Optimisation de performances et maîtrise de la fiabilité dans la conception de systèmes de production

Milia Habib

Thèse
de doctorat
de l'UTT

Milia HABIB

Optimisation de performances et maîtrise de la fiabilité dans la conception de systèmes de production

Spécialité :
Optimisation et Sûreté des Systèmes

2017TROY0025 **Année 2017**

Thèse en cotutelle avec l'Université Libanaise - Beyrouth - Liban

Université Libanaise

utt
UNIVERSITÉ DE TECHNOLOGIE
TROYES

THESE

pour l'obtention du grade de

DOCTEUR de l'UNIVERSITE DE TECHNOLOGIE DE TROYES

Spécialité : OPTIMISATION ET SURETE DES SYSTEMES

présentée et soutenue par

Milia HABIB

le 29 septembre 2017

Optimisation de performances et maîtrise de la fiabilité dans la conception de systèmes de production

JURY

M. B. IUNG	PROFESSEUR DES UNIVERSITES	Président
M. D. AÏT-KADI	PROFESSEUR TITULAIRE	Rapporteur
M. N. CHEBBO	PROFESSEUR	Directeur de thèse
M. I. JARKASS	ENSEIGNANT CHERCHEUR	Directeur de thèse
M. N. REZG	PROFESSEUR DES UNIVERSITES	Rapporteur
M. Z. SARI	PROFESSEUR TITULAIRE	Examinateur
M. F. YALAOUI	PROFESSEUR DES UNIVERSITES	Directeur de thèse
Mme A. YALAOUI	MAITRE DE CONFERENCES - HDR	Examinateur

Personnalité invitée

M. H. CHEHADE	DOCTEUR, CHERCHEUR ASSOCIE UTT

Remerciements

Ce mémoire est le résultat d'un travail de recherche de trois ans. Il n'aurait pu être mené à terme sans le soutien de nombreuses personnes que je tiens vivement à remercier.

En premier lieu, je tiens à remercier très sincèrement mes directeurs de thèse, Monsieur Farouk YALAOUI, Professeur à l'Université de Technologie de Troyes et directeur du Laboratoire d'Optimisation des Systèmes Industriels (LOSI) pour m'avoir accueilli dans son équipe, mais également pour son soutien, sa rigueur scientifique, sa clairvoyance et ses hauts caractères personnels que j'ai beaucoup appréciés tout au long des travaux de ma thèse, Monsieur Nazir CHEBBO, Professeur et directeur du Laboratoire des Systèmes Industriels (LSI) à l'IUT-Saïda, Université Libanaise, et Madame Iman JARKASS, Enseignant chercheur à l'IUT-Saïda, Université Libanaise, pour m'avoir fait confiance, pour leur support, et leur encouragement qui m'ont permis de réaliser cette thèse dans d'excellentes conditions.

Je ne saurais assez remercier Monsieur Hicham CHEHADE, Chercheur associé à l'UTT et président de l'entreprise OPTA-LP, pour sa participation dans le suivi de cette thèse, sa disponibilité et ses conseils précieux.

J'exprime toute ma profonde gratitude aux membres du jury qui me font l'honneur de participer à l'examen de cette thèse : Monsieur Daoud AIT KADI, Professeur à l'Université Laval au Quebec, et Monsieur Nidhal REZG, Professeur à l'Université de Lorraine en leur qualité de rapporteurs. Madame Alice YALAOUI, Maître de conférences-HDR à l'Université de Technologie de Troyes, Monsieur Benoit IUNG, Professeur à l'Université de Lorraine, et Monsieur Zaki SARI, Professeur à l'Université Aboubekr Belkaïd, Tlemcen en leur qualité d'examinateurs. Ils ont également contribué par leurs nombreuses remarques et suggestions à améliorer la qualité de ce mémoire, et je leur en suis très reconnaissante.

Je tiens à remercier tous le personnel administratif de l'école doctorale de l'UL et de l'UTT, en particulier, Monsieur Fawaz EL OMAR, le directeur de l'ED de l'UL, Monsieur Régis LENGELLE,

le directeur de l'ED de l'UTT ainsi que Madame Pascale Denis, Madame Isabelle LECLERCQ, Madame Jana ELHAJJ et Madame Zeinab IBRAHIM pour toute aide qu'ils m'ont apportée lors de ce trajet doctoral. Je remercie vivement Monsieur Mohamad KHALIL, directeur de la plateforme de Biotechnologie (AZM) à l'école doctorale de l'UL pour son support et sa présence encourageante.

Je remercie l'Université Libanaise et l'Université de Technologie de Troyes pour avoir financé cette thèse dans le cadre de la convention de thèse en cotutelle entre l'UL et le réseau UT-INSA.

Mes vifs remerciements à tous les membres du Laboratoire LOSI pour leur accueil, pour la bonne ambiance de travail et leur amitié. Ma gratitude s'étend aussi aux Madame Véronique BANSE et Madame Bernadette ANDRE du pôle ROSAS à l'UTT pour leur gentillesse et leur disponibilité.

Je souhaite aussi remercier tous mes collègues et mes amis dans les deux universités UL et UTT pour leur soutien durant ces trois années et leur bonne humeur à toute épreuve. Nous avons partagé de bons moments.

Enfin, j'adresse mes plus chaleureux remerciements à mes parents Sleiman et Saniya, toute ma famille et ma belle-famille en particulier ma belle-mère Aida, pour leur soutien inconditionnel si précieux. Un grand merci à mon cher mari Jalal et notre petite princesse Joelle d'avoir cru en moi et d'avoir tout fait pour que ce moment aboutisse.

A Jalal, mon mari,

A Joelle, notre fille.

Résumé

Cette thèse porte sur les problèmes de conception et d'optimisation de la fiabilité des systèmes avec la prise en compte de la dépendance redondante. Nous nous intéressons d'abord à la conception de systèmes réparables dépendants de type parallèle et k sur n : G. Après avoir rappelé le modèle de la dépendance redondante présenté dans la littérature pour les systèmes parallèles, nous proposons un modèle plus général pour les systèmes k sur n : G. Ce modèle permet de quantifier la dépendance de défaillance entre les composants redondants du système. Nous évaluons également la disponibilité stationnaire du système avec la prise en compte de la dépendance à l'aide des modèles markoviens. Nous étudions ensuite la conception des systèmes réparables séries k sur n en considérant la notion de la dépendance redondante. Ces problèmes sont traités sous deux approches d'optimisation : mono et multicritère. Dans l'approche monocritère, nous abordons, dans un premier temps, le problème de minimisation des coûts sous contrainte d'une disponibilité exigée. Nous proposons de le résoudre en utilisant le solveur LINGO et en développant des algorithmes génétiques et des algorithmes d'optimisation par colonies de fourmis. Ces algorithmes sont ensuite améliorés par une recherche locale. Dans un deuxième temps, nous étudions le problème dual de maximisation de la disponibilité que nous le résolvons à l'aide des algorithmes génétiques et LINGO. Dans l'approche multicritère, nous considérons simultanément les deux objectifs. Nous proposons des algorithmes multiobjectifs basés sur NSGA-II et SPEA-II.

Mots-clés

- Conception
- Redondance (ingénierie)
- Fiabilité
- Systèmes d'aide à la décision
- Optimisation mathématique

Abstract

This thesis deals with the design and optimization problems of the reliability of the systems taking into account the redundant dependency. First, we focus on the design of repairable dependent parallel and k out of $n : G$ systems. After recalling the redundant dependency model presented in the literature for parallel systems, we propose a more general model for the k out of $n : G$ systems. This model allows quantifying the failure dependence between the redundant components of the system. We also evaluate the stationary system availability considering the dependence based on the Markov models. Then, we study the design of the series repairable k out of n systems considering the redundant dependency notion. These problems are treated under two optimization approaches : single and multicriteria. In the single criterion approach, we first address the minimization problem of the costs under a required availability constraint. We propose to solve it by using the LINGO solver and by developing genetic algorithms and ant colony optimization algorithms. These algorithms are then improved by a local search. In a second step, we study the dual problem of availability maximization which we solve using the genetic algorithms and LINGO. In the multicriteria approach, we simultaneously consider both objectives. We propose multi-objective algorithms based on the Non-dominated Sorting Genetic Algorithms (NSGA-II) and the second version of the Strength Pareto Evolutionary Algorithm (SPEA-II).

Keywords

- Design
- Redundancy (engineering)
- Dependability
- Decision support systems
- Mathematical optimization

Table des matières

Introduction **1**
Contexte général et motivations . . . 1
Organisation de thèse et contributions . . . 3
Organisation du manuscrit . . . 3
Liste de publications . . . 6

Introduction générale **1**

1 Contexte de l'étude, généralités et état de l'art **9**
1.1 Introduction . . . 10
1.2 Sûreté de fonctionnement des systèmes . . . 10
1.2.1 Fiabilité . . . 11
1.2.2 Disponibilité . . . 12
1.2.3 Maintenabilité . . . 13
1.2.4 Sécurité . . . 14
1.2.5 Principales caractéristiques utilisées en fiabilité . . . 14
1.2.5.1 Taux de défaillance . . . 15
1.2.5.2 Taux de réparation . . . 16
1.2.5.3 Grandeurs temporelles . . . 16
1.2.5.4 Lois de probabilité . . . 17
1.3 Défaillance . . . 18
1.3.1 Principales causes de défaillance . . . 18
1.3.2 Dépendance et classification . . . 18
1.4 Modélisation . . . 20
1.4.1 Modèles des systèmes . . . 20
1.4.1.1 Systèmes séries . . . 20
1.4.1.2 Systèmes parallèles . . . 21
1.4.1.3 Systèmes séries-parallèles ou parallèles-séries . . . 22
1.4.1.4 Systèmes redondants k/n . . . 22
1.4.1.5 Systèmes à redondance passive . . . 23
1.4.1.6 Systèmes cohérents . . . 23
1.4.1.7 Systèmes multi-états . . . 24
1.4.1.8 Autres systèmes . . . 25
1.4.2 Techniques de modélisation . . . 25

1.4.2.1 Méthodes sans l'EE . . . 25
1.4.2.2 Méthodes avec l'EE . . . 27
1.4.2.3 Méthodes de simulation . . . 30
1.4.3 Synthèse sur les techniques de modélisation . . . 32
1.5 Approches d'amélioration de la sûreté de fonctionnement des systèmes . . . 33
1.5.1 Allocation de fiabilité . . . 33
1.5.2 Allocation de redondance . . . 35
1.5.3 Allocation combinée de fiabilité et de redondance . . . 36
1.5.4 Autres problèmes d'allocation . . . 37
1.6 Optimisation . . . 37
1.6.1 Méthodes de résolution . . . 38
1.6.1.1 Méthodes exactes . . . 38
1.6.1.2 Méthodes approchées . . . 40
1.6.1.3 Outils dédiés . . . 44
1.6.2 Optimisation multiobjectif . . . 45
1.6.2.1 Formulation . . . 45
1.6.2.2 Notion de dominance . . . 45
1.6.2.3 Métriques de performance . . . 46
1.6.2.4 Approches de résolution . . . 49
1.6.3 Synthèse sur les méthodes d'optimisation . . . 51
1.7 Problématique et périmètres d'études . . . 51
1.8 Conclusion . . . 53

2 Optimisation des systèmes parallèles et k/n à composants dépendants 55
2.1 Introduction . . . 56
2.2 Etat de l'art sur les modèles de dépendances de défaillance . . . 56
2.3 Partie 1 : les systèmes parallèles . . . 59
2.3.1 Notions sur la dépendance redondante . . . 59
2.3.2 Description du problème . . . 60
2.3.3 Modèle mathématique . . . 62
2.3.4 Méthodes de résolution . . . 63
2.3.4.1 LINGO . . . 63
2.3.4.2 Algorithmes Génétiques . . . 64
2.3.5 Applications numériques . . . 66
2.3.5.1 Protocole de tests . . . 66
2.3.5.2 Analyse des résultats . . . 70
2.4 Partie 2 : les systèmes k sur n . . . 73
2.4.1 Description du système . . . 73
2.4.2 Modélisation de la dépendance redondante . . . 74
2.4.3 Description du problème d'optimisation . . . 76
2.4.4 Problème primal PPR . . . 77
2.4.4.1 Modèle mathématique . . . 77
2.4.4.2 Méthodes de résolution . . . 78
2.4.4.2.1 Résolution par LINGO . . . 78

