Habilitation à Diriger des Recherches

Rémy Dupas

Habilitation à Diriger des Recherches

présentée à

L'Université d'Artois

par

Rémy DUPAS

Amélioration de performance des systèmes de production : apport des algorithmes évolutionnistes aux problèmes d'ordonnancement cycliques et flexibles

Soutenue le 10 décembre 2004

Jury :

Rapporteurs :		
	Philippe Chrétienne	Professeur, Université Paris-VI
	Bernard Penz	Professeur, Institut National Polytechnique de Grenoble
	Marie-Claude Portmann	Professeur, Institut National Polytechnique de Lorraine
Examinateurs :		
	René Soenen	Professeur, Université Claude Bernard Lyon 1
	Christian Tahon	Professeur, Université de Valenciennes
	Gilles Goncalves	Professeur, Université d'Artois
	Daniel Jolly	Professeur, Université d'Artois

Je tiens à remercier :

- Les membres du jury qui ont accepté d'être rapporteurs et examinateurs de mon travail ; c'est un honneur et un encouragement fort à poursuivre mon activité de recherche.

- Gilles ainsi que Daniel, avec qui j'ai partagé les doutes et les difficultés liés à la création ex nihilo d'une entité de recherche et qui m'ont aidé à rendre cette gageure possible.

- Mon collègue Jean Pierre Parsy à qui je dois beaucoup pour l'apprentissage et la pédagogie de la programmation en Java.

- le Docteur Albert Dupas, mon père, qui par sa grande humanité, sa soif de connaissance et son intelligence a été un exemple et m'a toujours encouragé, notamment en m'aidant à traduire des articles en anglais jusqu'à sa quatre-vingt dixième année… Il n'aurait d'ailleurs pas souhaité que j'écrive ces quelques mots.

- Mon épouse Valérie, pour les relectures bienveillantes de ce document.

Je dédie ce travail à Romain, Pauline …

Introduction générale

Ce mémoire traite de l'amélioration de performance des systèmes de production discrets au niveau opérationnel selon deux points de vue complémentaires.

Le premier concerne les problèmes rencontrés dans les applications réelles du domaine industriel. La caractéristique essentielle de ces systèmes est leur complexité qui rend difficile la modélisation et l'optimisation par des approches analytiques. Sur ce plan, l'apport de notre travail réside dans l'intérêt d'une approche combinant de multiples méthodes de résolution pour l'aide à l'amélioration de performance de ces systèmes. Il s'agit de la mise œuvre des outils et concepts de la simulation des systèmes à événements discrets associés aux outils d'analyse systématique que sont les plans d'expériences, pour l'aide à l'analyse des performances du système. Il s'agit également du couplage de cette approche de la simulation avec des méthodes d'optimisation méta-heuristiques développées en recherche opérationnelle pour l'amélioration de performance du système. Cette combinaison de méthodes de résolution se révèle un moyen efficace de pallier la difficulté du problème énoncé et d'apporter des solutions efficaces.

Le second point de vue abordé dans ce travail concerne le domaine plus académique des problèmes d'ordonnancement et plus particulièrement les problèmes de nature cyclique et flexible. La contribution du mémoire porte sur l'exploitation du concept des algorithmes évolutionnistes pour la résolution de ces problèmes. Le paradigme des algorithmes évolutionnistes ou évolutifs s'inspire du processus de l'évolution naturelle. Il est apparu que les méthodes d'optimisation approchées en général et évolutionnistes en particulier n'ont été que très peu mises en œuvre pour la résolution de ces deux catégories de problèmes d'ordonnancement dont la complexité est de manière générale de type NP-difficile. L'idée développée à travers nos travaux est de montrer le potentiel de ces algorithmes pour la résolution des problèmes cycliques et flexibles et donc d'étendre leur domaine de résolution qui était jusqu'alors essentiellement restreint aux problèmes d'ordonnancement standards, non cycliques.

L'organisation du mémoire correspond aux deux points de vue mentionnés ci-dessus et comporte deux parties principales.

La première partie propose une méthode générale d'amélioration de performance des systèmes de production qui est constituée de deux étapes. La première étape est centrée sur l'évaluation de performance du système en utilisant les plans d'expérience dans le contexte de la simulation des systèmes à événements discrets. La seconde étape aborde la mise en œuvre de méthodes méta-heuristiques pour l'amélioration de performance. Ces travaux ont été validés sur une ligne de production de grande série du secteur automobile.

La seconde partie de ce mémoire aborde la résolution de problèmes d'ordonnancement par les méthodes évolutionnistes. Cette partie se décompose plus précisément en six chapitres dont les deux premiers concernent un état de l'art des domaines que nous abordons: dans un premier temps un aperçu du domaine vaste de

l'ordonnancement est donné en insistant plus particulièrement sur les problèmes cycliques puis dans un deuxième temps, une présentation des méthodes évolutionnistes est proposée en se focalisant sur l'application de ce concept au domaine particulier de l'ordonnancement. Les trois chapitres suivants abordent la résolution par les algorithmes génétiques de problèmes d'ordonnancement dans le domaine cyclique ou flexible. Le troisième chapitre propose une méthode de modélisation par réseaux de Petri et une résolution approchée du job shop cyclique linéaire. Le quatrième chapitre décrit une méthode de résolution des problèmes d'ordonnancement en production flexible manufacturière. Enfin, le cinquième chapitre présente une méthode de résolution du job shop flexible multicritère. Finalement, nous présentons un ensemble de perspectives de ces travaux de recherche qui portent sur les trois types de problématiques d'ordonnancement étudiés ainsi que sur les problèmes de transport et notamment les problèmes de routage de véhicules.

Partie A Amélioration de performance des systèmes de production

Cette première partie traite du problème général de l'amélioration de performance des systèmes de production discrets. Les niveaux décisionnels concernés sont préférentiellement les niveaux "tactiques" et "opérationnels". Néanmoins, la complexité des problématiques du monde réel tend ici à être prise en compte en intégrant l'ensemble des contraintes de fonctionnement et sans se restreindre au cadre précis d'un problème académique.

Nous proposons dans ce domaine une approche de l'amélioration de performance basée sur la simulation des systèmes discrets. L'intégralité de cette approche, qui est issue de travaux effectués dans le cadre du GRP[1], est décrite en Annexe 8. Elle comporte deux volets qui sont résumés succinctement ci-dessous.

I Apport des plans d'expériences pour l'évaluation de performance en simulation

Le premier volet concerne de l'évaluation de performance. Dans ce domaine une approche originale basée sur les plans d'expériences dans le contexte de la simulation a été proposée. Cette approche est développée dans Dupas (2003a) et en Annexe 8.3. Cet article dresse notamment les particularités de l'utilisation des plans d'expériences sur un système simulé. De plus, un cas d'étude montre l'intérêt réel de la technique des plans d'expériences dans ce domaine. Il s'agit d'un outil puissant et adapté qui doit donc faire partie de la palette des outils indispensables à l'analyse de performances.

L'ensemble des potentialités des plans d'expériences n'a cependant pas été exploité. Il conviendrait d'envisager d'une part leur utilisation couplée avec les techniques d'analyse de la variance pour une analyse de meilleure qualité et d'autre part leur exploitation dans le cadre d'une stratégie de robustesse au bruit. En outre, sur le plan pratique, des outils d'aide à la mise en œuvre des plans d'expériences existent ; il conviendrait cependant de pouvoir les intégrer plus facilement au sein des progiciels de simulation.

II Apport des approches méta-heuristiques pour l'amélioration de performance

Le seconde volet concerne l'amélioration de performance des systèmes de production basée sur l'utilisation des techniques méta-heuristiques. Cette approche est développée dans Burlat (2003) et en Annexe 8.2 Cet article décrit notamment les principales approches d'optimisation méta-heuristiques exploitables dans le cadre de l'amélioration de performance des systèmes par couplage avec la simulation.

La méthode proposée a été mise en oeuvre dans une application industrielle réelle; celle-ci est présentée in extenso dans l'article de Cavory (2001) qui est donné en Annexe 8.1. Ces travaux concernent une ligne de production de grande série dans le domaine automobile. Cette ligne est constituée d'une succession de machines reliées par des convoyeurs. Les changements d'outils sur les machines sont exécutés de manière périodique après

[1] Groupement de Recherche en Productique ; thème évaluation de performance.

l'usinage d'un nombre déterminé de pièces. L'objectif vise à améliorer le rendement de la ligne, par une gestion optimisée au cours du temps des changements d'outils, de façon à minimiser leur impact sur le flux de production. L'approche adoptée est basée sur le couplage entre un algorithme génétique et un simulateur à événements discrets. Chaque solution ou individu définit le paramétrage de chacun de ces changements d'outils sur toutes les machines de la ligne. Le simulateur évalue le paramétrage proposé en effectuant une simulation d'un modèle "allégé" de la ligne de production. L'approche permet d'obtenir un gain de production significatif.

De nombreux problèmes relatifs à l'utilisation des méta-heuristiques pour l'amélioration de performance restent à approfondir et portent notamment sur le choix de la méthode adaptée à une application donnée ainsi que le réglage des paramètres afférents à chaque méta-heuristique.

Notons enfin que cette thématique est étudiée dans le cadre du groupe de travail "Modélisation Multiple et Simulation" (MMS) du GDR "Modélisation Analyse et Conduite des Systèmes dynamiques" (MACS), pôle "Sciences et Techniques de la Production de Biens et de Services" (STP).

Partie B **Méthodes de résolution évolutionnistes en ordonnancement cyclique et flexible**

I Ordonnancement : généralités et problèmes cycliques

L'objet de ce chapitre n'est pas de réaliser une présentation exhaustive du vaste domaine de l'ordonnancement ; un nombre important d'ouvrages ou d'articles de synthèse sont disponibles sur ce sujet. La communauté de recherche française est active dans ce domaine et plusieurs groupes de travail sont dédiés à l'étude de ces problèmes, tels que GOThA[2] (Groupe de Recherche en Ordonnancement Théorique et Appliqué) , Bermudes[3] et ORDO.

Il s'agit simplement dans ce chapitre d'introduire les concepts et la notation de base qui permettent de décrire dans un cadre unifié les trois problèmes étudiés ensuite. L'accent est mis sur la présentation du problème du job shop. En effet, ce problème classique d'ordonnancement possède de fortes similarités avec les trois problèmes sur lesquels se focalise notre travail. Finalement, la problématique de l'ordonnancement cyclique et les principales classes de problèmes rencontrés dans ce domaine sont abordées.

I.1 Notions de base en ordonnancement

Les problèmes d'ordonnancement sont présents dans tous les secteurs d'activités de l'économie depuis l'informatique jusqu'à l'industrie manufacturière. C'est pour cette raison qu'ils ont fait et continuent de faire l'objet de nombreux travaux de recherche (Chrétienne, 1997).

Résoudre un problème d'ordonnancement consiste à ordonnancer i.e. programmer ou planifier, dans le temps l'exécution des tâches en leur attribuant les ressources nécessaires matérielles ou humaines de manière à satisfaire un ou plusieurs critères préalablement définis, tout en respectant les contraintes de réalisation (GOThA, 1993 ; Lopez 2001).

Les problèmes d'ordonnancement d'ateliers constituent sûrement pour les entreprises une des difficultés importantes de leur système de gestion et de pilotage. En effet, c'est à ce niveau que doivent être prises en compte les caractéristiques réelles multiples et complexes des ateliers, ainsi que les perturbations, aussi bien internes i.e. panne de machine, absence d'opérateurs, qu'externes i.e. variation de la demande, qui viennent les modifier.

Dans ce type d'ordonnancement, les ressources sont généralement des machines qui ne peuvent réaliser qu'une tâche ou opération à la fois i.e. ressource disjonctive, et chaque travail à ordonnancer concerne un produit ou un lot de produits à fabriquer en respectant une gamme de fabrication. Cette gamme précise un ordre strict entre les opérations qui la composent : on parlera de contraintes de précédence.

Les problèmes d'ordonnancement d'ateliers se décomposent en trois grandes classes de problèmes selon que la gamme de fabrication est commune à tous les travaux, i.e. atelier à cheminement unique ou *flow shop*, spécifique à chaque travail i.e. atelier à cheminements multiples ou *job shop*, ou que cette gamme n'est pas définie i.e. atelier à cheminement libre *open shop*.

[2] http://www-poleia.lip6.fr/~sourd/gotha/

[3] http://bermudes.univ-bpclermont.fr/

Dans le contexte de l'ordonnancement d'ateliers dans lequel se placent les travaux que nous présentons dans ce mémoire, les tâches sont appelées couramment opérations. En conséquence, cette unique dénomination est employée dans la suite de ce document.

Cette introduction est structurée en quatre parties qui abordent successivement les notations utilisées puis les hypothèses de travail communes à l'ensemble des chapitres de cette seconde partie (partie B) ainsi que le rappel de quelques définitions courantes en ordonnancement et enfin, la présentation d'un outil de modélisation du problème job shop. Ce problème a été choisi en raison d'une forte parenté avec les trois problèmes étudiés par la suite.

I.1.1 Notations

En préalable il convient de rappeler la notation usuelle des problèmes d'ordonnancement qui est formée du triplet Ñ | Ò | Ó, dans lequel Ñ décrit les caractéristiques machines du problème, Ò représente les caractéristiques des travaux et Ó désigne le ou les critères à optimiser. A titre d'exemple, le job shop à trois machines, deux jobs, ayant pour critère C_{max}, se note J3|n=2|C_{max}. Cette notation de base (Graham, 1979) a été entendue afin de pouvoir représenter l'ensemble des variantes des problèmes d'ordonnancement.

Les notations couramment utilisées dans le domaine de l'ordonnancement et communes à l'ensemble des chapitres de ce document sont mentionnées ci-dessous.

- Ensemble des m machines : $M = \{M_k\}_{k=1}^{m}$

- Ensemble des n travaux ou jobs à réaliser : $J = \{J_i\}_{i=1}^{n}$.

- Ensemble des opérations : $T = \{o_{ij}\}$ $i \in \{1,..,n\}$, $j \in \{1,..,m_i\}$ où o_{ij} est la $j^{ième}$ opération du job J_i et m_i est le nombre d'opérations du job J_i. N correspond au nombre total d'opérations : $N = \sum_{i=1}^{n} m_i$. De manière complémentaire, l'ensemble des opérations du job J_i se note: $T_i = \{o_{ij} \in T\}$ $j \in \{1,..,m_i\}$.

- Le temps d'exécution de l'opération o_{ij} est noté p_{ij} ; il est défini en nombre d'unité de temps (notée u.t.).

> **Note 1**: dans le cas spécifique du job shop flexible étudié au chapitre V (Partie B), dans lequel il existe plusieurs machines capables de réaliser l'opération o_{ij} avec des temps de réalisation différents, le temps d'exécution de l'opération o_{ij} sur la machine M_k est noté p_{ijk}.

- Chaque job J_i consiste en une succession ordonnée de m_i opérations appelée gamme : $o_{i1} - o_{i2} - .. - o_{imi}$ dans laquelle – désigne la relation de précédence entre les opérations. Chaque opération de la gamme est affectée à une machine de l'ensemble M.

> **Note 2:** dans certains problèmes, une seconde notation peut être utilisée pour les machines : la machine affectée à l'opération o_{ij} est notée m_{ij}.

- t_{ij} et C_{ij} représentent respectivement les dates de début réel et de fin de l'opération o_{ij}

- C_i : date de fin du travail J_i ; $C_i = \max(C_{ij})$ 1Rj Rn_i

- C_{max}: date d'achèvement de l'ordonnancement; $C_{max} = \max (C_i)$ 1RiRn ; la minimisation du C_{max} est appelée classiquement *makespan*.

Dans le cas d'opérations répétitives ou cycliques ayant un nombre infini d'occurrences, la notation de Hanen et Munier (1995) est utilisée :

- la $q^{iéme}$ occurrence de l'opération "a" se note <a,q>

- $t_a(q)$ représente la date de début de l'occurrence <a, q>

En complément, remarquons que la date de fin de la $q^{ième}$ exécution du job J_i , notée C_i^q , est définie par:

$$C_i^q = \max_{j=1}^{m_i} \{t_{ij}(q) + p_{ij}\}$$

Enfin, deux autres notations non spécifiques au domaine de l'ordonnancement sont utilisées dans la suite du document:

- **v** symbolise le vecteur v

- $\lceil x \rceil$: entier supérieur ou égal à x.

I.1.2 Hypothèses de travail

Les hypothèses communes à l'ensemble des problèmes traités dans notre travail sont mentionnées ci-dessous.

Les machines sont:

- indépendantes les unes des autres et disponibles en un seul exemplaire

- disponibles pendant toute la période de l'ordonnancement : il n' y a pas d'arrêt ou de panne

- mono-opération: elles exécutent une seule opération à un instant donné.

Les opérations sont :

- non-préemptives : une opération en cours d'exécution ne peut être interrompue

- simplifiées en considérant que les temps de préparation des machines et les durées de transport entre deux opérations sont négligeables

- associées à des temps opératoires déterministes.

Le contexte de l'ordonnancement recherché est de type statique : l'ensemble des jobs à ordonnancer est connu à l'avance : il n'y a pas d'aléas, ni d'occurrences de nouveaux jobs au cours de cet ordonnancement.

I.1.3 Définitions et propriétés

Ce paragraphe consiste à rappeler certaines définitions et propriétés essentielles qui permettent de caractériser le type des solutions recherchées et ainsi restreindre la taille de l'espace de recherche dans un

problème d'ordonnancement. Plus précisément, l'ensemble des ordonnancements admissibles, respectant toutes les contraintes d'un problème, peut être décomposé en plusieurs sous-classes définies ci-dessous (Esquirol, 1999).

Définition I-1 Propriété de dominance

Soit S l'ensemble des solutions ou ordonnancements d'un problème. Un sous-ensemble d'ordonnancements est dit dominant, pour un critère donné, si ce sous-ensemble contient au moins une solution optimale du problème.

Définition I-2 Critère régulier

Un critère à minimiser Z, fonction des dates de fin d'ordonnancement C_i, est dit régulier si : $\forall x,y \in S, \forall i \in \{1,..,n\}$ $C_i(x) \# C_i(y) \Rightarrow Z(C_1(x),..,C_n(x)) \# Z(C_1(y),..,C_n(y))$ avec $C_i(x)$ respectivement $C_i(y)$, la date d'achèvement du job i dans l'ordonnancement x, respectivement y.

De cette définition, il est possible de déduire que si un ordonnancement x est meilleur qu'un ordonnancement y, alors il existe au moins un C_i tel que $C_i(x)$ est strictement inférieur à $C_i(y)$.

Définition I-3 Ordonnancement semi-actif

Soit $x \in S$ un ordonnancement de S et S_x l'ensemble des ordonnancements dont les séquences d'opérations sur les machines sont identiques. Un ordonnancement x appartient à la classe des ordonnancements semi-actifs si et seulement si $\exists y \in S_x$ tel que $\forall i \in \{1,..,n\}, C_i(y) \# C_i(x)$ et $j \neq i \in \{1,..,n\} \mid C_i(y) < C_i(x)$

Dans un ordonnancement semi-actif, il est impossible d'avancer une opération sans modifier la séquence des opérations sur la ressource : toutes les opérations sont calées, soit sur l'opération qui la précède dans sa gamme, soit sur l'opération qui la précède sur la machine utilisée.

Définition I-4 Ordonnancement actif

Un ordonnancement $x \in S$ appartient à la classe des ordonnancements actifs si et seulement si $\exists y \in S$ tel que $\forall i \in \{1,..,n\}$, $C_i(y) \# C_i(x)$ et $j \neq i \in \{1,..,n\} \mid C_i(y) < C_i(x)$

En conséquence, dans un ordonnancement actif, il est impossible d'avancer une opération, sans reporter le début d'une autre opération.

Définition I-5 Ordonnancement sans délai

Un ordonnancement $x \in S$ appartient à la classe des ordonnancements sans délai ou sans retard si et seulement si aucune opération n'est mise en attente alors qu'une machine est disponible pour l'exécuter.

La Figure I-1 représente le diagramme d'inclusion des classes d'ordonnancements. Elle fait apparaître que les ordonnancements sans retard sont inclus dans le sous-ensemble des ordonnancements actifs qui sont eux-même inclus dans le sous-ensemble des ordonnancements semi-actifs.

La propriété majeure suivante (Baker, 1974) lie les critères réguliers à la classe des ordonnancements actifs :

Propriété I-1

L'ensemble des ordonnancements semi-actifs est dominant dans les problèmes d'optimisation d'un critère régulier et le sous-ensemble des ordonnancements actifs est le plus petit ensemble dominant.

En conséquence, dans le cas d'un critère régulier, la recherche d'une solution optimale peut être limitée à l'ensemble des ordonnancements actifs, ce qui restreint ainsi la taille de l'espace de recherche.

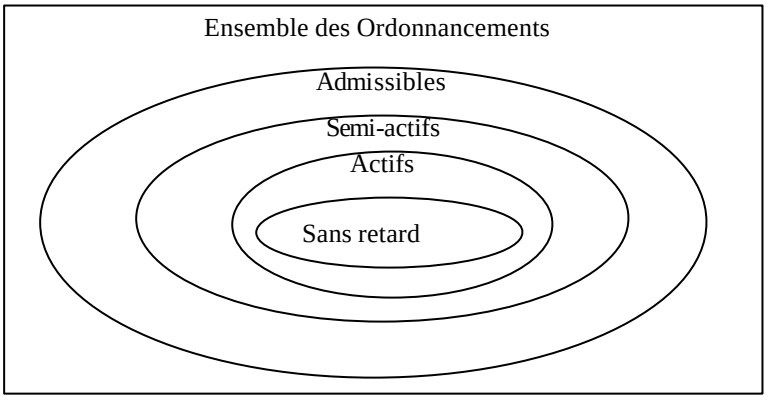


Figure I-1 Inclusion des classes d'ordonnancements

I.2 Le problème du job shop

Le problème général du job shop à n jobs et m machines ayant pour critère la minimisation du *makespan*, noté $Jm||C_{max}$, est un problème de la classe NP-difficile dont l'espace des solutions est au maximum égal à $(n!)^m$. Ce problème d'atelier a été étudié de manière intense ces dernières années et de nombreuses méthodes de résolution exactes ou approchées ont été proposées. Une présentation du problème du job shop est donnée ci-dessous en détaillant successivement un algorithme de construction d'ordonnancements actifs, une méthode de modélisation du problème basée sur les graphes et un aperçu des méthodes de résolution de ce problème.

I.2.1 Algorithme de génération d'ordonnancements actifs

L'Algorithme I-1 décrit la méthode de Giffler et Tompson (1960) qui permet de générer l'ensemble des ordonnancements actifs pour un problème de type job shop.

Cet algorithme est fréquemment utilisé par couplage avec des méthodes de résolution génétiques, c'est pourquoi il est décrit ci-dessous. Cet algorithme détermine au sein de l'ensemble de toutes les opérations restant à placer, celle dont la date de fin est la plus petite, notée o*. Etant donné M_k la machine affectée à cette opération o*, l'opération effectivement placée dans l'ordonnancement est alors sélectionnée parmi l'ensemble de toutes les opérations en conflit sur M_k. La génération de tous les ordonnancements actifs est obtenue en explorant tous les choix possibles de résolution de conflit.

Algorithme I-1 Construction d'ordonnancements actifs (Giffler et Tompson, 1960)

Algorithme I-1

Faire

Déterminer l'ensemble des opérations restant à ordonnancer: E= { o_{ij} ∈ T | o_{ij} non encore ordonnancée}

Calculer la date de fin (C_o) de toutes les opérations o de E

Rechercher o* ∈ E ayant la plus petite date de fin (C_{o*}=min C_o, o ∈ E); soit M_k la machine affectée à o*

Déterminer G l'ensemble des opérations non ordonnancées de M_k, en conflit avec o*

Sélectionner aléatoirement une opération de G

Supprimer l'opération choisie de E

TantQue E@i i.e. il reste des opérations à ordonnancer

I.2.2 Modélisation basée sur les graphes

Le job shop, comme d'autres catégories de problèmes d'ordonnancement, peut être formalisé de façon analytique ou modélisé à l'aide de graphes. Les graphes disjonctifs (Roy, 1970) constituent une méthode de modélisation élégante du problème job shop, utilisé au chapitre III et que nous avons donc choisi de rappeler ici.

Plus précisément, un graphe disjonctif est défini par le triplet G(T,U,D) dans lequel T représente l'ensemble des sommets associés aux opérations, U correspond à l'ensemble des arcs orientés représentant les contraintes de précédence des gammes opératoires et D est associé à l'ensemble des arcs non orientés représentant les conflits d'utilisation d'une ressource.

L'exemple de la Figure I-2 représente un problème constitué de trois machines et huit opérations à ordonnancer. L'arc orienté entre les opérations 1 et 2 exprime une contrainte de précédence entre ces deux opérations. Trois machines sont disponibles pour réaliser l'ensemble des opérations. La première A peut réaliser les opérations 1, 5 et 6. La seconde, appelée B, peut réaliser les opérations 2, 4 et 7. La troisième machine réalise les tâches 3 et 8.

Un ordonnancement admissible est défini par le choix d'un sens pour chaque arc non orienté de D, en respectant la contrainte de ne pas générer de circuit dans le graphe. Dans la solution proposée en Figure I-3 la machine A exécute d'abord l'opération 1 puis l'opération 6 et enfin l'opération 5. La durée totale de l'ordonnancement correspond alors au plus long chemin entre les nœuds début et fin, notés respectivement D et F.

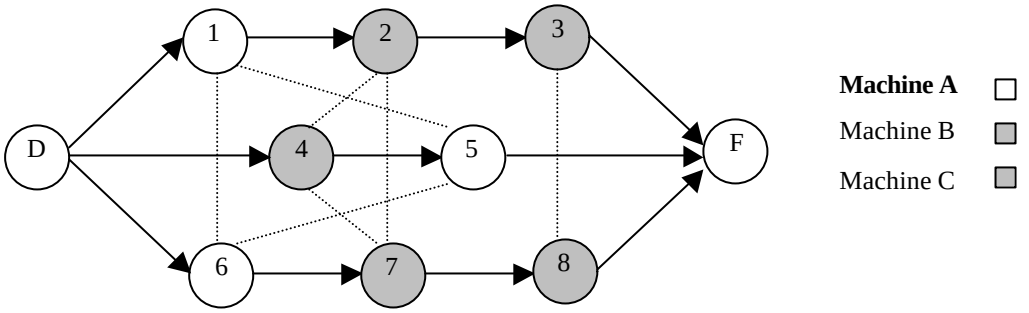


Figure I-2 Job shop : modélisation par les graphes disjonctifs

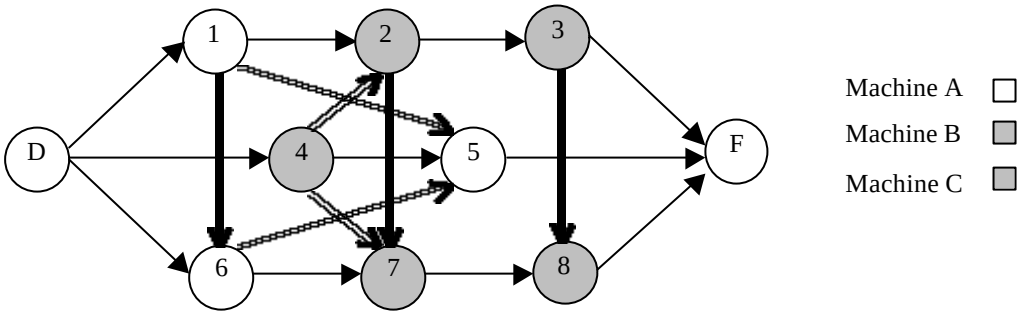


Figure I-3 Une solution au problème de la Figure I-2

I.2.3 Un aperçu des méthodes de résolution

Les méthodes de résolution de ce problème sont très nombreuses (Jain, 1998 ; Jain, 1999 ; Blazewicz, 2001). Les méthodes exactes telles que le *Branch and Bound* encore appelé "Procédure par Séparation et Evaluation" parcourent l'arbre des solutions en élaguant les branches sous-optimales. Pour le problème de minimisation du *makespan*, une borne inférieure du critère d'optimisation est utilisée pour réaliser cet élagage. Ces méthodes ont fait l'objet d'intenses recherches et permettent d'atteindre la solution optimale en quelques minutes pour certains problèmes comportant une centaine d'opérations (N=100).

Parallèlement à ces méthodes exactes, de nombreux travaux se sont focalisés sur les méthodes approchées en raison soit, de leur facilité d'implémentation, soit de leur faible complexité, soit encore en raison de leur caractère générique. Les principales classes de méthodes sont à titre non exhaustif :

- les règles de priorité. Elles sont utilisées pour ordonnancer les opérations sur les machines en se basant sur des règles telles que "Premier arrivé, premier servi ou *FIFO*", "Opération la plus courte d'abord ou *SPT* " etc. Ces règles sont utilisées au sein de l'algorithme de Giffler et Thompson pour la génération d'ordonnancement actif. Elles sont utilisées également pour la construction d'ordonnancement sans retard et sont aussi appelées algorithmes de liste. Plus précisément, à l'instant t, parmi les opérations disponibles i.e. celles dont les prédécesseurs sont achevés, l'opération de plus grande priorité est affectée. Plus généralement, les algorithmes de listes procèdent d'abord à l'élaboration d'une liste de priorités qui est utilisée ensuite pour la construction d'une solution. Les temps de calcul sont relativement courts, mais aucune heuristique ne domine les autres et la qualité des résultats, en distance par rapport à l'optimum, est faible.

- l'heuristique de la "machine goulet" ou *Shifting bottleneck procedure*. Son principe est le suivant :identifier dans un premier temps la machine critique M_i dans l'ensemble des opérations non ordonnancées ; ordonnancer dans un deuxième temps M_i, en utilisant un algorithme exact pour une seule machine ; enfin ré-optimiser les machines déjà ordonnancées. Cette heuristique a été la première à résoudre de façon optimale un problème test formé de 10 jobs à 10 opérations i.e. N=100 opérations au total, et a donné lieu à de nombreux descendants.

- les approches méta-heuristiques. Parmi celles-ci, la méthode de recherche tabou et le recuit simulé sont décrits dans Reeves (1995) et présentés succinctement dans l'Annexe 8 (Dupas, 2003). Les algorithmes évolutionnistes sont décrits au chapitre suivant. La recherche tabou se distingue car elle permet d'obtenir les meilleures performances parmi l'ensemble des approches de résolution possibles. Les performances des réseaux de neurones ainsi que les approches par satisfaction de contraintes demeurent à ce jour faibles.

Il est important de noter que l'ensemble des approches ont des difficultés à résoudre les instances de taille importante et que la plupart d'entre elles améliorent leur performance par des hybridations avec d'autres approches telles que des méthodes de recherche locale. Dans le cas des algorithmes évolutionnistes, l'hybridation est essentielle pour l'obtention de bonnes performances sur des instances de problème de plus d'une centaine d'opérations i.e. N > 100.

I.3 Ordonnancement cyclique

Les notions générales de l'ordonnancement ayant été rappelées ci-dessus, il convient à présent de préciser la problématique de l'ordonnancement cyclique. En effet, ce domaine possède des particularités importantes à la fois au niveau de la définition même du problème d'ordonnancement ainsi qu'au niveau des domaines d'application abordés, conférant ainsi à ce type de problématique une réelle complexité.

I.3.1 Domaines d'application

Les principaux domaines d'activité dans lesquels les problèmes d'ordonnancement cyclique peuvent apparaître sont mentionnés ci-dessous.

- les systèmes de production manufacturiers comportent de nombreuses activités à caractère cyclique. Les opérations de production sur les lignes de production de grandes séries sont fortement répétitives. Ces opérations cycliques apparaissent également dans le domaine de la maintenance systématique. Par exemple, les changements d'outils sur les machines automatisées sont des opérations cycliques : elles sont déclenchées après un nombre de cycles d'usinages fixé. Ces changements d'outils ont donc de multiples occurrences au cours du temps.

- dans le domaine informatique, les calculateurs parallèles posent également des problèmes d'ordonnancement cyclique. L'exécution des structures de contrôle de type itératif sur ces calculateurs à architectures parallèles engendre des problèmes de nature cyclique, lors de la conception de compilateurs.

- certaines activités humaines possèdent également un caractère cyclique. A titre d'exemple, les tâches de réalisation d'emplois du temps au sein d'un organisme de formation consiste généralement à trouver un cycle ou un motif qui peut être répété sur une période donnée telle que le trimestre ou l'année. La conception des tableaux

de marche des systèmes de transport urbain est un autre exemple de problématique cyclique puisque les opérations de ramassage se répètent à une fréquence fixée.

I.3.2 Définitions

Ce paragraphe présente une caractérisation des solutions ainsi que les mesures de qualité d'un problème d'ordonnancement cyclique. Il définit successivement les notions de contraintes de précédences et graphes linéaires et aborde enfin le rôle des réseaux de Petri dans l'ordonnancement cyclique.

I.3.2.1 Caractérisation des solutions

Les solutions d'un problème d'ordonnancement cyclique peuvent dans certains cas être caractérisées sous l'une des deux formes suivantes :

Définition I-6 Périodicité

Un ordonnancement périodique de période "*TC*", nombre réel positif, est défini comme suit:

$$\forall o_{ij} \in T, \forall n \in \mathbb{N} \text{ et } n>1,\ t_{i,j}(n) = t_{i,j}(1) + TC \cdot (n-1)$$

Définition I-7 K-périodicité

La notion de suite k-périodique introduite par Carlier (1988) permet de définir la k-périodicité d'un ordonnancement. Etant donné, k un entier positif et TC un nombre réel positif :

$$\forall o_{ij} \in T, \exists N_0 \mid \forall n \in \mathbb{N} \text{ et } n \geq N_0,\ t_{i,j}(n+k) = t_{i,j}(n) + TC$$

L'entier k est appelé facteur de périodicité, *w* la période ou temps de cycle et k/*TC* représente la fréquence.

Les N_0 premières occurrences de l'opération cyclique t_{ij} correspondent au régime transitoire, tandis que les suivantes définissent le régime permanent.

I.3.2.2 Mesure de qualité des ordonnancements cycliques

L'évaluation de la qualité globale d'un ordonnancement cyclique est généralement basée sur le temps de cycle dont la minimisation équivaut à améliorer le rendement ou débit, des tâches cycliques. Alternativement, cette évaluation porte aussi sur le temps d'écoulement d'un job ou d'un produit. Définissons ci-dessous, le temps de cycle d'une opération o_{ij}, d'un job puis le temps d'écoulement d'un job et la notion d'encours :

Définition I-8 Temps de cycle moyen d'une opération o_{ij}

$$\mathrm{Tc}_{ij} = \lim_{q\to\infty} \frac{t_{ij}(q) + p_{ij}}{q}$$

Définition I-9 Temps de cycle moyen d'un job J_i

Le temps de cycle moyen du job J_i, noté TC_i correspond à la durée moyenne entre deux occurrences du job J_i, calculée sur un horizon infini. Etant donné C_i^q représentant la date de fin de la $q^{ième}$ exécution de J_i : $C_i^q = \max_{j=1}^{m_i}\{t_{ij}(q) + p_{ij}\}$, TC_i peut donc être défini comme suit :

$$TC_i = \lim_{q\to\infty} \frac{1}{q} C_i^q$$

Définition I-10 Temps d'écoulement moyen d'un job

Le temps d'écoulement de la $q^{ième}$ exécution du job J_i, noté TE_i^q, correspond à la durée de réalisation complète de cette occurrence de J_i:

$$TE_i^q = C_i^q - t_{i1(q)}$$

Le temps d'écoulement moyen d'un job correspond à la durée moyenne de réalisation complète des occurrences du job J_i calculée sur un horizon infini :

$$TE_i = \lim_{q\to\infty} \frac{1}{q} \sum_{k=1}^{q} TE_i^k$$

Définition I-11 Encours

Enfin, une mesure corollaire est l'encours qui correspond au nombre de produits en cours de fabrication à chaque temps de cycle. Dans le contexte d'un ordonnancement périodique, l'ordonnancement qui est "infini" en théorie, est réduit en pratique à l'étude d'un cycle. En conséquence, $\forall q > 1, \mathrm{TE}_i^q = \mathrm{TE}_i$

L' encours EC_i du job J_i est donc défini comme suit:

$$\mathrm{EC}_i = \left\lceil TE_i / TC_i \right\rceil$$

La Figure I-4 présente le cas particulier d'un ordonnancement périodique. Il s'agit d'un problème à un seul job constitué de huit opérations et nécessitant trois machines. Les temps opératoires associés à ces opérations sont donnés dans le Tableau I-1. Dans cet exemple, le temps d'écoulement ou temps nécessaire à la réalisation d'un produit est de 16 u.t. et le temps de cycle ou période vaut 9 u.t. Dans cet exemple l'encours vaut 2 u.t.

