HAL open science

Optimisation de la politique de lotissement et de séquencement pour une ligne de production soumise aux aléas

Kseniya Schemeleva

▶ To cite this version:

Kseniya Schemeleva. Optimisation de la politique de lotissement et de séquencement pour une ligne de production soumise aux aléas. Autre. Ecole Nationale Supérieure des Mines de Saint-Etienne, 2010. Français. NNT : 2010EMSE0592 . tel-00667950

HAL Id: tel-00667950

https://theses.hal.science/tel-00667950v1

Submitted on 8 Feb 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

NNT : **2010 EMSE 0592**

THÈSE

présentée par

Kseniya SCHEMELEVA

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

Spécialité : Génie Industriel

Optimisation de la politique de lotissement et de séquencement pour une ligne de production soumise aux aléas

soutenue à Saint-Etienne, le 13 décembre 2010

Membres du jury

Président :	Gilles Goncalves	Professeur, Université d'Artois
Rapporteurs :	Bernard Penz	Professeur, INPG, Grenoble
	Imed Kacem	Professeur, Université Paul Verlaine, Metz
Examinateur(s) :	Lyes Benyoucef	Chargé de Recherche, HDR, INRIA
	Michel Gourgand	Professeur, ISIMA, Clermont-Ferrand
	Xavier Delorme	Maître Assistant, ENSM-SE, Saint-Étienne
Directeurs de thèse :	Alexandre Dolgui	Professeur, ENSM-SE, Saint-Étienne
	Fréderic Grimaud	Maître Assistant, ENSM-SE, Saint-Étienne

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

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

AVRIL	Stéphane	MA	Mécanique & Ingénierie	CIS
BATTON-HUBERT	Mireille	MA	Sciences & Génie de l'Environnement	SITE
BENABEN	Patrick	PR 1	Sciences & Génie des Matériaux	CMP
BERNACHE-ASSOLLANT	Didier	PR 0	Génie des Procédés	CIS
BIGOT	Jean-Pierre	MR	Génie des Procédés	SPIN
BILAL	Essaïd	DR	Sciences de la Terre	SPIN
BOISSIER	Olivier	PR 1	Informatique	G2I
BORBELY	Andras	MR	Sciences et Génie des Matériaux	SMS
BOUCHER	Xavier	MA	Génie Industriel	G2I
BOUDAREL	Marie-Reine	PR 2	Génie Industriel	DF
BOURGOIS	Jacques	PR 0	Sciences & Génie de l'Environnement	SITE
BRODHAG	Christian	DR	Sciences & Génie de l'Environnement	SITE
BURLAT	Patrick	PR 2	Génie industriel	G2I
COLLOT	Philippe	PR 1	Microélectronique	CMP
COURNIL	Michel	PR 0	Génie des Procédés	SPIN
DAUZERE-PERES	Stéphane	PR 1	Génie industriel	CMP
DARRIEULAT	Michel	IGM	Sciences & Génie des Matériaux	SMS
DECHOMETS	Roland	PR 1	Sciences & Génie de l'Environnement	SITE
DESRAYAUD	Christophe	MA	Mécanique & Ingénierie	SMS
DELAFOSSE	David	PR 1	Sciences & Génie des Matériaux	SMS
DOLGUI	Alexandre	PR 1	Génie Industriel	G2I
DRAPIER	Sylvain	PR 2	Mécanique & Ingénierie	SMS
DRIVER	Julian	DR 0	Sciences & Génie des Matériaux	SMS
FEILLET	Dominique	PR 2	Génie Industriel	CMP
FOREST	Bernard	PR 1	Sciences & Génie des Matériaux	CIS
FORMISYN	Pascal	PR 1	Sciences & Génie de l'Environnement	SITE
FORTUNIER	Roland	PR 1	Sciences & Génie des Matériaux	SMS
FRACZKIEWICZ	Anna	DR	Sciences & Génie des Matériaux	SMS
GARCIA	Daniel	MR	Génie des Procédés	SPIN
GIRARDOT	Jean-Jacques	MR	Informatique	G2I
GOEURIOT	Dominique	MR	Sciences & Génie des Matériaux	SMS
GRAILLOT	Didier	DR	Sciences & Génie de l'Environnement	SITE
GROSSEAU	Philippe	MR	Génie des Procédés	SPIN
GRUY	Frédéric	MR	Génie des Procédés	SPIN
GUY	Bernard	MR	Sciences de la Terre	SPIN
GUYONNET	René	DR	Génie des Procédés	SPIN
HERRI	Jean-Michel	PR 2	Génie des Procédés	SPIN
INAL	Karim	PR 2	Microélectronique	CMP
KLÖCKER	Helmut	DR	Sciences & Génie des Matériaux	SMS
LAFOREST	Valérie	CR	Sciences & Génie de l'Environnement	SITE
LERICHE	Rodolphe	CR CNRS	Mécanique et Ingénierie	SMS
LI	Jean-Michel	EC (CCI MP)	Microélectronique	CMP
LONDICHE	Henry	MR	Sciences & Génie de l'Environnement	SITE
MALLIARAS	George Grégory	PR 1	Microélectronique	CMP
MOLIMARD	Jérôme	MA	Mécanique et Ingénierie	SMS
MONTHEILLET	Frank	DR 1 CNRS	Sciences & Génie des Matériaux	SMS
PERIER-CAMBY	Laurent	PR 2	Génie des Procédés	SPIN
PIJOLAT	Christophe	PR 1	Génie des Procédés	SPIN
PIJOLAT	Michèle	PR 1	Génie des Procédés	SPIN
PINOLI	Jean-Charles	PR 0	Image, Vision, Signal	CIS
STOLARZ	Jacques	CR	Sciences & Génie des Matériaux	SMS
SZAFNICKI	Konrad	MR	Sciences & Génie de l'Environnement	SITE
THOMAS	Gérard	PR 0	Génie des Procédés	SPIN
TRIA	Assia		Microélectronique	CMP
VALDIVIESO	François	MA	Sciences & Génie des Matériaux	SMS
VAUTRIN	Alain	PR 0	Mécanique & Ingénierie	SMS
VIRICELLE	Jean-Paul	MR	Génie des procédés	SPIN
WOLSKI	Krzysztof	DR	Sciences & Génie des Matériaux	SMS
XIE	Xiaolan	PR 1	Génie industriel	CIS

Glossaire :

PR 0	Professeur classe exceptionnelle
PR 1	Professeur 1ère catégorie
PR 2	Professeur 2ème catégorie
MA(MDC)	Maître assistant
DR	Directeur de recherche
Ing.	Ingénieur
MR(DR2)	Maître de recherche
CR	Chargé de recherche
EC	Enseignant-chercheur
IGM	Ingénieur général des mines

Centres :

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

Dernière mise à jour le : 13 septembre 2010

École Nationale Supérieure des Mines
de Saint-Étienne

N° d'ordre : 0592

Kseniya Schemeleva

Optimization of a lot-sizing and sequencing problem for an imperfect production line

Speciality : Industrial Engineering

Keywords : Process, Planning, Scheduling, Lot-Sizing, Uncertainties, Performance evaluation, Adequacy of architecture-algorithm

Abstract :

This thesis contains the research study of a multi-product lot-sizing and sequencing problem for an imperfect production line. Two types of uncertainties were taken into account: the random yield (because of defective items) and random lead time (due to machine breakdowns). The sequence dependent set-up time between different products was also taken into account.

This problem came from the electronics industry, more precisely from the automated manufacturing of several types of Printed Circuit Board. Since the plant was completely automated, the considered production line (like the rest of the plant) worked most of the day without any other staff but maintenance one, so it was necessary to consider a production schedule for the next 24 hours. This schedule was repeated every day. The problem was to define the optimal quantities of products to be manufactured (lot sizes) and the sequence of these lots to optimize a given factor. The problem addressed belongs to three research domains: 1) optimal lot-sizing for the imperfect production systems (or lot-sizing under uncertainty), 2) deterministic lot-sizing and scheduling, 3) scheduling with set-up times or (and) costs. In the literature we can find many examples of problems belonging to one or the intersection of two of these areas. But we did not find any work that deals with similar problem to ours.

Since the problem is too complicated as it is, we looked for ways of modeling it, which would have allowed us to solve it. We found three cases where the original problem could be decomposed into several parts, each of which could be converted to a known problem of Operations Research. Then we worked on the lot-sizing part of the decomposed problem. Meanwhile, we showed how other parties could be resolved.

These kinds of problems are very important for a supply chain optimizing. Their solutions help to organize an efficient production, which in turn allows to make significant financial gains to company.

École Nationale Supérieure des Mines
de Saint-Étienne

N° d'ordre : 0592

Kseniya Schemeleva

Optimisation de la politique de lotissement et de séquencement pour une ligne de production soumise aux aléas

Spécialité: Génie Industriel

Mots clefs : Procédé, Planification, Ordonnancement, Lotissement, Aléas, Evaluation de performances, Adéquation architecture-algorithme

Résumé :

Les travaux de recherche effectués dans le cadre de cette thèse concernent un problème de *lotissement* et de *séquencement* pour une ligne de production *imparfaite*. Deux types d'aléas sont pris en compte : le *rendement aléatoire* (à cause des rebuts) et le temps *d'exécution aléatoire* (à cause des pannes machines). Les temps de *changement* de *série* dépendant de la séquence des produits sont également pris en compte.

Le problème est issu de d'une *fabrication automatisée* (usine-automate) des circuits imprimés et il a été posé lors de la conception du système de gestion de production de l'atelier fabriquant les partons conducteurs de plusieurs types. Étant donné que l'usine était complètement automatisée, l'atelier (comme le reste de l'usine) travaillait la plupart de la journée sans personnel autre que celui de maintenance, alors il fallait construire un planning de production pour les 24 heures suivantes. Ce planning devait être répété chaque jour. Le problème consistait à définir les quantités optimales de produits à traiter (*tailles* de *lots*) et *l'ordre* de *passage* des lots dans une ligne de production afin d'optimiser un critère.

Le problème traité appartient à trois domaines de recherche: 1) lotissement optimale pour les systèmes de production imparfaits (ou lotissement sous incertitudes); 2) ordonnancement et lotissement déterministe; 3) ordonnancement avec des temps ou (et) coût de changement de série (set-up). Dans la littérature scientifique nous trouvons beaucoup d'exemples de problèmes appartenant au un ou à l'intersection de deux de ces domaines. Par contre, nous n'avons pas trouvé les travaux qui traites de problèmes identiques au notre.

Etant donné que le problème est trop compliqué tel qu'il est, nous avons cherché des façons de son modélisation qui nous permettrons le résoudre. Nous avons trouvé trois cas où le problème initial peut être décomposé en plusieurs parties, chacune entre lesquelles peut être transformé dans un problème connu de la Recherche Opérationnelle. Ensuite nous avons travaillé que sur la partie lotissement du problème décomposé tout en montrant comment les autres partis peuvent être résolus.

Les problèmes de ce type sont très importants pour l'optimisation d'une chaine logistique. Ces résolutions aident d'organiser la production à la manière efficace, que permet aux entreprises de faire des gains financiers importants.

Remerciements

Je tiens avant tout à exprimer ma profonde gratitude à mes directeurs de thèse, Monsieur Alexandre Dolgui et Monsieur Frédéric Grimaud, pour la confiance qu'ils m'ont accordé, la qualité de leurs conseils, le constant investissement dont ils ont fait preuve ainsi que leurs encouragements. Je les remercie également pour m'avoir donné l'opportunité d'assister à plusieurs congrès internationaux. J'ai beaucoup appris à leur coté.

Je suis très reconnaissante envers Monsieur Gilles Goncalves, qui à accepté de présider le jury et d'examiner mon travail.

Je remercie vivement Monsieur Bernard Penz et Monsieur Imed Kacem qui m'ont fait l'honneur d'être les rapporteurs de ce manuscrit, pour l'intérêt qu'ils ont porté à ce travail, le temps qu'ils y ont consacré et pour leur remarques extrêmement constructives qui ont permis d'améliorer cette thèse.

J'adresse un grand merci à Monsieur Lyes Benyoucef et Monsieur Michel Gourgand qui ont accepté de faire partie de mon jury et d(examiner ma thèse.

Je souhaite remercier tout particulièrement Monsieur Xavier Delorme pour ses conseils, sa disponibilité et pour son aide précieuse au quotidien.

Mes remerciements vont également à tous les membres du Centre Génie Industriel et Informatique et, plus particulièrement à l'équipe MSGI, que j'ai côtoyé durant ces trois années. Je les remercie pour leur accueil, leur soutient et leur convivialité.

Je tiens à remercier à mes collègues qui sont désormais devenues les amies : Nilou, Natasha, Evgeniy, Khouloud, Sergey, Olga pour tous les moments passés ensemble au travail et dehors.

Enfin, ma gratitude et mes remerciements vont à ma famille qui m'a toujours encouragé et soutenu dans les moments difficiles au cours de ces longues années d'études, et à Thomas pour sa patience, sa bonne humeur et son aide à tout moment pendant la période de rédaction.

Table des matières

Liste des figures **vii**

Liste des tableaux **ix**

Liste des algorithmes **xi**

Introduction générale **1**

1 Travaux antérieurs et objectifs de l'étude **5**

1.1 Gestion de Production 5

1.2 Lotissement 9

1.3 Ordonnancement 19

1.4 Lotissement et ordonnancement 26

2 Problématique **31**

2.1 Hypothèses du problème 32

2.2 Notations et fonctionnement de la ligne 33

2.3 Analyse du problème 37

2.3.1 Dimensionnement de lots 37

2.3.2 Séquencement 40

3 Modèle Déterministe : approche FPTAS **43**

3.1 Description du système de production étudié 44

3.2 Des propriétés communes 45

3.3 Problème P-Temps 46
3.4 Problème P-Coût 51
3.5 Fully Polynomial Time Approximation Scheme (FPTAS) 55
3.6 Expérimentations numériques avec FPTAS 60

4 Modèle Déterministe : programmes linéaires 67

4.1 Modèles Linéaires pour le problème P-Coût-tailles-FD 67
4.1.1 Modèle en variables entières 68
4.1.2 Modèle avec des variables binaires 69
4.1.3 Expérimentations numériques - comparaison FPTAS avec les deux modèles linéaires 70
4.2 Problème P-Coût-Tailles : ajustement des méthodes 76
4.2.1 FPTAS pour le problème P-Coût-Tailles-$\alpha\beta$ 78
4.2.2 Modèle linéaire pour le problème P-Coût-Tailles-$\alpha\beta$ 80
4.2.3 Expérimentations numériques pour le problème PCT-$\alpha\beta$ 81

5 Modèle Probabiliste : décomposition et algorithmes approchés 87

5.1 Modélisation Mathématique de P-Prob 89
5.1.1 Notations 89
5.1.2 Modélisation des aléas 92
5.1.3 Modèle mathématique du problème 99
5.2 Méthode d'optimisation pour P-Prob 100
5.2.1 Calcul des bornes pour les tailles de lots 100
5.2.2 Décomposition 101
5.2.3 Un exemple 103
5.3 Résolution du problème P-Prob-P2(i) 105
5.3.1 Recherche Locale 105
5.3.2 Algorithme Génétique 106
5.3.3 Résultats expérimentaux 114

Conclusions et Perspectives **121**

Annexes **124**

A Les origines du problème traité **125**

B PCT-FD **: performances de FPTAS approché** **127**

B.1 PCT-FD : Temps de calcul de FPTAS (sec.) approché 128

B.2 PCT-FD : Les erreurs relatives des solution de FPTAS approché 129

C PCT-$\alpha\beta$ **: performances de FPTAS approché** **131**

C.1 PCT-$\alpha\beta$: Temps de calcul moyen de FPTAS approché 132

C.2 PCT-$\alpha\beta$: Erreurs relatives des solutions approchées de FPTAS 133

D Problème P-Prob**-P2(i)** **135**

D.1 Transformation de P-Prob-P2(i) en problème de Sac à Dos 135

D.2 Une procédure de programmation dynamique 137

Bibliographie **139**

Table des figures

1 Chaîne logistique 1

1.1 Gestion de la production 6

1.2 Les coûts de modèle EOQ 10

2.1 Flow-shop 32

2.2 Flow-shop généralisé 32

2.3 Histogramme de répartition des tailles de lots 38

2.4 Histogramme représentant la répartition de la valeur de Makespan 39

2.5 Convergence de la probabilité pour une solution 40

3.1 La solution optimale 46

3.2 Temps de calcul de FPTAS (sec.) pour $n \in [10; 150]$ en mode exact 62

3.3 Temps de calcul (sec.) avce et sans BIP pour $n \in [10; 150]$ en mode exact . . 62

3.4 Temps de calcul de FPTAS (sec.) pour $n \in [10; 150]$ 63

3.5 Les erreurs relatives de FPTAS (sec.) pour $n \in [10; 150]$ 64

5.1 Un exemple de traitement d'un lot sur une ligne de 4 machines 90

5.2 Une illustration avec des temps de pannes et de rebuts 91

5.3 Un processus de renouvellement (pannes machine) 94

5.4 Deux processus de renouvellement alternés 94

5.5 Modélisation des pannes machine et des réparations 95

5.6 Superposition de processus de Poisson 96

5.7 Partition de processus de Poisson 96

5.8 Schéma de décomposition 102

5.9 Schéma de l'algorithme Génétique (AG) proposé 109

A.1 Un exemple de circuits imprimé 125

Liste des tableaux

2.1 Une solution pour le problème de planification 34

2.2 Les données d'une instance de problème avec 8 lots 35

2.3 Le temps de set-up (en heures) 35

2.4 Probabilités p_{iq}, $i = 1, \ldots, n$ et $q = 1, \ldots, m$ 36

2.5 Taux de pannes et de réparations des machines 36

2.6 Les tailles de lots le plus fréquentes pour l'exemple 39

2.7 Une solution 40

3.1 Les intervalles pour la génération des données initiales des instances 61

4.1 Temps de calcul moyen de FPTAS, CPLEX(Ent) et CPLEX (Bin) 71

4.2 Temps de calcul moyen de FPTAS, CPLEX(Ent) et CPLEX(Bin) avec différents intervalles de α 72

4.3 Temps de calcul moyens pour FPTAS, CPLEX(Ent) et CPLEX(Bin) avec différents intervalles de la demande 73

4.4 Temps de calcul moyens pour FPTAS (optimal), CPLEX(Ent) et CPLEX(Bin) avec différents intervalles de coût de retard 74

4.5 Temps de calcul moyens pour FPTAS, CPLEX(Ent) et CPLEX(Bin) avec différents intervalles de T-ratio 75

4.6 Temps CPU pour FPTAS, CPLEX(Ent) et CPLEX(Bin) quand $d_i \in [1; 5]$, $c_i = 1$, $\alpha_i \in [0,001; 0,3]$ et le T-ratio $\in [0,9; 0,95]$ 76

4.7 Temps de calcul moyens de CPLEX($\alpha\beta$), FPTAS et BIP 81

4.8 Temps moyen de FPTAS et de CPLEX($\alpha\beta$) pour différents intervalles de α 82

4.9 Temps de calcul moyen de FPTAS et CPLEX($\alpha\beta$) pour différents intervalles de la demande 83

4.10 Temps de calcul moyen de FPTAS et CPLEX($\alpha\beta$) pour différents intervalles du coût de retard 83

4.11 Temps de calcul moyen de FPTAS et CPLEX($\alpha\beta$) pour différents intervalles de T-ratio 84

4.12 Temps CPU de CPLEX($\alpha\beta$) et de FPTAS pour $d_i \in [1; 5]$, $c_i = 1$ et T-ratio $\in [0,9; 0,95]$ 85

5.1 Solutions π^{*i} pour les problèmes P-Prob-P1(i) 104

5.2 Solutions x^{*i} pour des problèmes P-Prob-2(i) 104

5.3 Exemple d'une solution pour un problème avec 8 lots 106

5.4 Les intervalles pour la génération de données initiales des instances 115

5.5 Quantité des instances résolues par **DP** 116

5.6 Comparatif des temps CPU, $n \in [4, 13]$ 117

5.7 Qualité des solutions fournies par **AG** et **RL** 118

5.8 Temps de calcul moyen de AG et RL 118

5.9 Valeur de solution moyenne de AG et RL 119

A.1 Fiche technique pour l'atelier de fabrication des patrons conducteur 126

B.1 Temps de calcul de FPTAS (sec.) approché 128

B.2 Les erreurs relatives des solution de FPTAS approché 129

C.1 Temps de calcul moyen de FPTAS pour $\varepsilon \in \{0;\ 0,01;\ 0,05;\ 0,1;\ 0,3;\ 0,5;\ 0,7;\ 1\}$ 132

C.2 Erreurs relatives des solutions approchées de FPTAS 133

Liste des Algorithmes

3.1 Programmation dynamique 52

3.2 Borne supérieure de coût d'insatisfaction 57

3.3 Algorithme A_ε :FPTAS pour le problème PCT-FD 59

3.4 L'approche complète 60

5.1 Algorithme glouton $G1$ 107

5.2 Recherche locale 108

5.3 Algorithme glouton $G2$ 111

5.4 Opération de croisement 112

5.5 Opération de mutation 113

D.1 L'algorithme DP pour le problème P-Prob-P2(i) 137

Introduction générale

Alors que l'économie de marché est aujourd'hui plus que jamais la proie de la concurrence, chaque entreprise se doit de trouver un moyen de produire des biens d'une qualité définie, tout en restant à un prix qui soit le plus attractif possible. Dans cette démarche, toute entreprise est soumise à certaines contraintes. Si l'on considère que la demande fait l'offre, il peut alors s'agir de satisfaire cette dernière, mais aussi de contraintes légales, de moyens, ou encore liées au marché. L'optimisation est au coeur de toutes ces démarches et celle qui nous intéresse tout particulièrement aujourd'hui est liée aux problèmes de planification d'une chaine d'approvisionnement.

Il existe plusieurs définitions d'une *Chaîne Logistique* (*Supply Chain*) dans la littérature - voir, par exemple, Lee et Billington (1993) et Pochet et Wolsey (2006). Trois fonctions différentes peuvent êtres incorporées dans la définition de la chaîne. L'approvisionnement de matières premières, la fabrication et la distribution des produits finis. Il s'agit en définitif d'une chaine d'entreprises inter-connectées ayant pour but de transformer des matières premières en produits qui pourront être consommés par les utilisateurs finaux. Une chaîne logistique se compose donc de fournisseurs, de fabricants, de clients et d'autres personnes qui leur sont associés (voir figure 1).

FIGURE 1 – Chaîne logistique

La gestion d'une telle chaine ou *Supply Chain Management* (SCM), est la pratique consis-

tant à optimiser tous les composants de la chaine logistique dans un seul but, celui de livrer aux clients, en temps et en heure, des produits d'une qualité donnée au meilleur prix. Pour plus d'informations, voir Giard (2003) et Axsäter (2006).

Les diverses décisions concernant l'optimisation de la Chaine Logistique peuvent être classées en plusieurs niveaux : stratégique, tactique, et opérationnel.

Les décisions *stratégiques* sont des décisions à long terme, elles concernent particulièrement l'implantation, la localisation des entrepôts, la recherche de fournisseurs, ou encore les stratégies de production.

Les considérations *tactiques* ont quant à elles une portée à moyen terme, et se reportent plus particulièrement à la planification de production, de distribution et de transport afin de réduire le coût final.

Et enfin le niveau opérationnel qui prend en compte les décisions stratégiques et tactiques, permet de s'adapter aux différents imprévus. Il s'agit ici d'ordres d'approvisionnement, d'ordonnancement, etc.

Ce mémoire est consacré à l'étude des problèmes survenant aux niveaux tactique et opérationnel, plus précisément aux problèmes de lotissement et d'ordonnancement, lorsque l'on définit les volumes de lots à fabriquer et leur ordre de fabrication.

Cette thèse est organisé en 5 chapitres.

Dans le chapitre 1, nous présentons la problématique générale de gestion des systèmes de production en décrivant brièvement les principales étapes du processus de production. Nous nous intéressons particulièrement aux problèmes de planification de la production, notamment aux problèmes de lotissement et d'ordonnancement. Ce chapitre contient également un état de l'art sur les travaux qui traitent de problèmes de lotissement et d'ordonnancement. Nous présentons les modèles connus en les classant en fonction du nombre de produits à fabriquer, de la structure de la ligne de production, et de divers facteurs aléatoires à prendre en compte.

Le chapitre 2 formule les principales hypothèses du problème de lotissement et de séquencement sous aléas que nous allons étudier par la suite. A l'aide d'une simulation nous montrons les gains possibles en temps et en niveau de service lorsqu'une optimisation est faite.

Les trois chapitres suivants traitent d'un problème de lotissement et de séquencement en utilisant des hypothèses différentes et s'appuyant sur des approches différentes, notamment quant à la manière d'appréhender les aléas.

Le chapitre 3 étudie un problème de séquencement et de dimensionnement de lots dans

le cas où la ligne de production est composée d'une seule machine. Deux types d'aléas sont pris en compte : le rendement aléatoire et le temps d'exécution aléatoire. Les temps de changement de série dépendant de la séquence sont également pris en compte. Nous étudions le problème avec deux fonctions objectifs : le 1er objectif est de minimiser le temps total de fabrication, à condition que toutes les demandes soient satisfaites (problème P-Temps) ; le second, est de minimiser le coût total linéaire d'insatisfaction de la demande, sous la condition que l'horizon de planification soit limité (problème P-Coût). Nous montrons que les décisions de séquencement et de lotissement peuvent être prises séparément dans les deux cas. Nous fournissons une démonstration de NP-difficulté de ces deux options du problème, quand la quantité de pièces défectueuses est aléatoire. Si le rendement et le temps total de réparation sont des fonctions connues de la quantité de pièces lancées en fabrication, nous montrons que le problème P-Temps peut être résolu en temps polynômial, alors que le problème P-Coût reste NP-difficile. Nous présentons une approche FPTAS pour résoudre un cas particulier du problème P-Coût.

Dans le chapitre 4 nous reprenons le cas particulier du problème P-Coût considéré dans le chapitre précédent et présentons deux modèles de programmation linéaire en variables mixtes. Les résultats d'expérimentations numériques et une comparaison des méthodes (avec l'approche FPTAS également) sont fournis. Ensuite nous montrons que FPTAS ainsi qu'un des modèles linéaires peuvent être généralisés pour un cas de problème P-Coût plus général.

Le chapitre 5 s'intéresse au même type de problème de planification, mais dans le cas où les variables aléatoires sont données par leurs lois de distribution (problème P-Prob). L'objectif est de trouver la séquence et les tailles de lot dans le but de maximiser le niveau de service pour un horizon de planification donné. Pour résoudre ce problème NP-difficile, nous utilisons une décomposition le ramenant à trois sous-problèmes : une énumération, un problème de voyageur du commerce et le problème de sac à dos. Nous nous intéressons à la résolution de la partie lotissement (problème P-Prob-P2(i)), c'est-à-dire du problème de sac à dos correspondant. Pour le résoudre nous proposons deux méthodes approchées : une Recherche Locale et un Algorithme Génétique. Une comparaison de ces méthodes, ainsi que leurs comparaisons avec une méthode exacte est présentée à la fin de ce chapitre.

Enfin nous présentons une conclusion sur le travail effectué et quelques perspectives de notre recherche.

Chapitre 1

Travaux antérieurs et objectifs de l'étude

Dans ce chapitre nous présentons un état de l'art concernant les problèmes de lotissement et d'ordonnancement. Nous allons commencer par expliquer le fonctionnement d'un système de gestion de production en faisant un focus sur le problème de planification de production. Nous donnons un exemple de classification selon différents paramètres qui peut être appliqué aux problèmes de ce type. Ensuite nous présentons des modèles classiques et puis citons les publications en les classant en fonction des hypothèses utilisées. Ce chapitre est organisé en trois parties : dans la première, nous présentons des problèmes de lotissement, dans la deuxième partie des problèmes d'ordonnancement sont analysés, la troisième partie est consacrée aux travaux où les décisions de lotissement et d'ordonnancement doivent être prises ensemble.

1.1 Gestion de Production

La problématique de la gestion des systèmes de production suscite de nombreux travaux de recherche. Les chercheurs essayent de trouver des formulations pour des problèmes industriels afin de proposer des solutions nouvelles et efficaces, lesquelles arriveront par la suite jusqu'aux entreprises pour contribuer à l'amélioration de leurs performances.

Globalement, un système de gestion de la production est organisé de la façon suivante (voir Figure 1.1). L'étape de ***description*** comprend la définition de la structure physique

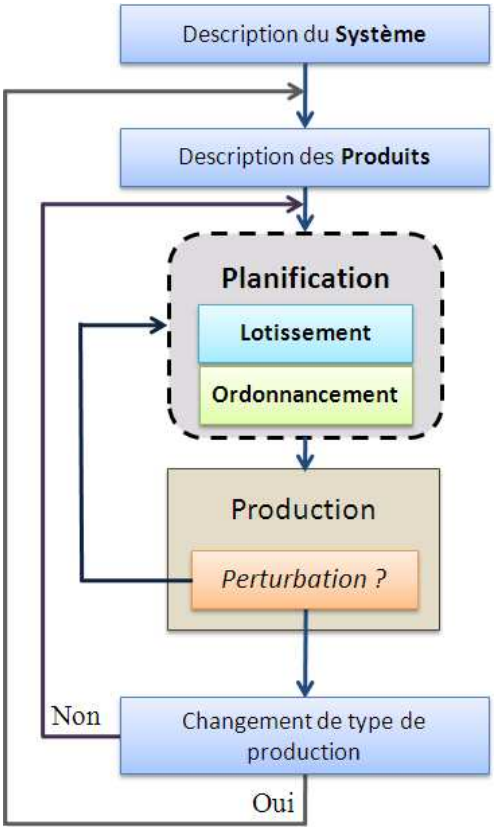


FIGURE 1.1 – Gestion de la production

du système de production, c.-à-d. la configuration de la ligne de production, le nombre de produits, les ressources utilisées, etc. Par description des produits, nous entendons les informations sur les gammes opératoires de divers produits, c.-à-d. des séquences d'opérations, la durée de traitement de produit de chaque type par chaque machine, ou bien encore l'ensemble des machines sur lesquelles chaque opération peut être effectuée.

La ***planification*** tient compte du carnet de commandes, du niveau des stocks (si l'on ne travaille pas en Juste-A-Temps) et des pièces en cours de fabrication. La planification de la production concerne le dimensionnement des lots des différents produits à fabriquer, le choix des périodes où ces lots doivent être produits, des machines sur lesquelles la production doit avoir lieu, l'affectation des tâches aux machines, etc.

La stratégie *Juste-à-temps* (JAT) est de n'avoir que les produits nécessaires en quantité demandée au moment voulu. JAT est un ensemble de techniques logistiques avec pour but d'améliorer la productivité globale. Le principe est que la production est amorcée par la demande.

Les problèmes ***d'ordonnancement*** et de ***lotissement*** font partie (ou sont des sous-

problèmes) de la problématique de la gestion de production. Souvent un ou plusieurs critères sont optimisés, par exemple :

- minimiser le temps total de fabrication (*Makespan*) ;
- minimiser le temps de retard (moyen, maximum) et (ou) d'avance ;
- minimiser divers coûts : coût de fabrication, de set-up, de stockage, de transport, etc.

L'ordonnancement d'un atelier est un des problèmes classiques de gestion de production. Il consiste à organiser dans le temps le fonctionnement d'un atelier (la réalisation de tâches en fonction des diverses contraintes). Un cas particulier d'ordonnancement est le séquencement, qui consiste à trouver une séquence de passage de pièces dans un système de production.

Les méthodes de dimensionnement des lots (*lotissement*) sont développées pour pouvoir décider si les pièces d'un produit doivent être fabriquées à un moment donné, et le cas échéant, en quelle quantité. Un *Lot* est un ensemble de produits du même type, traités successivement dans un système de production.

Lorsque la ***production*** est lancée, il faut essayer de respecter au mieux ce qui a été décidé lors des phases de planification et d'ordonnancement. Cependant, dans la vie réelle, l'exécution d'un projet ne peut pas toujours être en parfaite conformité avec le planning en raison de divers événements aléatoires. Souvent les événements aléatoires sont liés aux problèmes mécaniques d'atelier comme des pannes machines ou une maîtrise insuffisante de procédés, comme les rebuts, etc. Dans le cas d'une indisponibilité momentanée d'une ressource, la répartition du travail parmi les autres ressources peut se faire grâce à l'ordonnancement dynamique. Dans d'autres cas, plus graves, les aléas peuvent générer des problèmes complexes, et il peut être nécessaire de faire une nouvelle planification.

La complexité d'un problème de planification dépend directement du ***système de production*** considéré, par conséquent, la classification de ces problèmes est principalement basée sur les caractéristiques et paramètres des systèmes de production. Ci-dessous, nous avons décrit les plus importants d'entre eux.

Le système de production peut être *mono-niveau* ou *multi-niveaux*. Dans le premier cas, le produit final est obtenu directement après traitement de la matière première, c.-à-d. un seul stade de production est nécessaire. L'obtention d'un produit final dans un système multi-niveaux demande plusieurs stades de fabrication.

L'horizon de planification est la période de temps, durant laquelle la fabrication de l'intégralité des pièces nécessaires sera effectuée. L'horizon de planification peut être *fini* ou *infini*. Dans la plupart des cas, si l'horizon de planification est fini, il est divisé en plusieurs

périodes. Il y a deux types de problèmes en fonction de la durée des périodes : *small bucket* et *big bucket*. Dans le premier cas, une seule pièce peut être fabriquée lors d'une période, tandis que dans le second cas, la durée des périodes étant assez importante, il est donc possible de produire plusieurs articles au cours d'une période. Si l'horizon de planification n'est pas divisé en périodes, nous avons à faire à un problème *mono*-période.

Il existe deux types de ***demande*** : *statique* et *dynamique*. Une demande statique implique que sa valeur reste invariable durant tout l'horizon de planification, alors que la valeur d'une demande dynamique peut varier d'une période à l'autre. Si la valeur de la demande est connue d'avance, nous avons affaire avec une demande *déterministe*, dans le cas contraire à une demande *stochastique*. Lorsqu'il est question de la demande stochastique, la distribution de la probabilité est souvent connue à partir d'une analyse des demandes pour les périodes passées.

Quand il n'y a pas de limitation de ressources, nous avons affaire à un problème dit sans *contrainte de capacité*. Il est possible d'avoir des limites de capacité du stock, de disponibilité des machines, du volume de fabrication par période, des matières premières, etc. Dans ce cas, les contraintes de capacité doivent absolument être respectées.

Le *nombre de produits finis* dans le système de production est également une caractéristique importante. De ce point de vue, il y a deux types de systèmes de production : *mono*- et *multi*-produits.

Souvent, quand on passe de la fabrication d'un produit à l'autre, un coût et un temps *de changement de série* (*set-up*) non négligeables sont ajoutés. Généralement, c'est le temps (coût) nécessaire pour la préparation des machines pour la fabrication des composants d'un nouveau type, l'adaptation des équipements, etc. Souvent les problèmes de planification doivent tenir compte du temps (coût) de changement de série.

Dans cette thèse nous étudions un problème de planification sous incertitudes (voir chapitre 2). Le problème consiste à définir les quantités optimales de produits à traiter (tailles de lots) et l'ordre de passage des lots dans une ligne de production afin d'optimiser un critère. Deux types d'incertitudes sont prises en compte : les aléas de rendement et le temps d'exécution aléatoire. Dans le reste de ce chapitre nous fournirons un état de l'art traitant divers problèmes de lotissement et ordonnancement (déterministes et stochastiques).

1.2 Lotissement

Depuis les travaux fondateurs de Wagner et Whitin, et Manne en 1958, plusieurs chercheurs se sont intéressés aux problèmes de lotissement sous ces différentes formes, en proposant des modélisations, des formulations ainsi que des méthodes de résolution.

En général, l'objectif d'un problème de lotissement est de déterminer la quantité de pièces de chaque type (s'il y en a plusieurs) à fabriquer dans chaque période (dans le cas multi-période) en tenant compte de la demande, des contraintes de capacité, de disponibilité, etc.

Problèmes Mono-période

L'un des plus anciens et fondamentaux modèles de lotissement est le *Quantité Économique de Commande* (EOQ) introduit par Harris en 1913. Le but de ce problème est de trouver la quantité de commande Q, qui minimise les coûts totaux de possession et de commande. Les hypothèses de modèle EOQ sont les suivantes :

- Production instantanée (*capacité infinie*, i.e. le lot entier est fabriqué simultanément) ;
- Disponibilité immédiate ;
- Demande déterministe et constante ;
- Coût de lancement est constante ;
- Il y a un seul type de produit en stock (mono-produit) ;
- Horizon de planification est infini.

Les notations pour ce modèle sont les suivantes : D est la demande annuelle, s est le coût de lancement, h est le coût de stockage (pour une unité de produit et pour une année). Pour trouver la valeur de Q optimale, il est nécessaire de minimiser le coût moyen annuel total :

$$Y(Q) = s\frac{D}{Q} + h\frac{Q}{2} \tag{1.1}$$

où sD/Q est le coût de lancement et $hQ/2$ est le coût de possession du stock. Figure 1.2 montre une représentation graphique de ces coûts. Pour résoudre ce problème il faut mettre la dérivée de la fonction de coût (1.1) à zéro. Nous obtenons :

$$Q = \sqrt{2sD/h} \tag{1.2}$$

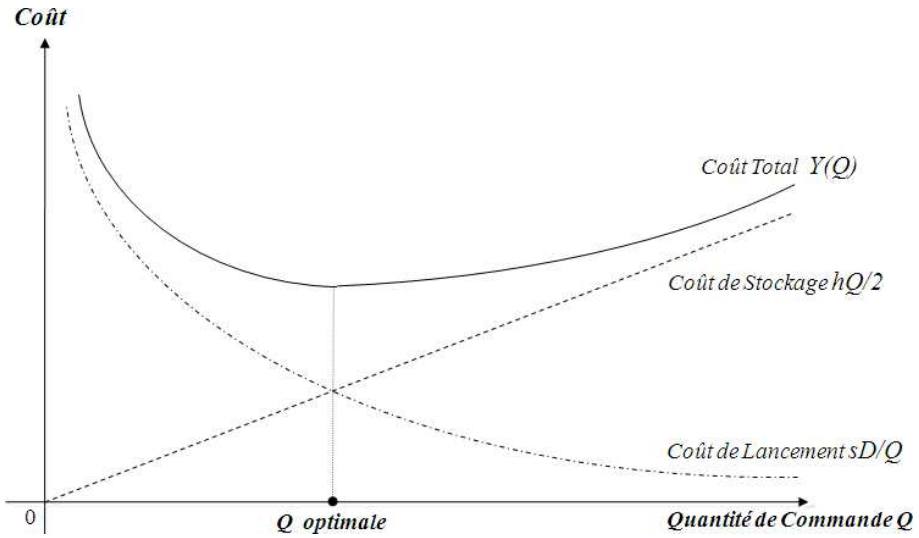


FIGURE 1.2 – Les coûts de modèle EOQ

Le problème EOQ a été beaucoup étudié dans la littérature. Il existe des nombreux articles contenants de différentes méthodes pour trouver la quantité économique. Par exemple, Grubbstrom (1996) a démontré que le problème EOQ peut être résolu en utilisant les dérivées algébriques, ce qui permet d'éviter l'utilisation des dérivées et ouvre l'approche à d'autres fonctions coûts. L'extension du problème EOQ le plus étudié est EOQ avec *backorders* ou ruptures de stock. Dans ce cas, par rapport au modèle EOQ, deux nouvelles notations apparaissent : le coût unitaire de rupture b et le nombre maximal de ruptures B (*variable de décision*). La fonction objectif peut être représentée de la manière suivante :

$$Y(Q,B) = s\frac{D}{Q} + h\frac{(Q-B)^2}{2Q} + b\frac{B^2}{2Q} \tag{1.3}$$

où $h\frac{(Q-B)^2}{2Q}$ est le coût total de stockage et $b\frac{B^2}{2Q}$ est le coût total de rupture. Grubbström et Erdem (1999) ont démontrés que la méthode des dérivées algébriques peut être utilisée pour résoudre cette extension du problème.

La *Quantité Économique de Production* (EPQ) est aussi une des extensions du modèle EOQ. Elle a été suggérée par Taft en 1918. La différence entre les deux modèles est le fait que les demandes dans le problème EPQ sont reçues progressivement au cours du processus de fabrication. Comme pour le problème EOQ, nous cherchons un équilibre entre le coût de possession et le coût de production. En utilisant des notations du modèle EOQ, le coût

total est représenté par l'expression suivante :

$$Y(Q) = s\frac{D}{Q} + h\frac{Q}{2}(1-r) \tag{1.4}$$

où r est le taux de production. La solution optimale du problème est :

$$Q = \sqrt{\frac{2sD}{h}}\sqrt{\frac{r}{r-D}} \tag{1.5}$$

Cárdenas-Barrón (2001) applique la même méthode (par dérivées algébriques) pour trouver la solution optimale du problème EPQ avec ruptures. Ronald *et al.* (2004) et Chang *et al.* (2005) proposent des améliorations pour les méthodes d'optimisation de EOQ et EPQ avec ruptures. Une approche similaire peut être trouvée dans Wu et Ouyang (2003) pour le problème de vendeur-acheteur intégré, le but est d'obtenir la quantité économique optimale pour l'acheteur et la quantité optimale des livraisons pour le vendeur.

Minner (2007) propose une autre méthode pour résoudre le problème EOQ. Elle est basée sur la comparaison des coûts pour les tailles de différents lots. Il a également démontré que sa méthode peut être appliquée pour les EOQ et EPQ avec backorders. Teng (2009) décrit encore une autre approche pour résoudre les problèmes ci-dessus, en utilisant une inégalité arithmético-géométrique. Cette méthode part du principe que la moyenne arithmétique est supérieure ou égale à la moyenne géométrique.

Les modèles EOQ et EPQ ont plusieurs hypothèses simplificatrices. Néanmoins, ces modèles forment une bonne base pour faire face aux problèmes réels. Nous trouvons également un grand nombre d'extensions et modifications de ces modèles dans la littérature.

Problèmes Multi-période

Les problèmes EOQ et EPQ ont une demande constante et un horizon de planification infini. En revanche, le modèle de *Wagner-Whitin* (problème de lotissement dynamique) suppose la demande **dynamique** et l'horizon de planification T **fini** et divisé en **plusieurs périodes** (jours, semaines, mois) : $t = 1, 2, \ldots, T$. Les notations restent les mêmes, sauf que les variables sont définies pour chaque période t, $t = 1, 2, \ldots, T$:

- Coût de lancement s_t ;
- Coût de stockage h_t ;
- Demande d_t pour la période t.

Le stock initial I_0 est nul. I_t est le niveau de stock à la fin de la période $t, t = 1, \ldots, T$. Le but est de trouver les quantités de pièces à fabriquer $(Q_1, Q_2, \ldots, Q_T)$ pour toutes les périodes afin de minimiser le coût total :

$$\begin{gathered} \sum_{t=1}^{T} (s_t Q_t + h_t I_t) \\ I_t = I_{t-1} + Q_t - d_t, \quad t = 1, 2, \ldots, T, \\ I_0 = 0, \quad Q_t,\ I_t \geq 0. \end{gathered} \tag{1.6}$$

En générale, trois types d'approches peuvent être appliquées pour l'optimisation :

- l'utilisation du modèle EOQ avec le niveau moyen de la demande, c.-à-d. $D' = \frac{\sum_{t=1}^{T} d_t}{T}$;
- l'approche de Wagner-Whitin qui permet de trouver une solution exacte pour certains cas spécifiques ;
- une méthode approchée, qui arrive à trouver des solutions de bonne qualité tout en restant relativement simple.

L'algorithme de Wagner-Whitin est l'approche par programmation dynamique en utilisant deux propriétés démontrées par Wagner et Whitin. Cet algorithme est largement utilisé dans la littérature pour résoudre les problèmes de lotissement dynamique. Evans (1985) a proposé une implémentation informatique efficace de l'algorithme. On peut trouver quelques autres exemples d'applications de l'algorithme dans Wagelmans *et al.* (1992), Potamianos et Orman (1996), Potamianos *et al.* (1997). Fordyce et Webster (1984) ont proposé une explication non-mathématique de l'algorithme de Wagner-Whitin.

Comme la plupart des approches exactes, l'algorithme de Wagner-Whitin a des difficultés pour résoudre des problèmes de grandes tailles (voir, par exemple, Saydam et Evans (1990) et Jeunet et Jonard (2000)), alors plusieurs approches heuristiques ont été développées. La plus connue parmi elles est l'heuristique de Silver-Meal. C'est une méthode qui cherche à minimiser le coût moyen par période en fonction du nombre de périodes. Pour plus de détails voir, par exemple, Omar et Deris (2001) et Hu *et al.* (2004). Nous pouvons mentionner aussi l'approche *Lot-pour-Lot*, où chaque commande comprend la quantité exacte des pièces nécessaires pour chaque période.