2.4.4.2.2 Résolution par l'algorithme génétique 78
2.4.5 Problème dual PDU 79
2.4.5.1 Modèle mathématique 79
2.4.5.2 Méthodes de résolution 79
2.4.5.2.1 Résolution par LINGO 79
2.4.5.2.2 Résolution par l'algorithme génétique 80
2.4.6 Applications numériques 80
2.4.6.1 Analyse des résultats du problème PPR 81
2.4.6.2 Analyse des résultats du problème PDU 82
2.5 Conclusion 88

3 Optimisation des systèmes séries k sur n à composants dépendants 93
3.1 Introduction 94
3.2 Description du système 95
3.3 Concept de la dépendance redondante et évaluation de la disponibilité 96
3.4 Description du problème d'optimisation 99
3.5 Problème PPR 100
3.5.1 Modèle mathématique 100
3.5.2 Méthodes de résolution 101
3.5.2.1 LINGO 101
3.5.2.2 Algorithme Génétique 103
3.5.2.2.1 Recherche locale pour l'AG hybride 104
3.5.2.3 Algorithme d'optimisation par colonies des fourmis (ACO) 105
3.5.2.3.1 Recherche locale pour l'ACO hybride 109
3.5.2.4 Méthode exacte 110
3.5.3 Réglage des paramètres 111
3.5.3.1 Réglage des paramètres de l'AG 112
3.5.3.2 Réglage des paramètres de l'ACO 112
3.6 Applications numériques 113
3.7 Problème PDU 119
3.7.1 Modèle mathématique 119
3.7.2 Méthodes de résolution 119
3.7.2.1 LINGO 119
3.7.2.2 Algorithme Génétique 120
3.7.3 Applications numériques 121
3.8 Conclusion 122

4 Optimisation multiobjective des systèmes séries k sur n à composants dépendants 125
4.1 Introduction 126
4.2 Formulation du problème 127
4.2.1 Description du système 127
4.2.2 Notations 128
4.2.3 Objectives et analyse des contraintes 129

4.2.3.1 Analyse de la disponibilité du système considérant la dépendance redondante . . . 129
4.2.3.2 Coût et poids du système . . . 130
4.2.4 Modèle mathématique . . . 131
4.3 Méthodes de résolution . . . 132
4.3.1 Algorithme NSGA-II . . . 135
4.3.1.1 NSGA-II avec pénalité (NSGA-II-P) . . . 135
4.3.1.2 NSGA-II avec une domination de contrainte (NSGA-II-CD) . . . 138
4.3.2 Algorithme SPEA-II . . . 138
4.3.3 Réglage des paramètres . . . 140
4.3.4 Méthode exacte . . . 142
4.4 Exemples illustratives et analyse de performances . . . 142
4.4.1 Critères de comparaison . . . 142
4.4.2 Exemples et résultats . . . 143
4.4.2.1 Scenario 1 . . . 143
4.4.2.2 Scenario 2 . . . 146
4.5 Conclusion . . . 149

Conclusion et perspectives **155**

Bibliographie **161**

Table des figures

1.1 Taux de défaillance en fonction du temps [1] 15
1.2 Représentation du MTTF, MUT, MDT, MTBF [2] 17
1.3 Représentation d'un système série 21
1.4 Représentation d'un système parallèle 21
1.5 Un exemple d'un arbre de défaillance 26
1.6 Graphe d'état d'un système à 1 composant 28
1.7 Principe d'un algorithme génétique [3] 41
1.8 Choix des plus courts chemins à parcourir par les fourmis [4] 43
1.9 Problématique 52

2.1 Classification de la dépendance redondante pour un système parallèle [5] 60
2.1 Coût du système parallèle pour différents niveaux de la dépendance redondante . . 69
2.2 Diagramme d'état de transition du système k/n 73
2.3 Variation de $h(i)$ pour les différentes classes de la dépendance redondante 76
2.4 Coût du système k/n par rapport au niveau de la dépendance redondante pour le problème primal 90
2.5 Disponibilité du système k/n par rapport aux différents niveaux de la dépendance redondante pour le problème dual 91

3.1 Structure du système séries-k/n 96
3.2 Diagramme de transition d'état de sous-système i (k_i/n_i) 96
3.3 Graphe orienté décrivant le problème RAP 107
3.4 Pourcentage de déviation relative entre le coût de l'AG hybride et FEM et entre l'ACO hybride et FEM pour les 4 exemples testés 117

4.1 Codage appliqué 133
4.2 Fronts Pareto obtenus par les trois algorithmes pour les problèmes $P1$ avec $N_g = 250$ 152
4.3 Comparaison des fronts Pareto obtenus par les différents algorithmes pour les problèmes $P1$, $P2$ et $P3$ ayant $N_g = 5000$ 153

Liste des tableaux

2.1 Paramètres de l'algorithme génétique 66
2.2 Différentes catégories de la variation de ΔC et ΔT 70
2.3 Distribution de solutions (%) en termes de variation de ΔC pour toutes les combinaisons testées (p, q) 70
2.4 Distribution de solutions (%) en termes de variation de ΔC pour les différents niveaux de dépendance représentés par θ 71
2.5 Distribution de solutions (%) en termes de variation de ΔC 71
2.6 Distribution de solutions (%) en termes de variation de ΔT pour toutes les combinaisons testées (p, q) 71
2.7 Distribution de solutions (%) en termes de variation de ΔT pour les différentes valeurs de θ 71
2.8 Distribution de solutions (%) en termes de variation de ΔT 72
2.9 Classification de la dépendance redondante 75
2.10 Comparaison LINGO/AG pour le problème PPR ayant $p = 0$ et $q = 1$ 83
2.11 Comparaison LINGO/AG pour le problème PPR ayant $p = 0$ et $q = 2$ 83
2.12 Comparaison LINGO/AG pour le problème PPR ayant $p = 0.25$ et $q = 1$ 84
2.13 Comparaison LINGO/AG pour le problème PPR ayant $p = 0.25$ et $q = 2$ 84
2.14 Comparaison LINGO/AG pour le problème PPR ayant $p = 0.5$ et $q = 1$ 85
2.15 Comparaison LINGO/AG pour le problème PPR ayant $p = 0.5$ et $q = 2$ 85
2.16 Comparaison LINGO/AG pour le problème PPR ayant $p = 1$ et $q = 1$ 86
2.17 Comparaison LINGO/AG pour le problème PPR ayant $p = 1$ et $q = 2$ 86
2.18 Comparaison LINGO/AG pour le problème PDU ayant $p = 1$ et $q = 1$ 87
2.19 Comparaison LINGO/AG pour le problème PDU ayant $p = 1$ et $q = 2$ 87

3.1 Notations 97
3.2 Codage appliqué au problème étudié 106
3.3 Facteurs de plan d'expérience de l'AG 112
3.4 Facteurs de plan d'expérience de l'ACO 113
3.5 Matrice des tests effectués 113
3.6 Ensemble de données des exemples testés 113
3.7 Meilleures solutions obtenues pour les exemples $E1$, $E2$ et $E3$ par les différentes métaheuristiques avec les détails statistiques 114
3.8 Meilleures solutions obtenues pour l'exemple $E4$ par les différentes métaheuristiques avec les détails statistiques 115

3.9 Résultats numériques des métaheuristiques hybrides, FEM et LINGO 116
3.10 Données des composants 122
3.11 Résultats numériques de l'algorithme génétique et LINGO pour le problème dual . 123

4.1 Exemple de croisement 134
4.2 Exemple de mutation 134
4.3 Données d'entrée des composants pour les applications numériques 145
4.4 Données d'entrée des sous-systèmes 146
4.5 Résultats expérimentaux pour le problème $P1$ en termes de NS, SP, HRS et CT. 146
4.6 Comparaison des fronts Pareto obtenus pour le problème $P1$ en termes de μ, C_1 et C_2. 146
4.7 Deux solutions sélectionnées du front Pareto de NSGA-II-P correspondant au problème $P1$ 147
4.8 Retour de l'investissement 147
4.9 Résultats de la méthode exacte FEM 149
4.10 Résultats expérimentaux obtenus par les algorithmes proposés pour les problèmes $P2$ et $P3$. 149
4.11 Comparaison des fronts Pareto de FEM et des algorithmes proposés pour les problèmes $P2$ et $P3$ 150

Notations

l	Nombre de sous-systèmes dans un système
i	Indice de sous-système ($i = 1, ..., l$)
n_i	Nombre des composants dans le sous-système i
k_i	Nombre minimum des composants requis pour le sous-système i
n_i^{max}	Nombre maximum de n_i
r_i	Nombre des équipes de réparation affectées au sous-système i
$g(.)$	Fonction de la dépendance redondante
a	Paramètre de dépendance
λ^i	Taux de défaillance nominal des composants du sous-système i
μ^i	Taux de réparation des composants du sous-système i
C_i^c	Coût d'acquisition d'un composant utilisé dans le sous-système i
C_i^r	Coût d'une équipe de réparation affectée au sous-système i
A_i	Disponibilité du sous-système i
A_s	Disponibilité du système
A_0	Valeur minimale exigée pour la disponibilité du système
C_s	Coût total du système
C_{max}	Budget maximal disponible pour la conception du système
h_i	Nombre de types disponibles pour les composants du sous-système i
z_i	Indice de type des composants utilisé dans le sous-système i, $z_i \in \{1, ..., h_i\}$
W_s	Poids du système
W_c	Borne supérieure de poids du système

Introduction

Contexte général et motivations

Dans le contexte actuel caractérisé par une forte concurrence économique, le développement de politiques de sécurité globale et de maîtrise et sûreté des systèmes, la phase de conception d'un processus industriel doit être réalisée de manière à garantir certaines exigences liées à sa sûreté, à la fiabilité des machines, à sa productivité, à son ergonomie, etc. Lors de cette phase de conception, les différentes alternatives fonctionnelles et technologiques permettant de répondre au cahier des charges doivent donc être analysées. Chaque alternative est caractérisée par sa fiabilité, sa maintenabilité, sa productivité, la durée de vie des composants, etc. Une étude poussée doit donc être menée de manière à choisir les composants à utiliser pour respecter un ensemble de contraintes (coût d'investissement, taux de performance, fiabilité du processus, productivité, etc.) et pour atteindre une conception optimale du système en question. De ce fait, la recherche sur l'optimisation de la fiabilité/disponibilité des systèmes dès la phase de conception est d'une grande importance.

L'allocation de fiabilité vise la conception des systèmes sûrs avec des coûts minimaux en adoptant une méthodologie d'optimisation efficace et applicable très facilement. Dans cette famille de modèles d'allocation, on distingue principalement :

- Les méthodes reposant sur la maximisation de la fiabilité sous contrainte de coût,
- Les méthodes reposant sur la minimisation des coûts sous contrainte de fiabilité,
- Les méthodes qui cherchent à maximiser la fiabilité et minimiser les coûts simultanément.

Des contraintes de volume, de poids ou de productivité peuvent être aussi prises en compte. Dans la majorité de ces problèmes, on doit décider de la fiabilité à allouer à chaque composant (et éventuellement du nombre de composants à mettre en parallèle), du type à attribuer (à partir d'un ensemble disponible dans le marché). On peut aussi s'intéresser aux différentes options tech-

nologiques et architecturales possibles. Une fois le problème d'allocation est formulé sous forme d'un problème d'optimisation, il reste toujours à faire le choix d'un algorithme de résolution. Il est important à noter que les problèmes d'optimisation de la disponibilité/fiabilité appartiennent en général à la classe NP-difficile. Cela signifie qu'il est très peu probable d'avoir une solution optimale avec un temps d'exécution polynomial. Les performances des méthodes de résolution sont limitées par la taille des problèmes. Il est irréaliste d'appliquer de méthodes exactes pour les problèmes de grande taille. Pour cela, les travaux les plus récents portent essentiellement sur le développement des heuristiques et des métaheuristiques.

