FIG. 7.1 – Structure du système d'aide à la décision MTE

Dans ce qui suit, nous énonçons le rôle de chaque élément à travers une étude de cas. Cette présentation s'axe sur l'application des méthodes d'optimisation développées dans ce mémoire. Notre but principal est d'illustrer leur utilisation dans des conditions industrielles, ainsi nous n'entrons pas dans les détails des règles « métier » utilisées.

7.3 Étude de cas

Les données que nous utilisons pour cette étude de cas nous ont été fournies par notre partenaire industriel, PSI SCEMM (Saint-Étienne), considéré comme leader sur le marché

français des systèmes d'usinage. Pour des raisons de confidentialité, certaines informations ont été modifiées. Nous avons comme pièce à fabriquer un carter de moteur, dont les faces culasse (à droite) et vilebrequin (à gauche) sont présentées dans la Figure 7.2.

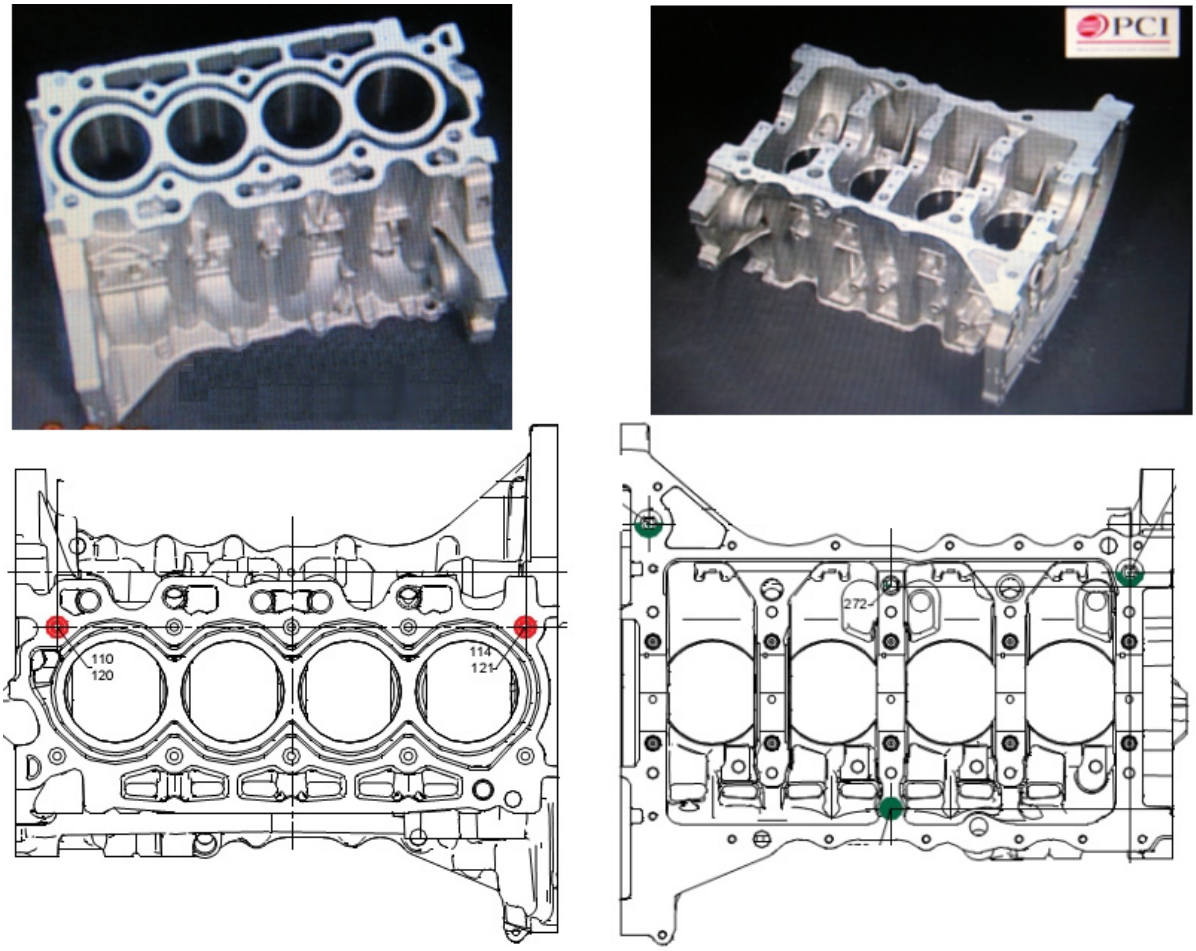


FIG. 7.2 – Pièce à usiner

Nous nous sommes fixé l'objectif de construire une machine de transfert (Chapitre 4) pour l'usinage des éléments suivants :

- face culasse : l'ébauche des trous fixation culasse et la finition des trous de départ usinage ;
- face vilebrequin : l'ébauche et la finition des trous fixation paliers, goupille et des trous de départ usinage.

L'impératif était de trouver une configuration de la machine de transfert qui comporte un nombre minimum possible de boîtiers et de postes de travail tout en respectant l'ensemble de contraintes.

La première étape consiste à saisir les propriétés générales de la pièce à usiner et du système d'usinage à concevoir. Les données de départ sont saisies moyennant la boîte de dialogue « Généralités » présentée dans la Figure 7.3.

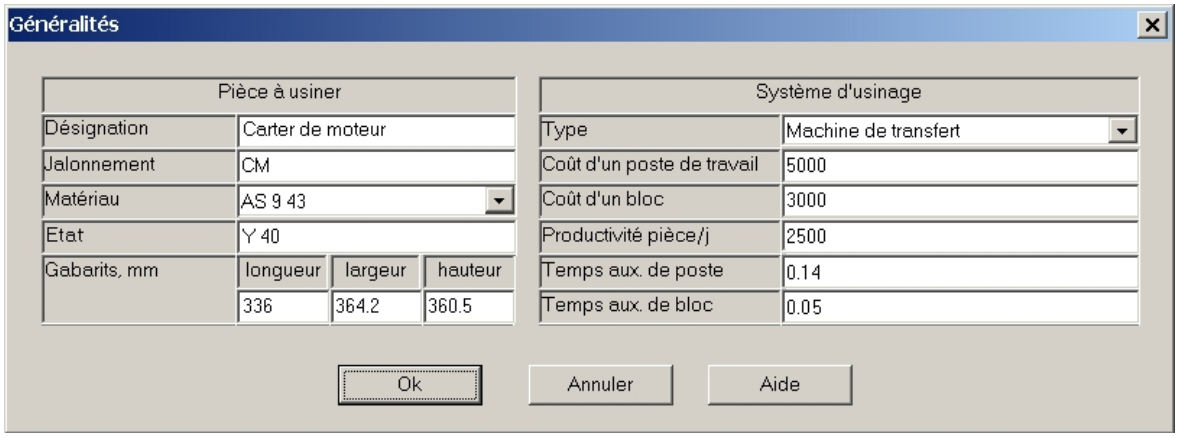


FIG. 7.3 – Boîte de dialogue pour saisir les propriétés générales de la pièce et du système d'usinage à concevoir

Avec les notations que nous utilisons, nous retrouvons dans cette boîte de dialogue : $T_0 = 0,38$ min, $\tau^p = 0,14$ min, $\tau^b = 0,05$ min, $C_1 = 5000$, $C_2 = 3000$. Nous avons fixé d'autres paramètres via d'autres boîtes de dialogue, de la manière suivante : $m_0 = 10$, $n_\exists = 2$, $n_0 = 2$. Ces paramètres servent de point de départ pour la conception du processus de fabrication, ils fournissent également certaines contraintes technologiques.

Comme nous l'avons énoncé dans le Chapitre 2, le processus de la conception du système d'usinage commence par la modélisation de la pièce à usiner.

7.4 Modélisation de la pièce par entités

Rappelons qu'afin de définir les actions élémentaires d'usinage nécessaires pour assurer le passage de la pièce brute à la pièce finie, il convient de passer par une étape charnière consistant en la modélisation de la pièce par entités. Cette modélisation est effectuée en général à l'aide d'un logiciel de type CAO. Comme nous l'avons déjà expliqué, les logiciels CAO commerciaux ne construisent pas de modèle de pièce en utilisant des entités. Afin d'obtenir de tels modèles, il convient d'abord de développer une taxinomie des entités à utiliser et ensuite de mettre en place des outils permettant de construire des modèles de pièces en entités moyennant un logiciel CAO (les différentes approches développées dans la littérature ont été présentées dans la Section 2.3.1).

Même s'il n'existe pas une seule et unique taxinomie des entités valable quelle que soit l'activité de fabrication considérée, il est cependant possible de recenser un certain nombre de caractéristiques géométriques et technologiques génériques à chaque entité. D'après [Shah et Mantyla, 1995], ces caractéristiques peuvent être classées en deux catégories : caractéristiques intrinsèques (propres à l'entité) et extrinsèques (liées à l'entité et à son environnement). D'autres travaux ont contribué à dresser une liste des caractéristiques de chaque catégorie, notamment [Brissaud, 1992]. Alors, les **caractéristiques intrinsèques** suivantes sont distinguées :

- le type de l'entité ;
- les paramètres décrivant la forme géométrique associée à l'entité ;
- l'état brut de l'entité ;
- la qualité technologique intrinsèque de l'entité ;
- les paramètres de matériaux.

Les **caractéristiques extrinsèques** d'entité jouent un rôle important lors de la formulation des contraintes de fabrication. Elles sont les suivantes :

- le positionnement de l'entité dans la pièce ;
- les interactions topologiques de l'entité avec les entités voisines ;
- les informations liées à son processus de fabrication.

Dans le système MTE, l'approche suivante est utilisée. Le système est construit autour du logiciel Autodesk Mechanical Desktop développé par la société Autodesk. Ainsi nous pouvons utiliser les outils graphiques de ce logiciel CAO pour la création des objets 3D. Pour faciliter la modélisation de pièces, un ensemble des entités les plus utilisées a été élaboré, le Tableau 7.5 en présente une partie. En utilisant les représentations graphiques et sémantiques de ces entités, nous avons développé une bibliothèque graphique des entités génériques qui a été implémentée dans le logiciel Autodesk Mechanical Desktop. Ainsi la modélisation d'une pièce par entités consiste en la sélection successive dans cette bibliothèque de types d'entités appropriés et en la spécification de leurs paramètres et leurs coordonnées. L'utilisation de cette bibliothèque permet de créer rapidement et aisément un modèle 3D de la pièce, et d'obtenir sa représentation en entités. La Figure 7.4 présente une boîte de dialogue à utiliser pour ajouter dans le modèle de la pièce une entité de type « trou cylindrique ».