Si nous ajoutons une contrainte de capacité finie de production, nous obtenons le problème de lotissement à capacité finie (CLSP). Ce problème est également un problème à plusieurs périodes. Le but est de trouver un plan de production (les types et les quantités de produits X_{it} à fabriquer pour chaque période) afin de minimiser le coût total. La demande d_{it} pour

chaque type de produit $i,\ i = 1, \ldots, n$ et pour chaque période $t,\ t = 1, \ldots, T$ est donnée. Nous connaissons également R_t la capacité disponible pour la période $t,\ t = 1, \ldots, T$. Un coût de lancement fixe s_{it} et un coût de production linéaire p_{it} sont connus ainsi que le coût de stockage h_{it}, proportionnel à la quantité de pièces en stock et au temps de stockage. Remarquons que malgré le fait que le coût de lancement (set-up) s_{it} dépend du type de produit et de la période, il est indépendant de la séquence des produits, ce qui n'est pas toujours le cas dans les problèmes réels. Les notations additionnelles pour ce problème sont les suivantes :

- stock I_{it} des produits de type i à la fin de la période t (on suppose que $I_{i1} = I_{iT} = 0$) ;
- variables binaires Y_{it} telles que

$$Y_{it} = \begin{cases} 1, \text{ si le produit } i \text{ est traité pendant la période } t \\ 0, \text{ sinon} \end{cases}$$

- borne supérieure M_{it} pour la production du produit i dans la période t,

$$M_{it} = \sum_{k=t}^{T} d_{ik}$$

- consommation de ressource a_i pour le produit i.

En prenant en compte tous les paramètres définis ci-dessus, le problème CLSP peut être formulé comme suit :

$$\text{Minimiser} \sum_{i=1}^{n} \sum_{t=1}^{T} s_{it} Y_{it} + p_{it} X_{it} + h_{it} I_{it} \tag{1.7}$$

$$\begin{gathered} \text{s.c.} \sum_{i=1}^{n} a_i X_{it} \leq R_t, \quad t = 1, \ldots, T, \\ X_{it} + I_{i,t-1} - I_{it} = d_{it}, \\ X_{it} \leq M_{it} Y_{it}, \\ Y_{it} \in \{0, 1\},\ X_{it} \geq 0,\ I_{it} \geq 0, \\ i = 1, \ldots, n, \quad t = 1, \ldots, T \end{gathered}$$

Le problème CLSP est NP-difficile, voir Bitran et Yanasse (1982) et Chen et Thizy (1990). Une analyse détaillée de ce problème avec un état de l'art sur les méthodes de résolution se

trouve dans Karimi *et al.* (2003). Étant donné que le problème est NP-difficile, la plupart des méthodes sont des heuristiques.

Pour terminer, nous allons citer un travail récent sur une extension de ce problème. Nascimento *et al.* (2010) abordent le problème CLSP avec plusieurs sites de production indépendants où les transferts entre les sites sont autorisés. Ils proposent une approche de type GRASP (Greedy Randomized Adaptive Search Procedure) ainsi qu'une procédure de "*path-relinking*" pour trouver des solutions intéressantes.

Problèmes de lotissement avec incertitudes

Le modèle de lotissement probabiliste le plus connu est le modèle de *Newsboy* ou *News-Vendor*. Dans ce modèle un paramètre incertain - la demande - apparaît. Dans la version classique du problème, un agent (newsvendor ou newsboy) achète une certaine quantité de produits (magazines) au début de la période (le matin, par exemple). Pendant la journée, il vend les produits sur le marché. La quantité de journaux (demande) vendus est aléatoire. Alors si le vendeur a surestimé la demande, il perd de l'argent pour les journaux non vendus, sinon il y a les opportunités perdues (manque à gagner). Le problème est de trouver la quantité X de produits optimale à acheter. Les hypothèses de ce problème sont les suivantes :

- Une seule période ;
- Un seul produit ;
- Demande D incertaine avec une loi de distribution connue avec une densité de probabilité $f(D)$ et la fonction de répartition $F(D)$;

La fonction du profit de vendeur est la suivante :

$$\pi = \begin{cases} (P^v - P^a)X - P^p(D - X), \text{ si } D \geq X \\ P^v D + P^s(X - D) - P^a X, \text{ si } D < X \end{cases} \tag{1.8}$$

où P^v est le prix de vente d'une unité, P^a est le prix d'achat, P^p est le coût de pénurie par unité manquée et P^s est le prix unitaire de surplus (le prix auquel il peut rendre les invendus). Il existe plusieurs extensions de ce modèle, Khouja (1999) les a classifié dans 11 catégories en fonction de : objectif, politique de prix, rendement aléatoire ou non, information connue sur la demande, quantité de produits, nombre de niveaux, de périodes, etc. Dans le cadre de cette thèse, nous nous intéressons aux cas avec plusieurs produits et rendements aléatoires

(quantité de journaux qu'on obtient en commandant une quantité donnée).

En pratique, la plupart des problèmes industriels comprend plusieurs produits et souvent il faut prendre en compte des contraintes de capacité. Ainsi dans l'article de Lau et Hing-Ling Lau (1995) une contrainte de capacité est introduite, c.-à-d. il y a une limite sur la quantité des journaux que l'agent peut acheter chaque jour. Les auteurs ont proposé une procédure basée sur "active set" méthode. Erlebacher (2000) a développé trois heuristiques pour le même extension du problème de Newsboy. Dans la publication de Abdel-Malek *et al.* (2004), nous trouvons comment obtenir la solution exacte, si la fonction de densité de la probabilité est uniforme et une méthode itérative qui permet de trouver une solution optimale (ou proche de l'optimum) pour le cas d'une fonction de densité générale continue. Ces idées ont été reprises et développées dans Abdel-Malek et Montanari (2005). Une autre méthode de résolution *binaire* peut être trouvée dans Zhang *et al.* (2009) pour une fonction de densité de la demande quelconque. Silver et Moon (2001) considèrent une extension avec un stock initial constituant des pièces convertibles, c.-à.-d. ces pièces peuvent être transformées dans n'importe quel type de produit final.

Le *rendement aléatoire* est un autre type d'incertitude, qu'il faut prendre en compte pour le lotissement. Cette incertitude complique considérablement la planification, car la quantité de produits en entrée et en sortie n'est pas la même et cette différence n'est pas connue avec certitude et peut conduire, par exemple, à des retards de livraison qui entraînent des pénalités. Yano et Lee (1995) présente une revue de la littérature sur les problèmes de lotissement avec un rendement aléatoire.

Un autre problème largement étudié dans la littérature est le *Problème de Lotissement Multiple* (*Multiple Lot-Sizing Production to Order* - MLPO). Grosfeld-Nir et Gerchak (2004) fournissent une revue des modèles et des résultats analytiques pour MLPO. Pour ce problème le rendement est aléatoire et la demande doit être satisfaite intégralement, alors plusieurs lancements peuvent être nécessaires pour l'atteindre. Il faut donc trouver la taille du lot initial X minimisant le coût total de production. Il existe quelques hypothèses générales pour le rendement Y_X : la probabilité que $Y_X < X$ est égale à 1 et Y_X est une fonction croissante de X. Les autres notations et hypothèses sont donc suivantes :

- Un seul type de produit ;
- La demande D est fixée ;
- Le coût de set-up s est connu ;
- Le coût unitaire de production p est également connu ;

- $P(y, n)$ est la probabilité que le rendement Y_n égal à y, si la taille de lot lancée est n, $P(y, n) = 0$ si $y > n$.

Il est supposé que pour chaque nouveau lancement nous avons un coût de set-up, alors le coût de production d'un lot de taille X est égal à $s + pX$. Si, après le lancement de X composants, le rendement Y_X est inférieur à D, le problème se répète avec la demande restante $D' = D - Y_X$. Dès que la production totale est supérieure ou égale à D, le processus s'arrête.

Le coût total prévu peut être calculé à l'aide de l'équation suivante :

$$V_D(n) = s + pn + P(0, n)V_D(n) + \sum_{y=1}^{D-1} P(y, n)V_{D-y} \tag{1.9}$$

où $n \geq 1$, $V_D = \min_n V_D(n) \equiv V_D(n_D)$ pour $D \geq 1$, et $V_D \equiv 0$ pour $D \leq 0$. La taille de lot optimale et le coût peuvent être calculés par une récursion.

Il existe plusieurs modèles pour estimer un rendement aléatoire. Les plus utilisés sont : le processus de Bernoulli, le Rendement stochastique proportionnel et le Rendement Interrompu géométrique :

1. Le processus de Bernoulli. Supposons sans perte de généralité que nous avons un seul type de produits à fabriquer. Pour utiliser ce modèle il faut que la probabilité p, qu'une pièce est de bonne qualité après sa fabrication soit connue et soit la même pour toutes les pièces. La dernière hypothèse signifie que la fabrication d'une pièce est complètement indépendante de celle des autres. Considérant cela, nous pouvons en déduire les équations suivantes :

- la probabilité d'avoir x pièces de bonne qualité à partir de n pièces traitées (loi *binomiale* $B(n, p)$) :
$$P(x, n) = \binom{n}{x} p^x (1 - x)^{n-x}$$
- la probabilité d'avoir x pièces de bonne qualité quand la quantité de rebut est égale à r (loi *binomiale négative* $NB(r, p)$) :
$$P(x, r) = \binom{x + r - 1}{r - 1} (1 - p)^r p^x \quad \text{for } r = 0, 1, 2, \ldots$$
- le nombre de pièces x à lancer afin d'avoir une pièce de bonne qualité (distribution *géométrique* $NB(1, p)$) :
$$P(x) = (1 - p)^{x-1} p$$

Voir des exemples dans Singh *et al.* (1988), Teunter et Flapper (2003), Dolgui *et al.* (2005).

2. Le rendement stochastique proportionnel est le cas le plus utilisé dans la littérature. Dans ce cas le pourcentage α de pièces défectueuses (ou de pièces de bonne qualité) est connu et est le même quelle que soit la taille du lot. Si, par exemple, X est la taille du lot, le rendement Y_X (c.-à-d. le nombre de pièces de bonne qualité) est :

$$Y_X = \alpha X$$

où α est la fraction aléatoire des pièces de bonne qualité, indépendante de X. La valeur α a une densité de probabilité $g(\cdot)$, une fonction de répartition $G(\cdot)$ et une moyenne μ.

3. Le rendement Interrompu géométrique décrit un processus de dégradation. Avec une probabilité de $(1-\theta)$ le processus passe *hors de contrôle*. Si le processus est hors de contrôle, la pièce traitée et toutes les pièces suivantes de ce lot sont défectueuses. Autrement dit, θ représente la fiabilité du processus de production. En utilisant ce modèle de modélisation nous pouvons obtenir la probabilité d'avoir y pièces de bonne qualité :

$$P(y,n) = (1-\theta)\theta^y, \quad y = 0, 1, 2, \ldots, n-1,$$

dans le cas $y = n$, $P(n,n) = \theta^n$.

Salameh et Jaber (2000) ont étendu les modèles EPQ/EOQ pour pouvoir tenir compte des articles de qualité imparfaite. Ils montrent que la Quantité Économique de Commande a tendance à augmenter quand le pourcentage des composants imparfaits augmente. Maddah et Jaber (2008) proposent un nouveau modèle pour le même problème en utilisant la théorie de renouvellements. Ce modèle permet d'avoir des expressions analytiques pour le profit espéré par unité de temps et pour la quantité optimale. Dans l'article (Eroglu et Ozdemir (2007)), un modèle EOQ avec des pièces défectueuses et des ruptures de stock a été examiné. Le taux de composants défectueux dans un lot est une variable aléatoire uniformément distribuée. L'influence du pourcentage de pièces défectueuses sur la solution optimale a été étudiée. D'autres exemples de modélisation des aléas peuvent être retrouvés dans : Yao (1988), Grosfeld-Nir et Gerchak (1990), Gerchak *et al.* (1994), Agnihothri *et al.* (2000), Wang et Gerchak (2000) , Haji *et al.* (2008), Li *et al.* (2008), etc.

Zhang et Guu (1998) étudient le problème de lotissement multiple (MLPO) avec un rendement suivant une loi de probabilité interrompue géométrique. Ils étudient en particulier le comportement de la fonction coût et des tailles de lots optimales. Guu et Liou (1999) travaillent sur le même problème et proposent un algorithme pour déterminer des tailles de

lots optimales. Ils démontrent que leur approche est plus efficace que celle de Zhang et Guu (1998). Dans le papier (Guu et Zhang (2003), l'étude précédente (Zhang et Guu (1998) a été élargie en ajoutant deux facteurs : les coûts de possession de stocks et un nombre fini de set-up. Grosfeld-Nir et Gerchak (1996) ont étudié pour le problème MLPO les propriétés de la solution optimale en fonction de valeurs de paramètres et de la façon avec laquelle les aléas sont modélisés.

Un exemple d'utilisation de la *loi uniforme discrète* $P(y, n) = \frac{1}{n+1}$ est présenté dans (Anily (1995). Plusieurs auteurs proposent des modèles avec une distribution quelconque, c.-à-d. toute distribution peut être utilisée. Zhang et Guu (1997) examinent les propriétés de la fonction de coût pour un problème de lotissement multiple avec une demande constante et la distribution de probabilité quelconque pour le rendement.

Certains modèles conçus pour un rendement déterministe ont été généralisés pour le cas de rendement aléatoire. Par exemple, le problème de Newsvendor avec le rendement aléatoire a été considéré dans (Abdel-Malek *et al.* (2008). Les auteurs ont appelé cette extension *Gardener problem.* Un rendement aléatoire peut être expliqué par le fait que le newsboy reçoit moins de journaux qu'il a commandé en raison de la météo. Dans les articles (Eroglu et Ozdemir (2007) et (Maddah et Jaber (2008) le modèle EOQ avec les pièces défectueuses et ruptures a été étudié. Liao *et al.* (2009) et Salameh et Jaber (2000) considèrent le problème EPQ avec une production imparfaite. Gerchak *et al.* (1994) propose une formulation pour le lotissement dans un système d'assemblage avec un rendement aléatoire. Les auteurs considèrent deux cas : avec les lois de probabilité de rendement identiques et non.

Les pannes "machines" sont une autre source d'aléas. Si une machine peut tomber en panne, le temps de bon fonctionnement devient aléatoire. Généralement, le temps de réparation est également aléatoire. Le temps total de réparation peut être représenté par une fonction du temps de production. Lin et Gong (2006) examinent l'effet des pannes machines sur le modèle de *Economic Production Quantity* (EPQ). Dans leur article, le temps de réparation après chaque panne est constant, mais le temps entre deux pannes est une variable aléatoire avec une loi de distribution exponentielle négative.

Souvent les problèmes considérés dans la littérature contiennent plusieurs types d'incertitudes à la fois. Par exemple, Gerchak *et al.* (1988) étudient un modèle de production avec un rendement et une demande aléatoires. Les auteurs proposent une analyse complète du problème pour une seule période, des propriétés du problème dans le cas de deux périodes et une structure de solution pour les cas où le nombre de périodes serait supérieur à 2. Ils supposent que la demande pour chaque période est une variable aléatoire. Les demandes

pour différentes périodes sont indépendantes et identiquement distribuées (i.i.d.), avec une densité de probabilité $g(\cdot)$ et une fonction de répartition $G(\cdot)$ communes. Le rendement est de type stochastique proportionnel.

1.3 Ordonnancement

La théorie de l'ordonnancement est une branche de la recherche opérationnelle qui s'intéresse au calcul des dates optimales pour l'exécution de tâches. Il est très souvent nécessaire d'affecter en même temps les ressources nécessaires à l'exécution de ces tâches. Un problème d'ordonnancement peut être considéré comme un sous-problème de planification dans lequel il s'agit de décider de l'exécution opérationnelle des tâches planifiées. L'objectif des problèmes d'ordonnancement est d'optimiser une fonction (par exemple le temps ou le coût total de production). La classification des problèmes d'ordonnancement la plus utilisée dans la littérature est construite à partir de la configuration du système de production. Il existe cinq types principaux de configuration des systèmes de production : *machine unique*, *machines parallèles*, *flow-shop*, *job-shop* et *open-shop*.

La majorité des problèmes d'ordonnancement suppose la présence de temps ou (et) coûts de lancement (changement de série, installation ou set-up) non-négligeables. En général le changement de série comprend les travaux nécessaires pour la préparation d'une machine (ou d'une ligne entière) pour produire des pièces d'un certain type, l'adaptation des outils, et l'acquisition des matières. Un temps (coût) de changement de série avec batching a lieu lorsque des tâches sont traitées par lots (palettes, conteneurs, cartons) et un temps (coût) de set-up précède le traitement de chaque lot. Ici un lot est un ensemble des pièces (tâches) de la même famille. Deux tâches (pièces) appartiennent à la même famille si, par exemple, leurs temps de fabrication sont les mêmes pour toutes les machines de la ligne. Tandis que les familles sont censées être définies à l'avance, la formation de lots est une partie du processus de décision opérationnel. Le temps (coût) de set-up est *dépendant de la séquence*, si sa durée (coût) dépend du type de pièce qui doit être fabriqué et du type de pièce précédent. Le temps (coût) est *indépendant de la séquence*, si sa durée (coût) dépend uniquement du lot à fabriquer. Dans un traitement sans lot, un temps (coût) de changement de série est engagé avant le traitement de chaque pièce. Le modèle correspondant peut aussi être considéré comme un modèle avec lots, dans lequel chaque lot est composé d'une seule pièce.

Le papier d'Allahverdi *et al.* (1999) contient une revue de la littérature pour les problèmes d'ordonnancement avec le coût (temps) de set-up. Les auteurs classifient ces problèmes sur

la base de la configuration de systèmes de production et du type de set-up (dépendant ou indépendant de la séquence).

Dans le cas d'*une seule machine*, toutes les tâches (produits) doivent être traitées par cette machine. Le problème consiste donc à définir un ordre (séquence) des tâches pour le cas mono-période et à définir les tâches et leur ordre pour chaque période, pour le cas multi-périodes.

Les *machines parallèles* exécutent les mêmes opérations, alors chaque tâche peut être traitée par n'importe laquelle de ces machines. Le temps de traitement peut être différent pour différentes machines. Trois types de machines parallèles existent :

- Machines *Identiques* : le temps de traitement d'une tâche est le même quelle que soit la machine (voir Min et Cheng (1999), Yalaoui et Chu (2002), Lee *et al.* (2006)) ;
- Machines *Uniformes* : chaque machine a une vitesse de traitement ; la durée d'une tâche est proportionnelle à cette vitesse (voir Guinet (1995), Balakrishnan *et al.* (1999), Chou *et al.* (2006)) ;
- Machines *Indépendantes* : le temps du traitement de chaque tâche est propre pour chaque machine, et ils sont indépendants les uns des autres (voir dans Kim *et al.* (2002), Chen et Wu (2006)).

Un *Flow-shop* est une ligne de production où toutes les tâches doivent être traitées par toutes les machines, lesquelles doivent être visitées dans le même ordre par toutes les tâches. Le problème de minimisation du temps total de production sur flow-shop est NP-difficile, si le nombre de machines est plus grand que 2, sauf quelques cas particuliers avec trois machines.

Dans une ligne de production *Job-shop*, chaque produit a sa propre séquence d'opérations, c.-à-d. il n'y a pas d'obligation pour une pièce d'être traitée par toutes les machines de la ligne. En revanche, la séquence de passage est fixée pour chaque pièce. Les problèmes de type job-shop sont également NP-difficiles dans la plupart des cas. *Open-Shop* est un cas spécifique de job-shop : la différence est que chaque pièce doit être traitée par un ensemble de machines donné, mais l'ordre des opérations est absolument libre.

Nous nous intéressons aux problèmes du type machine unique et à des lignes de production du type flow-shop de permutation (séquence des tâches est la même pour toutes les machines). Les sous-sections suivantes contiennent un état de l'art pour ces deux types de problèmes.

Mono-machine

Un des problèmes de base en ordonnancement des tâches sur une seule machine (voir Szwarc (1996)) a les hypothèses suivantes :

- Il y a n tâches (*jobs*) ;
- La machine ne peut traiter qu'un seul job à la fois ;
- Chaque job k est disponible au temps zéro ;
- Le temps de traitement du job k est égal à t_k ;
- Tous les jobs ont la date d'échéance L commune et fixée ;
- L'avance ainsi que le retard d'un job sont pénalisés : α_k - coût d'avance et β_k coût de retard de job k, $\quad k = 1, \ldots, n$.

Le problème est de trouver les dates de fin $C_k, k = 1, \ldots, n$ de toutes les tâches afin de minimiser la *fonction de pénalité* :

$$\sum_{k=1}^{n} (\alpha_k E_k + \beta_k T_k) \tag{1.10}$$

où $E_k = \max(0, L - C_k)$ est l'avance et $T_k = \max(0, C_k - L)$ est le retard de la tâche k. La minimisation des coûts d'avance et de retard correspond à la stratégie *Juste-à-temps*.

Une revue des problèmes d'ordonnancement déterministes avec la date d'échéance commune pour les lignes de production contenant une seule machine ou des machines parallèles a été fournie par Baker et Scudder (1990) et Gordon *et al.* (2002). Notamment nous pouvons voir que le problème (1.10) pour certains cas est NP-difficile. Webster et Baker (1995) ont proposé une revue de la littérature sur l'ordonnancement avec lots sur une machine. Ils traitent trois modèles : l'ordonnancement avec disponibilité des pièces, l'ordonnancement avec disponibilité des lots et le traitement des lots. D'autres exemples du problème d'ordonnancement sur une seule machine peuvent être trouvés dans (Kacem (2010)) et (Kacem et Mahjoub (2009)).

Hoogeveen et Van de Velde (1996) proposent une approche de type *Branch-and-Bound* pour le problème ci-dessus pour le cas où chaque job a sa propre date d'échéance L_i, par contre les coûts d'avance α et de retard β, $\quad \beta > \alpha$ sont les mêmes pour tous les jobs. D'autres travaux sur ce problème peuvent être cités, comme par exemple Kim et Yano (1994), Sourd et Kedad-Sidhoum (2003), Sourd et Kedad-Sidhoum (2008), Kedad-Sidhoum et Sourd (2010), etc. Dans le dernier travail mentionné, les résultats de calcul montrent que des solutions de

bonne qualité, pour des problèmes avec plus de 100 tâches, peuvent être trouvées en quelques secondes.

Gupta et Smith (2006) examinent le problème de la planification sur une machine afin de minimiser le retard total avec le temps de changement de série dépendant de la séquence. Ils présentent deux algorithmes, une heuristique de type recherche locale et une *Greedy Randomized Adaptive Search Procedure* (GRASP). GRASP est une multi-start meta-heuristique souvent utilisée pour les problèmes d'optimisation combinatoire. Elle se compose de deux phases : une phase de construction, suivie d'une phase d'amélioration. Dans la phase de construction, une solution réalisable est trouvée. Dans la phase d'amélioration, il faut construire l'ensemble des solutions-voisines et trouver la meilleure parmi elles, qui sera la solution initiale pour l'itération suivante. La procédure s'arrête quand un optimum local est atteint. Ce processus (les deux phases) est répété un nombre de fois déterminé, et la meilleure solution trouvée est signalée comme la solution finale.

Asano et Ohta (1999) présentent un problème d'ordonnancement sur une machine avec contraintes de temps d'achèvement et les périodes d'inactivité. Les auteurs proposent un algorithme heuristique qui optimise les débuts et les fins de façon à minimiser la somme du coût de stockage et de rupture.

L'ordonnancement sur une machine unique est beaucoup discuté. Brucker et Kovalyov (1996) minimisent le nombre de jobs en retard ; Cheng et Kovalyov (2001) étudient le problème de minimisation de consommation totale des ressources, et le problème de minimisation du retard maximum, ils considèrent que le temps de set-up et le temps de fabrication dépendent des ressources ; Melouk *et al.* (2004) proposent une approche de recuit simulé pour minimiser le Makespan ; voir aussi Cheng et Kovalyov (1995), Cheng *et al.* (2001), Ng *et al.* (2004), etc.

Flow-Shop

Selon Bagchi *et al.* (2006), des problèmes d'ordonnancement dans un flow-shop peuvent être classifié en deux principales catégories : *cycliques* et *non-cycliques*. Dans les problèmes non-cycliques, des jobs doivent être ordonnés sur un horizon de planification afin d'optimiser un critère, tel que le makespan ou la somme des retards. L'ordonnancement cyclique est lié avec le terme *minimal part set* (MPS). Si, par exemple, à la fin de la période de planification nous avons besoin de 20 pièces de type 1 et 30 pièces de type 2, alors le MPS contient 2 pièces de type 1 et 3 pièces de type 2. Le processus de production est une séquence des MPS.

Chacune des deux catégories de flow-shops (cyclique et non-cyclique) peuvent être divisées en : *no-wait*, *blocking*, *buffer illimité*, *buffer limité*. Dans un flow-shop sans attente (no-wait), chaque job doit être traité dès le début jusqu'à la fin sans interruption sur une machine ou entre deux machines. Dans un flow-shop avec blocage (blocking), un job fini ne peut pas quitter la machine, où il a été traité jusqu'au moment où la machine suivante est libre. Les flow-shop des deux derniers types contiennent des buffers entre les machines, où des pièces peuvent être stockées.

Le problème de minimisation du Makespan (C_{max}) dans un flow-shop (sauf la situation quand $m = 2$) est NP-difficile (Garey et Johnson (1979)). Cheng *et al.* (2000) proposent une revue des articles sur l'ordonnancement des flow-shops et expliquent les difficultés dans les formulations et solutions de ces problèmes. *L'énumération complète*, les techniques de *Séparation et Évaluation* (Branch and Bound) ou la *Programmation dynamique* peuvent déterminer la séquence optimale pour des problèmes de petites tailles, mais des approches heuristiques sont nécessaires pour résoudre des problèmes de tailles plus importantes.

Lee (1997) étudie un problème d'ordonnancement d'un flow-shop avec deux machines avec une contrainte de disponibilité pour une des deux machines. Il considère le cas où les machines ne sont pas toujours disponibles, par exemple, en raison d'une panne ou d'un entretien préventif. Si au début d'une période, une machine continue à traiter les jobs inachevés dans la période précédente, elle est considérée comme non-disponible au début de cette période. Dans cet article, le problème a été simplifié : l'auteur a supposé que les périodes de non-disponibilité sont données. L'approche a été généralisée dans Lee (1999) pour le cas où la contrainte de disponibilité est imposée sur deux machines. Espinouse *et al.* (1999) ont démontré que le problème est NP-difficile même pour le cas d'une seule période d'indisponibilité sur une seule machine et qu'il est NP-difficile dans le sens fort si le nombre de périodes d'indisponibilité est aléatoire. Les auteurs ont également proposé un algorithme heuristique pour résoudre ce problème. Une autre heuristique pour le même problème avec une contrainte de disponibilité a été développée dans Cheng et Wang (2000), les auteurs démontrent que la borne supérieure d'erreur est $\frac{1}{3}$.

Nous nous intéressons aux problèmes d'ordonnancement dans un flow-shop sans attente. Les hypothèses du modèle de base sont les suivantes :

- Temps de fabrication t_{ij} des pièces sont déterministes et ils sont donnés à l'avance ;
- Traitement d'une pièce ne peut pas être interrompu ;
- L'ordre des opérations est fixé pour chaque pièce i, $i = 1, \ldots, n$, il est le même pour toutes les pièces ;

- Une pièce i peut ne pas être traitée par une machine j, si $t_{ij} = 0$;
- Une machine ne peut pas traiter plus qu'une pièce à la fois.

Les fonctions objectif peuvent être différentes, par exemple : la minimisation de la somme de tous les temps d'achèvement des jobs, la minimisation du Makespan, etc.

Allahverdi (2003) étudie le problème d'ordonnancement dans un flowshop avec le but de minimiser la somme de Makespan et Flowtime moyen (MFT). Le MFT est égal à la somme des temps d'achèvement des pièces, divisée par le nombre de pièces. Les méthodes heuristiques ont été proposées pour le cas de 2 machines et m machines.

La transformation d'un problème d'ordonnancement flow-shop en un problème de Voyageur de Commerce ou VDC est une technique souvent utilisée dans la littérature. Le problème de minimisation du Makespan de permutation a été étudié par Saadani *et al.* (2005). Les auteurs proposent une heuristique pour le cas du problème où les machines travaillent continuellement, sans interruption à partir du début de la première pièce jusqu'à la fin de traitement de la dernière. Kalczynski et Kamburowski (2007) traitent le même problème avec une condition additionnelle : le flow-shop travaille sans attente. Les auteurs considèrent deux cas : 2- et multi-machines flow-shop.

Caricato *et al.* (2007) ont développé une approche basée sur la formulation VDC pour le problème de minimisation de makespan pour un flow-shop *hybride*. Un flow-shop hybride est un flow-shop où il y a des machines parallèles. Dans cet article des pièces sont groupées dans des lots prédéfinis, le théorie de graphes est utilisée pour la modélisation.

Un état de l'art de problèmes d'ordonnancement flow-shop qui peuvent être modélisés comme un problème de VDC est présenté par Bagchi *et al.* (2006). Les auteurs montrent que les approches basées sur une formulation VDC sont assez performantes pour résoudre les problèmes de ce type.

Ordonnancement avec des événements aléatoires

Un temps d'exécution aléatoire signifie, que le temps total d'exécution réel peut être différent du temps d'exécution planifié. Au niveau de la planification, pour minimiser les conséquences des aléas, les entreprises utilisent généralement des stocks, et *Material Requirement Planning* (MRP) pour les gérer. Un état de l'art sur les problèmes de gestion de production sous aléas avec utilisation de la stratégie MRP a été présenté dans Dolgui et Prodhon (2007). Des exemples de problèmes de ce type avec les méthodes de résolution correspondantes peuvent être trouvés dans de nombreux articles - Molinder (1997), Dolgui et

Ould-Louly (2002), Gurnani et Gerchak (2007), Louly et Dolgui (2009), etc.

La différence principale entre les problèmes d'ordonnancement déterministes et probabilistes est la présence de facteurs incertains tels que les pannes machines, le temps de fabrication aléatoire. La prise en compte de plusieurs types d'incertitude à la fois est difficile. Dans la littérature nous trouvons le plus souvent des modèles pour chaque type d'aléas séparément. Deux types d'approches sont appliqués : *l'ordonnancement statique* ou *l'ordonnancement dynamique*. Dans le cas "statique" le planning de traitement des pièces est déterminé avant le début de la fabrication contrairement au cas "dynamique", où le planning est construit au fur et à mesure d'évènements qui arrivent.

Quand une machine est soumise aux pannes aléatoires, les périodes où elle est disponible pour la production alternent avec de périodes où elle est indisponible. L'intervalle de temps où la machine marche sans interruption (panne), est appelé *Uptime*. Notons que quand une panne arrive, l'uptime doit être remis à zéro (réinitialisé). Le temps de réparation est appelé *Downtime*. Alors nous pouvons décrire ce processus par la séquence U_k, D_k, où U_k est le k-ème uptime et D_k et le k-ème downtime de la machine. La durée de chaque U_k et D_k sont des variables aléatoires. Un processus stochastique $\{N(t) : t \geq 0\}$, qui représente le nombre d'*événements* qui ont eu lieu avant le temps t, est souvent associé avec les uptimes. Dans ce cas les événements sont des pannes. Souvent ce processus $N(t)$ est appelé le *processus de comptage*.

Donnons quelques exemples de la littérature. Un problème de minimisation de *Makespan* (la date de fin de traitement de la dernière pièce) sur une machine a été étudié dans Kasap *et al.* (2006). La machine est soumise aux pannes aléatoires, qui interrompent son fonctionnement normal. Dans le problème considéré, une tâche doit être retravaillée entièrement, si elle n'était pas terminée au moment d'arrivée d'une panne.

Les hypothèses du problème sont donc les suivantes :

- Toutes les pièces $j, \quad j \in J$ sont disponibles au temps zéro ;
- Le temps *Uptime* est la variable aléatoire avec une distribution continue dans l'intervalle $[0, c]$;
- t_j est le temps de traitement de pièce j sur la machine ;
- La variable aléatoire z_j représente la période de temps durant laquelle la pièce j occupe la machine.

Il est admis que $c > \sum_{j \in J} t_j$. L'espérance mathématique de makespan est donnée par

$$\mathbb{E}C_{\max} = \sum_{j \in J} \mathbb{E}z_j, \tag{1.11}$$

où $\mathbb{E}z_i$ est l'espérance mathématique de z_i.

Adiri *et al.* (1989) ont démontré que si la panne est unique durant la période de planification, si le moment de son arrivée est connu avant le début de la fabrication, et si les temps de fabrication des tâches sont également donnés, alors déterminer s'il existe ou non un *ordonnancement* avec *flowtime* inférieur ou égal à une valeur donnée est NP-complèt.

Allahverdi et Savsar (2001) considèrent un problème d'ordonnancement dans un flowshop avec deux machines. Le critère est de minimiser le makespan. Les temps de set-up, pas inclus dans le temps de fabrication, sont pris en compte. Les machines peuvent tomber en panne. Le processus de comptage $N(t)$ et le temps de réparation peuvent suivre n'importe quelle loi de probabilité.

Un problème d'ordonnancement stochastique pour un flow-shop de permutation avec m machines est traité par Gourgand *et al.* (2003). Les variables aléatoires sont les temps de fabrication $t_{i,j}$, où i est une tâche et j est une machine. Les $t_{i,j}$ suivent la loi de probabilité exponentielle $P[t_{ij} < t] = 1 - \exp^{-t\ \mu_{ij}},\ t \geqslant 0$ avec un taux μ_{ij} connu. L'objectif est de minimiser le Makespan. Afin de résoudre le problème d'évaluation des performances, les auteurs proposent un algorithme récursif, basé sur une chaîne de Markov. Alcaide *et al.* (2002) proposent une procédure permettant de transformer le problème de minimisation du Makespan pour un flow-shop avec pannes machines dans une séquence finie de problèmes sans panne machine. Les durées des périodes de disponibilité ainsi que les durées des périodes d'indisponibilité sont des variables aléatoires. Les auteurs proposent une approche pour construire des ordonnancements réalisables pour les intervalles entre deux pannes. A partir de ces ordonnancements réalisables, ils construisent un ordonnancement optimal.

1.4 Lotissement et ordonnancement

Des problèmes de lotissement et d'ordonnancement avec des temps et/ou coûts de set-up jouent un rôle important dans beaucoup d'industries. Il y a également un grand nombre d'articles où les problèmes de lotissement et d'ordonnancement sont traités ensemble. Potts et Van Wassenhove (1992) intègrent les décisions de lotissement dans un problème de planification avec batching. Le batching est appliqué pour éviter les set-up supplémentaires. Le

lotissement est utilisé ici pour découper les tâches quand elles sont composées de plusieurs pièces identiques. Le fractionnement d'une tâche conduit souvent à la diminution du temps de fabrication. Cet article propose un modèle général qui combine le batching, le lotissement et l'ordonnancement ainsi qu'une revue des travaux sur ce sujet. Drexl et Kimms (1997) proposent une revue des modèles de lotissement et de séquencement et expliquent la différence entre eux. Le focus est mis sur les modèles de temps continu et les modèles avec plusieurs niveaux.

Potts et Kovalyov (2000) proposent une revue de littérature et de méthodes de résolution sur les modèles qui intègrent les décisions de planification et de batching. L'article de Zhu et Wilhelm (2006) présente une revue de la littérature sur les problèmes d'ordonnancement et de lotissement avec des temps (coûts) de set-up dépendants de la séquence. L'article de l'état de l'art d'Allahverdi *et al.* (2008) est consacré à la même thématique. Les résultats sont classifiés selon la configuration de la ligne, l'application de batching ou non, la dépendance ou l'indépendance des temps (coûts) de set-up. Les méthodes communes des solutions sont *Branch and Bound*, *programmation linéaire*, *programmation dynamique*, *heuristiques* et *méta-heuristiques*.

Modèles déterministes

Le problème de *Lotissement et d'Ordonnancement Economique* (ELSP) a été introduit en 1958 par Jack Rogers, qui a étendu le modèle EOQ au cas multi-produits, où il faut déterminer à la fois les tailles de lots et leur séquence.

Plus précisément, le problème ELSP consiste à déterminer un plan de production pour n ($n \geq 2$) produits sur une machine, qui à la fois minimise le coût et satisfait la demande d_i pour chaque type de produits i, $i = 1, \ldots, n$. Des coûts a_i et des temps de set-up s_i sont engagés pour chaque changement de type de produit traité sur la machine. Le coût de stockage h_i (pour chaque unité par unité de temps) est basée sur la quantité de produits stockés. L'objectif est de déterminer la quantité des articles de chaque type à fabriquer pour chaque période, ainsi que l'ordre de leur fabrication.

Soit p_i le taux de production pour i, Q_i la quantité des pièces à fabriquer et t_i^{lot} le temps de production d'un lot de produit i, $i = 1, \ldots, n$. Le temps t_i^{lot} inclut le temps de fabrication lui-même et le temps de set-up :

$$t_i^{lot} = s_i + \frac{d_i}{p_i} T_i,$$

où T_i est la durée entre deux lancements consécutifs de produit i, $i = 1, \ldots, n$. Le coût par

unité de temps est égal à la somme des coûts moyens de set-up et de stockage :

$$C_i = \frac{a_i}{T_i} + \frac{h_i(p_i - d_i)r_i T_i}{2p_i} \tag{1.12}$$

La solution du problème est un ensemble $T = \{T_1, T_2, \ldots, T_n\}$. Chaque T_i doit être suffisamment longue pour pouvoir fabriquer toutes les pièces nécessaires de type i avant le début du prochain cycle. L'objectif est donc de trouver un ensemble T réalisable qui minimise le coût total :

$$\sum_{i=1}^{n} C_i \tag{1.13}$$

L'ordonnancement T est réalisable si :

- au plus une pièce est traitée par la machine à tout moment ;
- le temps total de chargement de la machine ne dépasse pas le temps disponible ;
- la demande est satisfaite pour chaque type de produits.

Elmaghraby (1978), Narro Lopez et Kingsman (1991) et Yao (2005) ont publié les états de l'art pour le problème ELSP et ses extensions. Le problème ELSP est NP-difficile (voir Hsu (1983)), alors la plupart des méthodes dévéloppées pour le résoudre sont des heuristiques.

Deux méthodes heuristiques ont été proposées par Geng et Vickson (1988). Khouja *et al.* (1998) montrent comment l'approche génétique peut être appliquée à l'ELSP. Moon *et al.* (2002) ont fourni un algorithme génétique hybride basé sur le TVLS (*time-varying lot sizes*). Ils essaient de trouver une longueur de cycle, une séquence de production, les durées de temps de production, et la durée des temps d'inactivité, de sorte que la séquence de production puisse être complétée dans le cycle choisi tout en minimisant le coût de set-up et de stockage. Chatfield (2007) a fait une brève revue des méthodes existantes pour ce problème et a proposé une procédure nommée "*Genetic Lot Scheduling*", qui applique un algorithme génétique et permet d'obtenir des solutions de bonne qualité. Pour d'autres exemples d'application des algorithmes génétiques aux problèmes de type ELSP et ses extensions, le lecteur est invité à consulter Sarker et Newton (2002).

Un algorithme de recherche en temps polynomial qui garantit un optimum global a été proposé dans Yao et Elmaghraby (2001) pour le problème ELSP avec la politique *puissance-de-deux* (un Algorithme génétique pour le même cas du problème peut être trouvé dans Sun *et al.* (2009). L'article Kovács *et al.* (2009) présente une approche de programmation mathématique et un modèle MIP. La présence des temps de set-up et de capacité limitée a

été prise en compte. Ici le but était de décider quels types de pièces seront fabriqués dans chaque période, la séquence et la quantité de produits de chaque type pour satisfaire une demande dynamique et minimiser le coût de set-up et de stockage.

Yao et Huang (2005) étudient un problème ELSP pour les produits périssables, c.-à-d. pour les produits qui ne peuvent pas être en stock très longtemps. Ce phénomène est assez courant, surtout dans l'industrie alimentaire. Les auteurs proposent un algorithme génétique hybride avec une procédure pour tester si l'optimum local est réalisable.

Un problème de lotissement pour un flow-shop sans attente est étudié dans (Emmons et Mathur (1995)). L'objectif est de minimiser le makespan pour n jobs de m types, où $n \gg m$. Des jobs du même type ont le même temps de fabrication sur les machines. Les auteurs ont reformulé leur problème en problème de Voyageur du Commerce, où chaque job correspond à une ville.

Un autre problème de lotissement multi-produit dans un flow-shop a été étudié par Ouenniche et Boctor (2001). Les demandes, les coûts, et les taux de production sont déterministes. L'approche heuristique proposée résout d'abord le problème de séquencement, ensuite les problèmes de lotissement et d'ordonnancement sont résolus simultanément.

Modèles stochastiques

Des évènements aléatoires peuvent arriver à n'importe quel moment du processus de fabrication : le temps d'exécution aléatoire, les rebuts (rendement aléatoire), la demande incertaine, les temps de livraison ou les délais d'approvisionnement aléatoires, etc. Dans cette thèse nous nous intéressons au séquencement et lotissement quand les quantités fabriquées et les temps nécessaires pour fabriquer les produits sont aléatoires. Nous avons constaté qu'il n'y pas beaucoup de publications sur le problème de lotissement et de séquencement sous incertitudes. Ci-dessous nous citons les travaux que nous avons trouvés.

Sox *et al.* (1999) présentent l'état de l'art en recherche sur le problème de lotissement et d'ordonnancement stochastique (SLSP) et ses extensions. Ce problème consiste à déterminer la planification de la production de plusieurs produits sur une machine quand la demande est aléatoire, la capacité de production est limitée et les coûts (temps) de set-up sont importants. Le problème de *Lotissement et d'Ordonnancement Économique Stochastique* (SELSP) est une autre extension du problème ELSP. Elle est caractérisée par une demande aléatoire et des temps de set-up et de fabrication également aléatoires. Winands *et al.* (2010) présentent tous les travaux sur la SELSP depuis la publication de Sox *et al.* (1999).

Dans le problème ELSP classique, il est supposé que la ligne de fabrication est parfaite, elle produit des pièces à un taux de production fixe et les produits finaux sont tous de bonne qualité. Tang et Teunter (2006) ont développé une autre version du problème ELSP pour considérer le cas de présence de flux de retour. Les hypothèses suivantes ont été utilisées : il n'y a pas de différence de qualité entre les pièces usinées et les pièces ré-usinées ; des retours sont collectés et stockés jusqu'au début du processus de ré-usinage ; la proportion (pourcentage) des retours β_i de pièces de type i, $i = 1, \ldots, n$ est donnée. Les auteurs ont proposé un algorithme exacte pour le cas où le temps de cycle est commun pour un seul lot de ré-usinage. L'inconvénient de cette approche est la complexité du modèle MIP. Teunter *et al.* (2009), pour le même problème, proposent quatre heuristiques assez efficaces. Haji *et al.* (2008) traitent également un problème ELSP avec rebuts et ré-usinage.

L'article Beraldi *et al.* (2006) est consacré à un problème de dimensionnement de lots sur les machines parallèles identiques. Il y a n types de produits, m machines, T périodes et les coûts de set-up dépendant de la séquence. Le temps de fabrication est incertain en raison de la variabilité des taux de production des machines. Il est supposé que chaque paramètre incertain est un processus stochastique en temps discret avec un espace probabiliste fini. L'objectif est de minimiser le coût total de set-up tout en respectant la contrainte que la demande pour chaque période doit être satisfaite.

Conclusion

Nous avons présenté dans ce chapitre un état de l'art des problèmes de lotissement et d'ordonnancement. Les problèmes ont été classifiés selon la quantité des périodes sur lesquelles la planification doit être effectuée, la quantité des machines sur la ligne de production, présence ou absence des événements aléatoires. Les problèmes que nous allons considérer dans les chapitres suivants se trouvent à l'intersection des domaines suivants : le lotissement optimal pour les systèmes de production imparfaits et le séquencement avec batching.

Chapitre 2

Problématique

L'objectif de ce chapitre est de présenter le problème de planification que nous allons étudier dans cette thèse. Ce problème consiste à définir les quantités optimales de produits à traiter et l'ordre de leurs passages dans une ligne de production afin d'optimiser un critère. En prenant un exemple et en utilisant un modèle de simulation, nous montrons la nécessité de développer des méthodes d'optimisation pour ce type de problème.

Dans ce qui suit nous décrivons un problème de planification de production *au jour le jour*. Le problème est issu de l'industrie électronique, où il fallait construire un planning de production pour les 24 heures suivantes, ce planning étant répété chaque jour. Le problème consistait à définir les quantités optimales de produits à traiter (tailles de lots) et l'ordre de passage des lots dans une ligne de production afin d'optimiser un critère. La ligne de production considérée est destinée à traiter des pièces de plusieurs types, qui seront ensuite envoyées dans l'atelier de montage des modules électroniques. L'un des critères était donc le taux de service de l'atelier de montage (pourcentage de commandes satisfaites). En effet, l'absence d'un composant pour le montage pouvait entrainer l'arrêt de l'atelier de montage. D'autres critères étaient le temps total de fabrication et le coût total de retard. Une courte description des ateliers de fabrication peut être trouvée dans Dolgui *et al.* (2005) et l'annexe A. Dans la suite de la thèse, nous nous limiterons à la ligne de fabrication des composants.

2.1 Hypothèses du problème

Nous traitons un problème de lotissement et de séquencement pour n types de produits, le problème est donc de type multi-produits. La ligne de production est constituée de m machines, chacune capable de traiter une seule pièce à la fois. Chaque produit doit passer par toutes les machines. La séquence d'opérations (machines) est la même pour tous les types de produits. Le temps de transfert d'une pièce d'une machine q à la machine suivante $q+1$ est négligeable. Considérant ces hypothèses, nous pouvons conclure que la ligne de production est un *flow-shop sans attente* et de *permutation*. Nous supposons qu'il n'est pas possible de soustraire une pièce de la production une fois qu'elle est engagée dans la ligne.

Rappelons qu'un *flow-shop* ou "fabrication en ligne" est un ensemble de machines sur lesquelles passent tous les produits dans le même ordre. Ces machines sont habituellement rangées en ligne, dans l'ordre dans lequel elles sont visitées (Figure 2.1). Le problème classique de ligne de type flow-shop est de trouver l'ordre de passage de n jobs tel que le Makespan ($C_{\max}$) soit minimisé.