Numéro d'opérations	1	2	3	4	5	5	7	8
Temps opératoires (u.t.)	1	1	1	3	3	1	1	1

Tableau I-1 Temps opératoires de la Figure I-4

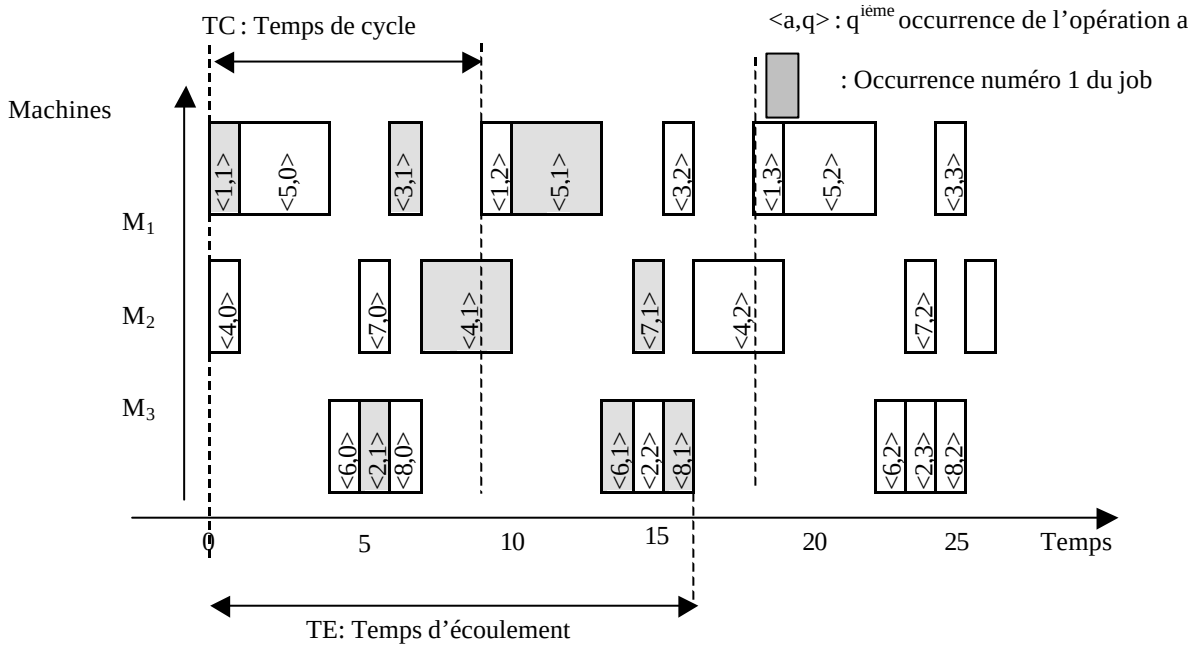


Figure I-4 Temps de cycle et temps d'écoulement d'un ordonnancement périodique

I.3.2.3 Contraintes et graphes linéaires

Définition I-12 Graphe linéaire

Une contrainte linéaire (Munier, 1996) entre deux opérations cycliques "a" et "b", notée e (a,b), est définie par deux nombres entiers strictement positifs α_{ab}, γ_{ab} et deux nombres entiers β_{ab}, ω_{ab}. Le décalage entre les occurrences des opérations "a" et "b" est caractérisé de la façon suivante :

$$\forall n > 0, t_a(a_{ab}n + b_{ab}) + p_a \leq t_b(g_{ab}n + w_{ab})$$

$t_a(\alpha_{ab}\, n + \beta)$: date de début de l'occurrence $<a, \alpha_{ab}n + \beta>$

p_a : durée de l'opération "a"

Cette inégalité contraint l'occurrence $\gamma_{ab}n + \omega_{ab}$ de l'opération "b", à débuter au plus tôt, après que l'occurrence $\alpha_{ab}n + \beta_{ab}$ soit terminée.

Cette contrainte linéaire e (a,b) entre les opérations "a" et "b" est représentée en Figure I-5. les nœuds représentent les opérations "a" et "b" dont les durées opératoires sont respectivement p_a et p_b ; l'arc représente la contrainte linéaire considérée. Ce type de contrainte de précédence entre deux opérations "a" et "b", traduit un écart uniformément croissant, respectivement décroissant, entre les occurrences des opérations "a" et "b". La Figure I-6 présente le diagramme de Gantt associé à la contrainte de la Figure I-5 dans le cas particulier ou le quadruplet (α_{ab}, β_{ab}, γ_{ab} ,ω_{ab}) vaut (2, 1, 1 ,0).

Les contraintes uniformes sont un cas particulier des contraintes linéaires lorsque $\alpha_{ab}=\gamma_{ab}=1$. Les contraintes uniformes expriment quant à elles un décalage constant entre les occurrences des opérations "a" et "b". Elles sont notées de manière simplifiée : $e(a,b)= (\beta_{ab}, \omega_{ab})$.

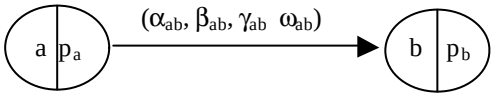


Figure I-5 Une contrainte linéaire

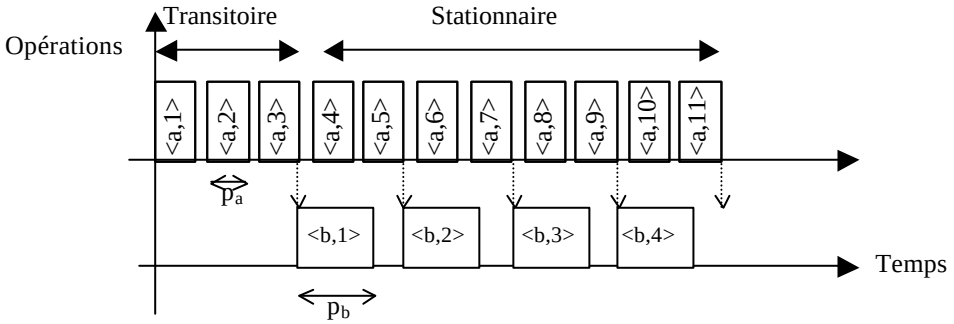


Figure I-6 Diagramme de Gantt associé à la Figure I-5 dans le cas (a_{ab}, b_{ab}, g_{ab} ,w_{ab}) = (2, 1, 1 ,0).

Propriété I-2

Les définitions précédentes entraînent que les ($\gamma_{ab} + \omega_{ab}$ -1) occurrences de l'opération "b" ne sont pas contraintes et peuvent donc démarrer librement.

L'ensemble T des nœuds associés aux opérations cycliques et l'ensemble C des contraintes de précédences linéaires entre ces nœuds forment un graphe linéaire G (T,C). Ces graphes de précédence à contraintes linéaires, notés GPCL dans la suite du document, sont utilisés au chapitre III.

Les résultats relatifs à ces graphes linéaires, formulés par (Munier, 1996) sont rappelés ci-dessous :

Définition I-13 Poids d'un chemin

Le poids d'un arc e (a,b) entre les nœuds "a" et "b" est défini par : $\gamma_{ab} / \alpha_{ab}$

Le poids d'un chemin u de G est P(u), tel que :

$$P(u) = \prod_{e_{ab} \in E(u)} \frac{g_{ab}}{a_{ab}}$$

avec E(u) : la liste des arcs e (a,b) entre les nœuds "a" et "b" du chemin u

Définition I-14 Hauteur d'un circuit

La hauteur d'un arc e (a,b) entre les nœuds "a" et "b" est défini par $\omega_{ab} - \beta_{ab}$

La hauteur d'un circuit u de G est H(u), tel que :

$$H(u) = \sum_{e_{ab} \in E(u)} v_{ab} - b_{ab}$$

avec E(u) : la liste des arcs e (a,b) entre les nœuds "a" et "b" du circuit u

Définition I-15 Expansion d'un graphe linéaire

Un graphe linéaire G (T,C) peut être transformé par une opération appelée "expansion" en un graphe uniforme équivalent si et seulement si :

$\forall$ e (a,b) $\in$ C, $n_a / \alpha_{ab} = n_b / \gamma_{ab} \in \mathbb{N}^*$

Dans ce cas, toute contrainte linéaire entre "a" et "b" peut être transformée en un ensemble de contraintes uniformes entre les n_a et n_b représentants respectifs des opérations "a" et "b". Considérons par exemple la contrainte linéaire e (a,b) =(2, 0, 3 ,1) entre les deux opérations a et b de la Figure I-7. L'expansion consiste à remplacer l'opération b par les trois opérations cycliques b1, b2 et b3. La $n^{ième}$ occurrence de b1, b2 et b3 remplace respectivement les occurrences 3(n-1)+1, 3(n-1) et 3n de l'opération "b". De même, l'opération a est remplacée par deux opérations a1 et a2. La contrainte e(a2,b1)=(0,1) exprime que l'occurrence (n+1) de l'opération b1 démarre au plus tôt après l'occurrence (n) de a2. Les contraintes entre b1, b2 d'une part et b2, b3 d'autre part sont des contraintes de précédence simple entre les occurrences identiques de ces opérations. La contrainte entre b3 et b1 contrôle l'exécution au plus tôt de l'occurrence (n+1) de b1 après l'occurrence (n) de b3. L'ensemble du processus d'expansion est mentionné sur le graphe de la Figure I-7.

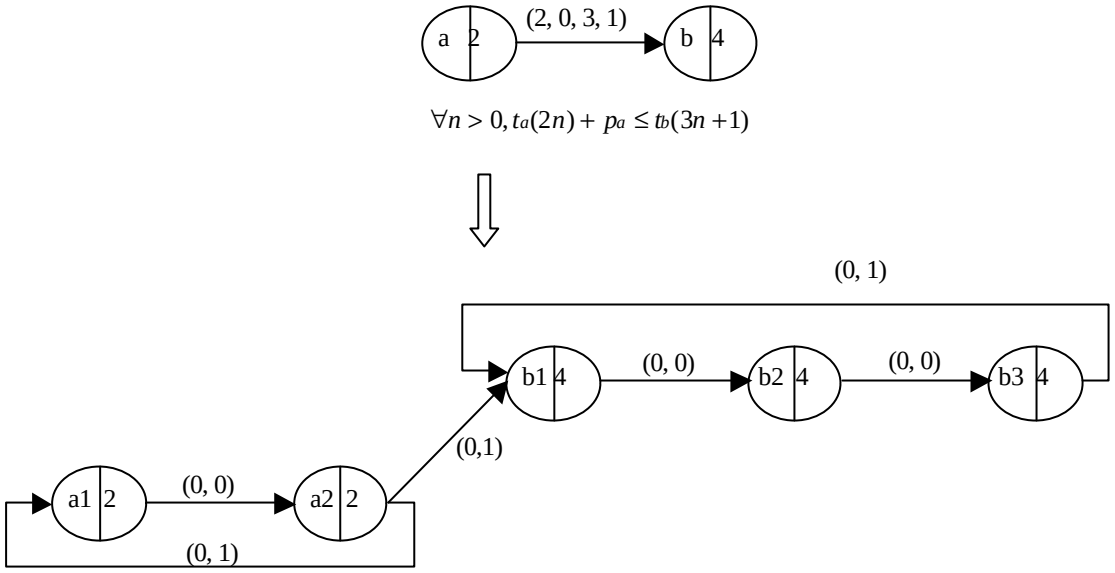


Figure I-7 Expansion d'une contrainte linéaire

Il n'est pas toujours possible de trouver un ensemble de solutions $\{n_i, i \in T\}$, pour toutes les contraintes du graphe. Les deux propriétés ci-dessous permettent néanmoins de se restreindre à des cas d'études favorables.

Propriété I-3 Cohérence d'un graphe linéaire : Condition Nécessaire

Un graphe linéaire G est dit cohérent i.e. il existe un ordonnancement infini respectant les contraintes de précédence, à condition que le poids de chaque circuit c de G, P(c), soit supérieur ou égal à 1.

Propriété I-4 " Expansibilité" des graphes unitaires

Les graphes unitaires i.e. graphes linéaires fortement connectés tels que le poids de chaque circuit vaut 1 sont "expansibles" (Hanen, 1995). De plus, le graphe linéaire initial est cohérent si et seulement si tout circuit du graphe expansé est de hauteur strictement positive (Munier, 1996).

Il est à noter que la taille du graphe expansé peut être exponentielle et que l'expansion n'est pas calculable en temps polynomial.

I.3.3 Utilisation des réseaux de Petri

Les réseaux de Petri sont utilisés dans de nombreux domaines pour spécifier et analyser le fonctionnement des systèmes complexes. Un aperçu de leur utilisation pour la modélisation et la résolution de problèmes d'ordonnancement et notamment les problèmes cycliques est décrit ci-dessous.

Rappelons tout d'abord que la sous-classe des réseaux de Petri appelée graphes d'événements est formée

de places ayant exactement une transition d'entrée et une transition de sortie : il n' y a donc jamais de conflit dans le tirage des transitions. Chrétienne (1986) a démontré que le fonctionnement au plus tôt des graphes d'événements temporisés permet d'obtenir un régime permanent périodique et il a caractérisé la fréquence de ce régime périodique. Il a également étudié le régime permanent de l'ordonnancement au plus tôt d'un problème modélisé par un graphe à contraintes uniformes fortement connecté, dans le cas où il n'y a pas contrainte de ressources (Chrétienne, 1985). Ce régime est k-périodique et la fréquence ainsi que le facteur de périodicité ont été déterminés. En outre, un algorithme polynomial de calcul de la fréquence a été fourni.

Le paradigme des réseaux de Petri peut également être exploité comme outil de résolution de problèmes. Les solutions d'un problème ainsi décrit correspondent aux séquences de tirs des transitions entre un état initial (marquage) et un état final accessible. Deux approches existent dans ce domaine. La première concerne les travaux basés sur l'équation fondamentale du réseaux de Petri et sa résolution par l'algèbre linéaire. La seconde moins répandue porte sur l'exploration du graphe d'accessibilité i.e. l'ensemble des états possibles du système, qui est un problème de nature combinatoire, à l'aide d'une technique d'abstraction logique (Yim, 2000), (Benasser, 2000).

La mise en œuvre de ce principe s'appuie sur les concepts de marquage et d'étape partiels :

- un marquage partiel est formé d'un couple : un marquage admettant des variables libres et des contraintes entre ces variables.
- une étape partielle est un vecteur de tir constitué de variables libres.

Le comportement d'un réseau de Petri muni de marquages partiels est une abstraction du formalisme usuel, car un tel marquage correspond à un ensemble de marquages classiques. La propriété fondamentale obtenue permet d'affirmer que l'énumération des valeurs satisfaisant les contraintes, fournit les solutions du problème réel sous forme de séquences d'étapes. L'intérêt essentiel de l'approche est d'éliminer les chemins inutiles ou conduisant à des blocages.

La méthode a été implantée avec différents *solveurs* de contraintes. Elle a permis de modéliser et de résoudre des problèmes d'ordonnancement et parmi eux des problèmes cycliques. Dans ce cas particulier, les marquages partiels initiaux et finaux ne sont pas totalement connus mais sont contraints d'être identiques.

Le paragraphe suivant propose une classification des problèmes cycliques.

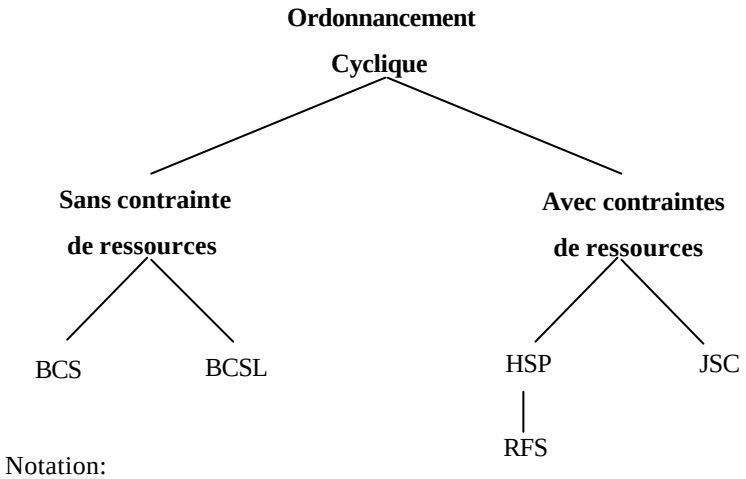
I.4 Taxonomie des problèmes cycliques

Les travaux de recherche relatifs aux problèmes d'ordonnancement cyclique sont fortement dépendants de leur domaine d'application d'origine. Ceci rend peu aisé la description de l'état de l'art dans ce domaine. Nous présentons dans ce paragraphe certains travaux dans ce champ de recherche. Cette présentation ne prétend pas être exhaustive. Un état de l'art complet du domaine a été proposé par les auteurs suivants (Serafini, 1989) (Hanen , 1995 ; Crama, 1997 ; Gentina, 2001). Ces travaux sont regroupés sous la forme d'une taxonomie en Figure I-8. Cette taxonomie comporte, deux branches principales. Il convient de noter que la dénomination anglaise des problèmes du domaine a été conservée lorsque leur équivalent français n'est pas usité dans la communauté de l'ordonnancement.

La partie gauche regroupe les problèmes cycliques sans contrainte de ressources :

- Le problème de référence est le *Basic Cyclic Scheduling Problem* (BCS). Il est constitué d'opérations génériques qui possèdent une infinité d'occurrences au cours du temps. Ces opérations sont soumises à des contraintes uniformes : chaque contrainte uniforme entre deux opérations génériques induit un ensemble de contraintes de précédence de sorte que la différence entre leur numéro d'occurrence est constante. L'objectif vise à trouver un ordonnancement qui minimise le plus grand temps de cycle parmi toutes les tâches et maximise donc le débit des opérations. Pour ce problème sans contraintes de ressources, il a été prouvé que les solutions périodiques sont dominantes au sens de la Définition I-1. Ce problème possède une solution exacte basée sur un algorithme de plus long chemin dont la complexité est $0(n^3 \log n)$, n étant le nombre d'opérations (Hanen, 1995),

- Le problème présenté ci-dessus est un cas particulier du *Basic Cyclic Scheduling Problem with Linear Precedence Constraints* BCSL dans lequel les contraintes de précédence sont linéaires. Ce problème à contraintes linéaires peut être transformé, lorsque la Propriété I-4 est respectée, sous la forme d'un BCS. Cette transformation est basée sur l'expansion du graphe à contraintes linéaires en un graphe à contraintes uniformes. Il est à noter que la taille du graphe résultant peut évoluer exponentiellement en fonction de la taille du graphe linéaire. La complexité du BCSL demeure ouverte.



Notation:

BCS : *Basic Cyclic Scheduling*

BCSL: *Basic Cyclic Scheduling with Linear Precedence Constraints*

HSP: *Hoist Scheduling Problem*

RFS: *Robotic Flow Shop*

JSC: *Job Shop* Cyclique

Figure I-8 Une taxonomie de l'ordonnancement cyclique

La partie droite de cette taxonomie contient divers problèmes dans lesquels les ressources sont en nombre limité. Les deux premiers problèmes sont proches du problème classique de flow shop mais possèdent des caractéristiques propres :

- Le *Hoist Scheduling Problem* (HSP) est rencontré dans le domaine des industries electro-métallurgiques : un produit doit passer par une succession d'immersion dans un ensemble de réservoirs (Baptiste, 2001). Ces réservoirs représentent une ressource de type machine. Les transports de pièces vers les différents réservoirs sont assurés par des robots (crochets). La première particularité de ce problème réside dans le fait que ces " crochets " partagent une ressource commune, le rail de transport, sur lequel il faut éviter les collisions. La seconde particularité de ce problème est que les temps de traitements des pièces sont remplacés par des bornes : bornes supérieures et inférieures des temps de trempage dans les réservoirs. Une infinité de pièces doit être traitée et l'objectif consiste à minimiser le temps de cycle du processus. Ce problème est NP-difficile dans le cas d'un robot unique et d'un seul job ou type de pièce.

- Le *Robotic Flow Shop* (RFS) est issu des cellules de production automatisées où plusieurs machines sont servies par un seul robot qui réalise les opérations de transport des pièces entre les machines depuis un point d'entrée jusqu'à un point de sortie (Crama, 2000). Le RFS peut être vu comme un cas particulier du HSP dans lequel il y a un seul robot et tous les jobs ont le même cheminement sur les machines. Un objectif classique consiste à trouver une séquence de déplacements du robot de manière à maximiser le débit pour un nombre de pièces entrantes infini. Ce problème est dans le cas général NP-difficile. Cependant une solution polynomiale existe $-0(m^3)$, m étant le nombre de machines, pour la recherche d'ordonnancement optimal à cycle unique dans le cas où toutes les pièces sont identiques (job unique).

- Le *Job Shop* Cyclique ou répétitif (JSC) est constitué d'un ensemble de jobs qui se répètent. Dans le cas dégénéré d'un seul job, une solution exacte a été fournie par Roundy (1992). Le problème traité consiste à minimiser le temps de cycle et le temps d'écoulement du job par une approche multicritère qui vise à obtenir un compromis entre ces deux critères. Draper (1999) a abordé le job shop périodique, au sens de la Définition I-6, en utilisant une approche basée sur la programmation par contraintes.

Les problèmes rencontrés en Production Flexible Manufacturière (PFM), aussi appelée *Flexible Manufacturing Systems*, peuvent avantageusement être modélisés sous la forme d'un job shop cyclique afin de réduire leur complexité. Ces problèmes de PFM sont caractérisés d'une part, par une certaine souplesse relative aux gammes de production et concernent d'autre part, des productions de petites et moyennes séries (Gentina, 2001). Ce type de production est organisé autour d'un système de transport pouvant véhiculer plusieurs types de pièces sur un porte-pièce dédié aussi appelé "palette", reliant des machines capables de réaliser plusieurs types d'opérations. La notion d'encours, caractéristique des systèmes de production, qui est assimilable sous certaines hypothèses simplificatrices aux nombres de palettes dans le système, revêt une importance particulière dans ce type de problématique. Il s'agit d'un problème complexe composé lui-même de plusieurs problèmes NP-difficile. Hillion (1989) a proposé une méthode de résolution approchée permettant de minimiser les encours tout en assurant un taux de production maximal.

La taxonomie présentée ci-dessus constitue une ébauche du domaine cyclique qui est incomplète. De nombreuses variantes existent pour les problèmes décrits telles que par exemple l'introduction de délais pour la réalisation des opérations ou le cas des problèmes à machines identiques. D'autres critères de classification tels que le type des contraintes – uniforme ou linéaire, le type d'ordonnancement recherché – périodique ou cyclique, peuvent être intégrés dans cette classification. Les problèmes d'ordonnancement cycliques sont donc majoritairement, soit des problèmes d'optimisation de type "NP-difficile", soit des problèmes dont la complexité

est inconnue ou "ouverte".

I.5 Conclusion

Les principales définitions ainsi que les problématiques de l'ordonnancement cyclique ont été introduites dans ce premier chapitre. Ces problèmes cycliques de même que les problèmes d'ordonnancement d'atelier sont dans le cas général des problèmes de type NP-difficile ou encore de complexité ouverte. De nombreux ouvrages tels que Carlier (1988) ou Dhaenens (2002) par exemple, abordent l'étude de la complexité et notamment de celle des problèmes d'ordonnancements. Néanmoins, d'un point de vue opérationnel ce type de complexité justifie l'utilisation de méthodes de résolution approchées pour traiter des instances de problème de taille importante.

Les méthodes approchées ont été peu développées pour la résolution de problèmes cycliques à l'exception des travaux de Chrétienne (1996; 1999) sur la redéfinition et l'exploitation des algorithmes de liste dans le cas particulier des problèmes à machines identiques, sans contraintes de ressources, modélisés par un graphe de précédence uniforme fortement connecté. Un algorithme de liste avec garantie de performance par rapport à un temps de cycle minimal a été proposé. Plus généralement, une garantie de performance pour tout algorithme de liste régulier a également été donnée pour ce problème cyclique.

C'est pourquoi, le chapitre suivant est dédié à une classe particulière de méthodes approchées de type méta-heuristique : les algorithmes évolutionnistes utilisés pour résoudre les problèmes cycliques et flexibles étudiés dans ce mémoire.

II Algorithmes évolutionnistes

Le recours à des méthodes de résolution approchées pour résoudre les problèmes d'ordonnancement, notamment les problèmes cycliques, a été justifié dans le chapitre précédent. Notre objectif vise donc à exploiter le potentiel des approches dites méta-heuristiques pour résoudre ces problèmes. En effet, cette catégorie de méthodes de résolution et en particulier les algorithmes évolutionnistes n'ont été que très peu utilisés à notre connaissance pour résoudre des problématiques cycliques.

De même que pour le chapitre précédent, l'objet n'est pas ici de réaliser une présentation exhaustive du domaine des algorithmes évolutionnistes qui est en plein foisonnement. Un nombre croissant de livres apparaît chaque année sur ce sujet. En outre, le groupe de travail "Evolution Artificielle" de l'Association Française d'Intelligence Artificielle (AFIA) est actif dans ce domaine par l'intermédiaire des JET (Journées Evolutionnaires Trimestrielles)[4].

Dans ce chapitre, notre démarche vise simplement à situer le cadre général dans lequel s'inscrivent travaux décrits dans les trois chapitres suivants. Il comporte donc un aperçu général sur les algorithmes évolutionnistes puis se focalise sur les méthodes évolutionnistes multicritères. Enfin il aborde l'utilisation des méthodes évolutionnistes dans le domaine de l'ordonnancement.

II.1 Principes généraux

Le paradigme des algorithmes évolutionnistes (De Jong, 2000 ; Bäck, 2000a) fondé dans les années 1960, consiste à s'inspirer des mécanismes de l'évolution naturelle et à utiliser le concept de populations d'individus ou solutions pour résoudre des problèmes du monde réel.

La mise en œuvre relativement aisée de ces algorithmes ainsi que les nombreux succès qu'elles ont obtenus ont contribué à leur développement spectaculaire ces vingt dernières années (Bäck, 2000b ; 2000c). Ce développement porte à la fois sur leur diffusion dans de très nombreux domaines d'application ainsi que sur l'étude et l'exploration des mécanismes d'évolution eux-mêmes. De nombreuses applications de ces méthodes concernent le domaine des systèmes de production (Pierreval, 2003) et notamment le domaine de l'ordonnancement (Portmann, 2001).

[4] http://www.afia.polytechnique.fr/

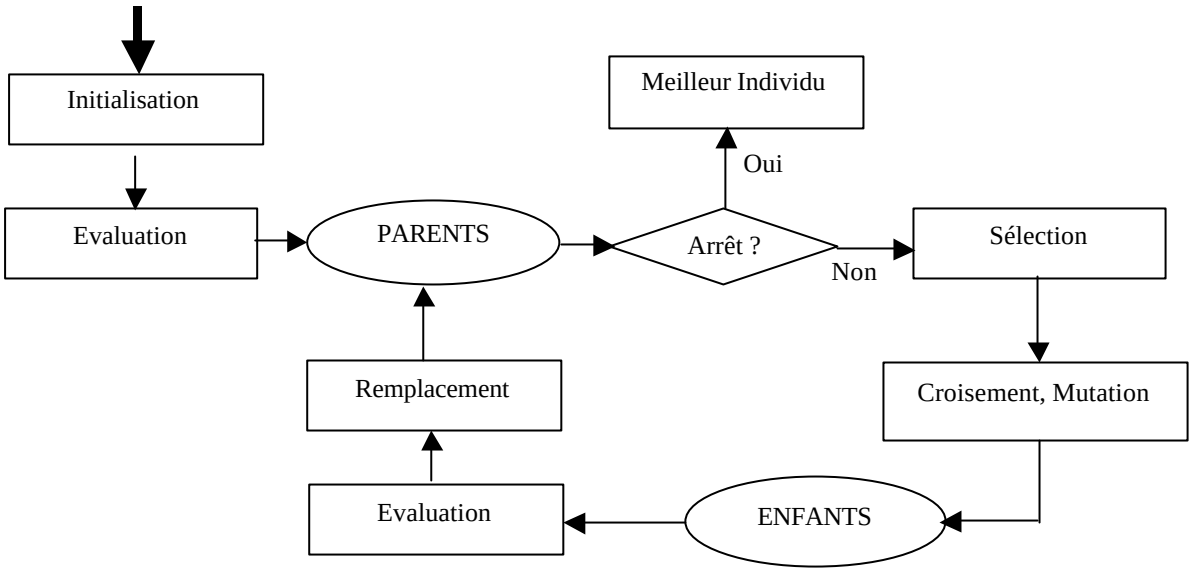


Figure II-1 Algorithmes évolutionnistes : (Schoenauer, 2001)

Le cycle principal d'un algorithme évolutionniste présenté en Figure II-1consiste à sélectionner des individus *parents* à partir d'une population de départ, qui en se reproduisant par des opérations de "croisement" et "mutation" engendrent de nouveaux individus. Une nouvelle population est ainsi constituée par "remplacement" de l'ensemble ou d'une partie de la population de départ, par ces nouveaux individus. La répétition de ce cycle de base engendre donc une succession de générations de solutions, jusqu'à satisfaction d'un critère donné de fin de cycle.

La "sélection" est basée sur une évaluation de chaque individu qui détermine le degré d'adaptation de cet individu dans son environnement ; le résultat de cette évaluation est appelé "*fitness* ou force" de l'individu.

Par analogie avec la biologie, la terminologie courante désigne la représentation d'un individu par le terme " génotype " et désigne le comportement de celui-ci dans un environnement donné par le terme " phénotype ".

Le modèle général des algorithmes évolutionnistes doit être adapté au problème traité. Il est nécessaire de définir en particulier la représentation des solutions ou génotypes, la fonction d'évaluation des solutions ainsi que les opérateurs génétiques. Les opérations de sélection et de remplacement sont, quant à elles, indépendantes des choix de représentation.

Le concept d'algorithme évolutionniste se décline en plusieurs modèles recensés ci-dessous et décrits en Figure II-2. ne s'agit pas de décrire en profondeur chacun de ces modèles mais simplement de préciser leurs principales caractéristiques. En outre, ce recensement n'est pas exhaustif ; d'autres techniques sont aux frontières du domaine telles que les " Classifiers "(Holland, 1986) par exemple ou telles que le modèle récent des colonies de fourmis (Dorigo ,1996).

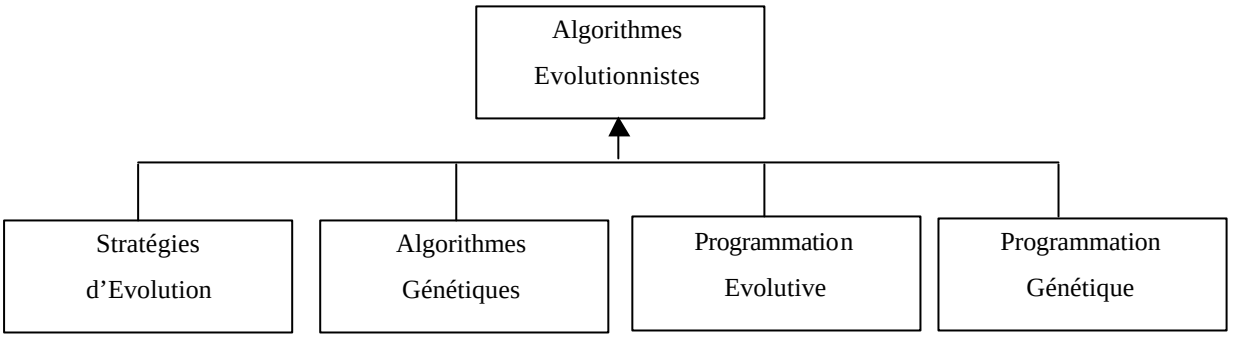


Figure II-2 Taxonomie des algorithmes évolutionnistes

- Programmation Evolutive ou *Evolutionary Programming* (Fogel, 1966). Ce modèle évolutionniste accentue l'utilisation de la mutation et n'utilise pas dans sa version originale la recombinaison des individus par croisement. Développé à l'origine pour l'évolution d'automates à état-fini, ce modèle est souvent appliqué à la résolution de problèmes d'optimisation à variables réelles. Dans ce cas, il utilise une mutation qui consiste à ajouter une perturbation Gaussienne à chaque composante du vecteur à variables réelles constituant l'individu. Cette perturbation est basée sur la performance de l'individu : l'idée consiste à faire subir des mutations importantes aux mauvais individus et inversement des mutation faibles aux bons individus. L'opérateur de sélection est de type probabiliste : il s'agit de la méthode du tournoi basée sur une compétition entre individus choisis aléatoirement. Il est à noter que la représentation des individus n'est pas contrainte à une forme spécifique de génome telle que dans une représentation linéaire de type chaîne binaire par exemple. En effet, l'opérateur de recombinaison qui induit de fortes contraintes de représentation, est ici absent.

- Stratégies d'Evolution ou *Evolution Strategies* (Schwefel, 1995[5]). Elles ont été développées pour résoudre des problèmes d'optimisation industriels à variables réelles dans lesquels il n'existe pas de fonction d'objectif analytique. Rappelons que dans le contexte général de l'optimisation, chaque individu représente un point dans l'espace de recherche des solutions potentielles au problème donné. Ce modèle des Stratégies d'Evolution utilise le principe de mutation sur les réels du modèle de la Programmation Evolutive. Cependant, ce principe a été affiné de sorte que la fonction de perturbation Gaussienne est contrôlée par l'ensemble de la population courante. Une mise en œuvre de ce principe est la règle des 1/5 de Rechenbergh (1973) : sur un intervalle de temps passé de profondeur donné, lorsque la proportion de descendants de bonne qualité i.e. nombre de mutation à succès, dépasse 20 % de la population totale, l'intensité de cette mutation est augmentée. Cette augmentation se traduit par un accroissement de la variance caractérisant la perturbation gaussienne. Elle est diminuée dans le cas opposé. Une interprétation possible de cette règle est la suivante : si la proportion de mutation réussie est élevée, l'espace de recherche exploré est restreint autour d'un optimum local, il faut donc diversifier la population en augmentant le taux de mutation. Ces approches utilisent un opérateur de sélection de type déterministe: les solutions dont le *fitness* est mauvais sont éliminées de la population. En outre, dans le modèle originel, les populations des parents et de leurs descendants sont généralement de taille différente.

[5] La référence d'origine (1965) est en allemand

Il convient de noter que le principe de mutation présenté ci-dessus est une caractéristique importante qui préfigure la mutation auto-adaptative et l'extension du processus d'évolution aux paramètres de contrôle de l'algorithme génétique. En effet, la variance de la mutation peut être ajustée au cours du temps par le processus d'évolution. Les méthodes évolutionnistes auto-adaptatives visent précisément à automatiser le réglage des paramètres de l'algorithme évolutionniste et ainsi à remédier aux méthodes empiriques du type " essais erreurs " qui sont employées dans la pratique.

- Algorithmes Génétiques ou *Genetic Algorithms* (Holland, 1962 ; Goldberg, 1989; Michalewicz, 1996). Contrairement au modèle de la Programmation Evolutive, les algorithmes génétiques favorisent l'utilisation du croisement comme principal opérateur de recherche. Il utilise cependant la mutation avec un faible pourcentage de probabilité. Une méthode de sélection de type probabiliste est utilisée. La représentation des individus - i.e. génotype, qui est à l'origine de type binaire, a été par la suite étendue à de nombreuses autres formes de représentation. L'opérateur de sélection est comme dans le modèle précédent de type probabiliste avec un tirage des individus basé sur la méthode de la roulette dans laquelle la probabilité de sélection est proportionnelle au *fitness* de l'individu.

- Programmation Génétique *ou Genetic Programming* (Koza, 1992) est une extension du modèle d'apprentissage des algorithmes génétiques à l'espace des programmes. Les individus formant une population sont donc des programmes candidats à la résolution d'un problème. Ces programmes sont exprimés sous la forme d'arbres sur lesquels les opérateurs génétiques produisent des transformations en vue d'obtenir un programme qui satisfait la résolution du problème choisi.

Ces différents modèles évolutionnistes dont les origines diffèrent et qui possèdent chacun leur spécificité tendent aujourd'hui à converger vers le modèle unique des algorithmes évolutionnistes intégrant les particularités de chacun des modèles.

Par ailleurs, deux types d'extensions fondamentales ont été développés à partir du concept de base de ces algorithmes:

- la première extension, concerne la résolution de problèmes d'optimisation sous contraintes (Schoenauer, 1993). Dans ce domaine, trois approches de résolution peuvent être distinguées: la première vise à pénaliser les individus ne satisfaisant pas les contraintes, appelés " individus infaisables ", en diminuant artificiellement leur *fitness*; la seconde approche possible tend à modifier par réparation les solutions infaisables pour satisfaire les contraintes; la dernière approche vise quant à elle, à engendrer uniquement des solutions faisables par une représentation et des opérateurs génétiques adaptés. Dans cette dernière catégorie, il convient de mentionner l'utilisation de "décodeurs" qui correspondent à des individus particuliers donnant des instructions sur la manière de générer des solutions acceptables.

- la seconde extension majeure des algorithmes évolutionnistes concerne la résolution de problèmes d'optimisation multicritères . Nous avons choisi de développer plus en détail ces approches dans le paragraphe suivant, car ce type d'approche évolutionniste a été exploité pour la résolution d'un problème d'ordonnancement dans la suite de notre travail.

II.2 Optimisation multicritère évolutionniste

Le vaste domaine de la décision multicritère est un domaine où se rencontrent de multiples disciplines :

mathématiques appliquées, informatique, automatique, sciences cognitives, et trouve des applications dans de nombreux domaines industriels, économiques et sociaux. Plus précisément, ce domaine de la décision basée sur plusieurs critères ou objectifs[6] peut être décomposé en deux sous-domaines (Vincke, 1989). Le premier est la prise de décision multicritère (MultiCriteria Decision Making i.e. MCDM) qui utilise des méthodes de résolution visant à agréger les différents critères sous la forme d'une fonction d'utilité ; cette fonction est ensuite optimisée. Une méthode couramment employée dans ce domaine est la théorie de l'utilité multi-attribut (Multi Attribute Utility Theory i.e.MAUT (Keeney, 1993). Le second domaine de la décision est l'aide à la décision multicritère (MultiCriteria Decision Aid i.e. MCDA) dont l'objectif est centré sur la modélisation d'un problème multicritère en tenant compte de l'expérience d'un " Décideur ". Les approches de type MCDA et leurs fondements sont dus essentiellement à l'école Française et utilisent notamment les notions de " surclassement " et de " préférence " (Roy, 1985).

Les méthodes d'optimisation multicritères visent à trouver les meilleures solutions parmi un ensemble de solutions possibles tendant à satisfaire au mieux un ensemble de critères à optimiser. Un très grand nombre de méthodes multicritères peut être recensé. Ces méthodes d'optimisation multicritères sont appliquées majoritairement dans le domaine de la prise de décision multicritère (MCDM) et notamment dans les problèmes d'ordonnancement (T'Kindt, 2002). Elles le sont également dans le domaine de l'aide à la décision multicritère (MCDA) et concernent plus particulièrement les problématiques dites de " choix " qui consistent à rechercher les meilleurs décisions pour un problème donné.

Le problème multicritère traité au chapitre V appartient quant à lui à la catégorie des problèmes de prise de décision multicritère, dans le cas particulier ou la fonction d'utilité n'est pas connue. L'objectif consiste alors à rechercher un ensemble de solutions non dominées.

Cette seconde partie du chapitre II est consacrée à la présentation des méthodes d'optimisation multicritères et de leur classification. Les méthodes évolutionnistes multicritères sont ensuite introduites et en dernier lieu l'approche NSGA -II utilisée au chapitre V, est décrite.