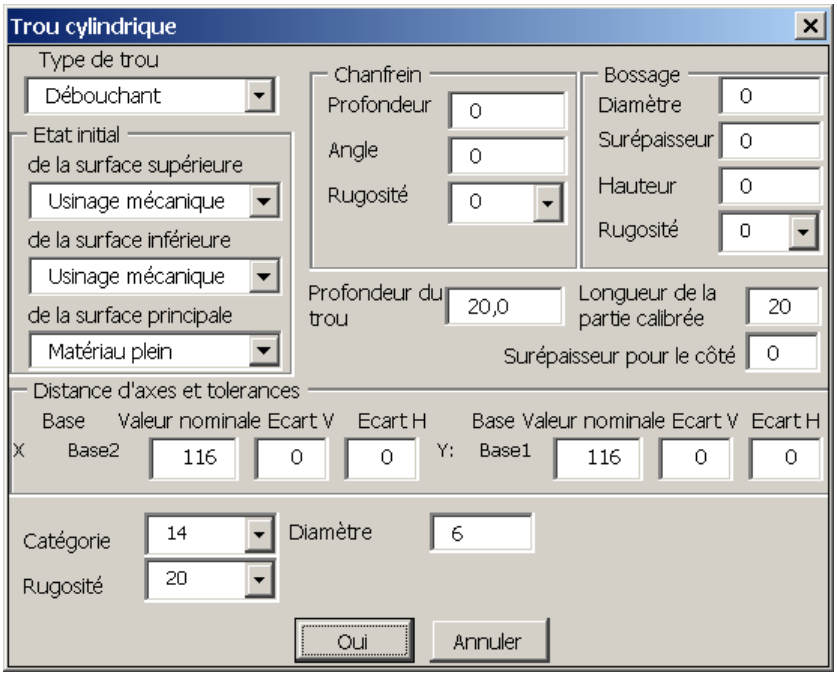


FIG. 7.4 – Boîte de dialogue pour créer une entité

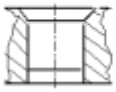
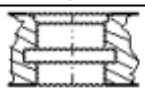
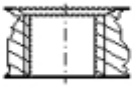
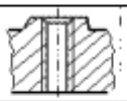
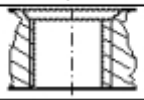
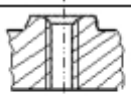
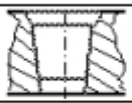
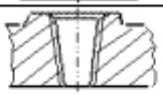
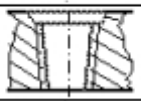
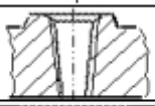
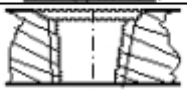
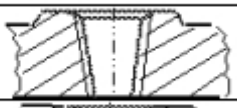
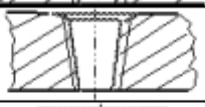
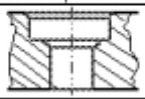
Numéro	Entité	Dessin
1	Trou cylindrique	
2	Trou cylindrique avec un bossage	
3	Trou cylindrique avec une rainure	
4	Trou cylindrique ayant un filet métrique	
5	Trou cylindrique ayant un filet métrique, avec un bossage	
6	Trou cylindrique ayant un filet au pas du gaz	
7	Trou cylindrique ayant un filet au pas du gaz, avec un bossage	
8	Trou conique avec un bossage	
9	Trou conique	
10	Trou conique ayant un filet anglais conique, avec un bossage	
11	Trou conique ayant un filet anglais conique	
12	Trou ayant un filet métrique conique, avec un bossage	
13	Trou ayant un filet métrique conique	
14	Trou ayant un filet conique au pas du gaz, avec bossage	
15	Trou ayant un filet conique au pas du gaz	
16	Trou cylindrique à deux échelons	

FIG. 7.5 – Quelques types d'entités utilisés

Pour la modélisation de la pièce dont nous avons disposée pour notre étude, nous avons notamment utilisé les deux types d'entités présentés dans la Figure 7.6, à savoir : « trou cylindrique à deux niveaux » (à gauche) et « trou cylindrique à trois niveaux » (à droite).

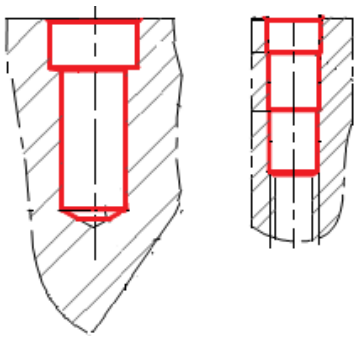


FIG. 7.6 – Types d'entités utilisées pour la modélisation de la pièce

Puisque l'Autodesk Mechanical Desktop est un outil commercial, une voie ad hoc a été aussi prévue pour les cas où son utilisation serait impossible. Dans une telle situation, l'utilisateur peut créer un modèle virtuel de la pièce en entités en spécifiant les entités et leurs coordonnées dans notre système sans utiliser l'Autodesk Mechanical Desktop.

Rappelons que la modélisation d'une pièce en entités se fait essentiellement en vue de pouvoir obtenir de manière quasi-automatique l'ensemble des processus d'usinage nécessaires pour fabriquer cette pièce. Dans ce qui suit, nous allons présenter l'approche mise en œuvre dans le système MTE pour accomplir cette tache.

7.5 Détermination des processus d'usinage des entités

Cette étape consiste à établir un processus d'usinage pour chaque entité, c'est-à-dire à définir, pour chaque entité, son processus de fabrication : les opérations, leur ordre et leurs caractéristiques (le régime d'usinage, l'outil de coupe et ses paramètres). Dans la pratique, pour chaque type d'entité, il existe souvent un ensemble des processus possibles. Le choix du processus à appliquer doit être effectué en fonction des critères géométriques, topologiques, technologiques, économiques et temporels reflétant le contexte d'usinage de chaque pièce. Afin de tenir compte de ces critères, les informations saisies par l'utilisateur sont analysées, comme par exemple les paramètres des entités et de la pièce usinée. Ces informations sont traitées à l'aide des connaissances issues du savoir-faire et de l'application des règles « métier ». Dans le système MTE, le choix d'un processus d'usinage est effectué de la façon présentée schématiquement dans la Figure 7.7.

Selon les paramètres des entités et de la pièce usinée (matériau, dimensions, productivité exigée, précision et rugosité d'usinage, etc.), le logiciel choisit tous les processus possibles pour l'usinage d'une entité à partir d'une base de données. Les processus sélectionnés sont ensuite classés en utilisant une matrice de préférence. L'utilisateur (décideur) peut accepter le processus classé en tête, le modifier ou en choisir un autre.

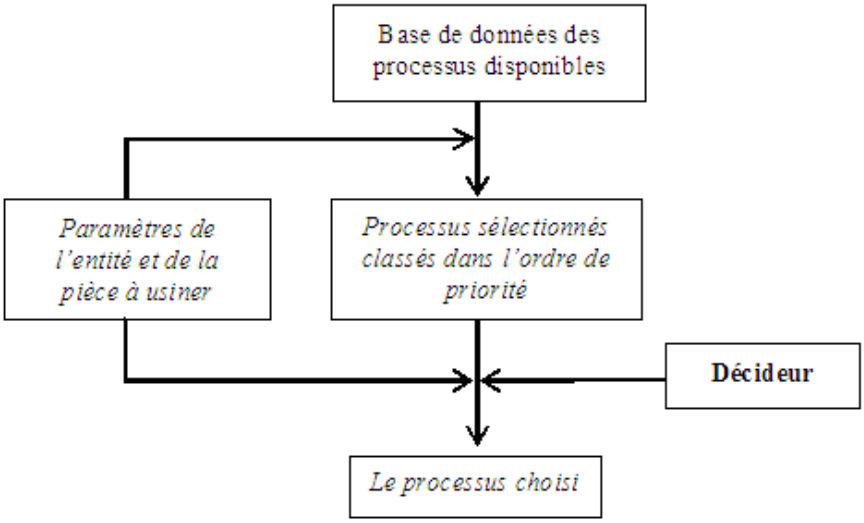


FIG. 7.7 – Choix d'un processus d'usinage pour une entité

Les processus d'usinage pour les entités dont les types ne sont pas recensés dans la base de données peuvent être formés par l'utilisateur en corrigeant les processus présents dans la base de données du système MTE ou en entrant directement les transformations technologiques nécessaires Les processus ainsi obtenus peuvent être ensuite sauvegardés dans la base de données du système. La figure 7.8 illustre une boîte de dialogue pour l'élaboration d'un processus d'usinage pour un trou cylindrique.

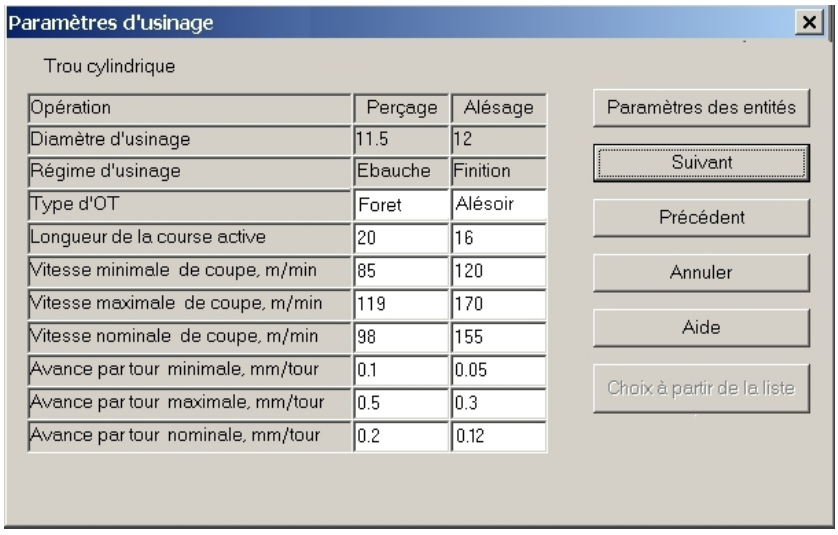


FIG. 7.8 – Boîte de dialogue pour établir le processus d'usinage d'un trou cylindrique

Dans le Tableau 7.1 sont recensées les opérations et leurs paramètres d'usinage (l'indice

« 0 » signifie « la valeur conseillée de l'intervalle des valeurs admissibles ») qui ont été choisis par le système MTE pour la réalisation des usinages nécessaires pour notre exemple. Pour déterminer la configuration de la machine de transfert réalisant la pièce donnée et son coût, il faut affecter ces opérations à des boîtiers et à des postes de travail. Pour distinguer les affectations admissibles et non, les contraintes suivantes sont utilisée : les contraintes de précédence, d'inclusion et d'exclusion au niveau des blocs et des postes de travail. Les règles « métier », employées à l'étape du choix des processus d'usinage, induisent plusieurs types de contraintes qui peuvent être établies automatiquement par le système. Dans ce qui suit, nous présentons le processus de leur établissement au sein du système MTE.

Ops	Description	l_i	$v_{c0}(i)$	n_i	$f_{z0}(i)$	z	$f_{n0}(i)$	$v_{f0}(i)$
1	Fraisage ébauche face culasse ∅ 250	440	2356	3000	0,1	12	1,2	3600
2-9	Lamage trou fixation culasse ∅ 12,64	45	163	4100	0,12	2	0,24	984
10-11	Alésage ébauche départ usinage ∅ 12	45	155	4100	0,12	2	0,24	984
12-13	Alésage ébauche départ usinage ∅ 12,5	28,4	161	4100	0,12	2	0,24	984
14-15	Alésage ébauche départ usinage ∅ 13,5	13,4	174	4100	0,12	2	0,24	984
16-17	Alésage finition départ usinage ∅ 12,5	45	161	4100	0,12	2	0,24	492
18-19	Alésage finition départ usinage ∅ 13	28,4	167	4100	0,12	2	0,24	492
20-21	Alésage finition départ usinage ∅ 14	13,4	180	4100	0,12	2	0,24	492
22	Fraisage finition face vilebrequin ∅ 400	570	2765	2200	0,125	20	2,5	5500
23-32	Alésage fixation paliers ∅ 8,43	50	119	4500	0,1	3	0,3	1350
33-42	Perçage ébauche goupille orientation carter chapeau ∅ 6	20	85	4500			0,12	540
43-44	Perçage ébauche départ usinage face vilebrequin ∅ 11,5	20	98	2700			0,2	540
45-54	Alésage fixation paliers ∅ 10,59	6,3	150	4500	0,1	3	0,3	1350
55-56	Alésage finition départ usinage face vilebrequin ∅ 12	16	155	4100	0,12	1	0,12	492
57-66	Taraudage + Detaraudage fixation paliers M $9 \times 1,25$	$2 \cdot 44$	40	1415			1,25	1769

TAB. 7.1 – Ensemble d'opérations et leurs paramètres

7.6 Établissement des contraintes

La première source de contraintes d'exclusion est l'emplacement des opérations. Par exemple, l'affectation de deux opérations au même boîtier multibroche est seulement possible si la distance entre les centres des trous correspondants sont supérieures à la valeur minimum possible permettant d'assurer la fixation des broches comportant les outils correspondants. En outre, les distances entre les centres de trous à usiner par le même boîtier doivent convenir aux gabarits des boîtiers multibroches disponibles.Pour le cas des machines de transfert, les opérations qui concernent des faces différentes de la pièce ne peuvent être exécutées ni par le même boîtier multibroche ni sur le même poste de travail si les postes de travail ne possèdent pas de dispositif pour faire tourner la pièce (ce qui est souvent le cas dans la pratique).

Les paramètres d'usinage caractérisant les opérations constituent une deuxième source de contraintes. Chaque opération est associée à une phase d'usinage et les opérations appartenant à des phases différentes ne peuvent pas être effectuées par le même boîtier multibroche. Pour cette raison, au moment du choix de processus d'usinage pour les entités, toutes les opérations faisant partie des processus choisis sont classées par le système MTE en 4 groupes successifs : ébauche, demi-finition, finition et opérations de taraudage. La succession technologique de ces phases d'usinage permet de créer automatiquement des contraintes de précédence entre les opérations. En outre, aucune opération d'un groupe ne peut être exécutée par le même boîtier multibroche avec toute autre opération appartenant à un groupe différent, ce qui permet d'établir de façon automatique des contraintes d'exclusion au niveau des blocs. Tenant compte du fait que les opérations i et j telles que $v_{f2}(i) \leq v_{f1}(j)$ ne peuvent pas être exécutées par le même boîtier multibroche, les paramètres v_{f1} et v_{f2} des opérations sont analysés en vue de faire surgir d'autres contraintes d'exclusion au niveau des blocs. En vue de déterminer les contraintes d'inclusion, les tolérances de position associées aux opérations sont analysées, car elles représentent la cause principale de ce type de contraintes.