FIGURE 2.1 – Flow-shop

Deux produits différents peuvent avoir des temps de fabrication différents sur la même machine. S'il existe des temps d'exécution nuls (la pièce n'étant pas traitée sur une machine particulière), on parle alors d'un flow-shop généralisé (Figure 2.2).

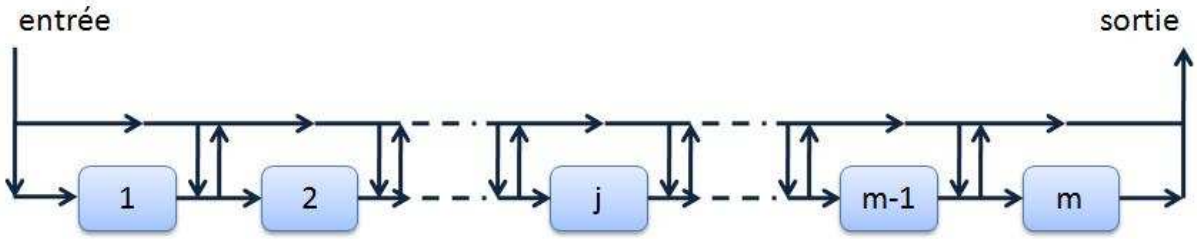


FIGURE 2.2 – Flow-shop généralisé

Le fait que des produits de tous les types soient traités par le même ensemble des machines et dans le même ordre, oblige à prendre en considération le *temps de set-up* entre la fabrication de deux produits de types différents. En effet, la plupart des problèmes de planification considèrent les temps (coûts) de set-up. Il est logique de supposer que la machine ne puisse

pas traiter de pièces (doit être inactive) pendant ce temps-ci. De plus, nous supposons que le set-up des machines ne peut être démarré qu'une fois la ligne de production est vide.

En règle générale, les machines dans la fabrication électronique ne sont pas parfaites, la possibilité de rebut est donc toujours présente. Les *aléas de rendement* sont un phénomène commun à beaucoup de systèmes de fabrication, alors il doit être pris en compte. Les conséquences de ce phénomène sont nombreuses : pour des clients, par exemple, cela signifie souvent ne pas recevoir la quantité commandée et(ou) ne pas la recevoir à l'heure. Les causes du rendement aléatoire varient en fonction de l'industrie, du type de processus et d'autres circonstances, mais il s'agit typiquement des imperfections liées à la ligne de production, aux matières, aux limitations de capacité des machines, et divers facteurs environnementaux comme la température et l'humidité, etc.

Lorsque la fabrication est lancée, les *pannes de certaines machines* peuvent interrompre le bon fonctionnement de la ligne. Parfois les pannes peuvent être négligées (petite durée ou une réparation immédiate), mais elles peuvent également créer des problèmes complexes. Dans ce cas il sera nécessaire d'arrêter la production pour un temps de réparation et parfois même de faire une nouvelle planification. Nous supposons que des pannes ne peuvent survenir sur les machines que lorsque ces dernières sont occupées (il est considéré que pendant le temps de set-up la machine "n'est pas occupée"). Nous supposons également que si une pièce est en cours de fabrication quand une panne arrive, le travail effectué sur cette pièce n'est pas perdu, c.-à-d. nous imposons un modèle du type *"preempt-resume"* plutôt que le modèle *"preempt-restart"* quand le travail effectué est perdu. Le planning doit être effectué pour une période de 24 heures, donc nous considérons que il n'y a pas de dégradation de machines.

Si une machine peut tomber en panne, le *temps d'exécution* devient aléatoire. Le délai d'exécution est une période de temps entre le début de processus de production et son achèvement. Un délai d'exécution incertain signifie que le temps de fabrication réel peut être différent de celui qui a été prévu.

2.2 Notations et fonctionnement de la ligne

Dans cette section, nous donnons quelques exemples du problème traité. Nous allons utiliser les notations suivantes :

- le nombre de types de produits à fabriquer n ;
- le nombre de machines sur la ligne de production m ;

- l'horizon de planification T_0 (inutile, si l'objectif est la minimisation du Makespan) ;
- la demande d_i en pièces pour chaque type de produits $i,\ i = 1, \ldots, n$;
- le temps de traitement t_{iq} d'une pièce de type i sur la machine $q,\ q = 1, \ldots, m$;
- le temps de préparation $s_{0,i},\ s_{0,i} \geq 0,\ i = 1, \ldots, n$, si la fabrication commence par une pièce de type i ;
- le temps de set-up $s_{i,j}$ entre un lot de produit i et un lot de produit j (rappelons qu'un lot peut être constitué d'une seule pièce) ;
- la quantité totale x_i de pièce de type $i,\ i = 1, \ldots, n$ à lancer en fabrication - **variables de décision** ;
- l'ordre de passage π des produits dans la ligne - **variable de décision** ;
- la quantité de pièces de bonne qualité x_i^b (ou rendement), que nous obtenons, si la quantité totale de pièces de type i lancée en production est x_i ;

La solution du problème est un planning, qui détermine la quantité de pièces à fabriquer pour chaque lot et l'ordre de passage des lots dans la ligne de production (voir Tableau (2.1)).

TABLE 2.1 – Une solution pour le problème de planification

Numéro de lot	1	2	3	4	5	6	7	8
Type de produit	1	5	3	8	2	7	4	6
Taille de lot	5	12	4	9	1	41	23	4

Pour cette solution, le premier lot à fabriquer est le lot de produit 1 en quantité de 5 pièces, ensuite le lot de produit 5 de taille 12 et ainsi de suite.

Pour donner une illustration numérique, un exemple avec 8 types de produit et 4 machines a été généré. L'horizon de planification T_0 est égal à un jour (24 heures). Le tableau 2.2 représente le niveau de la demande d_i pour chaque type de produit $i,\ i = 1, \ldots, 8$ et le temps unitaire de fabrication. Sans perte de généralité, nous supposons que le temps de traitement unitaire t_i, fourni par le tableau 2.2, ne dépend que du type de produit et est le même pour toutes les machines sur la ligne, c.-à-d. $t_{iq} = t_i$, $q = 1, \ldots, 4$.

Le tableau 2.3 contient les temps de set-up. La valeur qui se trouve à l'intersection de la ligne i, $i = 0, \ldots, 8$ et de la colonne j, $j = 1, \ldots, 8$ est le temps de set-up nécessaire pour reconfigurer les machines pour la production des pièces de type j après la fabrication d'un lot de pièces de type i. Ce temps de set-up s_{ij} est le maximum des temps de set-up des

TABLE 2.2 – Les données d'une instance de problème avec 8 lots

Numéro de Lot	*Demande* d_i(pièce)	*Temps unitaire* t_i(heure)
1	25	0,065
2	35	0,035
3	60	0,04
4	75	0,015
5	40	0,045
6	65	0,025
7	85	0,02
8	30	0,075

TABLE 2.3 – Le temps de set-up (en heures)

$i \setminus j$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
0	0,21	0,29	0,33	0,23	0,37	0,32	0,29	0,36
1	0	0,27	0,25	0,35	0,38	0,25	0,28	0,31
2	0,31	0	0,24	0,21	0,28	0,37	0,33	0,28
3	0,35	0,29	0	0,36	0,34	0,37	0,38	0,29
4	0,23	0,39	0,33	0	0,32	0,37	0,31	0,24
5	0,36	0,24	0,27	0,23	0	0,25	0,27	0,28
6	0,27	0,37	0,21	0,32	0,38	0	0,31	0,29
7	0,33	0,23	0,25	0,36	0,38	0,25	0	0,33
8	0,32	0,3	0,37	0,34	0,28	0,33	0,21	0

machines de la ligne. Enfin, si $i = j$ le temps de set-up est égal à zéro.

Considérons un lot de type 2 et supposons qu'il est le premier à être fabriqué. Le temps de set-up $s_{0,2}$, qui précède la fabrication égal à $0,29$. Si nous ne prenons en compte ni rebuts, ni pannes, la première pièce finie va quitter la 4-ème machine dans $s_{0,2} + t_2 \times 4 = 0,43$ heures, la deuxième dans $0,43 + 0,035 = 0,465$, et ainsi de suite.

Pour avoir la possibilité de faire une illustration d'un gain qui peut avoir lieu si l'on cherche une séquence et des tailles de lots optimales, nous allons utiliser un modèle de simulation créé sous ARENA. Un gros avantage de la simulation est lié à la possibilité d'utiliser n'importe quelle loi de distribution pour la modélisation des rebuts et des pannes, ce qui est souvent impossible à faire de manière analytique. Néanmoins, dans cette section

nous supposons que les lois de distribution des pannes et des réparations sont exponentielles négatives. Cela signifie que les pannes se produisent d'une manière continue et indépendante suivant un taux moyen constant. Les rebuts suivent la loi de Bernoulli. Cette hypothèse a été prise pour avoir des résultats compatibles avec nos modèles analytiques.

Nous avons besoin des données suivantes :

- p_{iq} la probabilité qu'une pièce de type i soit de bonne qualité après son traitement sur la machine q, où $i = 1, \ldots, n$ et $q = 1, \ldots, m$;
- $1/u_q$ le taux moyen de panne pour la machine q, $q = 1, \ldots, m$;
- $1/r_q$ le taux moyen de réparation pour la machine q, $q = 1, \ldots, m$;

TABLE 2.4 – Probabilités p_{iq}, $i = 1, \ldots, n$ et $q = 1, \ldots, m$

Lot	Machine			
	1	**2**	**3**	**4**
1	0,94	0,96	0,94	0,97
2	0,99	0,92	0,94	0,95
3	0,98	0,99	0,97	0,99
4	0,95	0,96	0,98	0,97
5	0,92	0,99	0,97	0,95
6	0,98	0,95	0,91	0,98
7	0,93	0,98	0,99	0,89
8	0,88	0,98	0,97	0,97

TABLE 2.5 – Taux de pannes et de réparations des machines

Machine	*MTTF, heures*	*MTTR, heures*
1	410	0,9
2	120	0,6
3	350	0,4
4	230	0,7

Les valeurs de ces paramètres sont dans les tableaux 2.4 et 2.5. Le premier tableau contient les probabilités p_{iq}. Le deuxième tableau donne les *Temps moyen de bon fonctionnement* (ou *Mean Time To Failure* en anglais) et les *Temps moyen de réparation* (ou *Mean Time To*

Repair) pour chaque machine q, $q = 1, \ldots, 4$. Nous les avons générés dans les intervalles suivants : $[100; 500]$ heures pour MTTF et $[0,3; 1]$ heures pour MTTR.

La moyenne des temps de bon fonctionnement (MTTF) est un paramètre de fiabilité d'une machine. Nous tenons à préciser ici que cette valeur n'intègre pas le temps de réparation. Le MTTF d'un outil est une variable aléatoire et ne peut être estimée que pour une situation concrète en prenant en compte l'utilisation de cet outil. L'unité de mesure du MTTF est souvent l'heure. Un autre paramètre, utilisé et lié au MTTF, est le taux de pannes λ (*Failure rate* en anglais ou FIT), $\lambda = \frac{1}{MTTF}$.

Le temps moyen de réparation (MTTR) est une mesure de base de la maintenance pour des outils (machines, etc.) que l'on peut réparer, et représente le temps moyen nécessaire pour réparer un outil. Mathématiquement, il s'agit du temps total de maintenance (réparations) pendant une certaine période, divisée par le nombre de pannes arrivées pendant cette période.

2.3 Analyse du problème

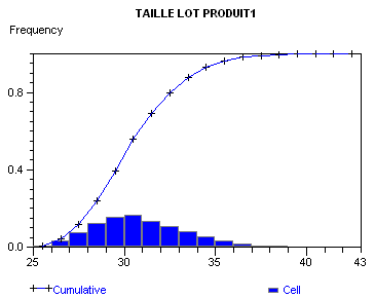
Pour faire nos démonstrations, nous allons considérer deux types de fonction objectif : 1) la minimisation du Makespan C_{max}, 2) la maximisation de la probabilité de satisfaire la demande globale (i.e. $x_i^b \geq d_i, \quad i =, \ldots, n$) sous la condition que la borne supérieure de l'horizon de planification T_0 soit donnée. Cette seconde option est connue sous le nom de *maximisation du niveau de service*. Pour obtenir la probabilité de satisfaire la demande pour une solution donnée (séquence des lots et leurs tailles), nous devons effectuer un grand nombre de lancements du modèle de simulation. Le résultat d'un lancement est 1 si nous avons obtenu le nombre suffisant de pièces de tous les types et 0 autrement. La fréquence du succès nous donne une estimation de la probabilité recherchée.

2.3.1 Dimensionnement de lots

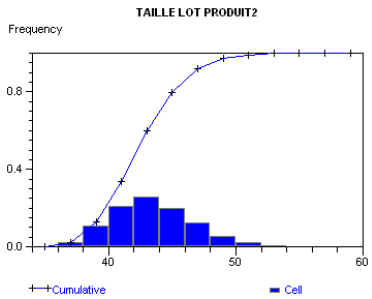
Nous considérons une ligne à entrée automatique, c.-à-d. où les pièces à traiter doivent être placées à l'avance dans le magasin de charge. Alors nous devons définir une séquence de lots et leurs tailles. Ce plan de fabrication sera réalisé en régime automatique.

Pour commencer nous ne prendrons en compte que le processus de fabrication de pièces, c.-à-d. nous supposons que la séquence de produits a été déjà trouvée (nous considérons que la séquence de lots π est $1, 2, \ldots, 8$).

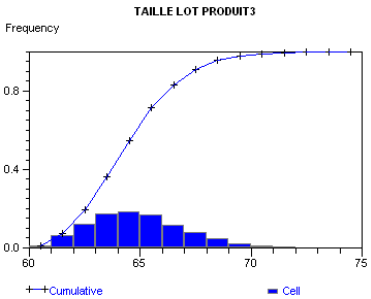
Le problème est donc le suivant : **déterminer la quantité** x_i de pièces de chaque type



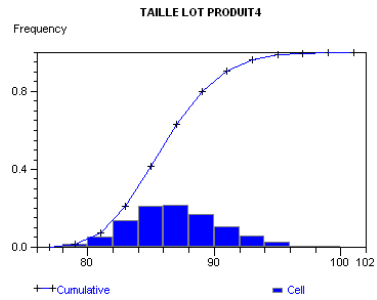
(a) Lot de type 1



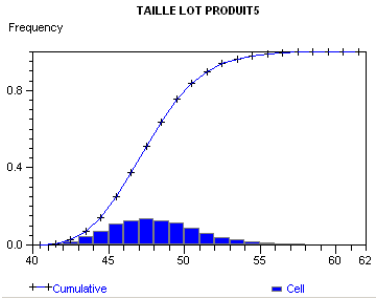
(b) Lot de type 2



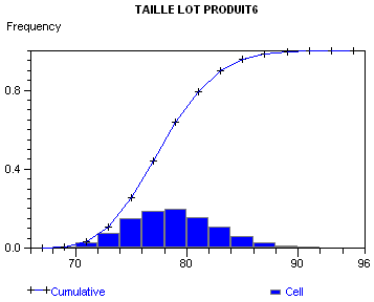
(c) Lot de type 3



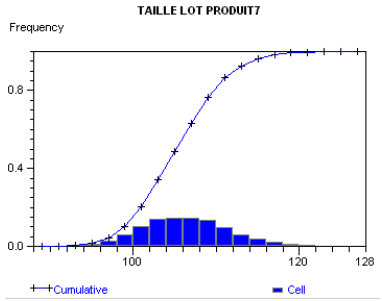
(d) Lot de type 4



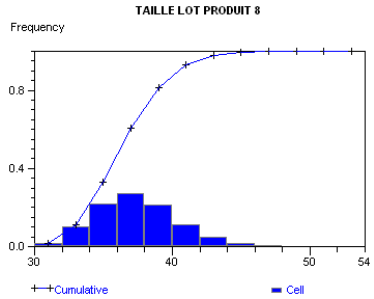
(e) Lot de type 5



(f) Lot de type 6



(g) Lot de type 7



(h) Lot de type 8

FIGURE 2.3 – Histogramme de répartition des tailles de lots

$i,\ i = 1, \ldots, n$ à lancer en fabrication pour optimiser un critère donné. Pour le critère C_{max} le problème consiste à déterminer la quantité x_i de telle manière, que x_i^b soit égal à d_i ; pour le critère de taux de service - les quantités $x_i,\ i = 1, \ldots, n$ maximisant la probabilité de satisfaire toutes les demandes tout en respectant la valeur T_0.

Dans la figure 2.3 nous présentons la répartition d'une valeur qui représente la taille de lot, c.-à-d. la quantité des pièces qu'il faut lancer pour obtenir la demande d_i pour chaque type de produit $i,\ i = 1, \ldots, n$. La courbe présentée sur chaque graphe est la fonction de répartition correspondante. Nous avons effectué 20000 lancements du modèle de simulation. Pour le lot de type 1 (voir la figure 2.3 (a)) la quantité des pièces nécessaire pour avoir $d_i = 25$ pièces de bonne qualité varie entre 25 et 43. Dans approximativement 17% des cas, la taille

de lot est égale à 30. Mais pour avoir une forte probabilité de satisfaire cette demande il faut augmenter cette quantité jusqu'à 39-40 (voir la courbe des pourcentages cumulés). On peut dire la même chose pour tous les autres lots.

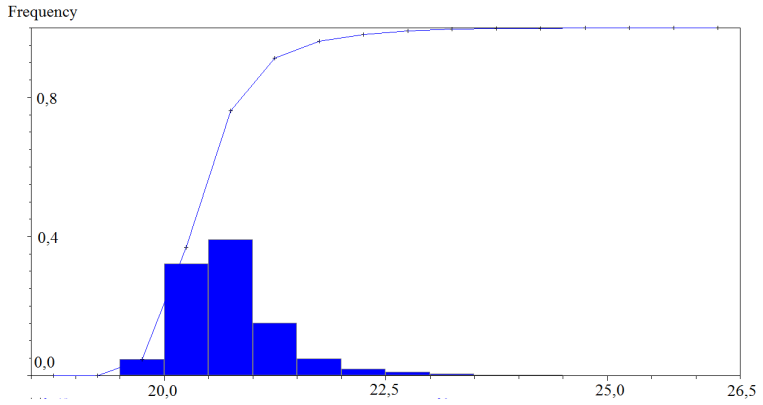


FIGURE 2.4 – Histogramme représentant la répartition de la valeur de Makespan

TABLE 2.6 – Les tailles de lots le plus fréquentes pour l'exemple

Numéro de lot	1	2	3	4	5	6	7	8
Type de produit	1	2	3	4	5	6	7	8
Quantité à fabriquer	31	43	64	86	48	78	106	37

Figure 2.4 présente un histogramme de la répartition des valeurs du Makespan pour l'exemple que nous considérons. Nous pouvons constater que pour plus de 40% des cas, la valeur de makespan est entre $20,5$ et 21 heures. Si on prend les tailles x_i les plus fréquentes pour chacun des 8 lots (voir la figure 2.3), la solution obtenue est la suivante : $x = (31, 43, 64, 86, 48, 78, 106, 37)$ (voir le tableau 2.6). En considérant le temps total de set-up (qui est égal à $2,29$ heures), le makespan moyen pour cette solution est égal à $20,741$ heures.

Nous avons lancé le modèle de simulation pour la solution présentée dans le tableau 2.6 afin de montrer qu'en changeant le critère d'optimisation nous obtenons des solutions très différentes. La valeur du niveau de service que nous obtenons pour la solution $x = (31, 43, 64, 86, 48, 78, 106, 37)$ est égal à $0,0139$. En revanche, en prenant une autre solution avec les tailles de lots plus élevées, par exemple $x = (39, 54, 72, 100, 58, 92, 123, 49)$, et en laissant la même séquence de lots, nous obtenons le niveau de service $0,9649$. En même temps la valeur de Makespan pour cette nouvelle solution est égale à $23,1$. Figure 2.5 montre la convergence de la probabilité pour la solution présentée dans le tableau 2.7 avec la séquence

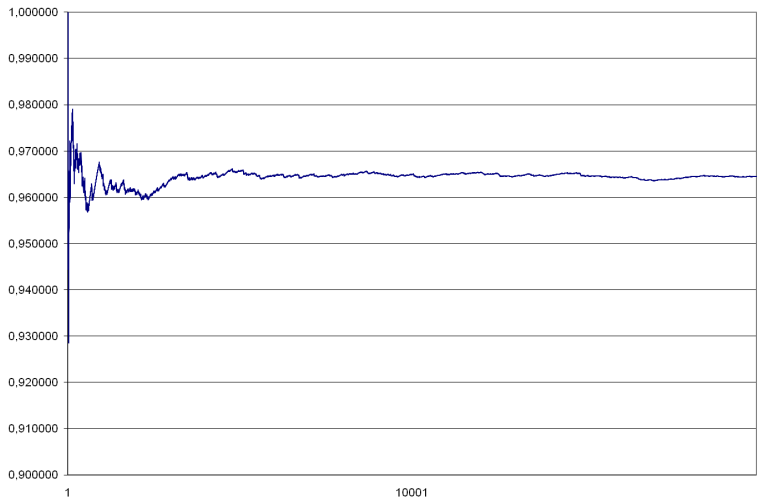


FIGURE 2.5 – Convergence de la probabilité pour une solution

de lots $\{8, 4, 6, 1, 7, 5, 2, 3\}$.

TABLE 2.7 – Une solution

Numéro de produit	1	2	3	4	5	6	7	8
Quantité à fabriquer	39	54	72	100	58	92	123	49

Montrons maintenant une autre propriété du problème. Fixons les tailles de lot (pour l'exemple que nous traitons dans ce chapitre, prenons celles présentées dans le tableau 2.7). Passons à l'étude de l'influence des temps de set-up.

2.3.2 Séquencement

Pour n'importe lequel des deux critères considérés, le temps de set-up total doit être minimisé, puisque cela diminue automatiquement le C_{max} et augmente le temps destiné à la fabrication, c.-à-d. que nous pouvons augmenter les volumes des lots (pour le deuxième critère).

Le problème est donc le suivant : **déterminer la séquence** de lots π minimisant le temps total de set-up, où $\pi = (\pi_1, \ldots, \pi_n)$, $\pi_i \in \{1, \ldots, n\}$ et $\pi_i \neq \pi_j$ quand $i \neq j$.

Soit $V = \{0, 1, \ldots, n\}$ l'ensemble des sommets d'un graphe et $E = \{e_{ij} | i, j \in \{1, \ldots, n\}\}$ l'ensemble des arcs. La distance (ou le poids) associée à l'arc $e_{i,j}$ est égale au temps de set-up entre un lot de type i et un lot de type j : $e_{i,j} = s_{ij}$ (voir tableau 2.3). Alors $G = (V, E)$ est un graphe orienté, asymétrique, puisque les distances $e_{i,j}$ et $e_{j,i}$ peuvent être différentes. Le

tableau 2.3 est une matrice de distances. Ajoutons n arcs : $e_{i0} = min\{s_{ij}\}$, $i, j \in \{0, 1, \ldots, n\}$ pour avoir un graphe complet (nous ajoutons la colonne 0 au tableau 2.3 qui représente des distances).

Il y a deux possibilités : nous pouvons imposer que tous les temps de set-up satisfassent *l'inégalité triangulaire* (notre cas) où pas. Rappelons, que l'inégalité triangulaire a la forme suivante :

$$s_{ij} + s_{jk} \geq s_{ik}, \tag{2.1}$$

où $i,\ j,\ k \in 0, 1, \ldots, n$ et $i \neq j \neq k$, c.-à-d. la distance directe s_{ik} n'est jamais plus grande qu'un détour via un autre sommet (j dans cet exemple). Si l'inégalité triangulaire est respectée, le problème de minimisation de temps total de set-up est équivalent au problème de recherche d'un circuit *Hamiltonien* (celui qui contient tous les sommets du graphe) d'un **poids minimal** dans un graphe complet G, ou au problème du *Voyageur de Commerce* (VDC) métrique (imposé par (2.1)), qui est NP-difficile. L'analyse du problème VDC peut être retrouvée dans Gutin et Punnen (2002). Pour notre cas, le problème est asymétrique. Comme nous sommes à la recherche d'un cycle, nous allons obtenir une solution du type $(\pi_1, \pi_2, \pi_3, \ldots, \pi_n, \pi_{n+1}, \pi_1)$, où tous les $\pi_i,\ \ i = 1, \ldots, n+1$ sont différents et chaque $\pi_i \in \{0, \ldots, n\}$. Après avoir obtenu le cycle, nous pouvons supprimer l'arc $e_{\pi_k, \pi_{k+1}}$, où $\pi_{k+1} = 0$; de manière à ce que le cycle commence par le sommet 0. Puisque le cycle passe une seule fois par chaque sommet, toutes les pièces du même type doivent être fabriquées dans le même lot.

En utilisant l'exemple ci-dessus, nous allons montrer un gain possible quand on choisit une bonne séquence. Par exemple, pour la séquence de lots suivante $(1, 2, 5, 4, 8, 7, 6, 3)$ le temps total de set-up est égal à $1,9$ heures. Tandis que pour la séquence $(8, 4, 6, 1, 7, 5, 2, 3)$ le temps de set-up est égal à $2,97$ heures. Donc si la séquence de lots n'est pas optimisée, il est possible de perdre jusqu'à $1,07$ heures, correspondant à $4,46\%$ d'un horizon de planification de 24 heures.

Si l'inégalité triangulaire n'est pas vérifiée (nous avons affaire à une instance du problème VDC non métrique), nous pouvons avoir des solutions optimales du problème avec plus d'un lot pour chaque type de produits. Dans sa définition, le problème classique du Voyageur de Commerce (métrique ou non) ne permet pas de visiter une ville plus d'une fois, mais de nombreuses applications n'ont pas cette contrainte. Car, par exemple, une instance du problème de VDC symétrique non-métrique peut être réduite à une instance métrique. Pour cela il faut remplacer le graphe d'origine par un graphe complet dans lequel la distance s_{ij} entre deux villes i et j est remplacée par le plus cours chemin entre i et j dans le graphe

initial.

Conclusion

Dans ce chapitre nous avons présenté le problème traité dans cette thèse. Nous avons montré les différentes variantes de ce problème obtenues en changeant les critères. Les études que nous avons effectuées en utilisant la simulation montrent qu'un gain important peut être obtenu si l'on cherche une séquence ainsi qu'une taille de lot optimale. C'est l'objectif de cette thèse.

Dans les chapitres suivants, pour différents critères et différentes formulations de ce problème, nous montrerons qu'une décomposition est possible quand les problèmes de séquencement et de dimensionnement de lots sont considérés comme des problèmes séparés et indépendants. Les chapitres 3 et 4 contiennent une formulation déterministe (quand les valeurs aléatoires sont remplacées par leurs moyennes ou quantiles) du problème sur une seule machine. Le chapitre 5 inclura une formulation probabiliste utilisant des lois de probabilité pour les variables aléatoires.

Chapitre 3

Modèle Déterministe : approche FPTAS

Dans ce chapitre nous étudions la problématique de séquencement et de dimensionnement de lots sur une machine. La machine est imparfaite sous deux aspects : elle peut d'une part produire des pièces défectueuses et d'autre part tomber en panne. Deux fonctions objectifs du problème sont considérées : le 1er objectif est de minimiser le temps total de fabrication, à condition que toutes les demandes soient satisfaites (problème P-Temps*) ; le second est de minimiser le coût total de non-satisfaction de la demande, sous la condition que l'horizon de planification soit limité (problème* P-Coût*). La présence de temps de changement de série complique le problème en ajoutant des contraintes supplémentaires. Nous allons utiliser l'approche qui consiste à remplacer les valeurs aléatoires par leurs caractéristiques déterministes (moyens, centiles, etc.) et travailler sur des modèles déterministes. Pour ces modèles nous allons démontrer que les décisions optimales de lotissement et de séquencement peuvent être prises séparément pour les deux problèmes. Il est également prouvé que les deux problèmes sont NP-difficiles, si la quantité de pièces défectueuses est aléatoire. Ensuite, nous considérons un cas particulier où le pourcentage des articles défectueux est donné pour chaque type de produits, et où le temps de réparation dépend directement du temps total de fabrication. Dans ce cas là, uniquement le problème* P-Coût *reste NP-difficile. Nous proposons une méthode de résolution* FPTAS *pour ce cas particulier du problème* P-Coût*. En utilisant des tests numériques, nous étudions la performance de l'approche proposée.*

3.1 Description du système de production étudié

Le système de production que nous considérons dans ce chapitre contient une seule ressource (ou machine). Cette machine est capable de traiter des pièces de différents types, mais pas plus qu'une pièce à la fois. Le *temps de changement de série* (dénommé ci-après le *temps de set-up*), dépendant de la séquence, sépare la fabrication de deux pièces de types différents. La machine est imparfaite :

- elle peut fabriquer des pièces de qualité insatisfaisante ;
- elle peut tomber en panne.

Le nombre de pièces défectueuses pour chaque produit est donné par une fonction entière croissante de la quantité des pièces lancées. Le temps total de réparation est donné par une fonction réelle croissante de quantités fabriquées des produits. Nous supposons qu'aucun composant ne peut être produit pendant le temps de changement de série ou le temps de réparation, et que le temps de set-up et le temps de réparation ne peuvent pas se chevaucher.

Soit n le nombre de types de produit à fabriquer. Pour chaque type de produit i, $i = 1, \ldots, n$ nous disposons des données suivantes :

- d_i - la demande pour les pièces de bonne qualité, $d_i > 0$;
- c_i - le coût unitaire de non-réalisation d'une pièce, $c_i > 0$;
- t_i - le temps de fabrication d'une pièce, $t_i > 0$;
- $s_{i,j}$ ($s_{j,i}$) - le temps de changement de série entre un lot de produit i et j (j et i), $s_{i,j} \geq 0, \quad s_{j,i} \geq 0$;
- $s_{0,i}$ - le temps de préparation ($s_{0,i} \geq 0$) lorsque la fabrication commence par une pièce de type i ;
- $f_i(x)$ - représente le nombre de pièces défectueuses pour un lot de produit i, si ce lot contient x composants, où $f_i(0) = 0$, et $f_i(x) \leq x$ pour chaque $x = 1, 2, \ldots$ (f_i, $i = 1, \ldots, n$ sont des fonctions entières croissantes) ;
- $T(x_1, \ldots, x_n)$ - la fonction représentant le temps total de réparation durant la période de production, où x_i correspond au nombre de pièces de type i. $T(\cdot)$ est une fonction réelle croissante et non négative.

Nous supposons que tous les paramètres du problème sont des variables entières et le temps de set-up satisfait l'*inégalité triangulaire*, c'est-à-dire :

$$s_{i,j} + s_{j,k} \geq s_{i,k} \tag{3.1}$$

pour $i = 0, 1, \ldots, n$, $j = 1, \ldots, n$, et $k = 1, \ldots, n$.

Les fonctions $f_i(\cdot)$, $i = 1, \ldots, n$ et $T(\cdot)$ peuvent être obtenues par une analyse statistique des données réelles ou à l'aide de la simulation. Le choix des caractéristiques statistiques (moyenne, centiles, etc.) dépend des spécificités de chaque problème ou application et doit être effectué par les utilisateurs. Dans la pratique, des valeurs moyennes sont souvent utilisées. Dans ce cas, chaque fonction f_i représente le pourcentage moyen de pièces défectueuses ; la fonction T quant à elle représente le temps total de réparation pendant le temps total de fabrication. Dans la section 3.3, nous expliquons les raisons pour lesquelles nous avons choisi de considérer ce cas particulier des fonctions $f_i(\cdot)$ et $T(\cdot)$. Ci-après nous allons appeler ce cas "*fraction défectueuse*" conformément à la littérature en *contrôle de qualité*.

Dans ce chapitre, nous allons étudier deux problèmes : P-Temps, où l'objectif est de minimiser le temps total de fabrication $C_{\max}$ (autrement dit minimiser le *Makespan*) et P-Coût, où l'objectif est de minimiser le coût total de non-réalisation de la demande, représenté par l'équation suivante :

$$\sum_{i=1}^{n} c_i \max\{0, d_i - (x_i - f_i(x_i))\} \tag{3.2}$$

sous la condition que la période de planification est limitée à T_0 (la borne supérieure fixe). Pour les deux problèmes, les variables de décision sont les tailles de lots $x = (x_1, x_2 \ldots, x_n)$ et leur séquence $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$.

3.2 Des propriétés communes

Les deux propriétés suivantes sont vérifiées pour les problèmes P-Temps et P-Coût.

Propriété 3.1 *Il existe une solution optimale pour chacun des deux problèmes* P-Temps *et* P-Coût*, dans laquelle toutes les pièces de même type sont fabriquées en un seul lot.*

Démonstration. Supposons qu'il existe une solution optimale s^0 contenant deux lots différents pour les pièces d'un même type. Sans perte de généralité supposons qu'il s'agit des lots L_j^1 et L_j^2 du produit j (voir figure 3.1). Nous pouvons rassembler les deux lots en ajoutant tous les composants du lot L_j^2 (qui doit être traité plus tard) à la fin du lot L_j^1. Notons que le temps total de fabrication de toutes les pièces de type j ne changera pas. Soit s^1 la solution obtenue après le déplacement du lot L_j^2.

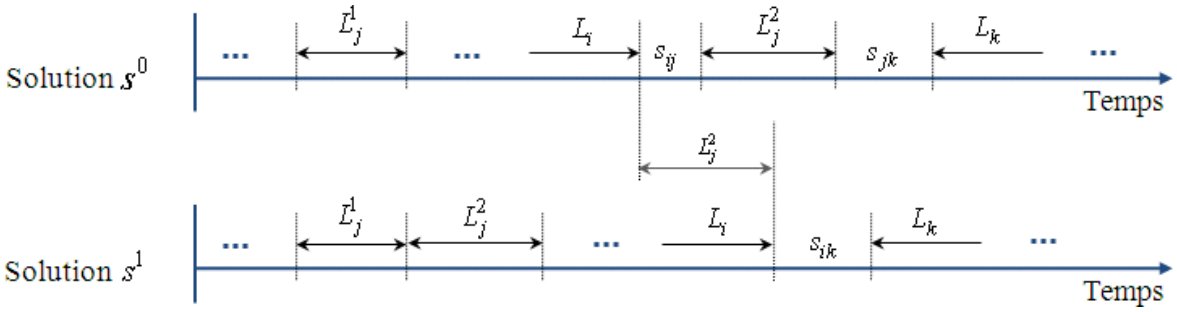


FIGURE 3.1 – La solution optimale

Supposons que dans la solution initiale s^0 le deuxième lot de type j est précédé par un lot L_i de type i, et succédé par un lot L_k de type k. Alors dans la nouvelle solution s^1, au lieu d'avoir des temps de set-up $s_{ij} + s_{jk}$, nous avons un temps de set-up s_{ik}. Tous les autres temps de set-up ne changeront pas. En tenant compte de la règle de triangularité, $s_{ij} + s_{jk} \geq s_{ik}$, nous pouvons conclure que ni la valeur du "Makespan" $C_{\max}$, ni le coût total d'insatisfaction de la demande ne vont augmenter. En conséquence la solution s^1 construite ci-dessus est une solution optimale pour n'importe lequel des deux problèmes P-TEMPS et P-COÛT. Le même raisonnement peut être appliqué à tous les autres lots (produits). □

Notons cependant que si l'inégalité triangulaire n'est pas vérifiée, il est possible d'avoir des solutions optimales qui contiennent plus qu'un lot pour chaque type de produit.

Propriété 3.2 *Il existe une solution optimale pour chacun des deux problèmes* P-TEMPS *et* P-COÛT*, pour laquelle les lots sont ordonnés suivant la séquence minimisant le temps totale de set-up.*

La preuve de cette propriété est évidente car la diminution du temps total de set-up ne peut augmenter ni le temps total de fabrication, ni le coût de la non-réalisation.

Dans le reste de ce chapitre, nous ne considérons que les situations pour lesquelles les Propriétés 3.1 et 3.2 sont vérifiées.

3.3 Problème P-TEMPS

Rappelons que l'objectif du problème P-TEMPS est de minimiser la date de fin de fabrication de la dernière pièce, $C_{\max}$, sous la condition que les demandes soient satisfaites pour tous les produits. Nous formulons le problème P-TEMPS de la manière suivante :

Problème P-TEMPS :

$$\begin{aligned} \textit{Minimiser} \quad & C(\pi, x) = s_{0,\pi_1} + \sum_{j=2}^{n} s_{\pi_{j-1},\pi_j} + \sum_{i=1}^{n} t_i x_i + T(x) \\ \textit{sous la contrainte} \quad & x_i - f_i(x_i) \geq d_i, \quad x_i \in \mathbb{N}, \quad i = 1, \ldots, n \end{aligned} \tag{3.3}$$

où s_{0,π_1} est le temps de préparation de la machine pour commencer la fabrication du premier lot de type π_1, $\pi_i \in \{1, 2, \ldots, n\}$ $\forall i = 1, 2, \ldots, n$, x_i représente la quantité de pièces dans le lot i à fabriquer (taille du lot), et x le vecteur contenant les tailles de tous les lots. Rappelons enfin que $\mathbb{N}$ est l'ensemble des entiers positifs.

Soit (π^*, x^*) la solution optimale pour le problème (3.3). Étant donné que la part de la fonction objective $C(\pi, x)$ liée à la séquence des lots π choisie est $S(\pi) := s_{0,\pi_1} + \sum_{j=2}^{n} s_{\pi_{j-1},\pi_j}$, et que cette somme n'a aucune influence sur la faisabilité du vecteur x, nous pouvons renforcer les propriétés 3.1 et 3.2 de la manière suivante :

Propriété 3.3 *Si les temps de set-up et de préparation sont strictement positifs, alors pour* ***chaque*** *solution optimale du problème* P-TEMPS*, toutes les pièces du même type sont fabriquées en un seul lot.*

Propriété 3.4 *Pour* **chaque** *solution optimale du problème* P-TEMPS*, le temps total de set-up $S(\pi)$ est minimisé.*

Ici, de manière analogue au chapitre 2, nous introduisons les temps de set-up artificiels $s_{i,0} = 0, \quad i = 1, \ldots, n$. La propriété 3.4 implique que si nous recherchons une solution optimale pour le problème P-TEMPS, nous devons *nécessairement* minimiser la valeur $S(\pi) = s_{0,\pi_1} + \sum_{j=2}^{n} s_{\pi_{j-1},\pi_j} + s_{\pi_n,0}$. Dans la communauté de la recherche opérationnelle le problème de minimisation de la fonction $S(\pi)$ est bien connu comme le problème du Voyageur de Commerce Asymétrique (voir chapitre 2). Nous avons $n+1$ villes $\{0, 1, \ldots, n\}$, chacune est représentée par le sommet d'un graphe complet, et les valeurs $s_{i,j}$ sont assignées aux arcs de ce graphe représentant les distances de la ville i à la ville j, $\forall i, j = \{1, 2, \ldots, n\}$.

Supposons que la séquence optimale $\pi^* := (\pi_1, \pi_2, \ldots, \pi_n)$ a été retrouvée, le temps total de set-up $S(\pi^*)$ a été calculé et tous les lots ont été renumérotés conformément à la séquence trouvée. Alors nous pouvons supprimer le terme $s_{0,\pi_1} + \sum_{j=2}^{n} s_{\pi_{j-1},\pi_j}$ de l'équation (3.3) et le problème P-TEMPS peut être réduit au problème suivant, nommé P-TEMPS-TAILLES (PTT).

Problème P-Temps-Tailles :

$$\begin{aligned} \textit{Minimiser} \quad & C_1(x) = \sum_{i=1}^{n} t_i x_i + T(x), \\ s.\ c. \quad & x_i - f_i(x_i) \geq d_i, \quad x_i \in \mathbb{N}, \quad i = 1, \ldots, n. \end{aligned} \tag{3.4}$$

Le fait que la fonction $T(x)$ soit croissante nous permet de conclure que le problème PTT est équivalent aux n problèmes suivants, nommés P_i, $i = 1, \ldots, n$, résolus simultanément.

Problème P_i : Minimiser x_i sous la contrainte $x_i - f_i(x_i) \geq d_i$, où $x_i \in \mathbb{N}$, $i = 1, \ldots, n$.

Nous allons maintenant prouver, que pour n'importe lequel des problèmes P_i, $i = 1, \ldots, n$, décider s'il y a une solution réalisable, est un problème NP-difficile, si la fonction $f_i(x_i)$ correspondante est représentée par une *boite noire*. Une boite noire pouvant être considérée comme un algorithme ou une formule qui calcule la valeur de $f(x)$ pour la valeur de x quelconque. Il n'est possible de voir que les entrées x et les sorties $f(x)$ de cette boite, le fonctionnement interne étant complètement opaque.

Théorème 3.1 *Soit $f(\cdot)$ une fonction entière croissante non négative représentée par une boite noire. La fonction $f(x)$ vérifie l'inégalité $f(x) \leq x$ pour chaque $x = 1, 2, \ldots$ Le problème P qui consiste à décider si $x - f(x) \geq d$, $x \in \mathbb{N}$ est NP-difficile.*

Démonstration. Pour démontrer la NP-difficulté du problème P nous allons utiliser le problème Partition, qui est NP-difficile (voir Garey et Johnson (1979)). Ce problème peut être formulé de manière suivante :

Problème Partition : *Soit $a_1, \ldots, a_m$ des entiers positifs, tels que $\sum_{j=1}^{m} a_j = 2A$. Le problème* Partition *consiste à déterminer s'il existe un vecteur $z = (z_1, \ldots, z_m)$, où $z_i \in \{0, 1\}$, $i = 1, \ldots, m$, tel que $\sum_{j=1}^{m} a_j z_j = A$?*

A partir d'une instance du problème Partition nous pouvons construire une instance du problème P comme suit : soit $d = 2^m - 1$ et

$$f(x) = \begin{cases} 0, & \text{si } x \in \{0, 1, \ldots, d-1\}, \\ x - d + 1, & \text{si } x \in \{2d+1, 2d+2, \ldots\}, \\ & \text{ou } x \in \{d, d+1, \ldots, 2d\} \text{ et } \sum_{j=1}^{m} a_j r_j (x-d) \neq A, \\ x - d, & \text{si } x \in \{d, d+1, \ldots, 2d\} \text{ et } \sum_{j=1}^{m} a_j r_j (x-d) = A. \end{cases} \tag{3.5}$$

Ici $r(x-d) = \Big(r_1(x-d), \ldots, r_m(x-d)\Big)$ est la *représentation binaire* du nombre $x-d$, $x-d \in \{0, 1, \ldots, d\}$ en utilisant m bits comme suit $x-d = \sum_{i=1}^{m} 2^{i-1} r_{k-i+1}(x-d)$. Nous pouvons vérifier que $f(x)$ est une fonction en nombres entiers non négative et non décroissante telle que $f(x) \leq x$ pour $x \geq 1$.

La longueur du nombre $d = 2^m - 1$ en codage binaire ne dépasse pas m. En utilisant la formule (3.5), nous pouvons calculer la valeur de $f(x)$ en $O(\log x)$ pour $x \in \{0, 1, \ldots, d-1\}$ et $x \in \{2d+1, 2d+2, \ldots\}$. Si $x \in \{d, d+1, \ldots, 2d\}$, la représentation binaire de $r(x-d)$ peut être trouvée en $O(m)$. L'utilisation de $r(x-d)$ permet de calculer la valeur de $f(x)$ en $O(m)$ aussi. Par conséquent, cette réduction est polynomiale en longueur de PARTITION (m).

La définition de la fonction $f(x)$ implique qu'il existe $x^* \in \mathbb{N}$ tel que $x^* - f(x^*) \geq d$ si et seulement si $x \in \{d, d+1, \ldots, 2d\}$ et $\sum_{j=1}^{m} a_j r_j(x^*-d) = A$, c.-à-d. le problème PARTITION a une solution. $\square$

La démonstration du théorème 3.1 est basée sur la démonstration de Cheng et Kovalyov (2002). Nous devons simplement tenir compte des particularités de notre problème, c.-à-d. le fait que $f(x)$ est une fonction entière croissante et non négative, et que $f(x) \leq x$ pour $x \geq 1$.

Les deux Corollaires suivants sont vérifiés :

Corollaire 3.1 *Le problème P_i est NP-difficile pour chaque $i = 1, \ldots, n$.*

Corollaire 3.2 *Le problème* P-TEMPS-TAILLES *est NP-difficile, même si $n = 1$ et sa fonction $f_1(x)$ est donnée par une boite noire.*

Les problèmes P_i sont NP-difficiles en raison de l'incertitude au niveau des fonctions $f_i(x)$. En revanche, pour les fonctions $f_i(x)$ spécifiques il peut exister des approches efficaces pour résoudre le problème PTT. Nous considérons ci-dessous un tel cas.

Soit $f_i(x) = \lfloor \alpha_i x + \beta_i x^{\frac{1}{2}} \rfloor$, où α_i et β_i sont des valeurs réelles non négatives telles que $f_i(x) \leq x$ pour $x \geq 1$, $i = 1, \ldots, n$. Rappelons que $\lfloor b \rfloor$ est la valeur entière la plus grande qui est inférieure ou égale à b. Le cas où $\beta_i = 0$, $i = 1, \ldots, n$ et $f_i(x) = \lfloor \alpha_i x \rfloor$, $i = 1, \ldots, n$, peut être utilisé pour modéliser la situation où le pourcentage des pièces défectueuses pour un produit est connu. Flapper *et al.* (2002) ont souligné que dans les systèmes de production l'incertitude dans le rendement existe souvent. Toutefois, ayant les données statistiques adéquates, il est possible de faire une estimation fiable du rendement, et s'assurer qu'en moyenne la quantité de pièces défectueuses peut être présentée comme un pourcentage du nombre total des pièces fabriquées. Ce pourcentage est assez stable. Inderfurth *et al.* (2005),

Teunter et Flapper (2003), Inderfurth *et al.* (2006), Inderfurth *et al.* (2007), Chiu *et al.* (2007) et Buscher et Lindner (2007) ont utilisé cette hypothèse dans leurs travaux. Dans la littérature en contrôle de qualité cette manière de modéliser un rendement aléatoire est connue sous le terme de *fraction défectueuse* (voir Gitlow (1989)).