II.2.1 Définitions et optimalité Pareto

Définition II-1

Un problème multicritère constitué de n variables, m contraintes et k critères vise à minimiser $F(\mathbf{x})=(f_1(\mathbf{x}),..,f_k(\mathbf{x}))$, sous la contrainte $g(\mathbf{x})<=0$ pour i=1,…,m ; $\mathbf{x}$ étant un vecteur solution ($\mathbf{x}=x_1,..,x_n$) d'un univers Ω de dimension n, encore appelé *espace de décision*. L'espace défini par le tuple ($f_1, f_2,..,f_k$), noté Λ, est appelé *espace des critères*. La fonction d'évaluation $F : \Omega \mapsto \Lambda$, d'un problème multicritère, fait donc correspondre un vecteur $\mathbf{y}$ ($\mathbf{y}=y_1,..,y_k$) de dimension k à tout vecteur $\mathbf{x}$ de dimension n de l'espace Ω.

Il convient tout d'abord de remarquer, que tout problème de maximisation peut se transformer en problème de minimisation. Il est également important de souligner que ces critères sont généralement antagonistes. Il s'agit donc de chercher des solutions représentant un compromis possible entre les critères. Dans ce domaine, le concept d'optimalité Pareto introduit par l'économiste V. Pareto au XIX^{ème} siècle est

[6]Les termes « critère » et « objectif », quoique non identiques, ont une sémantique proche (Chankong, 1985) ; le terme « critère » est employé préférentiellement dans ce manuscrit comme c'est généralement le cas en ordonnancement.

fréquemment utilisé (Pareto, 1896). La notion de dominance Pareto et les concepts afférents sont donc présentés ci-après :

Définition II-2

Un vecteur $\mathbf{u} = (u_1,..,u_k)$ domine un vecteur $\mathbf{v} = (v_1,..,v_k)$ si et seulement si: $\forall i \in \{1,..,k\}$, $f_i(\mathbf{u}) \succeq f_i(\mathbf{v})$ et $\exists j \in \{1,..,k\} \mid f_j(\mathbf{u}) < f_j(\mathbf{v})$. La relation de dominance est souvent notée $\succ$.

Cette relation traduit donc que le vecteur **u** domine le vecteur **v**, au sens de Pareto si et seulement si il existe une dimension de l'espace dans laquelle **u** est strictement meilleur que **v** et si **u** n'est pas moins bon que **v** dans toutes les autres dimensions de cet espace.

Définition II-3

Une solution $\mathbf{x}^*$ est Pareto optimal si et seulement si il n'existe pas une solution **x** telle que F(**x**) domine F($\mathbf{x}^*$) : F(**x**) $\succ$ F($\mathbf{x}^*$). Ces solutions sont aussi appelées solutions admissibles, efficaces, non-dominées ou non-inférieures.

Définition II-4

Pour un problème donné défini par F(x), l'ensemble Pareto optimal ($\wp^*$) est défini par : $\wp^* = \{\mathbf{x} \in \Omega \mid \nexists \mathbf{x}' \in \Omega : F(\mathbf{x}') \succ F(\mathbf{x})\}$

Définition II-5

Pour un problème donné défini par F(x) et un ensemble Pareto optimal $\wp^*$, le front Pareto est défini par : $\wp \ddot{o} * = \{\mathbf{u} \in \Lambda \mid \mathbf{u} = F(\mathbf{x}) \text{ et } \mathbf{x} \in \wp^*\}$

Etant donné la taille de l'espace de recherche des problèmes traités, ce front est souvent difficile à atteindre, c'est pourquoi la plupart des algorithmes n'offre généralement qu'une approximation de celui-ci.

Notons que ces définitions confèrent à la relation de dominance, une structure de relation d'ordre partiel strict , car cette relation est transitive mais non réflexive et non antisymétrique.

II.2.2 Typologie des méthodes d'optimisation multicritères

Dans le cas général, la résolution de problèmes d'optimisation multicritères qui vise à établir un compromis entre plusieurs critères, met en œuvre des mécanismes d'optimisation ainsi que des mécanismes de prise de décision dans lesquels intervient un décideur. Le domaine de la prise de décision multicritère distingue à cet égard trois schémas possibles de combinaison de ces deux mécanismes complémentaires :

- les approches "a posteriori" dans lesquelles le décideur intervient en amont du processus d'optimisation, pour définir la fonction d'agrégation des différents critères

-les approches "a priori" font intervenir le décideur en aval du processus d'optimisation, en lui présentant des solutions Pareto optimales, voire des solutions non dominées sur lesquelles il exerce son choix final.

- les approches "interactives" combinent de manière cyclique et incrémentale les deux mécanismes mentionnés : le décideur définit des préférences qui sont prises en compte au cours du processus de résolution du problème.

Dans le domaine particulier de l'optimisation méta-heuristique multicritère, de nombreuses approches de résolution ont été développées. Leur présentation est ici restreinte aux approches évolutionnistes qui peuvent être classées en trois catégories (Talbi, 1999):

- la transformation vers un problème mono-critère consiste à combiner les divers critères en les pondérant. Ces méthodes sont de type "a priori" . Dans cette catégorie, citons les méthodes d'agrégation, E-contrainte et la programmation par but.

- les approches non-Pareto utilisent des opérateurs qui traitent séparément les différents critères. Citons les algorithmes génétiques à sélection parallèle, à sélection lexicographique ou basée sur une reproduction multi-sexuelle. Il s'agit principalement d'approches de type "a posteriori ".

- les approches Pareto utilisent directement la notion d'optimalité Pareto. Elles visent à atteindre deux buts : d'une part converger vers la frontière Pareto optimale et d'autre part obtenir des solutions diversifiées –i.e. réparties sur toute cette frontière. Ces approches appartiennent également aux approches de type "a posteriori".

La plupart des approches évolutionnistes multicritères courantes utilisent donc les approches Pareto "a posteriori " (Van Veldhuizen, 2000; Coello, 1999). Ces approches utilisées pour le problème du job shop flexible qui est présenté au chapitre V, sont détaillées dans le paragraphe suivant

II.2.3 Algorithmes évolutionnistes multicritères

L'adaptation des algorithmes évolutionnistes pour la résolution de problème multicritère (Deb, 2001) porte d'une part, sur l'étape d'évaluation des individus de manière à prendre en compte l'ordre partiel défini par la relation de dominance et d'autre part, sur l'étape de sélection. L'objectif de ces méthodes est de favoriser la recherche de solutions non dominées tout en conservant une diversité suffisante. Les principales spécificités des algorithmes développés dans ce domaine sont présentées ci-dessous puis un algorithme particulier mis en œuvre au chapitre V est approfondi.

II.2.3.1 Principales caractéristiques des algorithmes évolutionnistes multicritères

II.2.3.1.a Classement ou *Ranking*

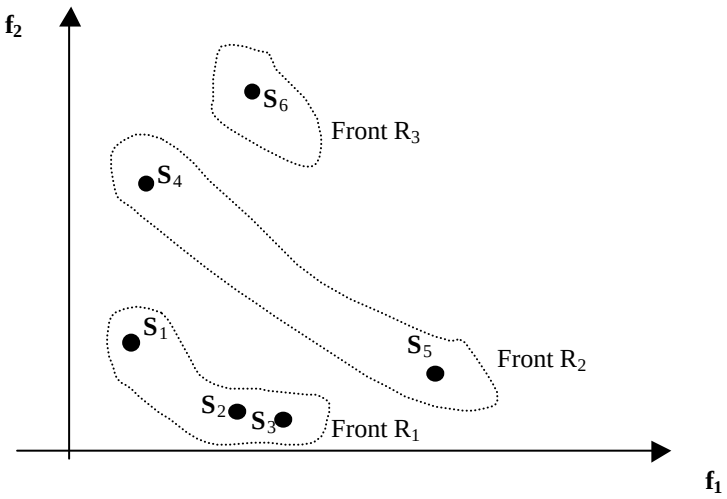
L'utilisation d'un algorithme évolutionniste dans un contexte multicritère nécessite de pouvoir associer une valeur scalaire unique, le *fitness*, au vecteur des critères F(**x**). L'étape de *ranking* consiste à classer les individus en leur donnant un rang. Le *fitness* est alors attribué en se basant sur le rang de chaque individu. Deux approches particulières du ranking sont mentionnées ci-dessous :

- NSGA *(NonDominated Sorting Genetic Algorithm)* (Srinivas, 1995) cette méthode attribue un rang égal à un pour tous les individus non dominés de la population courante qui forment le front Pareto R_1 ($R_1 = \wp \ominus *$). La méthode procède récursivement en attribuant le rang k, noté r_k, aux individus dominés uniquement par les

individus appartenant à l'ensemble $R_1 \subset R_2 \subset .. \subset R_{k-1}$.-i.e. le rang r_k est attribué aux individus non dominés de la population initiale de laquelle ont été retirés les individus de rang 1 à k-1. Ce processus récursif s'arrête lorsqu'un rang unique a été associé à tous les individus de la population courante.

- MOGA *(Multi-Objective Genetic Algorithm)* (Fonseca, 1995): le rang d'un individu est proportionnel au nombre d'individus le dominant ; plus précisément, le rang de l'individu i, r_i est défini par : $r_i=1+Dom(i)$ avec Dom(i), le nombre d'individus dominant l'individu i. Un individu non dominé de la population initiale possède donc le rang numéro un.

La Figure II-3 illustre l'attribution des rangs selon ces deux approches.



Solution	Rang
S_1	1
S_2	1
S_3	1
S_4	2
S_5	3
S_6	4

a. NSGA **b. MOGA**

Figure II-3 *Ranking* a) dans NSGA et b) dans MOGA

Préalablement à l'utilisation d'une méthode de sélection de type proportionnel, il est nécessaire de différencier le *fitness* des solutions de même rang, ce qui est l'un des objets du *niching*.

II.2.3.1.b Techniques de formation des niches ou *Niching*

Les méthodes de *niching* (Mahfoud, 2000) étendent le concept des algorithmes évolutionnistes dans les domaines qui requièrent de localiser et maintenir de multiples optima tels que par exemple l'optimisation de fonctions multimodales. Dans les problèmes d'optimisation multicritères, le *niching* a pour but d'obtenir des solutions diversifiées réparties de manière la plus uniforme possible le long du front Pareto. Pour cela le concept des niches écologiques qui définit une région de l'espace autour d'une solution considérée, a été utilisé. Le principe consiste donc à dégrader le *fitness* des solutions situées dans une région de l'espace de recherche contenant une grande concentration de solutions – i.e. niches.

Dans NSGA (Srinivas, 1995) une fonction de partage ou *sharing* basée sur une mesure de distance entre individus vient donc altérer la fonction de *fitness* originale. La nouvelle valeur du *fitness* f' d'un individu i, vaut

l'ancien *fitness* divisé par son compteur de niche m(i) : f'(i) = f(i) / m(i). Le compteur de niche est défini en utilisant la fonction de partage "Sh" entre l'individu i et l'ensemble des individus de la population P: $m(i)= \sum_{j \in P, j \neq i} Sh(d_{ij})$. La fonction de partage "Sh" est une fonction de la distance d_{ij} entre individus qui retourne la valeur un si ces individus sont identiques et zéro si ces individus dépassent un seuil de dissimilitude ; cette fonction représentée en Figure II-4 est définie comme suit : $Sh(d_{ij})=1-(d_{xy}/\sigma_{share})^{\alpha}$ si $d<\sigma_{share}$ et 0 sinon. Les paramètres σ_{share} et α sont respectivement la taille des niches et un paramètre de forme de la fonction de partage.

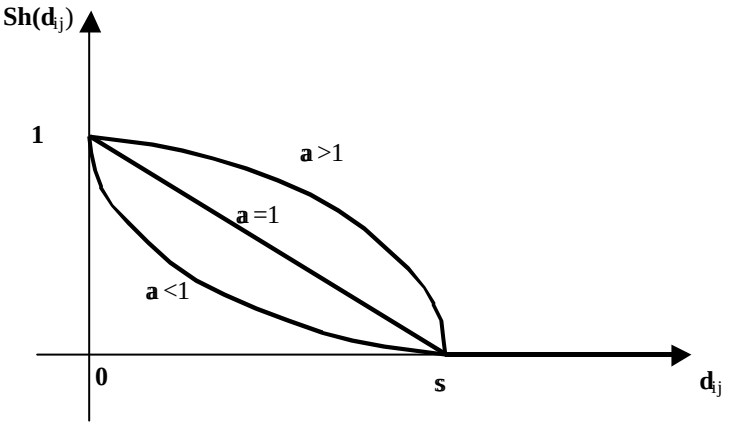


Figure II-4 Fonction de partage de NSGA

A titre d'exemple, l'effet du *niching* sur les solutions du premier front Pareto de la Figure II-3, consiste à diminuer le fitness des solutions S_2 et et S_3 sans modifier celui de S_1.

La distance entre individus peut être appréciée de différentes manières : soit en se référant au *fitness* des individus, il s'agit de *sharing* phénotypique , soit en se référant à leur génotype appelé *sharing* génotypique, soit encore par une combinaison de deux mesures de distance dans le cas du *sharing* combiné. D'autres approches du "*niching* " consistent à utiliser les techniques de *crowding* qui insèrent de nouveaux individus dans une population en remplaçant certains individus identiques. Comme précédemment, la similarité entre individus repose sur une mesure de distance génotypique, phénotypique ou combinée.

II.2.3.1.c Elitisme

D'un point de vue général le mécanisme d'élitisme dans les algorithmes évolutionnistes vise à lier la durée de vie des individus à leur performance : les individus dont le *fitness* est bon sont gardés pendant plusieurs générations. Dans le domaine de l'optimisation multicritère, la sélection élitiste consiste à maintenir une seconde population appelée archive, contenant les solutions non dominées trouvées au cours des différentes générations de l'algorithme évolutionniste. Les individus de cette population participe avec une certaine probabilité à l'étape de sélection et donc à la reproduction de nouveaux individus. En outre des techniques de regroupements ou *clustering* sont employées pour limiter la taille de cette archive. Cette technique a été mise en œuvre dans l'algorithme SPEA *Strength Pareto Evolutionary Algorithm SPEA* (Zitzler, 1999).

II.2.3.2 Etude de l'algorithme évolutionniste multicritère :NSGA-II

La sélection d'un algorithme multicritère évolutionniste pour résoudre le problème étudié au chapitre V, s'appuie sur trois critères. Le premier est l'utilisation de l'élitisme : en effet, une comparaison des principaux algorithmes du domaine a été effectuée par Zitler en se basant sur un ensemble de problèmes tests (Zitzler, 2000). Cette comparaison a notamment permis de mettre en évidence que l'élitisme est un facteur essentiel dans l'optimisation évolutionniste multicritère. Le second critère est l'utilisation d'un mécanisme de maintenance de la diversité tel que le *niching* ne nécessitant pas de paramétrage spécifique. A titre d'exemple, la mise en œuvre du mécanisme de *niching* dans l'algorithme NSGA est complexe car elle impose de choisir plusieurs paramètres parmi lesquels la taille des niches et la mesure de distance entre individus, qui ont une influence sur la qualité des résultats obtenus. Enfin, le dernier critère est la complexité des algorithmes mis en œuvre. A titre d'exemple, la complexité du classement des individus non dominés dans l'implémentation initiale de NSGA est $O(n^3)$ avec n, la taille de la population, ce qui est coûteux dans le cas de populations de grande taille.

L'algorithme mis en œuvre au chapitre V pour traiter le job shop flexible est l'algorithme NSGA-II proposé par Deb (2000) qui est une évolution de l'algorithme NSGA. Cet algorithme intègre d'une part un mécanisme d'élitisme et utilise d'autre part une technique de *niching* simplifiée basé sur un mécanisme de *crowding*. La présentation de cet algorithme comporte trois parties : la définition de la distance de *crowding*, la description de la méthode de sélection utilisant cette distance et enfin la présentation du cycle principal de cet algorithme

Distance de *crowding* : cette distance utilise la notion de cuboï de représenté en pointillé sur la Figure II-5, dans l'espace des critères à deux dimensions. Plus précisément, cette distance correspond au demi-périmètre du cuboï de défini par les deux plus proches voisins de la solution concernée. Les distances de *crowding*, notées d_i pour le point i, associées aux points de la Figure II-5 sont dans l'ordre croissant : d_B, d_C, d_D et d_A. En effet, le point B est associé au plus petit cuboï de: sa distance de *crowding* est faible car il est situé dans une zone de forte densité de l'espace de recherche. Les cuboï des points A et D s'étendent à l'infini et les distances de *crowding* correspondantes sont donc maximales.

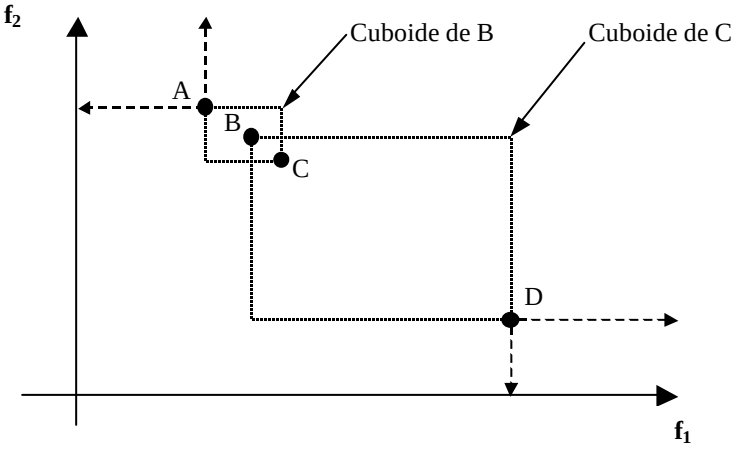


Figure II-5 Calcul de la distance de *crowding* (Deb, 2000)

Opérateur de sélection : il s'agit d'un opérateur par tournoi qui utilise la distance de *crowding* d_i présentée ci-dessus. Le principe de cet opérateur est le suivant : étant donné deux solutions i et j en compétition pour la sélection, le vainqueur est la solution i si l'une des deux conditions suivantes est remplie : soit le rang r_i est meilleur que celui de r_j i.e. $r_i < r_j$, soit r_i est égal à r_j, mais la distance de *crowding* d_i est meilleur que d_j i.e. $d_i > d_j$. Ainsi, pour résoudre le conflit de sélection en cas de solutions appartenant au même front Pareto, la solution gagnante est celle qui réside dans la région la moins "dense" de l'espace de recherche i.e. celle qui a la plus grande distance de *crowding*.

Etapes de NSGA-II : elles sont mentionnées dans l'Algorithme II-1. La stratégie élitiste consiste dans un premier temps à réunir les populations des parents P_t et des descendants D_t de la génération t dans le même ensemble U_t. La population suivante P_{t+1} est créée à partir des x premiers fronts complets de Ut de telle sorte que le nombre d'individus contenus dans ces x premiers fronts soit inférieur ou égal à n. Cette population P_{t+1} est alors complétée par les meilleurs individus du front de rang r_{x+1} au sens de la distance de *crowding*. La sélection par tournoi décrite précédemment ainsi que les opérateurs de recombinaison et de mutation sont alors utilisés pour créer la nouvelle population de descendants D_{t+1}.

Algorithme II-1 Cycle principal de NSGA-II

Algorithme II-1
Réunir dans U_t (taille 2n) les populations parents P_t (taille n) et descendants D_t (taille n): $U_t = P_t \cup D_t$
Classer les solutions de U_t par rang de solutions non dominées et identifier les fronts Pareto $R_1, R_2..R_m$
Créer une nouvelle population P_{t+1} (taille n) en incluant les solutions des x premiers front Pareto : $R_1, .. R_x$
Trier les solutions de R_{x+1} par distance de *crowding* croissante
Compléter si nécessaire P_{t+1} par ajout de solutions i de R_{x+1} | d_i est grand i.e. solutions dispersées
Créer la population D_{t+1} à partir de P_{t+1} en utilisant la sélection par tournoi avec *crowding*

NSGA-II dont les avantages ont été mentionnés précédemment présente deux types de limitations (Deb, 2000). Premièrement, la complexité de l'algorithme de classement des solutions non dominées bien qu'étant au plus du type $O(n^2)$ porte sur une population de taille 2n, alors que la plupart des algorithmes similaires utilisent une population de taille n. La seconde limitation est due à l'utilisation de la distance de *crowding* qui ne garantit pas la convergence de l'algorithme. En effet, il a été constaté que lorsque les solutions Pareto optimales présentent des groupements, certaines de ces solutions sont écartées de la nouvelle population P_t en raison de leur distance de *crowding* faible avant d'être réintégrées dans les générations suivantes ; ce qui conduit donc à une perte de convergence de l'algorithme.

II.3 Algorithmes évolutionnistes en ordonnancement

Les algorithmes génétiques ont été appliqués à la résolution de problèmes d'ordonnancement et notamment aux problèmes classiques de flow shop et job shop (Portmann, 2001; Lee 2000; Cheng, 1999). Ce paragraphe présente d'une part les principes généraux de représentation et de conception des opérateurs génétiques pour les problèmes d'ordonnancement ; il décrit d'autre part deux méthodes de représentation et deux opérateurs de recombinaison utilisés pour le job shop car ce problème possède des similarités fortes avec les problèmes traités au cours des chapitres suivants.

II.3.1 Généralités

Les problèmes d'ordonnancement sont dans le cas général des problèmes de la classe NP-difficile. Dans ce domaine des problèmes dits " combinatoires ", le théorème (Wolpert 1995) connu sous le nom de *no free lunch*, montre qu'il n'existe pas d'heuristique qui soit meilleure qu'une autre : pour toute heuristique, il existe au moins une instance du problème pour laquelle une autre méthode lui est supérieure. L'importance et la portée pratique de ce théorème suscitent des avis contradictoires (Reeves, 2003a). Néanmoins du point de vue pratique, pour atteindre de bonnes performances, il est le plus souvent nécessaire d'introduire dans les méthodes de résolution, une certaine connaissance ou information sur l'instance ou le problème à résoudre. Dans le contexte d'une résolution par les algorithmes génétiques ou plus généralement par les algorithmes évolutionnistes, cette nécessité a des conséquences importantes sur la représentation des solutions et les opérateurs génétiques:

- en premier lieu, la méthode de représentation des solutions elle-même est primordiale. Une caractéristique importante est le sous-ensemble de l'espace de recherche qui est potentiellement exploré à l'aide du codage. Si le codage ne couvre qu'une part restreinte de l'espace de recherche –i.e. phénotype, il n'y aucune garantie, dans le cas général, que ce sous-espace contienne les solutions optimales. Dans le cas d'un problème d'ordonnancement, il est judicieux de se restreindre à un sous-ensemble d'ordonnancements dominants si celui-ci est connu : c'est le cas, à titre d'exemple, du sous-ensemble des ordonnancements actifs pour les critères réguliers.
- l'opérateur de recombinaison ou croisement doit être conçu dans la mesure du possible de manière à conserver les bonnes caractéristiques des deux parents, afin de construire des descendants de bonne qualité (Portmann 2001). Le théorème des *schémas* (Goldberg, 1989) relatif à la convergence des algorithmes génétiques, dont la validité n'est que partielle et restreinte au cas des représentations binaires montre en effet qu'il vaut mieux situer à proximité les gènes dépendants de manière à éviter leur séparation par la recombinaison. La conception des opérateurs de croisement pour la résolution d'un problème d'ordonnancement tend donc à suivre cette recommandation.
- enfin, il s'avère utile d'introduire dans la population initiale qui est générée aléatoirement dans le cas standard, une certaine proportion de solutions de bonne qualité si celles-ci sont disponibles. De même, ceci s'applique à la résolution de tous les problèmes et en particulier aux problèmes d'ordonnancement.

La suite de cette présentation est orientée vers la description de deux méthodes particulières mais représentatives de codage du job shop.

II.3.2 Méthodes de codage du job shop

De multiples représentations ont été proposées pour résoudre ce problème par une approche génétique (Mattfeld, 1995; Bagchi, 1999). Il s'agit de codages soit strictement binaires (Nakano, 1991), soit exploitant la modélisation du job shop sous la forme de graphe disjonctif (Tamaki, 1992), soit encore des codages basés sur des listes de préférences indiquant l'ordre de passage des jobs sur les machines, ou utilisant des heuristiques de priorité (Falkenauer, 1991). L'ensemble de ces approches peut être classé en deux catégories :

- codage "direct" : un chromosome représente directement un ordonnancement. La fonction d'évaluation s'applique donc directement au chromosome. L'ensemble de l'espace de recherche génotypique est couvert. En revanche, cette approche nécessite généralement des opérateurs génétiques complexes de manière à produire des solutions faisables par réparation des solutions non admissibles.

- codage "indirect" : un chromosome contient des règles ou " heuristiques " de construction d'un ordonnancement. Ces règles de construction sont interprétées par un décodeur qui élabore une solution faisable. Dans le contexte de l'ordonnancement ce décodeur est appelé "ordonnanceur". Ce type de codage évite la réalisation d'opérateurs génétiques complexes mais ne couvre pas nécessairement la totalité de l'espace de recherche.

Parmi l'ensemble des méthodes existantes, nous avons choisi de décrire en détail deux méthodes particulières, qui illustrent chacune, l'une des classes précitées. En outre, les deux classes de codage ont été utilisées dans les chapitres suivants : la méthode de résolution développée au chapitre IV pour l'ordonnancement cyclique en production flexible manufacturière est de type direct alors que celles présentées aux chapitres III et V concernant respectivement le job shop cyclique à contraintes linéaires et le job shop flexible sont de type indirect.

II.3.2.1 Méthode directe de Yamada (1992)

Cette méthode consiste à utiliser un chromosome contenant les dates de fin de réalisation des opérations ainsi qu'un algorithme réalisant des ordonnancements actifs. Le chromosome utilisé peut être vu comme une matrice à deux dimensions. Chaque ligne représente un job et chaque colonne les opérations. Chaque élément de cette matrice contient le numéro d'opération et la date de fin de réalisation de cette opération. La Figure II-6 présente un chromosome dans le cas d'un problème à 6 jobs composés chacun de 6 opérations.

job \ o_{ij}						
J_1	C_{11}=1	C_{12}=4	C_{13}=22	C_{14}=37	C_{15}=45	C_{16}=55
J_2	C_{21}=8	C_{22}=13	C_{23}=23	C_{24}=38	C_{25}=48	C_{26}=52
J_3	C_{31}=6	C_{32}=10	C_{33}=18	C_{34}=27	C_{35}=28	C_{36}=49
J_4	C_{41}=13	C_{42}=18	C_{43}=27	C_{44}=30	C_{45}=38	C_{46}=54
J_5	C_{51}=22	C_{52}=25	C_{53}=30	C_{54}=42	C_{55}=51	C_{56}=53
J_6	C_{61}=16	C_{62}=19	C_{63}=28	C_{64}=32	C_{65}=42	C_{66}=43

C_{ij} date de fin de l'opération o_{ij}

Figure II-6 : Chromosome pour le job shop selon Yamada (1992)

Le croisement de Yamada (1992) exploite l'algorithme de Giffler et Tompson présenté au premier chapitre, qui permet de générer des ordonnancements actifs. Cet opérateur de croisement nommé GA/GT est décrit dans l'Algorithme II-2. L'opérateur GA/GT construit un ordonnancement actif à partir d'une matrice d'héritage binaire "H" et de deux parents, nommés $Parent_1$ et $Parent_2$, qui sont eux-mêmes des ordonnancements actifs. Cette matrice d'héritage de dimension m x n (m : nombre de machines, n: nombre de jobs) qui est initialisée à chaque paire de descendants créés, sert à déterminer de quel parent l'enfant hérite lors du choix d'une opération à ordonnancer. Plus précisément, lors d'un conflit d'utilisation concernant la machine M_k et portant sur la $p^{ième}$ opération à placer sur cette machine, l'opération choisie est issue du chromosome $Parent_2$ si la valeur H [k][p] est nulle alors qu'elle est issue de $Parent_1$ si H [k][p] vaut 1.

Algorithme II-2

Initialiser aléatoirement la matrice binaire H de dimension m x n (m machines, n jobs)

Faire

Soit $E = \{ o_{ij} \in T \mid o_{ij}$ non encore ordonnancée$\}$

Rechercher $o^* \in E$ ayant la plus petite date de fin ($C_{o^*}=\min C_o, o \in E$) et soit M_k la machine affectée à o^*

Déterminer G l'ensemble des opérations non ordonnancées de M_k, en conflit avec o^*

Soit p, le numéro de l'opération à placer sur M_k i.e p-1 opérations sont déjà placées sur M_k

Si H [k][p] =1 **Alors**

Choisir $o \in G$ sur le chromosome **$Parent_1$** | t_o est minimum

i.e. o est l'opération qui démarre le plus tôt

Sinon

Choisir $o \in G$ sur le chromosome **$Parent_2$** | t_o est minimum

FinSi

Placer l'opération o au plus tôt dans l'ordonnancement et supprimer o de E

TantQue E@i i.e. il reste des opérations à ordonnancer

Algorithme II-2 Opérateur GA/GT

L'opérateur de mutation proposé consiste à modifier la sélection de l'opération sur le parent dont hérite le chromosome descendant, en utilisant un choix alternatif à l'opération démarrant au plus tôt.

L'initialisation de la population est réalisée à partir d'un algorithme capable de réaliser des ordonnancements actifs.

Le codage est direct dans ce cas, car il est possible de trouver directement la valeur du *fitness* à partir de la matrice représentant le chromosome. De plus, l'opérateur GA/GT ne crée aucune solution infaisable.

II.3.2.2 Méthode indirecte des permutations avec répétitions de Bierwirth (1995)

Cette méthode de représentation indirecte est basée sur la notion de permutations avec répétitions : dans le chromosome, le même allèle est répété plusieurs fois.

Considérons par exemple un problème de job shop comportant 3 jobs constitués de 3 opérations. Dans ce

cas un chromosome est composé d'une succession de neuf gènes ayant pour valeur possible l'ensemble J={ J_1, J_2, J_3}. Les allèles J_1, J_2 et J_3 seront présents trois fois dans le chromosome car chaque job est constitué de trois opérations. Ce chromosome est ensuite évalué à l'aide d'un ordonnanceur qui affecte les opérations aux machines et les ordonne au cours du temps. En utilisant le chromosome de la Figure II-7, l'ordonnanceur affectera en priorité la première tâche du job J_1, puis la deuxième tâche du job J_1, puis la première tâche du job J_2 et ainsi de suite jusqu'à la troisième tâche du job J_2.

J_1	J_1	J_2	J_3	J_2	J_1	J_3	J_3	J_2

Figure II-7 Permutation avec répétition : exemple de chromosomes

L'intérêt de cette représentation est de définir un ordre total dans l'ensemble des permutations des opérations de tous les jobs, sans générer aucune solution infaisable. Une méthode de recherche locale par amélioration permet de construire un ordonnancement actif à partir d'un premier ordonnancement correspondant au chromosome (Bierwirth, 1995). Cet ordonnancement actif est obtenu en opérant des glissements " globaux " à gauche sur certaines tâches -i.e. glissements qui inversent l'ordre des opérations sur une ressource. De plus le " forçage génétique " est utilisé pour mettre à jour le chromosome correspondant. Cette méthode de résolution permet de réutiliser les opérateurs génétiques développés pour les problèmes de type voyageur de commerce basés sur une représentation sous la forme de permutation tels que ceux présentés ci-dessous.

II.3.3 Opérateurs de croisement

De nombreux opérateurs de croisement ont été proposés pour les problèmes d'ordonnancement et les représentations de type permutation. Nous restreignons leur présentation à deux d'entre eux: d'une part l'opérateur *Order cross over* et d'autre part l'opérateur *Partially Mapped cross over* utilisé au chapitre III.

II.3.3.1 OX : order cross over

Trois variantes de cet opérateur existent. Nous présentons la version proposée par Davis(1985) qui génère les descendants en trois étapes. L'élaboration du descendant $Enfant_1$ à partir de deux parents $Parent_1$ et $Parent_2$ est décrite ci-dessous. Un raisonnement dual est applicable pour la construction du deuxième descendant $Enfant_2$.

Etape 1 : choisir dans les deux parents une sous-séquence interne, comprise entre deux points de coupure tirés aléatoirement ; cette sous-séquence interne <2,4,6> est mentionnée en caractère gras ci-dessous.

$Parent_1$: 1 3 5 7 9 | **2 4 6** | 8 10

$Parent_2$: 10 1 9 2 8 | **7 3 4** | 6 5

Etape 2 : recopier la sous-séquence interne du $Parent_1$ dans le descendant $Enfant_1$ aux mêmes positions et retirer du chromosome $Parent_2$ les allèles compris dans cette sous-séquence.

$Enfant_1$: • • • • • | **2 4 6** | • •

$Parent_2$: 10 1 9 • 8 | **7 3** • | • 5

Le chromosome $Parent_2$ permet alors de former une séquence d'allèles résiduelles en partant du deuxième point de coupure et en considérant le chromosome comme une chaîne circulaire. A titre indicatif la séquence obtenue dans notre exemple est la suivante : 5, 10, 1, 9, 8, 7, 3.

Etape 3: compléter les gènes disponibles du descendant $Enfant_1$ en lui transmettant dans l'ordre les allèles issus de la séquence résiduelle précédente.

$Enfant_1$: 1 9 8 7 3 | **2 4 6** | 5 10

$Parent_2$: 10 1 9 • 8 | **7 3** • | • 5

Cet opérateur vise donc à conserver dans les descendants les positions relatives des allèles du second parent. L'opérateur OX est donc d'un intérêt potentiel pour les problèmes d'ordonnancement dans lesquels l'ordre de réalisation des opérations est primordial.

II.3.3.2 PMX : partially mapped cross over

Cet opérateur (Goldberg, 1985) agit en une succession de quatre étapes mentionnées ci-dessous pour élaborer le descendant $Enfant_1$. Un raisonnement dual est applicable pour la construction du descendant $Enfant_2$.

Etape 1 : Cette étape est identique à celle de l'opérateur OX.

$Parent_1$: 1 3 5 7 9 | **2 4 6** | 8 10

$Parent_2$: 10 1 9 2 8 | **7 3 4** | 6 5

Etape 2: recopier la sous-séquence interne du $Parent_1$ dans le descendant $Enfant_1$ aux mêmes positions. Notons, en vue de procéder à la dernière étape de ce processus, qu'il est alors possible d'établir entre les deux points de coupure, une table de correspondance entre les allèles du $Parent_2$ et de l'$Enfant_1$ (2->7 ; 4->3 ; 6->4).

Enfant$_1$: • • • • • | **2 4 6** | • •

Parent$_2$: 10 1 9 2 8 | **7 3 4** | 6 5

Etape 3: compléter les gènes libres du descendant Enfant$_1$ en lui transmettant, les allèles non encore utilisés (i.e. sans conflit) issus du Parent$_2$ en respectant leur position. Les allèles déjà utilisés (i.e. en conflits) sont traités à l'étape suivante.

Enfant$_1$: 10 1 9 • 8 | **2 4 6** | • 5

Parent$_2$: 10 1 9 2 8 | **7 3 4** | 6 5

Etape 4: terminer l'élaboration du descendant Enfant$_1$ en résolvant successivement les différents conflits. Chaque allèle en conflit est remplacé par l'allèle correspondant de la table mentionnée à l'étape 2. Dans l'exemple 7 remplace 2 (2->7) et 3 remplace 6 (6->4 puis 4-> 3).

Enfant$_1$: 10 1 9 7 8 | **2 4 6** | 3 5

Parent$_2$: 10 1 9 2 8 | **7 3 4** | 6 5

Cet opérateur privilégie dans le descendant l'incorporation d'une sous-séquence complète de l'un des parents, tout en conservant au maximum leur position. Dans l'exemple ci-dessus, les sous-séquences <2,4,6> de Parent$_1$ et <10,1,9> de Parent$_2$ se retrouvent dans le descendant Enfant$_1$ aux mêmes positions.

En conclusion, et bien qu'il n'existe pas à notre connaissance d'étude complète de comparaison des performances des méthodes de codage du job shop par les algorithmes génétiques, il semble à ce jour que l'utilisation de représentations strictement binaires soit dépassée. Les approches indirectes qui utilisent l'algorithme de Giffler et Thompson pour la production d'ordonnancements actifs tant au niveau de l'ordonnanceur, qu'au niveau des opérateurs génétiques semblent plus performantes. Les meilleurs résultats publiés combinent dans le gène la paire d'information suivante : méthodes d'ordonnancement et heuristiques de sélection de tâches (Hart, 1998). La méthode d'ordonnancement définit le type de l'ordonnancement actif ou sans délai. Les heuristiques sont choisies parmi un ensemble de douze règles classiques en ordonnancement telles que par exemple la règle EDD (Earliest Due Date) du plus petit délai. Ces règles sont exploitées pour sélectionner une opération dans l'ensemble des opérations en conflit sur une machine.

II.4 Conclusion

Une introduction au paradigme des méthodes évolutionnistes a été présentée dans ce deuxième chapitre. Ces méthodes inspirées des mécanismes de l'évolution naturelle connaissent un développement très important dû à la source d'inspiration géniale et immense que constitue l'évolution naturelle mais aussi aux bonnes performances qu'elles obtiennent sur des problèmes réels académiques ou industriels. Cette présentation a été orientée d'une part vers les méthodes d'optimisation multicritères et d'autre part vers leur utilisation pour la résolution de problèmes d'ordonnancement, qui constitue les thèmes abordés dans les chapitres suivants.

III Job shop cyclique à contraintes linéaires

Ce chapitre propose tout d'abord une modélisation du problème de job shop cyclique à contraintes linéaires. Le principe général de résolution approchée de ce problème est abordé dans un deuxième temps. La présentation décrit d'une part l'ordonnanceur au centre de notre approche de résolution et d'autre part la mise en oeuvre de l'algorithme génétique. Finalement une validation de cette approche est présentée en s'appuyant sur un ensemble de problèmes tests.

Il convient de noter que l'intégralité de ce chapitre a été publiée sous les références (Cavory, 2003) et (Dupas, 2003b). J'ai toutefois choisi de le décrire intégralement ci-dessous afin de présenter les trois applications cibles étudiées dans le domaine de l'ordonnancement, de façon uniforme.