De plus, les systèmes modernes sont très dynamiques et sont souvent soumis à des mécanismes de dégradation, à des interventions externes (réparation, etc.). Des dépendances et des interactions complexes (matérielles, humaines, environnementales, etc.) sont fréquemment présentes. On distingue, en particulier, la dépendance des défaillances entre les différents composants qui a un impact potentiel sur les performances fiabilistes du système. Les défaillances des composants et les interventions déterministes sur le système sont modélisées globalement par des processus stochastiques. De ce fait, il est devenu important d'identifier et de prendre en compte des interventions et du comportement des systèmes au cours du temps à la phase de conception avant l'installation. De plus, lorsque la durée de vie du système est supposée infinie, une conception basée sur le fonctionnement stationnaire sera plus intéressante afin d'optimiser la performance du système à long terme.

Ce sujet de thèse s'inscrit dans l'esprit de la thématique de recherche transversale de l'Université de Technologie de Troyes (UTT) et certains de ses PST (Programmes Scientifiques et Techniques) visant à développer une méthodologie intégrée, ainsi que les outils associés, capable de permettre à une filière/entreprise/collectivité de pouvoir mettre en place une démarche de conception innovante de ses systèmes. Il rejoint également la thématique de recherche principale du Laboratoire des Systèmes Industriels (LSI) de l'Université Libanaise (UL) concernant la maîtrise, la fiabilité et l'optimisation de la maintenance des systèmes de production. Les travaux menés dans ce manuscrit sont en collaboration avec l'UL et constituent la suite de plusieurs travaux effectués depuis plusieurs années à l'UTT dans le domaine de la conception et de la fiabilité. Nous nous intéressons particulièrement à la conception des systèmes réparables à composants dépendants pour diffèrentes configurations. L'objectif est de mettre en place de modèles et d'algorithmes de résolution pour tout d'abord évaluer les performances et ensuite optimiser la conception des systèmes étudiés tout en considérant la dépendance redondante entre les composants.

Organisation de thèse et contributions

Cette thèse apporte plusieurs contributions au domaine de la conception des systèmes sûrs. Un modèle de la dépendance redondante a été proposé pour les systèmes k sur n : G. Ce modèle permet de quantifier la dépendance de défaillance entre les composants du système. Il est à noter que la configuration k/n (avec $k < n$) est très répandue dans les systèmes à tolérance de pannes, les applications industrielles, etc. Elle révèle qu'un système à n composants fonctionne si au moins k composants parmi les n fonctionnent. De plus, plusieurs modèles et approches d'optimisation ont été développés qui se diffèrent par :

- le type de systèmes pris en compte (parallèle, k/n, série k/n),
- les variables de décisions optimales à trouver qui peuvent être de nature discrète (le nombre des composants à mettre en parallèle, le type des composants, le nombre des équipes de réparation) ou continues (taux de défaillance, taux de réparation),
- les contraintes considérées (poids, coût, disponibilité),
- les méthodes de résolution (méthodes exactes, approchées sans ou avec hybridation, un solveur dédié).

Quant aux grandeurs à optimiser, nos travaux se sont concentrés essentiellement sur deux critères qui intéressent les équipes de conception : la minimisation des coûts et la maximisation de la disponibilité. Pour cela, nous avons tenté dans un premier temps de résoudre simultanément les deux problèmes d'optimisation en monocritère. Un seul parmi les deux critères essentiels est pris dans chaque résolution comme fonction objective tout en considérant l'autre comme contrainte. Dans un deuxième temps, nous avons étudié également l'approche multicritère et cela en considérant les deux critères simultanément. L'ensemble des modèles et des méthodes développés peut former un outil d'aide à la décision qui répond d'une façon efficace au besoin industriel.

Organisation du manuscrit

Ce mémoire constitué de quatre chapitres est organisé comme suit :

Le chapitre 1 est consacré à l'introduction des différentes notions et définitions nécessaires à la description du sujet d'étude de cette thèse ainsi qu'à la compréhension de la contribution scientifique des travaux réalisés. Nous présentons d'abord les différents concepts de la sûreté de fonctionnement (SDF). Puis, nous décrivons les dépendances qu'un système peut avoir en

mettant l'accent sur les défaillances dépendantes. Nous nous intéressons, par la suite, aux modèles ainsi qu'aux techniques d'évaluation et d'amélioration de la performance des systèmes. Nous présentons un état de l'art pour les modèles considérés et les méthodes de résolution utilisées pour l'optimisation mono et multiobjectif en mettant l'accent sur leur application dans le domaine fiabiliste. Enfin, nous précisons la problématique abordée et nous terminons ce chapitre par une conclusion.

Le chapitre 2 est consacré à l'étude du problème de conception des systèmes parallèles et des systèmes k/n à composants dépendants. Le nombre de composants à utiliser ainsi que leurs caractéristiques en termes de taux de défaillance et de taux réparation sont à déterminer pour ces systèmes. Nous rappelons les modèles de dépendances de défaillance en mettant l'accent sur la dépendance redondante. Dans un premier temps, nous décrivons un problème d'optimisation d'un système parallèle à composants dépendants avec comme fonction objective la minimisation du coût du système sous contrainte d'une disponibilité exigée. Dans un deuxième temps, nous étudions le problème de conception du système k/n. Nous présentons une extension du concept de la dépendance redondante à tel système. Une fonction de dépendance est introduite pour quantifier la dépendance de défaillance entre les composants redondants du système. Nous évaluons ensuite la disponibilité du système k/n en utilisant les chaines de Markov, tout en considérant la notion de la dépendance redondante. Ensuite, nous considérons deux problèmes d'optimisation : primal et dual. Le premier vise à minimiser le coût du système sous contrainte d'une disponibilité exigée. Alors que le deuxième a pour objectif de maximiser la disponibilité sous contrainte d'un budget maximal. Nous proposons de méthodes de résolution basées sur le solveur LINGO et l'algorithme génétique (AG). Des applications numériques sont utilisées pour tester l'efficacité des méthodes proposées. Des conclusions clôturent ce chapitre.

Dans le chapitre 3, nous nous intéressons au problème d'allocation de redondance avec dépendance redondante dans les systèmes réparables séries k sur n. Vu que la dépendance de défaillance et les ressources de réparation affectent significativement la disponibilité du système, il est important de les prendre en compte lors de la phase de conception. Après avoir défini une fonction particulière de la dépendance redondante, les expressions explicites de la disponibilité stationnaire du système relatives aux différents niveaux de dépendance sont obtenues. Un problème d'optimisation non linéaire est ensuite formulé en monocritère. Il vise à minimiser les coûts associés aux composants

et aux équipes de réparation sous contrainte d'une disponibilité requise du système. Pour chaque niveau de dépendance, nous déterminons le nombre de composants et le nombre des équipes de réparation à allouer à chaque sous-système afin d'avoir la meilleure configuration économique et fiable du système tout entier. Des méthodes de résolution basées sur l'algorithme génétique, les algorithmes d'optimisation par colonies de fourmis, et leur hybridation avec une recherche locale sont développées. De problèmes de différentes tailles sont testés afin d'évaluer l'efficacité des approches d'optimisation proposées. Les solutions des meilleurs algorithmes sont comparées à celles obtenues par une méthode exacte et par le solveur LINGO. Nous abordons, par la suite, le problème de maximisation de la disponibilité sous contrainte de coût pour les systèmes k sur n. Des conclusions et perspectives pour ce problème monocritère clôturent ce chapitre.

L'aspect multiobjectif du problème de conception des systèmes séries k sur n est abordé dans le chapitre 4. Nous commençons par une description et une modélisation mathématique du problème étudié. Nous nous concentrons sur deux objectifs : maximisation de la disponibilité et minimisation du coût sous contrainte du poids donné du système. Les composants appartenant au même sous-système sont supposés identiques et peuvent être dépendants. Ils sont choisis parmi un ensemble disponible dans le marché. En plus du nombre de composants redondants et de nombre des équipes de réparation à attribuer à chaque sous-système, nous considérons le type de composants et leur niveau de dépendance comme des variables de décision. Nous présentons ensuite de méthodes d'optimisation qui sont basées sur la deuxième version de l'algorithme génétique par dominance de Pareto (NSGA-II) et l'algorithme SPEA-II. Plusieurs techniques de gestion des contraintes sont considérées. Afin d'analyser les performances des différentes méthodes développées, une application numérique est établie et une comparaison basée sur de différentes métriques bien connues est présentée.

Une conclusion générale termine cette thèse et des perspectives de recherche sont présentées.

Liste de publications

Articles dans des journaux internationaux référencés avec comité de lecture

[J.1] **Milia Habib**, Farouk Yalaoui, Hicham Chehade, Nazir Chebbo, Iman Jarkass, "Multi-objective design optimization of repairable k-out-of-n subsystems in series with redundant dependency", International Journal Of Production Research, doi :10.1080/00207543.2017.1346319.

Articles dans des conférences internationales avec actes répertoriés dans les bases de données internationales

[C.1] **Milia Habib**, Hicham Chehade, Farouk Yalaoui, Nazir Chebbo, Iman Jarkass, "Availability optimization of a redundant dependent system using genetic algorithm", *IFAC-PapersOnline* Volume 49, Issue 12, 2016, Pages 733-738, ISSN 2405-8963, *IFAC 8th International Conference on Manufacturing Modelling, Management and Control (MIM 2016)*, Troyes, France, 28-30 June, 2016.

[C.2] **Milia Habib**, Farouk Yalaoui, Hicham Chehade, Nazir Chebbo, Iman Jarkass, "Design optimization and redundant dependency study of series k-out-of-n :G repairable systems", *IFAC-PapersOnLine*, Volume 49, Issue 28, 2016, Pages 126-131, ISSN 2405-8963, *3rd IFAC Workshop on Advanced Maintenance Engineering, Services and Technology (AMEST 2016)*, Biarritz, France, 19-21 October, 2016.

[C.3] **Milia Habib**, Hicham Chehade, Farouk Yalaoui, Nazir Chebbo, Iman Jarkass, "Maximization of system availability with failure dependency", 2016 *IEEE International Multidisciplinary Conference on Engineering Technology (IMCET* 2016*)*, Pages 115-119, doi : 10.1109/IMCET.2016.7777437, Beirut, Lebanon, 2-4 November 2016.

[C.4] **Milia Habib**, Farouk Yalaoui, Hicham Chehade, Nazir Chebbo, Iman Jarkass, "Availability analysis and redundant dependency modeling of k-out-of-n :G system", 8 pages, *10th International Conference On Mathematical Methods In Reliability (MMR 2017)*, Grenoble, France, 3-6 Juillet 2017.

Communications dans des conférences internationales sans actes

[C.1] **Milia Habib**, Farouk Yalaoui, Hicham Chehade, Nazir Chebbo, Iman Jarkass, "Evaluation et optimisation des performances des systèmes de production à la phase de conception", *22nd Scientific International Conference The Social Avenues of the Research, (LAAS'*16*)*, Holy Spirit University of Kaslik, Lebanon, 14-15 April, 2016.

Chapitre 1

Contexte de l'étude, généralités et état de l'art

Résumé :

Ce chapitre est consacré à la description du contexte général de l'étude. Dans un premier temps, nous introduisons les terminologies et les notions caractérisant la sûreté de fonctionnement des systèmes (SDF). Nous nous intéressons également à décrire et classifier les défaillances qu'un système peut subir dans le domaine de la sûreté en mettant l'accent sur les défaillances dépendantes. Dans un deuxième temps, nous décrivons les techniques de modélisation qui permettent d'évaluer les performances des systèmes. Nous analysons ainsi les principales approches d'amélioration de la SDF. Par la suite, nous présentons les différentes méthodes d'optimisation, y compris l'aspect multiobjectif en mettant l'accent sur leur application dans le domaine fiabiliste. Finalement, nous présentons le contexte dont cette thèse fait l'objet. Dans ce cadre, nous décrivons la problématique étudiée basée sur la maîtrise et l'optimisation de la fiabilité/disponibilité dans la conception des systèmes industriels.