Nous allons considérer le cas $f_i(x) = \lfloor \alpha_i x + \beta_i x^{\frac{1}{2}} \rfloor$, $i = 1, \ldots, n$. L'inégalité $\lfloor \alpha_i x + \beta_i x^{\frac{1}{2}} \rfloor \leq x$ est équivalente à $\alpha_i x + \beta_i x^{\frac{1}{2}} \leq x$, si x est un nombre entier. Transformons la dernière inégalité comme suit :

$$\alpha_i x + \beta_i x^{\frac{1}{2}} \leq x \Longleftrightarrow x^{\frac{1}{2}} \left(x^{\frac{1}{2}}(1 - \alpha_i) - \beta_i \right) \geq 0 \tag{3.6}$$

Nous considérons que $x \in \mathbb{N}$, alors $x^{\frac{1}{2}}$ est toujours positif et $x^{\frac{1}{2}}(1-\alpha_i)-\beta_i \geq 0$ est valide pour n'importe quelle valeur de $x \in \{1, 2, \ldots\}$, si elle est vérifiée pour $x = 1$, c.-à-d., l'inégalité $1 - \alpha_i \geq \beta_i$ est valide. Par conséquent, nous allons exiger $0 \leq \beta_i \leq 1 - \alpha_i < 1$, $i = 1, \ldots, n$.

Pour résoudre le problème P_i il faut trouver la valeur de x_i la plus petite possible respectant la condition suivante :

$$x_i - \lfloor \alpha_i x_i + \beta_i x_i^{\frac{1}{2}} \rfloor \geq d_i \tag{3.7}$$

où $x_i \in \mathbb{N}$, ce qui est équivalent à

$$\alpha_i x_i + \beta_i x_i^{\frac{1}{2}} < x_i - d_i + 1$$

Examinons la fonction suivante :

$$g_i(x_i) := (1 - \alpha_i)x_i - \beta_i x_i^{\frac{1}{2}} - d_i + 1 > 0 \tag{3.8}$$

Elle est convexe avec la valeur minimale au point

$$x_i^{min} = \frac{\beta_i^2}{4(1 - \alpha_i)^2}$$

Puisque $\beta_i \leq 1-\alpha_i,\ i = 1, \ldots, n$, alors $x_i^{min} = \frac{\beta_i^2}{4(1-\alpha_i)^2} < 1$ et chaque solution entière positive de (3.8) doit satisfaire $x_i > x_i^+$, où x_i^+ est la solution positive de l'équation $g_i(x_i) = 0$. Nous avons donc

$$x_i^+ = \frac{\beta_i^2 + \beta_i \sqrt{4(d_i - 1)(1 - \alpha_i) + \beta_i^2} + 2(d_i - 1)(1 - \alpha_i)}{2(1 - \alpha_i)^2} \tag{3.9}$$

En tenant compte de l'explication ci-dessus, nous pouvons déduire que les tailles de lots

optimales pour le problème PTT sont déterminées par le vecteur $x^* = (x_1^*, \ldots, x_n^*)$, où

$$x_i^* = \begin{cases} x_i^+ + 1, & \text{si } x_i^+ \text{ est entier} \\ \lceil x_i^+ \rceil, & \text{sinon} \end{cases} \quad i = 1, \ldots, n \tag{3.10}$$

Ici $\lceil \cdot \rceil$ est l'opérateur qui arrondit au plus petit entier supérieur. Pour le cas "fraction défectueuse", où $\beta_i = 0$, $i = 1, \ldots, n$, nous avons

$$x_i^* = \begin{cases} \frac{d_i - 1}{1-\alpha_i} + 1, & \text{si } \frac{d_i-1}{1-\alpha_i} \quad \text{est entier} \\ \lceil \frac{d_i-1}{1-\alpha_i} \rceil, & \text{sinon} \end{cases} \quad i = 1, \ldots, n \tag{3.11}$$

Nous concluons cette section avec la proposition suivante.

Proposition 3.1 *Le problème* P-Temps-Tailles *peut être résolu en* $O(n)$ *si* $f_i(x) = \lfloor \alpha_i x + \beta_i x^{\frac{1}{2}} \rfloor$, *où* α_i *et* β_i *sont les nombres réels non négatifs, tels que* $0 \leq \beta_i \leq 1 - \alpha_i < 1$, $i = 1, \ldots, n$.

3.4 Problème P-Coût

Dans le problème P-coût le coût total de non-satisfaction de la demande doit être minimisé sous la condition que l'horizon de planification (des temps de set-up et des temps de réparation inclus) soit inférieur ou égal à T_0. Nous formulons le problème P-Coût de la manière suivante :

Problème P-Coût :

$$\textit{Minimiser} \quad W(x) := \sum_{j=1}^{n} c_{\pi_j} \max\{0, d_{\pi_j} - (x_{\pi_j} - f_{\pi_j}(x_{\pi_j}))\} \tag{3.12}$$

$$s.\ c. \quad Q(x) := s_{0,\pi_1} + \sum_{j=2}^{n} s_{\pi_{j-1},\pi_j} + \sum_{j=1}^{n} t_{\pi_j} x_{\pi_j} + T(x) \leq T_0 \quad x_i \in \mathbb{N},\ i = 1, \ldots, n$$

Il n'y a pas d'obligation de satisfaire la demande pour la date T_0 (la livraison peut être retardée). En utilisant l'algorithme de programmation dynamique (voir Algorithme 3.1), proposée par Held et Karp (1962) et développé pour le problème du *Voyageur du Commerce*, nous pouvons trouver la permutation optimale pour tous les ensembles de lots. La complexité de cet algorithme est $O(n^2 2^n)$.

Dans cet algorithme, $S(Y, i)$ est le temps minimal de set-up total (la valeur optimale de set-up pour toute la séquence) pour fabriquer un ensemble de produit Y, à condition que le produit $i \in Y$ soit traité en dernier. Pour chaque ensemble Y, le temps de set-up cumulé minimal $S^*(Y)$ peut être calculé de la manière suivante :

$$S^*(Y) = \min_{i \in Y}\{S(Y, i)\} \tag{3.13}$$

en $O(|Y|)$. Toutes les valeurs $S^*(Y)$ peuvent être calculées en $O(n^2 2^n)$ temps. Enfin pour l'ensemble Y des produits à fabriquer et le temps total de set-up $S^*(Y)$, la permutation optimale $\pi^*(Y)$ peut être obtenue en $O(n)$ par *retour sur trace* ou *backtracking* (en anglais).

Algorithme 3.1: Programmation dynamique

Initialisation :
$Y \in \{1, \ldots, n\}$
$|Y| = 1$
pour chaque $(i = 1\,;\, i \leq n\,;\, i \leftarrow i + 1)$ **faire**
  $Y = \{i\}$
  $S(Y, i) \leftarrow s_{0,i}$
fin
Récursion :
pour chaque $(|Y| = 2\,;\, |Y| \leq n\,;\, |Y| \leftarrow |Y| + 1)$ **faire**
  $S(Y, i) = \min_{j \in Y \setminus \{i\}}\{S(Y \setminus \{i\}, j) + s_{j,i}\}$
fin

Dans le reste de cette section, nous allons considérer que la décision de sélection a été effectuée (nous savons quels produits seront effectivement fabriqués). Nous supposons qu'une permutation optimale de leurs lots a été trouvée. Pour faciliter la présentation soit $N = \{1, \ldots, n\}$ l'ensemble des produits sélectionnés pour la fabrication, alors nous avons la valeur de $S^*(N)$ pour le temps cumulé de set-up. De manière analogue, x_i indique la taille du lot de produit i, $i = 1, \ldots, n$ et $x = (x_1, \ldots, x_n)$ est le vecteur représentant les tailles de tous les lots. En tenant compte de l'ensemble N, nous pouvons réduire le problème P-Coût au problème de lotissement suivant (désigné comme P-Coût-Tailles ou PCT par la suite).

Problème P-Coût-Tailles :

$$\textit{Minimiser} \quad W(x) := \sum_{i=1}^{n} c_i \max\{0, d_i - (x_i - f_i(x_i))\} \tag{3.14}$$

$$s.\ c. \quad Q(x) := \sum_{i=1}^{n} t_i x_i + T(x) \leq T_1, \quad x_i \in \mathbb{N},\ i = 1, \ldots, n$$

où $T_1 = T_0 - S^*(N)$.

La fonction (3.14) minimise le coût total d'insatisfaction de la demande et la contrainte associée limite la période de planification à T_1.

Supposons que nous avons un seul produit à fabriquer, c.-à-d. $|N| = 1$ et la fonction $f_1(x)$ est représentée par une boîte noire. Alors le problème de décider s'il existe une solution x au problème PCT telle que $W(x) \leq 0$ est équivalent au problème de recherche d'une solution réalisable du problème P_i, qui est NP-difficile (voir le théorème 3.1). Par conséquent, la proposition suivante est vérifiée.

Proposition 3.2 *Problème* P-COÛT-TAILLES *est NP-difficile même si* $n = 1$ *et la fonction* $f_1(x)$ *donnée par une boîte noire.*

Comme dans la section précédente, nous allons étudier un cas particulier du problème PCT - "fraction défectueuse". Dans ce cas, comme précisé auparavant, les fonctions représentant le nombre de pièces défectueuses et le temps total de réparation sont les suivantes :

$$f_i(x) = \lfloor \alpha_i x \rfloor, \quad i = 1, \ldots, n, \quad \text{et} \quad T(x) = \sum_{i=1}^{n} \gamma_i t_i x_i \tag{3.15}$$

où α_i et γ_i sont des nombres réels, tels que $0 \leq \gamma_i < 1$, $0 \leq \alpha_i < 1$, pour $i = 1, \ldots, n$. Nous appelons ce problème P-COÛT-TAILLE-FD (ou PCT-FD), où l'abréviation "FD" signifie "fraction défectueuse". Nous formulons le problème PCT-FD comme suit :

Problème P-COÛT-TAILLE-FD :

$$\begin{aligned} \textit{Minimiser} \quad & W^{fd}(x) := \sum_{i=1}^{n} c_i \max\{0, d_i - (x_i - \lfloor \alpha_i x_i \rfloor)\} \\ s.c. \quad & Q^{fd}(x) := \sum_{i=1}^{n} (1 + \gamma_i) t_i x_i \leq T_1, \quad x_i \in \mathbb{N},\ i = 1, \ldots, n \end{aligned} \tag{3.16}$$

La fonction (3.16) et la contrainte qui suit sont des versions modifiées des fonctions (4.8) pour le cas "FD" du problème. Maintenant nous allons démontrer que contrairement au problème PTT, pour lequel il est possible de trouver une solution optimale en temps polynomial $O(n)$, le problème PCT-FD est NP-difficile.

Théorème 3.2 *Problème* PCT-FD *est NP-difficile même si* $\alpha_i = 1/3$, $d_i = 2$ *et* $\gamma_i = 0$, $i = 1, \ldots, n$.

Démonstration. Notons que $\gamma_i = 0$ signifie que nous ignorons les pannes. Nous allons

utiliser de nouveau le problème PARTITION (voir la formulation dans Théorème 3.1). A partir d'une instance du problème PARTITION nous construisons une instance du problème PCT-FD de la manière suivante :

$$n = m, \quad T_1 = 3A, \quad \alpha_i = 1/3, \quad d_i = 2, \quad \gamma_i = 0$$
$$\text{et} \quad c_i = t_i = a_i \quad \text{pour chaque} \quad i = 1, \ldots, n$$

Nous allons donc démontrer que le problème PARTITION a une solution **si et seulement** s'il existe une *solution réalisable* x, telle que $W^{fd}(x) \leq A$, pour l'instance du problème PCT-FD, construite ci-dessus, .

Condition nécessaire ("Si"). Supposons qu'il existe une solution réalisable $x = (x_1, \ldots, x_n)$ du problème PCT-FD telle que $W^{fd}(x) \leq A$. Les fonctions W^{fd} et Q^{fd} deviennent

$$W^{fd}(x) = \sum_{i=1}^{n} a_i \max\{0, 2 - (x_i - \lfloor \frac{1}{3} \rfloor)\} \quad \text{et} \quad Q^{fd}(x) := \sum_{i=1}^{n} a_i x_i \leq T_1 \tag{3.17}$$

Séparons les indices i, $i = 1, \ldots, n$ en deux ensembles X_1 and X_2 comme suit :

$$X_1 = \{i \mid x_i = 1\}, \; X_2 = \{i \mid x_i \geq 2\} \tag{3.18}$$

Remarquons que si $x_i \in \{2, 3, \ldots\}$, alors $\max\{0, 2 - (x_i - \lfloor \frac{x_i}{3} \rfloor)\} = 0$, car $x_i - \lfloor \frac{x_i}{3} \rfloor$ est toujours supérieur ou égal à 2. Ceci implique que les inégalités suivantes

$$W^{fd}(x) = \sum_{i \in X_1} a_i \leq A \quad \text{et} \quad Q^{fd}(x) = \sum_{i \in X_1} a_i + 2 \sum_{i \in X_2} a_i \leq 3A$$

doivent être vérifiées. Étant donné que $X_1 \cup X_2 = \{1, \ldots, m\}$ et $\sum_{i \in X_1 \cup X_2}^{m} a_i = 2A$, nous obtenons

$$Q^{fd}(x) = \sum_{i \in X_1} a_i + 2(2A - \sum_{i \in X_1} a_i) = 4A - \sum_{i \in X_1} a_i \leq 3A \tag{3.19}$$

Considérant que $W^{fd}(x) = \sum_{i \in X_1} a_i \leq A$, nous pouvons conclure que $\sum_{i \in X_1} a_i = A$. Alors l'ensemble $X := X_1$ est la solution requise du problème PARTITION.

Condition suffisante ("Seulement Si"). Soit X est une solution du problème PAR-

TITION. Déterminons le vecteur $x = (x_1, \ldots, x_n)$ de la manière suivante :

$$x_i = \begin{cases} 2, \text{ si } \quad i \in X \\ 1, \quad \text{sinon} \end{cases} \quad , i = 1, \ldots, n \tag{3.20}$$

Le coût total de retard $W^{fd}(x)$ est

$$W^{fd}(x) = \sum_{i=1}^{m} a_i \max\left\{0, 2 - \left(x_i - \left\lfloor \frac{x_i}{3} \right\rfloor\right)\right\} = \tag{3.21}$$

$$= \sum_{i=1}^{m} a_i \max\{0, 2 - x_i\} = \sum_{i=1}^{m} a_i(2 - x_i) = \sum_{i \notin X} a_i = A \tag{3.22}$$

et le temps total de fabrication est égal à

$$Q^{fd}(x) = \sum_{i=1}^{m} a_i x_i = 2 \sum_{i \in X} a_i + \sum_{i \notin X} a_i = 3A = T_1 \tag{3.23}$$

Ainsi le vecteur x est une solution réalisable de l'instance du problème PCT-FD construite ci-dessus. □

Dans la section suivante nous allons proposer une approche de type FPTAS pour résoudre le problème PCT-FD.

3.5 Fully Polynomial Time Approximation Scheme (FPTAS)

Nous commençons cette section par donner une définition formelle de FPTAS.

Définition 3.1
Un algorithme A est un FPTAS *pour un problème d'optimisation* P*, si pour un ensemble I de données du problème et une valeur $\varepsilon > 0$ fixée, l'algorithme A peut trouver une solution S du problème* P *avec les paramètres I en un temps polynômial en fonction de $\frac{1}{\varepsilon}$ et de la taille de* I*. Cette solution est telle que*

$$|SO(I) - S| \leq \varepsilon SO(I)$$

où SO(I) est la valeur de la solution optimale du problème P *avec les données I.*

Soit L le plus grand dénominateur parmi les fractions irréductibles représentant les nombres α_i (les coefficients des fonctions de rendement) et γ_i (les coefficients de fonction de temps de réparation), $i = 1, \ldots, n$. Et soit $p_{\max} = \max\{\max_{1\leq i\leq n}\{c_i, d_i, t_i\}, L\}$ le paramètre maximal. Alors la famille $\{A_\varepsilon\}$ des algorithmes $(1+\varepsilon)$-approchés construit un FPTAS pour le problème PCT-FD. La complexité de chaque algorithme A_ε est limitée par un polynôme de n, $\log p_{\max}$, et $1/\varepsilon$. La méthode FPTAS nous permet de trouver deux choses : la solution optimale et des solutions approchées.

Soit le vecteur $x^* = (x_1^*, \ldots, x_n^*)$ la solution optimale du problème PCT-FD. Rappelons que x est la solution réalisable pour ce problème si $Q^{fd}(x) \leq T_1$. Par conséquent, si $Q^{fd}(x^{(1)}) > T_1$ pour le vecteur unitaire $x^{(1)} = (1, \ldots, 1)$, alors il n'y a pas de solution réalisable pour le problème PCT-FD. Nous supposons donc que $Q^{fd}(x^{(1)}) \leq T_1$. Soit h_i la taille de lot minimale, telle que la demande pour des pièces i de bonne qualité soit satisfaite, alors

$$h_i = \min\{x \mid x \in \mathbb{N}, d_i - x + \lfloor \alpha_i x \rfloor \leq 0\} = \begin{cases} \frac{d_i-1}{1-\alpha_i} + 1, & \text{si } \frac{d_i-1}{1-\alpha_i} \text{ est entier} \\ \lceil \frac{d_i-1}{1-\alpha_i} \rceil, & \text{sinon} \end{cases} \quad i = 1, \ldots, n$$

Si $Q^{fd}(x^{(h)}) \leq T_1$, où $x^{(h)} = (h_1, \ldots, h_n)$, alors $x^* = x^{(h)}$ est une solution optimale, puisque $W^{fd}(x^{(h)}) = 0$. Supposons que ce n'est pas le cas, c.-à-d. que la période de planification T_1 est insuffisante pour satisfaire toutes les demandes, quand $x = x^{(h)}$.

Pour appliquer l'algorithme, nous avons besoin des bornes inférieure (BI) et supérieure (BS) pour le coût d'insatisfaction, telles que $0 < BI \leq W^{fd}(x^*) \leq BS$. Étant donné que $Q^{fd}\left(x^{(h)}\right) > T_1$, l'inégalité $W^{fd}(x) > 0$ est satisfaite pour chaque solution réalisable x, c.-à-d. que au moins une pièce d'un produit va manquer et $W^{fd}(x^*) \geq \min_{1\leq i\leq n}\{c_i\}$. La borne inférieure BI peut être déterminée comme suit

$$BI := \min_{1\leq i\leq n}\{c_i\} \geq 1 \tag{3.24}$$

Pour trouver une borne supérieure BS, nous utilisons l'algorithme 3.2.

Pour présenter FPTAS, nous allons formuler un problème "arrondi" ("rounded" en anglais), nommé P-Rou. Chaque algorithme exact pour le problème P-Rou est un algorithme $(1+\varepsilon)$ approché pour le problème PCT-FD (la démonstration est similaire à Kovalyov (1996)). Soit $\delta = \max\{1, \frac{\varepsilon BI}{n}\}$ un paramètre d'échelle. Remarquons que si $\varepsilon = 0$ c'est-à-dire

Algorithme 3.2: Borne supérieure de coût d'insatisfaction

```
1   Générer une solution x^0 = (x_1^0, x_2^0, ..., x_n^0) où x_i^0 ∈ [1, x_i^h] ∩ ℕ // solution
      initiale
2   Calculer Q^fd(x^0) = Σ_{i=1}^n (1 + γ_i) t_i x_i^0
3   si (Q^fd(x^0) < T_1) alors
4       tant que (Q^fd(x^0) < T_1) faire
5           pour chaque (i = 1; i ≤ n; i ← i + 1) faire
6               si (x_i^0 < x_i^h) alors
7                   x_i^0 ← x_i^0 + 1
8               fin
9           fin
10      fin
11  sinon
12      tant que (Q^fd(x^0) > T_1) faire
13          pour chaque (i = 1; i ≤ n; i ← i + 1) faire
14              si (x_i^0 > 1) alors
15                  x_i^0 ← x_i^0 − 1
16              fin
17          fin
18      fin
19  fin
20  Calculer le coût d'insatisfaction de la demande pour la solution x^0
```

$\delta = 1$, alors la solution trouvée par FPTAS est une solution optimale du problème PCT-FD.

Problème P-Rou :

$$\textit{Minimiser} \;\; w^{fd}(x) := \sum_{i=1}^{n} \left\lfloor \frac{v_i(x_i)}{\delta} \right\rfloor \tag{3.25}$$

$$s.\ c.\ q^{fd}(x) \leq T_1, \quad \text{et} \quad x_i \in \{r_i(0), r_i(1), \ldots, r_i(\left\lfloor \frac{BS}{\delta} \right\rfloor)\}, i = 1, \ldots, n \tag{3.26}$$

où

$$v_i(x_i) = c_i \max\{0, d_i - (x_i - \lfloor \alpha_i x_i \rfloor)\} \tag{3.27}$$

$$r_i(l) = \min\{x \mid x \in Z_+, \left\lfloor \frac{v_i(x)}{\delta} \right\rfloor \leq l\}, \; l = 0, 1, \ldots, \left\lfloor \frac{BS}{\delta} \right\rfloor \tag{3.28}$$

Les inégalités

$$\left\lfloor \frac{v_i(x)}{\delta} \right\rfloor \leq l \quad \text{et} \quad d_i - x + \lfloor \alpha_i x \rfloor < \frac{(l+1)\delta}{c_i}$$

sont équivalentes. Étant donné que la partie gauche de la dernière inégalité est toujours une valeur entière, nous obtenons :

$$d_i - x + \lfloor \alpha_i x \rfloor < \frac{(l+1)\delta}{c_i} \Longleftrightarrow d_i - x + \lfloor \alpha_i x \rfloor < \left\lceil \frac{(l+1)\delta}{c_i} \right\rceil$$
$$\Longleftrightarrow \left\lceil \frac{(l+1)\delta}{c_i} \right\rceil + x - d_i > \alpha_i x$$

La dernière inégalité nous donne les valeurs de x :

$$x > \frac{d_i - \lceil \frac{(l+1)\delta}{c_i} \rceil}{1 - \alpha_i} := h_i^{(l)} \tag{3.29}$$

Alors

$$r_i(l) = \begin{cases} \max\{1, h_i^{(l)} + 1\}, & \text{si } h_i^{(l)} \text{ est entier} \\ \max\{1, \lceil h_i^{(l)} \rceil\}, & \text{sinon} \end{cases}$$
$$l = 0, 1, \ldots, \left\lfloor \frac{BS}{\delta} \right\rfloor, \quad i = 1, \ldots, n$$

Tous les paramètres du problème P-Rou peuvent être déterminés en $O\left(n \left\lfloor \frac{BS}{\delta} \right\rfloor\right) = O\left(n^2 \frac{BS}{BI}\right)$ temps.

Soit le vecteur x^0 une solution optimale du problème P-Rou et $q_j^{fd}(f)$ représentent la valeur minimale de la fonction $q_j^{fd}(x) := \sum_{i=1}^{j}(1+\gamma_i)t_i x_i$, où $x = (x_1, \ldots, x_j)$ tel que $\sum_{i=1}^{j} \left\lfloor \frac{v_i(x_i)}{\delta} \right\rfloor = f$. Le problème P-Rou peut être résolu par l'approche de programmation dynamique 3.3.

La complexité de l'algorithme A_ε est $O\left(\frac{n^3}{\varepsilon^2}\left(\frac{BS}{BI}\right)^2\right)$. Pour améliorer sa performance, nous pouvons appliquer la procédure d'amélioration des bornes (*Bound Improvement Procedure ou BIP*) expliquée dans Tanaev *et al.* (1998) (traduction anglaise dans Chubanov *et al.* (2006)). Cette procédure est adaptée pour un problème de minimisation quelconque et permet de trouver la valeur W^0 telle que $0 < W^0 \leq W^{fd}(x^*) \leq 3W^0$ en $O\left(n^3 \log\log \frac{BS}{BI}\right)$ temps.

Soit W^* la valeur minimale pour le coût d'insatisfaction recherché. Nous allons également utiliser la notation $A_\varepsilon(BI, BS, \varepsilon)$ pour l'algorithme $\{A_\varepsilon\}$, où BI est la borne inférieure, BS est la borne supérieure de coût d'insatisfaction, et ε est la "précision" (la borne supérieure

Algorithme 3.3: Algorithme A_ε :FPTAS pour le problème PCT-FD

1 $j \leftarrow 0$

2 ***Initialisation*** :
3 **si** $(f = 0 \; et \; j = 0)$ **alors**
4 $\quad Q_j^{fd}(f) = 0$
5 **sinon**
6 $\quad Q_j^{fd}(f) = \infty$
7 **fin**

8 ***Récursion*** :
9 **tant que** $(j < n)$ **faire**
10 $\quad j \leftarrow j + 1$
11 $\quad$ **pour chaque** $(f = 0; f \leq \left\lfloor \frac{BS}{\delta} \right\rfloor; f \leftarrow f + 1)$ **faire**
12 $\quad\quad Q_j^{fd}(f) = \min \left\{ Q_{j-1}^{fd}\left(f - \left\lfloor \frac{v_j(x_j)}{\delta} \right\rfloor\right) + (1 + \gamma_j) t_j x_j \mid x_j \in \{r_j(0), \ldots, r_j(f)\} \right\}$
13 $\quad$ **fin**
14 **fin**

15 ***Solution optimale*** :
16 *Calculer* $W^{fd}(x^0) = \min \left\{ f \mid Q_n^{fd}(f) \leq T_1, \; f = 0, 1, \ldots, \left\lfloor \frac{BS}{\delta} \right\rfloor \right\}$ `// le coût d'insatisfaction minimal`
17 *Trouver la solution optimale* x^0 *en utilisant la méthode de retour sur trace*

pour l'erreur de calcul). Nous pouvons appliquer la procédure BIP puisque l'algorithme vérifie les propriétés suivantes :

- Pour chaque valeur positive de BI et BS, l'algorithme $A_\varepsilon(BI, BS, \varepsilon)$ trouve une solution réalisable W^A telle que la condition suivante soit vérifiée :

$$W^A \leq BI + BS \quad \text{si} \quad W^* < BS$$

- Le temps de calcul de $A_\varepsilon(BI, BS, \varepsilon)$ est limité par $P(n, \frac{BS}{BI}$.

L'approche complète avec la procédure BIP intégrée est présentée ci-dessous (Algorithme 3.4).

La famille des algorithmes $\{A_\varepsilon\}$ contenant la procédure BIP constitue un FPTAS pour le problème PCT-FD. L'application de la procédure BIP permet de réduire considérablement le temps de calcul de l'algorithme. Le théorème suivant est vérifié.

Algorithme 3.4: L'approche complète

```
1   Trouver BI et BS // en utilisant l'equation (3.24) et l'algorithme 3.2
2   W^0 ← BI,  j ← 1,  a_j ← 0
3   Trouver b_j ∈ ℕ tel que (2^{b_j−1} BI) < BS ≤ (2^{b_j} BI)
4   k_j ← ⌈(a_j + b_j)/2⌉
5   W^{(k_j)} ← 2^{k_j−1} BI
6   Répéter :
7   Appliquer A_ε(W^{(k_j)}, 2W^{(k_j)}, 1)
8   si (W^A ≤ 3W^{(k_j)}) alors
9       F^0 ← W^{(k_j)}
10      si (k_j = b_j) alors
11          Quitter
12      fin
13      si (k_j < b_j) alors
14          a_{j+1} ← a_j
15          b_{j+1} ← k_j
16      fin
17  sinon
18      si (k_j = b_j) alors
19          Quitter
20      fin
21      si (k_j < b_j) alors
22          a_{j+1} ← a_j
23          b_{j+1} ← k_j
24      fin
25  fin
26  j ← j + 1
27  Appliquer A_ε(BI, BS, ε) avec ε choisi
```

Théorème 3.3 *Il existe un FPTAS pour le problème* PCT-FD *avec le temps de calcul* $O\Big(\frac{n^3}{\varepsilon^2} + n^3 \log\log(\sum_{i=1}^{n} c_i d_i)\Big)$.

3.6 Expérimentations numériques avec FPTAS

Les objectifs des expériences numériques sont : 1) évaluer les temps de calcul de FPTAS pour différentes combinaisons de paramètres du problème PCT-FD, et 2) analyser la qualité

des solutions obtenues avec le FPTAS . Toutes les expérimentations ont été exécutées sur un SUN UltraSPARC IIIi avec 1593 Mhz CPU et 16Gb de RAM. Pour chaque instance du problème, les paramètres ont été générés à partir d'une distribution uniforme.

Nous avons fait varier certains paramètres pour avoir des familles d'instances. Les intervalles de cette variation sont présentés dans le tableau 3.1. Notons que $T_1 = T - ratio * T^*$, où T^* est le temps minimal, nécessaire pour le traitement de toutes les pièces d'une solution optimale du problème PTT-FD (voir l'equation (3.11)), qui sont les tailles de lots minimales pour que la demande pour chaque type de produits soit satisfaite. Toutes les combinaisons possibles des intervalles de valeurs des paramètres donnent 36 familles d'instances. Nous avons généré 10 instances pour chacune des familles, ce qui revient à $10 \times 36 = 360$ instances pour chaque taille de problème.

TABLE 3.1 – Les intervalles pour la génération des données initiales des instances

Paramètre	*Intervalles*		
Demande d_i	$[1; 5]$	$[1; 20]$	$[1; 100]$
Coût d'insatisfaction c_i	$[1; 1]$	$[1; 5]$	$[1; 20]$
Taux de rebuts α_i	$[0,001; 0,3]$	$[0,2; 0,5]$	
Taux de réparation γ_i	$[0,001; 0,4]$		
T-ratio	$[0,75; 0,9]$	$[0,9; 0,95]$	
Temps de fabrication t_i	$[10; 50]$		

Le temps de calcul de FPTAS en mode exact est présenté dans la Figure 3.2. Nous avons ajouté trois courbes de tendance pour évaluer la complexité expérimentale de l'algorithme et pouvoir le comparer avec la complexité théorique (au pire des cas). La courbe grise avec des noeuds représente le temps moyen de calcul de FPTAS pour le nombre de lots $n \in \{10, \ldots, 150\}$, la courbe bleu est une courbe de régression polynomiale avec un degré maximal égal à 2 (*C. de T. P2* sur la figure) ; la courbe rouge *C. de T. P3* est une courbe de régression de degré 3 ; la courbe verte *C. de T. Puissance* représente la courbe de régression puissance. Au-dessous du graphique nous affichons les équations correspondantes pour chaque courbe et les coefficients de détermination R^2. Pour les instances générées, les trois courbes de tendance sont proches, mais la régression puissance a la valeur de R^2 la plus élevée.

La Figure 3.3 illustre le temps CPU de la procédure BIP par rapport au temps moyen de calcul de FPTAS. Il y a trois courbes pour le BIP en fonction de la taille du problème : le

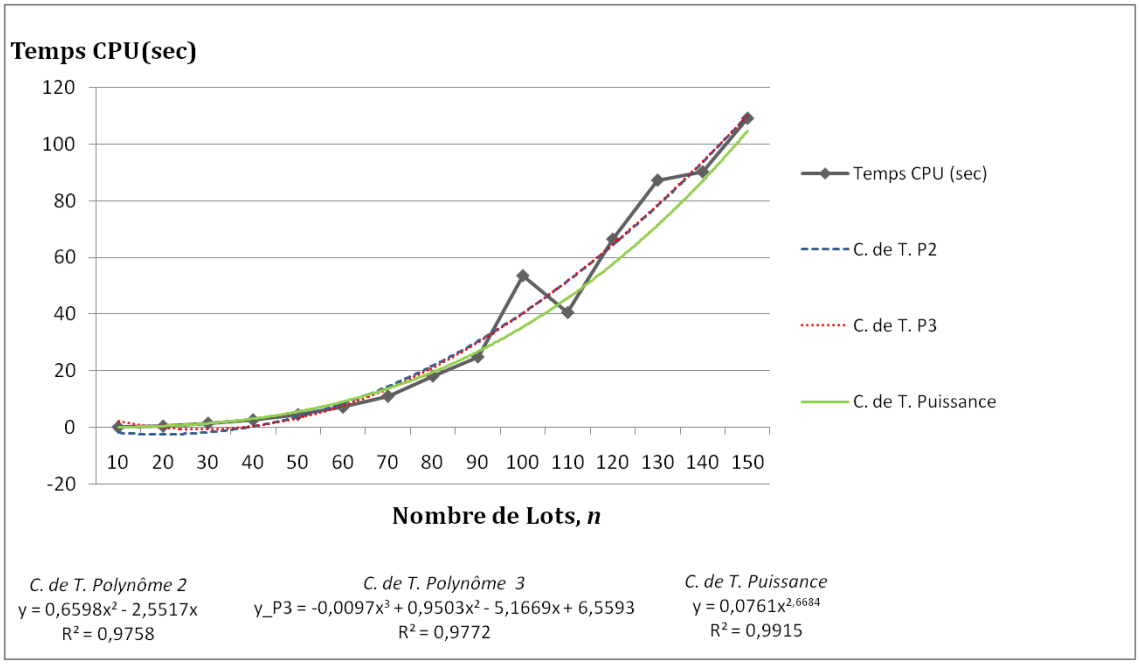


FIGURE 3.2 – Temps de calcul de FPTAS (sec.) pour $n \in [10; 150]$ en mode exact

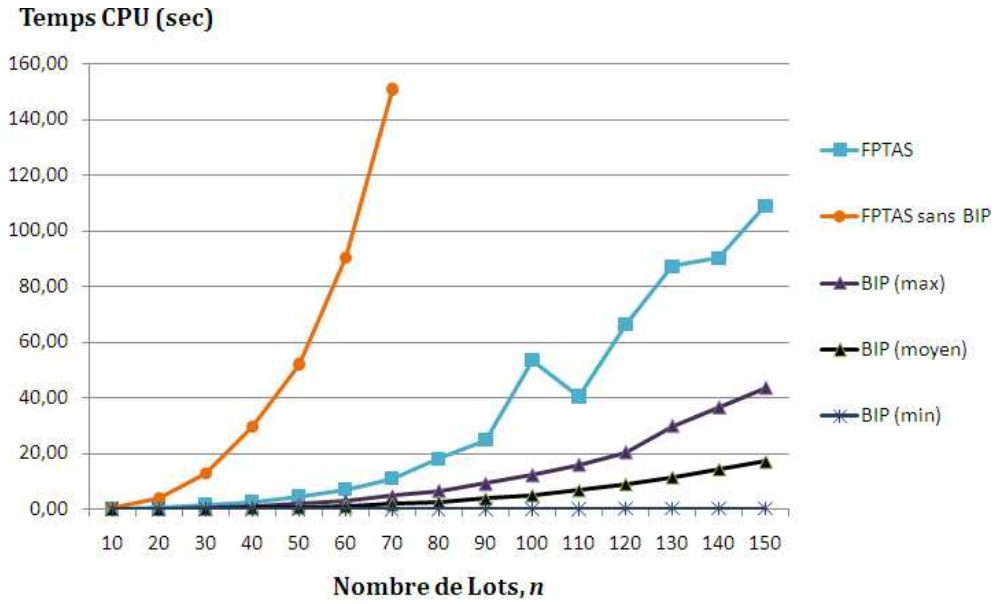


FIGURE 3.3 – Temps de calcul (sec.) avce et sans BIP pour $n \in [10; 150]$ en mode exact

temps minimal *"BIP (min)"* , moyen *"BIP (moyen)"* sur 360 instances et le temps maximal *"BIP (max)"* pour chaque taille de problème n, $n \in \{10, 20, \ldots, 150\}$. Nous constatons qu'il y a des cas où le temps de calcul de BIP prend une partie non négligeable du temps de calcul

de l'algorithme en général. Il existe même des cas où cette partie prend 90% du temps total. En même temps, sans procédure BIP, FPTAS est beaucoup plus lent (voir la courbe orange *"FPTAS sans BIP"*), l'utilisation de BIP donc est justifiée malgré son coût important en terme de calcul.

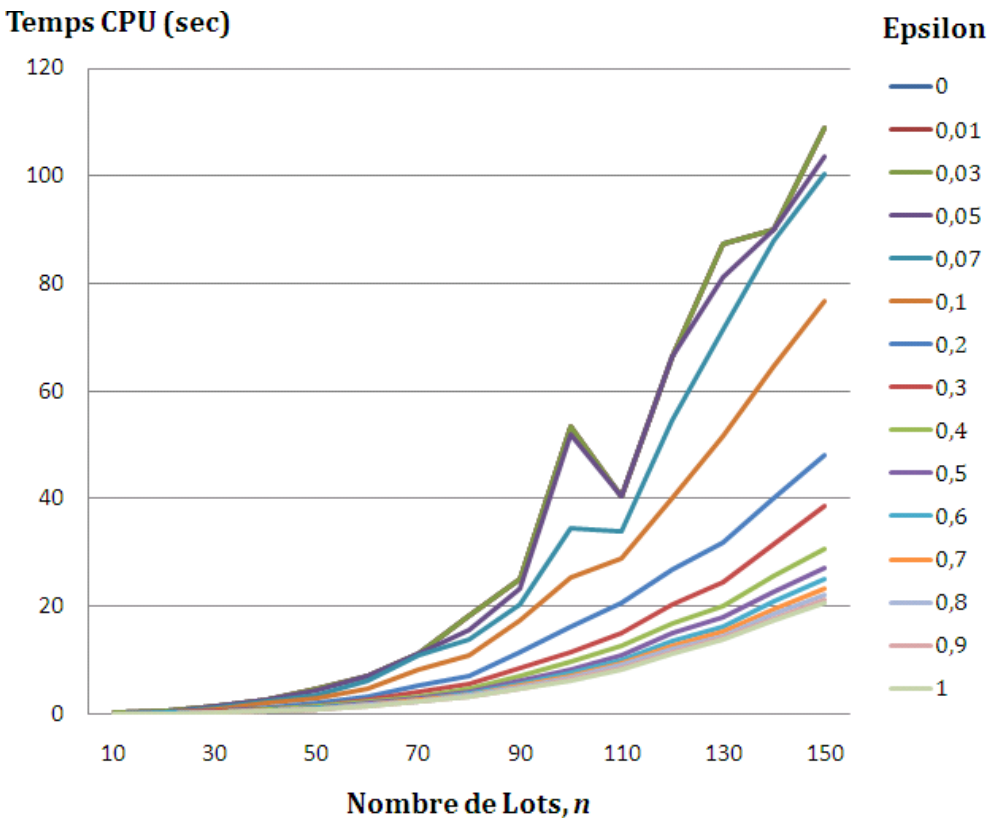


FIGURE 3.4 – Temps de calcul de FPTAS (sec.) pour $n \in [10; 150]$

Le temps de calcul de FPTAS (sec.) est présenté dans la figure 3.4. Pour chaque taille de problème (nombre de lots) $n \in \{10, 20, \ldots, 150\}$, 10 instances ont été générées et testées pour les valeurs ε, $\varepsilon = 0,01;\ 0,03;\ 0,05;\ 0,07;\ 0,1;\ 0,2;\ 0,3;\ 0,4;\ 0,5;\ 0,6;\ 0,7;\ 0,8;\ 0,9;\ 1$. Les courbes qui correspondent à $\varepsilon = 0,01;\ 0,03;\ 0,05$ ne sont pas visibles, parce que dans ces cas l'algorithme a trouvé des solutions exactes pour toutes les instances du problème, donc ces courbes sont cachées par la courbe $\varepsilon = 0$. L'analyse de ces résultats montre que le temps de calcul augmente quand le nombre de lots augmente et la borne supérieure ε d'erreur relative diminue. De plus, le nombre de lots semble avoir beaucoup d'influence sur le temps de calcul. Pour des lots de taille petite ($x_i = 30$ par exemple), l'influence de ε est beaucoup moins importante (voir négligeable) que pour les grands lots (voir les chiffres dans l'Annexe B.1).

La figure 3.5 donne les erreurs relatives des solutions approchées obtenues avec FPTAS. Les valeurs numériques sont dans le tableau B.2 de l'Annexe. Toutes les valeurs ont été arrondies à la deuxième décimale. A partir de la figure nous pouvons constater que les erreurs relatives réelles (pour les instances testées) sont beaucoup plus petites que la borne supérieure ε.

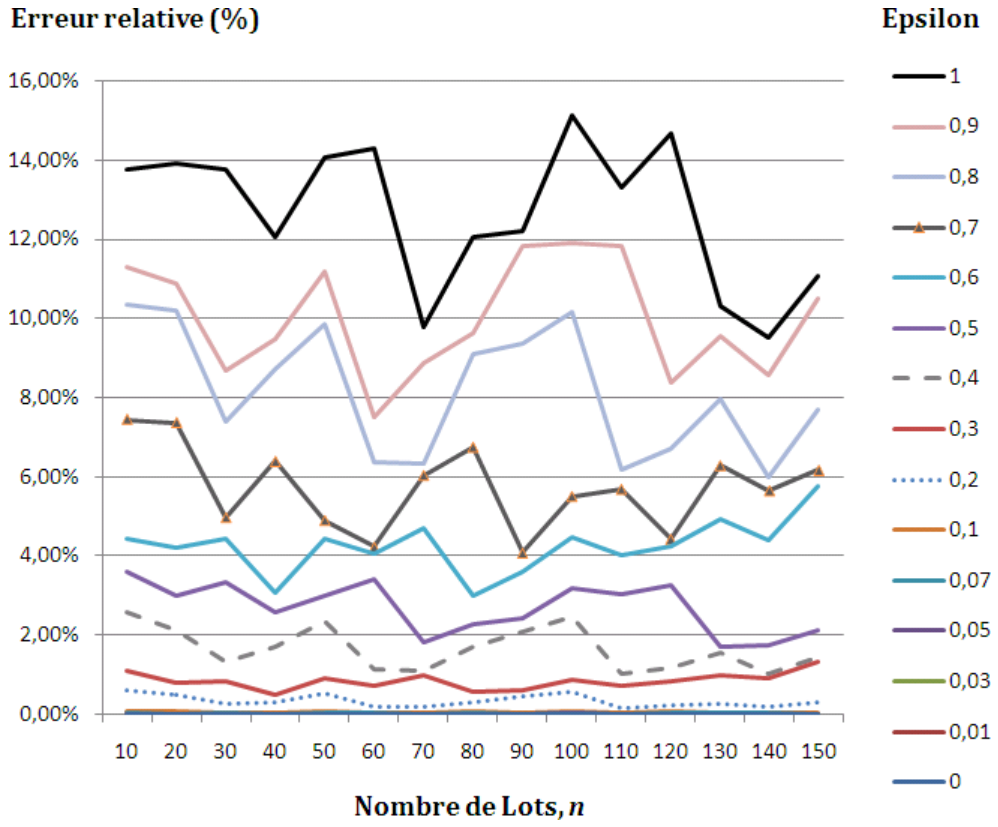


FIGURE 3.5 – Les erreurs relatives de FPTAS (sec.) pour $n \in [10; 150]$

Conclusion

Dans ce chapitre nous avons étudié deux problèmes P-TEMPS et P-COÛT. L'objectif était de trouver les quantités de pièces de chaque type à fabriquer et leur séquence afin d'optimiser un critère : minimiser le makespan pour le problème P-TEMPS et minimiser le coût total d'insatisfaction pour les pièces non-fabriquées pour le problème P-COÛT. Nous avons démontré que pour ces deux problèmes les décisions de lotissement et de séquencement peuvent être prises séparément. Ensuite nous avons prouvé que les problèmes de lotissement, P-TEMPS-TAILLES et P-COÛT-TAILLES, sont NP-difficiles, si les fonctions représentant le

nombre de pièces défectueuses sont aléatoires. Le problème P-Coût-Tailles est NP-difficile même dans le cas où les pourcentages de pièces défectueuses dans un lot sont connus pour chaque type de produit (cas "fraction défectueuse" ou FD). Nous avons montré qu'une solution optimale du problème P-Temps-Tailles dans le cas FD peut être trouvée en $O(n)$. Pour le problème P-Coût-Tailles-FD, nous avons développé une approche FPTAS avec le temps $O\left(\frac{n^3}{\varepsilon^2} + n^3 \log\log(\sum_{i=1}^{n} c_i b_i)\right)$. Les expérimentations numériques montrent que l'algorithme FPTAS peut résoudre des problèmes de grande taille en un temps raisonnable et que l'erreur relative expérimentale est beaucoup plus petite que la borne d'erreur supérieure ε.