III.1 Définitions et hypothèses

Le job shop cyclique à contraintes linéaires est modélisé comme suit :

- $J = \{ J_i \}_{i=1}^{n}$ ensemble des jobs à réaliser et o_{ij}, l'opération j du job J_i
- $< o_{ij} , q >$: $q^{ième}$ occurrence de l'opération cyclique o_{ij}
- C : ensemble des contraintes linéaires entre les opérations cycliques
- $M = \{ M_k \}_{k=1}^{m}$ ensemble des machines et m_{ij} la machine affectée à l'opération o_{ij} avec un temps opératoire p_{ij}.
- C_i^q : date de fin de la $q^{ième}$ exécution de J_i : $C_i^q = \max_{j=1}^{m_i} \{ t_{ij}(q) + p_{ij} \}$,

Rappelons que ces opérations génériques sont :

- non-réentrantes: deux exécutions d'une même opération ne peuvent pas se superposer
- non-préemptives : les opérations débutées ne peuvent être interrompues
- pré-affectées aux machines

Le modèle considéré comporte de plus les trois restrictions suivantes :

- chaque machine ne peut traiter qu'une seule opération à la fois
- chaque opération d'un job ne peut être exécutée que par une seule machine à la fois
- chaque job visite chaque machine une fois au plus.

La modélisation du job shop cyclique exprimée ci-dessus possède une singularité importante, qui la distingue de celle du job shop standard : les contraintes de précédence relatives au job portent dans le cas général sur des opérations dont les numéros d'occurrence ne sont pas identiques.

Un exemple de job shop cyclique est présenté en Figure III-1. Il est formé de 3 jobs (J_1,J_2,J_3) constitué chacun de 3 opérations cycliques :

- Ensemble des machines : M= {M_1,M_2,M_3}

- Ensemble des jobs : J={ J_1,J_2, J_3}

- Affectation des opérations :

$m_{1,1}=M_1$; $m_{1,2}=M_3$; $m_{1,3}=M_2$; $m_{2,1}=M_2$;

$m_{2,2}=M_1$; $m_{2,3}=M_3$; $m_{3,1}=M_1$; $m_{3,2}=M_2$; $m_{3,3}=M_3$;

- Temps d'exécutions :

p_{11}=3ut ; p_{12}=4ut; p_{13}=2ut; p_{21}=2ut; p_{22}=6ut; p_{23}=3ut; p_{31}=1ut; p_{32}=2ut; p_{33}=7ut;

Il convient de noter que ce graphe linéaire doit vérifier les propriétés importantes suivantes:

- la Propriété I-4 qui exprime la condition nécessaire et suffisante de cohérence d'un graphe unitaire et l'existence d'un ordonnancement infini. Dans le cas particulier des graphes cycliques représentant les jobs à réaliser (voir Figure III-1), la condition à respecter pour les valeurs des contraintes linéaires de l'ensemble T_i des tâches du job cyclique J_i est donc: $\prod g_{o_{ij}o_{ij'}} / a_{o_{ij}o_{ij'}}=1$; avec o_{ij}, $o_{ij'} \in T_i$, et $Co_{ij},o_{ij'} \in C$.

Les valeurs du graphe de la Figure III-1 ont été choisies de manière à respecter cette condition de cohérence.

- l'absence d'interblocage dans le graphe doit également être respectée. A cette fin, et puisque selon la Propriété I-2 les premières occurrences des opérations sont non contraintes**,** il est nécessaire de supposer que ces occurrences peuvent être réalisées à l'état initial. A titre d'exemple, pour le job J_3 la première occurrence des opérations o_{31} et o_{32} peut être réalisée lors de l'initialisation. Plus particulièrement, pour la contrainte $e(o_{31},o_{32})$= (1,0,2,0), $\gamma_{31\,32}+\omega_{31\,32}$ -1 vaut 1, donc la première occurrence de o_{32} peut être réalisée à l'état initial.

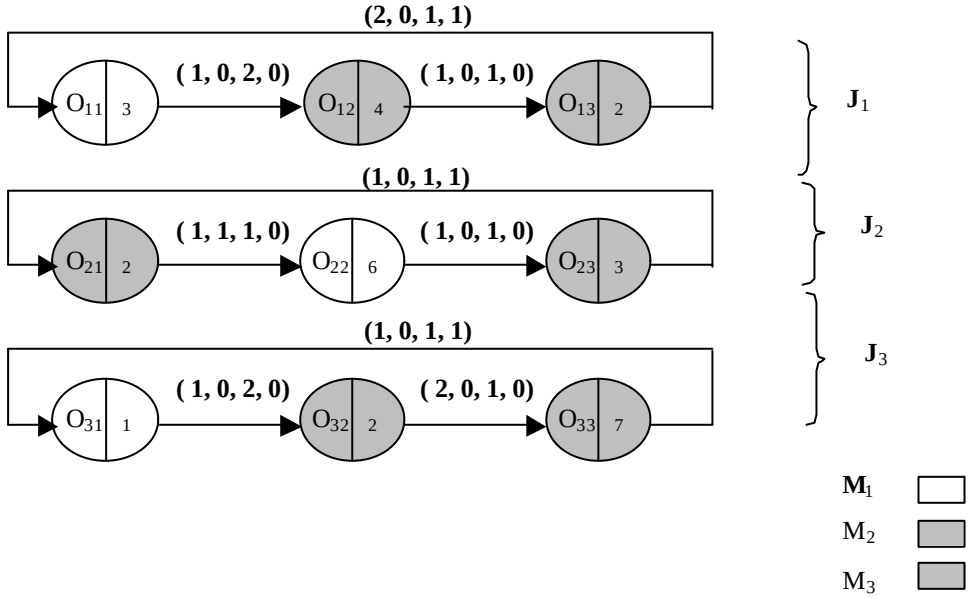


Figure III-1 Exemple de job shop cyclique à contraintes linaires

L'objectif vise à trouver un ordonnancement, i.e. à déterminer les dates de début de chaque occurrence des opérations génériques qui minimise la moyenne des temps de cycle des jobs TC_{moy} . Cet ordonnancement

est infini en théorie, mais il est calculé en pratique sur un horizon fixé.

Le temps de cycle moyen du job J_i a été défini au premier chapitre (Définition I-9). Afin d'implémenter le calcul de C_i^q, nous avons choisi d'ajouter une opération fictive de durée nulle o_{i*} pour chaque job i. Elle permet de comptabiliser le nombre de jobs complètement réalisés car elle n'est exécutée qu'une seule fois pour chaque instance du job. Considérons à titre d'exemple le job J_3 représenté en Figure III-2: la contrainte linéaire $e(o_{33}, o_{31})=(1,0,1,1)$ est décomposée en deux contraintes linéaires $e(o_{33}, o_{3*})=(1,0,1,0)$ et $e(o_{3*}, o_{31})=(1,0,1,1)$. Cette décomposition peut être prouvée aisément en écrivant les relations d'ordre équivalentes à ces contraintes – voir Définition I-12.

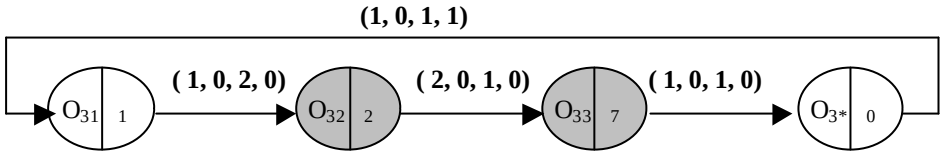


Figure III-2 Job J_3 complété pour le calcul de TC_3

En conséquence, la moyenne du temps de cycle du job J_i est définie comme suit : $TC_i = \lim_{q\to\infty} \frac{1}{q} C_i^q$

Définition III-1 Temps de cycle moyen des jobs

Dans le contexte des systèmes de production, le job représente un produit à réaliser. Le but final étant d'optimiser le rendement exprimé en nombre de jobs produits et donc de minimiser leur temps de cycle. Une hypothèse supplémentaire considérée est qu'il n'y pas de programme de production imposé. L'objectif est donc formalisé comme suit :

$$Min TC_{moy} \text{ avec } TC_{moy} = \sum_{i=1}^{n} TCi / n$$

Pour le problème étudié qui comporte des contraintes de ressources, la dominance des solutions périodiques n'ayant pas été démontrée pour le critère choisi, la recherche est donc focalisée sur les solutions cycliques minimisant la moyenne des temps de cycle des jobs.

Par ailleurs il convient d'ajouter que la spécification ci-dessus induit deux caractéristiques importantes relatives au type de problème traité :

- la proportion relative des différents jobs n'est pas imposée puisque ceux-ci sont indépendants, ce qui est directement observable sur le graphe de la Figure III-1.

- l'encours ou taux de recouvrement, correspondant au nombre de jobs identiques en cours d'exécution simultanément, est imposé par les valeurs de la contrainte linéaire qui relie la première à la dernière opération

d'un job. Cette contrainte de retour sur la tâche initiale de chaque job évite donc l'engorgement du système cyclique dû à un nombre incontrôlé d'occurrences réalisées.

A notre connaissance, la complexité du job shop cyclique à contraintes linéaires est un problème ouvert. Néanmoins sous réserve de respecter la Propriété I-4, l'expansion du graphe linéaire engendre un graphe uniforme équivalent qui définit alors un problème de job shop à contraintes uniformes lui-même de complexité NP-difficile. Pour cette raison, l'utilisation de méthodes de résolution approchées telles que les méta-heuristiques est justifiée, y compris pour des instances de problèmes de taille limitée. Le paragraphe suivant présente une approche de résolution génétique du problème cyclique formulé ci-dessus.

III.2 Approche de résolution et codage

Dans le problème étudié, une méthode de codage direct paraît difficile à mettre en oeuvre, en raison d'une part du nombre potentiellement infini d'opérations à ordonnancer et d'autre part du type de solutions recherchées qui n'est pas strictement périodique. Par ailleurs, pour la résolution de job shop standard i.e. non cyclique, les travaux récents utilisent principalement un codage indirect en couplant l'algorithme génétique avec un ordonnanceur performant permettant de générer des ordonnancements actifs – i.e. ordonnancement tel qu'aucune opération ne peut être avancée sans en reporter une autre (voir Chapitre I).

Notre approche (Cavory, 2003) est donc basée sur un codage indirect qui est présenté en Figure III-3

- Le premier module est l'ordonnanceur qui construit un ordonnancement pas à pas en respectant les contraintes de précédences définies dans le graphe linéaire. En cas de détection de conflits d'utilisation d'une même ressource par de multiples opérations au même instant, l'ordonnanceur utilise les règles de résolution proposées dans le génome. Pour ce faire, ce module exécute le graphe linéaire traduit sous la forme d'un modèle de réseau de Petri.

- Le second module est l'algorithme génétique qui fait évoluer une population d'individus engendrés par la reproduction de chromosomes parents en utilisant les opérateurs de croisement et de mutation. Ce module génétique est décrit après la description de l'ordonnanceur qui est donnée ci-dessous.

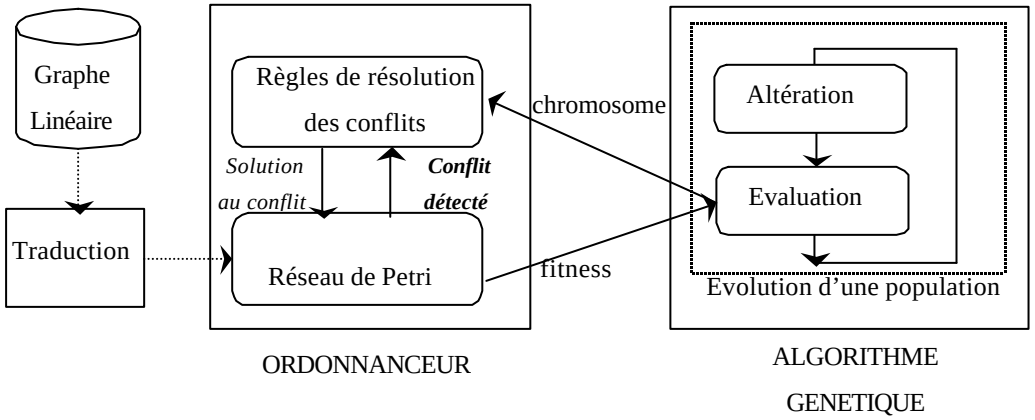


Figure III-3 Approche générale de résolution

III.2.1 Ordonnanceur

Le principe de fonctionnement de l'ordonnanceur consiste à construire une solution en " simulant " le parcours du graphe de précédence à contraintes linéaires, noté GPCL dans la suite du chapitre. Dans cet objectif, une modélisation de ce graphe à l'aide des réseaux de Petri T-temporisés et pondérés a été utilisée. Celui-ci est noté RdP par la suite.

III.2.1.1 Modélisation

Le RdP associé au GPCL est constitué de places et de transitions qui représentent les opérations à réaliser et les jetons sont les occurrences de ces opérations.

La Figure III-4 présente le RdP associé à un GPCL contenant une seule contrainte entre deux opérations "a" et "b": $e(a,b) = (\alpha_{ab}, \beta_{ab}, \gamma_{ab}, \omega_{ab})$ dans le cas particulier où β_{ab} et ω_{ab} sont nuls et où les opérations "a" et "b" nécessitent une même ressource notée "Ress 1" . La transformation du GPCL en RdP nécessite d'examiner successivement, les noeuds, les arcs, puis l'initialisation du système et enfin la gestion des ressources.

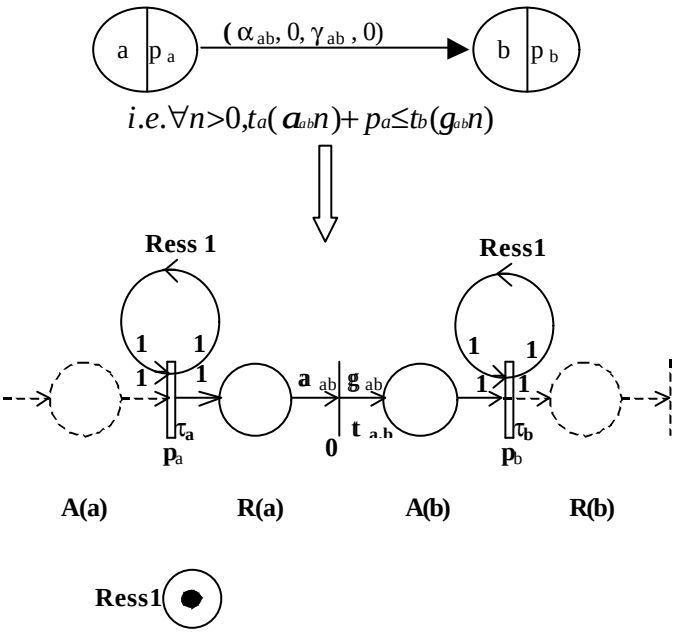


Figure III-4 Transformation GPCL en RdP

III.2.1.2 Noeud du GPCL

Un noeud "a" du GPCL est représenté par deux places A(a) et R(a) séparées par une transition τ_a. Les jetons situés dans la place A(a) représentent l'ensemble des occurrences réalisables de l'opération "a". Les jetons présents dans R(a) représentent l'ensemble des dernières occurrences réalisées de l'opération "a", qui

permettront lorsque la contrainte sera satisfaite de débloquer la réalisation des occurrences suivantes de l'opération "b". La transition τ_a représente le temps de réalisation de l'opération "a". Si le noeud "a" ne possède aucun arc entrant, la place A(a) devient inutile car dans ce cas toutes les occurrences de l'opération "a" sont réalisables sans contrainte. Pour cette raison, sur la Figure III-4, la place A(a) est en pointillé.

III.2.1.3 Arc du GPCL

Les arcs du GPCL représentent la contrainte linéaire qui bloque la réalisation de certaines occurrences d'opérations. Un arc du GPCL entre les opérations "a" et "b", est associé à une transition $\tau_{a,b}$ qui sépare les places R(a) et A(b). Le temps de tirage de cette transition est nul, car celle-ci représente la satisfaction de la contrainte linéaire. Pour tirer cette transition, il faut que l'opération "a" soit réalisée α_{ab} fois pour autoriser les γ_{ab} occurrences suivantes de l'opération "b". Lors du tirage de la transition $\tau_{a,b}$, α_{ab} jetons sont retirés de R(a) et γ_{ab} jetons sont placés dans la place A(b). Si aucun arc ne sort du noeud "b", la place R(b) devient inutile, ce qui est représenté en pointillé sur la Figure III-4

La Figure III-4 présente la transformation d'une contrainte linéaire simplifiée ne générant pas de régime transitoire, ce qui équivaut à attribuer la valeur nulle au paramètre β_{ab}. Dans le cas général, il convient d'intégrer dans le réseau de Petri, les β_{ab} jetons supplémentaires requis pour tirer la première fois la transition τ_a. Dans cet objectif, le nombre de jetons nécessaire pour tirer cette transition devient $\alpha_{ab}+B_{ab}(a)$, avec la fonction B_{ab} définie comme suit :

- $B_{ab}(a) = \beta_{ab}$ si il s'agit du premier tirage de $\tau_{a,b}$
- $B_{ab}(a) = 0$ sinon

III.2.1.4 Initialisation du RP

L' Algorithme III-1 définit le marquage initial du RdP dans le cas général d'une contrainte linéaire entre deux tâches "a" et "b" : $e(a,b) = (\alpha_{ab}, \beta_{ab}, \gamma_{ab}, \omega_{ab})$. Son objectif est de permettre le démarrage de l'ordonnanceur. Le principe consiste à supposer que toutes les premières occurrences non contraintes des opérations peuvent être réalisées à l'instant initial (voir Propriété I-2). Cet algorithme s'applique à tout noeud n_a du graphe, associé à l'opération cyclique "a".

Algorithme III-1

Pour tous les noeuds n_a du graphe
  Si $\exists\, e(b,a) \in C \mid e(b,a) = (\alpha_{ba}, \beta_{ba}, \gamma_{ba}, \omega_{ba})$
    Si $\gamma_{ba} + \omega_{ba} - 1 \geq 1$
      Ajouter $(\gamma_{ba} + \omega_{ba} - 1)$ jetons dans A(a)
    FinSi
  FinSi
FinPour

Algorithme III-1 Initialisation du Réseaux de Petri

III.2.1.5 Gestion des ressources

L'exemple de la Figure III-4 comporte une ressource unique, capable de réaliser les opérations "a" et "b", qui est représentée par le label "Ress1". Ce label permet d'alléger la représentation du RdP ; il désigne une place contenant un nombre de jetons équivalent à la capacité de cette ressource i.e nombres d'opérations exécutables par cette ressource. Les conflits potentiels d'utilisation de cette ressource sont résolus à l'aide de l'algorithme génétique qui propose un ordre de passage des jobs.

III.2.1.6 Exemple de fonctionnement

Ce fonctionnement est illustré à partir du graphe de la Figure III-5, en prenant les hypothèses simplificatrices suivantes de manière à simplifier le RdP et son fonctionnement :

- Les valeurs du paramètre β sont toutes nulles.
- Il n'y a pas de conflit de ressources
- L'opération fictive o_{i*} de terminaison des jobs a été supprimée.

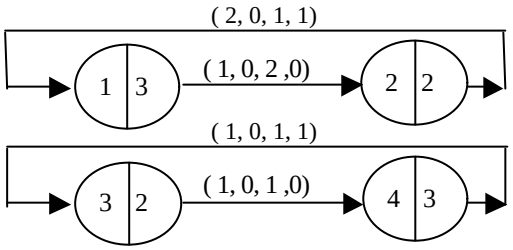


Figure III-5 GPL simplifié

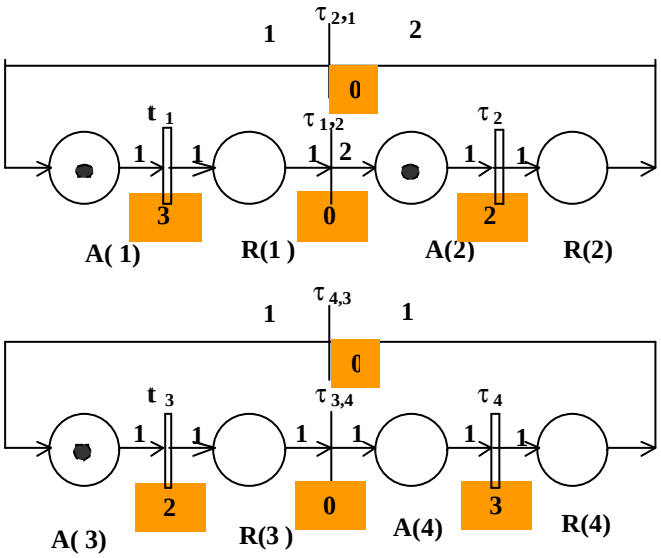


Figure III-6 Rdp initial associé au GPCL de la Figure III-5

La Figure III-6 présente donc le RdP dans l'état initial, associé à ce graphe linéaire simplifié. Les places A(i) contenant un jeton correspondent aux opérations réalisables au démarrage qui sont définies à l'aide de l'Algorithme III-1. Il s'agit dans cet exemple des opérations 1, 2 et 3.

La Figure III-7 décrit l'évolution temporelle du marquage des places associées à la Figure III-6 . L'état initial du RdP correspond à l'instant "0" du diagramme . Dans ce diagramme, l'axe des temps comporte des instants notés "i" qui correspondent au franchissement d'une transition relative à la fin de réalisation d'une opération cyclique donnée; il comporte également des instants fictifs notés "i'" correspondant quant à eux aux franchissements d'une transition relative à la satisfaction d'une contrainte linéaire dont la durée est nulle. Dans cet exemple, le système retourne à l'état initial à l'instant 5'.

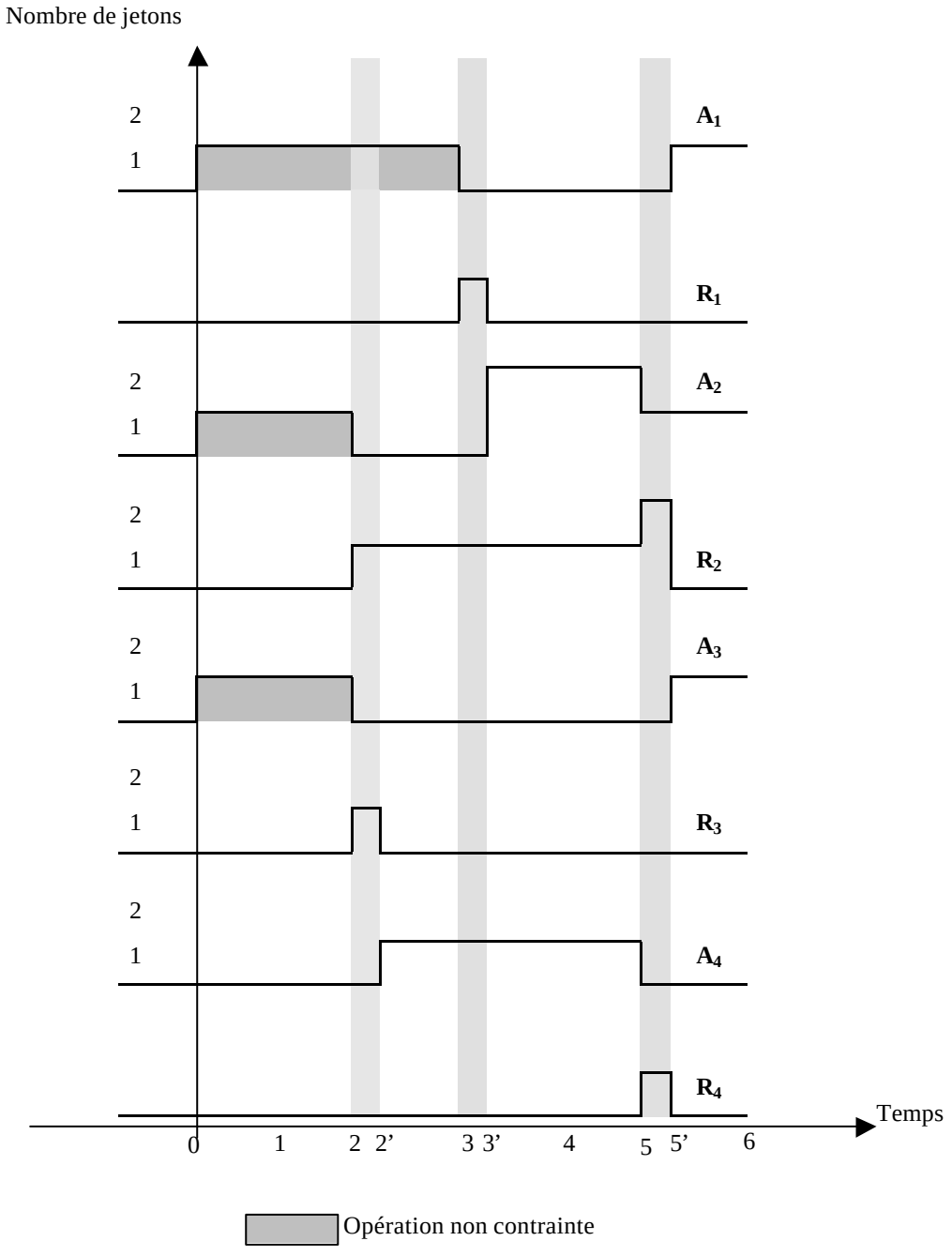


Figure III-7 Diagramme temporel du marquage des places

Le simulateur décrit ci-dessus évalue donc la fonction de coût ou *fitness* qui permet à l'algorithme génétique d'évaluer la pertinence de chaque solution. La mise en œuvre de cet algorithme est donnée dans le paragraphe suivant.

III.2.2 Algorithme génétique

Cette mise en œuvre nécessite de spécifier les trois points suivants : la méthode de représentation des individus, les opérateurs génétiques employés et la fonction *fitness*.

III.2.2.1 Codage des solutions

Un chromosome peut être vu comme composé d'un ensemble de sous-chromosomes, chacun étant relatif à une machine. Chaque sous-chromosome consiste en une liste de préférences, qui mentionne l'ordre de priorité des opérations sur la machine. Les allèles sont donc les jobs concernés. La position du gène dans chaque sous-chromosome définit l'ordre de priorité dans lequel ce job est exécuté sur la machine. Dans l'exemple de la Figure III-1, si le sous-chromosome relatif à la machine M_1 contient les valeurs suivantes $\{J_2, J_1, J_3\}$, alors l'ordre de traitement des opérations proposé est d'abord J_2 (opération o_{22}) puis J_1 (opération o_{11}), et finalement J_3 (opération o_{31}). Cette méthode de représentation a été proposée dans le cadre du job shop standard non cyclique par plusieurs auteurs tel que (Falkenauer, 1991) par exemple.

Cependant ce génotype se contente de proposer une priorité de passage des opérations sur les machines, sans considérer les numéros d'occurrences de ces opérations. C'est pourquoi, lors d'un conflit de ressources, le choix de l'opération à exécuter est effectué selon l'Algorithme III-2. Cet algorithme qui définit un ordre de priorité sur les tâches procède en deux étapes :

- Dans un premier temps, T_r' sous-ensemble de T_r, est créé ; il contient une sélection des tâches en conflit basée sur le nombre d'occurrences des opérations déjà réalisées. Plus précisément, un nombre plafond d'occurrences réalisées est utilisé pour sélectionner les tâches. Ce nombre plafond est défini comme étant la somme d'un constante L, le nombre minimum d'occurrences déjà réalisées, et d'un paramètre variable de hauteur, H. Il est à noter que si H possède la valeur nulle, toutes les opérations ayant le plus faible nombre d'occurrences réalisées sont choisies pour engendrer l'ensemble Tr' ; inversement si H est maximum, toutes les opérations de l'ensemble T_r sont inclues dans l'ensemble Tr' i.e. $\{T_r\} = \{ Tr' \}$.

- Une tâche est ensuite sélectionnée dans T_r' en fonction de la priorité mentionnée dans le chromosome pour la machine concernée. Cette tâche est ensuite placée par le simulateur dans l'ordonnancement en cours d'élaboration.

Pour illustrer le principe de résolution de conflit, considérons le sous-chromosome précédent $\{J_2, J_1, J_3\}$ relatif à la machine M_1. Deux exemples de fonctionnement de cet algorithme sont présentés utilisant tous deux une valeur nulle pour le paramètre de " hauteur " H.

Algorithme III-2

Entrée: T_r: Ensemble des opérations en conflit

Entrée: M_k: Machine cible des opérations en conflit

Sortie: $<o_{ij},q>$ occurrence sélectionnée dans T_r

Création de T_r': sous-ensemble de T_r, des opérations en conflit ayant un nombre d' occurrences réalisées inférieur ou égal à L+H

L : nombre minimum d'occurrences déjà réalisées parmi toutes les opérations de T_r

H : paramètre "hauteur" variant de 0 à H_{max}, H_{max} i û

Sélection de $<o_{ij,}\ q>$ dans T_r', opération ayant la plus haute priorité sur la machine M_k, en fonction de l'heuristique définie par le chromosome courant

Algorithme III-2 Résolution de conflits

Exemple 1*:* l'ensemble T_r des opérations en conflit contient les occurrences suivantes $\{<o_{11}, 2>, <o_{22}, 2>, <o_{31}, 2>\}$qui sont exécutables au même instant t. Le premier niveau de résolution sélectionne ces trois occurrences dans l'ensemble Tr', car elles ont toutes le même numéro d'occurrences. Puis le second niveau sélectionne l'occurrence $<o_{22},2>$, car J_2 est situé en première position dans le chromosome.

Exemple 2: L'ensemble T_r des opérations en conflit contient les occurrences suivantes $\{<o_{11}, 2>, <o_{22}, 3>, <o_{31}, 2>\}$qui sont exécutables au même instant t. Le premier niveau de résolution sélectionne dans l'ensemble Tr' les occurrences $<o_{11}, 2>$ et $<o_{31}, 2>$ car l'opération $<o_{22}, 3>$ possède un numéro d'occurrence réalisé plus élevé que les deux autres. Le second niveau sélectionne l'occurrence $<o_{11}, 2>$, car le job J_1 est plus prioritaire que le job J_3.

Il est important de noter que dans une approche indirecte telle que présentée ci-dessus, la totalité de l'espace de recherche n'est généralement pas explorée. Seul un espace de recherche restreint à la transformation opérée par la méthode de codage du chromosome est exploré. Il n'y pas de garantie que cet espace exploré contienne les valeurs optimales du problème.

III.2.2.2 Opérateurs génétiques

Les opérateurs génétiques suivants sont appliqués indépendamment sur chaque sous-chromosome :

- Le croisement est de type *PMX (Partially-mapped crossover)* décrit au chapitre II (Goldberg, 1985). Notons que cet opérateur est adapté à la représentation de type "permutation" qui est celle de notre chromosome. Cependant dans notre méthode qui utilise un codage indirect, il convient de remarquer que l'intérêt de cet opérateur est limité au fait d'obtenir une solution valide.

- La mutation est une simple mutation de type *swap* qui échange deux allèles pris aléatoirement dans un sous-chromosome.

III.2.2.3 Fitness

La fonction d'évaluation calcule sur un horizon donné, le temps de cycle moyen des jobs TC_{moy} correspondant au *fitness* du chromosome. L'exécution du Réseau de Petri associé au graphe linéaire permet d'élaborer un ordonnancement faisable du type "au plus tôt " et de calculer le temps cycle moyen TC_{moy} .

Une évaluation de l'approche précédemment décrite est présentée dans le paragraphe suivant en s'appuyant sur un ensemble de données de tests.

III.3 Validation et résultats

Cette validation requiert au préalable de rechercher une borne permettant d'évaluer les performances de l'approche proposée. A cet effet trois bornes ont été proposées :

- Une première borne minimum est obtenue par simulation. Elle consiste à supprimer les contraintes de ressources dans chaque problème test et à récupérer la valeur fournie par le simulateur. Cette borne est appelée B0 dans le Tableau III-1. Il est à noter que cette borne tient compte du régime transitoire induit par les paramètres β et ω du GPCL.

- Une seconde borne B1 similaire à B0 est obtenue en ignorant les contraintes de ressources et en développant le graphe uniforme obtenu par "expansion" du graphe linéaire, pendant une durée correspondant à la création d'une occurrence du job J_i. Dans ce graphe développé, chaque nœud représente une occurrence d'opération cyclique et chaque arc représente une contrainte de précédence simple. Dans chaque graphe associé au job J_i le chemin critique correspondant au plus long chemin dans ce graphe fournit la borne notée BJ_i. En conséquence une borne minimum de la moyenne des temps de cycle des jobs, TC_{moy} , peut être obtenue en calculant la moyenne des BJ_i . Cette borne minimum, appelée B1, est donc égale à : $B1=\sum_{i=1}^{n} BJi/n$

- Une troisième borne B2 est obtenue en ignorant les contraintes de précédence entre les opérations. Pour une machine M_j, une borne BM_j est calculée en considérant les temps opératoires de toutes les occurrences des opérations cycliques nécessaires pour réaliser tous les jobs affectés à cette machine, en supposant que les contraintes de précédence n'existent pas. En conséquence, une borne minorante du temps de cycle des jobs est : $B2=\sum_{j=1}^{m} BM_j/m$

Un exemple de calcul des bornes B1 et B2 est présenté en Annexe 1.

La validation de notre approche s'appuie sur un ensemble de quatre problèmes tests qui ont été proposés pour le job shop cyclique à contraintes linéaires (Cavory, 2000 ; Dupas 2001). Ces problèmes tests sont référencés par deux nombres n1_n2 : n1 étant le nombre de machines et n2 le nombre de jobs.

Les résultats obtenus sont mentionnés dans le Tableau III-1

Problèmes tests (machines_jobs)	Borne B0	Borne B1	Borne B2	Temps de Cycle Moyen (Tc_{moy}) GA
5_5	15.9	15.8	18.4	26.1
5_10	19.7	19.7	43.6	62.4
6_6	17.6	17.5	23.1	32.7
6_12	20.2	20	55.33	69.9

Tableau III-1 Résultats comparatifs :temps de cycle moyen des jobs (TC_{moy})

Ce tableau comporte pour chaque problème, le temps de cycle moyen obtenu par l'algorithme génétique, le nombre de solutions explorées par l'algorithme génétique et les bornes inférieures B0, B1 et B2 de chaque instance de problème.

Ces résultats ont été obtenus avec le paramétrage du Tableau III-2.

Paramètres	
Taille de la population	50
Probabilité croisement PMX	0.8
Taux de mutation	0.1
Méthode de sélection	tournoi
Nombre d'exécutions	10
Initialisation de la population	aléatoire
Nombre de générations	29,4 (moyenne sur les 4 instances)

Tableau III-2 Paramétrage de l'algorithme génétique

Le temps moyen de convergence de l'algorithme génétique est de l'ordre de 900 secondes sur un Pentium III sous Linux. Dans ces essais, l'ordonnanceur utilise un algorithme de résolution de conflits dont le paramétrage impose que la hauteur H soit égale à H_{max} i.e toutes les occurrences des opérations sont considérées pour générer l'ensemble T_r'.

Ces résultats peuvent être interprétés comme suit :

- Les bornes B0 et B1 sont très proches l'une de l'autre, ce qui contribue à la validation pratique de notre simulateur.

- L'écart entre la borne B2 et les meilleurs solutions trouvées par l'approche GA est compris entre 20 % et 30 % de la valeur des meilleures solutions.

Ces résultats permettent donc de valider globalement l'approche que nous avons proposée.

III.4 Conclusion

Ce chapitre présente la modélisation par réseaux de Petri et une résolution approchée par un algorithme génétique, du problème de job shop cyclique à contraintes linéaires. L'algorithme génétique est basé sur un codage indirect qui propose un ordre de priorité de passage des jobs sur les machines. Cet ordre de passage est utilisé par l'ordonnanceur pour résoudre les conflits de ressource. Outre la modélisation du problème par réseaux de Petri, ce travail apporte donc une méthode originale d'utilisation du concept des algorithmes génétiques pour les problèmes cycliques à contraintes linéaires et constitue donc une extension des méthodes de résolution du job shop standard par ces algorithmes. Cette méthode a été validée par comparaison avec des bornes minimums sur un ensemble de données de tests proposé spécifiquement pour ces problèmes.

Cette méthode constitue une première tentative de résolution du problème. Il reste bien entendu à améliorer significativement les performances de notre méthode de résolution et aussi la qualité de nos bornes. Certaines pistes d'amélioration sont envisagées dans le dernier chapitre relatif aux perspectives.

IV Ordonnancement cyclique en Production Flexible Manufacturière

Le problème d'ordonnancement cyclique en Production Flexible Manufacturière (PFM) (Jain et Elmaraghy, 1997) ainsi que les hypothèses de travail sont présentés dans un premier temps. Le principe général de résolution de ce problème par une approche génétique est décrit dans un deuxième temps. Finalement une validation de cette approche est présentée en s'appuyant sur un ensemble de problèmes tests.

IV.1 Définitions et hypothèses

Le problème étudié est décrit en trois temps. Tout d'abord, un exemple simplifié servant de support à la présentation du problème, est énoncé. Ensuite, la démarche globale de résolution du problème est présentée. Enfin, le problème d'ordonnancement est spécifié.

IV.1.1 Exemple de système de Production Flexible Manufacturière

L'exemple considéré est un système de production constitué de cinq machines produisant trois types de pièces, appelées gammes ou jobs, et utilisant p ressources de transport i.e. palettes. Les données de ce problème présenté par Gentina (2001) sont énoncées ci-dessous :

- Ensemble des gammes ou jobs: J={ A, B, C }

  Note 1 : le changement de notation remplaçant les "Ji" par les lettres "A","B", etc, vise à améliorer la lisibilité des exemples du chapitre.

- Ensemble des machines : M= {M_1,M_2,M_3,M_4,M_5}

- Ensemble de "p" palettes ou portes pièces : P={ P_1,.., P_p}

Les trois gammes du système de production sont modélisées sous la forme d'un réseau de Petri présenté en Figure IV-1. Il s'agit d'un réseau de Petri T_temporisé : les transitions portent donc les durées opératoires. Les ressources de transport i.e. palettes, ne sont pas représentées dans cette modélisation.

La gamme de fabrication est flexible, ce qui se traduit par l'existence de divergences à la sortie de certains nœuds. Ainsi par exemple dans la gamme A, il y a deux routages possibles selon que la pièce utilise la machine M_3 ou la machine M_6. Le ratio de routage Ñ , respectivement 1-Ñ, spécifie la proportion des pièces de la gamme A qui utilisent la machine M_6, respectivement M_3. D'autres formes de flexibilité peuvent être considérées mais ne sont pas prises en compte dans cette étude (Camus, 1997). La gamme B ne comporte pas de ratio de routage car les deux branches de cette gamme sont équivalentes : elles utilisent les mêmes machines avec les mêmes temps opératoires.