Dans ce qui suit, nous présentons les contraintes établies par le système MTE pour notre exemple : les contraintes d'exclusion au niveau des postes de travail (Figure 7.9) et au niveau des blocs (Tableau 7.2), ainsi que les contraintes de précédence (Figure 7.10).

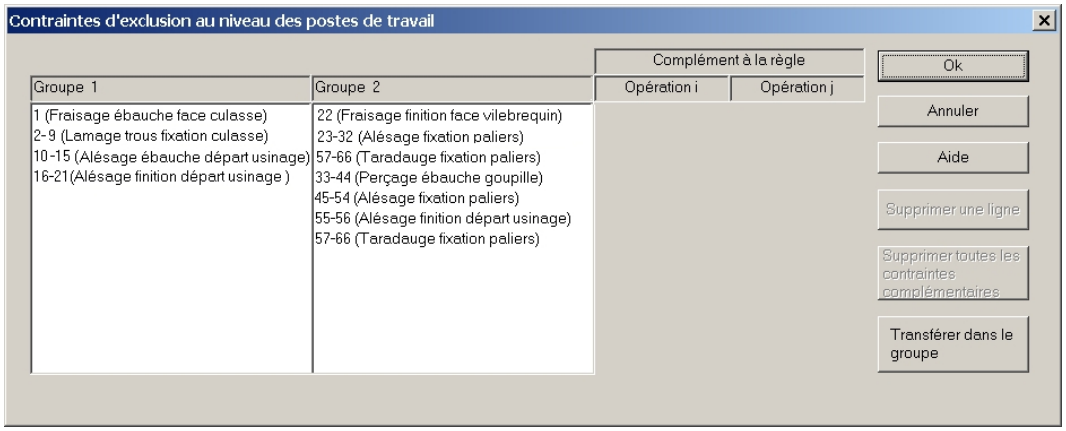


FIG. 7.9 – Définition des contraintes d'exclusion au niveau des postes de travail

Opération i	Opérations incompatibles dans un bloc
1	2-66
2-15	1, 16-66
16-21	1-15, 22-66
22	1-21, 23-66
23	1-22, 33, 55-66
24	1-22, 34, 55-66
25	1-22, 35, 55-66
26	1-22, 36, 55-66
27	1-22, 37, 55-66
28	1-22, 38, 55-66
29	1-22, 39, 55-66
30	1-22, 40, 55-66
31	1-22, 41, 55-66
32	1-22, 42, 55-66
33	1-22, 23, 45, 55-66
34	1-22, 24, 46, 55-66
35	1-22, 25, 47, 55-66
36	1-22, 26, 48, 55-66
37	1-22, 27, 49, 55-66
38	1-22, 28, 50, 55-66
39	1-22, 29, 51, 55-66
40	1-22, 30, 52, 55-66
41	1-22, 31, 53, 55-66
42	1-22, 32, 54, 55-66
43-44	1-22, 45-54, 65-66
45	1-22, 33, 55-66
46	1-22, 34, 55-66
47	1-22, 35, 55-66
48	1-22, 36, 55-66
49	1-22, 37, 55-66
50	1-22, 38, 55-66
51	1-22, 39, 55-66
52	1-22, 40, 55-66
53	1-22, 41, 55-66
54	1-22, 42, 55-66
55-56	1-54, 57-66
57-66	1-56

TAB. 7.2 – Contraintes d'exclusion au niveau des blocs

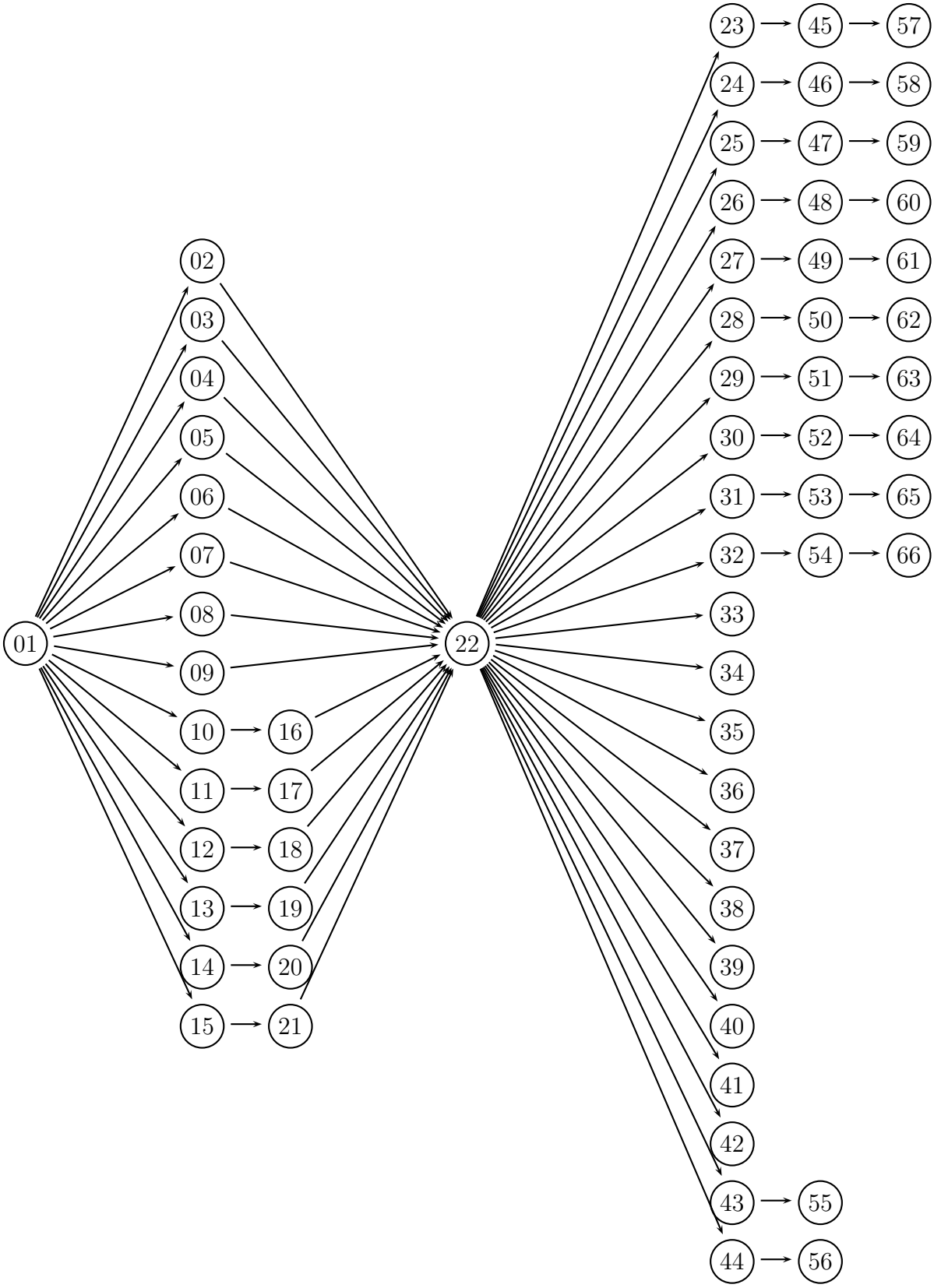


Fig. 7.10 – Graphe de précédence

La Figure 7.9 montre la boîte de dialogue permettant la définition des contraintes d'exclusion au niveau des postes de travail. Selon les contraintes établies par le système MTE, aucune opération appartenant au groupe 1 ne peut être affectée au même poste de travail avec toute opération du groupe 2. Puisqu'il est impossible de prévoir tous les cas de figure possibles, l'utilisateur peut introduire des contraintes complémentaires ou étendre les contraintes proposées. Pour cette fin, la section « Complément à la règle » est prévue.

Après l'étape d'établissement de contraintes, l'utilisateur peut procéder à l'étape de résolution du problème d'optimisation de la configuration du système d'usinage.

7.7 Modélisation et résolution du problème

Toutes les méthodes d'optimisation présentées dans ce mémoire ont été implémentées au sein du système MTE. La méthode de résolution à appliquer est sélectionnée par le système MTE en fonction du type de système d'usinage à boîtiers multibroches à concevoir. Lorsqu'il existe plusieurs méthodes de résolution pour le type du système d'usinage à concevoir, l'utilisateur peut choisir parmi les méthodes disponibles celle qu'il préfère utiliser et régler ses paramètres de départ.

La solution optimale du problème, trouvée par le système MTE, peut être consultée moyennant la boîte de dialogue « Résultats de calcul ». La solution optimale pour notre exemple est présentée dans la Figure 7.11. Le dessin de la machine de transfert conçue est présenté dans la Figure 7.12.

Résultats du calcul

Poste	Bloc	Opérations	Course active, mm	Vitesse d'avance mm/min	Temps, d'usinage min	Temps, de poste min
1	Bloc 1.	1 (Fraisage ébauche face culasse)	440	3600	0.122	0.312
2	Bloc 1.	2-15 (Lamage trous fixation culasse, Alésage ébauche départ usinage)	45	984	0.046	0.377
	Bloc 2.	16-21(Alésage finition départ usinage)	45	492	0.091	
3	Bloc 1.	22 (Fraisage finition face vilebrequin)	570	5500	0.1	0.29
4	Bloc 1.	23-32, 45-54 (Alésage fixation paliers)	50	1350	0.037	0.302
	Bloc 2.	57-66 (Taradauge fixation paliers)	88	1769	0.025	
5	Bloc 1.	33-44 (Perçage ébauche goupille orientation carter chapeau)	20	540	0.037	0.31
	Bloc 2.	55-56 (Alésage finition départ usinage face vilebrequin)	16	492	0.033	

Annuler | Ok | Aide | Aperçu | Paramètres auxiliaires | Insertion d'une position | Fiche de validation

FIG. 7.11 – Solution optimale

Il est à noter qu'il est possible d'afficher toutes les solutions optimales trouvées par l'approche par graphe, si cette méthode est utilisée pour la résolution du problème. Dans ce cas, chacune des solutions peut être consultée et l'utilisateur a la possibilité de choisir parmi elles celle qu'il préfère mettre en place.