Chapitre 4

Modèle Déterministe : programmes linéaires

Dans ce chapitre nous reprenons le problème P-Coût-Tailles *(ou* PCT*) présenté dans le chapitre précédant. Nous proposons deux modèles linéaires pour résoudre le cas "fraction défectueuse" du problème* PCT *(où le pourcentage d'articles défectueux est donné pour chaque type de produit, et où le temps de réparation dépendra directement du temps total de fabrication). En utilisant des tests numériques, nous étudions la performance des approches proposées et les comparons à l'approche FPTAS, proposée dans le chapitre précédent. Ensuite, en prenant une fonction plus générale pour la quantité de pièces défectueuses, nous montrons comment ajuster la méthode* FPTAS *ainsi qu'un des modèles linéaires pour résoudre ce nouveau problème. Les résultats numériques contenant les performances des deux méthodes pour ce problème sont présentés.*

4.1 Modèles Linéaires pour le problème P-Coût-tailles-FD

Comme c'était évoqué précédemment, nous allons commencer par présenter deux modèles linéaires pour le problème PCT-FD. Toutes les hypothèses prises auparavant restent valides. Rappelons la formulation du problème :

Problème P-Coût-Taille-FD :

$$\begin{aligned} &\textit{Minimiser} \quad W^{fd}(x) := \sum_{i=1}^{n} c_i \max\{0, d_i - (x_i - \lfloor \alpha_i x_i \rfloor)\} \\ &s.c. \quad Q^{fd}(x) := \sum_{i=1}^{n} (1+\gamma_i) t_i x_i \leq T_1, \quad x_i \in \mathbb{N},\ i = 1, \ldots, n \end{aligned} \tag{4.1}$$

4.1.1 Modèle en variables entières

Pour obtenir le modèle linéaire en variables entières nous allons linéariser la fonction objective. Comme il était montré auparavant, la solution optimale x^* (les tailles de lots) du problème (4.1), sans la contrainte sur la période de planification, peut être trouvée comme suit :

$$x_i^* = \begin{cases} \frac{d_i - 1}{1-\alpha_i} + 1, & \text{si } \frac{d_i-1}{1-\alpha_i} \text{ est entier} \\ \lceil \frac{d_i-1}{1-\alpha_i} \rceil, & \text{sinon} \end{cases} \qquad i = 1, \ldots, n \tag{4.2}$$

Nous allons utiliser les valeurs x_i^* comme les limites supérieures pour chaque taille de lot x_i recherchée, $i = 1, \ldots, n$:

$$1 \leq x_i \leq x_i^*$$

Soit $y_i = \lfloor \alpha_i x_i \rfloor$. En transformant cette égalité nous avons $y_i \leq \alpha_i x_i < y_i + 1$, d'où nous obtenons le système d'inégalités suivant :

$$\begin{cases} \alpha_i x_i - y_i \geq 0, \\ \alpha_i x_i - y_i < 1, \end{cases} \qquad i = 1, \ldots, n \tag{4.3}$$

Soit $\lambda_i = \max\{0, d_i - (x_i - \lfloor \alpha_i x_i \rfloor)\} = \max\{0, d_i - (x_i - y_i)\}$. Cette dernière équation peut être remplacée par les deux inégalités suivantes :

$$\lambda_i \geq d_i - (x_i - y_i)$$

$$\lambda_i \geq 0$$

où $i \in \{1, \ldots, n\}$.

Donc nous avons tout ce qu'il faut pour avoir un modèle linéaire.

Modèle Linéaire 1 :

Fonction objectif :

$$\textit{Minimiser} \quad \sum_{i=1}^{n} c_i \lambda_i \tag{4.4}$$

Sous contraintes :

$$\sum_{i=1}^{n} t_i \left(1+\gamma_i\right) x_i \leq T_1$$
$$\alpha_i x_i - y_i \geq 0, \ \forall i \in \{1,\ldots,n\}$$
$$\alpha_i x_i - y_i \leq 1-\varepsilon, \ \forall i \in \{1,\ldots,n\}$$
$$\lambda_i \geq d_i - \left(x_i - y_i\right), \ \forall i \in \{1,\ldots,n\}$$

où ε est une valeur positive, suffisamment petite pour ne pas influencer la solution du problème. Les intervalles de valeurs possibles, $i \in \{1,\ldots,n\}$:

$$1 \leq x_i \leq x_i^*$$
$$0 \leq y_i \leq \lfloor \alpha_i x_i^* \rfloor$$
$$\lambda_i \geq 0$$

Les définitions des types de variables à utiliser :

$$x_i, y_i \in Z, \quad \lambda_i \in R, \quad \forall i \in \{1,\ldots,n\}$$

4.1.2 Modèle avec des variables binaires

Pour formuler le deuxième modèle linéaire, nous introduisons des variables binaires λ_{ij} :

$$\lambda_{ij} = \begin{cases} 1, \text{ si la taille de lot } \ i \ \text{ égale à } \ j \\ 0, \text{ sinon} \end{cases} \qquad i = 1,\ldots,n, \ j = 1,\ldots,x_i^* \tag{4.5}$$

Le **Modèle Linéaire 2** est donc le suivant :

Fonction objectif :

$$Minimiser \sum_{i=1}^{n} \sum_{j=1}^{x_i^*} c_i \max\{0, d_i - (j - \lfloor \alpha_i j \rfloor)\} \lambda_{ij} \tag{4.6}$$

Contraintes :

$$\sum_{i=1}^{n} \sum_{j=1}^{x_i^*} t_i \left(1 + \gamma_i\right) j \lambda_{ij} \leq T_1$$

$$\sum_{j=1}^{x_i^*} \lambda_{ij} = 1, \forall i \in \{1, \ldots, n\}$$

Types des variables :

$$\lambda_{ij} \in \{0, 1\}, \ \forall i \in \{1, \ldots, n\}, \text{ et } j \in \{1, \ldots, x_i^*\}$$

4.1.3 Expérimentations numériques - comparaison FPTAS avec les deux modèles linéaires

Dans cette section nous allons comparer les performances des deux modèles linéaires présentés ci-dessus, avec ceux de la méthode FPTAS. Les modèles linéaires ont été programmés en C++ et ont été résolus avec CPLEX v.11. Pour FPTAS nous prenons $\varepsilon = 0$, c.-à-d. nous l'utilisons en mode exact, puisque CPLEX a résolu toutes les instances de manière exacte. Dans ce qui suit le modèle linéaire en variables entières sera appelé CPLEX(Ent) et le modèle avec les variables binaires CPLEX(Bin).

Le tableau 4.1 présente les temps CPU moyens de calcul pour les trois méthodes, obtenus en testant 360 instances pour chaque taille de lot. Nous pouvons constater que la méthode la plus rapide pour l'ensemble des instances générées est CPLEX(Ent). Notre réalisation de la méthode FPTAS est beaucoup plus lente que les modèles linéaires utilisés avec CPLEX.

Le tableau 4.2 présente les résultats pour le cas où nous faisons varier le pourcentage de rebut. Avec des valeurs de α plus petites, les modèles linéaires sont plus rapides. Pour FPTAS, même si la tendance générale parait être la même il y a 5 exceptions (en gras dans le tableau).

Le tableau 4.3 montre l'évolution des temps de calcul des trois méthodes en fonction de

Table 4.1 – Temps de calcul moyen de FPTAS, CPLEX(Ent) et CPLEX (Bin)

Nombre de lots	*Temps CPU, sec*		
	CPLEX (Ent)	***CPLEX (Bin)***	***FPTAS***
10	0,02	0,04	0,10
20	0,03	0,07	0,45
30	0,03	0,10	1,48
40	0,04	0,13	2,66
50	0,04	0,16	4,51
60	0,05	0,19	7,15
70	0,06	0,22	11,01
80	0,06	0,25	18,11
90	0,07	0,30	24,91
100	0,08	0,32	53,52
110	0,09	0,35	40,53
120	0,10	0,39	66,40
130	0,11	0,42	87,22
140	0,13	0,48	90,17
150	0,13	0,51	109,05

la demande. Le temps de calcul le plus élevé correspond au plage de la demande $[1; 100]$. En passant de $[1; 100]$ à $[1; 5]$ le temps de calcul moyen de FPTAS a été réduit de $45,8$ fois, le temps de CPLEX(Ent) de $4,4$ fois et le temps de CPLEX(Bin) de $22,2$ fois (pour $n = 150$ lots). L'approche FPTAS est plus sensible que les modèles linéaires au longueur de la plage de la demande.

Les temps de calcul moyens pour les exemples où le coût de non-réalisation unitaire est soit égal à 1, soit généré dans un intervalle $[1; 20]$ ou $[1; 5]$ sont reportés dans le tableau 4.4. Pour l'algorithme FPTAS, nous pouvons constater une réduction du temps CPU assez proportionnelle à la diminution de la borne supérieure du coût unitaire de non-réalisation. En revanche, le temps de calcul pour les modèles CPLEX(Ent) et CPLEX(Bin) restent assez stables (la valeur moyenne du temps CPU change peu). Indiquons simplement qu'en passant le coût unitaire de $[1; 5]$ à une valeur fixe de 1 pour CPLEX(Bin), le temps CPU diminue pour les problèmes de 20, 60, 70, 110, 120, 140 et 150 lots, et croît pour les problèmes de 30, 90 et 130 lots. Pour le modèle CPLEX(Ent), le temps CPU reste le même jusqu'à 80 lots, et à partir de $n = 90$ lots le temps de calcul augmente quand le coût unitaire passe de $[1; 5]$ à 1.

TABLE 4.2 – Temps de calcul moyen de FPTAS, CPLEX(Ent) et CPLEX(Bin) avec différents intervalles de α

Nombre	***FPTAS , Alpha***		***CPLEX(Ent), Alpha***		***CPLEX(Bin), Alpha***	
de lots	$[0,001; 0,3]$	$[0,2; 0,5]$	$[0,001; 0,3]$	$[0,2; 0,5]$	$[0,001; 0,3]$	$[0,2; 0,5]$
10	0,07	0,14	0,02	0,02	0,04	0,05
20	0,42	0,48	0,02	0,03	0,06	0,07
30	1,30	1,65	0,03	0,04	0,09	0,11
40	**2,67**	**2,64**	0,03	0,04	0,12	0,14
50	**5,16**	**3,86**	0,04	0,05	0,15	0,17
60	6,60	7,70	0,04	0,06	0,16	0,21
70	10,27	11,76	0,05	0,07	0,19	0,24
80	15,83	20,38	0,05	0,07	0,22	0,28
90	**26,95**	**22,87**	0,06	0,08	0,25	0,34
100	51,00	56,04	0,06	0,09	0,28	0,36
110	**42,09**	**38,98**	0,07	0,10	0,31	0,39
120	60,86	71,95	0,08	0,12	0,32	0,46
130	69,43	105,01	0,08	0,13	0,37	0,47
140	**96,25**	**84,09**	0,09	0,16	0,42	0,54
150	98,13	119,97	0,10	0,17	0,44	0,59

Table 4.3 – Temps de calcul moyens pour FPTAS, CPLEX(Ent) et CPLEX(Bin) avec différents intervalles de la demande

Nombre de lots	***FPTAS*** ***Demande***			*Nombre de Lots*	***CPLEX(Ent)*** ***Demande***			*Nombre de lots*	***CPLEX(Bin)*** ***Demande***		
	[1; 100]	[1; 20]	[1; 5]		[1; 100]	[1; 20]	[1; 5]		[1; 100]	[1; 20]	[1; 5]
10	0,28	0,02	0,01	**10**	0,02	0,02	0,01	**10**	0,09	0,03	0,02
20	1,23	0,09	0,03	**20**	0,03	0,03	0,02	**20**	0,15	0,04	0,02
30	4,05	0,30	0,08	**30**	0,04	0,03	0,02	**30**	0,23	0,05	0,02
40	7,23	0,58	0,17	**40**	0,05	0,04	0,02	**40**	0,30	0,06	0,03
50	12,19	1,08	0,26	**50**	0,06	0,05	0,03	**50**	0,38	0,07	0,03
60	19,25	1,72	0,48	**60**	0,07	0,05	0,03	**60**	0,45	0,08	0,03
70	29,32	3,03	0,69	**70**	0,08	0,06	0,03	**70**	0,53	0,08	0,04
80	50,03	3,34	0,95	**80**	0,09	0,07	0,04	**80**	0,62	0,09	0,04
90	67,34	5,95	1,43	**90**	0,10	0,07	0,04	**90**	0,74	0,10	0,04
100	150,84	7,96	1,75	**100**	0,11	0,08	0,04	**100**	0,79	0,12	0,05
110	107,70	11,35	2,55	**110**	0,12	0,09	0,05	**110**	0,87	0,13	0,05
120	182,38	13,29	3,54	**120**	0,15	0,10	0,05	**120**	0,98	0,13	0,06
130	240,73	16,94	3,99	**130**	0,17	0,10	0,05	**130**	1,06	0,14	0,06
140	242,79	21,94	5,78	**140**	0,22	0,12	0,05	**140**	1,23	0,15	0,06
150	298,80	21,82	6,53	**150**	0,22	0,13	0,05	**150**	1,33	0,14	0,06

TABLE 4.4 – Temps de calcul moyens pour FPTAS (optimal), CPLEX(Ent) et CPLEX(Bin) avec différents intervalles de coût de retard

Nombre de lots	***FPTAS*** ***coût***			*Nombre de Lots*	***CPLEX(Ent)*** ***coût***			*Nombre de lots*	***CPLEX(Bin)*** ***coût***		
	$\{1\}$	$[1;5]$	$[1;20]$		$\{1\}$	$[1;5]$	$[1;20]$		$\{1\}$	$[1;5]$	$[1;20]$
10	0,02	0,03	0,26	**10**	0,02	0,02	0,02	**10**	0,04	0,04	0,05
20	0,12	0,19	1,04	**20**	0,02	0,03	0,03	**20**	0,06	0,07	0,07
30	0,33	0,60	3,50	**30**	0,03	0,03	0,04	**30**	0,10	0,09	0,11
40	0,77	1,46	5,74	**40**	0,04	0,04	0,04	**40**	0,12	0,12	0,14
50	1,43	2,43	9,68	**50**	0,04	0,04	0,05	**50**	0,15	0,15	0,18
60	2,46	3,37	15,61	**60**	0,05	0,05	0,05	**60**	0,17	0,19	0,20
70	3,39	5,96	23,69	**70**	0,05	0,05	0,06	**70**	0,20	0,22	0,22
80	4,72	8,93	40,67	**80**	0,06	0,06	0,07	**80**	0,24	0,24	0,28
90	8,13	12,59	54,01	**90**	0,07	0,06	0,08	**90**	0,29	0,28	0,32
100	11,34	15,77	133,45	**100**	0,08	0,07	0,09	**100**	0,30	0,30	0,35
110	14,25	22,52	84,82	**110**	0,09	0,08	0,09	**110**	0,31	0,35	0,39
120	19,35	29,29	150,56	**120**	0,10	0,09	0,10	**120**	0,36	0,41	0,39
130	22,84	31,70	207,12	**130**	0,11	0,10	0,11	**130**	0,41	0,40	0,45
140	23,59	44,07	202,84	**140**	0,14	0,11	0,13	**140**	0,44	0,47	0,53
150	36,78	56,06	234,30	**150**	0,15	0,12	0,13	**150**	0,48	0,50	0,55

Le dernier paramètre que nous avons fait varier est le "T-ratio", puisque si la valeur de T_1 est très proche de la valeur de T^*, nous pourrons fabriquer une grande partie des pièces demandées et donc le coût total de retard sera relativement petit. Nous avons constaté, que ce ratio n'influence pas (ou presque) le temps de calcul pour les modèles linéaires (voir le tableau 4.5). En revanche, pour le FPTAS, ce paramètre est critique.

Table 4.5 – Temps de calcul moyens pour FPTAS, CPLEX(Ent) et CPLEX(Bin) avec différents intervalles de T-ratio

Nombre de lots	***FPTAS*** ***T-ratio***		***CPLEX(Ent)*** ***T-ratio***		***CPLEX(Bin)*** ***T-ratio***	
	$[0,75; 0,9]$	$[0,9; 0,95]$	$[0,75; 0,9]$	$[0,9; 0,95]$	$[0,75; 0,9]$	$[0,9; 0,95]$
10	0,17	0,03	0,02	0,02	0,05	0,04
20	0,76	0,13	0,03	0,02	0,07	0,06
30	2,67	0,28	0,03	0,03	0,11	0,10
40	4,57	0,75	0,04	0,04	0,13	0,13
50	7,96	1,06	0,05	0,04	0,17	0,15
60	12,61	1,69	0,05	0,05	0,19	0,18
70	19,57	2,46	0,06	0,05	0,23	0,21
80	32,79	3,42	0,07	0,06	0,27	0,24
90	44,07	5,75	0,08	0,07	0,31	0,28
100	99,87	7,17	0,09	0,07	0,32	0,31
110	72,02	9,04	0,09	0,08	0,36	0,34
120	120,44	12,36	0,11	0,09	0,41	0,37
130	159,86	14,58	0,11	0,10	0,43	0,41
140	159,54	20,80	0,13	0,12	0,49	0,47
150	194,90	23,19	0,14	0,13	0,55	0,48

Enfin, dans le tableau 4.6 nous montrons les temps de calcul en fonction du nombre de lots pour le cas où la demande $d_i \in [1; 5]$, le coût unitaire dde non-réalisation c_i est égal à 1, les pourcentages de rebuts α_i sont dans $[0,001; 0,3]$ et le T-ratio est dans l'intervalle $[0,9; 0,95]$. Pour les instances de petites tailles, l'approche FPTAS arrive à trouver la solution exacte plus vite que CPLEX avec les modèles linéaires. Ce n'est plus le cas pour les exemples avec un nombre de lots plus important. Notons quand même, que les temps de calcul pour ces jeux de données sont comparables pour les trois méthodes.

Pour la fonction $f(x)$ (FD), qui représente le nombre de rebuts dans un lot de taille x, le modèle CPLEX(Ent) parait plus efficace du point de vue du temps CPU. Mais, comme nous allons le montrer dans la section suivante, les approches FPTAS et CPLEX(Bin) peuvent

TABLE 4.6 – Temps CPU pour FPTAS, CPLEX(Ent) et CPLEX(Bin) quand $d_i \in [1;5]$, $c_i = 1$, $\alpha_i \in [0,001; 0,3]$ et le T-ratio $\in [0,9; 0,95]$

Nombre de lots	*Temps CPU, sec*		
	CPLEX (Ent)	***CPLEX (Bin)***	***FPTAS***
10	0,012	0,014	0,000
20	0,018	0,016	0,010
30	0,016	0,026	0,010
40	0,018	0,026	0,022
50	0,010	0,030	0,032
60	0,020	0,032	0,054
70	0,020	0,038	0,056
80	0,022	0,036	0,096
90	0,026	0,042	0,118
100	0,030	0,044	0,148
110	0,030	0,052	0,316
120	0,030	0,058	0,394
130	0,030	0,054	0,412
140	0,030	0,058	0,518
150	0,036	0,058	0,806

être généralisées à des fonctions $f(x)$ plus générales.

4.2 Problème P-COÛT-TAILLES : ajustement des méthodes

Dans cette section nous avons pour but l'adaptation des méthodes FPTAS et CPLEX(Bin) au cas où $f(x)$ serait plus générale. Nous allons reprendre la fonction $f^1(x) = \lfloor \alpha x + \beta x^{\frac{1}{2}} \rfloor$, qui a été présentée dans la section 3.3, et montrer que le schéma, développé pour résoudre le problème P-COÛT-TAILLES-FD, peut être appliqué, après quelques modifications, pour le problème P-COÛT-TAILLES avec la fonction $f^1(x)$ représentant le nombre de rebuts. Nous allons également montrer, que le modèle linéaire en variables binaires peut également être facilement adapté pour la résolution de ce problème. Dans le reste de ce chapitre nous allons utiliser la notation $f^1(x)$ au lieu de $f(x)$.

Rappelons la formulation mathématique du problème P-COÛT-TAILLES :

Problème P-Coût-Tailles :

$$\begin{aligned} &\textit{Minimiser} \quad W(x) := \sum_{i=1}^{n} c_i \max\{0, d_i - (x_i - f_i(x_i))\} \\ &s.\ c. \quad Q(x) := \sum_{i=1}^{n} t_i x_i + T(x) \leq T_1, \quad x_i \in \mathbb{N},\ i = 1, \ldots, n \end{aligned} \tag{4.7}$$

Soit $f_i^1(x) = \lfloor \alpha_i x + \beta_i x^{\frac{1}{2}} \rfloor$ la fonction représentant le nombre de rebuts, où α_i et β_i, $i = 1, \ldots, n$ sont des coefficients réels non-négatifs et $T(x) = \sum_{i=1}^{n} \gamma_i t_i x_i,\ \ x = (x_1, \ldots, x_n)$, $0 \leq \gamma_i < 1,\ \ i = 1, \ldots, n$ la fonction représentant le temps total de réparation. En utilisant ces notations nous pouvons reformuler le problème ci-dessus de la manière suivante :

Problème P-Coût-Tailles-$\alpha\beta$:

$$\begin{aligned} &\textit{Minimiser} \quad W^{\alpha\beta}(x) := \sum_{i=1}^{n} c_i \max\left\{0, d_i - (x_i - \lfloor \alpha_i x_i + \beta_i x_i^{\frac{1}{2}} \rfloor)\right\} \\ &s.\ c. \quad Q^{\alpha\beta}(x) := \sum_{i=1}^{n} t_i x_i + \sum_{i=1}^{n} \gamma_i t_i x_i \leq T_1, \quad x_i \in \mathbb{N},\ i = 1, \ldots, n \end{aligned} \tag{4.8}$$

Nous nous référons ici à l'analyse faite dans la section 3.3 quand nous avons discuté les équations (3.6) - (3.10). En imposant la condition $f^1(x) \leq x,\ \ \forall x \geq 1$ sur la fonction f^1, nous obtenons l'inégalité qui doit être vérifiée pour chaque pair de variable α_i et β_i :

$$0 \leq \beta_i \leq 1 - \alpha_i < 1, \quad i = 1, \ldots, n \tag{4.9}$$

Les tailles de lots x_i^* minimum nécessaires, pour satisfaire la demande, peuvent être trouvées comme suit :

$$x_i^* = \begin{cases} x_i^+ + 1, & \text{si } x_i^+ \text{ est entier} \\ \lceil x_i^+ \rceil, & \text{sinon} \end{cases} \quad i = 1, \ldots, n \tag{4.10}$$

où

$$x_i^+ = \frac{\beta_i^2 + \beta_i \sqrt{4(d_i - 1)(1 - \alpha_i) + \beta_i^2} + 2(d_i - 1)(1 - \alpha_i)}{2(1 - \alpha_i)^2} \tag{4.11}$$

Dans la section 3.4, nous avons démontré (voir le théorème 3.2) que le problème PCT-FD est NP-difficile, même si $\alpha_i = 1/3$, la demande est la même et fixée pour tous les produits ($d_i = 2$) et $\gamma = 0$ (c.-à-d. la machine ne tombe pas en panne). Le problème PCT-FD est

un cas particulier du problème P-Coût-Tailles-$\alpha\beta$ avec $\beta = 0$, alors la démonstration de NP-difficulté de la section 3.4 est également valable pour P-Coût-Tailles-$\alpha\beta$ (PCT-$\alpha\beta$).

4.2.1 FPTAS pour le problème P-Coût-Tailles-$\alpha\beta$

Un vecteur x est une solution réalisable pour le problème PCT-$\alpha\beta$ si $Q^{\alpha\beta}(x) \leq T_1$. Par conséquent, si $Q^{\alpha\beta}(x^{(1)}) > T_1$ pour le vecteur unitaire $x^{(1)} = (1, \ldots, 1)$, alors il n'existe aucune solution réalisable pour ce problème. Sans perte de généralité, nous supposons donc que $Q^{\alpha\beta}(x^{(1)}) \leq T_1$. Soit h_i la taille minimale, telle que la demande pour les pièces de bonne qualité soit satisfaite, alors $h_i = x_i^*$ (voir l'équation (4.10)). Si $Q^{\alpha\beta}(x^{(h)}) \leq T_1$, pour $x^{(h)} = (h_1, h_2, \ldots, h_n)$, alors la solution optimale est trouvée (avec le coût total d'insatisfaction de la demande $W^{\alpha\beta}(x^{(h)}) = 0$). Supposons que ce n'est pas le cas.

Pour déterminer les bornes supérieure et inférieure pour le coût de retard, nous utilisons la même approche, que dans la section 3.4 pour le problème PCT-FD. La borne inférieure est $BI := \min_{1 \leq i \leq n}\{c_i\} \geq 1$. Pour calculer une borne supérieure BS, il faut appliquer l'algorithme 3.2.

Supposons que les bornes BI et BS ont déjà été calculées. Alors, le problème P-ROU pour le problème PCT-$\alpha\beta$ peut être formulé de la manière suivante :

Problème P-Rou :

$$\textit{Minimiser} \;\; w^{\alpha\beta} := \sum_{i=1}^{n} \left\lfloor \frac{v_i(x_i)}{\delta} \right\rfloor \tag{4.12}$$

$$s.\ c. \quad q^{\alpha\beta}(x) \leq T_1, \tag{4.13}$$

$$\text{et } x_i \in \{r_i(0), r_i(1), \ldots, r_i(\left\lfloor \frac{BS}{\delta} \right\rfloor)\},\ i = 1, \ldots, n \tag{4.14}$$

où

$$v_i(x_i) = c_i \max\{0, d_i - (x_i - \lfloor \alpha_i x + \beta_i x^{\frac{1}{2}} \rfloor)\}$$

et

$$r_i(l) = \min\{x \mid x \in \mathbb{N}, \left\lfloor \frac{v_i(x)}{\delta} \right\rfloor \leq l\},\ l = 0, 1, \ldots, \left\lfloor \frac{BS}{\delta} \right\rfloor$$

Les transformations et les équivalences suivantes sont vérifiées :

$$\left\lfloor \frac{v_i(x)}{\delta} \right\rfloor \leq l \Longleftrightarrow$$
$$\left\lfloor \frac{c_i \max\{0, d_i - (x_i - \lfloor \alpha_i x + \beta_i x^{\frac{1}{2}} \rfloor)\}}{\delta} \right\rfloor \leq l \Longleftrightarrow$$

$$\max\{0, d_i - (x_i - \lfloor \alpha_i x + \beta_i x^{\frac{1}{2}} \rfloor)\} < \frac{(l+1)\delta}{c_i} \Longleftrightarrow$$
$$d_i - x_i + \lfloor \alpha_i x + \beta_i x^{\frac{1}{2}} \rfloor < \frac{(l+1)\delta}{c_i} \Longleftrightarrow$$
$$d_i - x_i + \lfloor \alpha_i x + \beta_i x^{\frac{1}{2}} \rfloor < \left\lceil \frac{(l+1)\delta}{c_i} \right\rceil \Longleftrightarrow$$
$$\alpha_i x + \beta_i x^{\frac{1}{2}} < x_i - d_i + \left\lceil \frac{(l+1)\delta}{c_i} \right\rceil \Longleftrightarrow$$
$$(1-\alpha_i)x_i - \beta_i x_i^{\frac{1}{2}} + \left\lceil \frac{(l+1)\delta}{c_i} \right\rceil - d_i > 0 \tag{4.15}$$

Soit $y_i(x_i) = (1-\alpha_i)x_i - \beta_i x_i^{\frac{1}{2}} + \left\lceil \frac{(l+1)\delta}{c_i} \right\rceil - d_i$. Pour notre problème, $x_i \in \mathbb{N},\ i = 1, \ldots, n$, alors ces fonctions $y_i(x_i),\ i = 1, \ldots, n$ sont convexes $\forall x_i \in \mathbb{N}$ car :

$$y_i^{'}(x_i) = (1-\alpha_i) - \beta \frac{1}{2x^{\frac{1}{2}}}$$
$$y_i^{"}(x_i) = \beta \frac{1}{4x^{\frac{3}{2}}}$$

De plus, comme $y_i^{"}(x_i) > 0$, le minimum est dans le point $x_i^{i_{min}}$, qui est calculé comme suit :

$$y_i^{'}(x_i) = (1-\alpha_i) - \beta \frac{1}{2x^{\frac{1}{2}}} = 0 \Rightarrow$$
$$x_i^{i_{min}} = \frac{\beta_i^2}{4(1-\alpha_i)^2} < 1$$

Alors chaque solution x_i d'inégalité (4.15) doit être supérieure à la solution positive $x_i^{>}$ de l'équation $y_i(x_i) = 0$ avec les racines :

$$x_i^{\frac{1}{2}>}, x_i^{\frac{1}{2}<} = \frac{\beta_i \pm \sqrt{\beta^2 - 4(1-\alpha_i)(\left\lceil \frac{(j+1)\delta}{c_i} \right\rceil - d_i)}}{2(1-\alpha_i)} \tag{4.16}$$

d'où nous obtenons la valeur recherchée :

$$x_i^{>} = h_i^l = \frac{\beta_i^2 - 2(1-\alpha_i)(\left\lceil \frac{(j+1)\delta}{c_i} \right\rceil - d_i) + \beta_i \sqrt{\beta_i^2 - 4(1-\alpha_i)(\left\lceil \frac{(j+1)\delta}{c_i} \right\rceil - d_i)}}{2(1-\alpha_i)^2} \tag{4.17}$$

Alors

$$r_i(l) = \begin{cases} \max\{1, h_i^j + 1\}\}, & \text{si } h_i^j \text{ est un entier} \\ \max\{1, \lceil h_i^j \rceil\}\}, & \text{sinon} \end{cases}$$

$$l = 0, 1, \ldots, \left\lfloor \frac{BS}{\delta} \right\rfloor, \quad i = 1, \ldots, n$$

Donc, nous avons obtenu tout ce qu'il faut pour appliquer l'algorithme FPTAS, ainsi que la procedure BIP de la section 3.4, au problème P-Coût-Tailles-$\alpha\beta$.

4.2.2 Modèle linéaire pour le problème P-Coût-Tailles-$\alpha\beta$

A partir du modèle CPLEX(Bin), présentée dans la section 4.1.2 nous pouvons obtenir un modèle linéaire pour le problème P-Coût-Tailles-$\alpha\beta$. Il suffit de remplacer la fonction de rebut dans le critère.

Modèle Linéaire - $\alpha\beta$:

Fonction objectif :

$$\textit{Minimiser} \quad \sum_{i=1}^{n} \sum_{j=1}^{x_i^*} c_i \max\left\{0, d_i - \left(j - \left\lfloor \alpha_i j + \beta_i \sqrt{j} \right\rfloor\right)\right\} \lambda_{ij} \tag{4.18}$$

Contraintes :

$$\sum_{i=1}^{n} \sum_{j=1}^{x_i^*} t_i \left(1 + \gamma_i\right) j \lambda_{ij} \leq T_1$$

$$\sum_{j=1}^{x_i^*} \lambda_{ij} = 1, \quad \forall i \in \{1, \ldots, n\}$$

Types des variables :

$$\lambda_{ij} \in \{0, 1\}, \quad \forall i \in \{1, \ldots, n\}, \quad j \in \{1, \ldots, x_i^*\}$$

Dans ce qui suit, nous allons nommer ce modèle CPLEX($\alpha\beta$).

4.2.3 Expérimentations numériques pour le problème PCT-$\alpha\beta$

Nous avons utilisé les mêmes plages de données pour les paramètres que dans le tableau 3.1, à l'exception de β_i, $i = 1, \ldots, n$ qui ont été générés dans l'intervalle $\beta_i \in [0, 1 - \alpha_i]$, $i = 1, \ldots, n$. De la même manière, que dans la section 3.6, nous avons généré 10 instances pour chacune de 36 familles de données.

Table 4.7 – Temps de calcul moyens de CPLEX($\alpha\beta$), FPTAS et BIP

Nombre de lots	*Algorithme*		
	CPLEX *($\alpha\beta$)*	***FPTAS***	***BIP***
10	0,05	0,11	0,01
20	0,07	0,45	0,06
30	0,10	1,49	0,18
40	0,13	3,22	0,40
50	0,16	6,23	0,71
60	0,19	9,70	1,23
70	0,23	10,91	1,90
80	0,27	17,63	2,75
90	0,30	33,92	3,92
100	0,33	34,24	5,26
110	0,36	56,57	7,10
120	0,40	55,12	9,26
130	0,46	67,79	11,69
140	0,47	90,60	14,55
150	0,51	117,51	17,46

Les temps de calcul de CPLEX($\alpha\beta$), de FPTAS en mode exact, et de la procédure BIP sont présentés dans le tableau 4.7. Nous pouvons constater, que les temps CPU de FPTAS et de BIP sont très proches des temps CPU correspondants pour le problème PCT-FD. Nous pouvons conclure la même chose à propos de CPLEX($\alpha\beta$).

Pour FPTAS approché, la tendance est la même que pour PCT-FD. Dans le tableau C.1 (dans l'annexe) nous présentons les temps de calcul de FPTAS pour $\varepsilon \in \{0;\ 0,01;\ 0,05;\ 0,1;\ 0,3;\ 0,5;\ 0,7;\ 1\}$. La dernière ligne du tableau donne la valeur moyenne pour toutes les instances. Notons que dans nos expériences, cette valeur est la même pour $\varepsilon = 0$ et $\varepsilon = 0,01$, car ici avec $\varepsilon = 0,01$ nous obtenons les solutions exactes pour toutes les instances. Conformément au tableau C.2 (dans l'annexe), même pour $\varepsilon = 0,5$, l'erreur moyenne reste tout à fait raisonnable (moins de 3%) avec le temps de calcul moyen relativement petit (8, 17

TABLE 4.8 – Temps moyen de FPTAS et de CPLEX($\alpha\beta$) pour différents intervalles de α

Nombre de lots	***FPTAS*** ***Alpha***		***CPLEX($\alpha\beta$)*** ***Alpha***	
	$[0,001; 0,3]$	$[0,2; 0,5]$	$[0,001; 0,3]$	$[0,2; 0,5]$
10	0,133	0,087	0,042	0,049
20	0,345	0,561	0,064	0,076
30	1,429	1,548	0,088	0,114
40	3,016	3,429	0,119	0,144
50	4,516	7,948	0,144	0,175
60	6,942	12,466	0,170	0,218
70	10,623	11,197	0,191	0,259
80	16,618	18,646	0,223	0,321
90	33,832	33,999	0,257	0,342
100	25,324	43,164	0,284	0,374
110	67,480	45,670	0,308	0,417
120	64,177	46,069	0,347	0,455
130	72,960	62,628	0,381	0,535
140	89,027	92,171	0,426	0,512
150	130,542	104,468	0,404	0,615
Moyenne	35,131	32,270	0,230	0,307

secondes) si on le compare au temps de calcul pour la recherche des solutions exactes (33,7 secondes).

Dans le tableau 4.8, nous présentons les résultats numériques pour le cas où nous faisons varier la valeur de α. Rappelons que pour le cas particulier du problème considéré dans cette section, la quantité de pièces défectueuses dans un lot dépend aussi du paramètre β. Nous pouvons constater que quand on passe de $[0,001; 0,3]$ à $[0,2; 0,5]$, les deux méthodes analysées se comportent différemment : si le temps CPU de CPLEX($\alpha\beta$) augmente dans tous les cas, le temps moyen CPU de FPTAS diminue pour les tailles de problème 10, 110, 120, 130, 150. Notons que, le temps total moyen est plus faible pour les valeurs de α dans l'intervalle $[0,2; 0,5]$, que quand elles sont dans l'intervalle $[0,001; 0,3]$.

Le tableau 4.9 montre l'évolution des temps de calcul des deux méthodes en fonction de la demande (tous les autres paramètres sont fixés). De la même façon que pour le problème P-COÛT-TAILLES-FD, la diminution des temps de calcul est importante pour les deux méthodes, si on passe d'une demande générée dans l'intervalle $[1; 100]$ à une demande dans intervalle $[1; 5]$.

Les temps de calcul moyens pour le cas où nous varions les plages de données pour le

Table 4.9 – Temps de calcul moyen de FPTAS et CPLEX($\alpha\beta$) pour différents intervalles de la demande

Nombre de lots	***FPTAS, Demande***			*Nombre de lots*	***CPLEX($\alpha\beta$), Demande***		
	$[1;100]$	$[1;20]$	$[1;5]$		$[1;100]$	$[1;20]$	$[1;5]$
10	0,303	0,021	0,005	**10**	0,089	0,029	0,020
20	1,239	0,095	0,026	**20**	0,152	0,038	0,021
30	4,132	0,267	0,066	**30**	0,229	0,050	0,024
40	8,809	0,683	0,174	**40**	0,303	0,061	0,029
50	17,290	1,148	0,257	**50**	0,380	0,068	0,031
60	26,821	1,862	0,428	**60**	0,472	0,079	0,032
70	29,326	2,772	0,632	**70**	0,556	0,085	0,034
80	47,844	4,195	0,857	**80**	0,681	0,097	0,038
90	94,619	5,853	1,275	**90**	0,749	0,108	0,041
100	92,934	8,094	1,704	**100**	0,820	0,123	0,044
110	156,032	11,529	2,164	**110**	0,916	0,125	0,047
120	148,211	14,125	3,034	**120**	1,009	0,142	0,052
130	182,703	16,578	4,102	**130**	1,178	0,144	0,052
140	247,230	19,809	4,758	**140**	1,208	0,150	0,049
150	320,344	26,526	5,645	**150**	1,333	0,147	0,049
Moyenne	91,856	7,570	1,675	**Moyenne**	0,672	0,096	0,038

Table 4.10 – Temps de calcul moyen de FPTAS et CPLEX($\alpha\beta$) pour différents intervalles du coût de retard

Nombre de lots	***FPTAS, Coût***			*Nombre de lots*	***CPLEX($\alpha\beta$), Coût***		
	$\{1\}$	$[1;5]$	$[1;20]$		$\{1\}$	$[1;5]$	$[1;20]$
10	0,018	0,032	0,280	**10**	0,045	0,045	0,048
20	0,113	0,172	1,075	**20**	0,068	0,068	0,075
30	0,357	0,521	3,588	**30**	0,097	0,100	0,106
40	0,671	1,338	7,657	**40**	0,118	0,127	0,148
50	1,523	2,605	14,567	**50**	0,139	0,154	0,185
60	2,567	4,795	21,749	**60**	0,181	0,183	0,219
70	3,195	6,106	23,428	**70**	0,218	0,214	0,243
80	5,641	9,194	38,061	**80**	0,268	0,251	0,296
90	7,843	10,850	83,054	**90**	0,268	0,304	0,326
100	9,739	19,489	73,503	**100**	0,309	0,327	0,351
110	13,340	22,777	133,607	**110**	0,355	0,348	0,384
120	18,753	27,441	119,174	**120**	0,386	0,397	0,420
130	18,748	31,675	152,959	**130**	0,439	0,431	0,505
140	24,481	44,140	203,176	**140**	0,446	0,470	0,491
150	33,519	65,441	253,555	**150**	0,466	0,504	0,559
Moyenne	9,367	16,438	75,295	**Moyenne**	0,253	0,262	0,290

TABLE 4.11 – Temps de calcul moyen de FPTAS et CPLEX($\alpha\beta$) pour différents intervalles de T-ratio

Nombre de lots	***FPTAS, Ratio***		***CPLEX($\alpha\beta$), Ratio***	
	$[0,75;0,9]$	$[0,9;0,95]$	$[0,75;0,9]$	$[0,9;0,95]$
10	0,194	0,025	0,046	0,046
20	0,812	0,094	0,071	0,070
30	2,656	0,321	0,103	0,099
40	5,844	0,600	0,138	0,124
50	11,390	1,074	0,168	0,151
60	17,610	1,798	0,202	0,187
70	19,290	2,530	0,237	0,213
80	31,287	3,977	0,282	0,262
90	62,349	5,483	0,319	0,279
100	61,246	7,242	0,338	0,320
110	104,035	9,115	0,380	0,345
120	97,129	13,117	0,409	0,393
130	119,199	16,389	0,479	0,437
140	160,261	20,937	0,493	0,444
150	211,832	23,178	0,546	0,473
Moyenne	60,342	7,059	0,281	0,256

coût d'insatisfaction unitaire, sont présentés dans le Tableau 4.10. En regardant les valeurs moyennes (dernière ligne) nous pouvons constater, que le temps CPU pour la méthode CPLEX(α, β) change peu en fonction de coût de retard.

Dans le tableau 4.11, l'intervalle de T-ratio varie entre $[0,75;0,9]$ et $[0,9;0,95]$. Plus T-ratio est proche de 1, plus le temps de calcul pour FPTAS est petit.

Enfin, dans le tableau 4.12 nous présentons les temps CPU pour le cas quand la demande $d_i \in [1;5]$, le coût unitaire d'insatisfaction c_i est égal à 1 pour tous les types de produit, le T-ratio est généré dans l'intervalle $[0,9;0,95]$. Dans la partie gauche du tableau, les résultats sont pour $\alpha \in [0,001;0,3]$, dans la partie droite pour $\alpha \in [0,2;0,5]$. Si nous regardons la valeur moyenne (la dernière ligne du tableau), l'approche FPTAS est plus rapide quand $\alpha \in [0,001;0,3]$. En même temps, pour plus de la moitié des instances (tailles de problème de 10 à 90) le temps CPU pour $\alpha \in [0,2;0,5]$ est plus faible. Pour les instances avec α dans l'intervalle $[0,001;0,3]$, FPTAS est en moyenne plus rapide pour $n \in [10;50]$ et pour les instances avec $\alpha \in [0,2;0,5]$ pour $n = [10;60]$.

TABLE 4.12 – Temps CPU de CPLEX($\alpha\beta$) et de FPTAS pour $d_i \in [1; 5]$, $c_i = 1$ et T-ratio $\in [0,9; 0,95]$

Nombre de lots	$\alpha \in [0,001; 0,3]$		*Nombre de lots*	$\alpha \in [0,2; 0,5]$	
	CPLEX *($\alpha\beta$)*	***FPTAS***		***CPLEX*** *($\alpha\beta$)*	***FPTAS***
10	0,020	0,000	**10**	0,020	0,000
20	0,020	0,010	**20**	0,021	0,010
30	0,021	0,010	**30**	0,024	0,010
40	0,027	0,016	**40**	0,026	0,014
50	0,028	0,026	**50**	0,030	0,019
60	0,031	0,037	**60**	0,032	0,024
70	0,030	0,060	**70**	0,034	0,042
80	0,037	0,084	**80**	0,037	0,077
90	0,040	0,111	**90**	0,042	0,105
100	0,040	0,141	**100**	0,043	0,173
110	0,044	0,169	**110**	0,044	0,176
120	0,044	0,247	**120**	0,052	0,324
130	0,047	0,319	**130**	0,051	0,226
140	0,048	0,332	**140**	0,044	0,312
150	0,042	0,383	**150**	0,048	0,709
Moyenne	0,035	0,130	**Moyenne**	0,037	0,148

Conclusion

Dans ce chapitre nous avons présenté deux modèles linéaires, CPLEX(Ent) et CPLEX(Bin), pour le problème P-COÛT-TAILLES-FD du chapitre précédent, où CPLEX(Ent) est un modèle en variables entières et CPLEX(Bin) est un modèle en variables binaires. Nous avons comparé les performances de ces deux modèles entre eux et avec l'algorithme FPTAS, présenté précédemment. Nous avons constaté que pour la fonction $f(x)$ représentant le nombre de rebuts dans un lot de taille x de type "fraction défectueuse", le modèle CPLEX(Ent) est généralement plus efficace du point de vue du temps CPU. Mais pour certaines familles d'instances, l'approche FPTAS est arrivé à trouver une solution optimale plus vite. Ensuite nous avons montré comment transformer les approches FPTAS et CPLEX(Bin) dans le cas où les fonctions $f(x)$ sont plus générales. Nous avons terminé ce chapitre par une comparaison des deux méthodes.

Chapitre 5

Modèle Probabiliste : décomposition et algorithmes approchés

Dans ce chapitre nous examinons notre problème de lotissement et de séquencement lorsque les aléas sont représentés sous forme de loi de distribution (problème P-Prob*). L'objectif est de trouver la séquence et les tailles de lots maximisant la probabilité de satisfaction de la demande pour tous les lots sous la contrainte d'un horizon de planification donné. Pour résoudre ce problème NP-difficile, nous utilisons une méthode par décomposition, permettant de traiter séparément les trois sous-problèmes : d'énumération pour trouver quel produit sera fabriqué en dernier, de séquencement pour définir la séquence des autres lots et de lotissement pour définir les tailles de lot. Comme nous l'avons démontré dans le chapitre 2, le problème de séquencement est équivalent au problème de Voyageur du Commerce Asymétrique et peut être résolu par les approches correspondantes. Dans ce chapitre, nous mettons donc l'accent sur la résolution de la partie lotissement du problème initial* P-Prob*, et pour le résoudre nous proposons deux méthodes approchées : Recherche Locale et Algorithme Génétique. La comparaison de leurs performances, ainsi qu'une comparaison avec une méthode exacte proposée par Dolgui* et al. *(2005) est présentée à la fin de ce chapitre.*

Nous rappelons que la ligne de production est de type *flow-shop* contenant m machines en séquence. Nous allons garder tous les paramètres définis dans le chapitre 2. Nous supposons que *l'inégalité triangulaire* est satisfaite pour les temps de set-up, c.-à-d. pour chaque i, j et

$k = 1, \ldots, n$ l'inégalité suivante est vérifiée :

$$s_{i,j} + s_{j,k} \geq s_{i,k} \tag{5.1}$$

Tenant compte de ce fait, nous pouvons conclure que tous les composants de même type doivent être fabriqués en un seul lot (voir les chapitres 2 et 3 pour les démonstrations).

Dans ce chapitre, l'objectif du problème est de maximiser la probabilité d'avoir la quantité voulue de pièces de bonne qualité pour tous les produits.