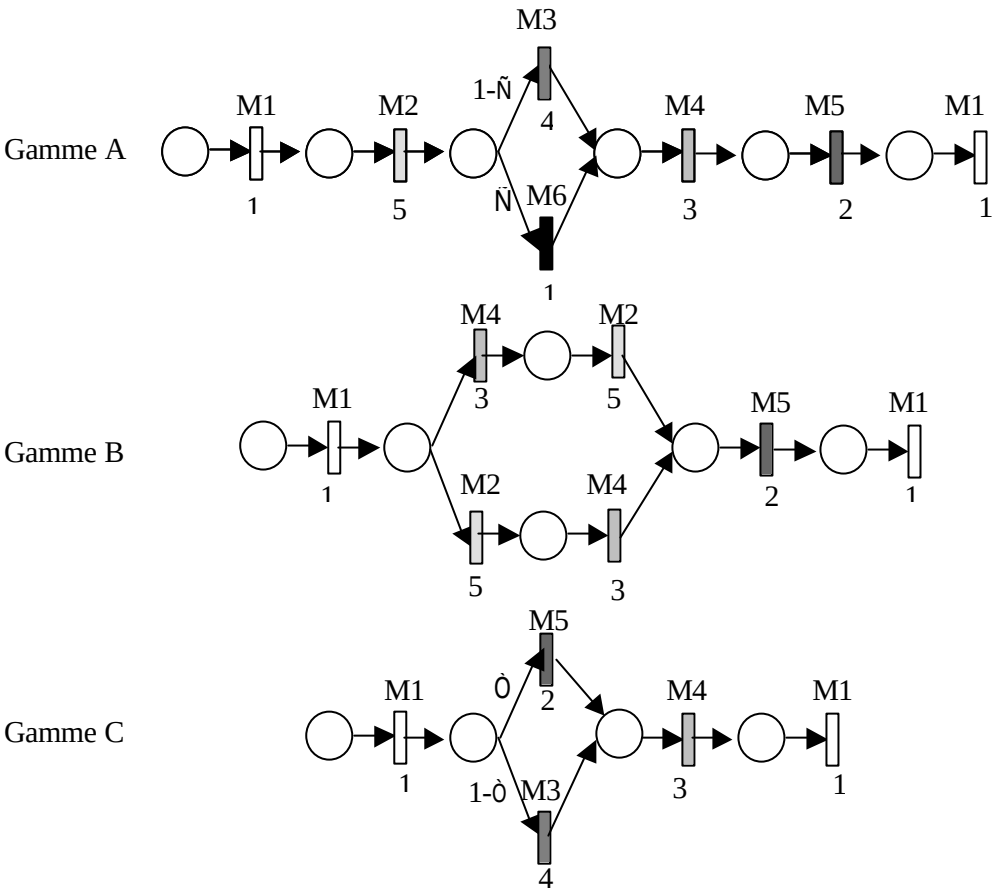


Figure IV-1 Exemple : Réseaux de Petri des gammes de fabrication

IV.1.2 Approche générale de résolution du problème de PFM

La résolution du problème de PFM est abordée ici dans le cadre de l'ordonnancement cyclique. Une approche non cyclique consisterait à réaliser séquentiellement toutes les pièces d'une gamme avant de pouvoir commencer les pièces de la gamme suivante. L'approche cyclique permet quant à elle de produire tous les types de pièces à chaque cycle de production du système dont la durée est appelée temps de cycle. Cette approche permet d'une part de limiter la complexité du problème en se focalisant sur la recherche d'un motif d'ordonnancement à répéter et ne nécessite pas l'ordonnancement de l'ensemble des occurrences des opérations à réaliser. D'autre part elle permet d'obtenir des gains de productivité grâce à l'optimisation de ce temps de cycle.

Le cadre de résolution fixé considère de plus les hypothèses suivantes :

Hypothèse 1 : l'affectation des pièces aux palettes est définitive : une palette n'est donc réutilisable qu'à l'instant ou la pièce précédente est sortie du système.

Hypothèse 2 :le système est constitué de palettes dédiées à un seul type de pièce et donc chaque pièce est affectée à un seul type de palette. Par exemple, une palette de type P_1 ne peut porter que les pièces de la gamme A.

Une alternative à cette hypothèse est le cas d'un système de palettes non dédiées ou banalisées, dans lequel une même palette peut servir pour plusieurs types de gammes. La notion de gammes est alors étendue aux macro-gammes, constituée chacune de plusieurs gammes juxtaposées. Il est à noter que l'approche proposée par la suite n'est pas limitée au cas des palettes dédiées.

Hypothèse 3: les phases transitoires de montée et de descente en charge du système ne sont pas considérées dans notre étude.

Ces problèmes de Production Flexible Manufacturière ont été étudiés en profondeur par (Gentina, 2001) et (Korbaa 2003) qui ont proposé une démarche de résolution qui est synthétisée ci-dessous. Cette démarche comporte deux étapes principales :

Etape 1 : La première étape vise d'une part à linéariser les gammes en utilisant les meilleurs branches des alternatives de chaque gamme et d'autre part à déterminer le flux optimal de production. Pour cette première étape de résolution, la contrainte concernant les encours est levée ; ceux-si sont supposés illimités donc non contraints. En conséquence, le temps de cycle optimal, noté TC, est imposé par la machine critique i.e. goulot. Les gammes linéarisées associées au problème ci-dessus sont présentées en Figure IV-2. Cette phase de la résolution du problème de PFM est détaillée par Korbaa(1998). Il s'agit donc d'une étape préliminaire qui est supposée résolue en amont du problème traité par notre approche de résolution.

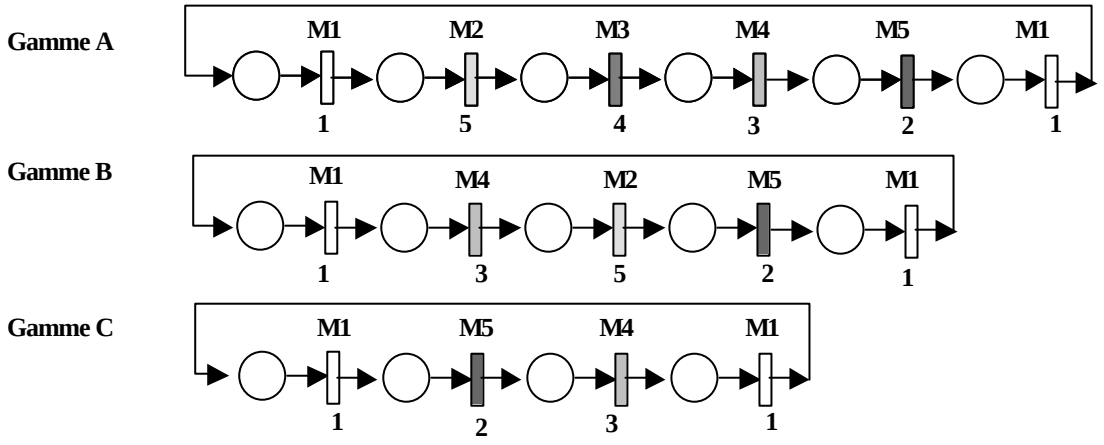


Figure IV-2 Gammes linéarisées

Etape 2 La seconde étape a pour objectif de minimiser les encours EC, au taux de production optimal. Il s'agit de déterminer un ordonnancement minimisant les encours de toutes les gammes. Dans le cadre des hypothèses simplificatrices adoptées, une borne minimale de l'encours d'une gamme est définie par le nombre entier de temps de cycle (TC) nécessaire à la réalisation de toutes les opérations de cette gamme en ignorant les contraintes de ressources. Une borne minimum B_{EC} de l'encours global de toutes les gammes est donc obtenue en cumulant les bornes de chaque gamme. Cette borne vaut donc (Korbaa, 1998):

$$B_{EC} = \sum_{i=1}^{n} \frac{\sum_{j=1}^{mi} p_{ij}}{TC}$$

IV.1.3 Formulation du problème d'ordonnancement

La formalisation mathématique complète du problème est donnée en Annexe 5. Une description simplifiée est décrite ci-dessous à partir de l'exemple précédent.

Après linéarisation des gammes, l'exemple considéré est reformulé comme suit :

$m_{A,1}=M_1$; $m_{A,2}=M_2$; $m_{A,3}=M_3$; $m_{A,4}=M_4$; $m_{A,5}=M_5$; $m_{A,6}=M_1$;
$m_{B,1}=M_1$; $m_{B,2}=M_4$; $m_{B,3}=M_2$; $m_{B,4}=M_5$, $m_{B,5}=M_1$;
$m_{C,1}=M_1$; $m_{C,2}=M_5$; $m_{C,3}=M_4$; $m_{C,4}=M_1$;

avec les temps d'exécutions suivants :

p_{A1}=1ut; p_{A2}=5ut; p_{A3}=4ut; p_{A4}=3ut; p_{A5}=2ut; p_{A6}=1ut;
p_{B1}=1ut; p_{B2}=3ut; p_{B3}=5ut; p_{B4}=2ut; p_{B5}=1ut;
p_{C1}=1ut; p_{C2}=2ut; p_{C3}=3ut ; p_{C4}=1ut;

L'objectif est la minimisation de l'encours global. Compte tenu des hypothèses fixées, cet encours global du système est défini par le nombre de palettes en circulation dans le système de production. Plus précisément, il est calculé comme étant la somme du nombre de temps de cycle nécessaire pour la réalisation de chaque gamme :

$$Min\ EC \text{ avec } EC = \sum_{i=1}^{n} ECi = \sum \left\lceil \frac{Ci - ti1}{TC} \right\rceil$$

Le Tableau IV-1 et le Tableau IV-2 reportent respectivement les charges cumulées des machines et les temps cumulés des opérations de chaque gamme. Il apparaît que la machine M_2 est la machine critique qui définit le temps de cycle optimal :TC=10 ut. La borne inférieure de l'encours est calculé comme suit: $BEc = \left\lceil \frac{16}{10} \right\rceil + \left\lceil \frac{12}{10} \right\rceil + \left\lceil \frac{7}{10} \right\rceil = 5$.

Machines	M_1	M_2	M_3	M_4	M_5
Charges	6	10	3	9	6

Tableau IV-1 Charges cumulées des machines

Gamme	A	B	C
Temps total	16	12	7

Tableau IV-2 Cumul des temps opératoires des gammes

Un exemple de calcul de l'encours de la gamme B est donné en Figure IV-3 :

$EC_B = (C_B - t_{B1})/TC = \lceil 17/10 \rceil = 2$.

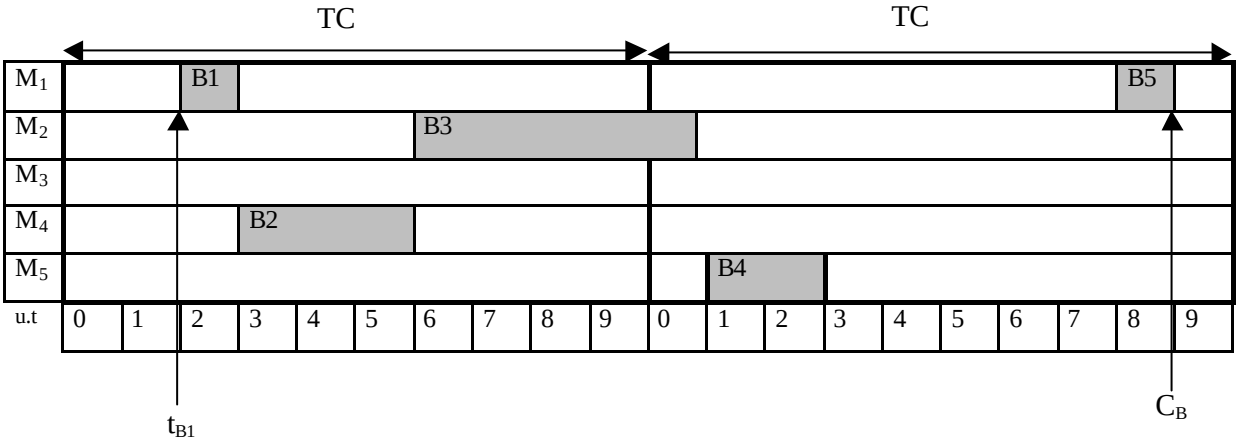


Figure IV-3 Exemple d'encours de la gamme B

Le travail présenté ci-dessous est focalisé sur la seconde étape de résolution qui calcule un ordonnancement respectant le temps de cycle optimal (TC) tout en minimisant les encours EC à l'aide d'un algorithme génétique. Il s'agit donc de manière classique de déterminer l'ordre de passage des pièces sur les machines en respectant les contraintes de précédence de la gamme de fabrication et leur date d'entrée sur ces machines. Il s'agit également de déterminer le nombre de palettes minimum nécessaires.

IV.2 Approche de résolution

IV.2.1 Codage génétique

Le principe du codage adopté, les opérateurs génétiques ainsi que le mécanisme d'élitisme sont décrits ci-dessous.

IV.2.1.1 Méthode de codage

Notre approche de résolution est basée sur un codage direct utilisant une représentation discrète du temps (Hsu, 2002). Chaque chromosome définit l'ordonnancement des opérations sur une période de temps restreinte au temps de cycle du système, noté TC. Ce chromosome est représenté par une matrice dont chaque élément correspond à une opération d'une gamme particulière. Chaque ligne de cette matrice est associée à une machine particulière et contient l'ensemble des opérations qui lui est affecté. Les colonnes numérotées spécifient le numéro d'ordre des opérations. Les opérations de chaque ligne sont donc ordonnées par ordre croissant de début de réalisation. Plus précisément, chaque élément de la matrice caractérise une opération et contient deux types d'informations : d'une part, la gamme et le numéro d'ordre de cette opération, d'autre part, sa date de début de réalisation. Chaque opération placée vérifie les contraintes de disponibilité de la machine considérée. Soit $o_{i',j'}$ l'opération qui précède et $o_{i'',j''}$ l'opération qui suit o_j sur cette machine, l'opération o_j vérifie donc les contraintes suivantes $C_{i',j'} < t_{ij}$ et $t_{ij}+p_{ij} < t_{i'',j''}$. A titre d'exemple "A2,5" correspond à la seconde opération de la gamme A qui démarre à l'instant 5. La Figure IV-4 présente le codage d'une solution possible au problème décrit en Figure IV-2

	1	2	3	4	5	6
M_1	A1,0	C1,1	B1,2	A6,7	B5,8	C4,9
M_2	B3,0	A2,5				
M_3	A3,0					
M_4	A4,0	B2,3	C3,6			
M_5	A5,1	C2,3	B4,5			

Figure IV-4 Représentation d'un chromosome

Les avantages de ce codage sont les suivants :

- Chaque chromosome représente un ordonnancement faisable sans qu'il y ait de conflit de ressources possible car les contraintes relatives aux opérations qui précèdent et qui suivent chaque opération sur une machine sont vérifiées par construction des solutions initiales.

- Le diagramme de Gantt de la Figure IV-5 peut être déduit directement à partir du chromosome et des durées des opérations.

- Les contraintes de précédence imposées par la gamme peuvent être ignorées dans le codage adopté. En effet dans le cas ou l'une des contraintes de précédence est violée dans l'intervalle du temps de cycle représenté par le chromosome, les opérations concernées correspondent dans ce cas à différentes occurrences de pièces en cours de fabrication. Ainsi par exemple en Figure IV-5, deux occurrences de pièces, notées q-1 et q, sont en cours de réalisation dans le cycle i. Les occurrences d'indice q des opérations de la gamme B, correspondent à la fabrication de la $q^{ième}$ pièce de type B et nécessitent deux temps de cycle pour être achevées (cycle i et cycle i+1) : les opérations B1 et B2, notées respectivement <B1,q> et <B2,q> sont réalisées au cycle i et les opérations B3 jusqu'à B5, notées <B3,q> à <B5,q>, sont réalisées au cycle i+1.

Il convient de noter que :

- les périodes d'inactivité des machines ne sont pas représentées explicitement dans le codage mais apparaissent en noir dans le diagramme de Gantt associé au chromosome.

- la ligne correspondante à la machine M_2 ne comporte aucune période d'inactivité puisque cette machine est la machine critique.

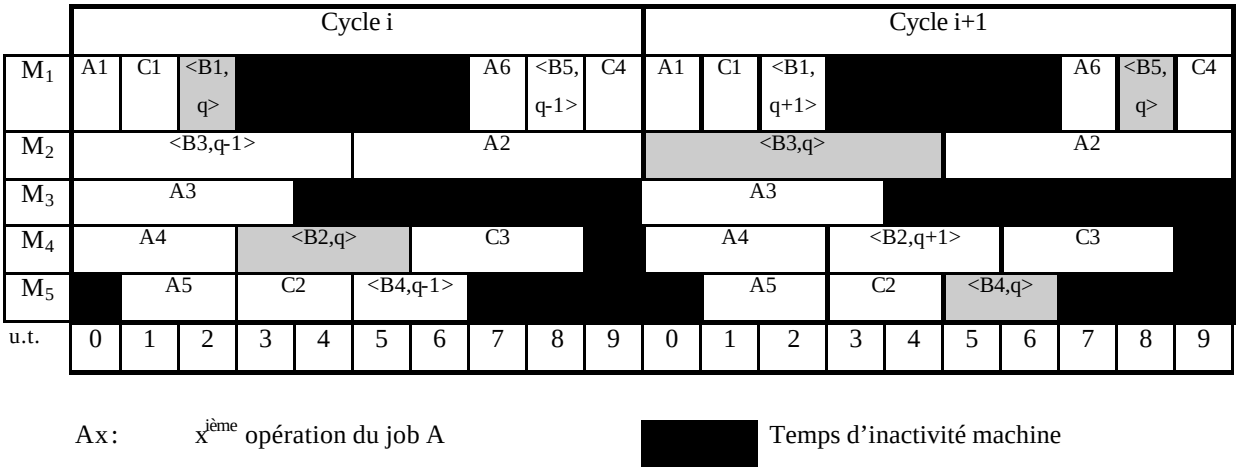


Figure IV-5 Diagramme de Gantt associé au chromosome de la Figure IV-4

IV.2.1.2 Codage des chevauchements.

Une opération en chevauchement débute dans un cycle et se termine dans le cycle suivant (Korbaa, 1998). Le chevauchement des opérations permet dans certains cas de diminuer l'encours (EC). L'exemple de la Figure IV-6 représente un ordonnancement sans chevauchement qui engendre l'encours suivant : EC_A= 3; EC_B= 3; EC_C=1 d'ou $EC_{Global} = \sum_{i=1}^{n} EC_i = 7$.

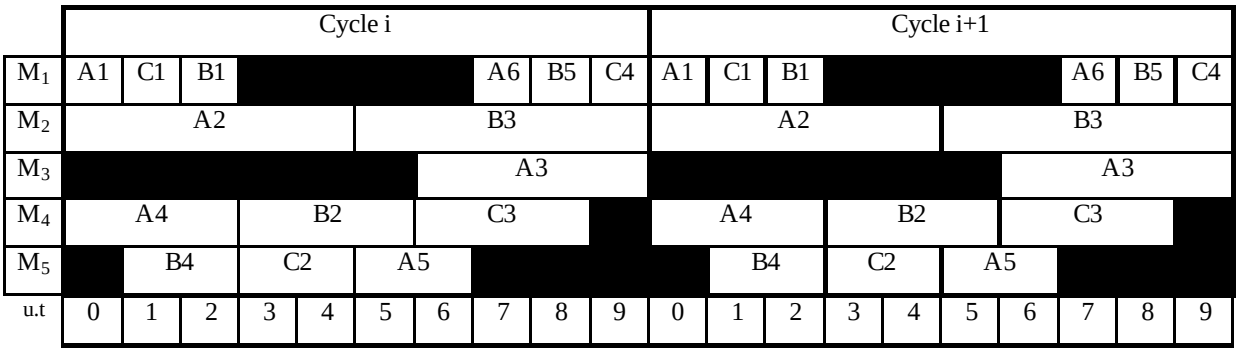


Figure IV-6 Ordonnancement sans chevauchement

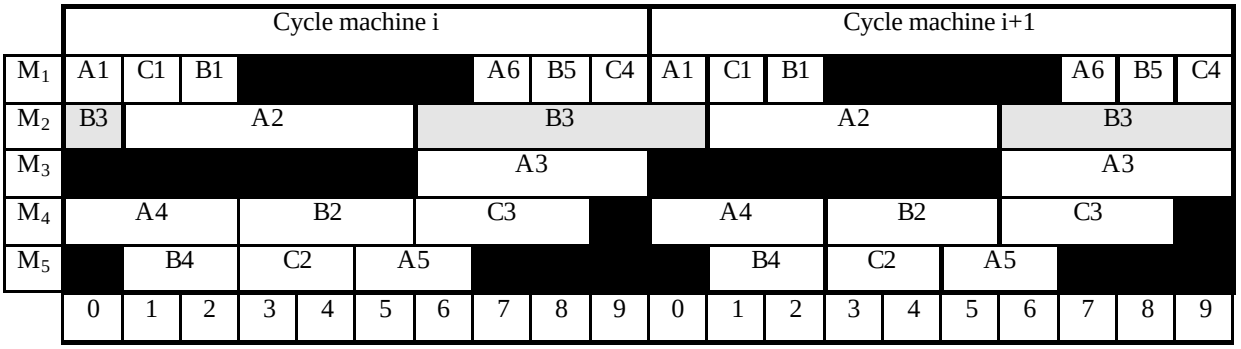


Figure IV-7 Ordonnancement avec chevauchement

La Figure IV-7 montre que si l'opération B3 chevauche deux cycles, alors l'encours des gammes A et B est diminué d'une unité. $EC_A = 2$; $EC_B = 2$; $EC_C = 1$; $EC_{Global} = 5$.

Dans le codage proposé, une opération est caractérisée par sa date de début sur la machine. En conséquence, ce codage permet de représenter les opérations en chevauchement sans modification. A titre d'exemple, l'occurrence q de l'opération B3, notée <B3,q> débute à l'instant 6 du cycle i et se termine à l'instant 1 du cycle i+1. L'ordonnancement de la Figure IV-8 concernant la machine M_2 est codé simplement par le chromosome de la Figure IV-9.

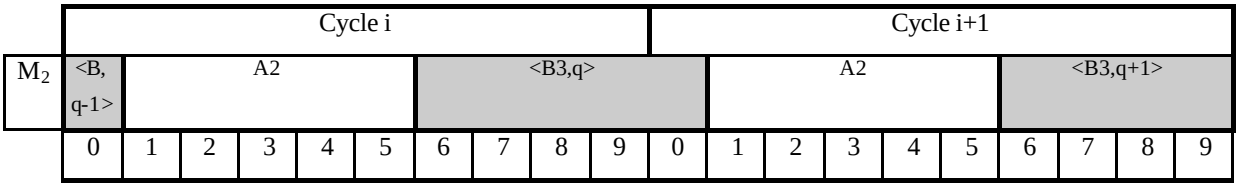


Figure IV-8 Représentation des chevauchements :diagramme de Gantt

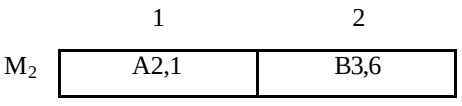


Figure IV-9 Représentation des chevauchements : chromosome correspondant

En résumé, le codage proposé permet d'une part de représenter les opérations en chevauchement sur deux cycles et permet d'autre part d'assurer une couverture totale de l'espace de recherche.

IV.2.2 Opérateurs génétiques

IV.2.2.1 Croisement

Un simple croisement à deux points a été utilisé en imposant de plus que la zone entre les points de coupure soit réduite à un seul élément. Cette zone sélectionnée aléatoirement correspond dans le codage proposé à une ligne horizontale dans la matrice associée. L'opérateur échange l'ordonnancement de la machine correspondante entre les deux parents. Il convient de noter qu'il n'est pas nécessaire de recalculer ou de modifier les dates de début des opérations des descendants obtenus. En effet l'échange effectué porte sur une ligne complète de chaque chromosome et grâce au codage adopté, les contraintes de précédence ne nécessitent pas d'être respectées. Cet opérateur préserve donc l'ordonnancement des opérations sur les machines des chromosomes parents. Seul l'encours global est affecté par cet opérateur génétique.

	1	2	3	4	5	6
M_1	A1,0	C1,1	B1,2	A6,7	B5,8	C4,9
M_2	B3,0	A2,5				
M_3	A3,0					
M_4	A4,0	B2,3	C3,6			
M_5	A5,1	C2,3	B4,5			

P_1

	1	2	3	4	5	6
M_1	A1,0	C1,1	B1,2	A6,7	B5,8	C4,9
M_2	A2,0	B3,5				
M_3	A3,3					
M_4	B2,0	A4,3	C3,6			
M_5	A5,2	C2,4	B4,6			

P_2

Figure IV-10 Chromosomes parents

	1	2	3	4	5	6
M_1	A1,0	C1,1	B1,2	A6,7	B5,8	C4,9
M_2	B3,0	A2,5				
M_3	A3,0					
M_4	B2,0	A4,3	C3,6			
M_5	A5,1	C2,3	B4,5			

E_1

	1	2	3	4	5	6
M_1	A1,0	C1,1	B1,2	A6,7	B5,8	C4,9
M_2	A2,0	B3,5				
M_3	A3,3					
M_4	A4,0	B2,3	C3,6			
M_5	A5,2	C2,4	B4,6			

E_2

Figure IV-11 Descendants produits par croisement

L'exemple des Figure IV-10 et Figure IV-11 décrit l'opérateur de croisement entre deux parents P_1 et P_2 , le point de coupure étant la machine M_3. Les descendants notés E_1 (respectivement E_2) sont obtenus en fusionnant l'ordonnancement associé aux machines M_1 à M_2 du chromosome parent P_1 (respectivement P_2) avec l'ordonnancement associé aux machines M_3 à M_5 de P_2 (respectivement dans P_1).

Ce croisement est décrit dans l'Algorithme IV-1

Algorithme IV-1
Sélectionner aléatoirement deux chromosomes parents P_1 et P_2
Sélectionner aléatoirement une machine M_k, k∈ {1,.., m}
Créer un enfant E_1(respectivement E_2) en fusionnant :
-l'ordonnancement des machines M_1 à M_{k-1} de P_1 (resp. P_1)
-l'ordonnancement de la machine M_k à M_m de P_2 (resp. P_1)

Algorithme IV-1 Croisement

IV.2.2.2 Mutation.

Elle vise à explorer l'espace de recherche. Plus précisément, l'opérateur de mutation choisi est une mutation par insertion. Son principe consiste tout d'abord à sélectionner aléatoirement une machine M_k d'un chromosome, puis à sélectionner aléatoirement une opération o_{ij} de cette machine ainsi qu'une date de début de réalisation *d*. La mutation consiste à modifier l'ordonnancement de la machine M_k en imposant que la date de début de l'opération o_{ij} soit *d.* Cette modification nécessite le déplacement de certaines opérations à proximité de la date d. Les dates de début de ces opérations nécessitent une modification.

Les Figure IV-12 et Figure IV-13 présentent une mutation sur la machine M_5 du chromosome de la Figure IV-7 qui entraîne une modification de la date de début de l'opération A5 à l'instant 2. Les opérations B4 et C2 sont déplacées afin d'éviter les conflits sur la machine M_5 : l'opération B4 est avancée d'une unité de temps et l'opération C2 est retardée d'une unité de temps. L'ordonnancement des autres machines est inchangé.

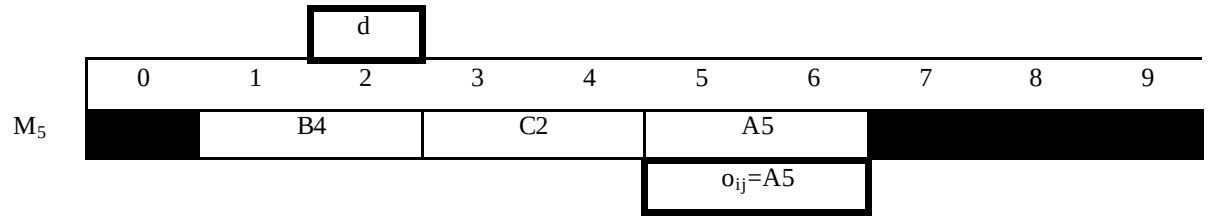


Figure IV-12 Chromosome avant mutation

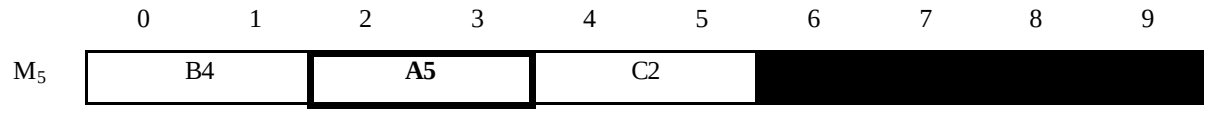


Figure IV-13 Chromosome après mutation

Cet opérateur de mutation est décrit dans l'Algorithme IV-2

IV.2.2.3 Population initiale

Cette population est générée par un tirage aléatoire uniforme. Elle comporte des individus en chevauchement sur deux cycles de production.

Algorithme IV-2
Sélectionner aléatoirement : - un chromosome - une machine M_k sur ce chromosome, k i {1,..,m} - une opération $o_{i,j}$ ordonnancée sur la machine M_k - une date d dans l'intervalle {0, .., TC–1} Sauvegarder l'opération $o_{i,j}$ et la supprimer de la machine M_k Si un conflit est détecté lors du déplacement de $o_{i,j}$ dans sa nouvelle position (date de début d), décaler circulairement vers l'avant les opérations en conflit avec la date de début de $o_{i,j}$ (par exemple B4 est avancée d'une unité de temps dans l'exemple ci-dessus) Si un conflit est détecté lors du déplacement de $o_{i,j}$ dans sa nouvelle position (date de fin d), décaler circulairement vers l'arrière les opérations en conflit avec la date de fin de $o_{i,j}$ (par exemple C2 est retardé d'une unité de temps dans l'exemple ci-dessus) Placer l'opération $o_{i,j}$ dans sa nouvelle position (date de début d)

Algorithme IV-2 Mutation

IV.2.2.4 Elitisme

Une caractéristique importante du problème étudié réside dans le fait que le fitness (i. e. encours) est une grandeur entière dont le domaine de variation est peu important (i.e. faible dispersion). A titre indicatif, le fitness du plus mauvais individu est généralement inférieur à deux fois la valeur optimale ou quasi-optimale trouvée. De plus, il est apparu dans de nombreuses exécutions de l'algorithme que la population contenait beaucoup d'individus ayant des génotypes identiques (i.e. clones). En conséquence, un mécanisme d'élitisme a été implémenté pour améliorer la performance de l'algorithme. Ce mécanisme est décrit dans l'Algorithme IV-3.

Algorithme IV-3
Soit :
-P_n (taille λ) la population de la génération n
-P'_{n+1} (taille λ) la population intermédiaire obtenue après sélection et mutation
-P_{n+1} (size λ) la nouvelle population de la génération n+1
Fusionner les populations P_n et P'_{n+1} dans P' (taille 2λ)
Trier tous les individus de P' par fitness décroissant
Effacer tous les clones P'
Sélectionner au plus β ($\beta < \lambda$) de P_{n+1} à partir des meilleures individus de P'
Compléter P_{n+1} avec des individus choisis aléatoirement

Algorithme IV-3 Elitisme

Ce mécanisme d'élitisme vise à maintenir la diversité de la population au cours des générations. En effet, tous les clones sont détruits. Par ailleurs la diversité est contrôlée par un paramètre β qui représente le nombre des meilleurs individus transférés dans la population finale de rang n+1. Il est à noter que β est inférieur à λ qui est la taille de la population ($\beta \leq \lambda$). Ce paramètre est augmenté de manière exponentielle au cours du processus générationnel. Au début de l'évolution (première génération) sa valeur est fixée à une constante β_0 et vaut λ pour la dernière génération

IV.3 Validation et résultats

Les résultats obtenus sont mentionnés dans le Tableau IV-4. Ceux-ci ont été obtenus avec le paramétrage de l'algorithme génétique du Tableau IV-3.

Paramètres	
Taille de la population	500
Probabilité croisement	0.7
Taux de mutation	0.3
Méthode de sélection	tournoi
Initialisation de la population	aléatoire
Nombre de générations	~5000 (variable selon le problème)

Tableau IV-3 Paramétrage de l'algorithme génétique

Le Tableau IV-4 mentionne les résultats obtenus sur un ensemble de huit problèmes tests

d'ordonnancement cyclique. Les instances correspondantes aux six premiers problèmes tests sont données en Annexe 6. Les deux dernières instances notées FT06 (Fisher, 1963) et LA04 (Lawrence, 1984) sont des instances classiques du job shop.

Ce tableau comporte en colonne, les informations successives suivantes :

- le nombre d'opérations du problème

- la valeur du temps de cycle en nombres d'unité de temps

- le nombre de solutions admissibles calculées en utilisant la formule décrite en Annexe 7. Cette information n'est qu'un indicateur relatif à la taille de l'espace de recherche mais ne constitue pas une preuve formelle de complexité du problème.

- la borne minimum théorique de chaque problème test calculée selon la formule du paragraphe IV.1.2.

- le résultat obtenu, exprimé en nombre de palettes, par la meilleure approche heuristique de résolution du problème lorsque celle-ci est connue.

- le meilleur résultat fourni par l'algorithme génétique.

- la solution optimale obtenue par programmation linéaire en nombres entiers à l'aide du solveur XPRESS-MP implémentant le modèle mathématique donné en Annexe 5.

Les principales conclusions de cette évaluation sont les suivantes :

- l'algorithme génétique égale les performances des meilleures heuristiques connues pour les six premiers problèmes.

- les solutions optimales sont atteintes par l'algorithme génétique dans six des huit problèmes tests à l'exception des problèmes de plus grande taille "FT06" et "LA04". Pour ces problèmes la performance de l'algorithme génétique n'est qu'à 20 % de la borne inférieure.

- en outre, la mise en œuvre d'une méthode de résolution exacte basée sur la programmation linéaire permet de montrer grâce aux problèmes (Ohl,1995) et (Korbaa,1998 instance a) que la borne inférieure n'est pas toujours atteignable.

A titre indicatif le temps de calcul de l'algorithme génétique sur un Pentium III 800 MHz sous Linux varie de une à soixante secondes selon les six premiers problèmes ; pour le problème FT06 ce temps est de l'ordre de cinquante heures.

Problèmes tests	Nombre d'opérations	Temps de cycle	Espace de recherche (*nombre de solutions admissibles*)	Borne minimum (*palettes*)	Meilleure heuristique (*palettes*)	Algorithme génétique (*palettes*)	Optimum : PLNE XPRESS-MP
Hillion,1987	15	8	16257024	5	5	5	5
Hillion,1988	9	6	10800	5	6	5	5
Valentin, 1994	13	11	79833600	5	5	5	5
Ohl, 1995	10	28	3.5E+10	4	5	5	5
Korbaa, 1998 (instance a)	23	24	8.6E+10	12	13	13	13
Korbaa, 1998 (instance b)	23	24	8.6E+10	9	9	9	9
Fisher, 1963 (FT06)	36	43	4.4E+38	7	NC[7]	8	7
Lawrence, 1984 (LA04)	50	537	3.9E+71	10	NC	12	NC

Tableau IV-4 Résultats comparatifs sur les problèmes tests d'ordonnancements cycliques

IV.4 Conclusion

Dans ce chapitre, nous avons proposé une méthode de résolution approchée basée sur un algorithme génétique pour le problème d'ordonnancement cyclique en Production Flexible Manufacturière. Cette résolution vient en aval d'une étape d'évaluation de performances prenant en compte la flexibilité des gammes.

L'algorithme génétique vise à minimiser l'encours du système de production, correspondant aux nombres de palettes utilisées, au taux de production maximum. Un codage direct et original ainsi que des opérateurs associés efficaces ont été proposés pour ce problème. L'intérêt essentiel de l'approche de résolution proposée est d'engendrer uniquement des solutions faisables et d'assurer une couverture totale de l'espace de recherche. La qualité des résultats obtenus sur un ensemble de problèmes tests d'ordonnancement cyclique permettent de valider en profondeur l'approche proposée.

Il est cependant nécessaire d'étudier en détail les aspects théoriques de la complexité de ce problème qui ont été négligés jusqu'à présent. De même, les performances de la méthode de résolution sont à améliorer. Certaines propositions sont développées dans le dernier chapitre relatif consacré aux perspectives.

[7] Non connu

V Job shop flexible multicritère

Ce chapitre aborde la résolution d'un problème d'ordonnancement flexible multicritère dans un contexte classique i.e. non cyclique. Cependant, contrairement au problème de PFM étudié au chapitre précédent dans lequel la flexibilité portait sur les gammes de fabrication, ce problème se caractérise par la flexibilité des machines.

Dans un premier temps, une introduction au job shop flexible est donnée. Ensuite l'approche de résolution évolutionniste multicritère basée sur la Pareto dominance est décrite (Bogarin, 2002) et une validation préliminaire de ces travaux est présentée. Enfin une approche d'aide à la comparaison des méthodes d'optimisation multicritères est décrite.

V.1 Définitions et hypothèses

Le job shop flexible est une extension du modèle job shop classique. Sa particularité essentielle réside dans le fait que plusieurs machines sont potentiellement capables de réaliser un sous-ensemble des opérations,. Plus précisément l'opération o_{ij} est associée à l'ensemble non vide $M_{ij} \subseteq \{M_1,..,M_m\}$ contenant toutes les machines pouvant effectuer cette opération.

Deux formulations très proches de ce problème existent dans la littérature :

- la première (Jurisch, 1992) considère que le temps d'une opération est identique pour toutes les machines capables de réaliser cette opération. Ce problème se note JMPM[8] $m|n|C_{max}$, dans le cas de la minimisation du *makespan*, pour un problème à m machines et n jobs.

- la seconde formulation (Brandimarte, 1993) comporte des durées opératoires qui peuvent être différentes pour une même opération réalisée sur des machines distinctes : le problème est alors appelé job shop flexible.