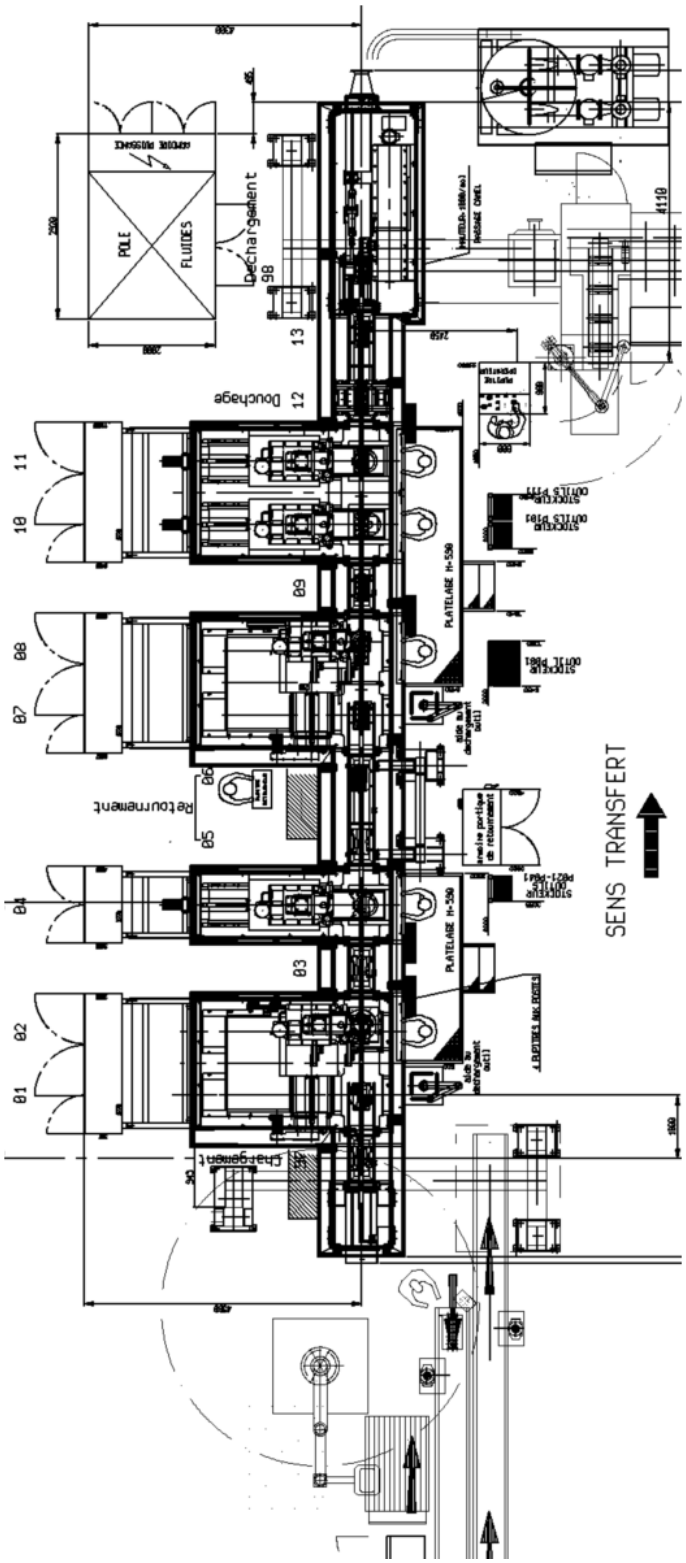


FIG. 7.12 – Dessin de la machine de transfert obtenue

7.8 Conclusion

Nous avons présenté un exemple de la conception d'une machine de transfert destinée à la fabrication d'un modèle de carter de moteur. Cette étude nous a permis de valider différents aspects de la méthodologie proposée dans ce manuscrit et de confronter des algorithmes développés à un problème réel.

À travers cet exemple, nous avons également présenté notre prototype de logiciel d'aide à la décision, destiné à supporter la conception en avant-projet des systèmes d'usinage. Ce logiciel comporte plusieurs modules permettant la modélisation de la pièce à fabriquer, la planification de son processus d'usinage, la configuration du système de fabrication, le choix de son agencement, et finalement le calcul de son coût (les deux derniers modules n'ont pas été présentés dans ce mémoire, pour plus de détails voir [Dolgui *et al.*, 2005*b*; Dolgui *et al.*, 2007*a*]). L'utilisation de ce logiciel permet de réduire le temps de la conception, de simplifier le travail de l'ingénieur et d'améliorer la qualité de solutions retenues. Au cœur de ce logiciel se trouvent les modèles mathématiques des problèmes d'optimisation de la configuration des systèmes d'usinage à boîtiers multibroches, développés dans ce mémoire. Ces problèmes d'optimisation sont liés au choix optimal du nombre de postes de travail et du nombre de boîtiers multibroches. Ce choix se base sur une répartition optimale des opérations entre les postes de travail en tenant compte des régimes d'usinage pour chaque boîtier.

Le logiciel, présenté dans ce chapitre, comporte plusieurs outils graphiques permettant d'interagir avec l'utilisateur. Il permet de saisir les caractéristiques de la pièce, les contraintes de précédence entre les opérations, les contraintes de compatibilité liées aux spécificités technologiques de postes de travail et de boîtiers, etc. Le choix du mode d'usinage, l'agencement et le calcul du prix du système de fabrication se font à partir de différentes bases de données qu'il est possible de compléter via un ensemble de boîtes de dialogue.

Conclusion générale

Dans ce mémoire, nous avons présenté une étude portant sur la conception en avant-projet des systèmes d'usinage à boîtiers multibroches. La particularité de tels systèmes par rapport aux autres systèmes d'usinage réside dans la possibilité d'exécuter simultanément plusieurs opérations à l'aide d'une seule unité d'usinage. Les éléments de base de ces systèmes ainsi que leurs modes de fonctionnement ont été présentés dans le Chapitre 1.

Dans le Chapitre 2, nous avons analysé les différentes activités décisionnelles qui font partie de la conception en avant-projet des systèmes d'usinage à boîtiers multibroches. Les résultats de cette analyse nous ont permis de structurer ce processus en étapes et de déterminer, pour chaque étape, la structure des données d'entrée et de sortie. Par ailleurs, nous avons pu constater l'absence de méthodes d'optimisation efficaces à appliquer à l'étape du choix de la configuration du système d'usinage. Compte tenu de l'importance de cette étape, due au fait que le coût et l'efficacité du système d'usinage conçu dépendent fortement de la qualité des résultats qui y sont obtenus, nous avons concentré nos travaux sur le développement de telles méthodes.

Après avoir analysé les décisions qui doivent être prises en amont et en aval de l'étape du choix de la configuration du système d'usinage, nous avons pu déterminer la structure des données pour le problème d'optimisation qui en relève et la structure des solutions nécessaires. En se basant sur ces résultats et en tenant compte des contraintes qui doivent être respectées, nous avons proposé une formulation mathématique générique pour le problème d'optimisation de la configuration des systèmes d'usinage à boîtiers multibroches. Ce problème consiste à regrouper en blocs les opérations nécessaires pour l'usinage d'une pièce (ou des pièces) donnée(s), sachant que les opérations affectées à un bloc sont à exécuter par un boîtier multibroche. Par conséquent, une telle affectation doit respecter les contraintes d'exclusion au niveau des blocs et les contraintes de précédence entre les opérations. À leur tour, les blocs créés doivent être affectés à des postes de travail. Lors de cette affectation, les contraintes d'inclusion, les contraintes d'exclusion au niveau des postes de travail, les contraintes de précédence entre les opérations, ainsi que la contrainte sur le temps de cycle doivent être prises en compte. L'objectif est de minimiser le coût du système d'usinage à boîtiers multibroches qui est estimé par le coût total de tous les postes de travail mis en place et tous les boîtiers qui y sont installés. Une des particularités de ce problème réside dans la dépendance entre le temps d'exécution des opérations et le contenu des blocs auxquels elles appartiennent.

Dans le Chapitre 3, nous avons comparé le problème d'optimisation que nous avons soulevé avec les problèmes déjà étudiés dans la littérature. Les résultats de cette comparaison

ont mis en évidence l'impossibilité de transformer le problème que nous traitons, sans simplifier sa structure, en un problème connu. Nous avons donc conclu que le développement des méthodes adaptées au contexte étudié était nécessaire.

Le modèle générique développé dans le Chapitre 2 a été ensuite utilisé comme un appui pour le développement des modèles mathématiques dédiés à des types de systèmes d'usinage différents, à savoir : machines de transfert, machines à table mobile et machines à table circulaire pivotante. Ainsi, trois problèmes d'optimisation différents ont été abordés dans ce mémoire. Chacun d'entre eux relève de la conception d'un système d'usinage à boîtiers multibroches différent et amène des contraintes particulières, nécessitant des modèles mathématiques différents. Pour cette raison, il était nécessaire de compléter le modèle générique par des contraintes propres à chaque type de système considéré.

Ainsi, dans le Chapitre 4, nous avons formulé le problème d'optimisation de la configuration des machines de transfert. Ces machines appartiennent à la classe des systèmes d'usinage à boîtiers multibroches ayant un mode d'activation séquentiel des boîtiers. Puisque le problème étudié est NP-difficile, le temps de calcul nécessaire pour sa résolution exacte croît de manière exponentielle en fonction de la taille du problème. Par conséquent, il est impossible d'obtenir des solutions optimales pour les problèmes de grande taille, comme cela a été montré dans [Finel, 2004]. Nous avons utilisé deux stratégies différentes pour pallier à cet inconvénient et obtenir des solutions efficaces, même pour les problèmes difficiles, avec des temps de calcul acceptables. La première stratégie, exposée dans le Chapitre 4, consiste à développer des procédures de pré-traitement à appliquer avant la phase d'optimisation. La deuxième stratégie, présentée dans le Chapitre 5, vise à mettre en place des méthodes approchées pour la résolution efficace des problèmes de grande taille.

Dans le but d'améliorer les performances des méthodes exactes et de rendre la résolution exacte accessible pour une gamme de problèmes plus large, nous avons élaboré, dans le Chapitre 4, des procédures de pré-traitement. Pour tester leur efficacité, nous avons utilisé un échantillon de problèmes de complexité différente. Les résultats obtenus montrent des améliorations significatives lors de la résolution de problèmes de grande taille, pour certains problèmes, par exemple, le temps de résolution a été divisé par 10 après l'application de procédures de pré-traitement. De plus, nous avons évalué l'impact des différentes contraintes du problème et du nombre d'opérations sur l'efficacité des procédures de pré-traitement mises en place.

Dans le Chapitre 5, nous avons développé des méthodes approchées. D'abord, nous avons proposé différentes améliorations pour une heuristique basée sur l'approche Monte Carlo, développée dans [Finel, 2004]. Ensuite, nous avons complété la meilleure variante de cette heuristique par une phase d'amélioration des solutions, réalisée à l'aide d'une décomposition dynamique et d'une résolution des sous-problèmes par une méthode exacte. Puis, nous avons transformé cette approche en une métaheuristique de type GRASP en ajoutant une règle de priorité permettant de sélectionner les opérations d'une manière plus rigoureuse lors de la construction des solutions initiales. Moyennant des expérimentations numériques, nous avons évalué les performances de chaque approche développée. De plus, nous avons utilisé un échantillon de problèmes dont la structure des données est similaire à celle des problèmes réels existant dans l'industrie. Les résultats obtenus nous ont permis de constater que les approches développées fournissent des solutions de bonne qualité pour des problèmes réalistes. Nous

avons également souligné les améliorations apportées par chacune des démarches entreprises. Nos perspectives à court terme consistent à étudier plus minutieusement les performances des approches développées, notamment à étudier l'impact de la structure du problème à résoudre.

Dans le Chapitre 6, nous avons étudié le problème d'optimisation de la configuration des machines à table mobile et à table circulaire pivotante.

D'abord, nous nous sommes intéressés au mode de fonctionnement des machines à table mobile. La particularité des machines de ce type réside dans leur mode de transport de la pièce, car une seule unité de pièce peut être traitée sur la machine pendant un cycle d'usinage. Une autre différence de ces machines vis-à-vis des machines de transfert, considérées dans les Chapitres 4 et 5, est due au mode d'activation de boîtiers qui est parallèle pour les machines à table mobile et séquentiel pour les machines de transfert.

Nous avons proposé un modèle mathématique en variables mixtes pour le problème d'optimisation de la configuration des machines à table mobile. Par conséquent, il est possible de résoudre ce problème à l'aide d'un logiciel d'optimisation, comme, par exemple, ILOG Cplex ou XPress MP. Cependant, l'étude des propriétés des problèmes industriels, nous a permis de constater que dans la pratique, les problèmes de ce type n'atteignent pas une taille très importante et sont en plus riches en contraintes d'exclusion aussi bien au niveau des blocs qu'au niveau des postes de travail. Ceci nous a orienté vers le développement d'une approche par graphe, car nous avons déjà constaté qu'elle fournit des meilleurs résultats, par rapport à l'utilisation d'ILOG Cplex, lors de la résolution des problèmes ayant une telle structure. L'approche par graphe est une approche exacte, basée sur la recherche des chemins les plus courts sous contraintes dans un graphe spécialement conçu (graphe d'états de la pièce lors de l'usinage). Le graphe d'états de la pièce est construit en explorant toutes les affectations d'opérations possibles et en tenant compte des contraintes du problème. De façon générale, la taille de ce graphe croît de manière exponentielle avec le nombre d'opérations nécessaires pour l'usinage de la pièce. Cependant, de nombreuses contraintes d'exclusion et de précédence qui caractérisent les problèmes réels, ainsi que les règles de dominance que nous avons développées, permettent de réduire le nombre d'états possibles, et par conséquent, de diminuer la taille du graphe et son temps de construction. Puisque l'algorithme, que nous avons proposé, construit le graphe d'états et résout le problème en même temps, il est possible d'obtenir toutes les solutions optimales avec un temps de calcul court. Cela a été notamment montré par l'étude d'un exemple industriel.