1.1 Introduction

La complexité croissante des systèmes industriels, leur implication grandissante dans la vie économique et sociale, la nécessaire minimisation de leur coût de construction et d'exploitation dans un monde très concurrentiel : tout nécessite qu'une grande attention soit accordée à la sûreté de fonctionnement dès la phase de conception. La sûreté de fonctionnement est un concept générique qui englobe des notions fondamentales telles que la fiabilité, la maintenabilité, la disponibilité et la sécurité [6].

Les systèmes industriels tels que les systèmes de production, sont composés souvent de plusieurs sous-systèmes comportant de composants dépendants, eux mêmes caractérisés par des comportements mettant en évolution différents phénomènes. La performance de ces systemes dépend des configurations adoptées, ainsi que des liens et des dépendances entre les composants. La modélisation et l'évaluation des mesures de sûreté de fonctionnement est un large domaine d'investigation.

Dans le monde industriel, la phase de conception consiste à étudier les différentes alternatives fonctionnelles et technologiques du système afin de mettre en place une démarche efficace permettant de répondre aux exigences des cahiers des charges. Chaque alternative est caractérisée par sa fiabilité, sa maintenabilité, sa productivité, etc. Une étude poussée doit donc être menée de manière à choisir les composants (et éventuellement leur nombre) à utiliser à partir de composants disponibles sur le marché, dont les attributs tels que le coût, le poids, la fiabilité, etc. peuvent être connus. Un ensemble de contraintes (coût d'investissement, taux de performance, fiabilité du processus, ...) sont à respecter. La conception de systèmes fiables peut être formulée alors par un problème d'optimisation combinatoire. L'optimisation de la fiabilité/disponibilité en phase de conception des systèmes est désormais l'une des principales préoccupations des entreprises. Ce problème présente un vrai intérêt dans le domaine de la sûreté des systèmes.

Dans ce contexte, le travail présenté dans ce mémoire se base sur l'optimisation de la conception et la maîtrise de la fiabilité/disponibilité de systèmes industriels. Nous présentons dans la section suivante le concept de la sûreté de fonctionnement des systèmes et ses principales composantes.

1.2 Sûreté de fonctionnement des systèmes

L'étude de sûreté de fonctionnement (SDF) est récente et elle s'est développée au cours du XXe siècle donnant ainsi naissance à une véritable science de l'ingénieur. Selon Villemeur [2], la

sûreté de fonctionnement est définie comme la Science des Défaillances, elle inclut leur connaissance, leur évaluation, leur prévision, leur mesure et leur maîtrise. Au sens strict, elle est définie comme l'aptitude d'une entité à satisfaire à une ou plusieurs fonctions requises dans des conditions données. Elle regroupe les activités d'évaluation prévisionnelle de la fiabilité, de la disponibilité, de la maintenabilité et de la sécurité. Ces activités visent à minimiser les risques d'accidents et de dysfonctionnement des systèmes. Leur prise en compte en phase de conception permet de mieux maîtriser et améliorer le fonctionnement des systèmes aussi bien sur le plan de la sécurité que sur le plan économique.

1.2.1 Fiabilité

On retrouve dans la littérature plusieurs définitions de la fiabilité, toutes étant justifiées et relatives à un secteur d'activité donné. L'Union Technique de l'Electricité (UTE), sur recommandation de la commission électrotechnique internationale, a proposé la définition suivante : la fiabilité (*Reliability*) est « *l'aptitude d'un dispositif à accomplir une fonction requise, dans des conditions données, pendant une durée donnée* ». Le terme dispositif désigne entité, c'est-à-dire tout composant, sous-système ou système que l'on peut considérer et essayer individuellement. Au sens mathématique, la fiabilité est généralemenent caractérisée ou mesurée par la probabilité que l'entité accomplisse une ou plusieurs fonctions requises dans des consitions données, pendant une durée donnée [7]. L'expression mathématique de $R(t)$ de l'entité S à un instant t est :

$$R(t) = Prob\left\{S \text{ non défaillante sur intervalle } [0,t]\right\} \tag{1.1}$$

Cette expression révèle que l'entité S n'ait occupé que des états de marche pendant la durée $[0,t]$. L'aptitude contraire est appelée « défiabilité », sa mesure est notée $\overline{R}(t)$:

$$\overline{R}(t) = 1 - R(t) \tag{1.2}$$

On distingue plusieurs types de fiabilité associée à un système [2] :

– La fiabilité prévisionnelle, qui estime une fiabilité future à partir de considérations sur la conception des systèmes. Elle est obtenue à partir d'un modèle mathématique, connaissant la fiabilité estimée de ses composants. Alain Pagès et Michel Gondran [7] ont présenté dans leur ouvrage en 1980 une synthèse des travaux effectués à cette époque sur les méthodes d'évaluation de la fiabilité prévisionnelle des systèmes. Ils ont été étroitement associés aux

préoccupations d'Electricité de France concernant la fiabilité des centrales, des réseaux de transport et d'interconnexion.

- La fiabilité intrinsèque, mesurée au cours d'essais spécifiques effectués selon un protocole d'essais bien défini.
- La fiabilité opérationnelle, évaluée en tenant compte des données obtenues à partir de l'observation et de l'analyse du comportement d'entités identiques dans les mêmes conditions opérationnelles.
- La fiabilité extrapolée qui résulte d'une extension (par extrapolation définie ou par interpolation) de la fiabilité opérationnelle à des durées ou des conditions différentes.

1.2.2 Disponibilité

Les éléments constituant le système sont susceptibles d'avoir des défaillances. Ces éléments peuvent être réparables ou non réparables. La disponibilité (*Availability*) instantannée est l'aptitude d'une entité à être en état d'accomplir une fonction requise dans des conditions données, à un instant donné. De manière générale, elle est mesurée par la probabilité que l'entité S soit non défaillante à l'instant t. L'expression mathématique de $A(t)$ de l'entité S à un instant t est [8] :

$$A(t) = Prob\left\{S \text{ non défaillante à l'instant } t\right\} \tag{1.3}$$

Cette grandeur montre l'efficacité de l'entité et sa remise en opération suite aux défaillances. Elle représente la mesure dans laquelle un système ou un composant est opérationnel et accessible lorsqu'on fait appel à lui [9]. Il est à noter que la disponibilité et la fiabilité sont équivalentes quand le système est non réparable. Dans le cas général (par exemple pour un système réparable), on trouve la relation suivante :

$$A(t) \geq R(t) \tag{1.4}$$

L'aptitude contraire de la disponibilité sera dénommée « indisponibilité », sa mesure est notée $\overline{A}(t)$:

$$\overline{A}(t) = 1 - A(t) \tag{1.5}$$

En pratique, on s'intéresse plus à l'évaluation de la disponibilié sur un intervalle de temps. On peut distinguer alors deux types qui sont souvent utilisés [10] :

- La disponibilité moyenne à temps de fonctionnement (*Average uptime availability*) : c'est la

portion du temps durant lequel l'entité ou le système S est disponible dans un intervalle de temps donné $[0, T]$, elle est donnée par :

$$A(T) = \frac{1}{T} \int_0^T A(t)\,\mathrm{d}t \tag{1.6}$$

– La disponibilité asymptotique (*Steady state availability)* : c'est la probabilité que le système soit opérationnel dans un intervalle de temps très large, elle est donnée par :

$$A(\infty) = \lim_{t \to \infty} A(t) \tag{1.7}$$

Il est à noter que pour les systèmes industriels avec des composants réparables, le paramètre le plus intéressant utilisé pour étudier les actions de maintenance est la disponibilité stationnaire [11].

1.2.3 Maintenabilité

La notion de maintenabilité (*Maintainability*) ne concerne que les systèmes réparables. C'est l'aptitude d'une entité à être maintenue ou rétablie dans un état dans lequel elle peut accomplir une fonction requise, lorsque la maintenance est réalisée dans des conditions données avec des procédures et des moyens prescrits [2].
La maintenabilité notée $M(t)$ est généralement mesurée par la probabilité que l'entité S, à l'instant t, soit en état d'accomplir ses fonctions à cet instant, sachant qu'elle était en panne à l'origine ($t = 0$) [2], [8].

$$M(t) = Prob\,\{S \text{ est réparée sur l'intervalle } [0, t]/S \text{ est en panne à } t = 0\} \tag{1.8}$$

Il existe de nombreuses politiques de maintenance. L'un des principaux critères de distinction est le moment de l'intervention par rapport à la panne. On distingue [2], [6] :

– La maintenance corrective se fait après la détection d'une panne. Elle suppose un nombre important de réparateurs (surtout sur des systèmes de production pour lesquels la disponibilité est cruciale) et une bonne gestion des pièces de rechange. Suivant la nature des interventions, on distingue deux types de remise en état de fonctionnement :
  – La réparation (maintenance curative) consiste à une remise en l'état initial (état de fonctionnement conforme aux conditions données),
  – le dépannage consiste à une remise en état provisoire qui sera obligatoirement suivi d'une réparation.

– La maintenance préventive ne consiste pas à réparer les pannes, mais à les anticiper. Elle est exécutée à des intervalles de temps prédéterminés et destinée à réduire la probabilité de défaillance du composant. Elle se subdivise à son tour en maintenances préventives systématiques, maintenances préventives conditionnelles, maintenances prévisionnelles et proactives. Plus de détails sur ces différents types peuvent être trouvés dans [6].

1.2.4 Sécurité

La sécurité (*Safety*) est l'aptitude d'une entité à éviter de faire apparaître, dans des conditions données, des événements critiques ou catastrophiques. Elle est généralement mesurée par la probabilité qu'une entité S ne laisse pas apparaître dans des conditions données, des événements critiques ou catastrophiques [2]. On distingue deux types de sécurité [12] :

– la sécurité-innocuité vise à se protéger des défaillances catastrophiques dont les conséquences sont inacceptables vis-à-vis du risque,
– la sécurité-confidentialité correspond à la prévention d'accès ou de manipulations non autorisées de l'information.

D'autres concepts peuvent être également inclus dans la sûreté de fonctionnement tels que la durabilité, la continuabilité, la serviabilité [2]. Ces grandeurs ne sont pas étudiées dans ce manuscrit.

1.2.5 Principales caractéristiques utilisées en fiabilité

Il existe de nombreuses grandeurs qui peuvent caractériser la sûreté de fonctionnement des systèmes. Dans cette partie, nous présentons les principales mesures utilisées en fiabilité [13].
Désignons par T la variable aléatoire caractérisant l'instant de défaillance d'un dispositif donné. Notons par $F(t)$ la fonction de répartition correspondante, elle représente la probabilté que le dispositif tombe en panne pendant l'intervalle de temps $[0, t]$. La fiabilité $R(t)$ s'exprime donc par la relation suivante :

$$R(t) = Prob\,(T > t) = 1 - Prob\,(T \leq t) = 1 - F(t) \tag{1.9}$$

Il en résulte que $R(t)$ est une fonction décroissante de t sur $[0, \infty[$ et que $R(t) + F(t) = 1$.
La densité de probabilité de défaillance, notée par $f(t)$, est telle que :

$$f(t) = \frac{dF(t)}{dt} = \lim_{\Delta t \to 0} \frac{Prob\,(t < T \leq t + \Delta t)}{\Delta T} \tag{1.10}$$

Soit encore :

$$f(t) = -\frac{dR(t)}{dt} \tag{1.11}$$

On a donc la relation entre la fiabilité et la densité de défaillance :

$$R(t) = 1 - \int_0^t f(t)dt \tag{1.12}$$

1.2.5.1 Taux de défaillance

Le taux de défaillance d'un dispositif à l'instant t est défini par la fonction $\lambda(t)$ telle que :

$$\lambda(t) = \lim_{\Delta t \to 0} \frac{Prob\,(t < T \leq t + \Delta T / T > t)}{\Delta T} = -\frac{dR(t)}{dt}\frac{1}{R(t)} = \frac{f(t)}{1 - F(t)} = \frac{f(t)}{1 - \int_0^t f(t)dt} \tag{1.13}$$

Physiquement, le produit $\lambda(t)\Delta t$ représente la probabilité conditionnelle qu'une défaillance du dispositif se produise dans l'intervalle de temps $[t, t + \Delta t]$ sachant que ce dispositif n'est pas tombé en panne avant t.