La résolution du job shop flexible revient à traiter deux problèmes d'optimisation combinatoire en minimisant un ou plusieurs critères: d'une part, un problème d'affectation de chaque opération sur une machine particulière, et d'autre part, un problème d'ordonnancement des opérations sur les machines. Parmi les applications de ce modèle, citons à titre d'exemple, l'affectation de ressources et l'ordonnancement des opérations dans un terminal portuaire intermodal (Zaffalon, 1998).

Nous abordons la résolution du job shop flexible tri-critère (Kacem, 2003) dans le cas particulier d'une flexibilité totale des machines i.e. toutes les machines sont capables d'effectuer toutes les opérations. Les trois critères considérés sont :

- **C1** : minimiser le temps total de production, pour réaliser l'ensemble des travaux i.e. critère $C_{max,}$ ou *makespan*.

- **C2** : minimiser la charge totale des machines qui représente le temps de travail cumulé pour toutes les machines.

- **C3** : minimiser la charge de la machine critique, c'est à dire de la machine qui a la plus forte charge de travail

[8] Multi-Purpose Machines

Le problème JMPM $m|n|C_{max}$ est NP-difficile dans le cas général car il est une extension du problème de job shop. Il est en particulier NP-difficile au sens fort dans les cas JMPM $2||C_{max}$ et JMPM $3|p_{ij}=1|C_{max}$ ainsi que NP-difficile au sens ordinaire dans le cas JMPM $|n=3|C_{max}$ (Jurisch, 1992). Par ailleurs, il a été démontré que pour un problème à k critères, il suffit que l'optimisation d'un critère soit un problème NP-difficile pour que le problème multicritère associé le soit également (Chen, 1993). En conséquence, le problème multicritère que nous étudions est lui-même NP-difficile. Ceci justifie le choix d'une méthode de résolution approchée telle que par exemple l'approche tabou mise en œuvre par Mastrolilli (2000) pour la minimisation du *makespan*.

Le but fixé est la recherche d'ordonnancements qui minimisent l'ensemble de ces trois critères. Le premier critère traduit un délai de réponse au client ; le second critère correspond à une volonté d'alléger les coûts d'utilisation des ressources de l'atelier en diminuant le temps total d'occupation des machines ; le dernier critère correspond à la nécessité de décharger la machine goulot. Ces critères peuvent être contradictoires. En effet l'équilibrage de charge –i.e. minimiser la charge critique, peut aller à l'encontre de la performance du délai de réponse au client.

L'objectif de nos travaux est donc de traiter conjointement les problèmes d'ordonnancement et d'affectation afin de trouver un ensemble d'ordonnancements admissibles et choisir parmi cet ensemble le ou les bons ordonnancements i.e. les plus proches possibles de l'optimum au sens des trois critères utilisés.

V.2 Approche de résolution et codage

L'approche de résolution proposée s'appuie sur l'algorithme multicritère NSGA-II (Deb, 2000) dont le choix et la description ont été donnés au deuxième chapitre. Il convient maintenant de spécifier le codage du problème sous forme de chromosomes ainsi que les opérateurs génétiques de croisement et de mutation.

V.2.1 Codage et opérateur de croisement

La méthode de résolution est basée sur le codage utilisé par (Meshgouni, 1999) pour ce problème. Plus précisément, chaque chromosome est représenté par une matrice dans laquelle chaque ligne représente la gamme de fabrication ou job et chaque colonne correspond à une opération. Chaque élément de cette matrice est composé de deux informations : le numéro de la machine affectée à l'exécution de l'opération et la date de début d'exécution de cette opération sur la machine qui lui a été affectée. Ce codage respecte par définition les contraintes de précédence de la gamme de fabrication. Les contraintes de ressources sont quant à elles respectées lors du calcul des dates de début et de fin d'exécution de chaque opération pour l'évaluation du *makespan*. Un exemple de codage est présenté en Figure V-1. Cet exemple utilise un problème simplifié composé de trois jobs et cinq machines ; les temps d'exécutions correspondants sont donnés en Annexe 2.

J_1	M1,0	**M2,4**	M4,9
J_2	**M2,0**	M5,4	**M2,9**
J_3	M5,0	M3,2	

Chrom0

Figure V-1 Représentation d'un chromosome ayant pour évaluation : C1=14 C2=26 C3=14

La méthode présentée ci-dessus assure un codage direct du sous-problème d'affectation des opérations aux machines. Le sous-problème d'ordonnancement est quant à lui codé de façon indirecte en considérant que le chromosome exprime implicitement une priorité parmi les ordres de fabrication : la première ligne de la matrice, respectivement la dernière, contient le job le plus prioritaire, respectivement le moins prioritaire. Cette priorité peut être par exemple être formulée par le responsable de fabrication. Elle intervient en cas de conflit d'utilisation d'une même machine par plusieurs jobs. Plus précisément dans l'exemple de la Figure V-1, les priorités sont les suivantes : J_1, puis J_2 et J_3. Il est à noter que ce codage n'assure pas une couverture totale de l'espace de recherche.

Un algorithme simple de parcours des opérations qui respecte les priorités ci-dessus mentionnées permet donc de calculer directement l'ordonnancement puis d'en déduire les critères choisis.

Croisement Le codage utilisé permet d'utiliser un croisement à deux points, sans qu'il y ait des solutions infaisables. Le point de coupure des chromosomes correspond dans cette représentation à une ligne horizontale dans la matrice associée ; l'opérateur échange alors les parties supérieures et inférieures à cette séparation entre les deux parents, puis recalcule les dates d'exécution des opérations. Il est à noter que cet opérateur conserve donc l'affectation des opérations aux machines issue des chromosomes parents.

V.2.2 Heuristiques de mutation

L'opérateur de mutation des algorithmes génétiques a pour but général d'explorer l'espace de recherche. L'opérateur classique " swap " qui est noté H_{swap} par la suite, consiste à échanger les machines affectées à deux opérations différentes sélectionnées aléatoirement sur un chromosome. Il ne se base pas sur une heuristique et sert simplement de référence pour les comparaisons ultérieures. Dans l'objectif d'améliorer les performances de l'algorithme de résolution, plusieurs opérateurs mettant en œuvre une recherche locale ont été proposés et évalués.

Contrairement à cet opérateur aléatoire, le principe de la mutation dirigée développée dans notre approche de résolution consiste à équilibrer au mieux la charge des deux machines de charges extrêmes. L'algorithme consiste à réaffecter une opération réalisée sur la machine la plus chargée, vers la machine la moins chargée (Bogarin, 2001). Cet algorithme de mutation est donné en Algorithme V-1 et comporte quatre étapes notées I, I, II et IV. Il se base sur l'utilisation d'une heuristique notée H, pour sélectionner une opération parmi l'ensemble des opérations réalisées sur la machine la plus chargée. L'évaluation d'une heuristique H pour un chromosome 'X' est notée f_H ('X').

Algorithme V-1

Début

Choisir aléatoirement un chromosome X **(I)**

Calculer les charges des machines **(II)**

Déterminer M_{max} (respectivement M_{min}) la machine la plus chargée (respectivement moins) **(III)**

Déterminer OM_{max} : ensemble des opérations o_i affectées à M_{max} **(IV)**

Pour i **de** 1 **à** Card(OM_{max}) **(V)**

Générer X_i par réaffectation virtuelle de o_i à M_{min}

Evaluer l'heuristique $f_H(X_i)$

FinPour

Opérer la réaffectation minimisant $f_H(X_i)$ **(VI)**

Fin

Algorithme V-1 Algorithme génétique de mutation dirigée

Heuristiques de mutation

Nous avons proposé plusieurs heuristiques, détaillées ci-dessous, pour effectuer cette mutation. Pour chacune de ces heuristiques, l'ensemble de toutes les opérations affectées à la machine la plus chargée sont potentiellement réaffectables.

Mutation par charge : "H_{Charge}**".** Cet opérateur défini dans (Mesghouni, 1999), consiste à choisir aléatoirement une opération de la machine la plus chargée et à réaffecter celle-ci vers la machine la moins chargée. Il s'agit donc d'une version simplifiée de l'algorithme de mutation présenté en Algorithme V-1, puisqu'il n'y a pas de réaffectation virtuelle dans ce cas.

Mutation par écart type : "$H_{Ecart\ Type}$**".** Le but de cet opérateur est de minimiser les écarts entre les charges des machines. La mutation consiste donc à choisir la réaffectation d'une opération de M_{max} qui minimise l'écart type des charges de toutes les machines.

Mutation par *Makespan*: " H_{Cmax}**".** Le but de cet opérateur est de minimiser le *makespan* de l'ordonnancement. La mutation consiste donc à choisir la réaffectation d'une opération de M_{max} qui minimise ce *makespan*.

Mutation par Sommation : "$H_{Sommation}$**".** Cette heuristique porte, à la différence des précédentes, sur l'ensemble des trois critères considérés. Elle consiste donc à choisir la réaffectation d'une opération de M_{max} qui minimise la somme des valeurs des trois critères C1, C2 et C3.

Mutation par Agrégation : " $H_{Agrégation}$ **".** Cet opérateur est proche du précédent. Il consiste à réaffecter une opération de M_{max} qui minimise la sommation normalisée des trois critères considérés.

Exemples. Des exemples de mutation sont présentés ci-dessous en s'appuyant sur le problème simplifié donné en Annexe 2. Le calcul des charges (**II**) des machines correspondant au chromosome avant mutation de la Figure V-1, est mentionné dans le Tableau V-1. Celui-ci montre que la machine M_2 est la plus chargée et M_1 la moins chargée. En conséquence, les trois opérations réalisées sur la machine M_2, apparaissant en gras sur la Figure V-1 peuvent potentiellement être réaffectées à la machine M_1. Ces trois réaffectations virtuelles concernent les opérations $o_{1,2}$, $o_{2,1}$, $o_{2,3}$ et sont notées respectivement: R1, R2, R3.

Machines	M_1	M_2	M_3	M_4	M_5
Charges	1	14	2	4	5

Tableau V-1. Charge des machines avant mutation

Considérons à titre d'exemple l'heuristique $H_{Agrégation}$. Le choix de l'opération réaffectée(**VI**) pour cette heuristique est basée sur la somme normalisée des trois critères C1, C2 et C3 qui est reportée dans le Tableau V-2. La réaffectation R1 opérée sur le chromosome "Chrom0" de la Figure V-1 engendrerait le chromosome " Chrom2 " dont l'évaluation par l'heuristique "$H_{Agrégation}$ " est: $f_{Agrégation}$(chrom2) = 12/16+24/30+9/10=2.45; les valeurs (16,30,10) étant les plus grandes valeurs courantes des critères respectifs C1, C2 et C3. Les deux autres réaffectations possibles R2 et R3 génèrent des chromosomes dont l'évaluation est de moins bonne qualité. Pour cette heuristique $H_{Agrégation}$, la meilleure réaffectation est donc R1, qui engendre le nouveau chromosome " Chrom2 " de la Figure V-2. L'opération qui a été effectivement réaffectée apparaît en gris dans les chromosomes résultants de la Figure V-2. Il faut noter que la valeur de l'heuristique " $H_{Agrégation}$ " après mutation est bien inférieure à la valeur avant mutation qui vaut : 14/16+26/30+14/14 = 2.617

	R1	R2	R3
C1 (max 16)	12	11	16
C2 (max 30)	24	23	30
C3 (max 10)	9	10	10
$f_{Agrégation}$	2.45	2.454	3.00

Tableau V-2 Evaluation de "H $_{Agrégation}$" après les réaffectations R1, R2 et R3 opérées sur Chrom0

Les chromosomes résultants des autres heuristiques de mutation sont mentionnés en Figure V-2.

J_1	M1,0	**M2,4**	M4,9
J_2	**M2,0**	M5,4	**M1,7**
J_3	M5,0	M3,2	

Chrom1 H_{Charge}

J_1	M1,0	**M1,1**	M4, 4
J_2	**M2,0**	M5,4	**M2,7**
J_3	M5,0	M3,2	

Chrom2 $H_{Ecart\ Type}$ et $H_{Agrégation}$

J_1	M1,0	**M2,1**	M4,6
J_2	**M1,1**	M5,2	**M2,6**
J_3	M5,0	M3,2	

Chrom3 H_{Cmax} et $H_{Sommation}$

Figure V-2 Chromosomes après mutation

Le Tableau V-3 reporte les valeurs des charges des machines après mutation en fonction des différentes heuristiques.

Charges	H_{Charge}	H_{Cmax} $H_{Sommation}$	$H_{Ecart\ Type}$ $H_{Agrégation}$
M_1	10	2	4
M_2	9	10	9
M_3	2	2	2
M_4	4	4	4
M_5	5	5	5

Tableau V-3 Charge des machines après mutation

V.3 Validation et résultats

Cette évaluation est détaillée en trois parties qui concernent successivement la description de l'application cible, puis la présentation des résultats obtenus et enfin leur évaluation.

V.3.1 Application

Le problème test utilisé, totalement flexible, se note JMPM $10|n=10|C_{max}$ (Bogarin, 2001 ; Kacem,

2001) . Il est donné en Annexe 3 et comporte dix jobs (n=10); chaque job comporte trois opérations et toutes les opérations peuvent être exécutées par l'ensemble des dix machines existantes (JMPM10), avec des temps d'exécution différents. Il s'agit donc d'ordonnancer les trente opérations sur les dix machines disponibles en respectant les gammes de fabrication.

L'algorithme fournit pour chaque exécution, un ensemble de solutions non dominées. Le temps d'exécution est de l'ordre de soixante quinze secondes sur un Pentium III à 500 Mhz sous Linux.

Les résultats présentés au paragraphe suivant ont été obtenus avec le paramétrage du Tableau V-4.

Paramètres	
Taille de la population	350
Probabilité croisement	0.9
Taux de mutation	0.4
Méthode de sélection	tournoi
Nombre d'exécutions	10
Initialisation de la population	aléatoire
Nombre de générations	120

Tableau V-4 Paramétrage de l'algorithme génétique

Il convient de remarquer que l'utilisation de l'élitisme a une incidence sur le paramétrage. Il favorise une convergence plus rapide et nécessite une population de taille importante.

V.3.2 Résultats

Le Tableau V-5 donne les *fitness* des solutions Pareto optimales obtenues pour chacune des méthodes de mutation à l'issue des dix exécutions. Ces solutions constituent une approximation du front Pareto optimal $\wp$ ö *. L'ordonnancement complet associé à la solution [7, 45, 5] est donné à titre indicatif en Annexe 4.

Heuristique	Meilleurs *fitness* [**C1,C2,C3**]
H_{Charge}	[8, 42, 6] – [8, 43, 5] – [7, 44, 6] – [7, 45, 5]
H_{Swap}	[8, 41, 7] – [8, 43, 6] – [7, 44, 6]
H_{Cmax}	[8, 42, 6]
$H_{Sommation}$	[7, 42, 6] – [8, 41, 7] – [8, 42, 5]
$H_{Ecart\ Type}$	[7, 46, 6] – [8, 43, 5] – [8, 42, 7]
$H_{Agrégation}$	[7, 42, 6] – [8, 44, 5] – [7, 45, 5] – [8, 41, 7]

Tableau V-5 *Fitness* des solutions Pareto optimales

Le Tableau V-6 quant à lui, présente les *fitness* des solutions Pareto optimales de l'ensemble de toutes les heuristiques envisagées. Il convient de remarquer que les heuristiques $H_{Sommation}$ et $H_{Agrégation}$ sont celles qui permettent d'atteindre un plus grand nombre de solutions Pareto optimales. Chacune d'elles permet d'atteindre trois solutions différentes de l'ensemble $\wp$ ö * approché.

Fitness [**C1,C2,C3**]	Heuristiques
[7, 45, 5]	H_{Charge} ,$H_{Agrégation}$
[7, 42, 6]	$H_{Agrégation}$,$H_{Sommation}$
[8, 41, 7]	$H_{Agrégation}$,$H_{Sommation}$,H_{Swap}
[8, 42, 5]	$H_{Sommation}$

Tableau V-6 Solutions Pareto optimales de l'ensemble de toutes les heuristiques

V.3.3 Etudes comparatives

L'évaluation de performances et la comparaison des approches dans le domaine de l'optimisation multicritère demeurent une question ouverte. Pour le problème étudié, le front Pareto optimal étant inconnu, trois types de comparaisons ont été effectués.

V.3.3.1 Premier comparatif.

Ce comparatif consiste à mesurer les résultats obtenus par rapport à une borne minorante obtenue indépendamment pour chaque critère. Ces bornes sont les suivantes :

- la borne minorante de C_{max} (BC1) est obtenue en supprimant les contraintes de ressources, et en recherchant le plus grand C_{max} formé des temps de réalisation les plus petits de chaque opération : $BC1=\max_i(\sum_j \min_k(p_{i,j,k}))$

- la borne minorante de la charge totale (BC2) correspond à la somme des temps de réalisation les plus petits pour chaque machine : $BC2 = (\sum_{i,j} \min_k(p_{i,j,k}))$

- la charge critique ne peut pas être inférieure au rapport de la borne minorante de la charge totale sur le nombre de machines dans le cas le plus défavorable où les machines sont supposées de capacité équivalente. $BC3 = \lceil BC2/m \rceil$ avec m le nombre de machines.

Les valeurs de ces bornes pour le problème traité sont mentionnées dans le Tableau V-7. Les heuristiques $H_{Sommation}$ et $H_{Agrégation}$ permettent d'atteindre chacune de ces bornes minorantes indépendamment sur des solutions Pareto optimales distinctes. Cette performance notable peut être attribuée à la taille modérée du problème test utilisé et à la nature même des données qui le constituent.

V.3.3.2 Deuxième comparatif.

Il s'agit d'une évaluation de performances relative, basée sur la comparaison avec une approche

évolutionniste non Pareto utilisant l'agrégation des différents critères pour la résolution de ce même problème (Kacem, 2001). Notons que la solution trouvée (7 45 5) reportée dans le Tableau V-7 par cette approche alternative est incluse dans l'ensemble Pareto optimal obtenu.

	C1	**C2**	**C3**
Bornes inférieures	BC1: 7	BC2: 41	BC3: 5
Meilleure solution selon (Kacem,2001)	7	45	5

Tableau V-7 Bornes minorantes et meilleures solutions selon (Kacem, 2001)

V.3.3.3 Troisième comparatif.

Ce dernier comparatif se base sur la métrique proposée par (Zitzler, 1999) pour la comparaison de populations de solutions Pareto optimales. Cette métrique également appelée taux de couverture de deux ensembles A et B i.e. populations d'individus, est définie par le ratio C(A,B) à valeurs dans l'intervalle [0,1]: $C(A,B)=(n_{BA}/n_B)$

- n_{BA} : nombres d'individus de B dominés par A

- n_B : nombre d'individus de B

Ce rapport représente la proportion d'individus d'une population B dominés par les individus d'une population A

Le Tableau V-8 est donc formé des ratio de dominance -i.e. C(A,B), entre tous les couples de populations Pareto optimales issues de toutes les heuristiques de mutation proposées. Plus précisément, l'ensemble des solutions trouvées au cours des dix exécutions associées à chaque heuristique a été pris en compte.

B \ A	H_{Swap}	H_{Charge}	H_{Cmax}	$H_{Sommation}$	$H_{EcartType}$	$H_{Agrégation}$
H_{Cmax}	0.859	0.9	0.9	1	0.779	1
$H_{EcartType}$	0.779	0.976	0.786	1	0.9	0.976
H_{Swap}	0.793	0.802	0.802	0.9	0.702	0.9
H_{Charge}	0.768	0.875	0.812	0.994	0.807	0.993
$H_{Agrégation}$	0.634	0.814	0.634	0.876	0.78	0.796
$H_{Sommation}$	0.470	0.542	0.48	0.8	0.5	0.678

Tableau V-8 Ratio C(A,B)[9] de comparaison des heuristiques de mutation

Cette métrique qui est non symétrique est interprétée de la manière suivante :

- si le ratio C(A,B) tend vers " un ", tous les individus de B sont dominés ou égaux à ceux de A

[9] B : une ligne, A : une colonne

- inversement, si C(A,B) tend vers " zéro ", aucun individu de B n'est dominé par ceux de A.

Le Tableau V-8 peut donc être interprété comme suit:

- L'heuristique " $H_{Sommation}$ " possède le ratio de dominance le plus faible vis-à-vis des autres heuristiques.

- Toutes les heuristiques possèdent un ratio de dominance faible par rapport à l'heuristique " $H_{Agrégation}$ ".

- Enfin, les heuristiques " $H_{Ecart\ Type}$ " et "H_{Cmax}" sont presque totalement dominées par les heuristiques " H_{Charge} ", " $H_{Sommation}$ " et " $H_{Agrégation}$ ".

Ces trois comparatifs convergent vers un classement des heuristiques qui met en évidence l'intérêt d'un opérateur de mutation dirigé intégrant l'ensemble des trois critères considérés. Les deux heuristiques les plus performantes sont l'agrégation et la sommation. Inversement, les heuristiques basées sur l'écart type ainsi que sur le *makespan* sont les moins performantes. L'heuristique fondée sur la charge seule, fournit quant à elle une performance intermédiaire.

En outre, les opérateurs "$H_{Agrégation}$" et "$H_{Sommation}$" atteignent les bornes minorantes de chacun des critères mais avec des solutions Pareto optimales distinctes. Cette comparaison permet donc de valider l'intérêt d'une approche de mutation dirigée par des heuristiques pour ce problème.

V.4 Méthode d'aide à l'évaluation de performances basée sur le surclassement

Le problème étudié dans ce chapitre relève de la problématique de la décision à partir de critères multiples. Les méthodes existantes dans ce domaine, appartiennent à l'une des trois approches opérationnelles suivantes :

- l'approche du critère unique de synthèse, évacuant toute incomparabilité : entre deux alternatives a et b, le décideur peut toujours faire un choix ou un classement.

- l'approche du surclassement de synthèse, acceptant l'incomparabilité.

- les approches du jugement local interactif avec itération essai-erreur.

La seconde approche sur laquelle nous nous focalisons est connue sous l'appellation de l' "Ecole Française" et propose des méthodes d'agrégation partielles ou de surclassement dont la caractéristique essentielle est d'accepter l'intransitivité et l'incomparabilité de deux actions ou solutions possibles[10]. Nous proposons dans ce paragraphe d'exploiter ce type d'approche dans le cadre de l'évaluation de performances des méthodes de résolution du job shop flexible.

En effet, l'évaluation de performances des méthodes multicritères demeure une question ouverte dont la difficulté est inhérente à la notion même d'optimisation dans un contexte multicritère. Différentes métriques de comparaisons existent selon que le front Pareto optimal est connu ou non (Hansen, 1998). Lorsqu'il est inconnu, des mesures de performances ont été proposées telles que la métrique de Zitztler (2000) utilisée au paragraphe précédent ou celle de Talbi (1999) qui mesure l'efficacité relative. Néanmoins, ces mesures ne permettent pas de

[10] L'usage du terme « action » employé dans le domaine de l'aide à la décision est ici conservé

synthétiser totalement et de clore l'analyse de l'efficacité de différentes méthodes.

Pour cette raison, une méthode de comparaison des populations Pareto optimales est décrite ci-dessous en vue d'évaluer l'efficacité des différentes heuristiques de mutation proposées au paragraphe précédent. Son objectif est de classer ces différentes populations et donc les heuristiques qui en sont la source, de la moins bonne à la meilleure en s'inspirant du principe des méthodes de surclassement dont les plus connues sont ELECTRE, PROMETHEE et leurs dérivés (Scharlig, 1985 ; 1996). Ces méthodes élaborent dans un premier temps des relations binaires appelées relations de surclassement en comparant deux à deux toutes les paires d'actions ou possibilités, en se basant sur une généralisation de la notion de critère et en intégrant les préférences du décideur. Dans un deuxième temps l'exploitation de cette relation est réalisée en vue d'aider à formuler une recommandation qui puisse apporter une réponse au problème de décision.

Plus précisément, la méthode PROMETHEE opère en trois étapes :

Etape 1: La préférence d'une action a_1 sur une action a_2 est introduite par le décideur, sous la forme d'une valeur numérique représentative du degré de validité de l'hypothèse considérant que a est meilleur que b, compte tenu de l'évaluation de ces deux actions .

Etape 2 : Le cumul des préférences pour toutes les actions b_i vis-à-vis de a_1, avec i @ 1, engendre un flux sortant représentant la puissance de l'action a_1 sur l'ensemble des autres. De même, un flux entrant représentatif de la faiblesse de a_1 vis-à-vis des autres actions est calculé.

Etape 3 : Un graphe de surclassement est construit entre toutes les actions possibles, puis analysé en vue de classer les actions possibles.

Par analogie avec cette approche de surclassement, une méthode de classement des populations Pareto optimales obtenues par les différentes heuristiques est proposée. Cette méthode s'inspire des deux dernières étapes de PROMETHEE sans toutefois appliquer la méthode dans sa globalité.

Le point de départ de cette méthode est le Tableau V-9, contenant les ratios de dominance des heuristiques de mutation qui sont calculés selon la métrique de Zitzler (2000). Chaque élément de ce tableau situé en colonne A et ligne B, noté C(A,B), représente le ratio de dominance de la population A par rapport à la population B. La méthode exploite la remarque suivante: la somme normalisée de tous les ratios de dominance de la ligne B, fournit une évaluation de la dominance globale de toutes les populations sur la population B qui est analogue à un flux entrant ; de même, la somme normalisée de tous les ratios de dominance de la colonne A, fournit donc une évaluation de la dominance globale de la population A sur toutes les autres populations qui est analogue à un flux sortant. Il est à noter que pour effectuer ces cumuls, la dominance d'une population par rapport à elle-même n'a pas été considérée car jugée non significative. A titre d'exemple, la puissance de l'heuristique H_{Cmax} sur les autres heuristiques est obtenue après division par six du cumul des ratios de dominance de toutes les lignes de la colonne H_{Cmax}, (i.e. 3.5144/6) dans le Tableau V-9

Le Tableau V-10 de surclassement des méthodes est alors élaboré en calculant le bilan des flux de préférence entrant et sortant pour chaque méthode. Le Tableau V-10 montre les résultats de cette comparaison. La différence entre le flux sortant et le flux entrant est calculé dans la colonne du bilan. Ce flux de préférence exprime le classement des méthodes, de la moins bonne H_{Cmax} à la meilleure $H_{Sommation}$.

B \ A	H_{Swap}	H_{Charge}	H_{Cmax}	$H_{Sommation}$	$H_{EcartType}$	$H_{Agrégation}$	**Faiblesse** de B
H_{Cmax}			0.9				
$H_{EcartType}$	0.779	0.976	0.786	1	0.9	0.976	0.75
H_{Swap}			0.802				
H_{Charge}			0.812				
$H_{Agrégation}$			0.634				
$H_{Sommation}$			0.48				
Puissance de A			0.59				

Tableau V-9 Principe de calcul des flux sortants "puissance" et sortants "faiblesse" à partir du tableau des Ratio C(A,B)

	Flux Sortant (Puissance)	Flux Entrant (Faiblesse)	Bilan	Classement
$H_{Sommation}$	0.79	0.44	0.35	1
$H_{Agrégation}$	0.76	0.62	0.14	2
H_{Charge}	0.67	0.73	-0.06	3
H_{Swap}	0.58	0.68	-0.1	4
$H_{Ecart\ Type}$	0.59	0.75	-0.16	5
H_{Cmax}	0.59	0.76	-0.17	6

Tableau V-10 Surclassement des heuristiques de mutation pour le job shop flexible

Les conclusions de cette étude de surclassement des heuristiques de mutation, sont globalement cohérentes avec l'interprétation formulée au paragraphe précédent et en faveur des heuristiques qui intègrent les trois critères considérés : C_1, C_2 et C_3. Elle confirme donc l'intérêt d'exploiter le principe de surclassement pour évaluer de façon systématique les performances des méthodes évolutionnistes multicritères. Il est indispensable de rappeler que le classement obtenu est totalement dépendant du choix de la mesure de comparaison des populations Pareto optimales. Davantage d'expérimentations sont nécessaires pour valider complètement cette démarche et la généraliser à l'interprétation des résultats fournis par les mesures de distance entre populations Pareto optimales.

V.5 Conclusion

Une approche évolutionniste multicritère basée sur la dominance Pareto a été utilisée pour résoudre un problème d'ordonnancement de type job shop flexible à trois critères. Ce problème se caractérise par une flexibilité totale au niveau des machines, capables de réaliser toutes les opérations avec des temps opératoires différents. La méthode de résolution proposée est basée sur la mise en œuvre de l'algorithme NSGA-II. Un

ensemble d'opérateurs, appelés heuristiques de mutation, effectuant une recherche locale orientée vers les meilleures solutions a été proposé. Ces heuristiques ont été évaluées d'une part par comparaison relative à l'aide d'une métrique spécifique à l'évaluation de performance des méthodes multicritères et d'autre part, par rapport à des bornes sur chaque critère indépendamment. Les résultats obtenus sur un problème test de taille modérée permettent de valider l'approche proposée. De surcroît une méthode d'aide à l'évaluation de performance des approches d'optimisation multicritères basé sur le principe du surclassement a été proposée et validée sur un problème de taille limitée.

Ce travail constitue une première approche de résolution évolutionniste multicritère du problème. Néanmoins, il paraît évident que la validation de notre approche est insuffisante quant à la taille et à la variété des instances traitées. Le choix d'aborder ce problème multicritère par l'approche Pareto est discutable en raison des inconvénients inhérents à cette méthode. Certains de ces points sont abordés dans le dernier chapitre relatif aux perspectives.

VI Perspectives

Le travail présenté dans les chapitres précédents étend le paradigme des algorithmes génétiques au domaine de l'ordonnancement cyclique et flexible. Ce dernier chapitre est consacré aux perspectives ouvertes par ces différents travaux, qui sont présentées selon trois axes : le premier porte sur les perspectives communes aux trois problèmes d'ordonnancement étudiés ; le second axe de développement concerne des voies de perspectives propres à chacun de ces problèmes; enfin, le dernier axe de développement vise à élargir les problématiques étudiées au domaine des transports.

VI.1 Perspectives communes aux problématiques cycliques et flexibles

Trois sujets potentiels d'approfondissement de ces recherches sont présentés ci-dessous et concernent en commun le job shop cyclique à contraintes linéaires, le job shop flexible et l'ordonnancement cyclique en production flexible manufacturière (PFM).

VI.1.1 Analyse comparative de performance et bornes minorantes

Les problèmes cycliques n'ont pas été étudiés de manière aussi intensive que le job shop classique. Ils représentent un large champ potentiel d'application pour les méthodes approchées. Il paraît donc utile de disposer de benchmarks spécifiques permettant d'évaluer les performances de ces méthodes approchées. Dans cet objectif, un jeu de problèmes tests a été proposé pour le job shop cyclique à contraintes linaires. Il s'agit d'un premier pas permettant d'effectuer cette validation. Il reste néanmoins à comparer les résultats obtenus par d'autres approches heuristiques ou méta-heuristiques sur ces problèmes et à étendre la taille des jeux de tests.

Concernant le job shop flexible, il est nécessaire de valider en profondeur le modèle de résolution proposé à l'aide de jeux de tests de taille plus importante que ceux utilisés dans le cinquième chapitre. Les problèmes tests de référence sont les suivants : (Barnes 1996 ; Brandimarte 1993 ; Dauzère-Pérès 1997 ; Hurink 1994).

La recherche de bornes inférieures de bonne qualité est impérative pour évaluer précisément la performance des approches génétiques proposées. Les bornes relatives au job shop flexible ainsi qu'au problème d'ordonnancement cyclique en PFM sont de bonne qualité. En revanche, pour le problème du job shop cyclique à contraintes linéaires, il convient d'améliorer les bornes minorantes présentées au troisième chapitre. Dans ce domaine, une possibilité consisterait à explorer l'intérêt éventuel d'une approche basée sur la méthode de relaxation Lagrangienne.

VI.1.2 Analyse de paysages

La notion d'analyse de paysage ou *fitness landscape* (Reeves, 2003b) a été introduite par les biologistes avec l'idée de représenter l'espace génotypique d'une espèce vivante sous la forme d'un paysage dans lequel les génotypes qui se ressemblent sont voisins. Les *fitness* associés aux différents génotypes créent une surface appelée paysage du *fitness*. Une métaphore géographique permet alors de qualifier le modèle de l'espace génotypique sous forme de vallée, pic, plateau ou arête.

Ce concept a été exploité en recherche opérationnelle. Il vise à caractériser la structure de l'espace de recherche de certains problèmes d'optimisation et apprécier la difficulté relative des différentes instances de problème. Cette structure est fortement dépendante de la représentation du problème et de l'opérateur de voisinage utilisé. Des analyses de paysage ont porté sur des instances de problèmes standards tels que le problème d'affectation quadratique (Bachelet, 1999) et également sur les problèmes d'ordonnancement tel que le job shop (Mattfeld, 1995). Une meilleure connaissance de l'espace de recherche permet alors de développer des méta-heuristiques adaptées et plus performantes. Cette connaissance conduit généralement à l'élaboration de méthodes hybrides ou encore à la modification de la fonction *fitness* de façon à atténuer la difficulté de parcours des zones délicates telles que les plateaux (Talbi, 2000).

Une approche pratique de l'analyse de paysage peut être opérée comme suit : un grand nombre de descentes basées sur le gradient est lancé à partir de solutions de départ uniformément réparties sur l'espace. Ces recherches locales explorent à chaque pas un sous-ensemble de tous les voisins du sommet courant et s'arrêtent dès lors qu'aucun voisin n'améliore le *fitness*. Au cours de cette descente, plusieurs paramètres ou indicateurs sont mesurés ; une valeur moyenne de ces paramètres peut alors être calculée. Une classification des problèmes tests peut alors être effectuée en fonction des valeurs de ces paramètres de manière à étudier la topologie de la distribution des optima locaux.

Nous proposons dans un premier temps de considérer pour les instances de problèmes cycliques les paramètres classiques utilisés pour le job shop standard :

- la longueur de marche définie comme le nombre de pas maximum effectués par la méthode de descente. Ce paramètre donne une information sur la rugosité du paysage : une longueur de marche courte décrit un paysage rugueux et inversement.

- l'entropie qui mesure la dispersion d'une population dans l'espace de recherche. Elle vaut un pour une population initiale uniformément répartie et zéro pour une population ayant convergé vers un optimum global. La mesure significative est la variation d'entropie entre la population initiale formée des points de départ des descentes aléatoires et la population finale.

- le diamètre qui est complémentaire à la dispersion et mesure la concentration de la population. Il est défini comme la distance maximum entre deux solutions de cette population. De même que précédemment, la mesure pertinente est la variation de ce diamètre.

Il convient donc de mettre en œuvre ce type d'analyse sur les problèmes que nous étudions. Une étude est en cours sur les instances difficiles du problème d'ordonnancement cyclique en PFM. Elle permettra d'une part de mieux appréhender les difficultés rencontrées par l'algorithme génétique et d'autre part d'améliorer son efficacité.

VI.1.3 Paramétrage et adaptation de l'algorithme génétique

Le paramétrage demeure une difficulté importante de la mise en œuvre d'un algorithme évolutionniste Elle constitue un frein notamment pour le développement de ce type de méthodes dans un contexte industriel. Face à cette difficulté deux approches peuvent être envisagées.

La première est une approche expérimentale ou pragmatique. Elle utilise la méthode des plans d'expérience, afin de structurer la démarche de réalisation des campagnes d'essais de paramétrage de l'algorithme génétique. La

mise en œuvre de tests statistiques permet alors une comparaison efficace de l'effet des différents paramétrages de l'algorithme génétique. Cette démarche a notamment été adoptée pour la mise en œuvre d'un algorithme génétique optimisant l'impact des opérations de changement d'outils sur le rendement d'une ligne de production de masse mono-produit (Cavory, 2001). Notons que cette approche a également été utilisée pour le paramétrage d'un algorithme génétique dédié à la résolution d'un problème de routage en contexte dynamique[11], qui est mentionné au paragraphe VI.3.2.

La seconde approche possible est issue des travaux sur l'adaptation des algorithmes évolutionnistes. De nombreuses voies de recherches existent dans ce domaine (Schoenauer, 2001). En effet, l'adaptation peut revêtir différentes formes et peut potentiellement concerner les différents composants de l'algorithme évolutionniste ; citons à titre indicatif le choix de représentation du problème, les opérateurs ou leurs probabilités d'application. Les modalités de changement sont elles aussi nombreuses selon qu'il s'agit de critères d'adaptation fixes, statistiques ou aléatoires.

Il paraît donc raisonnable de se focaliser dans un premier temps sur l'adaptation du taux de mutation en utilisant une règle simple telle que la règle du 1/5 de Rechenberg (1973) pour diminuer les difficultés actuelles liées au paramétrage des algorithmes de résolution proposés. Dans une étape ultérieure, plusieurs scénarios d'adaptation sont à explorer dans l'objectif de simplifier leur mise en œuvre.

VI.2 Perspectives spécifiques aux problèmes traités

Chaque problème étudié au cours de ce travail peut être approfondi comme mentionné ci-dessous.

VI.2.1 Job shop cyclique à contraintes linéaires

Une extension[12] possible de ces travaux concerne le problème connexe dans lequel les solutions recherchées sont périodiques. En effet, bien que la dominance des solutions périodiques ne soit pas démontrée pour ce problème, ce type de solutions présente l'intérêt de pouvoir utiliser un codage direct plus efficace et de s'affranchir de la simulation au niveau de l'algorithme génétique. L'idée à développer consiste à exploiter le codage utilisé en PFM dont l'efficacité a été prouvée en s'appuyant sur :

- l'expansion du graphe linéaire sous forme d'un graphe à contraintes de précédences uniformes
- la détermination de la machine critique pour obtenir le temps de cycle optimal
- la recherche d'un motif d'ordonnancement périodique sur un temps de cycle.

De plus, ce type de codage assurerait une couverture de la totalité de l'espace de recherche, ce qui n'est pas le cas dans l'approche proposée au troisième chapitre.
Concernant ce dernier problème périodique, deux points précis sont approfondis :

[11] thèse en cours de H. Housroum

[12] article soumis à RAIRO en 2004

- une modélisation en Programmation Linéaire en Nombres Entiers est proposée. Celle-ci permet, outre une spécification précise du problème, de résoudre de manière exacte des instances de taille réduite et d'apprécier par comparaison les performances de l'algorithme génétique.
- une seconde approche de résolution approchée basée sur une méthode originale couplant les réseaux de neurones[13] et la relaxation Lagrangienne est à l'étude. Elle permettra également de mener à bien des comparaisons de performances et de proposer des approches hybrides exploitant les points forts de ces deux techniques.