Comme nous l'avons déjà expliqué dans le chapitre 2, ce type de problème se pose dans des ateliers de production électroniques, où une ligne de fabrication traite des produits de plusieurs types pour la chaîne d'assemblage qui suit. La ligne étudiée doit approvisionner le système d'assemblage en *juste-à-temps* (JAT), c.-à-d. tous les articles qui seront utilisés pour le montage durant la période $r + 1$ doivent être livrés d'ici la fin de la période en cours r. Au début de la période r, nous connaissons le niveau de la demande d_i pour tous les types de produits nécessaires pour effectuer l'assemblage de la période $r + 1$. Donc à chaque période r il faut prendre la décision de la quantité de composants de chaque type à fabriquer pendant la période r pour pouvoir assembler toutes les pièces demandées à la période $r + 1$. Le problème considéré doit donc être résolu au début de chaque période de fabrication, le modèle utilisé est un modèle *mono-période*. Dans ce type de production, il y a un pourcentage de rebuts non négligeable, c.-à-d. qu'il est possible que la qualité de certains produits soit non satisfaisante. Le contrôle de qualité est effectué après la dernière opération de la ligne (aucun contrôle de qualité intermédiaire), en outre, les machines sont souvent arrêtées en raison de pannes, il faut donc tenir compte des temps de réparation.

Pour atteindre l'objectif pour une telle ligne, nous devons optimiser :

- La séquence de lots $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$: le temps total d'exécution dépend essentiellement de la présence des temps de set-up, dépendant eux-mêmes de la séquence.
- Les tailles de lots $x = (x_1, x_2, \ldots, x_n)$: à cause des rebuts et des pannes machines, il est difficile de trouver les quantités de pièces exactes à lancer en fabrication pour satisfaire la demande. L'augmentation du nombre des pièces à traiter pour un produit quelconque augmente la probabilité d'obtenir la quantité de pièces désirée pour ce produit, mais en même temps, le temps restant pour la production des autres produits et les réparations diminue, ce qui peut amener à une diminution de la probabilité totale.

Soit x_i^b - la quantité de pièces de bonne qualité pour le produit i, $i = 1, \ldots, n$ avant la fin de l'horizon de planification, obtenue en lançant en fabrication x_i pièces.

En utilisant les notations ci-dessus, on peut formuler notre problème comme suit :

Problème P-Prob :

$$\textit{Maximiser} \quad P\left(x_i^b \geq d_i, \quad i = 1, \ldots, n | \, (\pi, x)\right) \tag{5.2}$$

sous la contrainte que l'horizon de planification soit inférieur ou égal à T_0.

5.1 Modélisation Mathématique de P-Prob

5.1.1 Notations

Dans ce chapitre nous allons utiliser les notations suivantes :

- nombre de types de produits à fabriquer n ;
- nombre de machines sur la ligne de production m ;
- horizon de planification T_0 ;
- demande d_i pour le produit i, $i = 1, \ldots, n$;
- temps de traitement t_i d'une pièce de type i sur n'importe quelle machine q de la ligne, $q = 1, \ldots, m$ (nous simplifions ici la réalité où t_{iq} peuvent être différents, cette hypothèse n'a rien de restrictif, elle est utilisée uniquement pour simplifier la présentation) ;
- temps de préparation $s_{0,i} \geq 0$, $i = 1, \ldots, n$, si la fabrication commence par une pièce de type i ;
- temps de set-up $s_{i,j}$ entre un lot de produit i et un lot de produit j (rappelons qu'un lot peut être constitué d'une seule pièce), $i, j = 1, \ldots, n$;
- probabilité p_{iq} qu'une pièce donnée de produit de type i soit de bonne qualité étant traitée par la machine q, où $i = 1, \ldots, n$ et $q = 1, \ldots, m$;
- taux de panne $1/u_q$ pour la machine q, $q = 1, \ldots, m$;
- taux de réparation $1/r_q$ de la machine q, $q = 1, \ldots, m$.

Donnons quelques explications complémentaires des notations utilisées. Prenons l'exemple d'un lot (voir la figure 5.1), et sans perte de généralité supposons que le lot i est le premier à être fabriquer. Le temps total de production pour ce lot contient :

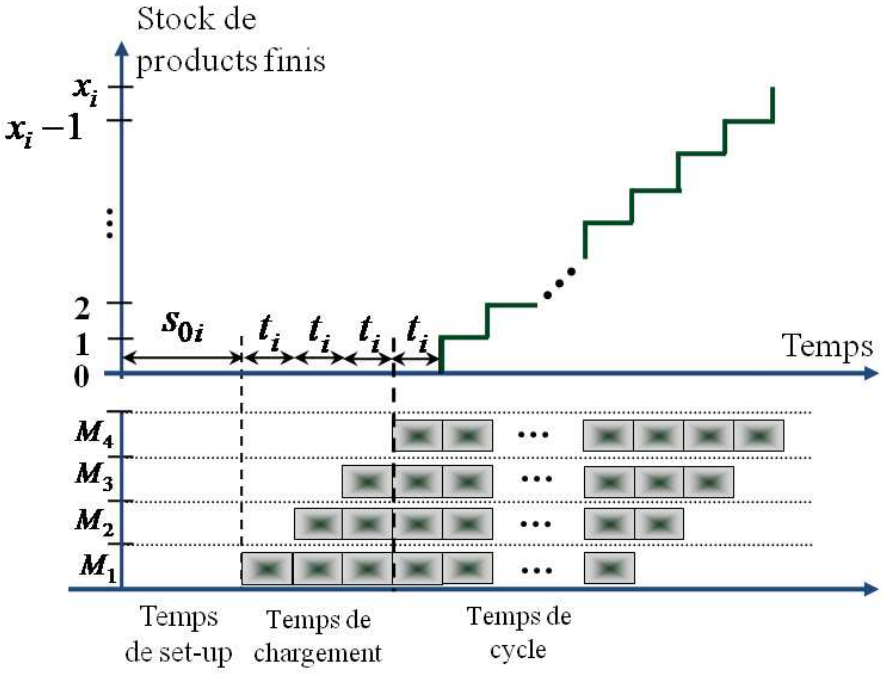


FIGURE 5.1 – Un exemple de traitement d'un lot sur une ligne de 4 machines

- *le temps de préparation* s_{0i} pour effectuer tous les réglages nécessaires pour la production des pièces de type i. Pour le lot j qui suit, ce sera le temps de set-up s_{ij}, $j \neq i$;
- *le temps de chargement* : au moment s_{0i}, la première pièce du lot i entre sur la première machine M_1. Au moment t_i, elle passe sur la machine suivante M_2 ; la deuxième pièce du lot (s'il y en a) entre alors sur la machine M_1, et ainsi de suite. La première pièce du lot i quittera la ligne à $s_{0,i} + mt_i$, la deuxième pièce sera finie à $s_{0,i} + mt_i + t_i$, et ainsi de suite. Nous appelons *le temps de chargement* la durée entre le début du traitement d'un lot et le moment où la première pièce entre sur la dernière machine, c.-à-d. pour le lot i il est égal à $(m-1)t_i$, pour n lots le temps total de chargement est $(m-1)\sum_{i=1}^{n} t_i$. Nous supposons que les réglages (set-up) ne peuvent pas être effectués pendant le temps de chargement, c.-à-d. toutes les machines de la ligne doivent être prêtes pour la fabrication avant que le processus de chargement ne commence.
- *le temps du cycle* : s'il n'y a pas de rebuts ni de pannes machines, le temps de production des x_i pièces de lot i est égal à $x_i t_i$. Nous supposons que les pannes ne peuvent arriver ni pendant le temps de set-up ni pendant le temps de chargement.

Un exemple de fabrication de lots avec des rebuts et des pannes est présenté dans la figure 5.2. Comme nous pouvons le voir, si une panne arrive, la production s'arrête pour

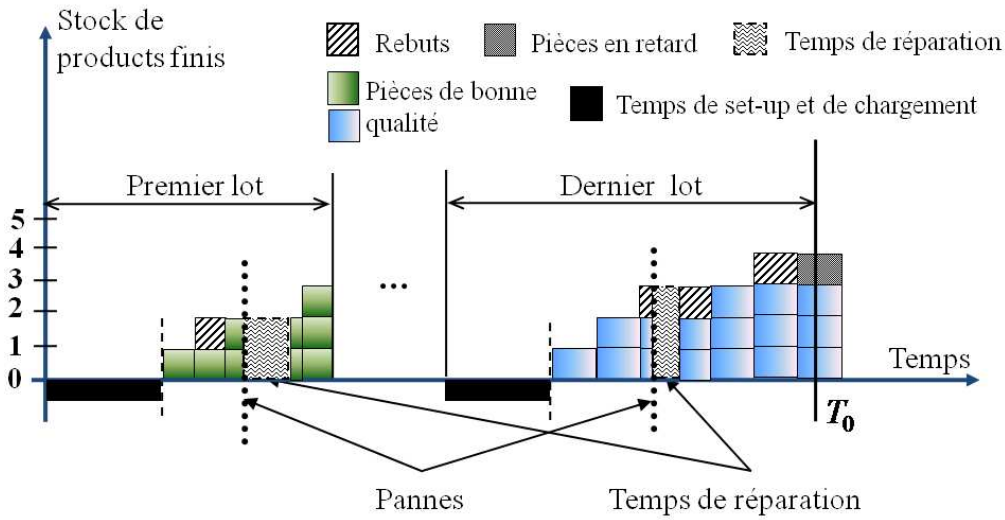


FIGURE 5.2 – Une illustration avec des temps de pannes et de rebuts

la durée de réparation. Soit T_s le *temps de sécurité prévu* (c.-à-d. le temps réservé à toutes les réparations des machines pendant la période $[0, T_0]$). Notons que si le temps total de réparation est supérieur au temps de sécurité prévu, un certain nombre de pièces du dernier lot (ou le dernier lot entièrement) ne vont pas être fabriqués. Bien sûr, cette remarque est également valable pour l'avant dernier lot et ainsi de suite, mais nous pouvons nous limiter qu'au dernier lot, car si aucune pièce de ce lot n'est fabriquée (la condition nécessaire pour regarder les autres lots) la probabilité que nous maximisons est nulle (quel que soit l'état des autres lots).

Dans ce chapitre, nous supposons que les pièces défectueuses ne peuvent pas être réparées, elles doivent donc être rejetées. Nous posons également l'hypothèse que les pannes et les rebuts sont indépendants.

La valeur du temps de sécurité T_s pour une solution (π, x) donnée peut être calculée comme suit :

$$T_s(\pi, x) = T_0 - \left(s_{0,\pi_1} + \sum_{j=2}^{n} s_{\pi_{j-1},\pi_j} + (m-1) \sum_{j=1}^{n} t_{\pi_j} + \sum_{j=1}^{n} t_{\pi_j} x_{\pi_j} \right) \tag{5.3}$$

où $s_{0,\pi_1} + \sum_{j=2}^{n} s_{\pi_{j-1},\pi_j}$ est le temps total de set-up, $(m-1)\sum_{j=1}^{n} t_{\pi_j}$ est le temps total de chargement et $\sum_{j=1}^{n} t_{\pi_j} x_{\pi_j}$ est le temps total de traitement des pièces. En augmentant la taille d'un lot x_i pour n'importe quel i, $\quad i = 1, \ldots, n$, la probabilité $P\left(x_i^b \geq d_i \mid x_i\right)$ de satisfaire la demande pour ce type de produit évidement va augmenter, mais le temps de sécurité prévu diminuera. Si le temps de sécurité T_s est insuffisant pour l'achèvement de toutes les réparations, la fabrication du dernier lot ne sera pas finie pour la fin de la période

de planification T_0 (voir la figure 5.2). Le temps de set-up non optimisé peut augmenter considérablement le temps total de production, et cela réduit le temps de sécurité T_s prévu et peut également provoquer le manque de temps pour la fabrication de toutes les pièces planifiées.

5.1.2 Modélisation des aléas

Pour le problème considéré, le critère retenu est le niveau de service global (probabilité d'obtenir les quantités nécessaires de pièces pour tous les types de produit). Comme dans les chapitres précédents, nous prenons en compte deux types d'incertitude à la fois : les rebuts et les pannes, c.-à-d. nous avons un *rendement aléatoire* et un *temps de travail effectif aléatoire*. Pour évaluer le niveau de service pour chaque solution tout en considérant ces deux facteurs aléatoires, nous allons utiliser les méthodes de modélisation des incertitudes expliquées ci-dessous.

Rendement aléatoire

Comme il a été mentionné ci-dessus, il n'y a pas de contrôle de qualité intermédiaire, alors nous ne pouvons savoir si la qualité d'une pièce est bonne ou non qu'après son traitement sur la dernière machine. La probabilité p_i qu'une pièce finie de type i soit de bonne qualité peut donc être calculée comme suit :

$$p_i = \prod_{q=1}^{m} p_{iq}, \quad i = 1, \ldots, n \tag{5.4}$$

où p_{iq} est la probabilité qu'une pièce donnée de type i, traitée sur la machine q est de bonne qualité.

Nous supposons également que la qualité d'une pièce t donnée est indépendante de la qualité d'autres pièces du même type. Nous pouvons donc utiliser le processus de *Bernoulli*, qui est une séquence finie (notre cas) ou infinie d'événements aléatoires indépendants $\gamma_1, \gamma_2, \ldots$, où chaque γ_j est égal soit à 1 (succès) ou à 0 (échec). La probabilité $P(\gamma_j = 1)$ d'un succès est connue et elle est la même pour tous les γ_j, $j = 1, 2, \ldots$ Si l'événement γ_j^i correspond à la qualité de la j-ème pièce de produit i, $i = 1, \ldots, n$, alors $P(\gamma_j^i = 1) = p_i$ pour chaque $i = 1, \ldots, n$ et $j = 1, 2, \ldots$ En utilisant la *loi de probabilité* binomiale nous pouvons calculer la probabilité d'obtenir exactement la quantité requise de pièces de bonne qualité pour

chaque lot i sachant la taille x_i du lot lancé :

$$f(d_i, x_i, p_i) = P(x_i^b = d_i \mid x_i) = \binom{x_i}{d_i} p_i^{d_i}(1-p_i)^{x_i-d_i} \tag{5.5}$$

où x_i^b est le nombre de succès et $\binom{x_i}{d_i} = \frac{x_i!}{d_i!(x_i-d_i)!}$ est le *coefficient binomial* (également noté comme $C_{d_i}^{x_i}$ dans la littérature). En utilisant cette loi de probabilité, nous pouvons calculer la probabilité d'obtenir *au moins* d_i pièces de bonne qualité à partir de x_i pièces lancées :

$$P(x_i^b \geqslant d_i \mid x_i) = \sum_{j=d_i}^{x_i} C_j^{x_i} p_i^j (1-p_i)^{x_i-j} \tag{5.6}$$

Temps de travail effectif aléatoire

Pour modéliser le temps de travail cumulé, nous allons utiliser un *Processus de Renouvellement*. Les processus de renouvellement sont des processus aléatoires constitués de séries d'événements, où les durées des périodes entre deux événements consécutifs sont des valeurs aléatoires positives indépendantes et de même loi. Les processus de renouvellement sont souvent appliqués pour la modélisation des appels téléphoniques (voir par exemple Andersson (2000)), ou une désintégration radioactive des atomes, les *temps de stockage aléatoires* ou encore pour une modélisation des files d'attente. Nous définissons le processus de renouvellement pour notre problème de la manière suivante :

Définition 5.1
Soit $\xi_1, \xi_2, \ldots$ une séquence de variables aléatoires indépendantes identiquement distribuées (i.i.d.) et positives, telles que $0 < \mathbb{E}[\xi_i] < \infty$, où $\mathbb{E}$ est une espérance mathématique. Soit ξ_i le temps entre les $(i-1)$-ème et i-ème pannes (voir la figure 5.3). Chaque J_k est défini comme le k-ème JUMP TIME *et représente le moment de l'arrivée de la k-ème panne. Les intervalles $[J_{k-1}, J_k]$ appelés* INTERVALLES DE RENOUVELLEMENT *représentent les périodes de bon fonctionnement de la machine. Logiquement, $J_k = \sum_{i=1}^{k} \xi_i$. Alors le processus $W^{bf}(t)$ tel que*

$$W^{bf}(t) = \sum_{k=1}^{\infty} \mathbb{I}_{\{J_k \leq t\}} = \max\{\, k : J_k \leq t \,\} \tag{5.7}$$

est appelé un **processus de renouvellement**, *où* $\mathbb{I}_{\{J_k \leq t\}} = \begin{cases} 1, & \text{si } J_k \leq t \\ 0, & \text{sinon} \end{cases}$.

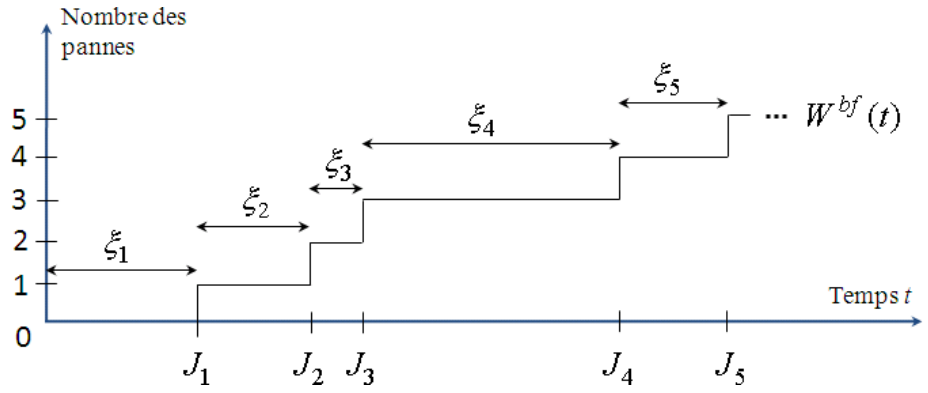


FIGURE 5.3 – Un processus de renouvellement (pannes machine)

En effet, le processus est appelé "renouvellement" parce qu'il "recommence tout à zéro". Ce modèle (voir la définition 5.1) peut être utilisé seulement si nous supposons que la machine est réparée immédiatement après chaque panne, ce qui n'est pas notre cas. Nous devons donc tenir compte d'un autre processus $W^r(t)$ de renouvellement qui représente les réparations dont les durées sont aussi des variables aléatoires strictement positives indépendantes et de même loi. Soit $\psi_1, \psi_2, \ldots$ une séquence de telles variables et $\mathbb{E}[\psi_i] < \infty$, $i = 1, 2, \ldots$ (voir la figure 5.4). Ici chaque R_k est le moment de la fin de k-ème réparation, $R_k = \sum_{i=1}^{k}(\xi_i + \psi_i) = J_k + \sum_{i=1}^{k} \psi_i \quad k = 1, 2, \ldots$

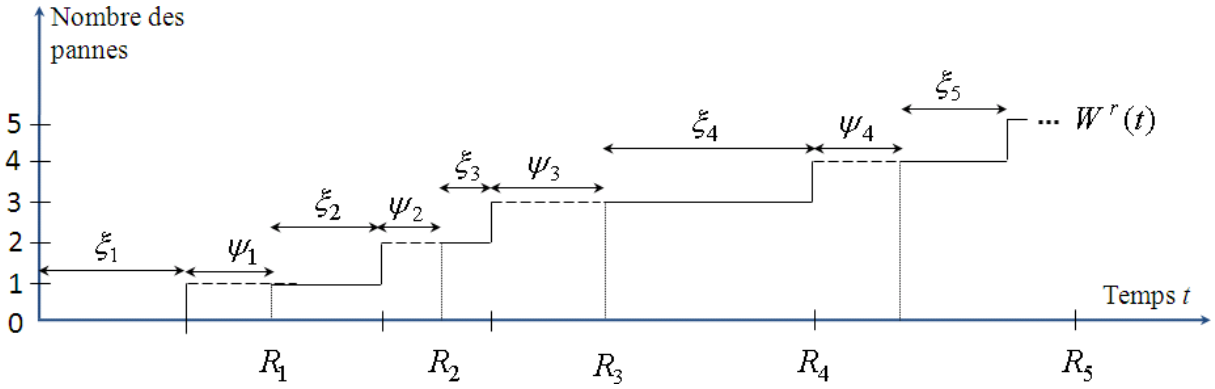


FIGURE 5.4 – Deux processus de renouvellement alternés

Nous avons donc deux *processus de renouvellement alternés* : $(\xi_1, \psi_1), (\xi_2, \psi_2), \ldots$

L'idée principale de notre modélisation est la suivante : nous supposons que l'horizon de planification peut être mathématiquement divisé en deux parties distinctes : la période de bon fonctionnement, quand il n'y a pas de panne, et la période d'inactivité, pendant laquelle toutes les réparations sont effectuées (voir la figure 5.5). Autrement dit, ce regroupement rassemble tous les temps de production dans une période et tous les temps de réparation

dans une autre, ce qui rend possible le calcul des probabilités en utilisant les deux processus de renouvellement. Par conséquent, dans la période de bon fonctionnement de la ligne, le temps de réparation est considéré comme égal à zéro (réparations instantanées). De la même manière, dans la période de réparation, le temps entre deux pannes est considéré comme égal à zéro (défaillance immédiatement après chaque réparation).

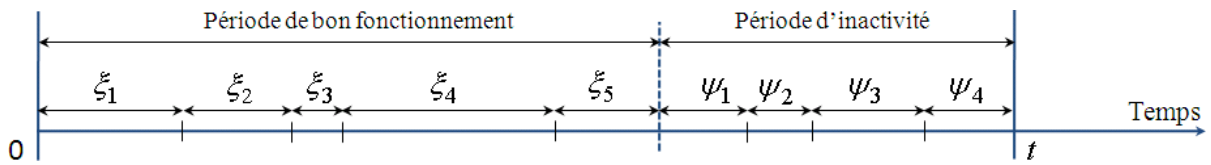


FIGURE 5.5 – Modélisation des pannes machine et des réparations

Un *processus de Poisson* est un cas particulier des processus de renouvellement. En faisant le lien avec notre problème, un processus de Poisson a les propriétés suivantes :

- Le nombre d'événements pour un intervalle ne dépend pas du nombre d'événements pour les autres intervalles de temps (il n'y a pas de mémoire) ;
- Le nombre d'événements (nombre des pannes) pour un intervalle de temps $[0, t]$ ne dépend que de la longueur de cet intervalle et du taux de la loi de Poisson ;
- La loi de probabilité pour les durées entre deux événements successifs est une loi exponentielle négative.

Nous supposons que $W^{bf}(t)$ et $W^{r}(t)$ sont des processus de Poisson (cette hypothèse est souvent faite dans le domaine de fiabilité).

Pour notre problème nous avons m machines en séquence sur la ligne de production. Alors pour modéliser les pannes, nous devons considérer pas un seul, mais une superposition de m processus de Poisson (voir la figure 5.6).

Théorème 5.1 *Soit* $\left\{W_q^{bf}(t), t \geq 0,\ q = 1, 2, \ldots, m\right\}$ *les processus de Poisson indépendants avec les intensités* $\lambda_1, \lambda_2, \ldots, \lambda_m$. *Alors la superposition*

$$W^{bf}(t) = \sum_{q=1}^{m} W_q^{bf}(t) \tag{5.8}$$

est un processus de Poisson avec l'intensité $\lambda = \lambda_1 + \lambda_2 + \ldots + \lambda_m$.

Rappelons que pour notre problème les temps moyens entre deux pannes (MTTF) est égal à $1/u_q$ pour chaque machine q, $q = 1, \ldots, m$. Tenant compte que dans nos notations

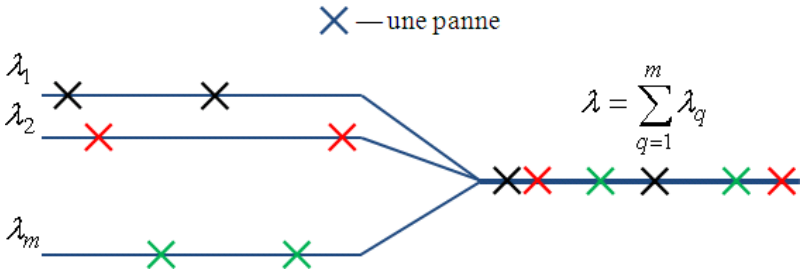


FIGURE 5.6 – Superposition de processus de Poisson

$u_q := \lambda_q$, l'intensité des pannes U pour la ligne entière est calculée comme suit :

$$U = \sum_{i=q}^{m} u_q \tag{5.9}$$

Il y a m machines sur la ligne et chacune a une durée moyenne de réparation (MTTR) $1/r_q$, $q = 1, 2, \ldots, m$ différente. Chaque événement (panne) dans le processus de Poisson $W^r(t)$ appartient à l'un de ces m-types (numéro de machine) : type q avec la probabilité p_q^r, $q = 1, 2, \ldots, m$. Alors nous allons *partitionner* le processus $W^r(t)$ qui représente toutes les réparations (voir la figure 5.7) en m processus de Poisson pour calculer le taux de réparation pour toute la ligne à partir des taux de réparation des machines.

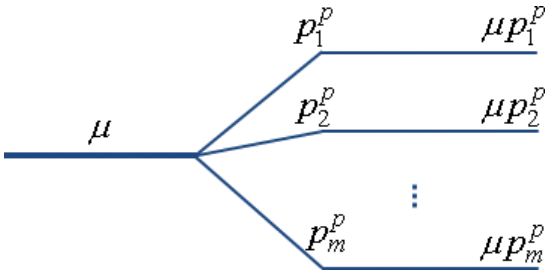


FIGURE 5.7 – Partition de processus de Poisson

Théorème 5.2 *(Nelson (1995)) Soit W^r le processus de Poisson avec l'intensité μ, et W_q^r, $q = 1, 2, \ldots, m$ m processus obtenus par la partition de W^r selon les probabilités p_q^r, $q = 1, 2, \ldots, m$, où $p_1^r + p_2^r + \ldots + p_m^r = 1$. Alors W_q^r sont des processus de Poisson indépendants avec l'intensité μp_q^r*

Dans notre cas les probabilités p_q^r sont égales à :

$$p_q^p = \frac{u_q}{\sum_{j=1}^{m} u_j} = \frac{u_q}{U}, \quad q = 1, 2, \ldots, m$$

Nous avons donc m processus de Poisson avec l'intensité $\frac{u_q}{U} r_q$ chacun. L'intensité R du processus cumulé W^r peut être calculée comme suit :

$$R = \sum_{q=1}^{m} r_q \frac{u_q}{U} \tag{5.10}$$

Les MTTF et MTTR de toute la ligne sont donc :

$$\frac{1}{U} = \left(\sum_{q=1}^{m} u_q \right)^{-1} \quad \text{et} \quad \frac{1}{R} = \left(\sum_{q=1}^{m} r_q \frac{u_q}{U} \right)^{-1} \tag{5.11}$$

Pour notre le modèle mathématique nous avons besoin de quelques notations supplémentaires :

- Le temps de fabrication planifié t :

$$t = T_0 - \text{temps de set-up} - \text{temps de chargement} =$$

$$= T_0 - \left(s_{0,\pi_1} + \sum_{i=2}^{n} s_{\pi_{i-1}\pi_i} \right) - (m-1) \sum_{j=1}^{n} t_{\pi_j}$$

- T^{bf} le temps **réel** de bon fonctionnement de la ligne, $T^{bf} \in [0, t]$;
- T^{rep} le temps total de réparation pour les pannes qui arrivent durant la période $[0, t]$;
- la fonction de répartition du temps de réparation $G(\cdot)$;
- la fonction de répartition du temps entre deux pannes $F(\cdot)$.

Si nous réservons pour le temps de fabrication cumulé s unités de temps, alors pour les réparations il reste $t - s$ unités de temps. En utilisant les lois de distribution exponentielles négatives, nous aurons les fonctions de répartition suivantes :

$$G(t-s) = 1 - \exp^{-R(t-s)} \quad \text{et} \quad F(s) = 1 - \exp^{-Us} \tag{5.12}$$

où t est le temps réservé pour la fabrication.

Revenons à la figure 5.5. En connaissant les lois de distribution de pannes et de réparations nous pouvons calculer la probabilité d'avoir un certain nombre de pannes pour la première période (période de bon fonctionnement) et la probabilité d'avoir un certain nombre de réparations pour la seconde (période d'inactivité). Ainsi ce modèle permet d'estimer la distribution de probabilités du temps total de bon fonctionnement de la ligne (*cumulative*

working time) T^{bf} pour un temps de fabrication t donné. En d'autres termes, nous pouvons calculer la probabilité $A(s,t)$ que le temps total de bon fonctionnement T^{bf} pour la période $[0,t]$ soit plus petit que la constante s. Cette probabilité est égale à la probabilité que le nombre des pannes au cours de la période de bon fonctionnement soit inférieur ou égal au nombre de réparations lors de la période d'inactivité (voir Dolgui (2002)). La formule est la suivante :

$$A(s,t) = P(T^{bf} < s) = \sum_{\nu=0}^{\infty} \left\{ G^{*\nu}(t-s) - G^{*\nu+1}(t-s) \right\} \times F^{*\nu+1}(s) \tag{5.13}$$

où l'indice $*\nu$ représente la ν-ème intégrale de convolution de F (ou G), qui peut être calculée comme suit :

$$F^{*\nu}(s) = \int_0^s F^{*(\nu-1)}(s-y)dF(y), \quad F^{*0}(s) \equiv 1 \tag{5.14}$$

Pour la loi de distribution exponentielle négative $F^{*\nu}(s)$ est égale à

$$F^{*\nu}(s) = 1 - \exp^{-Us} \times \left\{ 1 + \frac{Us}{1!} + \frac{(Us)^2}{2!} + \ldots + \frac{(Us)^{\nu-1}}{(\nu-1)!} \right\} \tag{5.15}$$

Le même type d'expression peut être obtenu pour la fonction $G^{*\nu}(\cdot)$, en remplaçant U pas R et s par $t-s$. En substituant $G^{*\nu}$ et $F^{*\nu}$ dans l'équation (5.13) par les formules obtenues, nous avons :

$$A(s,t) = P(T^{bf} < s) = \exp^{-(Us+R(t-s))} \times \sum_{\nu=1}^{\infty} \frac{(Us)^{\nu}}{\nu!} \sum_{j=0}^{\nu-1} \frac{(R(t-s))^j}{j!} \tag{5.16}$$

Comme nous pouvons voir dans l'équation (5.5), la somme du temps total réel de bon fonctionnement et du temps réel de réparation est égale à la durée du temps de fabrication, c.-à-d. $T^{bf} + T^{rep} = t$, alors la transformation suivante est vérifiée :

$$A(s,t) = P(T^{bf} < s) = P\left(t - T^{bf} > t - s\right) = \\ 1 - P\left(t - T^{bf} \leq t - s\right) = 1 - P(T^{rep} \leq t - s) \tag{5.17}$$

La valeur de $(t-s)$ correspond au temps de sécurité T_s (voir l'équation (5.3)) prévu pour les réparations. Nous allons calculer la probabilité de satisfaire la demande globale (pour tous les lots) pour chaque solution (π, x) donnée. Alors, le temps de bon fonctionnement T^{bf} prévu est égal au temps total de fabrication des pièces : $\sum_{i=1}^{n} t_i x_i$.

5.1.3 Modèle mathématique du problème

Fixons les $(n-1)$ premiers lots, alors le temps restant T^{nr} pour la production de dernier lot est :

$$T^{nr}(\pi, x) = T_0 - \left(s_{0,\pi_1} + \sum_{j=2}^{n} s_{\pi_{j-1},\pi_j} + (m-1) \sum_{j=1}^{n} t_{\pi_j} + \sum_{j=1}^{n-1} t_{\pi_j} x_{\pi_j} \right) \tag{5.18}$$

Ce temps sera utilisé pour la fabrication de ce dernier lot et pour toutes les réparations.

Nous ne considérons que des cas où au moins d_i pièces **de chaque lot** seraient fabriquées, autrement la probabilité finale serait égale à zéro. Alors, en utilisant T^{nr}, la fonction objectif peut être transformée de la manière suivante (Dolgui *et al.* (2005)) :

Problème P-PROB1 :

$$\textit{Maximiser} \quad P\left(x^b_{\pi_j} \geq d_{\pi_j} \mid (\pi, x)\right) = \prod_{j=1}^{n-1} P(x^b_{\pi_j} \geq d_{\pi_j} \mid x_{\pi_j}) \times P\left(x^b_{\pi_n} \geq d_{\pi_n} \mid x_{\pi_n}, T^{nr}(\pi, x)\right) \tag{5.19}$$

Notons que si la fonction objectif est représentée par l'équation (5.19), les probabilités de satisfaire la demande pour les $n-1$ premiers lots ne dépendent que du nombre de pièces défectueuses. En revanche, la probabilité $P\left(x^b_{\pi_n} \geq d_{\pi_n} | x_{\pi_n}, T^{nr}(\pi, x)\right)$ d'obtenir tous les composants nécessaires du dernier lot dépend du nombre de rebuts et du temps de réparation des machines, et elle peut être calculée comme suit :

$$P\left(x^b_{\pi_n} \geq d_{\pi_n} \mid x_{\pi_n}, T^{nr}(\pi, x)\right) = \sum_{z=d_{\pi_n}}^{x_{\pi_n}} P\left(x^b_{\pi_n} \geq d_{\pi_n} \mid z\right) \times P\left(x_{\pi_n} = z \mid T^{nr}(\pi, x)\right) \tag{5.20}$$

où $P\left(x_{\pi_n} = z \mid T^{nr}(\pi, x)\right)$ est la probabilité d'avoir suffisamment de temps pour fabriquer exactement z composants du dernier lot. A son tour, cette probabilité peut être calculée comme suit :

$$P\left(x_{\pi_n} = z \mid T^{nr}(\pi, x)\right) = P\left(x_{\pi_n} \geq z \mid T^{nr}(\pi, x)\right) - P\left(x_{\pi_n} \geq z+1 \mid T^{nr}(\pi, x)\right) \tag{5.21}$$

Ici $P\left(x_{\pi_n} \geq z \mid T^{nr}(\pi, x)\right)$ est la probabilité de fabriquer au moins z pièces de type π_n, qui est équivalente à la probabilité que le temps de réparation effective T^{rep} ne dépasse pas la

valeur T^s. La fonction objective complète peut donc être représentée comme suit :

$$P(x^b_{\pi_j} \geqslant d_{\pi_j} \mid \pi_j) = \prod_{j=1}^{n-1} P(x^b_{\pi_j} \geq d_{\pi_j} \mid x_{\pi_j}) \times$$
$$\sum_{z=d_{\pi_n}}^{x_{\pi_n}} P\left(x^b_{\pi_n} \geq d_{\pi_n} \mid z\right) \times \Big(P\Big(x_{\pi_n} \geq z \mid T^{nr}(\pi, x)\Big) - P\Big(x_{\pi_n} \geq z+1 \mid T^{nr}(\pi, x)\Big)\Big) \quad (5.22)$$

Notons que les problèmes P-Prob et P-Prob1 sont équivalents. Dans ce qui suit nous allons garder la notation P-Prob. Pour ces deux problèmes la fonction objectif peut être exprimée par l'expression (5.22). Dans les sections suivantes nous décrivons les méthodes de résolution de ce problème.

5.2 Méthode d'optimisation pour P-Prob

5.2.1 Calcul des bornes pour les tailles de lots

Dans cette section, nous montrons comment calculer des bornes inférieure et supérieure pour les tailles des lots (Dolgui *et al.* (2005)). Ces bornes seront utilisées pour l'évaluation de chaque solution réalisable.

Supposons que nous avons une solution avec le niveau de service β, c.-à-d. la probabilité de satisfaire la demande globale est égale à β. Alors une autre solution est intéressante uniquement si son niveau de service est supérieur où égal à β, c.-à-d. que le niveau de service de chaque produit pris séparément doit être supérieur ou égal à β :

$$P(x^b_i \geq d_i \mid x_i) \geq \beta$$

Le même raisonnement s'applique quand un *niveau de service minimal* de la ligne est fixé depuis le début, c.-à-d. nous avons à chercher des solutions avec le niveau de service global plus grand que (ou égal à) ce niveau de service minimal fixé. Soit β ce niveau de service donné, alors nous pouvons trouver les quantités minimales x_i^{min} de pièces de chaque type à lancer en fabrication :

$$x_i^{min} = \min\left\{z| \quad P(x^b_i \geq d_i|z) \geq \beta, \quad z = d_i, d_i+1, \ldots\right\} \quad (5.23)$$

en utilisant l'equation (5.6).

Dans (5.23) x_i^{min} est la taille du lot i, telle que la probabilité de satisfaire la demande d_i soit supérieure ou égale à β. Quand la valeur de β est connue, une solution est considérée comme réalisable si nous avons suffisamment de temps pour fabriquer au moins x_i^{min} pièces de chaque type $i = 1, 2, \ldots, n$.

De la même façon, la taille maximale x_i^{max} pour chaque lot peut être calculée comme suit :

$$x_i^{max} = \min\left\{z| \quad P(x_i^b \geq d_i|z) \geq 1 - \delta, \quad z = d_i, d_i + 1, \ldots\right\} \tag{5.24}$$

où δ est une valeur relativement petite, $\delta > 0$. La valeur de x_i^{max} est la taille minimale du lot i telle que la probabilité d'obtenir au moins d_i composants de bonne qualité soit suffisamment proche de 1. Dans un certain nombre de cas, la borne supérieure x_i^{max} peut être améliorée en utilisant la valeur x_i^{upp} suivante :

$$x_i^{upp} = \min\left\{x_i^{max},\ x_i^{min} + T_s(\pi, x^{min})/t_i\right\} \tag{5.25}$$

où $x^{min} = (x_1^{min}, \ldots, x_n^{min})$. Nous utilisons cette condition complémentaire (5.25) puisque dans certains cas la valeur de x_i^{max} calculée suivant (5.24) peut être très grande. La valeur $x_i^{min} + T_s(\pi, x^{min})/t_i$ représente la quantité maximale de composants de type i que nous pouvons **réellement** fabriquer sous les conditions suivantes : les tailles des autres lots sont les plus petites possibles (x_j^{min}), où $j = 1, \ldots, n,\ i \neq j$; il n'y a pas de temps perdu, c.-à-d. aucune machine n'est tombée en panne.

5.2.2 Décomposition

Une approche de décomposition pour le problème P-PROB à été proposée par Dolgui *et al.* (2005). Ici, nous allons brièvement la présenter sous une autre forme, et en utilisant l'exemple du Chapitre 2. La décomposition nous permet de résoudre le problème P-PROB comme indiqué dans la figure 5.8, où le premier niveau est une simple énumération. Au deuxième niveau, nous avons un cas particulier du problème du *Voyageur de commerce*(VDC) et au troisième niveau nous obtenons une modification du problème de *Sac à Dos* (Knapsack problem en anglais).

Soit Π l'ensemble de toutes les permutations π sur $\{1, 2, \ldots, n\}$ et $X = X_1 \times X_2 \times \ldots \times X_n$. Chaque $X_i = [x_i^{min}, x_i^{upp}]$ est l'ensemble de toutes les valeurs possibles de x_i, c.-à-d. l'ensemble des tailles de lot possibles pour le produit i, $i = 1, \ldots, n$. Soit (π^*, x^*) la solution optimale du problème initial (nommé P-PROB). Rappelons que la fonction objectif du problème P-PROB est représentée par l'équation (5.22).

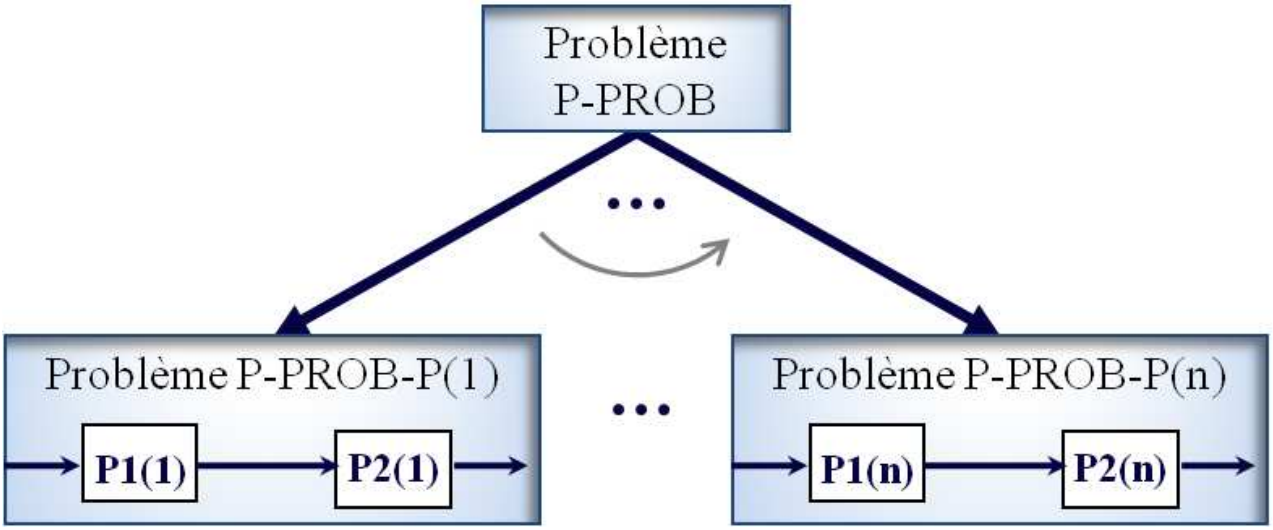


FIGURE 5.8 – Schéma de décomposition

Le premier niveau de décomposition consiste en la séparation du problème P-PROB en n sous-problèmes équivalents P-PROB-P(i), $i = 1, \ldots, n$. Pour le P-PROB-P(i) le dernier lot dans la séquence π^* recherchée correspond au produit i, $i = 1, \ldots, n$ (c.-à-d. $\pi_n^* = i$). Nous obtenons tous les n problèmes P-PROB-P(i), $i = 1, \ldots, n$ par une simple énumération.

Soit $\Pi(i)$ l'ensemble de toutes les permutations $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ tel que $\pi_n = i$. Alors le problème P-PROB-P(i) peut être formulé de la façon suivante :

Problème P-PROB-P**(i)** :

$$\textit{Maximiser} \quad P\left(x_{\pi_j}^b \geq d_{\pi_j} | \, (\pi, x)\right) = \prod_{j=1}^{n-1} P(x_{\pi_j}^b \geq d_{\pi_j} | x_{\pi_j}) \times P\Big(x_i^b \geq d_i | x_i, T^{nr}(\pi, x)\Big) \quad (5.26)$$

où $(\pi, x) \in \Pi(i) \times X$.

La résolution de chaque problème P-PROB-P(i) donne une solution optimale (séquence et tailles de lot) pour le cas $\pi_n = i$, c.-à-d. quand i est le dernier lot à fabriquer. Lorsque tous les problèmes P-PROB-P(i) sont résolus ($i = 1, \ldots, n$), n solutions "optimales" sont obtenues. Nous pouvons choisir la meilleure parmi elles, elle sera la solution optimale (π^*, x^*) du problème P-PROB.

Chaque problème P-PROB-P(i), à son tour, peut être décomposé en deux sous-problèmes : P-PROB-P1(i) et P-PROB-P2(i).

Donnons les définitions de ces problèmes. Quand le dernier lot $\pi_n = i$ est fixé, la séquence des lots n'a d'influence que sur la valeur du temps de traitement total, c.-à-d. sur la proba-

bilité $P\left(x_n^b \geq d_n \mid x_n, T^{nr}(\pi, x)\right)$ utilisé dans la fonction objectif. Alors la minimisation du temps total de set-up va augmenter cette probabilité et n'aura aucun effet sur les autres parties de la fonction objectif. Pour optimiser la séquence des lots, quand le dernier lot est connu, il faut résoudre le problème suivant :

Problème P-PROB-**P1(i)** :

Trouver une séquence π^{*i} pour

$$\textit{Maximiser} \quad S(\pi) = s_{0,\pi_1} + \sum_{j=1}^{n-1} s_{\pi_{j-1},\pi_j} + s_{\pi_{n-1},i}, \quad \pi \in \Pi(i) \tag{5.27}$$

Ce problème est équivalent au problème du VDC. La minimisation de $S(\pi)$ correspond à trouver le *chemin Hamiltonien* le plus court dans un graphe orienté complet (si les arcs du graphe sont évalués par le temps de set-up entre les deux produits correspondants).

Supposons alors que la séquence π^{*i} qui minimise la somme (5.27) a été trouvée, et que nous avons numéroté les lots conformément à cette séquence. Nous pouvons alors formuler le problème P-PROB-P2(i) correspondant.

Problème P-PROB-P2(I) : Trouver les tailles de lots $x^* = (x_1, x_2, \ldots, x_n)$ pour :

$$\textit{Maximiser} \quad P\left(x_j^b \geq d_j \mid (\pi^{*i}, x)\right) = \tag{5.28}$$
$$\prod_{j=1}^{n-1} P(x_j^b \geq d_j \mid x_j) \times P\left(x_n^b \geq d_n \mid x_n, T^{nr}(\pi^{*i}, x)\right)$$

L'augmentation du nombre de pièces dans un lot j entraîne l'augmentation de la probabilité $P(x_j^b \geq d_j \mid x_j)$, mais diminue le temps T^{nr} donc peut diminuer la probabilité finale. Dolgui *et al.* (2005) ont montré comment transformer ce problème en une modification du problème "Sac à Dos" et ont proposé une approche de programmation dynamique pour le résoudre (voir l'annexe D).

5.2.3 Un exemple

En utilisant l'exemple du chapitre 2, nous allons montrer un exemple de recherche de la solution optimale pour chaque dernier lot fixé. La procédure de programmation dynamique de Dolgui *et al.* (2005) a été utilisée (codée en C++, voir l'annexe D.2) pour résoudre les sous-problèmes P-PROB-P2(i), $i = 1, \ldots, n$. Pour les sous-problèmes P-PROB-P1(i) nous

avons utilisé un algorithme génétique, qui a été lancé 50 fois pour chaque $i = 1, \ldots, n$ avec 400 générations dans chaque lancement. Le résultat retenu est la meilleure solution trouvée parmi les 50 solutions obtenues. Le tableau 5.1 montre les meilleures séquences π^{*i} obtenues pour chaque problème P-Prob-P1(i). La dernière colonne contient le temps total de set-up $S(\pi^{*i})$.