On a les relations suivantes entre la fiabilité et le taux de défaillance :

$$\lambda(t) = \frac{f(t)}{R(t)} \tag{1.14}$$

$$\lambda(t) = -\frac{d[LogR(t)]}{dt} \Leftrightarrow R(t) = e^{-\int_0^t \lambda(t)dt} \tag{1.15}$$

Il est fréquant que des dispositifs présentent un taux de défaillance en fonction du temps $\lambda(t)$ suivant une courbe dite « en baignoire ». Cette courbe décrit l'évolution des composants au cours de leur cycle de vie qui suit trois phases comme montre la figure 1.1 [2], [1], [14]. Les trois périodes

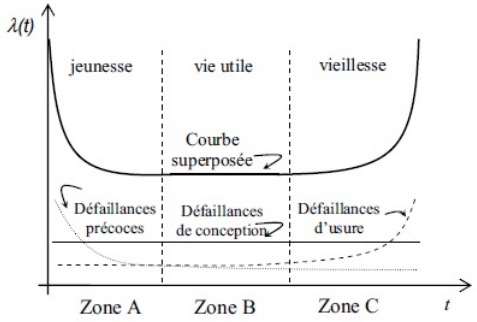


FIGURE 1.1 – Taux de défaillance en fonction du temps [1]

de la courbe en baignoire sont :

- période de jeunesse dans laquelle le taux de défaillance de composant décroit rapidement à cause de l'élimination des défauts de jeunesse et au rodage, c'est la période de défaillance précoce (Zone A),
- période de vie utile dans laquelle le taux de défaillance est approximativement constant. Les différents composants ont prouvé leur robustesse aux défauts de jeunesse. Ils sont en phase de maturité (Zone B),
- période de vieillissement dans laquelle le taux de défaillance augmente rapidement en fonction du temps, c'est la période de défaillance d'usure (Zone C).

1.2.5.2 Taux de réparation

Pour une entité réparable, le taux de réparation instantanné ($\mu(t)$) exprime la probabilité pour qu'une entité S, qui a été en panne pendant un temps t, retrouve son aptitude à remplir sa fonction dans l'unité de temps qui suit.
Mathématiquement, le taux de réparation instantanné $\mu(t)$ s'écrit [2] :

$$\mu(t) = \lim_{\Delta t \to 0} \frac{P[S \text{ est réparée sur } [t, t+\Delta t] / S \text{ en panne sur} [0, t]}{\Delta T} \tag{1.16}$$

1.2.5.3 Grandeurs temporelles

De nombreuses grandeurs temporelles peuvent caractériser l'état de fonctionnement du système : avant défaillance, entre défaillance, entre défaillance et réparation, etc. [14]. Villemeur [2] présente quelques temps caractérisant de la SdF. De même, il les illustre au moyen d'un graphique décrivant la relation existant entre eux (figure 1.2).

- La durée moyenne du bon fonctionnement du système jusqu'à la première défaillance (MTTF : *Mean Time To First Failure*). La définition est :

$$MTTF = \int_0^\infty R(t)dt = \int_0^\infty tf(t)dt \tag{1.17}$$

- La durée moyenne des temps techniques de réparation (MTTR : *Mean Time To Repair*),
- La durée moyenne de fonctionnement après réparation (*MUT* : *Mean Up-Time*),
- La durée moyenne d'indisponibilité du système (MDT : *Mean Down Time*),
- La moyenne des temps entre deux défaillances d'un système réparable (MTBF : *Mean Time*

Between Failure).

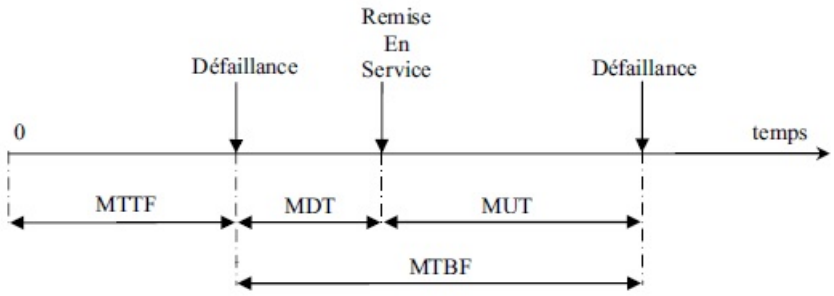


FIGURE 1.2 – Représentation du MTTF, MUT, MDT, MTBF [2]

1.2.5.4 Lois de probabilité

Il existe de lois fréquemment utilisées en sûreté de fonctionnement qui permettent d'analyser la durée de vie d'un composant. Les principales sont la loi exponentielle et la loi de Weibull [2].

– La loi exponentielle décrit le cas de composant dans la zone de vie utile. Le taux de défaillance constant indique une distribution de défaillance exponentielle. Cette loi est sans mémoire et souvent exploitée par des processus markoviens homogènes. Elle peut servir pour l'étude de dégradation des composants électroniques. Elle est très souvent retenue dans la littérature car elle conduit à des calculs simples.

$$\lambda(t) \equiv \lambda \Rightarrow R(t) = exp(-\lambda t), \quad MTTF = \frac{1}{\lambda} \tag{1.18}$$

De même, si le taux de réparation $\mu(t)$ est constant, on en déduit :

$$\mu(t) \equiv \mu \Rightarrow M(t) = 1 - exp(-\mu t), \quad MTTR = \frac{1}{\mu} \tag{1.19}$$

– La loi de Weibull est largement utilisée pour modéliser les données de fiabilité en raison de son paramétrage aisé et multiforme.

$$\lambda(t) = \frac{\beta}{\theta}\left[\frac{t}{\theta}\right]^{\beta-1}, \theta > 0, \beta > 0, t \geq 0 \tag{1.20}$$

Les paramètres θ et β sont respectivement les paramètres d'échelle et de forme. La loi de

Weibull permet de repésenter un taux de défaillance croissant ($\beta > 1$) ainsi que décroissant ($\beta < 1$) avec le temps. En général, elle correspond bien aux composants mécaniques pour lesquels les taux de défaillances sont rarement constants.

1.3 Défaillance

1.3.1 Principales causes de défaillance

Les défaillances des composants du système peuvent avoir plusieurs causes. Une classification a été faite par Villemeur [2] qui regroupe les défaillances en trois catégories : défaillance première, défaillance seconde et défaillance de commande. La défaillance première résulte d'une cause interne d'un composant et non de la défaillance d'un autre composant. Pour un composant en exploitation, elle peut être dûe au vieillissement naturel, à des problèmes d'usure, à des défauts de conception, fabrication ou de spécifications techniques, etc. La défaillance seconde est une défaillance d'un composant dont la cause directe ou indirecte est une défaillance d'un autre composant. Autres causes peuvent aussi être à l'origine de cette défaillance telles que des contraintes excessives en dehors de la conception, des conditions particulières dans l'environnement, des erreurs humaines. La défaillance de commande est dûe aux signaux incorrects de commande et de contrôle.
Nous nous intéréssons dans cette thèse à la deuxième catégorie où la défaillance d'un composant peut affecter autres composants dans le système. C'est l'un des aspects des défaillances dépendantes. Dans le paragraphe suivant, on décrit brièvement les défaillances dépendantes pour bien situer nos travaux.

1.3.2 Dépendance et classification

Les systèmes modernes sont souvent des systèmes multi-composants très dynamiques et soumis à des interventions externes (maintenance, etc.). Des dépendances et des interactions complexes (matérielles, humaines, environnementales, etc.) sont fréquemment présentes. D'après Thomas [15], les composants d'un système maintenu peuvent avoir trois types d'interactions : la dépendance économique, la dépendance structurelle et la dépendance stochastique. La dépendance économique intervient lorsque le coût de remplacement ou de réparation d'un ensemble de composants est moins que la somme de leurs coûts de remplacement ou de réparation individuellement. La dépendance structurelle intervient lorsque la maintenance d'un composant défaillant entraîne un arrêt de fonctionnement de certains autres composants. La dépendance stochastique est

présente lorsque l'état d'un composant peut affecter l'état des autres composants ou leurs taux de défaillance. Elle existe souvent dans les systèmes industriels. La prise en compte de ce type de dépendance dès la phase de conception permet une évaluation plus précise de la performance du système. Pour cela, nous nous concentrons dans ce manuscrit sur la dépendance stochastique qui met en jeu les défaillances dépendantes des composants.
En effet, de nombreux travaux de la littérature ont considéré que les composants du système tombent en panne indépendamment les uns des autres. Ceci a été modélisé en supposant que les variables d'états des composants sont des variables aléatoires indépendantes. Cette hypothèse simplifie considérablement la modélisation ainsi que l'analyse statistique du système [8]. Cependant, pour la plupart des cas pratiques, cette hypothèse de l'indépendance de défaillance est souvent violée [16], [17], [18].

Nous pouvons distinguer plusieurs groupes de défaillances dépendantes, parmi lesquels nous citons [8] :

- Les défaillances à cause commune (DCC) : ce sont des défaillances dépendantes dans lesquelles deux ou multiples états d'échec des composants existent simultanément, ou dans un court intervalle de temps et sont dûs à une même cause [8]. Les principales causes de ces défaillances peuvent être des conditions environnementales extrêmes (vibrations, radiations, humidité, inondation, etc.), défaillance d'un équipement externe au système, erreurs humaines, erreurs dans la conception ou l'installation du système, erreurs de maintenances [19]. La cause principale de DCC n'est pas alors la défaillance d'un autre composant dans le système. Les défaillances à cause commune ont été considérées dans plusieurs travaux tels que de Guey *et al.* [20] en 1986, Pan *et al.* [21] en 1995, Xie *et al.* [22] en 2004 , Levitin *et al.* [23] en 2013 ou encore de Yuge *et al.* [24] en 2016.
- Les défaillances en cascade, appelées aussi propagation d'échecs dans lesquelles la défaillance d'un composant se propage vers un ensemble de composants du système. Nous pouvons les trouver dans les systèmes de production, les systèmes de distribution d'électricité, etc. Plusieurs auteurs se sont intéressés à ce type de dépendance comme Chang *et al.*[25] et Eusgeld *et al.* [26] en 2011, Levitin *et al.* [23] en 2013.
- Dépendance redondante : Une dépendance parmi des composants est appelée dépendance redondante si chacun des composants est une redondance pour les autres composants. Ce concept a été défini par Yu *et al.* [27] en 2006 pour les systèmes parallèles (à composants en

redondance active). Cette dépendance considère la distribution de charge entre les composants et présente un vrai intérêt pour l'optimisation de la SDF des systèmes. Prenons l'exemple d'un système à charge répartie entre les composants redondants du système. La défaillance d'un composant peut entraîner une charge accrue sur les composants survivants et, par conséquent, une probabilité de défaillance plus élevée. La dépendance redondante a fait l'objet de plusieurs travaux [28], [29], [5], [30], [31]. Nous nous sommes intéressés dans ce travail à la modélisation de cette dépendance pour différents types des systèmes que nous présentons dans les chapitres suivants.

Il est à noter que les études récentes de la SDF accordent une grande attention aux problèmes des défaillances dépendantes. Plusieurs concepts et classifications sont proposés dans la littérature afin d'évaluer d'une manière plus concrète et précise les performances des systèmes réels. Un état de l'art plus détaillé sera fait dans le chapitre suivant.

1.4 Modélisation

La modélisation est une étape essentielle qui préoccupe l'ingénieur fiabiliste. Elle vise à analyser les systèmes en vue d'évaluer leur fiabilité/disponibilité. Un système peut être décrit par un ensemble des composants comportant des interactions entre eux afin d'assurer une fonction requise. La médélisation peut être illustrée par un diagramme de fiabilité. Celui-ci est constitué des blocs représentant généralement des composants, des sous-systèmes ou des fonctions. Elle consiste à rechercher les liens entre ces blocs [2].