En se basant sur le modèle développé pour le cas des machines à table mobile, nous avons également proposé une formulation en variables mixtes du problème d'optimisation de la configuration des machines à table circulaire pivotante. Comme pour les machines à table mobile, ce problème peut donc être résolu avec un logiciel d'optimisation.

Les perspectives de l'étude des problèmes d'optimisation présentés dans le Chapitre 6 relèvent de la nécessité d'une analyse plus approfondie des performances des méthodes exactes proposées, notamment leur dépendance de la structure du problème à résoudre. Cette analyse permettrait également d'évaluer la nécessité du développement des méthodes approchées pour ces problèmes d'optimisation.

Enfin, dans le Chapitre 7, nous avons présenté une étude de cas qui porte sur la résolution

d'un problème réel de conception d'une machine de transfert destinée à la fabrication d'un modèle de carter de moteur. Cette étude nous a permis de valider différents aspects de la méthodologie proposée dans ce mémoire et de confronter les algorithmes développés à un problème de taille industrielle.

À travers cette étude de cas, nous avons également pu exposer un prototype de logiciel d'aide à la décision, baptisé MTE et destiné à supporter la conception en avant-projet des systèmes d'usinage à boîtiers multibroches. Ce logiciel comporte plusieurs modules permettant la modélisation de la pièce à fabriquer, la planification de son processus d'usinage, la configuration du système d'usinage, le choix de son agencement, et finalement le calcul de son coût (les deux derniers modules n'ont pas été présentés dans ce mémoire, pour plus de détails voir [Dolgui *et al.*, 2005*b*; Dolgui *et al.*, 2007*a*]).

L'utilisation de ce logiciel pour la conception en avant-projet des systèmes d'usinage augmente l'efficacité des solutions proposées. L'intégration des méthodes d'optimisation dans le logiciel permet de trouver des solutions optimales ou proches de l'optimum. Ceci conduit à la réduction du temps de conception, et par conséquent, permet à l'ingénieur de répondre plus rapidement et d'une manière plus précise aux appels d'offre. Hormis l'exemple présenté dans ce mémoire, nous avons testé notre logiciel sur d'autres cas industriels, et il a montré son efficacité.

Les études réalisées dans le cadre de cette thèse ouvrent la voie à d'autres études qui peuvent être menées dans les directions suivantes :

- la modélisation d'autres problèmes d'optimisation liés à la conception et l'utilisation des systèmes d'usinage ;
- l'application d'autres techniques de recherche opérationnelle pour la résolution de ces nouveaux problèmes ainsi que des problèmes déjà étudiés. Par exemple, en analysant l'état de l'art dans le Chapitre 3, nous pouvons remarquer que les approches utilisant la génération de colonnes et les algorithmes génétiques sont très souvent sollicités pour la résolution des problèmes d'équilibrage de chaînes d'assemblage. Ainsi leur application dans le cadre de la conception de systèmes d'usinage à boîtiers multibroches pourrait être une voie intéressante.

Bibliographie

Agnetis, A., A. Ciancimino, M. Lucertini et M. Pizzichella (1995). Balancing flexible lines for car components assembly. *International Journal of Production Research* **33**, 333–350.

Aĝpak, K. et H. Gökçen (2005). Assembly line balancing: two resource constrained cases. *International Journal of Production Economics* **96**, 129–140.

Agrawal, P.K. (1985). The related activity concept in assembly line balancing. *International Journal of Production Research* **23**, 403–421.

Amen, M. (2000*a*). An exact method for cost-oriented assembly line balancing. *International Journal of Production Economics* **64**, 187–195.

Amen, M. (2000*b*). Heuristic methods for cost-oriented assembly line balancing: A survey. *International Journal of Production Economics* **68**, 1–14.

Amen, M. (2001). Heuristic methods for cost-oriented assembly line balancing: A comparison on solution quality and computing time. *International Journal of Production Economics* **69**, 255–264.

Amen, M. (2006). Cost-oriented assembly line balancing: Model formulations, solution difficulty, upper and lower bounds. *European Journal of Operational Research* **168**, 747–770.

Amiolemhen, P.E. et A.O.A. Ibhadode (2004). Application of genetic algorithms - determination of the optimal machining parameters in the conversion of a cylindrical bar stock into a continuous finished profile. *International Journal of Machine Tools and Manufacture* **44**, 1403–1412.

Anderson, E.J. et M.C. Ferris (1994). Genetic algorithms for combinatorial optimization: the assembly line balancing problem. *ORSA Journal on Computing* **6**, 161–173.

Andres, C., C. Miralles et R. Pastor (2006). Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. *European Journal of Operational Research*. À paraître, doi :10.1016/j.ejor.2006.07.044.

Anwer, N. (2000). *Méthodologie d'analyse de raisonnement pour la génération automatique des gammes d'usinage en fraisage. Contribution à la caractérisation des entités par analyse des contraintes d'usinabilité*. Thèse de doctorat. Ecole Normale Supérieure de Cachan, Paris (France).

Arcus, A.L. (1966). COMSOAL: A computer method of sequencing operations for assembly lines. *International Journal of Production Research* **4**, 259–277.

Askin, R.G. et M. Zhou (1997). A parallel station heuristic for the mixed-model production line balancing problem. *International Journal of Production Research* **35**, 3095–3105.

Bard, J.F. (1989). Assembly line balancing with parallel workstations and dead time. *International Journal of Production Research* **27**, 1005–1018.

Bartholdi, J.J. (1993). Balancing two-sided assembly lines: A case study. *International Journal of Production Research* **31**, 2447–2461.

Bautista, J. et J. Pereira (2002). Ant algorithms for assembly line balancing. *Lecture Notes in Computer Science* **2463**, 65–75.

Bautista, J. et J. Pereira (2007). Ant algorithms for a time and space constrained assembly line balancing. *European Journal of Operational Research* **177**(3), 2016–2032.

Bautista, J., R. Suarez, M. Mateo et R. Companys (2000). Local search heuristics for the assembly line balancing problem with incompatibilities between tasks. *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*. San Francisco, CA. pp. 2404–2409.

Baybars, I. (1986). A survey of exact algorithms for the simple assembly line balancing. *Management Science* **32**, 909–932.

Baykasoglu, A. et L. Özbakir (2006). Stochastic U-line balancing using genetic algorithms. *International Journal of Advanced Manufacturing Technology*. *doi* : $10.1007/s00170-005-0322-4$.

Becker, C. et A. Scholl (2006). A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research* **168**, 694–715.

Belmokhtar, S. (2006). *Lignes d'usinage avec équipements standard*: *modélisation, configuration et optimisation*. Thèse de doctorat. École Nationale Supérieure des Mines de Saint-Etienne.

Belmokhtar, S., A. Bratcu et A. Dolgui (2006*a*). Modular machining line design and reconfiguration: some optimization methods. *Manufacturing the Future*: *Concepts-Technologies-Visions* (V. Kordic, A. Lazinica et M. Merdan, Eds.). pp. 125–152. pro literatur Verlag.

Belmokhtar, S., A. Dolgui, N. Guschinsky et G. Levin (2006*b*). Integer programming models for logical layout design of modular machining lines. *Computers and Industrial Engineering* **51**, 502–518.

Belmokhtar, S., A. Dolgui, I. Ignatenko et X. Delorme (2007). Optimizing modular machining line design problem with mixed activation mode of machining units. *Decision Making in Manufacturing and Services*. À paraître.

Bernard, A. (2003). *IC2 Productique - Information - Commande - Communication*. FAO - Hermès Science Publications.

Bezdek, E.J., D.C. Thompson, K.L. Wood et R.H. Crawford (1999). Volumetric feature recognition for direct engineering. *Direct Engineering*: *Toward Intelligent Manufacturing* (A. Kamrani et P. Sferro, Eds.). pp. 15–69. Editions Kluwer Academic.

Bhattacharjee, T.K. et S. Sahu (1990). Complexity of single model assembly line balancing problems. *Engineering Costs and Production Economics* **18**, 203–214.

Boctor, F. (1995). A multiple-rule heuristic for assembly line balancing. *Journal of the Operational Research Society* **46**, 62–69.

Boucher, T.O. (1987). Choice of assembly line design under task learning. *International Journal of Production Research* **25**, 513–524.

Boysen, N. et M. Fliedner (2006). A versatile algorithm for assembly line balancing. *European Journal of Operational Research*. to appear.

Boysen, N., M. Fliedner et A. Scholl (2006). A classification of assembly line balancing problems. *European Journal of Operational Research*. À paraître, doi :10.1016/j.ejor.2006.10.010.

Brans, J.P. et B. Marechal (1994). The PROMCALC & GAIA decision support system for multicriteria decision aid. *Decision Support Systems* **12**, 297–310.

Brissaud, D. (1992). *Système de conception automatique de gammes d'usinage pour les industries manufacturières*. Thèse de doctorat. Université Joseph Fournier, Grenoble.

Brown, J. (2004). *Digital Manufacturing. The PLM Approach to better Manufacturing Processes*. Tech-Clarity.

Bukchin, J. (1998). A comparative study of performance measures for throughput of mixed model assembly line balancing in JIT environment. *International Journal of Production Research* **36**(10), 2669–2685.

Bukchin, J., E.M. Dar-El et J. Rubinovitz (2002). Mixed-model assembly line design in a make-to-order environment. *Computers and Industrial Engineering* **41**, 405–421.

Bukchin, J. et J. Rubinovitz (2002). A weighted approach for the assembly line design with station paralleling and equipment selection. *IIE Transactions* **35**(1), 513–524.

Bukchin, J. et M. Tzur (2000). Design of flexible assembly line to minimize equipment cost. *IIE Transactions* **32**, 585–598.

Bukchin, Y. et I. Rabinowitch (2006). A branch-and-bound based solution approach for the mixed-model assembly line-balancing problem for minimizing stations and task duplication costs. *European Journal of Operational Research* **174**, 492–508.

Buxey, G.M. (1974). Assembly line balancing with multiple stations. *Management Science* **20**, 1010–1021.

Buxey, G.M., N.D. Slack et R. Wild (1973). Production flow line system design-a review. *AIIE Transactions* **5**, 37–48.

Calos, H. et A. Whitlock (1986). *Monte Carlo Methods, vol* 1 : *Basics*. John Wiley, New York.

Capacho, L. et R. Pastor (2005). ASALBP: the Alternative Subgraphs Assembly Line Balancing Problem. Technical Report IOC-DT-P-2005-5. UPC, Barcelona, Spain.

Capacho, L. et R. Pastor (2006). The ASALB Problem with Processing Alternatives Involving Different Tasks: Definition, Formalization and Resolution. *The 2006 International Conference on Computational Science and its Applications, ICCSA 2006, Lecture Notes in Computer Science* (M. Gavrilova et al., Ed.). Vol. 3982. pp. 554–563. Springer-Verlag.

Capacho, L., **O. Guschinskaya**, A. Dolgui et R. Pastor (2006*a*). An evaluation study of approximate methods for a line balancing problem with assembly alternatives. *Proceedings of the 8th International Conference on the Modern Information Technology in the Industrial Enterprises*. Budapest, Hungary, pp. 199–204.

Capacho, L., **O. Guschinskaya**, A. Dolgui et R. Pastor (2006*b*). Heuristic methods to solve the alternative subgraphs assembly line balancing problem. *Proceedings of the 2006 IEEE Conference on Automation Science and Engineering*. Shangai Pudong, China, pp. 513–518.

Capacho, L., R. Pastor, A. Dolgui et **O. Guschinskaya** (2007). An evaluation of constructive heuristic methods to solve the alternative subgraphs assembly line balancing problem. *Journal of Heuristics*. À paraître.

Carraway, R.L. (1989). A dynamic programming approach to stochastic assembly line balancing. *Management Science* **35**, 459–471.

Cetim (2000). Tournage dur et usinage à grande vitesse (UGV). Technical report. Centre technique des industries mécaniques.

Chakravarty, A.K. (1988). Line balancing with task learning effects. *IIE Transactions* **20**, 186–193.

Chan, K.C.C., P.C.L. Hui, K.W. Yeung et F.S.F. Ng (1998). Handling the assembly line balancing problem in the clothing industry using a genetic algorithm. *International Journal of Clothing Science and Technology* **10**(1), 21–37.

Chen, R.-S., K.-Y. Lu et S.-C. Yu (2002). A hybrid genetic algorithm approach on multi-objective of assembly planning problem. *Engineering Applications of Artificial Intelligence* **15**, 447–457.