VI.2.2 Ordonnancement cyclique en Production Flexible Manufacturière

Il convient tout d'abord d'approfondir, du point de vue théorique, l'étude de la complexité de ce problème. En effet, comme c'est généralement le cas pour de nombreux problèmes issus d'applications industrielles, la preuve théorique permettant de déclarer que ce problème est NP-difficile n'a pas été apportée, bien que la complexité des méthodes de résolution proposées soit elle-même exponentielle.

Par ailleurs, il est utile d'améliorer notre méthode de résolution approchée de ce problème, en hybridant notre algorithme génétique avec un opérateur de recherche local.

VI.2.3 Job shop flexible multicritère

Une voie potentielle à explorer concerne l'utilisation des méthodes d'aide à la décision pour étendre les travaux dans le domaine de l'optimisation évolutionniste multicritère. Ceux-ci ont à ce jour essentiellement porté sur des approches a posteriori dans lesquelles l'étape de décision intervient après l'étape de recherche effectuée par l'algorithme évolutionniste. Ces approches utilisent majoritairement l'optimalité Pareto. Cette notion est intrinsèquement limitée et pose deux problèmes particuliers: le choix de solutions parmi l'ensemble des solutions du front Pareto, surtout dans le cas ou celui-ci contient un grand nombre de solutions ; l'absence de compromis de la Pareto dominance qui ne tient pas compte de la proximité des solutions vis-à-vis du front Pareto. Pour remédier à cela, des travaux récents visent à intégrer la notion de préférence dans l'optimisation évolutionniste soit dans une approche interactive soit dans une approche a priori dans l'objectif d'intégrer les notions de compromis propres aux approches de surclassement dans la recherche de solutions (Coello, 2000). Le problème du job shop flexible constitue une cible naturelle pour poursuivre les investigations dans ce domaine.

VI.3 Perspectives dans le domaine des transports

Cette dernière partie concerne les travaux que nous souhaitons développer dans le domaine de la modélisation et l'optimisation des systèmes de transport. Deux raisons essentielles nous conduisent à orienter les recherches dans cette direction : premièrement, les projets régionaux du "Contrat de Plan Etat Région" (CPER 2000-2006) auxquels participe le laboratoire (LGI2A) sont axés sur la thématique "transport" ; deuxièmement les similitudes et les liens entre les problématiques d'ordonnancement et de routage sont importants (Beck, 2003). A titre indicatif, la résolution du VRP à plusieurs véhicules peut se décomposer en deux phases : une phase d'affectation des véhicules aux clients et une seconde phase d'ordonnancement des clients sur chaque véhicule.

Nous abordons ces problèmes dans le cadre général de la recherche opérationnelle. Le paragraphe suivant est focalisé sur l'étude d'un sous-problème particulier de routage de véhicules dans un contexte dynamique.

Les problèmes que nous avons ciblés concernent l'élaboration des tournées de véhicules aussi appelée VRP[14]. Ce problème-type académique constitue une réduction très forte des problèmes réels d'optimisation de tournées de véhicules. Il n'en demeure pas moins que du point de vue de l'étude de la complexité des problèmes, il appartient à la catégorie NP-difficile. Une présentation du problème VRP de base est donnée, avant d'introduire la variante du VRP dynamique sur laquelle nous avons choisi de nous focaliser.

VI.3.1 VRP conventionnel

Dans sa version canonique, le problème VRP qui est schématisé en Figure VI-1 se compose des éléments suivants :

- un ensemble de clients à livrer
- un dépôt central à partir duquel les livraisons sont effectuées
- un ensemble de véhicules identiques effectuant les livraisons chez les clients à partir du dépôt
- un réseau de communications ou routes à double sens ; ce réseau forme un graphe complètement connecté dans lequel les nœuds sont les clients plus le dépôt et les arêtes sont les routes.

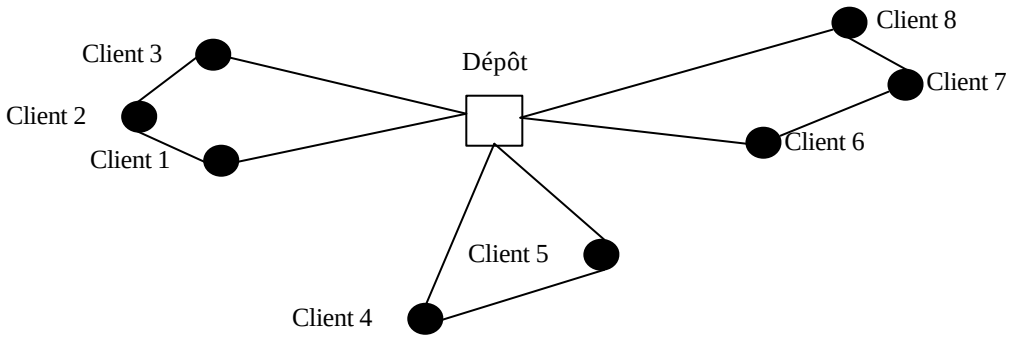


Figure VI-1 Problème de tournées de véhicules

Les contraintes associées à ce problème exigent que :

- tous les clients soient livrés

---

[13] Collaboration avec l'équipe PCRG de l'UCD en Irlande

[14] Vehicle Routing Problem

- chaque client soit visité une seule fois

- la demande cumulée de tous les clients d'une tournée n'excède pas la capacité du véhicule.

L'objectif est de minimiser le coût de ces tournées de véhicule, sachant que chaque trajet est associé à un coût élémentaire représentant une distance ou un coût de déplacement. Le problème bien connu du voyageur de commerce ou TSP[15] est un cas particulier du VRP constitué d'une seule tournée et sans contrainte de capacité.

Les variantes de ce problème de base sont nombreuses. Citons, parmi toutes les extensions existantes, les variantes suivantes :

- la possibilité d'associer au cours d'une même tournée des opérations de livraison et de ramassage

- l'existence de plusieurs dépôts : chaque client doit être livré à partir de l'un de ces dépôts

- l'obligation fixée par chaque client d'effectuer une livraison dans une fenêtre de temps déterminée par une borne inférieure et une borne supérieure. Cette version très étudiée est appelée VRPTW : *Vehicle Routing Problem with Time Windows*.

- le VRP dans un contexte dynamique décrit au paragraphe suivant.

L'intérêt de ce problème générique est qu'il permet de formaliser de nombreux problèmes de transport réels. Un premier exemple d'application dans le contexte des transports multi-modaux concerne la livraison et le ramassage des conteneurs ou caisses entre un centre de transbordement et les clients répartis autour de celui-ci, (Niérat, 1992). Un second exemple concerne le bus "à la demande" dans le contexte des transports urbains : les passagers sont ramassés et déposés à différents endroits, en fonction de leur demande. Ce problème est référencé sous le nom : *Dynamic Dial-A-Ride Problem* dans un contexte *Many-to-Many* i.e plusieurs lieux de ramassages et de dépôts. Plusieurs méthodes approchées heuristiques (Laporte, 1999) ou basées sur les algorithmes génétiques (Bräysy, 2001) (Prins, 2002), ont été utilisées pour résoudre différentes formes de problèmes statiques de routage de véhicules.

VI.3.2 VRP dynamique

Un état de l'art complet des problèmes de VRP dans un contexte dynamique et de leurs applications est donné dans (Larsen, 2000). La version dynamique du VRP se caractérise globalement par les deux assertions suivantes : l'ensemble des données pertinentes pour la définition du routage n'est pas disponible lorsque que le calcul d'élaboration de œlui-ci est lancé ; certaines données peuvent changer après que la route initiale ait été élaborée.

Le caractère dynamique de la problématique peut potentiellement concerner différents paramètres afférents au problème VRP statique tels que la demande du client, sa position, la fenêtre de temps, le nombre ou la capacité des véhicules. La forme la plus courante concerne l'apparition de nouvelles demandes clients au cours du temps.

Un exemple de scénario de VRP dynamique est donné en Figure VI-2. L'itinéraire pré-établi avant l'apparition de nouvelles demandes est tracé en trait fin continu. L'apparition de nouvelles requêtes identifiables par les cercles blancs engendre une modification des routes prévues. L'insertion du nouveau client dans la

[15] Traveling Salesman Problem

tournée engendre des modifications importantes dans le cas de la tournée de gauche ou mineures dans le cas de celle de droite.

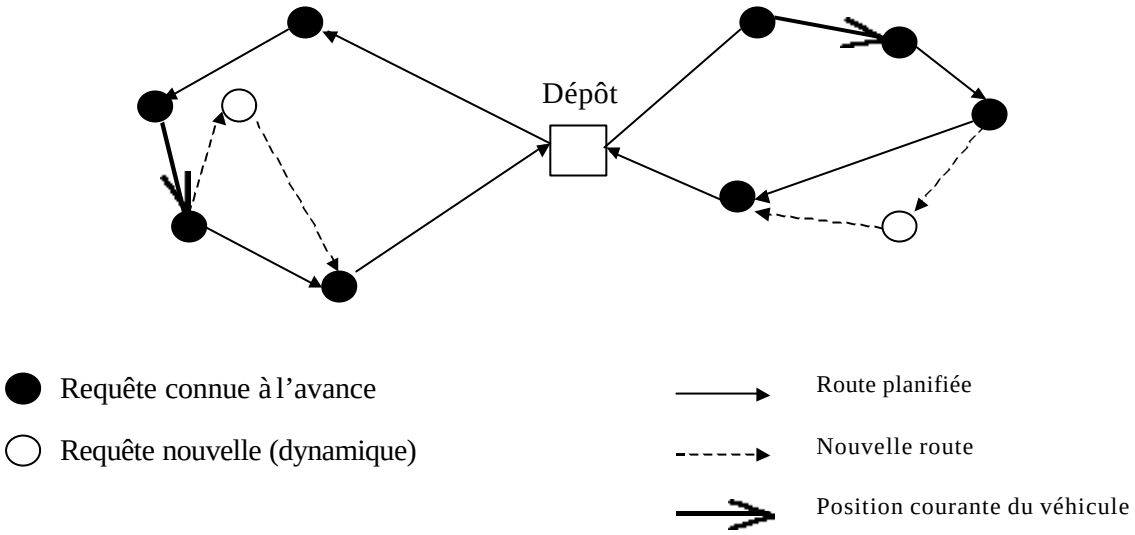


Figure VI-2 Exemple de scénario de VRP dynamique

VI.3.3 Une approche évolutionniste de résolution du VRP dynamique

Un aperçu des approches possibles de mise en œuvre d'un algorithme évolutionniste dans un contexte dynamique est donné ci-dessous ainsi qu'un résumé très succinct des travaux en cours.

Dans le domaine de l'optimisation en contexte dynamique il convient de citer l'état de l'art réalisé par Branke (2002) relatif aux algorithmes évolutionnistes et les travaux connexes dans le domaine de l'ordonnancement dynamique décrit par Artigues (2001). En premier lieu, le concept même d'optimisation dans un contexte dynamique est à préciser : la recherche de l'une des deux catégories de solutions est à privilégier:

- les solutions robustes : il s'agit de solutions qui ne sont pas les meilleures mais qui assurent, sans modification, de bonnes performances pour tous les scénarios d'évolution possibles à partir de l'état courant.

- les solutions flexibles : contrairement aux solutions robustes, les solutions flexibles peuvent être adaptées aisément aux changements de l'environnement. A titre d'exemple, dans le contexte du TSP, une relation d'ordre partielle sur l'ensemble des clients peut être appréhendée comme une forme de solutions flexibles.

Du point de vue des mécanismes évolutionnistes eux-mêmes, plusieurs possibilités existent pour s'adapter à un environnement changeant (Schoenauer, 2001). Trois d'entre elles sont présentées ci-dessous :

- la première utilise l'approche de résolution statique et redémarre l'algorithme évolutionniste à chaque changement de l'environnement. Cette approche la plus simpliste ne permet pas de capitaliser et d'exploiter les connaissances entre les différentes phases d'optimisation ; elle ne peut pas être utilisée si les contraintes de

temps réel sont fortes. En revanche, cette approche est obligatoire lorsque le changement de représentation est induit par un changement d'environnement.

- la seconde approche consiste à conserver et à faire évoluer une population sans redémarrer l'algorithme à chaque modification de l'environnement; dans cette approche, l'adaptation de la mutation est fréquemment utilisée. L'idée sous-jacente étant d'augmenter le taux de mutation après l'occurrence d'un changement d'environnement afin de favoriser la dispersion de la population ayant déjà partiellement convergé, avant de poursuivre le processus de recherche. Cette approche est appelée "hypermutation". De manière générale, les méthodes auto-adaptatives sont potentiellement exploitables pour atteindre cet objectif.

- enfin une troisième possibilité concerne l'adjonction d'une mémoire auxiliaire pour stocker les bonnes solutions et les ré-utiliser lorsque cela s'avère nécessaire. Cette mémoire peut être particulièrement utile et bénéfique lorsque le problème abordé comporte des changements d'environnement périodiques dans lesquels des situations peuvent se répéter.

Sur ce thème de l'optimisation dynamique de tournées de véhicules, deux problèmes cibles sont en cours d'étude.

- le premier problème abordé est le *Partially Dynamic Traveling Repair-man Problem* (PDTRP): un réparateur doit se déplacer pour effectuer des réparations chez un ensemble de clients dont certains sont connus dès le départ de la tournée (clients statiques) et d'autres arrivent en cours de journées (clients dynamiques). Il est à noter que, dans ce problème il n'y a pas de contraintes de fenêtres de temps imposées par les clients. Une comparaison de différentes stratégies de résolution est en cours en se basant sur les jeux de données proposés par Larsen (2000). Les stratégies étudiées sont notamment l'utilisation d'heuristiques de résolution du type "Plus Proche Voisin", ou la mise en œuvre d'une stratégie de recherche exhaustive sur un nombre restreint de clients connus ou encore, l'utilisation d'un algorithme évolutionniste fonctionnant sur le principe du redémarrage en statique à chaque modification de l'environnement.

- le second problème est le *Dynamic Vehicle Routing Problem Time Windows* (DVRPTW) (Kilby 1998): ce problème diffère du précédent par l'existence de contraintes de fenêtres de temps imposées par les clients à servir et également par l'utilisation d'une flotte de plusieurs véhicules. De même que dans le problème précédent, deux types de demandes existent : des demandes statiques et des demandes dynamiques. L'approche que nous avons développée[16] est basée sur la mise en œuvre d'un algorithme génétique utilisant un codage indirect de longueur variable, couplé à un simulateur de routage. Il s'agit d'une approche dans laquelle l'algorithme génétique calcule en permanence la meilleure route en prenant en compte l'ajout de nouveaux clients et la suppression des clients déjà servis. La propriété essentielle de cette approche est qu'elle ne nécessite pas de redémarrage complet de l'algorithme génétique à chaque changement de l'environnement. Cette approche a été validée expérimentalement et s'avère performante en comparaison de l'approche Tabou proposée par Gendreau (1999). En outre une méthode de réglage des paramètres de l'algorithme génétique basée sur les plans d'expérience a été développée afin de fournir une aide au réglage de l'algorithme génétique dans ce contexte dynamique.

[16] thèse en cours de H. Housroum

VI.3.4 Vers l'étude du VRP en contexte réel

Dans ce domaine, les travaux envisagés s'orientent vers l'étude de problèmes de routage en contexte réel qui comportent d'une part un Système d'Information Géographique contenant toutes les informations cartographiques des réseaux routiers, qui comportent d'autre part un système temps réel de positionnement GPS (*Global Positioning System*) couramment utilisé par les véhicules et qui disposent enfin de la remontée des informations en temps réel à partir de l'instrumentation des voies de circulation.

Cet objectif ultime doit être restreint en déclinant plusieurs extensions possibles des problèmes étudiés jusqu'à présent (DVRPTW et PDTRP) qui se limitent à une modélisation très simpliste de la réalité. Parmi les extensions possibles, il convient de citer la prise en compte des phénomènes d'aléas liés aux problèmes de congestion du trafic ou encore l'apparition des pannes de véhicules, et enfin l'imperfection des données relatives au modèle de routage étudié. Ces données imparfaites sont à titre d'exemple des temps de trajet variables au cours du temps ou de l'espace géographique parcouru, des temps de trajets ou des fenêtres de temps imprécis ou incertains modélisés sous la forme de sous-ensemble flous, voire encore la prise en compte de temps de trajets stochastiques.

Plus précisément deux sujets sont actuellement en cours d'étude avec pour objectif d'affiner les modèles de temps de trajet et remplacer le modèle de vitesse constante utilisé jusqu'à présent. Ces deux cibles ont été choisies car, à notre connaissance, les travaux dans le domaine du VRP avec des temps de trajets variables sont peu nombreux.

Une première étude porte sur le problème PDTRP et vise à intégrer un modèle de vitesse dépendant d'une part de la période temporelle de la journée et d'autre part du type de route empruntée en s'inspirant des travaux de (Ichoua, 2003) dans ce domaine. La journée est décomposée en trois intervalles de temps correspondant respectivement à l'heure de pointe matinale, le milieu de journée où la circulation est plus fluide et l'heure de pointe de fin de journée. Trois types de route sont distingués et correspondent respectivement à des artères à petite, moyenne et grande vitesse. Le calcul du temps de trajet entre deux clients s'effectue alors par une simulation temporelle qui prend en compte les caractéristiques précitées.

Le second sujet en cours d'étude concerne le DVRPTW et vise à compléter le distancier entre clients par un modèle de vitesse dépendant de l'espace géographique. Plus précisément, cet espace géographique est décomposé en rectangles de dimensions variables inclus les uns dans les autres sans se chevaucher. Chaque rectangle définit une zone géographique affectée d'une vitesse propre. L'ensemble de tous les rectangles est inclus dans le rectangle englobant l'espace total. Une structure de données de type R-Tree (*Rectangle-Tree*) est utilisée pour représenter cet espace géographique et permettre le calcul des temps de trajet. Ce modèle est en cours d'intégration dans la plate-forme logicielle de résolution du DVRPTW.

VI.4 Conclusion

Un ensemble de directions de travail a été esquissé dans ce dernier chapitre avec un niveau de profondeur variable et propre à chaque voie de perspective. Il est à noter que ces travaux se situent sur deux axes orthogonaux et complémentaires : ils englobent des questions d'ordre théorique ou méthodologique relatives aux méthodes d'optimisation et plus particulièrement aux méthodes évolutionnistes ; ils incluent également la

dimension applicative concernant le domaine des transports et plus particulièrement la famille des problèmes de type VRP dans un contexte dynamique.

Conclusion générale

Ce mémoire synthétise les travaux de recherche que j'ai effectués depuis 1996 au sein de l'Université d'Artois.

Ces travaux sont positionnés au croisement de deux champs thématiques. Le premier concerne l'amélioration de performance des systèmes de production discrets et de la résolution de problèmes d'ordonnancement de nature cyclique et flexible. Le second champ de recherche attenant à ces travaux est celui des algorithmes évolutionnistes qui appartient au domaine général des méthodes informatiques évolutives ou *Evolutionary Computing*. Le domaine des algorithmes évolutionnistes a connu un essor important ces dernières années consécutif au développement des algorithmes génétiques et possède un important potentiel de développement et d'application aux problématiques industrielles réelles. Il puise son inspiration dans le processus biologique de l'évolution naturelle dont la connaissance s'accélère de façon remarquable ces dernières années. Ces avancées dans le domaine des "sciences du vivant" représentent un exceptionnel potentiel de développement pour de nombreux secteurs des sciences de l'ingénieur.

Il n'en demeure pas moins, et ceci ressort des travaux qui ont été menés dans ce mémoire mais aussi dans des applications connexes, que l'exploitation de ces algorithmes comme outils d'optimisation ne peut pas se faire par une simple application du schéma canonique initial des algorithmes génétiques. En effet, pour espérer atteindre des performances de bonne qualité sur des instances de taille importante, il est impératif d'adapter et d'améliorer ce schéma canonique.

Pendant cette période, je me suis efforcé de développer une activité de recherche basée sur une conjonction constante d'activités théoriques et appliquées, liées parfois à des contrats industriels. En outre, la participation à des groupes de recherche régionaux et nationaux a permis à cette recherche de se placer dans un contexte de liaison permanente avec les travaux menés par d'autres équipes de recherche.

Je souhaite pouvoir continuer à développer mes activités universitaires dans ce contexte de travail enrichissant quoique non dépourvu de difficultés inhérentes à la jeunesse de la structure universitaire dans laquelle j'ai évolué pendant cette période.

Bibliographie

Artigues C., Roubellat F., Ordonnancement d'atelier en temps réel, Chapitre 13 dans Ordonnancement de la Production, eds Lopez P. et Roubellat F., Hermès Sciences, IC2 Productique, pages 259-293, 2001.

Benasser A., L'accessibilité dans les réseaux de Petri : une approche basée sur la programmation par contraintes, Thèse de Doctorat, Université de Lille 1, janvier 2000.

Bachelet V., Méta-heuristiques parallèles hybrides : application au problème d'affectation quadratique, Thèse Université de Lille, 1999.

Bäck T., Fogel D.B. and Michalewicz T., Evolutionary Computation 1 : basic algorithms and operators, Institute of physics publishing, 2000a.

Bäck T., Evolutionary Computation 1: basic algorithms and operators, T. Back, D.B. Fogel & T. Michalewicz (Eds), Introduction to evolutionary algorithms, chapitre 7, Institute of Physics Publishing, Bristol and Philadelphia, pages 27-38, 2000b.

Bäck T. , Fogel D.B. and Michalewicz T., Evolutionary Computation 2: advanced algorithms and operators, Institute of physics publishing, 2000c.

Backer K. R., Introduction to sequencing and scheduling, John Wiley & Sons, 1974.

Bagchi T. P., Multiobjective scheduling by genetic algorithms, Chapitre 5: job shop scheduling, Kluwer Academic Publishers, 1999.

Baptiste P., Bloch C., Varnier C., Ordonnancement des lignes de traitement de surface, Chapitre 9 dans Ordonnancement de la Production, eds Lopez P. et Roubellat F., Hermès Sciences, IC2 Productique, pages 259-293, 2001.

Barnes, J. W. and Chambers J. B., Flexible job shop scheduling by tabu search, Graduate program in operations research and industrial engineering. The University of Texas at Austin. Technical report series, ORP96-09, http://www.cs.utexas.edu/users/jbc/, 1996.

Beck J-C., Prosser P. and Selensky E., Vehicle routing and job shop scheduling: what's the difference?, ICAPS 2003, Thirteenth International Conference on Automated Planning and Scheduling, Trento, Italy, pages 267-276, 3-9 june 2003.

Bierwith C., A Generalized Permutation Approach to Jobshop Scheduling with Genetic Algorithms. OR Spektrum, vol 17, pages 87-92, 1995.

Blazewicz J., Ecker K.H., Pesch E., Schmidt G., Weglarz J., Scheduling computer and manufacturing processes, Chapitre 8: scheduling in job shops, Second edition, Springer 2001.

Bogarin V., Utilisation d'Heuristiques de Mutation Dirigée pour la Résolution du Job Shop Flexible Multiobjectif par une approche évolutionniste, Mémoire de D.E.A, Université des Sciences et Techniques de Lille, 2001.

Bogarin V., Hsu T., Dupas R., Jolly D., Une approche évolutionniste multi-objectifs pour la résolution du job shop flexible, CIFA 2002, Conférence Internationale Francophone d'Automatique 8-10 juillet 2002, Nantes, France, 2002.

Brandimarte P., Routing and Scheduling in a flexible job shop by tabu search, Annals of Operations Research 22, pages 158-183, 1993.

Branke J., Evolutionary Optimization in Dynamic Environments, Kluwer Academic Publishers, 2002.

Bräysy O. and M. Gendreau. Genetic Algorithms for the Vehicle Routing Problem with Time Windows. Internal Report STF42 A01021, SINTEF Applied Mathematics, Department of Optimisation, Oslo, Norway. 2001.

Burlat P., Marcon E., Senechal O., Dupas R., Démarches d'évaluation et de pilotage de la performance, Chapitre 3 dans Evaluation de performances des systèmes de production, Hermès, Traité IC2 série Productique, France 2003.

Camus H., Conduite des systèmes Flexibles de Production Manufacturière par composition de régimes permanents cycliques : modélisation et évaluation de performances à l'aide des Réseaux de Petri, Thèse de Doctorat, Université de Lille 1, mars 1997.

Carlier J., Chrétienne P., Problèmes d'ordonnancements : modélisation, complexité, algorithmes, Masson 1988.

Cavory G., Une approche génétique pour la résolution d'ordonnancements cycliques , Thèse de Doctorat, Université d'Artois, 2000.

Cavory G., Dupas R. and Goncalves G., A genetic approach to the scheduling of preventive maintenance tasks on a single product manufacturing production line , International Journal of Production Economics, 74, pages 135-146, 2001.

Cavory G., Dupas R., Goncalves G, A genetic approach to solving the problem of cyclic job shop scheduling with linear constraints, European Journal of Operational Research 2003 *à paraître*.

Chankong V., Haimes Y. Y., Thadathil J., Zionts S., Multiple criteria optimization : a state-of-the-art review. In Proceedings of the sixth International Conference on Multiple-Criteria Decision Making, pages 36-90, 1985.

Chen C. L., Bulfin R., Complexity of single machine, multi-criteria scheduling problems. European Journal of Operational Research, 70, pages 115-125, 1993.

Cheng R.W., Gen M., Tsujimura Y., A tutorial survey of job-shop scheduling problems using genetic algorithms, partII : hybrid genetic search strategies , Computers and Industrial Engineering, vol. 36, n°2, pages 343-364, 1999.

Chrétienne P., Analyse des régimes transitoire et permanent d'un graphe d'événements temporisé. Techniques Sciences Informatique , 4, 1, 1, pages 127-142, 1985.

Chrétienne P., Timed event Graphs: a solution to the minimum reachability time between two states, Journal system and Software, 1 et 2 1986.

Chrétienne P., List schedules for cyclic scheduling, Rapport de recherche Litp 1996/34, 1996

Chrétienne P., Coffman E.G., Lenstra, J.K., Liu Z., Scheduling Theory and Its Applications , John Wiley & Sons Ltd, 1997.

Chrétienne P., On the Graham's Bound for cyclic Scheduling, Rapport de recherche LIP6 1999-013, 1999.

Coello C., An Updated Survey of Evolutionary Multiobjective Optimization Techniques: State of the Art and Future Trends, Congress on Evolutionary Computation, IEEE Service Center, Washington D.C. ,July, pages 3-13, 1999.

Coello C.A. , Handling Preferences in evolutionary multiobjective optimization : a survey, Congress on evolutionary computation, volume 1, pages 30-37, Piscataway, New Jersey, july, 2000.

Crama Y., Combinatorial optimisation models for production scheduling in automated manufacturing systems, European Journal of Operational Research, n° 99, pages 136-153, 1997.

Crama Y., Kats V., van de Klundert J.and Levner E., Cyclic scheduling in robotic flowshops, Annals of Operations Research: Mathematics of Industrial Systems 96, pages 97-124, 2000.

Dauzère-Pérès S., Paulli, J., An integrated approach for modeling and solving the general multiprocessor job shop scheduling problem using tabu search, Annals of Operations Research 70, pages 281-306, 1997.

Davis, L., Applying adaptive algorithms to epistatic domains, Proceedings of the International Joint Conference on Artifcial Intelligence, Morgan Kaufmann, pages 162-164, 1985.

Deb K. and al., A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II, Indian Institute of Technology, KanGAL, 200001, Kanpur, India, 2000.

Deb K., Multi-Objective Optimization usingvolutionary Algorithms, John Wiley & Sons, 2001.

De Jong K., Fogel D. B., Schwefel H-P, Evolutionary Computation 1: basic algorithms and operators, T. Back, D.B. Fogel & T. Michalewicz (Eds), A history of evolutionary computation, Chapitre 6, Institute of physics publishing, 2000.

Deming D.E., Out of the crisis : Quality Productivity and Competitive Position, Cambridge University Press, 1982.

Dhaenens C., Espinouse M-L., Penz B., Problèmes combinatoires classiques et techniques de résolution, Chapitre 3 dans Recherche opérationnelle et réseaux: méthodes d'analyse spatiale, ermes Lavoisier 2002.

Dorigo M., Maniezzo V., Colorni A, The ant System: optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics-Part B, 26, 1, pages 29-41, 1996.

Dupas, R., Benchmarks of cyclic job shop with linear constraints. http://www.iut-bethune.univ-artois.fr/dupas/benchCyclique/indexBench.html, 2001.

Dupas R., Fourmaux D., Goncalves. G., Utilisation des plans d'expériences pour l'évaluation de performances en simulation, Chapitre 9 dans Evaluation de perfomances des systèmes de production, Hermès, Traité IC2 série Productique, France 2003a.

Dupas R., Goncalves G., Job shop cyclique à contraintes linéaires : modélisation et résolution par un algorithme génétique, Journal Européen des Systèmes Automatisés N° 7-8 , pp 929-954, 2003b.

Draper L. D., Jonsson, A. K., Clements D. P. and Joslin, D. E., Cyclic Scheduling , Proceedings of the 16th International Joint Conference on Artificial Intelligence IJCAI'99, Stockholm, Sweden July 31-August 6 1999.

Esquirol P. et P. Lopez , Ordonnancement de la Production, editions Economica, 1999.

Falkenauer E., Bouffouix S., A genetic algorithm for job shop, Proceedings of IEEE International Conference on Robotics and Automation, pages 824-829, 1991.

Fisher H., Thompson G.L., Probabilistic learning combinations of local job-shop scheduling rules, J. F. Muth G.L. Thompson (eds.), Industrial Scheduling, Prentice Hall, Englewood Cliffs, New Jersey, pages 225-251, 1963.

Fogel L.J., Owens A. J. and Walsh M. J., Artificial Intelligence through Simulated Evolution, New York: Wiley, 1966.

Fonseca C. and Fleming P., Multiobjective Genetic Algorithms Made Easy: Selection, Sharing and Mating Restriction, Proceedings of the First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, Sheffield, UK, pages 42-52, September 1995.

Gendreau M., Potvin J-Y, Fleet management and logistics, chapter Dynamic Vehicle Routing and dispatching, pages 115-126, Kluwer Academic Publishers, 1998.

Gendreau, F. Guertin, J-Y. Potvin, E. Taillard. Parallel Tabu Search for Real-Time Vehicle Routing and Dispatching, Transportation Sciences 33 (4), pp 381-390, 1999.

Gentina J.C., Korbaa O., Camus H., Problèmes d'ordonnancement cyclique , Chapitre 7 dans Ordonnancement de la Production, eds Lopez P. et Roubellat F., Hermès Sciences, IC2 Productique, pages 197-223, 2001.

Giffler B., Thompson G.L. , Algorithms for Solving Production Scheduling Problems. Operations Research, vol. 8, pages 487-503, 1960.

Goldberg D.E., Genetic Algorithms in Search, Optimisation and Machine Learning, Addison-Wesley publishing compagny, 1989.

Goldberg D., Linge R., Alleles, loci and the traveling salesman problem. Proceedings of the first International Conference on Genetic Algorithms, Grefenstette editor, Hilldale NJ, 1985.

Graham R. L., Lawler E. L., Lenstra J. K. and Rinnooy Kan A. H. G., Optimisation and approximation in deterministic sequencing and scheduling: a survey, Annals Disc. Math., 5, pages 287-326, 1979.

GOThA, Les problèmes d'ordonnancements, RAIRO-Recherche Opérationnelle 27, pages 77-150, 1993.

Hanen C. et Munier A., Cyclic scheduling on parallel processors: an overview , Scheduling Theory and Its Applications : Chretienne P., Coffman E.G., Lenstra J.K. and Liu Z. editors , Chapitre 7, John Wiley & Sons Ltd, 1995.

Hansen M., Jaszkiewicz A., Evaluating the quality of approximations to the non-dominated set. IMM Tecnical Report IMM-REP-1998-7, Technical University of Denmark, 1998.

Hart E. et Ross P.M., A heuristic combination method for solving job-shop scheduling problems, Parallel Problem Solving from Nature V, eds. A.E. Eiben, T. Bäck, M. Schoenauer, H.P. Scwefel, Springer-Verlag Lecture Notes in Computer Science 1498, pages 845-854, Springer-verlag, 1998.

Hillion H.P., Proth J. M. et Xie X.L., A heuristic algorithm for the periodic scheduling and sequencing job-shop problem, 26th IEEE Conference on Decision and Control, Los Angeles, U.S.A. pages 612-617, 1987.

Hillion H.P. et Proth J. M., Analyse de fabrications non linéaires et répétitives à l'aide des Graphes d'Evénements Temporisés, RAIRO, Vol 22, n° 2, septembre 1988.

Hillion H.P., Proth J.M., Performance evaluation of job-shop systems using timed event-graphs, IEEE Transactions on Automatic Control, vol 34, n°1, pages 3-9, 1989.

Holland J.H., Outline for a logical theory of adaptive systems, ACM Journal 9, pages 297-314, 1962.

Holland J.H., Holyoak K. J., Nisbett R. E., Thagard P. R., Induction: processes of Inference, Leraning and Discovery, Cambridge MA, MIT Press, 1986.

Hurink E., Jurish B., Thole M., Tabu search for the job shop scheduling problem with multi-purpose machine, Operations Research Spektrum 15, pages 205-215, 1994.

Hsu T., Dupas R., Goncalves G., A genetic approach to solving the problem of FMS cyclic scheduling , 2002 IEEE International Conference on Systems, Man and Cybernetics, October 6-9, Hammamet, Session Ordonnancement Cyclique, 2002.

Ichoua S., Gendreau M., Potvin J-Y., Vehicle dispatching with time-dependent travel times, European Journal of Operational Research 144, pages 379-396, 2003.

Jacot J.H., Micaelli J.P., La performance économique des entreprises, ed Hermes, 1996.

Jain A. K., and Elmaraghy H. A., Production scheduling/rescheduling in flexible manufacturing, International Journal of Production Research, Vol 35, n°1, pages 281-309, 1997.

Jain A. S., Meeran S., A state of the art review of job-shop scheduling techniques, Technical Report, Department of Applied Physics Electronic and Mechanical Engineering, University of Dundee, Scotland, 1998.

Jain A. S., Meeran S., Deterministic job-shop scheduling : Past, present and future, European Journal of Operational Research 113, pages 390-434, 1999.

Jurisch B., Scheduling jobs in shops with multi-purpose machines, Ph. D. thesis, Fachbereich Mathematik/Informatik, Universitat Osnabruck, 1992.

Kacem I., Hammadi S. and Borne P., Direct chromosome representation and advanced genetic operators for job shop problems, Proceedings of the International Conference on Computational Intelligence for Modelling Control and Automation (CIMCA'01), Las Vegas, USA, 9-11 July 2001.

Kacem I., Ordonnancement multicritère des job-shops flexibles : formulation, bornes inférieures et approche évolutionniste coopérative, Thèse de doctorat, Université de Lille 1, 2003.

Keeney R.L. and Raiffa H., Decisions with Multiple Objectives: Preferences and Value Trade-offs. Cambridge University Press, Cambridge, UK, 1993.

Kilby, P., P. Prosser and P. Shaw. Dynamic VRPs: A Study of Scenarios. Report APES-06-1998, Computing Science, Glasgow University, Scotland. 1998.

Korbaa O., Commande cyclique des systèmes flexibles de production manufacturière à l'aide des réseaux de Petri : de la planification à l'ordonnancement des régimes transitoires, Thèse de doctorat, Université de Lille 1, 1998.

Korbaa O., Contribution à la conception et l'optimisation des systèmes de transport et de production, Mémoire d'habilitation à diriger des recherches, Université de Lille I, 18 décembre 2003.

Koza J.R., Genetic Programming, Cambridge MA, MIT Press, 1992.

Laporte, G., M. Gendreau, J.-Y. Potvin and F. Semet (1999). Classical and Modern Heuristics for the Vehicle Routing Problem. International Transactions in Operational Research 7, 285-300. 1999.

Larsen A., The dynamic vehicle routing problem, PhD Thesis, Technical University of Denmark, 2000.

Lawrence S., Resource constrained project scheduling : an experimental investigation of heuristic scheduling techniques, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1984.

Lee M-K., Yamakawa T., Genetic Algorithm Approaches to job shop scheduling problems : an overview, International Journal of Knowledge-Based Intelligent Engineering Systems, vol 4, n°2, April 2000.

Lopez P., Roubellat F., Ordonnancement de la production, Hermes Sciences , IC2 Productique, 2001.

Mahfoud S. W., Evolutionary Computation 2: advanced algorithms and operators, T. Back, D.B. Fogel & T. Michalewicz (Eds), Niching methods, chapitre 13, Institute of Physics Publishing, Bristol and Philadelphia, 2000.

Mastrolilli M., Gambardella L.M., Effective neighborhood functions for the flexible job shop problem, Journal of Scheduling, volume 3, Issue 1, pages 3-20, 2000.

Mesghouni K., Application des algorithmes évolutionnistes dans les problèmes d'optimisation en ordonnancement de la production, Université des Sciences et Technologies de Lille, janvier 1999.

Michalewicz Z., Genetic Algorithms + Data Structures = Evolution Programs, 3rd edn, Springer Verlag, Berlin Heidelberg New York, 1996.

Mattfeld, D.C., Evolutionary Search and the Job Shop. Springer-Verlag, 1995.

Munier A., The basic cyclic scheduling problem with linear precedence constraints , Discrete Applied Mathematics, n°64, pages 219-238, 1996.

Nakano R., Yamada T., Conventional Genetic Algorithm applicable to large-scale job shop problems, Proceedings of 4th International Conference On Genetic Algorithms, pages 474-479, 1991.

Niérat P., Transport combiné rail-route : contraintes et performances des dessertes routières, proceedings of the sixth world conference on transport research, Lyon, pages 2733-2743, 1992.

Ohl H., Fonctionnement répétitifs de systèmes flexibles de production manufacturière : analyse et optimisation des performances à l'aide des Réseaux de Petri, Thèse de Doctorat, Université de Lille 1, septembre 1995.

Pareto V., Cours d'économie politique, volume I et II, Lausanne 1896.

Pierreval H., Caux C., Paris J.-L., Viguier F., Evolutionary approaches to the design and organization of manufacturing systems, Computers & Industrial Engineering, vol 44, pages 339-364, 2003.