1.4.1 Modèles des systèmes

Nous présentons dans cette partie les différents modèles de systèmes utilisés en sûreté de fonctionnement [2], [7], [6].

1.4.1.1 Systèmes séries

Un système est dit du type série lorsque la défaillance de l'un de ses composants entraîne la défaillance du système.

Considérons un système série à n composants (Figure 1.3). Prenons E_i l'événement « le composant i ayant une fiabilité $R_i(t)$ fonctionne sur l'intervalle $[0, t]$ ». Suite à la définition du système série, la fiabilité du système est alors donnée par la probabilité de l'événement $E_s = (E_1$ et E_2 et ... et

E_n). En d'autres termes, c'est la probabilité que chacun de ces composants soit en fonctionnement normal sur $[0, t]$.

$$R(t) = P(E_s) = \prod_{i=1}^{n} R_i(t) \tag{1.21}$$

La relation (1.21) est vraie si les n composants sont stochastiquement indépendants.

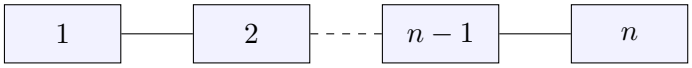


FIGURE 1.3 – Représentation d'un système série

1.4.1.2 Systèmes parallèles

Un système parallèle est un système dont la défaillance se produit si et seulement si tous ses composants sont défaillants. C'est le cas de la redondance active.
Considérons un système parallèle à n composants (Figure 1.4). Prenons E_i l'événement « le composant i ayant une fiabilité $R_i(t)$ fonctionne sur l'intervalle $[0, t]$ ». Suite à la définition du système parallèle, la fiabilité du système est alors donnée par la probabilité de l'événement $E_p = (E_1$ ou E_2 ou ... ou $E_n)$, en d'autre terme, c'est la probabilité qu'au moins un de ces composants soit en fonctionnement normal sur $[0, t]$.

$$R(t) = P(E_p) = 1 - \prod_{i=1}^{n}(1 - R_i(t)) \tag{1.22}$$

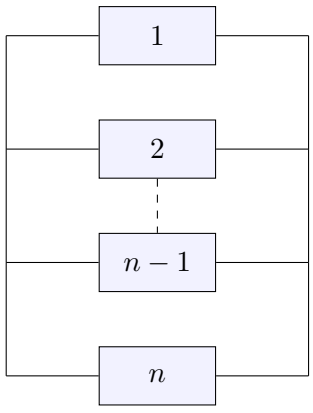


FIGURE 1.4 – Représentation d'un système parallèle

1.4.1.3 Systèmes séries-parallèles ou parallèles-séries

Un système série-parallèle est un système constitué de sous-systèmes parallèles en série. Tandis qu'un système parallèle-série est un système constitué de sous-systèmes séries en parallèle. Prenons E_{S_i} l'événement « le sous-système i fonctionne sur l'intervale $[0, t]$ ». La fiabilité du système série-parallèle est donnée par :

$$R(t) = P(E_{S_1} \text{ et } E_{S_2} \text{ et } ... \text{ et } E_{S_n}) \tag{1.23}$$

La fiabilité du système parallèle-série est donnée par :

$$R(t) = P(E_{S_1} \text{ ou } E_{S_2} \text{ ou } ... \text{ ou } E_{S_n}) \tag{1.24}$$

Il est à noter que ce sont les relations entre la défaillance du système et les défaillances des composants qui précisent les termes « séries » ou « parallèles » et non pas l'architecture réelle du système.

1.4.1.4 Systèmes redondants k/n

On distingue entre deux notions k sur n ou k/n. Ces notions sont k sur $n : G$ $(k/n : G)$ et k sur $n : F$ $(k/n : F)$ avec $k \leq n$. La première désigne un système à n composants qui fonctionne si et seulement si au moins k composants parmi les n fonctionnent. Tandis que la deuxième décrit un système à n composants qui est en panne si au moins k composants parmi n sont en panne. Une redondance partielle est alors implémentée dans ces configurations [11].
Soit X_t la variable aléatoire qui compte à l'instant t le nombre de composants qui fonctionnent dans le système. La fiabilité d'un système k sur $n : G$ s'écrit alors :

$$R(t) = P(X_t \geq k) \tag{1.25}$$

La fiabilité d'un système k sur $n : F$ est donnée par :

$$R(t) = 1 - P(X_t < n - k) \tag{1.26}$$

Il est à noter que deux types particuliers sont aussi présents : le système consécutif $k/n : G$ $(k \leq n)$, qui fonctionne lorsque k composants consécutifs parmi les n fonctionnent [32] et le système consécutif $k/n : F$ $(k \leq n)$ [33] qui tombe en panne dès que k composants consécutifs parmi les

n sont en panne. Les systèmes séries et parallèles sont des cas particuliers des systèmes k/n. Ils correspondent respectivement à $k = n$ et $k = 1$.

1.4.1.5 Systèmes à redondance passive

Un système à n composants est dit en redondance passive si un composant parmi les n fonctionne à la fois. Dès que le composant actif tombe en panne, un autre prend le relève à l'aide des commutateurs. Le calcul de la fiabilité du système est souvent simplifié en supposant que les commutateurs sont parfaits et que le démarrage d'une redondance passive est fait sans délai [34]. Considérons un système constitué de deux composants, dont un composant est en redondance passive lorsque l'autre est en service. Soit T_1 et T_2 les deux variables aléatoires correspondant aux durées de fonctionnement des composants. La fiabilité du système s'écrit alors :

$$R(t) = P(T_1 + T_2) \geq t \tag{1.27}$$

1.4.1.6 Systèmes cohérents

Considérons un système à n composants dont le dysfonctionnement dépend de l'occurence des événements dits « de base ». Désignons deux variables binaires $x_i, i = 1, ..., n$ et z_i pour représenter respectivement l'état du composant i et l'occurence de l'événement indésirable [27].

$$x_i = \begin{cases} 0, & \text{si le composant } i \text{ est défaillant.} \\ 1, & \text{si le composant } i \text{ est non défaillant} \end{cases} \tag{1.28}$$

$$z_i = \begin{cases} 0, & \text{si l'événement indésirable survient.} \\ 1, & \text{si l'événement indésirable ne survient pas} \end{cases} \tag{1.29}$$

La fonction de structure est donnée par la fonction ψ :

$$z = \psi(x), x = (x_1, ..., x_n) \tag{1.30}$$

Considérons deux vecteurs $x = (x_1, ..., x_n)$ et $y = (y_1, ..., y_n)$. On définit une relation d'ordre de la manière suivante :

$$x \leq y \Leftrightarrow \forall i \; x_i \leq y_i \tag{1.31}$$

La fonction de structure est dite cohérente si et seulement si elle vérifie la relation suivante :

$$\forall x, y \quad x \leq y \Rightarrow \psi(x) \leq \psi(y) \tag{1.32}$$

Un système ayant une fonction de structure cohérente est dit cohérent. Notons que la plupart des systèmes étudiés sont des systèmes cohérents [34]. Cependant, il existe plusieurs types de systèmes réels qui sont non cohérents. Ces derniers ont une fonction de structure qui n'augmente pas de manière monotone avec le nombre d'unités en fonctionnement. Pusieurs travaux ont étudié ce type des systèmes dans la littérature [35], [36], [37], [38].

Les systèmes non cohérents ont été modélisés et évalués initialement par Heidtmann [35]. Ils sont caractérisés par trois paramètres k, l et n et sont définis par des systèmes k-to-l-out-of-n. Ces systèmes fonctionnent lorsque pas moins que k et pas plus que l parmi les n unités fonctionnent. Notons si $l = n$, le système se réduit au système cohérent k-out-of-$n : G$. Les systèmes qui tombent en panne dans le cas où il y a une sous-production ou une surproduction d'unités ou services font partie des systèmes non cohérents. Considérons comme exemple, un système multiprocesseur k-to-l-out-of-n où les ressources telles que les bus, la mémoire et l'unité I/O sont partagés entre les différents processeurs. Ce système n'atteindra pas sa capacité maximale quand moins que k processeurs sont utilisés. D'autre part, lorsque le nombre de processeurs utilisés excède l, sa performance diminue aussi en raison de la congestion de trafic sur un bus de bande passante limitée. Ainsi, on modélise ce système en supposant qu'il échoue pour ces deux cas extrêmes. Ces modèles peuvent aussi trouver dans les systèmes de communications, les réseaux informatiques ou de transport [36].

1.4.1.7 Systèmes multi-états

Les systèmes multi-états (SMEs) sont des systèmes qui peuvent avoir différents niveaux de performance et plusieurs modes de défaillance. Le comportement de tels systèmes est modélisé alors par plus de deux états avec des niveaux de performances différents associés à chacun de ces états. Ce type de systèmes peut concerner n'importe quelle configuration structurelle série-parallèle, k sur n, etc. [39], [40]. Les systèmes binaires font un cas particulier des systèmes multi-états.

En ce qui concerne sa fonction de structure, le SME a le même concept que le système cohérent. Considérons un SME à n composants dont chacun a k états différents. L'état de composant i à l'instant t est supposé donné par $g_i(t), i = 1, ..., n.$ qui prend ses valeurs à partir des K états possibles. La performance du système est donc une fonction de l'ensemble des performances des

composants :

$$\phi(g_1(t), ..., g_n(t)) \tag{1.33}$$

Désignons par W la demande de service du système, une fonction d'acceptante F est alors définie :

$$F(\phi(g_1(t), ..., g_n(t))) \geq W \tag{1.34}$$

1.4.1.8 Autres systèmes

Dans certains cas, les systèmes peuvent prendre une différente forme des configurations précédentes (séries, parallèles, etc.) et souvent plus complexe. Lorsqu'il en est ainsi, des méthodes d'analyses plus poussées sont nécessaires pour évaluer les performances du système.

1.4.2 Techniques de modélisation

Il existe de nombreuses méthodes utilisées dans les études de SDF et qui sont destinées à l'évaluation de la disponibilité et/ou de la fiabilité des systèmes industriels. Nous nous intéressons ici aux méthodes reposant sur la représentation du comportement d'un système et ses états. Nous distinguons les méthodes sans emploi de l'espace des états (EE), avec emploi de l'espace des états (EE) et les méthodes de simulation.

1.4.2.1 Méthodes sans l'EE

Dans ce qui suit, nous présentons quelques méthodes de modélisation des systèmes sans se reposer sur l'espace des états comme les arbres de défaillance et les coupes minimales.

Arbre de défaillance

L'arbre de défaillance (*Fault Tree*), dénommé encore arbre des causes a été inventé en 1961. Cet arbre permet de représenter graphiquement les combinaisons d'événements qui conduisent à l'occurence de l'evénement indésirable (défaillance) du système [2]. Il est constitué de niveaux successifs tel que chaque événement soit obtenu à partir des événements du niveau inférieur par l'intermédiaire des portes logiques, principalement ET et OU et ses variantes. On réitère le processus jusqu'à ce que l'on aboutisse à des événements pour lesquels la recherche des causes ne s'impose plus. Ces événemnets élémentaires sont indépendants entre eux et probabilisables. Le système est considéré défaillant si le résultat de la porte la plus haute est VRAI. Un exemple simple d'un arbre de

défaillance avec six événements élémentaires est presenté par la figure 1.5 [41]. Désignons par P_i la probabilité d'occurence de l'événement i, la probabilité de l'événement X est donnée :

$$P_X = (P_1 \cap P_2) \cup (P_3 \cap P_4) \cup (P_5 \cap P_6) \tag{1.35}$$

Les arbres de défaillances ont été largement utilisés pour déterminer la sûreté de fonctionnement des systèmes. Nous citons à titre exemple, les travaux de Dhillon *et al.* [41] en 1979, Dugan *et al.* [42] en 1993, Contini *et al.* [43] en 2008 et Kabir [44] en 2017. La dernière référence récapitule les arbres des défaillances standards, leurs limitations ainsi que leurs extensions.