Chiang, W.C. (1998). The application of a tabu search metaheuristic to the assembly line balancing problem. *Annals of Operations Research* **77**, 209–227.

Chow, W.M., Ed. (1990). *Assembly line design*: *methodology and applications*. Marcel Deker Inc., New York, Basel.

Cohen, Y. et M.E. Dar-El (1998). Optimizing the number of stations in assembly lines under learning for limited production. *Production Planning and Control* **9**(3), 230–240.

Coromant Sandvik® (2006). Main catalogue. http ://www.coromant.sandvik.com/uk.

Coromant Sandvik® (2007). Main catalogue. http ://www2.coromant.sandvik.com/coromant /catalogue2007/main_d.pdf.

Corominas, A., R. Pastor et J. Plans (2007). Balancing assembly line with skilled and unskilled workers. *OMEGA The International Journal of Management Science*. À paraître, doi :10.1016/j.omega.2006.03.003.

Cus, F. (2000). The inclusion of the geometrical shape of the cutter into the optimisation of the milling process. *International Journal of Advanced Manufacturing Technology* **16**, 392–403.

Cutkosky, M., J.M. Tenenbaum et D. Muller (1988). Features in process based design. *Proceedings of the 1988 ASME International Computers in Engineering Conference and Exhibition*. pp. 557–562.

Dagnino, A. (1994). Integrated architecture for assembly planning in an electronics manufacturing environment. *Integrated Manufacturing Systems* **5**(4/5), 77–86.

Dashchenko, A.I., Ed. (2003). *Manufacturing Technologies for Machines of the Future 21st Century*. Springer.

Dashchenko, A.I., Ed. (2005). *Technologies for motor-car construction*. Academicheskiy proekt: Triksta. (En russe).

Dashchenko, A.I., Ed. (2006). *Reconfigurable Manufacturing Systems and Transformable Factories*. Springer.

Decker, K.M. (1991). The Monte Carlo method: Theory and application. *Computer Methods in Applied Mechanics and Engineering* **89**, 463–483.

Deckro, R.F. (1989). Balancing cycle time and workstations. *IIE Transactions* **21**, 106–111.

Delorme, X. (2003). *Modélisation et résolution de problèmes liés à l'exploitation d'infrastructures ferroviaires*. Thèse de doctorat. Université de Valenciennes et du Hainaut-Cambrésis.

Deneux, D., Ed. (2002). *Méthodes et modèles pour la conception concourante*. Habilitation à Diriger des Recherches. Université de Valenciennes et du Hainaut Cambrésis.

Dereli, T., I.H. Filiz et A. Baykasoglu (2001). Optimizing cutting parameters in process planning of prismatic parts by using genetic algorithms. *International Journal of Production Research* **39**(15), 3303–3328.

Derigent, W. (2005). *Méthodologie de passage d'un modèle CAO vers un modèle FAO pour des pièces aéronautiques : Prototype logiciel dans le cadre du projet USIQUICK*. Thèse de doctorat. Université Henri Poincaré, Nancy-I.

Derras, C. (1998). *Formalisation de l'imprécision informationnelle et des incertitudes décisionnelles des connaissances expertes pour la génération de processus de fabrication*. Thèse de doctorat. Université Henri Poincaré, Nancy-I.

Dimitriadis, S.G. (2006). Assembly line balancing and group working: A heuristic procedure for workers' groups operating on the same product and workstation. *Computers and Operations Research* **33**, 2757–2774.

Dolgui, A., B. Finel, **O. Guschinskaya**, N. Guschinsky, G. Levin et F. Vernadat (2006*a*). Balancing large-scale machining lines with multi-spindle heads using decomposition. *International Journal of Production Research* **44**(18-19), 4105–4120.

Dolgui, A., B. Finel, N. Guschinsky, G. Levin et F. Vernadat (2005*a*). A heuristic approach for transfer lines balancing. *Journal of Intelligent Manufacturing* **16**(2), 159–171.

Dolgui, A., B. Finel, N. Guschinsky, G. Levin et F. Vernadat (2006*b*). MIP approach to balancing transfer lines with blocks of parallel operations. *IIE Transactions* **38**, 869–882.

Dolgui, A. et J.M. Proth (2006). *Les systèmes de production modernes*. Lavoisier.

Dolgui, A., **O. Guschinskaya**, N. Guschinsky et G. Levin (2005*b*). Conception de systèmes de fabrication: prototype d'un logiciel d'aide à la decision. *Journal of Decision Systems* **14**(4), 489–516.

Dolgui, A., **O. Guschinskaya**, N. Guschinsky et G. Levin (2006*c*). "Machine Tools Engineering": Integrated decision support system for the logical layout design of unit head machines. *Proceedings of the International Conference on Global Manufacturing and Innovation*. Coimbatore, India, CD-ROM,9 pages.

Dolgui, A., **O. Guschinskaya**, N. Guschinsky et G. Levin (2006*d*). Optimization in design of unit head machines with a mobil table. *Information Control Problems In Manufacturing* 2006 : *A Proceedings volume from the 12th IFAC International Symposium* (A. Dolgui, G. Morel et C. E. Pereira, Eds.). Vol. 2. Elsevier Science. pp. 413–418.

Dolgui, A., **O. Guschinskaya**, N. Guschinsky et G. Levin (2007*a*). Decision making and support tools for design of machining systems. *Encyclopedia of Decision Making and Decision Support Technologies* (F. Adam et P. Humphreys, Eds.). À paraître.

Dolgui, A., N. Guschinsky et G. Levin (1999). On Problem of Optimal Design of Tranfer Lines with Parallel and Sequential Operation. *Proceedings of the 7th IEEE International Conference on Emerging Technologies and Factor Automation* (J.M. Fuertes, Ed.). Vol. 1. Barcelona, Spain. pp. 329–334.

Dolgui, A., N. Guschinsky et G. Levin (2006*e*). A special case of transfer lines balancing by graph approach. *European Journal of Operational Research* **168**(3), 732–746.

Dolgui, A., N. Guschinsky et G. Levin (2007*b*). Exact and heuristic algorithms for balancing transfer lines when a set of available spindle heads is given. *International Transactions in Operational Research*. À paraître.

Dolgui, A. et I. Ihnatsenka (2007). Branch and bound algorithm for a transfer line design problem: stations with sequentially activated multi-spindle heads. *European Journal of Operational Research*. À paraître.

Dolgui, A., N. Guschinsky, G. Levin et J.M. Proth (2008). Optimisation of multi-position machines and transfer lines. *European Journal of Operational Research* **185**(3), 1375–1389.

Driscoll, J. et D. Thilakawardana (2001). The definition of assembly line balancing difficulty and evaluation of balance solution quality. *Robotics and Computer Integrated Manufacturing* **17**, 81–86.

Easton, F., B. Faaland, T.D. Klastorin et T. Schmitt (1989). Improved network based algorithms for the assembly line balancing problem. *International Journal of Production Research* **27**, 1901–1915.

Erel, E. et H. Gökcen (1999). Shortest route formulation of mixed-model assembly line balancing problem. *European Journal of Operational Research* **116**, 194–204.

Erel, E. et S.C. Sarin (1998). A survey of the assembly line balancing procedures. *Production Planning and Control* **9**(5), 414–434.

Feo, T.A. et M.G.C. Resende (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters* **8**, 67–71.

Festa, P. et M.G.C. Resende (2001). GRASP: An annotated bibliography. *Essays and Surveys on Metaheuristics* (C.C. Ribeiro et P. Hansen, Eds.). pp. 325–367. Kluwer Academic Publishers.

Finel, B. (2004). *Structuration de lignes d'usinage* : *méthodes exactes et heuristiques*. Thèse de doctorat. Université de Metz.

Finel, B., A. Dolgui et F. Vernadat (2006). A random search and backtracking procedure for transfer line balancing. *International Journal of Computer Integrated Manufacturing*. À paraître.

Fleszar, K. et K.S. Hindi (2003). An enumerative heuristic and reduction methods for the assembly line balancing problem. *European Journal of Operational Research* **145**, 606–620.

Gadidov, R. et W. Wilhelm (2000). A cutting plane approach for the single-product assembly system design problem. *International Journal of Production Research* **38**(8), 1731–1754.

Garro, B. (1992). *Conception d'éléments physiques de systèmes de production*. Thèse de doctorat. Université de Nancy.

Geelink, R. (1996). *Flexible definition of forms features*. Thèse de doctorat. Université de Twente (Pays-Bas).

Gehrlein, W.V. et J.H. Patterson (1978). Balancing single model assembly lines: comments on a paper by E.M. Dar-El (Mansoor). *AIIE Transactions* **10**, 109–112.

Ghosh, S. et R. Gagnon (1989). A comprehensive literature review and analysis of the design, balancing and scheduling of assembly lines. *International Journal of Production Research* **27**(4), 637–670.

Gökçen, H. et K. Aĝpak (2006). A goal programming approach to simple U-line balancing problem. *European Journal of Operational Research* **171**, 577–585.

Graves, S.C. et B.W. Lamar (1983). An integer programming procedure for assembly design problems. *Operations Research* **31**(3), 522–545.

Graves, S.C. et C.S. Redfield (1988). Equipment selection and task assignment for multiproduct assembly system design. *The International Journal of Flexible Manufacturing Systems* **1**, 31–50.

Grieves, M. (2005). *Product Lifecycle Management*: *Driving the Next Generation of Lean Thinking*. McGraw Hill.

Groupe GAMA (1990). *La gamme automatique en usinage*. Editions Hermès, Paris.

Guschinskaya, O., A. Dolgui, N. Guschinsky et G. Levin (2005). A combined heuristic approach for optimization of a class of machining lines. *Proceedings of the 2005 IEEE International Conference on Automation Science and Engineering*. IEEE, Edmonton, Canada. pp. 154–159.

Guschinskaya, O., A. Dolgui, N. Guschinsky et G. Levin (2008*a*). A heuristic multi-start decomposition approach. *European Journal of Operational Research* **189**, 902–913.

Guschinskaya, O., A. Dolgui, N. Guschinsky et G. Levin (2007*b*). New reduction methods for the transfer line balancing problem. *Preprints of IMS' 2007*: *IFAC Workshop On Intelligent Manufacturing Systems* (F.A. Andelas, C. Pereira et F. Torres, Eds.). Alicante, Spain. pp. 75–80.

Guschinskaya, O., A. Dolgui, N. Guschinsky et G. Levin (2007*c*). A scheduling problem for multi-spindle head machines with a mobile table. *Computers and Operations Research*. À paraître, doi:10.1016/j.cor.2007.10.010.

Guschinskaya, O. et A. Dolgui (2006). A comparative evaluation of exact and heuristic methods for transfer lines balancing problem. *Information Control Problems In Manufacturing* 2006 : *A Proceedings volume from the 12th IFAC International Symposium* (A. Dolgui, G. Morel et C. E. Pereira, Eds.). Vol. 2. Elsevier Science. pp. 395–400.

Guschinskaya, O. et A. Dolgui (2007*a*). Balancing transfer lines with multi-spindles machines using GRASP. *Preprints of the 4th IFAC conference on Management and Control in Manufacturing and Logistics* (O. Bologa, I. Dumitrache et F.G. Filip, Eds.). Vol. 2. Sibiu, Romania. pp. 225–226.

Guschinskaya, O. et A. Dolgui (2007*b*). A comprehensive comparative analysis of exact and heuristic methods for transfer line balancing problems. *International Journal of Production Economics*. À paraître.

Guschinskaya, O. et A. Dolgui (2007*c*). Heuristic methods for a transfer line balancing problem. *Proceedings of the 19th International Conference on Production Research* (J.A. Ceroni, Ed.). Valparaiso, Chile, CD-ROM, 6 pages.

Guschinskaya, O. et A. Dolgui (2007*d*). Une méthode exacte pour l'équilibrage de lignes d'usinage avec machines parallèles et changements d'outils. *Livre des résumés, FRANCORO V / ROADEF 2007*. Grenoble, France. pp. 225–226.

Gutjahr, A.L. et G.L. Nemhauser (1964). An algorithm for the line balancing problem. *Management Science* **11**, 308–315.

Hackman, S.T., M.J. Magazine et T.S. Wee (1989). Fast, effective algorithms for simple assembly line balancing problems. *Operations Research* **37**, 916–924.

Hammersley, J.M. et D.C. Handscomb (1964). *Monte Carlo Method*. Methuem, London.