Table 5.1 – Solutions π^{*i} pour les problèmes P-Prob-P1(i)

	Numéro dans la séquence								*Temps de*
Problèmes	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	*set-up(sec)*
P1(1)	5	6	3	8	7	2	4	1	2
P1(2)	1	6	3	5	4	8	7	2	1,92
P1(3)	**1**	**2**	**5**	**4**	**8**	**7**	**6**	**3**	**1,9**
P1(4)	1	6	3	8	7	2	5	4	1,91
P1(5)	1	6	3	8	7	2	4	5	1,93
P1(6)	1	3	5	2	4	8	7	6	1,95
P1(7)	1	6	3	5	2	4	8	7	1,91
P1(8)	*1*	*6*	*3*	*5*	*7*	*2*	*4*	*8*	*1,96*

Le tableau 5.2 montre les solutions optimales pour chaque problème P-Prob-P2(i), $i = 1, \ldots, n$. La dernière colonne donne la probabilité de satisfaction de la demande (niveau de service pour chaque problème). Dans les colonnes 2 - 9, nous avons les tailles de lot correspondantes.

Table 5.2 – Solutions x^{*i} pour des problèmes P-Prob-2(i)

	Numéro de lot								*Probabilité*
Problèmes	1	2	3	4	5	6	7	8	*finale*
P2(1)	41	54	72	100	58	92	123	46	0,969547
P2(2)	39	56	72	100	57	92	123	46	0,966277
P2(3)	39	54	73	99	57	92	123	46	0,964216
P2(4)	38	53	72	101	57	92	122	46	0,960182
P2(5)	39	54	72	100	60	92	123	46	0,968483
P2(6)	39	54	72	99	57	94	123	46	0,963894
P2(7)	39	54	72	100	57	92	125	46	0,965109
P2(8)	**39**	**54**	**72**	**100**	**58**	**92**	**123**	**49**	**0,974574**

Remarquons que chaque probabilité obtenue dans la dernière colonne est le résultat de l'application de l'approche de décomposition au problème initial P-PROB, puis de la résolution des problèmes de VDC avec le dernier lot i fixé P-PROB-P1(i) et ensuite de la résolution des problèmes de lotissement P-PROB-P2(i), $i = 1, \ldots, n$.

Comme nous pouvons le voir, la meilleure séquence des lots (en gras dans le tableau 5.1) avec la valeur du temps de set-up la plus faible, a été obtenue en supposant que un lot de produit 3 est le dernier. Le temps total de set-up de cette séquence est égal à $1,9$ heures. Par contre, comme nous pouvons le constater dans le tableau 5.2, le niveau de service le plus élevé a été obtenu pour P-PROB-P2(8) (en gras dans le tableau 5.2). Donc, la probabilité (niveau de service) maximale est obtenue avec la séquence π^* en italique dans le tableau 5.1, qui ne donne pas le temps de set-up le plus faible.

Par la suite, nous ne nous intéressons qu'à la résolution des problèmes P-PROB-P2(i), puisque il y a une quantité de travaux importante traitant des problèmes de voyageur de commerce. Nous supposons, que la séquence de lots π^{*i} a été trouvée en utilisant une des techniques connues. Tous les lots sont renumérotés conformément à la séquence trouvée.

5.3 Résolution du problème P-PROB-P2(i)

Nous proposons deux méthodes pour les problèmes P-PROB-P2(i) : recherche locale et algorithme génétique.

5.3.1 Recherche Locale

La recherche locale est une méthode qui est souvent utilisée pour résoudre les problèmes combinatoires. La procédure commence par la création d'une solution initiale. Cette solution peut être obtenue de manière aléatoire ou trouvée avec l'aide d'un algorithme heuristique. Ensuite, en utilisant un critère de voisinage (qui dépend du problème considéré), l'espace de toutes les solutions-voisines doit être construit. Après avoir effectué une évaluation de toutes les solutions de cet espace, la meilleure solution est choisie pour remplacer la solution initiale à la prochaine itération. L'itération suivante commence et ainsi de suite. Le critère d'arrêt de la procédure peut être : a) la limite du temps de calcul alloué est atteinte ; b) la meilleure solution actuelle ne peut pas être améliorée (un optimum local a été trouvé).

Rappelons que les variables de décision pour un problème P-PROB-P2(i) sont les tailles de lots $x_j,\ j = 1, \ldots, n-1$. La taille **effective** du dernier lot est définie après cette optimisation,

parce qu'elle dépend de T^{nr}.

En prenant compte les réflexions ci-dessus, chaque solution peut être représentée comme un tableau de nombres entiers, où chaque élément représente la taille de lot appropriée et la longueur de chaque solution est égale à $n-1$ (si nous avons n types de produit). Le tableau 5.3 présente un exemple d'une solution pour un problème avec 8 lots, la taille initiale du dernier lot est fixé à x_n^{upp} pour toutes les solutions traitées.

TABLE 5.3 – Exemple d'une solution pour un problème avec 8 lots

Numéro de produit	1	2	3	4	5	6	7
Quantité à fabriquer	21	57	3	17	12	14	8

Pour générer une solution, nous appliquons l'algorithme glouton **G1** présenté ci-dessous (l'algorithme 5.1). L'idée principale est de prendre une solution initiale y^0, où toutes les tailles de lots y_i^0 sont mises à leurs valeurs minimales x_i^{min}. Ensuite, en augmentant la taille de lot avec la valeur de probabilité $P(x_i^b \geq d_i \mid y_i^0)$ minimale, nous pouvons augmenter la probabilité totale de satisfaire la demande pour la solution actuelle. Si nous avons réussi à augmenter la probabilité, nous continuons ce processus de la même façon (nous changeons une valeur dans chaque itération), sinon - l'optimum local a été trouvé et l'algorithme s'arrête.

Pour introduire l'algorithme de recherche locale, nous avons besoin de la définition suivante :

Définition 5.2
Soit y^1 et y^2 deux solutions arbitraires du problème. Alors la distance *$d(y^1, y^2)$ entre les solutions y^1 et y^2 est la suivante :*

$$d\left(y^1, y^2\right) = \sum_{s=1}^{n-1} |y_s^1 - y_s^2| \tag{5.29}$$

Pour construire l'algorithme de recherche locale, prenons la dernière solution y^j, obtenue avec l'algorithme **G1** et appliquons la procédure décrite dans l'algorithme 5.2.

5.3.2 Algorithme Génétique

Dans cette section, nous proposons un algorithme génétique (AG) pour résoudre les problèmes P-PROB-P2(i). Rappelons que l'objectif de ces problèmes est de trouver les tailles

Algorithme 5.1: Algorithme glouton $G1$

1 $j \leftarrow 0$

2 *Créer* $y^0 = \left(y_1^0, y_2^0, \ldots, y_{n-1}^0\right)$, où $y_i^0 = x_i^{min}$, $i = 1, \ldots, n-1$
`// solution initiale`

3 *Calculer* $T^{bf}(y^0) = \sum_{i=1}^{n-1} t_i y_i^0$ `// le temps de fabrication pour` y^0

4 $I_{min}^0 \leftarrow \arg\min_{i \in \{1,\ldots,n-1\}} \left(P\left(x_i^b \geq d_i \mid y_i^0\right)\right)$

5 $p_0 \leftarrow P\left(x_i^b \geq d_i, \quad i = 1, \ldots, n \mid y^0\right)$

6 *Créer* y^1 telle que : $y_i^1 = \begin{cases} y_i^0 + 1, & \text{si} \quad i = I_{min}^0 \\ y_i^0, & \text{sinon} \end{cases} \qquad i = 1, \ldots, n-1$

7 $j \leftarrow 1$

8 **tant que** $\left(T^{pr}(y^j) < T - S(\pi^*) - (m-1)\sum_{i=1}^{n} t_i\right)$ ***et*** $\left(P\left(x_i^b \geq d_i, \quad i = 1, \ldots, n \mid y^j\right) > p_{j-1}\right)$ **faire**

9 $\quad p_j \leftarrow P\left(x_i^b \geq d_i, \quad i = 1, \ldots, n \mid y^j\right)$

10 $\quad I_{min}^j \leftarrow \arg\min_{i \in \{1,\ldots,n-1\}} \left(P\left(x_i^b \geq d_i \mid y_i^j\right)\right)$

11 $\quad$ *Créer* y^{j+1} telle que : $y_i^{j+1} = \begin{cases} y_i^j + 1, & \text{si } i = I_{min}^j \\ y_i^j, & \text{sinon} \end{cases} \qquad i = 1, \ldots, n-1$

12 $\quad j \leftarrow j + 1$

13 **fin**

de lot qui maximisent la probabilité (5.2.2). Les algorithmes génétiques ont été initialement proposées par Holland (1975), Goldberg (1989). Pour une description générale de AG, voir également Cox (2005), Said (2005), Leardi (2001). Il y a quelques articles présentant un AG pour différents modèles de dimensionnement des lots. Par exemple, Lee *et al.* (1997) proposent un AG pour la résolution d'un problème composé (lotissement et ordonnancement simultanément) afin de minimiser le makespan. Dans Hernandez (1999), un algorithme génétique est proposé pour un problème de lotissement avec un seul produit, sans limitation de ressources et mono-niveau. Dellaert *et al.* (2000) suggèrent un algorithme génétique avec un codage binaire pour résoudre un problème de dimensionnement de lots à plusieurs niveaux. Gaafar (2006) a appliqué un AG à un problème de lotissement déterministe avec ordonnancement des lots.

Nous utilisons une procédure standard pour des algorithmes génétiques (voir la figure

Algorithme 5.2: Recherche locale

Entrées : Solution y, obtenu avec $G1$
Sorties : Maximum local

1 *Créer l'ensemble $N(y)$ des solutions-voisines pour la solution $y = (y_1, \ldots, y_{n-1})$ comme suit :*
$N(y) = \left\{ y^{'} | \quad d(y, y^{'}) = 1,\ x_i^{min} \leq y_i^{'} \leq x_i^{upp},\ i = 1, \ldots, n-1 \right\}$
2 *Évaluer toutes les solutions de l'ensemble $N(y)$* `// voir la section 5.3.2`
3 **si** $\left(\exists \quad y^{'} \in N(y) : f(y^{'}) > f(y)\right)$ **alors**
4 $\quad y_{max}^{'} = y^{'}$ tel que $f(y^{'}) = \max\left\{ f(y^{'}) \mid y^{'} \in N(y) \right\}$
5 $\quad y \leftarrow y_{max}^{'}$
6 $\quad$ *Revenir au pas* 1 *de l'algorithme*
7 **sinon**
8 $\quad$ *Quitter*`// maximum local a été atteint`
9 **fin**

5.9) : i) la création d'un ensemble de solutions (*population initiale*), ii) l'application des opérations de croisement et de mutation. Pour améliorer l'algorithme nous utilisons une procédure de recherche locale à la fin de chaque itération. Notons que les algorithmes génétiques hybrides avec une procédure de recherche locale intégrée sont appelés les algorithmes *Mémétiques* (Moscato (1989)).

Codage des solutions et fonction d'évaluation (fitness)

Comme il a été expliqué dans la section précédente, chaque solution (*chromosome*) peut être représentée par un tableau de taille $n-1$ (voir le tableau 5.3).

Il faut définir une fonction *d'évaluation* (fonction *fitness* par la suite) pour pouvoir estimer la performance de chaque solution. Nous cherchons à maximiser le niveau de service global, alors pour calculer le fitness d'un chromosome, nous devons utiliser l'équation (5.16), où une somme infinie est présente. Néanmoins, nous pouvons trouver des bornes inférieures et supérieures pour obtenir une valeur approximative de cette probabilité comme suit (Dolgui (2002)) :

$$\overline{P\left(T^{rep} \leq T^{s} | T^{nr}\right)} = \exp\left\{ -\left(UT^{fb} + RT^{s}\right) \right\} \times \sum_{\nu=0}^{L} \left(\frac{(RT^{s})^{\nu}}{\nu!} \sum_{j=0}^{\nu} \frac{\left(UT^{fb}\right)^{j}}{j!} \right) \tag{5.30}$$

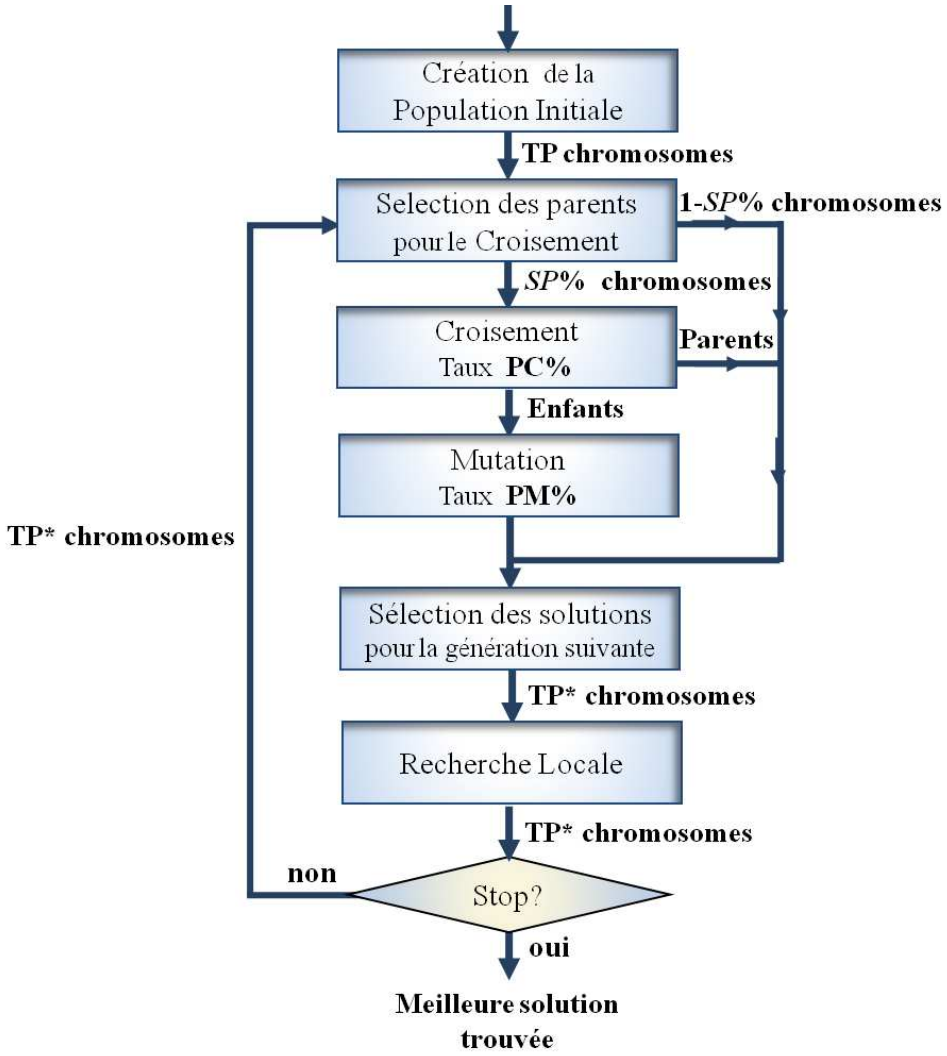


FIGURE 5.9 – Schéma de l'algorithme Génétique (AG) proposé

$$\underline{P\left(T^{rep} \leq T^{s} | T^{nr}\right)} = 1 - \exp\left\{-\left(UT^{fb} + RT^{s}\right)\right\} \times \sum_{\nu=1}^{L} \left(\frac{\left(UT^{fb}\right)^{\nu}}{\nu!} \sum_{j=0}^{\nu-1} \frac{\left(RT^{s}\right)^{j}}{j!} \right) \tag{5.31}$$

Il faut choisir une valeur L (le nombre d'éléments de la somme) assez grande pour obtenir un écart assez faible entre les deux limites. Nous allons la choisir en utilisant l'équation suivante :

$$\overline{P\left(T^{rep} \leq T^{s} | T^{nr}\right)} - \underline{P\left(T^{rep} \leq T^{s} | T^{nr}\right)} \leq 2\varepsilon \tag{5.32}$$

où ε est une valeur suffisamment petite, $\varepsilon > 0$.

La valeur de *fitness* $f(g)$ d'une solution g est donc la suivante :

$$f(g) = \prod_{i=1}^{n-1} P\left(x_i^b \geq d_i \mid g_i\right) \times P\left(x_n^b \geq d_n \mid x_n^{upp},\ T^{nr}(g)\right) \tag{5.33}$$

Notons que toutes les évaluations des solutions sont effectuées sous l'hypothèse que $x_n = x_n^{upp}$.

Pour diminuer le temps de calcul, après la première génération du AG, nous initialisons la matrice $M\langle X, p\rangle$, où chaque ligne $i, \quad i = 1, \ldots, n-1$ contient les probabilités $P(x_i^b \geq d_i \mid x_i)$ pour chaque $x_i \in [x_i^{min}, x_i^{upp}]$, de la manière suivante :

$$M[i][x_i - x_i^{min}] = P(x_i^b \geq d_i \mid x_i) \tag{5.34}$$

Notons que la quantité d'éléments dans chaque ligne peut n'est pas être la même.

Création de la population initiale

Soit la taille de la population initiale égale à TP. Pour avoir, dès le début, quelques solutions de bonne qualité, nous appliquons trois algorithmes heuristiques. Le premier (**G1**), a été présenté dans la section 5.3.1. L'idée du deuxième algorithme (**G2**) (voir algorithme 5.3) est la suivante : nous créons une solution g^0, où chaque gène $g_i^0, \quad i = 1, \ldots, n$ est égal à x_i^{upp} - la valeur maximale de la taille de lot correspondant. Ensuite, en diminuant la taille de lot, pour lequel la probabilité de satisfaire la demande est maximale, nous espérons augmenter le fitness de la solution g^0. Si c'est le cas, nous continuons à diminuer les valeurs correspondantes (une pour chaque itération), sinon l'optimum local a été atteint et l'algorithme s'arrête.

Le troisième algorithme (**G3**) consiste à trouver la solution "moyenne"

$$g^{mne} = \left(g_1^{mne}, \ldots, g_{n-1}^{mne}\right) \text{ où } g_i^{mne} = \lfloor \left(x_i^{min} + x_i^{max}\right)/2 \rfloor$$

Les trois algorithmes **G1**, **G2** et **G3** nous donnent 3 solutions (chromosomes). Les $(TP-3)$ chromosomes de la population initiale qui restent sont générés aléatoirement, en respectant les limites inférieure x_i^{min} et supérieure x_i^{upp} pour chaque gène g_i. Pour éviter les solutions irréalisables, la condition suivante doit être vérifiée :

$$\sum_{i=1}^{n-1} t_i g_i + t_n x_n^{min} < T_0 - S(\pi^*) - (m-1)\sum_{i=1}^{n} t_i \tag{5.35}$$

Algorithme 5.3: Algorithme glouton $G2$

1 $j \leftarrow 0$

2 *Créer* $g^0 = \left(g_1^0, g_2^0, \ldots, g_{n-1}^0\right)$, où $g_i^0 = x_i^{min}, \quad i = 1, \ldots, n-1$ `// solution initiale`

3 *Calculer* $T^{bf}(g^0) = \sum_{i=1}^{n-1} t_i g_i^0$ `// le temps de fabrication pour` g^0

4 $I_{max}^0 \leftarrow \arg\max_{i \in \{1,\ldots,n-1\}} \left(P\left(x_i^b \geq d_i \mid g_i^0\right)\right)$

5 $p_0 \leftarrow P\left(x_i^b \geq d_i, \quad i = 1, \ldots, n \mid g^0\right)$

6 **tant que** $\left(T^{bf}(g^0) + t_n x_n^{min} > T - S(\pi^*) - (m-1)\sum_{i=1}^{n} t_i\right)$ **faire**

7 $\quad g_{I_{max}^0}^0 \leftarrow g_{I_{max}^0}^0 - 1$

8 $\quad I_{max}^0 \leftarrow \arg\max_{i \in \{1,\ldots,n-1\}} \left(P\left(x_i^b \geq d_i \mid g_i^0\right)\right)$

9 **fin**

10 *Créer* $g^1 : g_i^1 = \begin{cases} g_i^0 - 1, & si \quad i = I_{max}^0 \\ g_i^0, & sinon \end{cases} \qquad i = 1, \ldots, n-1$

11 $j \leftarrow 1$

12 **tant que** $\left(P\left(x_i^b \geq d_i, \quad i = 1, \ldots, n \mid g^j\right) > p_{j-1}\right)$ **faire**

13 $\quad p_j \leftarrow P\left(x_i^b \geq d_i, \quad i = 1, \ldots, n \mid g^j\right)$

14 $\quad I_{max}^j \leftarrow \arg\min_{i \in \{1,\ldots,n-1\}} \left(P\left(x_i^b \geq d_i \mid g_i^j\right)\right)$

15 $\quad$ *Créer* g^{j+1} tel que $g_i^{j+1} = \begin{cases} g_i^j + 1, & \text{si } i = I_{max}^j \\ g_i^j, & \text{sinon} \end{cases} \qquad i = 1, \ldots, n-1$

16 $\quad j \leftarrow j + 1$

17 **fin**

Si la solution g est une solution réalisable, nous la comparons avec l'ensemble des solutions déjà générées pour éviter : 1) les clones, 2) les solutions qui sont trop proches les unes des autres (la distance (5.29) est inférieure à une certaine valeur).

Sélection et croisement

Pour construire les couples de parents pour le croisement, nous utilisons une *sélection par tournoi*. Pour cela, nous choisissons au hasard deux chromosomes de la population courante. Ensuite, celui qui a la valeur de fitness la plus grande, va devenir le premier parent, le second chromosome est rejeté dans la population. Le deuxième parent est choisi de la même manière. Cette procédure doit être répétée autant de fois que nécessaire pour obtenir $SP\%$ solutions qui sont sélectionnées comme des parents pour le croisement.

Algorithme 5.4: Opération de croisement

Entrées : Couple de parents g^{p1} et g^{p2}

pour chaque $(i = 1; \quad i < n;\ i \leftarrow i + 1)$ **faire**
  Générer une valeur $s_i \in [0, 1]$; `// probabilité que le gène` i `sera changé`
  si $(s_i < PC/100)$ **alors**
    $g_i^{<} \leftarrow \min\left\{g_i^{p1}, g_i^{p2}\right\}$
    $g_i^{>} \leftarrow \max\left\{g_i^{p1}, g_i^{p2}\right\}$
    $g_i^{e1} \leftarrow \text{uniform}\left[g_i^{<}, g_i^{<}\right]$
    $g_i^{e2} \leftarrow \text{uniform}\left[g_i^{<}, g_i^{<}\right]$
  sinon
    $g_i^{e1} \leftarrow g_i^{p1}$
    $g_i^{e2} \leftarrow g_i^{p2}$
  fin
fin
Ajouter g_i^{e1} *et* g_2^{e2} *dans la population*

La probabilité de croisement est fixée à $PC/100$ (la même pour chaque paire de parents potentiels), cela veut dire qu'il n'est pas sûr que chaque couple de parents choisi à l'étape précédente va produire deux nouveaux descendants. Pour réaliser l'opération de croisement nous générons aléatoirement une valeur s dans l'intervalle $[0, 1]$. Si l'inégalité $s \leq PC/100$ est vérifiée, l'opération de croisement est appliquée pour cette paire de parents. La procédure de croisement pour une paire de parents g^{p1} et g^{p2} est donnée par l'algorithme 5.4.

Mutation

L'opération de mutation est appliquée à tous les enfants obtenus lors du dernier croisement. La probabilité de mutation est égale à $PM/100$ pour chaque enfant. Soit le chromosome

$g = (g_1, \ldots, g_{n-1})$ un enfant nouvellement créé. Pour déterminer quels gènes de ce chromosome vont muter, nous utilisons la même probabilité $PM/100$ de mutation, c.-à-d. chaque gène i, $\quad i = 1, \ldots, n-1$ du chromosome g a une probabilité PM de muter. L'opération de mutation pour un chromosome g est présentée par l'algorithme 5.5.

Algorithme 5.5: Opération de mutation

Entrées : Un enfant g

pour chaque $(i = 1; \quad i < n; \quad i \leftarrow i + 1)$ **faire**
  Générer une valeur $mv_i \in [0, 1]$ `// probabilité que le gène` i `sera changé`
  si $(mv_i < PM/100)$ **alors**
    Construire l'ensemble M^i *des valeurs pour le gène* i *comme suit :*
    $M^i = \left\{ -\lceil \frac{x_i^{upp} - x_i^{min}}{4} \rceil, \ldots, \lceil \frac{x_i^{upp} - x_i^{min}}{4} \rceil \right\} \bigcap \left\{ [x_i^{min} - g_i, -2] \bigcup [2, x_i^{upp} - g_i] \right\}$
    Choisir aléatoirement un élément m^i *de l'ensemble*
    $g_i^{mut} \leftarrow g_i + m^i$
  sinon
    $g_i^{mut} \leftarrow g_i$
  fin
fin

Après avoir effectué toutes les mutations, il faut évaluer les enfants créés avec la fonction fitness, trier la population dans l'ordre décroissant de la valeur fitness et effectuer l'insertion des enfants dans la population à l'aide des opérations "sélection" et "remplacement".

Sélection et remplacement

Nous pouvons mettre à jour la valeur β en fonction de la meilleure solution dans la population en cours : si la valeur de fitness f^1 de la meilleure solution est plus grande que le niveau de service actuel, mettons $\beta = f^1$, puis recalculons toutes les valeurs x_i^{min} en utilisant l'equation (5.23). En considérant les nouvelles valeurs de x_i^{min}, nous vérifions que chaque gène g_i de toutes les solutions dans la population est toujours dans l'intervalle $[x_i^{min}, x_i^{max}]$. Si ce n'est pas le cas (c.-à-d. $\exists g_i < x_i^{min}$) mettons $g_i \leftarrow x_i^{min}$. A la fin de cette procédure il faut mettre à jour la matrice $M\langle X, p\rangle$. Cela rend l'algorithme plus efficace puisque nous diminuons en permanence les intervalles pour des tailles de lots et ne créons que des solutions potentiellement intéressantes.

La procédure de suppression de clone inclut l'élimination des solutions identiques et des solutions qui sont très proches les unes des autres. Pour atteindre ce dernier objectif, nous calculons la distance entre chaque paire de solutions de la population (voir la définition 5.2). Si cette distance est strictement inférieure à 3 pour une paire de solutions, nous en éliminons une, celle qui a la valeur de probabilité la plus faible.

Si, après cette démarche, le nombre de solutions dans la population est inférieure à TP, nous régénérons (si c'est possible) la quantité requise de solutions, en respectant la "distance" entre elles. Des fois, surtout pour des problèmes de petites tailles, il n'est pas possible de trouver la quantité de solutions nécessaire, alors, après avoir effectué 20 essais infructueux pour générer une solution, nous diminuons le taille de la population. Si le nombre de solutions dans la population est supérieur à TP, une stratégie de *sélection par rang* (ou d'élitisme) est utilisée pour choisir les TP solutions pour la prochaine génération.

La recherche locale

La procédure de recherche locale est généralement appliquée à la meilleure solution obtenue dans la génération en cours. Nous utilisons une itération de l'algorithme présenté dans la section 5.3.1. Si la meilleure solution actuelle n'a pas été changée depuis la dernière génération et ne peut donc être améliorée en utilisant la recherche locale (un maximum local a été atteint), alors la recherche locale sera appliquée à la solution avec la deuxième valeur la plus élevée de fitness (la plus grande parmi les solutions restantes).

L'algorithme génétique s'arrête quand la limite de temps allouée est atteinte.

5.3.3 Résultats expérimentaux

Tous les tests ont été effectués sur un SUN UltraSPARC IIIi avec 1593 Mhz CPU et 16Gb de RAM. Tous les paramètres des exemples testés ont été générés de manière aléatoire basée sur des distributions uniformes. Pour générer les problèmes à tester, nous avons fait varier les paramètres suivants : demande $d_i \quad i = 1, \ldots, n$, MTTF $1/u_q \quad q = 1, \ldots, m$, MTTR $1/r_q \quad q = 1, \ldots, m$ et le T-ratio. Ce dernier est le ratio de la somme du temps destiné à la fabrication des pièces et du temps de chargement par rapport à l'horizon de planification T_0. Les intervalles des valeurs sont présentés dans le tableau 5.4, où $i = 1, \ldots, n$ et $q = 1, \ldots, m$. L'horizon de planification T_0 est égal à 24 heures.

Remarquons que toutes les combinaisons possibles des intervalles des valeurs de pa-

TABLE 5.4 – Les intervalles pour la génération de données initiales des instances

Paramètre	*Intervalles*	
Demande d_i	$[10; 50]$	$[20; 30]$
MTTR $1/r_q$	$[0,3; 1]$	$[0,5; 0,6]$
MTTF $1/u_q$	$[50; 500]$	$[200; 300]$
T-ratio	$[0,5; 0,6]$	$[0,7; 0,8]$

ramètres donnent 16 familles d'instances. Nous avons généré 10 instances pour chacune des familles, ce qui donne $10 \times 16 = 160$ instances pour chaque taille de problème.

Nous avons fixé le temps total de set-up à 2 heures. La probabilité du niveau de service minimal β est la même pour tous les produits et elle est égale à $0,7$. Les probabilités p_i d'avoir une pièce de bonne qualité pour le produit i, $i = 1, \ldots, n$ sont générées dans l'intervalle $[0,8; 0,95]$.

Nous supposons que le temps total de fabrication (le temps de chargement inclus) sera dans l'intervalle $[\alpha_1 T_0; \alpha_2 T_0]$ (il y a deux possibilités : soit $[\alpha_1; \alpha_2] = [0,5; 0,6]$, soit $[\alpha_1; \alpha_2] = [0,7; 0,8]$). Pour générer les valeurs t_i, $i = 1, \ldots, n$, nous sommes passé par les étapes suivants :

- Générer la valeur du coefficient α dans l'intervalle $[\alpha_1; \alpha_2]$
- Calculer le temps total approximatif de fabrication $T_{fab} = \alpha T_0$
- Calculer la valeur suivante :

$$N_p c = \sum_{i=1}^{n} d_i + n(m-1)$$

  Cette valeur représente la quantité totale de pièces de tous les types dont nous avons besoin, plus un équivalent en pièces du temps de chargement pour tous les produits ;
- Diviser T_{fab} par la quantité des pièces $N_p c$, pour obtenir une valeur approximative du temps de fabrication d'une pièce $t_{piece} = T_{fab}/N_p c$;
- Générer les valeurs t_i, $\quad i = 1, \ldots, n$ dans l'intervalle $[0,5 t_{piece}; 1,5 t_{piece}]$

Pour chaque instance générée, nous avons effectué 5 lancements de l'approche génétique. Les paramètres de l'algorithme génétique ont été choisis de façon empirique en fonction des résultats obtenus dans de nombreux tests. Ce sont les suivants :

- La taille de la population initiale flexible, c.-à-d. dépendant de la taille du problème $TP = 4\sqrt{n}$;
- Le pourcentage des solutions sélectionnées pour le croisement $SP = 50\%$;
- La probabilité de croisement $CP/100 = 0.8$;
- La probabilité de mutation $PM/100 = 0.1$.

Dans tout ce qui suit, nous allons utiliser les notations suivantes pour les approches testées :

- **AG** - l'approche génétique avec la recherche locale et la taille de la population $4\sqrt{n}$;
- **RL** - l'algorithme de recherche locale ;
- **DP** - l'algorithme de programmation dynamique.

Pour les problèmes avec un nombre de lots relativement petit, nous pouvons comparer les solutions obtenues par AG et RL avec les solutions exactes obtenues en utilisant l'algorithme de programmation dynamique de (Dolgui *et al.* (2005)) (qui est présenté dans l'annexe D.1). Les résultats expérimentaux pour les instances avec un nombre de lots n de 4 à 13, sont présentés dans les tableaux 5.5, 5.6 et 5.7.

TABLE 5.5 – Quantité des instances résolues par **DP**

Nombre de lots	*Quantité résolue (sur 160)*	*Quantité non-résolue*	
		limite de temps	*problème de mémoire*
4	160	-	-
5	160	-	-
6	160	-	-
7	143	15	2
8	58	-	102
9	44	1	115
10	32	-	128
11	29	-	131
12	21	1	138
13	15	-	145

Dans le tableau 5.5, nous montrons la quantité d'instances de chaque taille résolues de manière exacte par **DP**. Précisons que nous avons limité le temps de calcul de **DP** à 1 heure (3600 secondes). Nous pouvons constater que :

- l'approche DP est arrivée à résoudre toutes les instances des tailles de 4 à 6 ;
- 15 instances de taille 7 n'ont pas été résolues à cause de la limite de temps ;
- à partir de la taille $n = 8$ l'approche n'arrive plus à résoudre des instances globalement à cause d'un manque de mémoire.

TABLE 5.6 – Comparatif des temps CPU, $n \in [4, 13]$

Nombre de lots	*Temps CPU* ***DP***			*Temps CPU*	
	Min	***Moyen***	***Max***	**AG**	**RL**
4	0,00	0,42	2,08	0,01	0,01
5	0,00	6,40	60,56	0,01	0,01
6	0,00	95,34	724,03	0,02	0,01
7	0,00	816,26 (sur 143)	3591,29	0,02	0,01
8	0,00	600,38 (sur 58)	3506,12	0,03	0,02
9	0,00	347,92 (sur 44)	2239,07	0,03	0,02
10	0,01	269,57 (sur 32)	2001,53	0,05	0,02
11	0,01	232,36 (sur 29)	1828,68	0,06	0,02
12	0,00	177,81 (sur 21)	1198,31	0,06	0,03
13	0,01	295,43 (sur 15)	2547,10	0,08	0,03

Dans le tableau 5.6, nous présentons les temps CPU des approches DP, AG et RL. Pour le **DP**, nous présentons le temps CPU minimal, moyen et maximal pour les instances résolues, dans le but de montrer que le temps de calcul varie beaucoup et dépend énormément des données du problème. Les temps CPU de **AG** et **RL** sont beaucoup moins élevés. Il faut noter aussi que le temps CPU de l'approche génétique a été limité à $0,2n$ secondes, où n est le nombre de lots du problème, c.-à-d. la limite pour $n = 4$ lots est $0,8$ secondes, pour $n = 5$ une seconde, etc. Puisque le temps de calcul pour l'approche AG est connu à l'avance, le temps de **AG** que nous avons mis dans le tableau 5.6 est le temps moyen (pour 5 lancements) lorsque la meilleure solution a été trouvée.

Dans le tableau 5.7, nous affichons les quantités d'instances, qui ont été résolues de manière exacte avec les algorithmes approchés (colonnes 3 et 4), ainsi que la quantité d'ins-

TABLE 5.7 – Qualité des solutions fournies par **AG** et **RL**

Nombre de lots	*Quantité d'inst.*	***Sol. Exacte (quantité)***		***Erreur Relative***	
		AG	**RL**	**AG**	**RL**
4	**160**	15	16	0,328%	0,488%
5	**160**	3	18	0,257%	0,482%
6	**160**	11	31	0,191%	0,390%
7	**143**	115	104	0,211%	0,725%
8	**58**	32	31	0,168%	0,807%
9	**44**	15	12	0,272%	1,066%
10	**32**	1	5	0,167%	0,800%
11	**29**	7	7	0,157%	0,989%
12	**21**	5	2	0,109%	0,789%
13	**15**	2	1	0,094%	0,774%

tances sur laquelle nous avons pu effectuer la comparaison (colonne 2). Malgré le fait que les algorithmes approchés n'aient pas trouvé souvent les solutions exactes, la qualité des solutions obtenues est très bonne, surtout pour l'approche génétique (voir les colonnes 5 et 6, où nous montrons les erreurs relatives (en %) pour les deux approches).

TABLE 5.8 – Temps de calcul moyen de AG et RL

Nombre de Lots	*AG*		*RL*
	Limite	***Meill. Sol***	***Temps***
10	2,000	0,045	0,021
20	3,999	0,185	0,071
30	5,998	0,546	0,153
40	7,996	1,018	0,313
50	9,993	1,936	0,500
60	11,990	2,677	0,764
70	13,988	3,903	1,139
80	15,983	5,081	1,551
90	17,979	7,339	2,222
100	19,973	8,959	2,729
110	21,970	11,370	3,786
120	23,964	13,475	4,431
130	25,953	16,281	5,375
140	27,944	18,988	6,505
150	29,931	21,623	8,112

Maintenant nous allons passer à la comparaison des algorithmes approchés : AG et RL

pour des problèmes de taille plus importante $n \in \{10, 20, \ldots, 150\}$. Le tableau 5.8 présente les temps de calcul moyens de AG et RL. Pour l'approche génétique nous montrons deux valeurs : 1) la valeur moyenne du temps utilisé (en considérant la limite de temps fixée) dans la colonne 2 ; 2) la valeur du temps moyen pour trouver sa meilleure solution (voir colonne 3). Pour le RL c'est simplement la valeur moyenne de temps CPU (colonne 4). Comme nous pouvons le voir, l'approche RL trouve son maximum local plus rapidement que l'approche génétique.

TABLE 5.9 – Valeur de solution moyenne de AG et RL

Nombre de Lots	*Moyenne des solutions*	
	AG	***RL***
10	0,7326	0,7311
20	0,6270	0,6250
30	0,5541	0,5520
40	0,5283	0,52640
50	0,4614	0,4594
60	0,4398	0,4377
70	0,4223	0,4206
80	0,4206	0,4187
90	0,4291	0,4273
100	0,3978	0,3960
110	0,4084	0,4069
120	0,3877	0,3862
130	0,3628	0,3615
140	0,3459	0,3447
150	0,3502	0,3492

Le tableau 5.9 contient les valeurs moyennes des solutions obtenues avec AG et RL. Rappelons qu'une solution pour le problème considéré est une probabilité, i.e. une valeur entre 0 et 1. La valeur moyenne correspondante au génétique est toujours plus élevée que la valeur de la recherche locale, ce qui veut dire que l'optimum local, trouvé par la recherche locale n'est pas une solution optimale du problème. Parmi les 2400 instances testées, la solution moyenne des 5 lancements de AG est supérieure à la valeur de la solution de RL pour 1267 instances et inférieure pour 6 instances uniquement.

Conclusion

Dans ce chapitre nous avons examiné un problème de lotissement et de séquencement sous aléas, qui sont représentés sous forme de lois de distribution. L'objectif était de maximiser le niveau de service global. Le problème P-Prob est NP-difficile. Nous avons utilisé une décomposition du problème initial en trois sous-problèmes : d'énumération, séquencement et dimensionnement de lots. Le problème d'énumération consiste à choisir la meilleure solution en terme de probabilité parmi les n obtenues. Le problème de séquencement est équivalent au problème de VdC. Le lotissement du problème initial est une modification du problème de Sac à Dos, auquel nous nous sommes intéressés dans ce chapitre. L'approche permettant de trouver une solution exacte au problème (proposé par Dolgui *et al.* (2005)) est peu efficace lorsque la taille du problème est supérieure à 7, alors nous avons proposé deux méthodes approchées : Recherche Locale et Algorithme Génétique. Ces méthodes permettent de trouver une solution en un temps satisfaisant. La comparaison de leurs performances est fournie.

Conclusions et Perspectives

Dans ce travail nous avons étudié un problème de lotissement et de séquencement pour une ligne de production imparfaite. Nous avons commencé par présenter la problématique générale de gestion des systèmes de production, puis les hypothèses du problème que nous avons considéré. Nous avons supposé que les machines sur la ligne de production sont imparfaites. Deux types d'aléas sont pris en compte : le rendement aléatoire (à cause des rebuts) et le temps d'exécution aléatoire (à cause des pannes machines). Les temps de changement de série dépendant de la séquence des produits sont également pris en compte.

Nous avons traité d'abord deux versions déterministes du problème pour deux critères différents : minimisation de Makespan sous la condition que la demande générale soit satisfaite (P-Temps) et minimisation de coût total de retard avec un horizon de planification donné (P-Coûts). Nous avons démontré que pour ces deux sous-problèmes, les décisions de lotissement et de séquencement peuvent être prises séparément l'une de l'autre. Ensuite nous avons prouvé que les deux options du problème P-Temps et P-Coûts sont NP-difficiles, si la quantité des pièce défectueuses dans un lot est aléatoire même s'il n'y a aucune panne. Puis, nous avons supposé que le rendement et le temps total de réparation sont des fonctions connues de la quantité de pièces lancées en fabrication. Dans ce cas, le problème de minimisation du Makespan peut être résolu en un temps polynomial, mais le problème de minimisation du coût total de retard reste NP-difficile. Ensuite nous avons proposé quelques méthodes de résolution pour un cas du problème P-Coût dans lequel les fonctions de rendement et de temps total de réparation ont une forme particulière.

Nous avons présenté une méthode de résolution FPTAS qui permet de trouver une solution (séquence des tailles de lots) exacte ainsi qu'une solution approchée avec une borne supérieure d'erreur ε donnée. Nous avons testé cette méthode pour plusieurs familles d'instances et constaté que : 1) pour les petites valeurs d'ε l'algorithme trouve une solution exacte 2) pour $\varepsilon < 0,5$ les erreurs relatives sont inférieures à 3%, ce qui est largement inférieur à la limite ε donnée. Ensuite nous avons présenté deux modèles de programmation linéaire

en variables mixtes. Nous les avons comparés sur un ensemble de tests numériques. Nous avons constaté qu'en général, pour les familles d'instances générées, les modèles linéaires avec CPLEX sont plus rapides, mais il y a un groupe d'instances non négligeable pour lesquels FPTAS trouve une solution optimale plus rapidement. L'avantage de la méthode FPTAS est lié au fait que sa complexité est limitée à $O\left(\frac{n^3}{\varepsilon^2}3^n \log\log(\sum_{i=1}^{n} c_i b_i)\right)$, tandis que la complexité des modèles linéaires est exponentielle. Ensuite, nous avons considéré le problème avec une fonction plus générale représentant le nombre des pièces défectueuses. Nous avons montré que l'approche FPTAS, ainsi que l'un des deux modèles linéaires sont assez flexibles et peuvent s'adapter à cette nouvelle formulation du problème. Les résultats expérimentaux sont également présentés.

Ensuite nous avons présenté une étude du même type de problème, mais pour le cas où des variables aléatoires sont données par leurs lois de distribution et la ligne de production étudiée est un flow-shop de permutation. Nous avons appelé ce problème P-Prob, car l'objectif était de trouver la séquence et les tailles de lot maximisant la probabilité de satisfaire la demande pour tous les produits dans un horizon de planification donné. Dans ce chapitre, nous supposons que le rendement de chaque type de produit suit la loi de Bernoulli avec une probabilité de succès donnée. La modélisation du processus de fonctionnement de la ligne de production (les périodes de bon fonctionnement et les périodes d'inactivité à cause des réparations) a été effectuée en utilisant des processus de renouvellement, où le temps de bon fonctionnement de chaque machine après la dernière réparation (MTTF) ainsi que le temps d'une réparation suivent des lois de distribution exponentielles négatives. Nous avons proposé une approche de décomposition, qui permet de résoudre le problème initial en le décomposant en trois niveaux : une énumération (P-Prob-P(i)), un problème de voyageur du commerce (P-Prob-P1(i)) et une modification du problème de sac à dos (P-Prob-P2(i)).

Nous avons proposé deux méthodes - une recherche locale et un algorithme génétique pour résoudre le sous-problème de sac à dos, qui correspond au problème de lotissement multi-produits. La recherche locale atteint un maximum local assez rapidement, mais l'algorithme génétique est généralement plus performant. Nous avons effectué des expérimentations numériques et les avons largement analysées.

Nous avons développé un modèle de simulation permettant, à partir des données d'entrée de notre problème et d'une solution, d'obtenir par simulation de monte-carlo une approximation de la probabilité du niveau de service. Ce modèle nous a permis de mieux appréhender le comportement du système de production étudié.

L'une des ouverture qu'offre cette thèse pourrait être exploitée d'une part en travaillant

à la substitution de la fonction d'évaluation par le modèle de simulation étudié, il serait alors possible de repousser les limites de nos modèles d'optimisation approchée en prenant en compte, par exemple, des lois de distribution autres qu'exponentielles, ou encore en intégrant d'autres variables aléatoires telles que le temps de fabrication. En changeant des lois de distribution dans ce modèle de simulation, nous pourrions éventuellement mener une étude de sensibilité des résultats par rapport au choix de ces lois.

D'autre part le développement d'une approche de l'optimisation au travers de la simulation permettrait d'étendre le problème considéré au cas où une dépendance entre les pannes et les rebuts existerait.

Nous pourrions également approfondir l'étude des différents résultats et afin de pouvoir envisager les "bonnes pratiques" d'utilisation de ces différents modèles, ainsi que l'étude des divers liens existant entre eux.

Il convient aussi de noter qu'étant donné la grande complexité du problème traité, nous avons proposé des méthodes de décomposition paramétrique. Dans ce contexte nous avons trouvé trois cas où le problème initial pouvait être décomposé en plusieurs parties indépendantes. Dans de futures recherches il serait donc possible d'étudier les cas où cette décomposition n'est pas possible.

Enfin, une autre perspective intéressante serait la recherche d'un algorithme permettant de trouver une borne inférieure au coût d'insatisfaction de la demande pour le problème P-Coût avec une garantie de résultat, qui donnerait la possibilité de créer un FPTAS fortement polynomial.

Les incertitudes et les débouchés liés au fondement même de la recherche en matière de simulation et d'ordonnancement, ouvrent invariablement un nombre grandissant de nouvelles possibilités, de variables intégrable, complexifiant toutes équations. Cette thèse n'y fait pas exception.