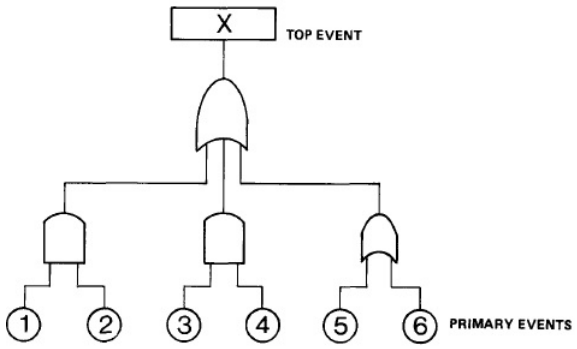


FIGURE 1.5 – Un exemple d'un arbre de défaillance

Les coupes minimales

Une coupe décrit un ensemble de blocs de base dont la défaillance aboutit à la défaillance du système. Une coupe est dite minimale si elle ne contient aucune autre coupe [2], [45]. En d'autre terme, si on retire n'importe quel bloc de la liste, le système reste en marche. L'ordre de la coupe est le nombre d'éléments dans la liste. L'intérêt de cette méthode est qu'elle permet d'établir qualitativement la liste des composants critiques, nommée aussi « les maillons faibles », dans le système. Par exemple, dans le cas d'un système série à n composants, le nombre de coupes minimales sera égale à n, une pour chaque composant. Dans le cas d'un système parallèle à n composants, on aura une seule coupe minimale formée de la liste des composants.

Autres méthodes

Parmi les méthodes sans l'EE, on peut citer encore le graphe de fiabilité [46], et la fonction

génératrice universelle [39].

1.4.2.2 Méthodes avec l'EE

Les méthodes de l'Espace d'Etat (MEE) ont été développées pour l'analyse de la sûreté de fonctionnement de systèmes réparables. Les premières utilisations dans le domaine industriel datent des années 1950 et concernaient une classe particulière de processus stochastique dits ≪ processus de Markov ≫. Ensuite, des processus non markoviens ont été introduits [47], [48].

Le principe de ces méthodes se base sur trois points [2] :

- L'identification et le classement de tous les états du système réparable en états de fonctionnement ou en états de panne. Si chaque composant a deux états (fonctionnement et panne) et si le système à n composants, le nombre maximum d'états est alors 2^n.
- Le recensement et l'identification des causes de toutes les transitions possibles entre ces différents états. En effet, au cours de la durée de vie du système, des états de panne peuvent apparaître à la suite de l'existence de défaillances de l'un ou plusieurs composants du système ou disparaître à la suite de réparation d'un composant.
- Le calcul de probabilité de système de se trouver dans les différents états au cours d'une période de sa vie.

L'espace d'état permet de construire d'un graphe des états. De tels graphes sont utilisés pour construire des processus stochastiques décrivant l'évolution du système. Ils sont constitués des sommets et des arcs. Chaque sommet symbolise un état du système. Chaque arc représente une transition entre deux sommets qu'il unit, et caractérisé par un taux de transition [7]. Par exemple, un arc orienté (i, j) est valué par le taux de transition de l'état i vers l'état j. La figure 1.6 représente le graphe d'état d'un système simple constitué d'un seul composant à deux états : fonctionnement (état 1) et en panne (état 2). Le taux de transition de l'état (1) à l'état (2) est le taux de défaillance de composant (λ). Le taux de transition de l'état (2) à l'état (1) est le taux de réparation du composant (μ).

Les différentes méthodes avec l'EE se basent sur la théorie stochastique. De ce fait, il serait utile de rappeler d'abord brièvement les processus stochastiques avant de décrire ces méthodes.

On appelle processus stochastique ou processus aléatoire toute famille de variables aléatoires X_t indexées par T définies sur un même espace de probabilité. L'indice t est souvent interprété comme le temps. A tout élément t de T, X_t correspond à l'état du processus à l'instant t [49]. Prenons l'exemple d'un système ayant $n+1$ états possibles, numérotés $0, 1, 2, ..., n$. Designons par X_t l'état

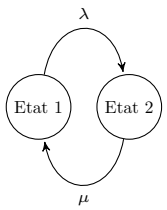


FIGURE 1.6 – Graphe d'état d'un système à 1 composant

du système à l'instant t. L'événement $[X_t = i]$ signifie que le système à l'instant t est à l'état i, $i = 0, 1, 2, ..., n$. La probabilité de cet événement est donnée par :

$$P_i(t) = P(X_t = i), i = 0, 1, 2, ..., n \tag{1.36}$$

Un cas particulier est lorsque le système à deux états : 0 (en fonctionnement) et 1 (en panne). La probabilité $P(X_t = 1)$ représente la disponibilité du système à l'instant t [28].

Processus Markoviens

Un processus est dit markovien si, à chaque instant t, son comportement futur ne dépend pas de son état passé mais seulement de son état présent [2]. De nombreux systèmes ont de transitions qui peuvent être décrits par des processus stochastiques satifaisant la propriété Markovienne. Reprenons l'exemple du système décrit précédemment ayant $n+1$ états, étant donné qu'il est à l'état i $(i = 1, ..., n)$ à l'instant t $(X_t = i)$, les futurs états X_{t+v} ne dépendent pas des états précédents X_u, $u < t$. En terme de probabilité, cette propriété Markovienne s'interprète par :

$$P(X_{t+v} = j | X_t = i; X_u = x(u); 0 \leq u \leq t) = P(X_{t+u} = j | X_t = i) \, \forall x(u); 0 \leq u \leq t \tag{1.37}$$

Le processus est une suite dite chaine de Markov à temps discret si $T \in \mathbb{N}$, et chaine de Markov à temps continu si $T \in \mathbb{R}$. La chaine de Markov est dite homogène si les probabilités de transition entre les différents états ne sont pas affectées par une translation dans le temps. Pour ces processus, les instants d'occurrence d'événements (des défaillances et/ou réparation) dans le système suivent des lois de probabilité exponentielles. D'autre part, quand la caractérisation exacte de l'état à un instant donné a besoin de l'information du temps associé, on dit que le processus est inhomogène [27]. Ce dernier permet généralement de considérer des taux de défaillance variant avec le temps. Il est à noter qu'autres processus stochastiques sont tels que le comportement futur du système depuis

un état donné ne dépend que du temps écoulé depuis l'entrée de cet état. Ce sont les processus semi-markoviens [50].

Dans ce manuscrit, nous nous intéressons aux processus markoviens homogènes en temps continu. Nous pouvons construire l'équation d'état d'un système associé à l'ensemble de ses états sous forme d'un système d'équations différentielles de Chapman Kolmogorov. Nous considérons ainsi que pour chaque état i, la probabiité que le système soit dans cet état à l'instant $(t + dt)$ est donnée par [50] :

$$\begin{aligned} P_i(t+dt) = P(&\text{système à l'état } i \text{ à l'instant } t \text{ et durant } (t+dt)) \\ &+ \sum_{j \neq i} P(\text{système à l'état } j \text{ à l'instant } t \text{ et dans l'état } i \text{ à l'instant } (t+dt)) \end{aligned} \quad (1.38)$$

Le système d'équations différentielles de Chapman Kolmogorov peut alors s'exprimer vectoriellement par :

$$\begin{cases} \frac{dP}{dt} = P(t).A, \\ P(0) \text{ connu} \end{cases} \quad (1.39)$$

où $P_t = [P_0, P_1, ..., P_i, ..., P_n]$ représente toutes les probabilités des états du PM et A est la matrice de transition du graphe d'état. La résolution de cette équation sous forme matricielle permet donc d'obtenir la probabilité $P_i(t)$ du système d'être dans l'état i à l'instant t connaissant les taux de transition du graphe et l'état de départ.

Même que les chaines de Markov induisent des hypothèses particulières quant aux distributions des processus. Cependant, cette hypothèse d'indépendance vis-à-vis des états occupés antérieurement par le système est souvent vérifiée dans les systèmes industriels. Les lois régissant le système évoluent suivant des temps exponentiellement distribués. Dans de nombreux systèmes, il serait mieux d'utiliser d'autres lois évenementielles (loi d'Erlang, de Weibull, etc.) pour représenter certains processus. Malgré cela, nous pouvons souvent retenir, en première approche, la distribution exponentielle en raison des possibilités de traitement qu'elle offre et de la robustesse naturelle de cette distribution [50].

Plusieurs travaux dans la littérature ont évalué la performance du système étudié en appliquant les processus markoviens. Chiang et al. [51] en 2001 a proposé une politique de maintenance dépendante de l'état du système. Le système considéré est soumis aux vieillissement et chocs mortels. La détérioration est qualifiée par un processus markovien continu. Yu *et al.* ont considéré

des systèmes dépendants, ils ont appliqué la chaine de Markov à temps continu afin d'évaluer la fiabilité en 2007 [28] et la disponibilté en 2014 [5] pour les systèmes étudiés.
Il est à noter qu'il existe d'autres modèles markoviens dans la littérature tels que les processus markoviens à taux de récompense [52], processus markoviens de décision [53].

Processus non Markoviens

En pratique, on distingue parmi les modèles stochastiques, les modèles markoviens et les non markoviens. Cette distinction correspond aux caractéristiques des distributions des processus stochastiques des systèmes Dans le cas où la propriété markovienne n'est pas valide à tous les instants (les distributions des processus stochastiques des systèmes, ne sont pas exponentielles), on peut se servir de l'une des trois options principales : associer des variables supplémentaires aux variables aléatoires non exponentielles, remplacer les distributions non exponentielles par des combinaisons des distributions exponentielles, ou chercher des périodes dans l'évolution du système où la propriété markovienne peut s'appliquer [27].

1.4.2.3 Méthodes de simulation

Dans ce qui suit, nous présentons quelques moyens d'évaluations de performances des systèmes utilisant la simulation comme la simulation de Monte Carlo et les réseaux de Petri.

Simulation de Monte Carlo

La simulation de type Monte Carlo (MC) consiste à simuler le comportement (exemple : fonctionnement ou panne) des composants d'un système afin d'en déduire une évaluation de sa sûreté de fonctionnement [2]. Elle peut se révéler intéressante notamment quand le système est trop grand ou trop complexe pour être résolu de manière analytique. Cette simulation se base sur le tirage au sort de nombres aléatoires. La quantité que l'on désire estimer correspond à l'espérance mathématique d'une variable aléatoire, évoluant selon un processus stochastique. L'estimation est obtenue en moyennant les résultats collectés lors d'un grand nombre d'histoires simulées du système. La précision du résultat dépend du nombre de simulations [50]. La difficulté d'application de la simulation de ce type réside donc généralement dans le grand nombre de simulations nécéssaires et dans le temps de calcul. L'estimation de mesures liées à des événements rares exige un grand nombre de simulations aboutissant à un temps de calcul prohibitif. De nombreuses techniques

d'accélération de la simulation permettent de réduire ces temps. Elles sont basées soit sur une diminution de la complexité du modèle, soit sur la réduction du nombre de scénarios à simuler, en favorisant l'apparition des événements rares [2], [14].

La simulation de type Monte Carlo peut se révéler aujourd'hui indisponsable dans des domaines aussi variés que la finance, les télécommunications, en ingénierie ou en physique, etc. [2], [54]. Par exemple, en physique des particules, elle permet grâce aux simulations probabilistes d'estimer la forme d'un signal ou la sensibilité d'un détecteur. Dans le domaine industriel où les systèmes sont grands et complexes, elle est également intéressante pour l'évaluation d'une mesure de la SDF. Donnons l'exemple des systèmes constitués par les grands réseaux électriques des différents pays, la simulation MC peut être un outil performant d'évaluation de leur fiabilité [55], [2], [56]. Cheng *et al.* [57] présente en 2009 un algorithme efficace pour calculer la fiabilité d'un système électrique qui considère la charge variable en fonction de temps et l'âge des équipements en utilisant la simulation de Monte Carlo.