Hamza, I., Z. Bouaziz et M. Haddar (2005). Optimisation du choix des outils de coupe pour l'usinage des poches quadrilatères en 2D1/2. *Proceedings of the 4th International Conference on Integrated Design and Production*. Casablanca, Morocco. p. 17.

Hart, J.P. et A.W. Shogan (1987). Semi-greedy heuristics: An empirical study. *Operations Research Letters* **6**, 107–114.

He, D.W. et A. Kusiak (1998). Designing an assembly line for modular products. *Computers and Industrial Engineering* **34**(1), 37–52.

Held, M., R.M. Karp et R. Shareshian (1963). Assembly line balancing dynamic programming with precedence constraints. *Operations Research* **11**, 442–459.

Helgeson, W.B. et D.P. Birnie (1961). Assembly line balancing using ranked positional weight technique. *Journal of Industrial Engineering* **12**, 394–398.

Hoffman, T.R. (1963). Assembly line balancing with a precedence matrix. *Management Science* **9**, 551–562.

Hoffman, T.R. (1992). EUREKA: a hybrid system for assembly line balancing. *Management Science* **38**, 39–47.

Hoffman, T.R. (1993). Response to note on microcomputer performance of "fable" on hoffmann's data sets. *Management Science* **39**, 1192–1193.

Hop, N. Van (2006). A heuristic solution for fuzzy mixed-model line balancing problem. *European Journal of Operational Research* **168**, 789–810.

Ignall, E.J. (1965). A review of assembly line balancing. *The Journal of Industrial Engineering* **16**, 244–254.

Inman, R.R. et M. Leon (1994). Scheduling duplicate serial stations in transfer lines. *International Journal of Production Research* **32**(11), 2631–2644.

Jackson, J.R. (1956). A computing procedure for a line balancing problem. *Management Science* **2**, 261–271.

Jacobe, P. (1992). *Génération ascendante de gamme d'usinage : proposition d'une méthode appliquée aux travaux de très grande série utilisant des machines à transfert circulaire.* Thèse de doctorat. Université de Franche-Comté.

Johnson, J.R. (1988). Optimally balancing large assembly lines with FABLE. *Management Science* **34**, 240–253.

Johnson, R.V. (1983). A branch and bound algorithm for assembly line balancing problems with formulation irregularities. *Management Science* **29**, 1309–1324.

Johnson, R.V. (1991). Balancing assembly lines for teams and work groups. *International Journal of Production Research* **29**(6), 1205–1214.

Kao, E.P.C. (1976). A preference order dynamic program for stochastic assembly line balancing. *Management Science* **22**, 1097–1104.

Kao, E.P.C. (1979). Computational experience with a stochastic assembly line balancing algorithm. *Computers and Operations Research* **6**, 79–86.

Kim, H. et S. Park (1995). A strong cutting plane algorithm for the robotic assembly line balancing problem. *International Journal of Production Research* **33**(8), 2311–2323.

Kim, Y.J., Y.K. Kim et Y. Cho (1998). A heuristic-based genetic algorithm for workload smoothing in assembly lines. *Computers and Operations Research* **25**(2), 99–111.

Kim, Y.K., Y. Kim et Y.J. Kim (2000). Two-sided assembly line balancing: A genetic algorithm approach. *Production Planning and Control* **11**(3), 44–53.

Kim, Y.K., Y.J. Kim et Y. Kim (1996). Genetic algorithm for assembly line balancing problem with various objectives. *Computers and Industrial Eginnering* **30**(3), 397–409.

Kimms, A. (2000). Minimal investment budgets for flow line configuration. *IIE Transactions* **32**, 287–298.

Koren, Y., U. Heisel, F. Jovane, T. Moriwaki, G. Pritchow, H. Van Brussel et A.G. Ulsoy (1999). Reconfigurable manufacturing systems. *CIRP Annals* **48**(2), 527–598.

Lapierre, S.D., A.B. Ruiz et P. Soriano (2006). Balancing assembly lines with tabu search. *European Journal of Operational Research* **168**(3), 826–837.

Lapierre, S.D. et A.B. Ruiz (2004). Balancing assembly lines: An industrial case study. *Journal of the Operational Research Society* **55**, 589–597.

Lee, H.F. et R.V. Johnson (1991). A line-balancing strategy for designing flexible assembly systems. *The International Journal of Flexible Manufacturing Systems* **3**, 91–120.

Lee, T.O., Y. Kim et Y.K. Kim (2001). Two-sided assembly line balancing to maximize work relatedness and slackness. *Computers and Industrial Engineering* **40**, 273–292.

Lefebvre, A., J. Renaud, L. Sabourin et G. Gogu (2001). Modélisation des contraintes de réalisation pour la conception de gammes d'usinage en grande série. *Revue Internationale de CFAO et d'informatique graphique* **16**(4), 433–443.

Leu, Y.-Y., L.A. Matheson et L.P. Rees (1994). Assembly line balancing using genetic algorithms with heuristic- generated initial populations and multiple evaluation criteria. *Decision Science* **25**, 581–606.

Levitin, G., J. Rubinovitz et B. Shnits (2006). A genetic algorithm for robotic assembly line balancing. *European Journal of Operational Research* **168**, 811–825.

Licon^{MT}® (2007). Highly flexible 3-axis machining cell in stand alone configuration. http ://www.licon.com/index.php.

Little, G., D. Clark, J. Corney et J. Tuttle (1998). Delta volume decomposition for multi-sided components. *Computer-Aided Design* **30**(9), 695–706.

Lombard, M. (2006). *Contribution de la modélisation informationnelle aux processus de conception et réalisation de produits manufacturiers : vers une ontologie métier*. Habilitation à Diriger des Recherches, Université Henri Poincaré Nancy I.

Lossent, L. (1997). *Contribution à la conduite d'étude de faisabilité de systèmes de fabrication*. Thèse de doctorat. Université de Nancy 1.

Macaskill, J.L.C. (1972). Production-line balances for mixed-model lines. *Management Science* **19**, 423–434.

Martin, P., A. D'Acunto, J. Arzur et L. Abt (2001). Intégration produit-process. Méthodologie de conception d'un système de fabrication dédié à la grande série. *Revue internationale de CFAO et d'informatique graphique* **16**(4), 459–476.

Masood, S. (2006). Line balancing and simulation of an automated production transfer line. *Assembly Automation* **26**(1), 69–74.

Mastor, A.A. (1970). An experimental investigation and comparative evaluation of production line balancing techniques. *Management Science* **16**, 728–746.

Matanachai, S. et C.A. Yano (2001). Balancing mixed-model assembly lines to reduce work overload. *IIE Transactions* **33**, 29–42.

McMullen, P.R. et G.V. Frazier (1997). A heuristic for solving mixed-model line balancing problems with stochastic task durations and parallel stations. *International Journal of Production Economics* **51**, 177–190.

McMullen, P.R. et G.V. Frazier (1998). Using simulated annealing to solve a multiobjective assembly line balancing problem with parallel workstations. *International Journal of Production Research* **36**, 2717–2741.

McMullen, P.R. et P. Tarasewich (2003). Using ant techniques to solve the assembly line balancing problem. *IIE Transactions* **35**, 605–617.

Mehrabi, M.G., A.G. Ulsoy et Y. Koren (2000). Reconfigurable manufacturing systems: Key to futur manufacturing. *Journal of Intelligent Manufacturing* **11**, 403–419.

Mehrabi, M.G., A.G. Ulsoy, Y. Koren et P. Heytler (2002). Trends and perspectives in flexible and reconfigurable manufacturing systems. *Journal of Intelligent Manufacturing* **13**, 135–146.

Merengo, C., F. Nava et A. Pozetti (1999). Balancing and sequencing manual mixed-model assembly lines. *International Journal of Production Research* **37**, 2835–2860.

Miao, M.G., N. Sridharan et J. J. Shah (2002). CAD-CAM integration using machining features. *International Journal of Computer Integrated Manufacturing* **15**(4), 296–318.

Miltenburg, J. (1998). Balancing U-lines in a multiple U-line facility. *European Journal of Operational Research* **109**, 1–23.

Minoux, M. (1983). *Programmation mathématique : Théorie et algorithmes*. Collection technique et scientifique des télécommunications DUNOD, Paris.

Miralles, C., J.P. García-Sabater, C. Andrés et M. Cardos (2007). Advantages of assembly lines in sheltered work centres for disabled. A case study. *International Journal of Production Economics*. À paraître, doi :10.1016/j.ijpe.2007.02.023.

Mînzu, V. et J.M. Henrioud (1998). Stochastic algorithm for tasks assignment in single or mixed model assembly lines. *Journal Européen des Systèmes Automatisés* **32**(7-8), 831–851.

Moodie, C.L. et H.H. Young (1965). A heuristic method of assembly line balancing for assumptions of constant or variable work element times. *Journal of Industrial Engineering* **16**(1), 23–29.

Nicosia, G., D. Pacciarelli et A. Pacifici (2002). Optimally balancing assembly lines with different workstations. *Discrete Applied Mathematics* **118**, 99–113.

Niebel, B.W. (1965). Mechanized process selection for planning new designs. *ASME paper*.

Nkasu, M.M. et K.H. Leung (1995). A stochastic approach to assembly line balancing. *International Journal of Production Research* **33**, 975–991.

Paris, H. (1995). *Contribution à la conception automatique des gammes d'usinage : le problème du posage et du bridage de pièces*. Thèse de Doctorat. Université Joseph Fournier.

Park, K., S. Park et W. Kim (1996). A heuristic for an assembly line balancing problem with incompatibility, range and partial precedence constraints. *Computers and Industrial Engineering* **32**(2), 321–332.

Pastor, R., C. Andres, A. Duran et M. Perez (2002). Tabu search algorithms for an industrial multi-product and multi-objective assembly line balancing problem, with reduction of the task dispersion. *Journal of the Operational Research Society* **53**, 1317–1323.

Pastor, R. et A. Corominas (2000). Assembly line balancing with incompatibilities and bounded workstation loads. *Ricerca Operativa* **30**, 23–45.

Pateloup, V., E. Duc et P. Ray (2004). Corner optimization for pocket machining. *International Journal of Machine Tools and Manufacture* **44**, 1343–1353.

Patterson, J.H. et J.J. Albracht (1975). Assembly line balancing: $0-1$ programming with Fibonacci search. *Operations Research* **23**, 166–174.

Pinnoi, A. et W.E. Wilhelm (1997). A branch and cut approach for workload smoothing on assembly lines. *INFORMS Journal on Computing* **9**, 335–280.

Pinnoi, A. et W.E. Wilhelm (1998). Assembly system design: A branch and cut approach. *Management Science* **44**, 103–118.

Pinto, P.A., D.G. Dannenbring et B.M. Khumawala (1981). Branch and bound and heuristic procedures for assembly line balancing with paralleling of stations. *International Journal of Production Research* **19**, 565–576.

Pitsoulis, L.S. et M.G.C. Resende (2002). Greedy randomized adaptive search procedures. *Handbook of Applied Optimization* (P.M. Pardaloset et M.G.C. Resende, Eds.). pp. 168–183. Oxford University Press, New York.

Ponnambalam, S.G., P. Aravindan et G.M. Naidu (1999). A comparative evaluation of assembly line balancing heuristics. *International Journal of Advanced Manufacturing Technology* **15**, 577–586.

Ponnambalam, S.G., P. Aravindan et G.M. Naidu (2000). A multi-objective genetic algorithm for solving assembly line balancing problem. *International Journal of Advanced Manufacturing Technology* **16**, 341–352.

Pruvot, F. (1993). *Conception et calcul des machines-outils - volume* 1. Presses polytechniques et universitaires romandes.

Rachamadugu, R. et B. Talbot (1991). Improving the equality of workload assignments in assembly lines. *International Journal of Production Research* **29**, 619–633.

Raouf, A. et C. Tsui (1982). A new method for assembly line balancing having stochastic work elements. *Computers and Industrial Engineering* **6**, 131–148.

Rekiek, B. (2001). *Assembly Line Design*: *Multiple objective grouping genetic algorithm and the balancing of mixed-model hybrid assembly line*. Thèse de doctorat. Université Libre de Bruxelles.

Rekiek, B., et A. Delchambre (2006). *Assembly Line Design*: *The Balancing of Mixed-Model Hybrid Assembly Lines with Genetic Algorithms*. Springer.