Si la finalité de la recherche ne consiste jamais qu'à améliorer et augmenter l'état des connaissances dans un domaine, la seule chose qu'il est concevable d'admettre sans crainte est que cette dernière est perpétuelle.

Annexe A

Les origines du problème traité

Le problème traité est issu de la fabrication automatisé (usine-automate) de circuits imprimés, en particulier d'un atelier fabriquant des patrons conducteurs. Aujourd'hui les circuits imprimés peuvent être trouvés dans tous les appareils électriques et servent de connecteurs entre leur divers composants (par exemple, les condensateurs, les résistances, etc.). Le *patron conducteur* est un réseau des "traces" visible à la surface de plaque (*substrat*, voir figure A.1)

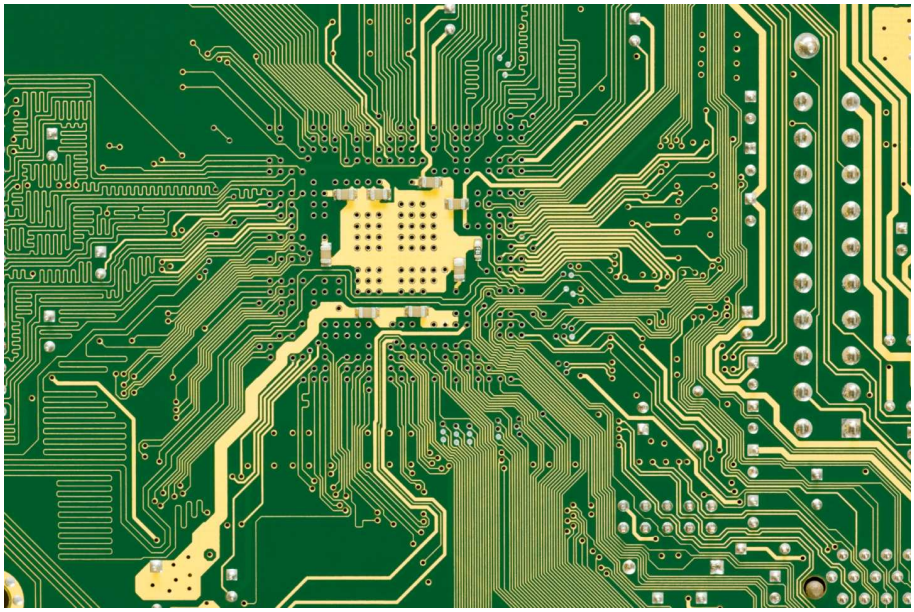


FIGURE A.1 – Un exemple de circuits imprimé

L'atelier considéré fournissait des patrons conducteurs de différents types pour l'atelier de montage des modules électroniques. Considérant le fait que l'usine était complètement

automatisée, l'atelier (comme le reste de l'usine) travaillait la plupart de la journée sans personnel autre que celui de maintenance, de telle sorte que :

- L'équipe du **matin**, composée du personnel de gestion de production était chargée de définir le plan de fabrication pour les 24 heures suivantes ;
- Les équipes du **soir** et de **nuit**, composées uniquement du personnel de maintenance (peu nombreux), ne permettait alors plus de changer le plan de fabrication.

Les circuits étaient stockés dans un système de magasin à charge automatique. Certaines conditions particulières sont à considérer. Cet entrepôt à commande numérique était cher et très limité en nombre de pièces, il fallait donc travailler avec un stock limité de pièces et considérer une durée maximale de rotation d'une journée. Pour ces raisons un système de gestion de production *juste à temps* fut conçu.

Voici quelque paramètres du système : un atelier constitué d'un flow-shop de permutation avec un taux de rebut important (de 20% à 40% en fonction du type de patron conducteur), ainsi que des pannes assez fréquentes de certaines machines (Le MTTF pouvait varier de 100 à 1000 heures ; le MTTR étant d'environ une demi-heure). Tableau A.1 contient plus d'informations techniques.

Table A.1 – Fiche technique pour l'atelier de fabrication des patrons conducteur

Opération	*Temps opératoire, heures*	*MTBF, heures*
Chanfreinage (de face)	0,1	1000
Perçage	0,1	1000
Installation de bondons	0,13	100
Nettoyage de plaques de verre	0,1	1000
Contrôle de plaques	0,25	1000
Formation de topologie à base de laser	2,25	360
Contrôle de topologie et de qualité	0,25	500

Un modèle probabiliste de ce problème a été décrit dans Dolgui (2002). Un modèle de programmation dynamique pour l'optimisation de taille de lots sur la base de ce modèle probabiliste a aussi été proposé dans Dolgui *et al.* (2005).

Annexe B

PCT-FD : performances de FPTAS approché

B.1 PCT-FD : Temps de calcul de FPTAS (sec.) approché

Table B.1 – Temps de calcul de FPTAS (sec.) approché

Nombre de lots	*Epsilon*														
	0	**0,01**	**0,03**	**0,05**	**0,07**	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
10	0,10	0,10	0,09	0,07	0,05	0,04	0,03	0,02	0,02	0,02	0,01	0,01	0,01	0,01	0,01
20	0,45	0,45	0,45	0,37	0,31	0,25	0,15	0,12	0,10	0,09	0,08	0,08	0,07	0,07	0,07
30	1,48	1,48	1,48	1,34	1,07	0,80	0,50	0,38	0,32	0,29	0,26	0,24	0,23	0,22	0,22
40	2,66	2,66	2,66	2,48	2,35	1,87	1,14	0,87	0,70	0,61	0,56	0,52	0,49	0,47	0,46
50	4,51	4,51	4,51	4,43	3,53	3,03	2,12	1,50	1,27	1,10	1,00	0,94	0,90	0,86	0,83
60	7,15	7,15	7,15	7,15	6,05	4,73	3,27	2,54	2,16	1,93	1,75	1,63	1,56	1,51	1,46
70	11,01	11,01	11,01	11,01	10,77	8,08	5,10	4,03	3,27	2,92	2,67	2,47	2,35	2,26	2,20
80	18,11	18,11	18,11	15,62	13,70	10,99	7,11	5,61	4,60	4,08	3,80	3,54	3,36	3,23	3,14
90	24,91	24,91	24,91	23,17	20,24	17,32	11,54	8,50	7,06	6,11	5,60	5,28	4,96	4,74	4,58
100	53,52	53,52	53,52	51,84	34,62	25,32	16,21	11,48	9,74	8,32	7,56	7,09	6,76	6,44	6,21
110	40,53	40,53	40,53	40,53	33,83	28,91	20,61	14,93	12,50	10,93	9,94	9,34	8,95	8,62	8,31
120	66,40	66,40	66,40	66,40	54,46	40,15	26,89	20,35	16,81	14,91	13,40	12,50	11,91	11,50	11,12
130	87,22	87,22	87,22	81,20	71,37	51,73	31,76	24,42	19,95	17,88	16,23	15,21	14,54	14,08	13,75
140	90,17	90,17	90,17	90,17	88,00	64,53	39,99	31,48	25,55	22,81	20,86	19,37	18,40	17,74	17,26
150	109,05	109,05	109,05	103,58	100,39	76,65	48,05	38,54	30,79	27,22	25,15	23,23	21,99	21,15	20,53

B.2 PCT-FD : Les erreurs relatives des solution de FPTAS approché

Table B.2 – Les erreurs relatives des solution de FPTAS approché

Nombre de lots	*Epsilon*													
	0,01	**0,03**	**0,05**	**0,07**	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
10	0,00%	0,00%	0,01%	0,03%	0,06%	0,59%	1,11%	2,57%	3,62%	4,45%	7,45%	10,34%	11,31%	13,78%
20	0,00%	0,00%	0,00%	0,02%	0,07%	0,50%	0,81%	2,11%	3,01%	4,22%	7,37%	10,21%	10,89%	13,93%
30	0,00%	0,00%	0,01%	0,02%	0,05%	0,28%	0,83%	1,32%	3,35%	4,43%	4,98%	7,39%	8,69%	13,75%
40	0,00%	0,00%	0,00%	0,00%	0,03%	0,31%	0,49%	1,69%	2,57%	3,06%	6,39%	8,73%	9,50%	12,05%
50	0,00%	0,00%	0,01%	0,02%	0,06%	0,52%	0,90%	2,34%	2,99%	4,45%	4,90%	9,86%	11,18%	14,08%
60	0,00%	0,00%	0,00%	0,02%	0,04%	0,20%	0,71%	1,12%	3,41%	4,06%	4,23%	6,39%	7,51%	14,29%
70	0,00%	0,00%	0,00%	0,01%	0,02%	0,17%	0,99%	1,10%	1,83%	4,69%	6,04%	6,34%	8,88%	9,77%
80	0,00%	0,00%	0,01%	0,02%	0,06%	0,31%	0,55%	1,71%	2,27%	3,00%	6,75%	9,11%	9,65%	12,07%
90	0,00%	0,00%	0,01%	0,01%	0,04%	0,44%	0,61%	2,10%	2,42%	3,61%	4,08%	9,38%	11,82%	12,22%
100	0,00%	0,00%	0,02%	0,03%	0,09%	0,58%	0,86%	2,46%	3,19%	4,46%	5,51%	10,17%	11,92%	15,15%
110	0,00%	0,00%	0,00%	0,01%	0,02%	0,14%	0,72%	1,01%	3,04%	4,00%	5,69%	6,20%	11,83%	13,30%
120	0,00%	0,00%	0,00%	0,02%	0,06%	0,21%	0,84%	1,18%	3,26%	4,24%	4,43%	6,73%	8,38%	14,68%
130	0,00%	0,00%	0,00%	0,03%	0,05%	0,27%	0,97%	1,55%	1,72%	4,92%	6,28%	7,98%	9,57%	10,31%
140	0,00%	0,00%	0,00%	0,02%	0,03%	0,17%	0,89%	1,03%	1,74%	4,39%	5,65%	5,98%	8,59%	9,51%
150	0,00%	0,00%	0,00%	0,00%	0,04%	0,29%	1,34%	1,45%	2,11%	5,78%	6,17%	7,69%	10,50%	11,08%

Annexe C

PCT-$\alpha\beta$: performances de FPTAS approché

C.1 PCT-$\alpha\beta$: Temps de calcul moyen de FPTAS approché

TABLE C.1 – Temps de calcul moyen de FPTAS pour $\varepsilon \in \{0;\ 0,01;\ 0,05;\ 0,1;\ 0,3;\ 0,5;\ 0,7;\ 1\}$

Nombre de lots	*Valeur de ε*							
	0	**0,01**	**0,05**	**0,1**	**0,3**	**0,5**	**0,7**	**1**
10	0,11	0,11	0,06	0,04	0,02	0,02	0,01	0,01
20	0,45	0,45	0,35	0,24	0,12	0,09	0,08	0,07
30	1,49	1,49	1,13	0,75	0,38	0,29	0,25	0,22
40	3,22	3,22	2,61	1,68	0,87	0,63	0,55	0,48
50	6,23	6,23	5,13	3,46	1,55	1,13	0,98	0,86
60	9,70	9,70	8,51	5,59	2,72	2,02	1,69	1,51
70	10,91	10,91	10,91	7,92	4,09	3,00	2,56	2,28
80	17,63	17,63	16,91	13,03	6,41	4,47	3,81	3,33
90	33,92	33,92	29,78	17,93	8,53	6,27	5,47	4,75
100	34,24	34,24	33,58	22,78	11,61	8,59	7,35	6,44
110	56,57	56,57	49,51	31,53	15,22	11,46	9,80	8,71
120	55,12	55,12	55,12	39,95	20,98	15,36	12,81	11,37
130	67,79	67,79	67,79	46,77	24,29	18,26	15,46	13,93
140	90,60	90,60	90,60	64,14	31,63	22,95	19,53	17,44
150	117,51	117,51	111,62	82,93	40,23	27,96	23,77	20,99
Moyenne	33,70	33,70	32,24	22,58	11,24	8,17	6,94	6,16

C.2 PCT-$\alpha\beta$: Erreurs relatives des solutions approchées de FPTAS

Table C.2 – Erreurs relatives des solutions approchées de FPTAS

Nombre de Lots	*Valeur de ε*						
	0,01	**0,05**	**0,1**	**0,3**	**0,5**	**0,7**	**1**
10	0,00%	0,013%	0,072%	1,295%	4,370%	9,021%	16,112%
20	0,00%	0,021%	0,054%	0,848%	3,196%	7,252%	13,100%
30	0,00%	0,003%	0,070%	1,029%	4,221%	6,017%	15,628%
40	0,00%	0,006%	0,061%	0,609%	2,564%	6,918%	12,771%
50	0,00%	0,016%	0,108%	1,110%	3,559%	5,630%	14,405%
60	0,00%	0,000%	0,045%	0,939%	3,648%	5,235%	14,641%
70	0,00%	0,000%	0,022%	0,961%	1,678%	6,265%	9,936%
80	0,00%	0,001%	0,034%	0,384%	2,078%	5,989%	11,102%
90	0,00%	0,006%	0,055%	0,701%	2,586%	4,719%	13,039%
100	0,00%	0,006%	0,062%	0,786%	2,757%	4,727%	14,009%
110	0,00%	0,005%	0,041%	0,835%	3,187%	6,034%	13,630%
120	0,00%	0,000%	0,024%	0,736%	3,169%	4,089%	13,303%
130	0,00%	0,000%	0,038%	0,753%	1,193%	5,278%	8,753%
140	0,00%	0,000%	0,021%	0,876%	1,698%	5,812%	10,093%
150	0,00%	0,001%	0,030%	1,177%	2,063%	5,699%	10,465%
Moyenne	0,00%	0,005%	0,049%	0,869%	2,798%	5,912%	12,732%

Annexe D

Problème P-Prob-P2(i)

D.1 Transformation de P-Prob-P2(i) en problème de Sac à Dos

Rappelons le problème à résoudre :

Problème P-Prob-**P2(i)** : Trouver la combinaison des tailles des lots $x^* = (x_1, x_2, \ldots, x_n)$, qui va minimiser la probabilité suivante :

$$P\Big(x_j^b \ \geq \ d_j \ \mid \ (\pi^{*i}, x)\Big) \ = \ \prod_{j=1}^{n-1} P(x_j^b \ \geq \ d_j | x_j) \ \times \ P\Big(x_n^b \ \geq \ d_n \ \mid \ x_n, T^{nr}(\pi^{*i}, x)\Big) \quad \text{(D.1)}$$

Nous allons montrer que ce problème peut être transformé en une modification du problème de Sac à Dos. Auparavant nous avons utilisé les notations suivantes :

$$t = T_0 - S(\pi^*) - (m-1)\sum_{i=1}^{n} t_i \quad \text{et} \quad T^{nr}(x) = t - \sum_{i=1}^{n-1} t_i x_i$$

Nous avons besoin des notations suivantes pour introduire le modèle mathématique et l'approche. Soit

- w_i^{min} et w_i^{max} les temps minimal et maximal de fabrication d'un lot de produit i, $i = 1, \ldots, n$, qui peuvent être calculés de la manière suivante :

$$w_i^{min} = t_i x_i^{min} \quad \text{et} \quad w_i^{max} = t_i x_i^{upp} \qquad \text{(D.2)}$$

où x_i^{min} est la taille de lot i minimale (voir (5.23)), et x_i^{upp} est la taille de lot i maximale, $i = 1, \ldots, n$ (voir (5.24) et (5.25)) ;

- V_i^{min} le temps minimale possible, nécessaire pour fabriquer les pièces de i premiers lots. C'est évident que ce temps minimal correspond au traitement des tailles de lots $(x_1^{min}, \ldots, x_i^{min})$ et peut être exprimé par l'équation suivante :

$$V_i^{min} = \sum_{j=1}^{i} t_j x_j^{min} = \sum_{j=1}^{i} w_j^{min},\ i = 1, \ldots, n-1 \tag{D.3}$$

- V_i^{max} le temps maximale possible, que nous pouvons affecter au traitement des i premiers lots. V_i^{max} est calculé comme suit :

$$V_i^{max} = \min\Big\{t - \sum_{j=i+1}^{n} w_j^{min},\ \sum_{j=1}^{i} w_j^{max}\Big\},\ i = 1, \ldots, n-1 \tag{D.4}$$

où $t - \sum_{j=i+1}^{n} w_j^{min}$ est le temps qui reste de T_0 après la soustraction du temps total de set-up, du temps de chargement et du temps minimale pour fabriquer les $n-i$ derniers lots, $i = 1, \ldots, n-1$.

En utilisant les notations ci-dessus, le problème P-Prob-P2(i) se transforme en problème de recherche d'un vecteur $w = (w_1, w_2, \ldots, w_n)$ des temps affectés à la fabrication de chaque lot, tel que la probabilité suivante soit maximisée :

$$P(w) = \prod_{j=1}^{n-1} P\left(\left(\frac{w_j}{t_j}\right)^b \geq d_j \mid \frac{w_j}{t_j}\right) \times P\left(\left(\frac{w_n}{t_n}\right)^b \geq d_n \mid x_n^{upp}, T^{nr}(w)\right) \tag{D.5}$$

où $w_i \in \{w_i^{min}, w_i^{min}+1, \ldots, w_i^{max}\}$. Ce dernier problème est une modification du problème de *Sac à Dos*. Si le vecteur $w^* = (w_1^*, w_2^*, \ldots, w_n^*)$ est la solution optimale pour ce problème, alors le vecteur $x^* = (x_1^*, x_2^*, \ldots, x_n^*) = \left(\frac{w_1^*}{t_1}, \frac{w_2^*}{t_2}, \ldots, \frac{w_n^*}{t_n}\right)$ est la solution optimale pour le problème initial P-Prob-P2(i).

Soit $V_i = \{V_i^1, V_i^2, \ldots, V_i^z\}$ l'ensemble **de toutes les valeurs** de temps possibles pour fabriquer les premiers i lots. L'approche par la programmation dynamique, proposée dans Dolgui *et al.* (2005), est présentée dans l'algorithme D.1, où la probabilité $P(w)$ est la probabilité maximale pour satisfaire la demande générale. Les tailles de lots peuvent être trouvées par retour en arrière.

D.2 Une procédure de programmation dynamique

Algorithme D.1: L'algorithme DP pour le problème P-PROB-P2(i)

$j \leftarrow 1$
$V_1 \in [V_1^{min}, V_1^{max}] \leftarrow \{w_1^{min}, w_1^{min} + t_1, \ldots, w_1^{max}\}$
pour chaque $(i = 1;\ i \leq |V_1|;\ i \leftarrow i + 1)$ **faire**
  Calculer $H_1(V_1^i) = P\left(\left(\frac{V_1^i}{t_1}\right)^b \geq d_1 \mid \frac{V_1^i}{t_1}\right)$
fin

Récursion :
tant que $(j < n)$ **faire**
  $j \leftarrow j + 1$
  $V_j = \{\emptyset\}$
  pour chaque $\left(V_{j-1}^k;\ k = 1;\ k \leq |V_{j-1}|;\ k \leftarrow k + 1\right)$ **faire**
    pour chaque $\left(y = w_j^{min};\ y \leq w_j^{upp};\ y \leftarrow y + t_j\right)$ **faire**
      Calculer $v = V_{j-1}^k + y$
      si $v \in \left[V_j^{min}; V_j^{max}\right]$ **alors**
        Ajouter v *dans l'ensemble* V_j
      fin
    fin
  fin
  pour chaque $(i = 1;\ i \leq |V_j| \quad i \leftarrow i + 1)$ **faire**
    Calculer
    $H_j(V_j^i) = \max\left\{H_{j-1}(V_{j-1}^k) \times P\left(\left(\frac{V_j^i - V_{j-1}^k}{t_j}\right)^b \geq d_1 \mid \frac{V_j^i - V_{j-1}^k}{t_j}\right)\right\}$
  fin
fin
Calculer la probabilité
$P(w) = \max\left\{H_{n-1}(V_{n-1}^k) \times P\left(x_n^b \geq d_n \mid x_n^{upp}, t - V_{n-1}^k\right) \mid V_{n-1}^k \in V_{n-1}\right\}$

Bibliographie

L. Abdel-Malek, R. Montanari et D. Meneghetti : The capacitated newsboy problem with random yield : The Gardener Problem. *International Journal of Production Economics*, 115(1):113–127, 2008.

L. Abdel-Malek, R. Montanari et L. Morales : Exact, approximate, and generic iterative models for the multi-product newsboy problem with budget constraint. *International Journal of Production Economics*, 91(2):189–198, 2004.

L. Abdel-Malek et R. Montanari : An analysis of the multi-product newsboy problem with a budget constraint. *International Journal of Production Economics*, 97(3):296–307, 2005.

I. Adiri, J. Bruno, E. Frostig et A. Rinnooy Kan : Single machine flow-time scheduling with a single breakdown. *Acta Informatica*, 26(7):679–696, 1989.

S. Agnihothri, J. Lee et J. Kim : Lot sizing with random yields and tardiness costs. *Computers and Operations Research*, 27(5):437–459, 2000.

D. Alcaide, A. Rodriguez-Gonzalez et J. Sicilia : An approach to solve the minimum expected makespan flow-shop problem subject to breakdowns. *European Journal of Operational Research*, 140(2):384–398, 2002.

A. Allahverdi : The two-and m-machine flowshop scheduling problems with bicriteria of makespan and mean flowtime. *European Journal of Operational Research*, 147(2):373–396, 2003.

A. Allahverdi, J. Gupta et T. Aldowaisan : A review of scheduling research involving setup considerations. *Omega*, 27(2):219–239, 1999.

A. Allahverdi, C. Ng, T. Cheng et M. Kovalyov : A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3):985–1032, 2008.

A. Allahverdi et M. Savsar : Stochastic proportionate flowshop scheduling with setups. *Computers and Industrial Engineering*, 39(3-4):357–369, 2001.

A. Andersson : Capacity study of statistical multiplexing for IP telephony. *SICS Research Report*, 2000.

S. Anily : Single-machine lot-sizing with uniform yields and rigid demands : robustness of the optimal solution. *IIE Transactions*, 27(5):634–637, 1995.

M. Asano et H. Ohta : Single machine scheduling to meet due times under shutdown constraints. *International Journal of Production Economics*, 60:537–547, 1999.

S. Axsäter : *Inventory control*. Springer Verlag, 2006.

T. Bagchi, J. Gupta et C. Sriskandarajah : A review of TSP based approaches for flowshop scheduling. *European Journal of Operational Research*, 169(3):816–854, 2006.

K. Baker et G. Scudder : Sequencing with earliness and tardiness penalties : a review. *Operations Research*, 38(1):22–36, 1990.

N. Balakrishnan, J. Kanet et V. Sridharan : Early/tardy scheduling with sequence dependent setups on uniform parallel machines. *Computers and Operations Research*, 26 (2):127–141, 1999.

P. Beraldi, G. Ghiani, E. Guerriero et A. Grieco : Scenario-based planning for lot-sizing and scheduling with uncertain processing times. *International Journal of Production Economics*, 101(1):140–149, 2006.

G. Bitran et H. Yanasse : Computational complexity of the capacitated lot size problem. *Management Science*, 28(10):1174–1186, 1982.

P. Brucker et M. Kovalyov : Single machine batch scheduling to minimize the weighted number of late jobs. *Mathematical Methods of Operations Research*, 43(1):1–8, 1996.

U. Buscher et G. Lindner : Optimizing a production system with rework and equal sized batch shipments. *Computers and Operations Research*, 34(2):515–535, 2007.

L. Cárdenas-Barrón : The economic production quantity (EPQ) with shortage derived algebraically. *International Journal of Production Economics*, 70(3):289–292, 2001.

P. Caricato, A. Grieco et D. Serino : TSP-based scheduling in a batch-wise hybrid flow-shop. *Robotics and Computer-Integrated Manufacturing*, 23(2):234–241, 2007.

S. Chang, J. Chuang et H. Chen : Short comments on technical note - The EOQ and EPQ models with shortages derived without derivatives. *International Journal of Production Economics*, 97(2):241–243, 2005.

D. Chatfield : The economic lot scheduling problem : A pure genetic search approach. *Computers and Operations Research*, 34(10):2865–2881, 2007.

J. Chen et T. Wu : Total tardiness minimization on unrelated parallel machine scheduling with auxiliary equipment constraints. *Omega*, 34(1):81–89, 2006.

W. Chen et J. Thizy : Analysis of relaxations for the multi-item capacitated lot-sizing problem. *Annals of Operations Research*, 26(1):29–72, 1990.

T. Cheng, A. Janiak et M. Kovalyov : Single machine batch scheduling with resource dependent setup and processing times. *European Journal of Operational Research*, 135 (1):177–183, 2001.

T. Cheng et G. Wang : An improved heuristic for two-machine flowshop scheduling with an availability constraint. *Operations Research Letters*, 26(5):223–229, 2000.

T. Cheng, J. Gupta et G. Wang : A review of flowshop scheduling Research with setup times. *Production and Operations Management*, 9(3):262–282, 2000.

T. Cheng et M. Kovalyov : Single machine batch scheduling with deadlines and resource dependent processing times. *Operations Research Letters*, 17(5):243–249, 1995.

T. Cheng et M. Kovalyov : Single machine batch scheduling with sequential job processing. *IIE Transactions*, 33(5):413–420, 2001.

T. Cheng et M. Kovalyov : An unconstrained optimization problem is NP-hard given an oracle representation of its objective function : a technical note. *Computers and Operations Research*, 29(14):2087–2091, 2002.

S. Chiu, C. Ting et Y. Chiu : Optimal production lot sizing with rework, scrap rate, and service level constraint. *Mathematical and Computer Modelling*, 46(3-4):535–549, 2007.

M. Chou, M. Queyranne et D. Simchi-Levi : The asymptotic performance ratio of an on-line algorithm for uniform parallel machine scheduling with release dates. *Mathematical Programming*, 106(1):137–157, 2006.

S. Chubanov, M. Kovalyov et E. Pesch : An FPTAS for a single-item capacitated economic lot-sizing problem with monotone cost structure. *Mathematical Programming*, 106(3):453–466, 2006.

E. Cox : *Fuzzy modeling and genetic algorithms for data mining and exploration.* Morgan Kaufmann Pub, 2005.

N. Dellaert, J. Jeunet et N. Jonard : A genetic algorithm to solve the general multi-level lot-sizing problem with time-varying costs. *International Journal of Production Economics*, 68(3):241–257, 2000.

A. Dolgui : Performance analysis model for systems described by renewal process. *Engineering Simulation (Electronic Modeling)*, 24(2):3–11, 2002.

A. Dolgui, G. Levin et M. Louly : Decomposition approach for a problem of lot-sizing and sequencing under uncertainties. *International Journal of Computer Integrated Manufacturing*, 18(5):376–385, 2005.

A. Dolgui et M. Ould-Louly : A model for supply planning under lead time uncertainty. *International Journal of Production Economics*, 78(2):145–152, 2002.

A. Dolgui et C. Prodhon : Supply planning under uncertainties in MRP environments : a state of the art. *Annual Reviews in Control*, 31(2):269–279, 2007.

A. Drexl et A. Kimms : Lot sizing and scheduling–Survey and extensions. *European Journal of Operational Research*, 99(2):221–235, 1997.

S. Elmaghraby : The economic lot scheduling problem (ELSP) : review and extensions. *Management Science*, 24(6):587–598, 1978.

H. Emmons et K. Mathur : Lot sizing in a no-wait flow shop. *Operations Research Letters*, 17(4):159–164, 1995.

S. Erlebacher : Optimal and heuristic solutions for the multi-item newsvendor problem with a single capacity constraint. *Production and Operations Management*, 9(3):303–318, 2000.

A. Eroglu et G. Ozdemir : An economic order quantity model with defective items and shortages. *International Journal of Production Economics*, 106(2):544–549, 2007.

M. Espinouse, P. Formanowicz et B. Penz : Minimizing the makespan in the two-machine no-wait flow-shop with limited machine availability. *Computers and Industrial Engineering*, 37(1-2):497–500, 1999.

J. Evans : An efficient implementation of the Wagner-Whitin algorithm for dynamic lot-sizing. *Journal of Operations Management*, 5(2):229–235, 1985.

S. Flapper, J. Fransoo, R. Broekmeulen et K. Inderfurth : Planning and control of rework in the process industries : a review. *Production Planning and Control*, 13(1):26–34, 2002.

J. Fordyce et F. Webster : Wagner-Whitin algorithm made simple. *Production and Inventory Management*, 25(2):21–30, 1984.

L. Gaafar : Applying genetic algorithms to dynamic lot sizing with batch ordering. *Computers and Industrial Engineering*, 51(3):433–444, 2006.

M. Garey et D. Johnson : *Computers and intractability. A guide to the theory of NP-completeness. A Series of Books in the Mathematical Sciences.* WH Freeman and Company, San Francisco, Calif, 1979.

P. Geng et R. Vickson : Two heuristics for the economic lot scheduling problem : an experimental study. *Naval Research Logistics*, 35(4):605–617, 1988.

Y. Gerchak, R. Vickson et M. Parlar : Periodic review production models with variable yield and uncertain demand. *IIE Transactions*, 20(2):144–150, 1988.

Y. Gerchak, Y. Wang et C. Yano : Lot sizing in assembly systems with random component yields. *IIE Transactions*, 26(2):19–24, 1994.

V. Giard : *Gestion de la production et des flux (avec CD-Rom, 3.).* Economica, 3e (25 avril 2003) édn, 2003. ISBN 978-2717844986.

H. Gitlow : *Tools and Methods for the Improvement of Quality.* CRC, 1989.

D. Goldberg : *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley, 1989.

V. Gordon, J. Proth et C. Chu : A survey of the state-of-the-art of common due date assignment and scheduling research. *European Journal of Operational Research*, 139(1):1–25, 2002.

M. Gourgand, N. Grangeon et S. Norre : A contribution to the stochastic flow shop scheduling problem. *European Journal of Operational Research*, 151(2):415–433, 2003.

A. Grosfeld-Nir et Y. Gerchak : Multiple lotsizing with random common-cause yield and rigid demand. *Operations Research Letters*, 9(6):383–387, 1990.

A. Grosfeld-Nir et Y. Gerchak : Production to order with random yields : single-stage multiple lot-sizing. *IIE Transactions*, 28(8):669–676, 1996.

A. Grosfeld-Nir et Y. Gerchak : Multiple lotsizing in production to order with random yields : Review of recent advances. *Annals of Operations Research*, 126(1):43–69, 2004.

R. Grubbstrom : Material requirements planning and manufacturing resource planning. *International Encyclopedia of Business and Management*, 1996.

R. Grubbström et A. Erdem : The EOQ with backlogging derived without derivatives. *International Journal of Production Economics*, 59(1-3):529–530, 1999.

A. Guinet : Scheduling independent jobs on uniform parallel machines to minimize tardiness criteria. *Journal of Intelligent Manufacturing*, 6(2):95–103, 1995.

S. Gupta et J. Smith : Algorithms for single machine total tardiness scheduling with sequence dependent setups. *European Journal of Operational Research*, 175(2):722–739, 2006.

H. Gurnani et Y. Gerchak : Coordination in decentralized assembly systems with uncertain component yields. *European Journal of Operational Research*, 176(3):1559–1576, 2007.

G. Gutin et A. Punnen : *The traveling salesman problem and its variations.* Kluwer Academic, 2002.

S. Guu et F. Liou : An Algorithm for the Multiple Lot Sizing Problem with Rigid Demand and Interrupted Geometric Yield. *Journal of Mathematical Analysis and Applications*, 234 (2):567–579, 1999.

S. Guu et A. Zhang : The finite multiple lot sizing problem with interrupted geometric yield and holding costs. *European Journal of Operational Research*, 145(3):635–644, 2003.

R. Haji, A. Haji, M. Sajadifar et S. Zolfaghari : Lot sizing with non-zero setup times for rework. *Journal of Systems Science and Systems Engineering*, 17(2):230–240, 2008.

M. Held et R. Karp : A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics*, 10(1):196–210, 1962.

S. G. Hernandez, W. : Genetic algorithms in lot sizing decisions. *IEEE Transactions on Evolutionary Computation*, (3):2280–2286, 1999.

J. Holland : *Adaptation in natural and artificial systems.* Ann Arbor MI : University of Michigan Press, 1975.

J. Hoogeveen et S. Van de Velde : A branch-and-bound algorithm for single-machine earliness-tardiness scheduling with idle time. *INFORMS Journal on Computing*, 8:402–412, 1996.

W. Hsu : On the general feasibility test of scheduling lot sizes for several products on one machine. *Management Science*, 29(1):93–105, 1983.

J. Hu, C. Munson et E. Silver : A modified Silver-Meal heuristic for dynamic lot sizing under incremental quantity discounts. *Journal of the Operational Research Society*, 55 (6):671–673, 2004.

K. Inderfurth, A. Janiak, M. Kovalyov et F. Werner : Batching work and rework processes with limited deterioration of reworkables. *Computers and Operations Research*, 33(6):1595–1605, 2006.

K. Inderfurth, M. Kovalyov, C. Ng et F. Werner : Cost minimizing scheduling of work and rework processes on a single facility under deterioration of reworkables. *International Journal of Production Economics*, 105(2):345–356, 2007.

K. Inderfurth, G. Lindner et N. Rachaniotis : Lot sizing in a production system with rework and product deterioration. *International Journal of Production Research*, 43 (7):1355–1374, 2005.

J. Jeunet et N. Jonard : Measuring the performance of lot-sizing techniques in uncertain environments. *International Journal of Production Economics*, 64(1-3):197–208, 2000.

I. Kacem : Fully polynomial time approximation scheme for the total weighted tardiness minimization with a common due date. *Discrete Applied Mathematics*, 158(9):1035–1040, 2010.

I. Kacem et A. Mahjoub : Fully polynomial time approximation scheme for the weighted flow-time minimization on a single machine with a fixed non-availability interval. *Computers and Industrial Engineering*, 56(4):1708–1712, 2009.

P. Kalczynski et J. Kamburowski : On no-wait and no-idle flow shops with makespan criterion. *European Journal of Operational Research*, 178(3):677–685, 2007.

B. Karimi, S. Fatemi Ghomi et J. Wilson : The capacitated lot sizing problem : a review of models and algorithms. *Omega*, 31(5):365–378, 2003.

N. Kasap, H. Aytug et A. Paul : Minimizing makespan on a single machine subject to random breakdowns. *Operations Research Letters*, 34(1):29–36, 2006.

S. Kedad-Sidhoum et F. Sourd : Fast neighborhood search for the single machine earliness-tardiness scheduling problem. *Computers & Operations Research*, 37(8):1464–1471, 2010.

M. Khouja : The single-period (news-vendor) problem : literature review and suggestions for future research. *Omega*, 27(5):537–553, 1999.

M. Khouja, Z. Michalewicz et M. Wilmot : The use of genetic algorithms to solve the economic lot size scheduling problem. *European Journal of Operational Research*, 110 (3):509–524, 1998.

D. Kim, K. Kim, W. Jang et F. Frank Chen : Unrelated parallel machine scheduling with setup times using simulated annealing. *Robotics and Computer-Integrated Manufacturing*, 18(3-4):223–231, 2002.

Y. Kim et C. Yano : Minimizing mean tardiness and earliness in single-machine scheduling problems with unequal due dates. *Naval Research Logistics*, 41(7):913–933, 1994.

A. Kovács, K. Brown et S. Tarim : An efficient MIP model for the capacitated lot-sizing and scheduling problem with sequence-dependent setups. *International Journal of Production Economics*, 118(1):282–291, 2009.

M. Kovalyov : A rounding technique to construct approximation algorithms for knapsack and partition-type problems. *Applied Mathematics and Computer Science*, 6:789–802, 1996.

H. Lau et A. Hing-Ling Lau : The multi-product multi-constraint newsboy problem : Applications, formulation and solution. *Journal of Operations Management*, 13(2):153–162, 1995.

R. Leardi : Genetic algorithms in chemometrics and chemistry : a review. *Journal of Chemometrics*, 15(7):559–569, 2001.

C. Lee : Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint. *Operations Research Letters*, 20(3):129–139, 1997.

C. Lee : Two-machine flowshop scheduling with availability constraints. *European Journal of Operational Research*, 114(2):420–429, 1999.

H. Lee et C. Billington : Material management in decentralized supply chains. *Operations Research*, 41(5):835–847, 1993.

I. Lee, R. Sikora et M. Shaw : A genetic algorithm-based approach to flexible flow-line scheduling with variable lot sizes. *IEEE Transactions on Systems, Man, and Cybernetics. Part B, Cybernetics*, 27(1):36, 1997.

W. Lee, C. Wu et P. Chen : A simulated annealing approach to makespan minimization on identical parallel machines. *The International Journal of Advanced Manufacturing Technology*, 31(3):328–334, 2006.

Q. Li, H. Xu et S. Zheng : Periodic-review inventory systems with random yield and demand : Bounds and heuristics. *IIE Transactions*, 40(4):434–444, 2008.

G. Liao, Y. Chen et S. Sheu : Optimal economic production quantity policy for imperfect process with imperfect repair and maintenance. *European Journal of Operational Research*, 195(2):348–357, 2009.

G. Lin et D. Gong : On a production-inventory system of deteriorating items subject to random machine breakdowns with a fixed repair time. *Mathematical and Computer Modelling*, 43(7-8):920–932, 2006.

M. Louly et A. Dolgui : Calculating safety stocks for assembly systems with random component procurement lead times : A branch and bound algorithm. *European Journal of Operational Research*, 199(3):723–731, 2009.

B. Maddah et M. Jaber : Economic order quantity for items with imperfect quality : Revisited. *International Journal of Production Economics*, 112(2):808–815, 2008.

S. Melouk, P. Damodaran et P. Chang : Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing. *International Journal of Production Economics*, 87(2):141–147, 2004.

L. Min et W. Cheng : A genetic algorithm for minimizing the makespan in the case of scheduling identical parallel machines. *Artificial Intelligence in Engineering*, 13(4):399–403, 1999.

S. Minner : A note on how to compute economic order quantities without derivatives by cost comparisons. *International Journal of Production Economics*, 105(1):293–296, 2007.

A. Molinder : Joint optimization of lot-sizes, safety stocks and safety lead times in an MRP system. *International Journal of Production Research*, 35(4):983–994, 1997.

I. Moon, E. Silver et S. Choi : Hybrid genetic algorithm for the economic lot-scheduling problem. *International Journal of Production Research*, 40(4):809–824, 2002.

P. Moscato : On evolution, search, optimization, genetic algorithms and martial arts : Towards memetic algorithms. *Caltech Concurrent Computation Program, C3P Report*, 826:1989, 1989.

M. Narro Lopez et B. Kingsman : The economic lot scheduling problem : theory and practice. *International Journal of Production Economics*, 23(1-3):147–164, 1991.

M. Nascimento, M. Resende et F. Toledo : GRASP heuristic with path-relinking for the multi-plant capacitated lot sizing problem. *European Journal of Operational Research*, 200(3):747–754, 2010.

R. Nelson : *Probability, stochastic processes, and queueing theory : the mathematics of computer performance modelling.* Springer, 1995.

C. D. Ng, T. Cheng et M. Kovalyov : Single machine batch scheduling with jointly compressible setup and processing times. *European Journal of Operational Research*, 153 (1):211–219, 2004.

M. Omar et M. Deris : The Silver-Meal Heuristic Method for Deterministic Time-Varying Demand. *Matematika*, 17(1):7–14, 2001.

J. Ouenniche et F. Boctor : The multi-product, economic lot-sizing problem in flow shops : the powers-of-two heuristic. *Computers and Operations Research*, 28(12):1165–1182, 2001.

Y. Pochet et L. Wolsey : *Production planning by mixed integer programming.* Springer Verlag, 2006.

J. Potamianos et A. Orman : An interactive dynamic inventory-production control system. *Journal of the Operational Research Society*, 47(8):1017–1028, 1996.

J. Potamianos, A. Orman et A. Shahani : Modelling for a dynamic inventory-production control system. *European Journal of Operational Research*, 96(3):645–658, 1997.

C. Potts et M. Kovalyov : Scheduling with batching : a review. *European Journal of Operational Research*, 120(2):228–249, 2000.

C. Potts et L. Van Wassenhove : Integrating scheduling with batching and lot-sizing : a review of algorithms and complexity. *Journal of the Operational Research Society*, 43 (5):395–406, 1992.

R. Ronald, G. Yang et P. Chu : Technical note : the EOQ and EPQ models with shortages derived without derivatives. *International Journal of Production Economics*, 92(2):197–200, 2004.

N. Saadani, A. Guinet et M. Moalla : A travelling salesman approach to solve the F/no-idle/Cmax problem. *European Journal of Operational Research*, 161(1):11–20, 2005.

Y. Said : *Handbook of Statistics : Data mining and data visualization*, vol. 24. North-Holland, 2005.

M. Salameh et M. Jaber : Economic production quantity model for items with imperfect quality. *International Journal of Production Economics*, 64(1-3):59–64, 2000.

R. Sarker et C. Newton : A genetic algorithm for solving economic lot size scheduling problem. *Computers and Industrial Engineering*, 42(2-4):189–198, 2002.

C. Saydam et J. Evans : A comparative performance analysis of the Wagner-Whitin algorithm and lot-sizing heuristics. *Computers and Industrial Engineering*, 18(1):91–93, 1990.

E. Silver et I. Moon : The multi-item single period problem with an initial stock of convertible units. *European Journal of Operational Research*, 132(2):466–477, 2001.

M. Singh, C. Abraham et R. Akella : Planning for production of a set of components when yield is random. *In Fifth IEEE/CHMT International Electronic Manufacturing Technology Symposium, 1988,'Design-to-Manufacturing Transfer Cycle'*, p. 196–200, 1988.

F. Sourd et S. Kedad-Sidhoum : The one-machine problem with earliness and tardiness penalties. *Journal of Scheduling*, 6(6):533–549, 2003.

F. Sourd et S. Kedad-Sidhoum : A faster branch-and-bound algorithm for the earliness-tardiness scheduling problem. *Journal of Scheduling*, 11(1):49–58, 2008.

C. Sox, P. Jackson, A. Bowman et J. Muckstadt : A review of the stochastic lot scheduling problem. *International Journal of Production Economics*, 62(3):181–200, 1999.

H. Sun, H. Huang et W. Jaruphongsa : A genetic algorithm for the economic lot scheduling problem under the extended basic period and power-of-two policy. *CIRP Journal of Manufacturing Science and Technology*, 2(1):29–34, 2009.

W. Szwarc : The weighted common due date single machine scheduling problem revisited. *Computers and Operations Research*, 23(3):255–262, 1996.

V. Tanaev, M. Kovalyov et Y. Shafransky : *Scheduling theory. Group technologies. Minsk, IEC NANB.* Russian, 1998.

O. Tang et R. Teunter : Economic lot scheduling problem with returns. *Production and Operations Management*, 15(4):488–497, 2006.

J. Teng : A simple method to compute economic order quantities. *European Journal of Operational Research*, 198(1):351–353, 2009.

R. Teunter et S. Flapper : Lot-sizing for a single-stage single-product production system with rework of perishable production defectives. *OR Spectrum*, 25(1):85–96, 2003.

R. Teunter, O. Tang et K. Kaparis : Heuristics for the economic lot scheduling problem with returns. *International Journal of Production Economics*, 118(1):323–330, 2009.

A. Wagelmans, S. Van Hoesel et A. Kolen : Economic lot sizing : An O (n log n) algorithm that runs in linear time in the Wagner-Whitin case. *Operations Research*, 40: 145–156, 1992.

Y. Wang et Y. Gerchak : Input control in a batch production system with lead times, due dates and random yields. *European Journal of Operational Research*, 126(2):371–385, 2000.

S. Webster et K. Baker : Scheduling Groups of Jobs on a Single Machine. *Operations Research*, 43(4):692, 1995.

E. Winands, I. Adan et G. Van Houtum : The stochastic economic lot scheduling problem : a survey. *European Journal of Operational Research*, (doi :10.1016/j.ejor.2010.06.011), 2010.

K. Wu et L. Ouyang : An integrated single-vendor single-buyer inventory system with shortage derived algebraically. *Production Planning and Control*, 14(6):555–561, 2003.

F. Yalaoui et C. Chu : Parallel machine scheduling to minimize total tardiness. *International Journal of Production Economics*, 76(3):265–279, 2002.

C. Yano et H. Lee : Lot sizing with random yields : A review. *Operations Research*, 43 (2):311–334, 1995.

D. Yao : Optimal Run Quantities for an Assembly System with Random Yields. *IIE Transactions*, 20(4):399–403, 1988.

M. Yao : The economic lot scheduling problem without capacity constraints. *Annals of Operations Research*, 133(1):193–205, 2005.

M. Yao et S. Elmaghraby : The economic lot scheduling problem under power-of-two policy. *Computers and Mathematics with Applications*, 41(10):1379–1393, 2001.

M. Yao et J. Huang : Solving the economic lot scheduling problem with deteriorating items using genetic algorithms. *Journal of Food Engineering*, 70(3):309–322, 2005.

A. Zhang et S. Guu : The multiple lot sizing problem with rigid demand and interrupted geometric yield. *IIE Transactions*, 30(5):427–431, 1998.

A. Zhang et S. Guu : Properties of the multiple lot-sizing problem with rigid demands and general yield distributions. *Computers and Mathematics with Applications*, 33(5):55–65, 1997.

B. Zhang, X. Xu et Z. Hua : A binary solution method for the multi-product newsboy problem with budget constraint. *International Journal of Production Economics*, 117 (1):136–141, 2009.

X. Zhu et W. Wilhelm : Scheduling and lot sizing with sequence-dependent setup : A literature review. *IIE Transactions*, 38(11):987–1007, 2006.