Portmann M.C., Vignier A., Algorithmes génétiques et ordonnancement , Chapitre 4 dans Ordonnancement de la Production, eds Lopez P. et Roubellat F., Hermès Sciences, IC2 Productique, pages 95-130, 2001.

Prins C., Un algorithme génétique très efficace pour le problème de tournées de véhicules, Conférence ROADEF-2002, ENST, 20-22 février 2002.

Rechenberg I., Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution. Fromman-Hozlboog Verlag, Stuttgart, 1973.
Reeves C.R., Modern heuristic techniques for combinatorial problems. MacGraw-Hill 1995.

Reeves C. R., Rowe J.E., Genetic algorithms and perspectives: a guide to GA theory, Chaptitre 4: no free lunch for GA's, Kluwer Academic Publishers, 2003a.

Reeves C. R., Rowe J.E., Genetic algorithms and perspectives: a guide to GA theory, Chapitre 9: landscapes, Kluwer Academic Publishers, 2003b.

Roundy R., Cyclic schedules for job-shops with identical jobs , Mathematics of Operations Research, vol 17, n°4, 1992.

Roy B., Sussmann, B., Les problèmes d'ordonnancement avec contraintes disjonctives. Notes D.S. n° 9 bis, SEMA, Paris, France, 1964.

Roy B., Méthodologie multicritère d'aide à la décision, Editions Economica 1985.

Schärlig A., Décider sur plusieurs critères, Presses polytechniques et universitaires romandes, 1985.

Schärlig A., Pratiquer Electre et Prométhée, Presses polytechniques et universitaires romandes, 1996

Schoenauer M., Xanthakis S., Constrained GA optimization, Proceeding of the 5th International Conference on Genetic Algorithms, pages 573-580. Morgan Kaufmann, 1993.

Schoenauer M., Adaptation et algorithmes evolutionnaires : définitions, terminologie et exemples, Journées Evolutionnaires Trimestrielles, http://lil.univ-littoral.fr/~fonlupt/JET6, Mars 2001.

Schweffel H-P, Evolution and Optimum Seeking, New York, Wiley 1995.

Serafini P., Ukovich W., A mathematical model for periodic scheduling problems, SIAM Journal Discrete Mathematics, Vol. 2 n°4, pp. 550-581, 1989.

Srinivas N. and Deb K., Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms, Evolutionary Computation, The Massachusetts Institute of Technology, 2, 3, pages 221-248, 1995.

Talbi EG., Méta-heuristiques pour l'optimisation combinatoire multi-objectif: état de l'art, Rapport Technique CNET, Université de Lille, octobre 1999.

Talbi E-G., Contributions à la résolution parallèle de problèmes d'optimisation combinatoire, Mémoire d'habilitation à diriger des recherches, Université de Lille I, 2000.

Tamaki, H., Nishikawa Y., Maintenance of diversity in a genetic Algorithm and an application to job shop scheduling, Proceeding IMACS/SICE pages 869-874, 1992.

T'Kindt V., J-C. Billaut, Multicriteria Scheduling: Theory, Models and Algorithms Springer, 2002.

Valentin C., Modeling and Analysis Methods for a Class of Hybrid Dynamic Systems, Proceedings of ADPM '94, Brussels, Belgium, pages 221-226, Nov. 1994.

Van Veldhuizen D., Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art, Evolutionary Computation, Summer 2000, 8, 2, pages 125-147.

Vincke P., L'aide multicritère à la décision, Statistiques et mathématiques appliquées, éditions Ellipses, 1989.

Wolpert D. H. and Macready W. G., No Free Lunch Theorems for Search, Rapport technique n° SFI-TR-95-02-010, Santa Fe, NM, USA, The Santa Fe Institute, february 1995.

Yamada, T. and Nakano, R., A genetic algorithm applicable to large scale Job Shop problems. In: Männer R., Manderick B. (Eds), Proceedings of the Second International Conference on Parallel Problem Solving from Nature, North Holland: Elsevier Science Publishers, pages 281-290, 1992.

Yim, P., Réseaux de Petri, logique et théorie des ensembles: contribution à l'étude des systèmes dynamiques discrets, Mémoire d'habilitation à diriger des recherches, Université de Lille I, 15 décembre 2000.

Zaffalon M. Rizzoli A. E., Gambardella L. M., Mastrolilli M., resource allocation and scheduling of operations in an intermodal terminal, 10th European Simulation Symposium and Exhibition, Simulation in Industry, Nottingham, United Kingdom, October 26-28, pages 520-528, 1998.

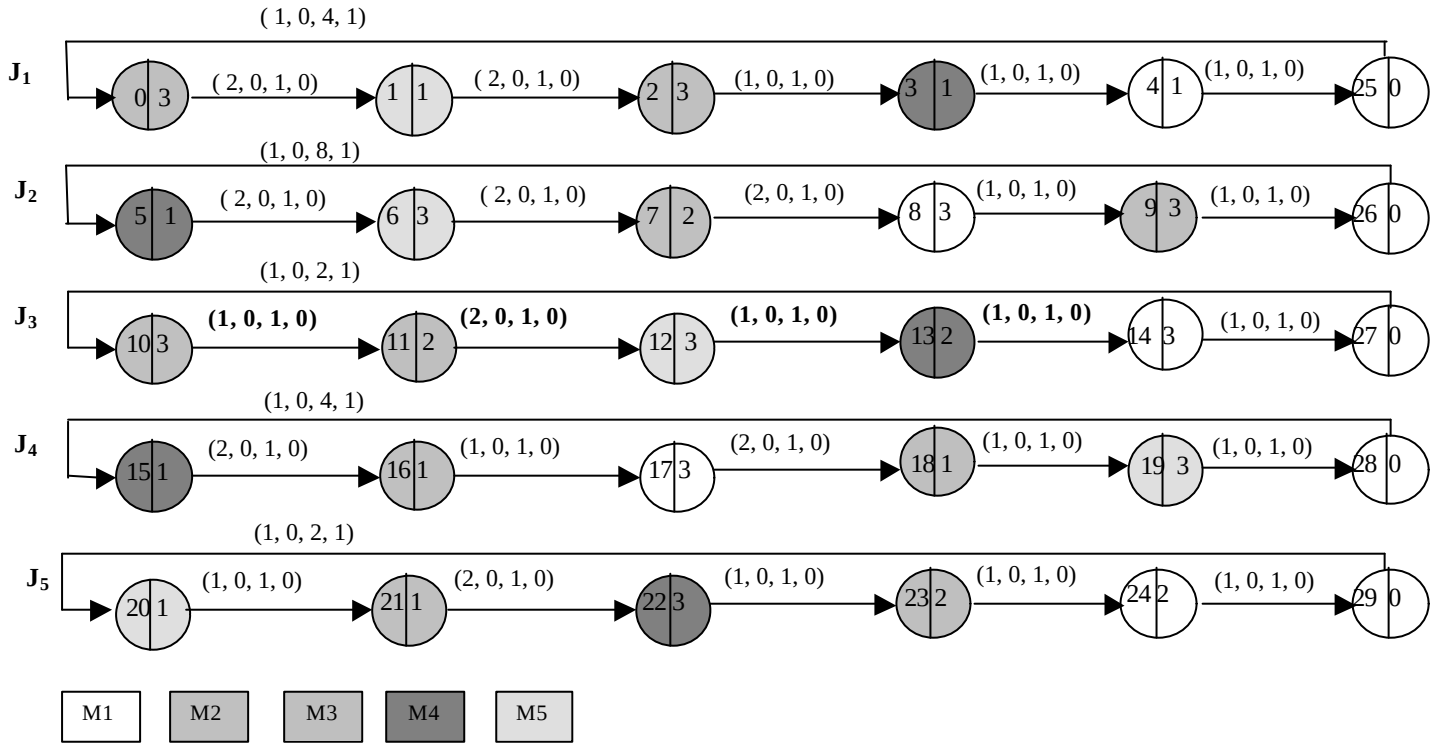
Zitzler E. and Thiele L., Multiobjective Evolutionary Algorithms: A comparative Case Study and the Strength Pareto Approach, IEEE Transactions on Evolutionary Computation, november, 3, 4, pages 257-271, 1999.

Zitzler E., Deb K., Thiele L., Comparisons of Multiobjective Evolutionary Algorithms: Empirical Results, Evolutionary Computation, The Massachusetts Institute of Technology, 8,2, pages 173-195, 2000.

Partie C Annexes

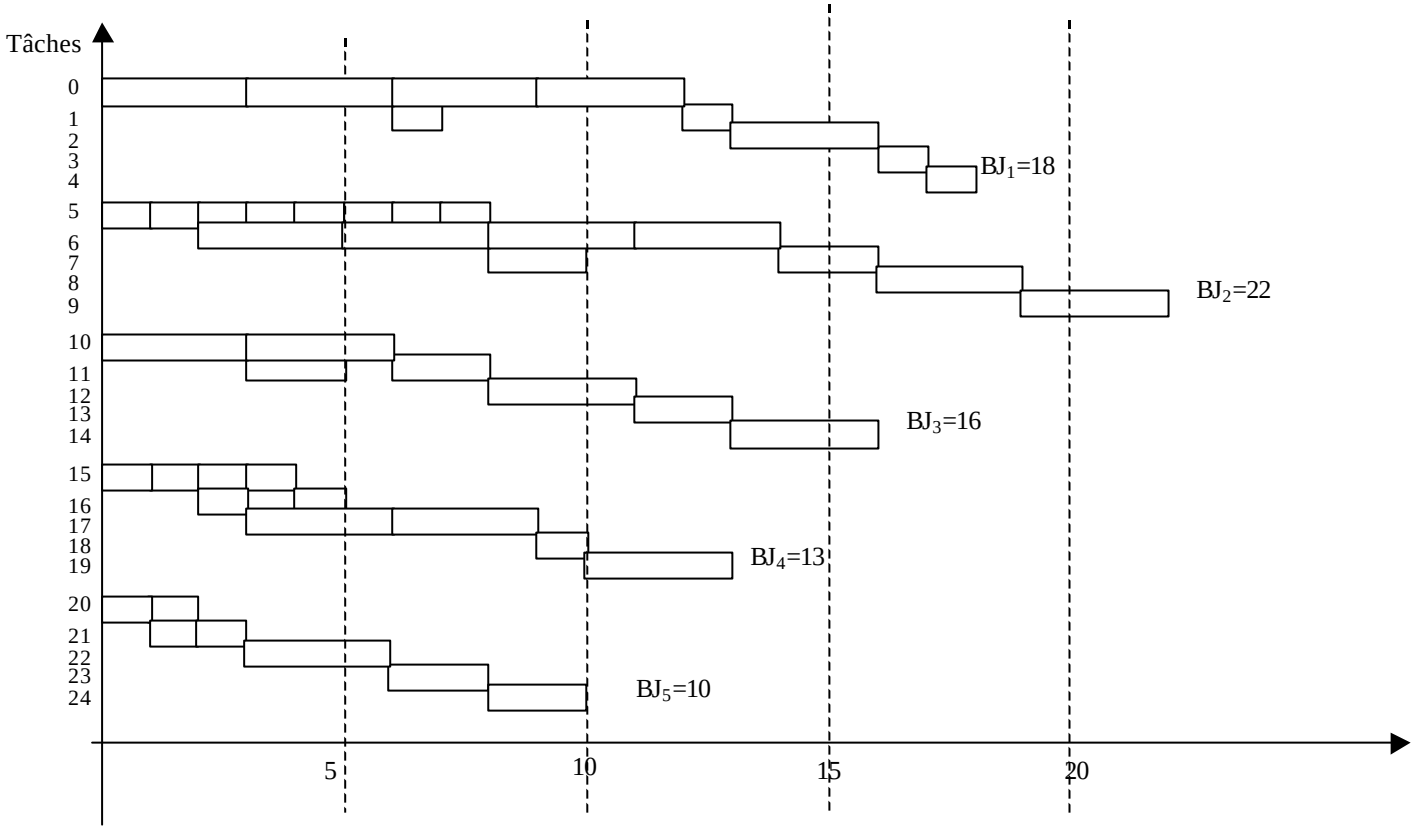
Annexe 1 Job shop cyclique à contraintes linéaires : exemple de calcul des bornes minorantes B1 et B2

1. Données du problème 5_5 [17]



[17] http://www.iut-bethune.univ-artois.fr/dupas/benchCyclique/indexBench.html.

2. Borne B1,sans contrainte de ressources: B1=15.8



3. Borne B2, sans contrainte de précédence: B2= 18.4

Numéro de tâches	Nombre d' occurrences	Temps opératoire	Temps total
4	1	1	1
8	1	3	3
14	1	3	3
17	2	3	6
24	1	2	2

Machine M1 : BM_1= 15

Numéro de tâches	Nombre d' occurrences	Temps opératoire	Temps total
2	1	3	3
7	2	2	4
10	2	3	6
16	2	1	2
21	2	1	2

Machine M2 : BM_2= 17

Numéro de tâches	Nombre d' occurrences	Temps opératoire	Temps total
0	4	3	12
9	1	3	3
11	2	2	4
18	1	1	1
23	1	2	2

Machine M3 : BM_3= 22

Numéro de tâches	Nombre d' occurrences	Temps opératoire	Temps total
3	1	1	1
5	8	1	8
13	13	2	2
15	4	1	4
22	1	3	3

Machine M4 : BM_4= 18

Numéro	Nombre d'	Temps	Temps

de tâches	occurrences	opératoire	total
1	2	1	2
6	4	3	12
12	1	3	3
19	1	3	3
20	2	1	2

Machine M5 : $BM_5 = 20$

Annexe 2 Job shop flexible: temps d'exécution des opérations de l'exemple

Job	o i,j	M_1	M_2	M_3	M_4	M_5
1	o1,1	1	9	3	7	5
	o1,2	3	5	2	6	4
	o1,3	6	7	1	4	3
2	o2,1	1	4	5	3	8
	o2,2	2	8	4	9	3
	o2,3	9	5	1	2	4
3	o3,1	1	8	9	3	2
	o3,2	5	9	2	4	3

Annexe 3 Job shop flexible : temps d'exécution des opérations du problème test

Job	o i,j	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
1	o 1,1	1	4	6	9	3	5	2	8	9	5
	o 1,2	4	1	1	3	4	8	10	4	11	4
	o 1,3	3	2	5	1	5	6	9	5	10	3
2	o 2,1	2	10	4	5	9	8	4	15	8	4
	o 2,2	4	8	7	1	9	6	1	10	7	1
	o 2,3	6	11	2	7	5	3	5	14	9	2
3	o 3,1	8	5	8	9	4	3	5	3	8	1
	o 3,2	9	3	6	1	2	6	4	1	7	2
	o 3,3	7	1	8	5	4	9	1	2	3	4
4	o 4,1	5	10	6	4	9	5	1	7	1	6
	o 4,2	4	2	3	8	7	4	6	9	8	4
	o 4,3	7	3	12	1	6	5	8	3	5	2
5	o 5,1	5	6	3	9	8	2	8	6	1	7
	o 5,2	6	1	4	1	10	4	3	11	13	9
	o 5,3	7	10	4	5	6	3	5	15	2	6
6	o 6,1	8	9	10	8	4	2	7	8	3	10
	o 6,2	7	3	12	5	4	3	6	9	2	15
	o 6,3	4	7	3	6	3	4	1	5	1	11
7	o 7,1	1	7	8	3	4	9	4	13	10	7
	o 7,2	3	8	1	2	3	6	11	2	13	3
	o 7,3	5	4	2	1	2	1	8	14	5	7
8	o 8,1	6	2	13	5	4	3	5	7	9	5
	o 8,2	5	7	11	3	2	9	8	5	12	8
	o 8,3	8	3	10	7	5	13	4	6	8	4
9	o 9,1	3	9	1	3	8	1	6	7	5	4
	o 9,2	4	6	2	5	7	3	1	9	6	7
	o 9,3	8	5	4	8	6	1	2	3	10	12
10	o10,1	4	3	1	6	7	1	1	6	20	6
	o10,2	3	1	8	1	9	4	1	4	17	15
	o10,3	9	2	4	2	3	5	2	4	10	23

Annexe 4 Job shop flexible : ordonnancement associé à la solution [7, 45, 5]

J_1	M_1,0	M_3,1	M_4,3
J_2	M_1,1	M_7,3	M_3,5
J_3	M_{10},0	M_4,1	M_8,2
J_4	M_7,0	M_{10},1	M_4,5
J_5	M_9,0	M_9,2	M_4,6
J_6	M_6,0	M_9,3	M_7,5
J_7	M_1,3	M_3,4	M_6,5
J_8	M_5,0	M_2,2	M_2,5
J_9	M_6,2	M_7,4	M_6,6
J_{10}	M_3,0	M_4,3	M_5,4

Rappel de la notation :

L' élément (M_k, t) situé en colonne "j" et ligne "i" de la matrice signifie que la $j^{ième}$ opération du job J_i est affectée à la machine k et démarre à l'instant "t"

Annexe 5 Ordonnancement cyclique en PFM.: formalisation du problème

Hypothèses

- La recherche d'un ordonnancement se réduit à l'étude d'un seul cycle ; le même motif est ensuite répété dans les cycles suivants
- Une palette libérée à l'instant "t" peut être réutilisée au même instant pour une nouvelle pièce de même type
- Les palettes sont des palettes dédiées ou non standardisées ; une palette est dédiée à un seul type de job.

Notation

- Soit m le nombre de machines
- Soit n le nombre de jobs
- Soit m_i le nombre d'opérations du job i
- Soit o_{ij} la $j^{ième}$ opération du job i
- Soit p_{ij} la durée de l'opération o_{ij}
- Soit $s(i,j)$ l'indice du successeur de l'opération o_{ij} : $s(i,j) = (j+1) \bmod m_i$, $i = 1, .., n$; $j = 1, .., mi$.
- Soit O l'ensemble des opérations $O= \{o_{ij} \mid i = 1, .., n; j = 1, .., m_i\}$.
- Soit E_r l'ensemble des opérations affectées à la machine r
- Soit E_r^* l'ensemble des paires d'opérations (O_{ij}, O_{kl}) appartenant à la même machine r. E_r^* est égal à l'ensemble E_r x E_r auquel les paires d'opérations identiques ont été retirées : $E_r^* = \{(O_{ij}, O_{kl}) \mid \exists \quad r \in \{1, .., m\}, o_{ij} \in E_r, o_{kl} \in E_r, (i,j) \neq (k,l)\}$

Les ensemble $(E_r, r = 1, .., m)$ forment une partition de O: $\bigcup_{r=1,.., m} E_r = O$ et $\bigcap_{r=1,.., m} E_r = \varnothing$

Le temps de cycle (TC) minimum donné par la machine goulot est: $TC = \max_{i=1,.., n} \left(\sum_{j=1}^{m_i} (p_{ij}) \right)$

Variables de décision

- t_{ij} est la date de début de l'opération o_{ij}. $t_{ij} \in \{0, .., TC\text{-}1\}$. t_{11} est fixée à 0.

- y_{ijkl} est utilisée pour déterminer l'ordre de séquencement de 2 opérations o_{ij} et o_{kl} dans un cycle. Deux opérations appartenant à la même machine ne peuvent pas être en recouvrement : l'une des deux précède nécessairement l'autre.
y_{ijkl} est donc une variable binaire relative à la séquence d'opération o_{ij} et $o_{kl} \in E_r$

y_{ijkl} = 1 si o_{ij} est ordonnancée avant o_{kl}

y_{ijkl} = 0 sinon.

Modèle mathématique

Minimiser l'encours : Min EC (1)

avec

$t_{ij} - t_{kl} \geq p_{kl} - y_{ijkl} * TC$ $\forall (o_{ij}, o_{kl}) \in E_r^*$ (2)

$t_{kl} - t_{ij} \geq p_{ij} - (1 - y_{ijkl}) * TC$ $\forall\ (o_{ij}, o_{kl}) \in E_r^*$ (3)

$t_{ij} \in \{0, .., TC\text{-}1\}$ $\forall\ o_{ij} \in O$ (4)

$y_{ijkl} \in \{0, 1\}$ $\forall\ (o_{ij}, o_{kl}) \in E_r^*$ (5)

L'objectif est de minimiser l'encours global (1).

Les contraintes (4), (5) précisent le domaine de définition des variables de décision.

Les contraintes disjonctives (2) (3) précisent le non recouvrement entre deux opérations appartenant à la même machine. En effet si o_{ij} et o_{kl} appartiennent à la même machine, soit o_{ij} est avant o_{kl} soit o_{kl} est avant o_{ij}. Dans le premier cas (voir Figure Annexe 5-1), y_{jikl} est égale à 1 et $t_{ij} - t_{kl} \geq p_{kl} - TC$ (2); de plus $t_{kl} - t_{ij} \geq p_{ij}$ (3). Le second cas correspond à la Figure Annexe 5-2. y_{jikl} est égale à 0 et $t_{ij} - t_{kl} \geq p_{kl}$ (2) ; de plus $t_{kl} - t_{ij} \geq p_{ij}$ - TC(3).

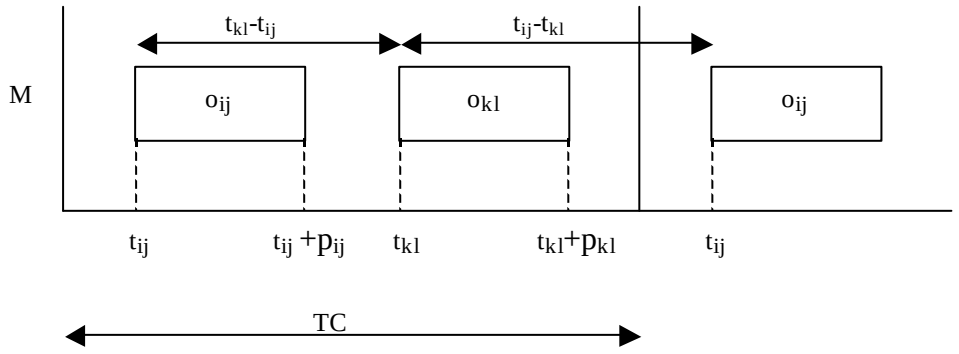


Figure Annexe 5-1 Contraintes disjonctives (2) et (3), cas o_{ij} avant o_{kl}

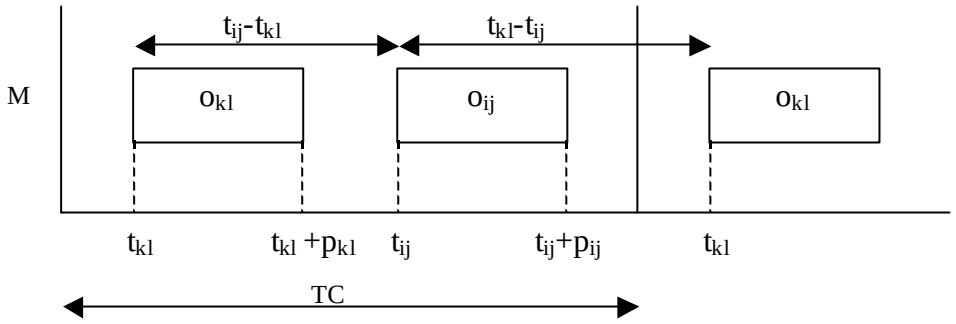


Figure Annexe 5-2 Contraintes disjonctives (2) et (3), cas o_{kl} avant o_{ij}

Il convient de noter que l'ensemble des contraintes ci-dessus n'inclut pas la contrainte de précédence entre les opérations de la même gamme. Cette contrainte est relâchée et intégrée dans la fonction d'objectif. Dans le cas ou une contrainte de précédence est violée dans le cycle courant, cette contrainte est satisfaite en choisissant l'opération suivante de la gamme dans le cycle suivant ; l'encours est donc incrémenté.

Fonction d'objectif

La fonction d'objectif correspond à l'encours minimum nécessaire. L'encours d'un job i est exprimé comme étant le nombre entier de temps de cycle pour réaliser toutes les opérations nécessaires à la production d'une pièce, en respectant les contraintes de précédence ainsi que les contraintes de ressources.

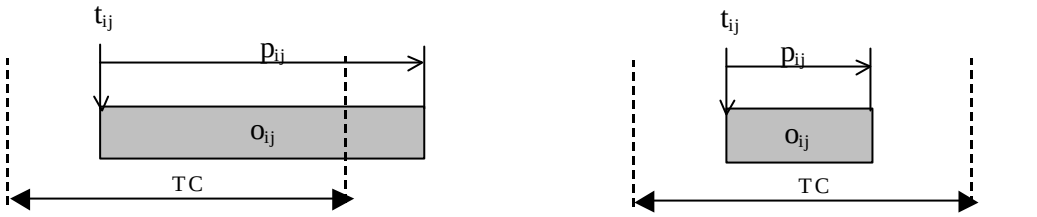
L'encours du job J_i est défini comme suit: $EC_i = \left\lceil \frac{\sum_{j=1}^{m_i} p_{ij} + \sum_{j=1}^{m_i-1} W_j}{TC} \right\rceil$ avec W_j représentant le temps d'attente entre l'opération o_{ij} et l'opération suivante $o_{is(i,j)}$ pour la même occurrence de pièce. La minimisation de l'encours EC_i est équivalent à la minimisation de la somme de tous les temps d'attente entre les opérations du job J_i.

Afin d'exprimer cet objectif et d'obtenir les valeurs du temps d'attente, une variable auxiliaire notée Z doit être utilisée. Lors de l'évaluation du temps d'attente entre o_{ij} et l'opération suivante $o_{is(i,j)}$, $Z_{ijs(i,j)}$ représente le nombre de temps de cycle dont le successeur doit être décalé afin d'obtenir un temps d'attente positif. Ceci correspond à une pénalité à chaque fois qu'une palette supplémentaire doit être ajoutée.

$$Z_{ijs(i,j)} \in \{0, 1, 2\} \qquad \forall o_{ij} \in O \qquad (6)$$

Pour exprimer la variable Z, un prédicat de franchissement de la limite du temps de cycle doit être défini ; il est appelé " **Franchissement** ". Ce prédicat, représenté en Figure Annexe 5-3, est défini comme suit:

- **Franchissement** (O_{ij}) est Vrai $\Rightarrow t_{ij}+p_{ij} > TC$ (voir Figure Annexe 5-3-a)
- **Franchissement** (O_{ij}) est Faux $\Rightarrow t_{ij}+p_{ij} <= TC$ (voir Figure Annexe 5-3-b)



a: **Franchissement (o_{ij}) est *vrai***

b: **Franchissement (o_{ij}) est *faux***

Figure Annexe 5-3 Prédicat de franchissement du temps de cycle

Il est alors possible de définir les valeurs de Z à l'aide de ce prédicat : 2 cas doivent être distingués :

Cas 1: **Franchissement** (o_{ij}) est *faux*: o_{ij} ne franchit pas la limite du temps de cycle.

Dans ce cas représenté en Figure Annexe 5-4, Z est définie comme suit :

- Si $t_{is(i,j)} >= t_{ij}+p_{ij}$ alors Z=0
- Si $t_{is(i,j)} < t_{ij}+p_{ij}$ alors Z=1

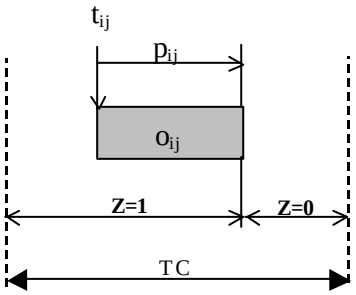


Figure Annexe 5-4 Valeurs de Z si Franchissement (o_{ij}) est *faux*

Cas 2: **Franchissement** (o_{ij}) est *Vrai*: o_{ij} franchit la limite du temps de cycle.

Dans ce cas représenté en Figure Annexe 5-5, Z est définie comme suit :

- Si $t_{is(i,j)} >= (t_{ij}+p_{ij}) \bmod TC$ alors Z=1
- Si $t_{is(i,j)} < (t_{ij}+p_{ij}) \bmod TC$ alors Z=2

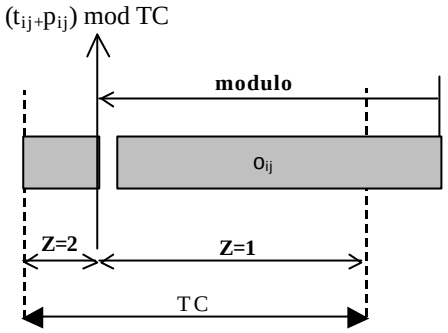


Figure Annexe 5-5 Valeurs de Z si Franchissement (o_{ij}) est *vrai*

En utilisant cette variable Z, l'encours EC_i du job i, peut être reformulé comme suit :

$$Ec_i = \left\lceil \frac{\sum_{j=1}^{mi} p_{ij} + \sum_{j=1}^{mi-1} (t_{is(i,j)} + z_{ijs(i,j)} \cdot TC - (t_{ij} + p_{ij}))}{TC} \right\rceil \quad (7)$$

L'encours global de tous les jobs est donné par: $Ec = \sum_{i=1}^{n} Ec_i$ (8)

Considérons à titre d'exemple la fonction d'objectif du problème donné en Figure Annexe 5-6. Celui-ci comporte un seul job ayant 4 opérations :A1, A2, A3 et A4 ; chaque opération a un temps opératoire de 4 unités de temps qui est égal au temps de cycle optimal (TC). Les valeurs des variables Z et W sont :

$Z_{A1A2}=1$; $w_1= t_{A2}+ Z_{A1A2}*TC- (t_{A1}+p_{A1})=3+1*4-(0+4)=3$
$Z_{A2A3}=2$; $w_2= t_{A3}+ Z_{A2A3}*TC- (t_{A2}+p_{A2})=2+2*4-(3+4)=3$
$Z_{A3A4}=2$; $w_3= t_{A4}+ Z_{A3A4}*TC- (t_{A3}+p_{A3})=1+2*4-(2+4)=3$

$$EC_A = \left\lceil \frac{(p_{A1}+p_{A2}+p_{A3}+dp_{A4})+(w_1+w_2+w_3)}{TC} \right\rceil \quad EC_A = \left\lceil \frac{(16)+(9)}{4} \right\rceil = \left\lceil \frac{25}{4} \right\rceil = 7$$

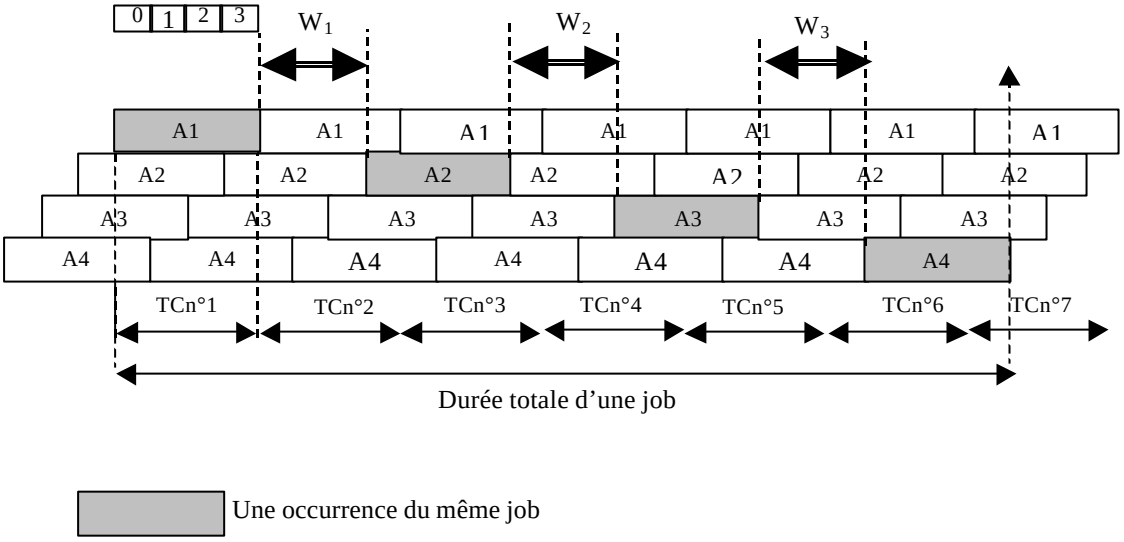


Figure Annexe 5-6 Calcul de l'encours

Annexe 6 Ordonnancement cyclique en PFM : problèmes tests

FORMAT DE DESCRIPTION DES PROBLEMES

(Référence du problème**)**

Nombre de machines / ensemble des machines
Nombre de jobs / ensemble des jobs (label_du_job , nombre_d'opérations)
Liste des gammes des jobs
Pour chaque job, toutes les opérations sont ordonnées et séparées par un point virgule ;
Chaque opération est donnée sous la forme : numéro_de_machine temps_opératoire ;

(HILLON, 1987)

4 / {M1, M2, M3, M4}
4 / {(A,4); (B,4); (C;3); (D,4)}

A: M1 1; M2 4; M3 3; M4 3;
B: M4 1; M2 2; M1 3; M3 1;
C: M1 2; M3 1; M4 1;
D: M3 3; M1 2; M2 1; M4 2;

(HILLION, 1988)

3 / {M1, M2, M3}
4 / {(A,3); (B,2); (C,2); (D,2)}

A: M1 1; M2 3; M3 3;
B: M3 1; M2 2;
C: M1 2; M3 1;
D: M1 2; M3 1;

(OHL, 1995)

6 / {M1, M2, M3, M4, M5, M6}
2 / {(A,6); (B,4)}

A: M1 18; M3 14; M2 12; M3 14; M2 10; M4 14;

B: M5 5; M6 4; M5 5; M6 4;

(VALENTIN, 1994)

3 / {M1, M2, M3}
5 / {(A,3); (B,3); (C,3); (D,2); (E,2)}

A: M1 2; M2 3; M3 2;
B: M1 2; M2 3; M3 2;
C: M1 2; M2 3; M3 2;
D: M2 1; M1 2;
E: M2 1; M1 2;

(KORBAA, 1998) instance a

9 / { M11 M12 M13 M2 M3 M41 M42 M5 M6}
7 / {(A,3); (B,3); (C,3); (D,4); (E,4); (F,3) (G,3)}

A: M11 11; M2 10; M41 8;
B: M12 11; M3 9; M6 8;
C: M42 12; M5 11; M6 8;
D: M2 4; M13 11; M41 6; M5 3;
E: M2 4; M41 6; M13 11; M5 3;
F: M11 13; M42 8; M3 9;
G: M12 13; M2 6; M5 7;

(KORBAA, 1998) instance b

9 / { M11 M12 M13 M2 M3 M41 M42 M5 M6}
2 / {(A,9); (B,14)}

A: M11 11; M2 10; M41 8; M12 11; M3 9; M6 8; M42 12; M5 11; M6 8;

B: M2 4; M13 11; M41 6; M5 3; M2 4; M41 6; M13 11; M5 3;M11 13; M42 8;M3 9; M12 13;M2 6; M5 7;

Annexe 7 Ordonnancement cyclique en PFM : nombre de solutions admissibles

Pour ce problème d'ordonnancement cyclique dont l'objectif est la minimisation de l'encours en respectant le temps de cycle optimal, il est possible d'évaluer le nombre de solutions admissibles obtenues par une méthode d'exploration naïve, de la manière suivante:

- soit n_k le nombre d'opérations affectées à la machine k:

$$n_k = \sum_{i=1}^{n} \sum_{j=1}^{mi} \delta_j, \text{avec} \delta_j = 1 \text{ si } m_{ij} = k \text{ et sinon } \delta_j = 0$$

- soit $nf_k = (TC - W_k)$ la durée d'inactivité de la machine k, au cours du temps de cycle TC, avec W_k la charge de la machine k. La valeur de nf_k correspond aux nombres d'opérations fictives de durée unitaire que l'on peut insérer dans un cycle.

Il est possible d'évaluer dans un premier temps le nombre, noté ϕ_k, d'ordonnancements possibles sans chevauchement sur la machine k. Ce nombre correspond aux permutations possibles de toutes les opérations réelles ou fictives unitaires du cycle, auxquelles il faut déduire toutes les permutations entre-elles d'opérations fictives car ces dernières produisent des solutions identiques. Donc ϕ_k est égal à: $\phi_k = (nf_k + n_k)! / nf_k!$.

Dans un deuxième temps, il s'agit de comptabiliser toutes les solutions obtenues par chevauchements d'opérations à partir des solutions précédentes. Ce nombre, noté φ_k, est obtenu comme suit : pour chaque opération dont la durée est strictement supérieure à un, le chevauchement est possible (voir terme $(p_{ij}-1)$ ci-dessous) ; pour chaque solution en chevauchement engendrée, toutes les opérations (fictives ou non) sauf l'opération en chevauchement considérée, peuvent être permutées de manière identique au cas sans recouvrement (voir terme $(nf_k + n_k - 1)! / nf_k!$).

Ainsi $$\varphi_k = \sum_{i=1}^{n} \sum_{j=1}^{mi} \left((p_{ij} - 1) \cdot \frac{(nf_k + n_k - 1)!}{nf_k!} \right) \mid m_{ij} = k$$

Finalement, le nombre total de solutions possibles, noté Ns, en tenant compte de toutes les machines du problème est :

$$Ns = \prod_{k=1}^{m} (\phi_k + \varphi_k).$$

A titre indicatif, le nombre de solutions admissibles (Ns) pour le problème Korbaa 1998 (instance b) est de l'ordre de 8.6E+10.

Annexe 8 Copies de Publications

1. IEPM 2001

Cavory G., Dupas R. and Goncalves G., A genetic approach to the scheduling of preventive maintenance tasks on a single product manufacturing production line , International Journal of Production Economics, 74, pages 135-146, 2001.

2. Hermes IC2, Evaluation de performance des systèmes de production

Burlat P., Marcon E., Senechal O., Dupas R., Démarches d'évaluation et de pilotage de la performance, Chapitre 3 dans Evaluation de performances des systèmes de production, Hermès, Traité IC2 série Productique, France 2003.

3. Hermes IC2, Evaluation de performance des systèmes de production

Dupas R., Fourmaux D., Goncalves. G., Utilisation des plans d'expériences pour l'évaluation de performances en simulation, Chapitre 9 dans Evaluation de performances des systèmes de production, Hermès, Traité IC2 série Productique, France 2003a.

4. EJOR 2003

Cavory G., Dupas R., Goncalves G, A genetic approach to solving the problem of cyclic job shop scheduling with linear constraints, European Journal of Operational Research 2003 *à paraître*.