Rekiek, B., A. Dolgui, A. Delchambre et A. Bratcu (2002*a*). State of art of assembly lines design optimisation. *Annual Reviews in Control* **26**(2), 163–174.

Rekiek, B., P. De Lit et A. Delchambre (2002*b*). Hybrid assembly line design and user's preferences. *International Journal of Production Research* **40**, 1095–1111.

Rekiek, B., P. De Lit, F. Pellichero, T. L'Eglise, P. Fouda, E. Falkenauer et A. Delchambre (2001). A multiple objective grouping genetic algorithm for assembly line design. *Journal of Intelligent Manufacturing* **12**, 467–485.

Renaud, J., V. Boly et V. Rault-Jacquot (2001). Knowledge capitalization and time reduction within the new product development process: some applications. *The International Journal of Concurrent Engineering*: *Research and Applications (CERA)* **9**(4), 295–307.

Resende, M.G.C. et C.C. Ribeiro (2003). Greedy randomized adaptive search procedures. *Handbook of Metaheuristics* (F. Glover et G. Kochenberger, Eds.). pp. 219–249. Kluwer Academic Publishers.

Resende, M.G.C., L.S. Pitsoulis et P.M. Pardalos (2000). Fortran subroutines for computing approximate solutions of MAX-SAT problems using GRASP. *Discrete Applied Mathematics* **100**, 95–113.

Rubinovitz, J. et G. Levitin (1995). Genetic algorithm for assembly line balancing. *International Journal of Production Economics* **41**, 343–354.

Ruby, J.-N. et R. Maranzana (2007). Gpa-664, fabrication assistée par ordinateur, support des cours. Université du Québec, École de technologie supérieure, http ://www.gpa.etsmtl.ca/cours/gpa664/Documents_2007_Ete/FAO_Fraisage_2007.pdf.

Sabuncuoglu, I., E. Erel et M. Tanyer (2000). Assembly line balancing using genetic algorithms. *Journal of Intelligent Manufacturing* **11**, 295–310.

Salomons, O.W. (1995). *Computer Support in the Design of Mechanical Products. Constraint specfication and satisfaction in feature-based design for manufacturing*. Thèse de doctorat. Université de Twente (Pays-Bas).

Salveson, M.E. (1955). The assembly line balancing problem. *Journal of Industrial Engineering* **6**(3), 18–25.

Sarin, S.C., E. Erel et E.M. Dar-El (1999). A methodology for solving single-model, stochastic assembly line balancing problem. *OMEGA The International Journal of Management Science* **27**, 525–535.

Sarker, B.R. et J.G. Shanthikumar (1983). A generalized approach for serial or parallel line balancing. *International Journal of Production Research* **21**, 109–133.

Sarma, S.E. et P.K. Wright (1996). Algorithms for the minimization of setups and tool changes in "simply fixturable" components in milling. *Journal of Manufacturing Systems* **15**(2), 95–112.

Sawik, T. (2002). Monolithic vs. hierarchical balancing and scheduling of a flexible assembly line. *European Journal of Operational Research* **143**, 115–124.

Scholl, A. (1999). *Balancing and sequencing of assembly lines*. Physica, Heidelberg.

Scholl, A. et C. Becker (2006). State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research* **168**, 666–693.

Scholl, A. et R. Klein (1997). SALOME: a bidirectional branch and bound procedure for assembly line balancing. *INFORMS Journal on Computing* **9**, 319–334.

Scholl, A. et R. Klein (1999*a*). Balancing assembly lines effectively - A computational comparison. *European Journal of Operational Research* **114**, 50–58.

Scholl, A. et R. Klein (1999*b*). ULINO: Optimally balancing U-shaped JIT assembly lines. *International Journal of Production Research* **37**, 721–736.

Scholl, A. et S. Voβ (1996). Simple assembly line balancing – heuristic approaches. *Journal of Heuristics* **2**, 217–244.

Scholl, A., N. Boysen et M. Fliedner (2007). The sequence-dependent assembly line balancing problem. *Operations Research Spectrum*. À paraître, doi : 10.1007/s00291-006-0070-3.

Seco CrownLoc® (2004). More precision holes at a lower cost. http :// www.secotools.com/upload/international/library/pdf/drilling/Seco%20CrownLoc%202004.pdf.

Shah, J.J., D.C. Anderson et Y.S. Kim (2001). A discourse on geometric feature recognition from CAD models. *Journal of Computing and Information Science in Engineering* **1**, 41–51.

Shah, J.J. et Mantyla, M., Eds. (1995). *Parametric and feature-based CAD/CAM*. Wiley-Interscience, New York (USA).

Shtub, A. (1984). The effect of incompletion cost on line balancing with multiple manning of work stations. *International Journal of Production Research* **22**, 235–245.

Sniedovich, M. (1981). Analysis of a preference order assembly line problem. *Management Science* **27**, 1067–1080.

Son, S.Y. (2000). Design Principles and Methodolgies for Reconfigurable Machining Systems. PhD thesis. University of Michigan.

Sotskov, Y.N., A. Dolgui et F. Werner (2001). Mixed graph coloring for unit-time job-shop scheduling. *International Journal of Mathematical Algorithms* **2**, 289–323.

Sphicas, G.P. et F.N. Silverman (1976). Deterministic equivalents for stochastic assembly line balancing. *AIIE Transactions* **8**, 280–282.

Stark, J., Ed. (2005). *Product Lifecycle Management*: 21*st century Paradigm for Product Realisation, Series*: *Decision Engineering*. Springer.

Suresh, G. et S. Sahu (1994). Stochastic assembly line balancing using simulated annealing. *International Journal on Production Research* **32**(8), 1801–1810.

Szadkowski, J. (1997). Critical path concept for multi-tool cutting processes optimization. *Manufacturing, Modeling, Management and Control IFAC Symposium*. Elsevier. Vienna, Austria. pp. 393–398.

Talbot, F.B., J.H. Paterson et W.V. Gehrlein (1986). A comparative evaluation of heuristic line balancing techniques. *Management Science* **32**, 430–454.

Tonge, F.M. (1960). Summary of a heuristic line balancing procedure. *Management Science* **7**, 21–39.

Toumine, A. (2007). Eléments de cours : usinage v1.1. http://gcppcinsa.insa-lyon.fr/ãtoumine/index.html.

Trabelsi, A., S. Meeran et M. Carrard (1995). Recognizing cavity volumes using a cell evaluated and directed adjacency graph (CEDAG). *Revue Internationale de CFAO et d'informatique graphique* **10**(6), 607–622.

Tseng, Y. et S. Joshi (1994). Recognizing multiple interpretations in 2D1/2 machining of pockets. *International Journal of Production Research* **32**(5), 1063–1086.

Tsujimura, Y., M. Gen et E. Kubota (1995). Solving fuzzy assembly-line balancing problem with genetic algorithms. *Computers and Industrial Engineering* **29**, 543–547.

Tzur, M. et A. Altman (2004). Minimization of tool switches for a flexible manufacturing machine with slot assignement of different tool sizes. *IIE Transactions* **36**, 95–110.

Urban, T.L. et W.C. Chiang (2006). An optimal piecewise-linear optimization of the U-line balancing problem with stochastic task times. *European Journal of Operational Research* **168**, 771–782.

Vijayakumar, K., G. Prabhaharan, P. Asokan et R. Saravanan (2003). Optimization of multi-pass turning operations using ant colony system. *International Journal of Machine Tools and Manufacture* **43**, 1633–1639.

Vilarinho, P.M. et A.S. Simaria (2002). A two-stage heuristic method for balancing mixed-model assembly lines with parallel workstations. *International Journal of Production Research* **40**, 1405–1420.

Villeneuve, F. (2003). Génération automatique des processus de fabrication. *Fabrication Assistée par Ordinateur* (A. Bernard, Ed.). Chap. 7, pp. 295–350. Editions HERMES.

Wang, J., T. Kuriyagawa, X.P. Wei et D.M. Guo (2002). Optimization of cutting conditions for single pass turning operations using a deterministic approach. *International Journal of Machine Tools and Manufacture* **42**, 1023–1033.

Wee, T.S. et M.J. Magazine (1986). Assembly line balancing as generalized bin packing. *Operations Reasearch Letters* **1**, 56–58.

Wei, S.-Y., C.-C. Lo et C. Chang (1997). Using throughput profit for selecting manufacturing process plan. *Computers and Industrial Engineering* **32**(4), 939–948.

Wilhelm, W.E. (1999). A column-generation approach for the assembly system design problem with tool changes. *International Journal of Flexible Manufacturing Systems* **11**, 177–205.

Williams, H.P., Ed. (1993). *Model building in mathematical programming*. John Wiley & Sons.

Wilson, J.M. (1986). Formulation of a problem involving assembly lines with multiple manning of work stations. *International Journal of Production Research* **24**, 59–63.

Womer, N.K. (1979). Learning curves, production rate and program cost. *Management Science* **25**, 312–319.

www1 (Octobre, 2007). ftp ://trf.education.gouv.fr/pub/edutel/siac/siac2/jury/2006/ caplp_ext/prod8.pdf.

Xu, X. et S. Hinduja (1998). Recognition of rough machining features in 2D1/2 components. *Computer-Aided Design* **30**(7), 503–516.

Yelle, L.E. (1979). The learning curve: historical review and comprehensive survey. *Decision Sciences* **10**, 302–328.

Zhang, G.W., S.C. Zhang et Y. S. Xu (2002). Research on flexible transfer line schematic design using hierarchical process planning. *Journal of Materials Processing Technology* **129**, 629–633.

École Nationale Supérieure des Mines de Saint-Étienne

N° d'ordre : 454 GI

Name Surname : Olga Guschinskaya

Thesis Title : Decision aid tools for the preliminary design of machining systems equipped with multi-spindle heads

Specialty : Industrial Engineering

Keywords : Machining systems, Line balancing, Combinatorial optimization, Graph theory, Mixed integer programming, Heuristics, Metaheuristics

Abstract

This work concerns the preliminary design stage for machining systems equipped with multi-spindle heads, which are widely used in machining industry for high volume production. At this design stage the challenge is to find for a given part an optimal system configuration by minimizing its cost and satisfying all technological, technical and economical constraints.

A number of combinatorial problems appearing at this design stage are discussed in this thesis, and in particular, the problem of configuration optimization is studied for the three following machining systems: transfer machines, machines with a mobile table and machines with a rotary table.

At first, different mathematical models are suggested for each of these systems. At second, a number of optimization methods are developed for their resolution, such as algorithms for lower bounds calculation, pre-processing procedures, exact and heuristic methods and mixed approaches using several of these algorithms.

On the basis of the obtained theoretical results, a prototype of decision support system for the preliminary design stage for machining systems is developed. This prototype has been successfully tested on several industrial cases.

École Nationale Supérieure des Mines
de Saint-Étienne

N° d'ordre : 454 GI

Prénom Nom : Olga Guschinskaya

Titre de la thèse : Outils d'aide à la décision pour la conception en avant-projet des systèmes d'usinage à boîtiers multibroches

Spécialité : Génie Industriel

Mots clefs : Systèmes d'usinage, Equilibrage des lignes, Optimisation combinatoire, Théorie des graphes, Programmation linaire en variables mixtes, Heuristiques, Métaheuristiques

Résumé

Les travaux de recherche effectués dans le cadre de cette thèse concernent essentiellement la conception en avant-projet de systèmes d'usinage dédiés à la production en grande série. Lors de cette phase de conception, l'objectif principal est de trouver, pour une pièce donnée, une configuration du système d'usinage qui satisfaisait, à coût minimal, les contraintes technologiques, techniques et économiques existantes.

Plusieurs problèmes combinatoires posés par cet objectif sont étudiés dans la thèse. Plus concrètement, nous nous sommes intéressés aux problèmes d'optimisation de la configuration des trois types de systèmes d'usinage suivants : machines de transfert, machines à table mobile et machines à table circulaire pivotante.

Pour chacun de ces systèmes, nous avons proposé, dans un premier temps, différents modèles mathématiques. Dans un deuxième temps, nous avons développé de nombreux outils d'optimisation dédiés à leur résolution, tels que des algorithmes de calcul des bornes inférieures, des procédures de pré-traitement, des algorithmes de résolution exacte et approchée, et des approches mixtes utilisant plusieurs de ces algorithmes.

Ces travaux ont permis de concevoir un prototype de logiciel destiné à supporter les différentes étapes de conception en avant projet d'un système d'usinage. Ce logiciel a été testé avec succès sur plusieurs cas industriels.