Réseaux de Petri

Les réseaux de Petri sont initialement développés par Carl Adam Petri en 1962. Ils permettent de modéliser le comportement du système sans connaître à priori l'ensemble de ses états. Ils présentent des caractéristiques intéressantes telles que les mécanismes de parallélisme, de synchronisation, de partage ou d'assemblage de ressources. Le système est représenté par des places, des transitions et des jetons. Le franchissement d'une transition par un jeton correspond à un événement fonctionnel ou dysfonctionnel possible du système. Ces transitions peuvent être associées à n'importe quel type de loi probabiliste, contrairement aux graphes d'états qui supposent des transitions suivant des lois exponentielles. La configuration à jetons est nommée aussi marquage du réseau. Le marquage définit l'état du système à un instant donné. L'ensemble de transitions représente quant à lui l'ensemble des événements dont les occurrences provoquent des modifications sur l'état du système [2], [50], [14].

Il existe différents types de réseaux de Petri : temporisés, interprétés, stochastiques, colorés, continus et hybrides. Les réseaux de Petri stochastiques sont souvent utilisés pour modéliser les systèmes de production et autres systèmes soumis à des phénomènes aléatoires. Nous trouvons également des applications de réseaux de Petri dans les systèmes manufacturiers, les systèmes de télécommunications et les réseaux de transport. Des réseaux de Petri hybrides ont été adoptés par Drighiciu *et al.* [58] en 2007 pour modéliser et évaluer les performances des systèmes manufacturies considérés comme

des structures hybrides. Long *et al.* [59] ont utilisé en 2016 des réseaux de Petri stochastiques colorés et étendus (ECSPN) pour modéliser les systèmes de production dans l'industrie 4.0. L'objectif était d'analyser et d'optimiser la disponibilité ainsi que les ressources associées. Trois modèles d'ECSPN sont construits et simulés à l'aide d'un logiciel REALIST pour modéliser les interactions et l'auto-organisation des systèmes.

1.4.3 Synthèse sur les techniques de modélisation

Le choix de la technique de modélisation est une mission assez difficile pour le concepteur du système. Une étude profonde doit se faire afin trouver la plus adéquate au système étudié. On résume dans la suite les points de faiblesse des principales méthodes déjà décrites dans la section précédente [2], [60].

Les Méthodes avec emploi de l'espace des états (EE) se révèlent très utile pour l'évaluation des systèmes réparables. Leur mise en oeuvre est facile si les systèmes peuvent être modélisés par des processus markoviens et sont de petite taille. Pour les grands systèmes, l'approche markovienne est confrontée à l'explosion combinatoire du nombre des états à prendre en compte. En plus, elle suppose que les durées de vie et de réparation des composants du système suivent toutes une loi exponentielle ce qui est peu réaliste dans certains cas (durée de vie de composants mécaniques soumis à usure).

Les méthodes sans l'EE permettent une description concise et statique des systèmes mais elles ne peuvent pas traiter l'aspect temporel des pannes ou des reconfigurations comme les réparations. Une autre limitation est qu'elles ne peuvent pas représenter les dépendances, occurant généralement dans les systèmes réels. L'indépendance stochastique parmi les composants est souvent assumée.

Les méthodes de simulation telles que la simulation de Monte Carlo et les réseaux de Petri révèlent une grande importance pour la modélisation des systèmes grands et complexes. Cependant, elles sont plus compliquées à mettre en oeuvre et la traçabilité des erreurs faites lors de la modélisation n'est pas très bonne. Elles sont disqualifiées par les temps de calcul énorme lorsque des événements rares sont impliqués.

1.5 Approches d'amélioration de la sûreté de fonctionnement des systèmes

Dans le domaine de la SDF des systèmes, nous trouvons plusieurs approches utilisées pour renforcer la performance des systèmes. Kuo *et al.* [61] ont cité les options principales pour améliorer la fiabilité du système.

- Augmenter la fiabilité des composants (allocation de fiabilité),
- Ajouter des composants redondants (allocation de redondance),
- Augmenter la fiabilité des composants et ajouter des composants redondants (allocation de fiabilité et de redondance),

Les premières études de la SDF ont porté sur l'allocation de la fiabilité. Puis, très vite, les chercheurs ont commencé à traiter l'allocation de redondance et la combinaision de ces deux problèmes [62]. Il est à noter que le terme allocation ici signifie allocation par optimisation. Nous donnons dans la suite une brève description des principaux problèmes d'allocation.

1.5.1 Allocation de fiabilité

Le problème d'allocation de fiabilité consiste à déterminer la fiabilité à allouer aux différents composants d'un système. La fiabilité du système ne doit pas être inférieure à une valeur prédéfinie tout en satisfaisant des contraintes de ressources. Celles-ci sont généralement le coût, le poids, le volume.

Deux modélisations sont fréquemment rencontrées pour ce problème, à savoir la maximisation de la fiabilité, sous contraintes de ressources (formulation primale), et la minimisation du coût (ou une autre ressource) sous contrainte de fiabilité (formulation duale).

Considérons un système constitué de n sous systèmes opérant selon une structure série. Soient m ressources telles que $g_{ij}(R_j)$ désigne la consommation de la ressource i $(1 \leq i \leq m)$ par le sous-système j $(1 \leq j \leq n)$ en fonction de la fiabilité de celui-ci notée R_j. Désignons par b_i la quantité totale disponible en ressource i. Le problème d'allocation de fiabilité pour ce système, peut être formulé de la manière suivante [63] :

$$Maximiser\ R_s = \prod_{j=1}^{n} R_j \tag{1.40}$$

sous contraintes :

$$\sum_{j=1}^{n} g_{ij}(R_j) \leq b_i, 1 \leq i \leq m \tag{1.41}$$

$$R_{j,min} \leq R_j \leq R_{j,max}, 1 \leq j \leq n \tag{1.42}$$

$$0 \leq R_j \leq 1, 1 \leq j \leq n \tag{1.43}$$

Selon les hypothèses considérées, des bornes inférieures et supérieures sont parfois imposées à la fiabilité ($R_{j,min}$ et $R_{j,max}$). Dans le cas d'un système quelconque constitué de n sous-système, la fonction objective (1.40) sera remplacée par $Maximiser\, R_s = f(R_1, R_2, ..., R_n)$. L'expression de la fonction f dépend de la nature du système étudié (série, parallèle, série-parallèle, etc.). La fiabilité du sous-système j, notée R_j, peut correspondre à la fiabilité d'un composant ou être une fonction de la fiabilité de plusieurs composants [62].

Désignons par $C_j(R_j)$ la fonction coût du sous-système j qui dépend de la fiabilité des composants du sous-système j. Notons R_0 la fiabilité requise du système. Le problème de minimisation du coût sous contrainte de fiabilité pour le système série-parallèle peut être formulé par :

$$Minimiser\, C_s = \sum_{j=1}^{n} C_j(R_j) \tag{1.44}$$

sous contraintes :

$$R_s = \prod_{j=1}^{n} R_j \geq R_0 \tag{1.45}$$

$$R_{j,min} \leq R_j \leq R_{j,max}, 1 \leq j \leq n \tag{1.46}$$

$$0 \leq R_j \leq 1, 1 \leq j \leq n \tag{1.47}$$

Le problème d'allocation de fiabilité a fait l'objet de plusieurs travaux [64], [65], [66], [67]. Yalaoui et al. [64] ont étudié en 2005 le problème d'allocation de fiabilité dans un système de type série-parallèle. Le système est constitué de technologies différentes en redondance active. Ils ont d'abord proposé de méthodes de résolution pour un problème à un seul étage (un sous-système) qui sont ensuite exploitées pour le cas multi-étages. Une condition théorique d'existence d'une

solution optimale est développée. En plus, une seconde approche basée sur une fonction approximative est proposée. Kuo et Wan [65] ont passé en revue des recherches effectuées jusqu'au 2007 concernant les problèmes d'optimisation de la fiabilité et leurs méthodologies de solution. Yue *et al.* [67] ont proposé en 2015 de méthodes pour résoudre l'allocation de fiabilité des logiciels mulitiples utilisés dans les systèmes multimédias. Des contraintes budgétaires ont été prises en compte.

1.5.2 Allocation de redondance

Le problème d'allocation de redondance (RAP) vise à déterminer le nombre de composants à mettre en parallèle (active, passive ou k/n) dans le système [68]. Cette allocation se fait de manière à maximiser la fiabilité ou la disponibilité du système sous contraintes de poids, d'espace, de budget ou autres. Comme pour le problème d'allocation de fiabilité, on rencontre deux principales formulations pour ce problème : la maximisation de fiabilité ou de disponibilité sous contrainte de ressources ou la minimisation du coût sous contrainte de fiabilité ou de disponibilité.
Notons $h_{ij}(x_j)$ la consommation de la ressource i $(1 \leq i \leq m)$ par le sous-système j $(1 \leq j \leq n)$ en fonction du nombre de composants en redondance x_j. Le nombre x_j est supposé borné par une valeur minimale l_j et une valeur maximale u_j. En considérant les mêmes notations présentées précédemment, le problème d'allocation de redondance peut être modélisé de la manière suivante [61], [63] :

$$Maximiser\ R_s = \prod_{j=1}^{n} R_j(x_j) \tag{1.48}$$

$$\sum_{j=1}^{n} h_{ij}(x_j) \leq b_i, 1 \leq i \leq m \tag{1.49}$$

$$l_j \leq x_j \leq u_j, 1 \leq x_j \leq n \tag{1.50}$$

La contrainte (1.49) implique que les fonctions $h_{ij}(x_j)$ sont séparables pour chaque sous-système [61]. Dans le cas où elles ne sont pas séparables, cette contrainte s'écrit : $h_i(x_1, x_2, ..., x_n) \leq b_i, i = 1, ..., m$. Autres formulations pour ce problème ont été présentées par Misra [69]. Pour les systèmes réparables, la disponibilité est la mesure appropriée pour évaluer la performance du système. La fonction objective (1.48) peut être remplacée par la maximisation de la disponibilité $Maximiser\ A_s = \prod_{j=1}^{n} A_j(x_j)$.

Le problème de minimisation du coût sous contrainte de fiabilité peut être formulé comme suit :

$$Minimiser\ C_s = \sum_{j=1}^{n} C_j(x_j) \tag{1.51}$$

sous contraintes :

$$R_s = \prod_{j=1}^{n} R_j(x_j) \geq R_0 \tag{1.52}$$

$$l_j \leq x_j \leq u_j, 1 \leq j \leq n \tag{1.53}$$

1.5.3 Allocation combinée de fiabilité et de redondance

Le problème d'allocation de fiabilité et de redondance regroupe les deux problèmes présentés précédemment. Il s'agit d'améliorer la fiabilité du système en déterminant les niveaux de redondance et la fiabilité des composants [70], [66]. Ce problème peut être formulé pour un système quelconque comme suit [61] :

$$Maximiser\ R_s = f(R_1, R_2, ..., R_n, x_1, x_2, ...x_n) \tag{1.54}$$

sous contraintes :

$$\sum_{j=1}^{n} d_{ij}(x_j, R_j) \leq b_i, 1 \leq i \leq m \tag{1.55}$$

$$l_j \leq x_j \leq u_j, x_j \in \mathbb{N}^*, 1 \leq j \leq n \tag{1.56}$$

$$0 \leq R_j \leq 1, 1 \leq j \leq n \tag{1.57}$$

Avec les notations introduites précédemment, on note $d_{ij}(x_j, R_j)$ la consommation de la ressource i par le sous-système j en fonction de x_j et r_j (cas des fonctions séparables). D'après Kuo et Prasad [61], ce problème de programmation en nombres mixtes est très difficile à résoudre, mais semble être une situation plus réaliste que l'allocation de fiabilité ou de redondance seule. Plusieurs travaux de la littérature ont traité l'allocation combinée de fiabilité et de redondance [71], [70], [66], [72]. Kim *et al.* [72] se sont intéressés en 2017 à un nouvel aspect du problème d'allocation de fiabilité et de redondance qui prend en compte la stratégie de redondance optimale à mettre en place (active ou passive). En outre, une fonction de la fiabilité d'un sous-système redondant passif ayant un commutateur imparfait est proposée. Le problème formulé est un modèle de programmation non

linéaire à nombres entiers mixtes. Il est résolu en utilisant un algorithme génétique.

1.5.4 Autres problèmes d'allocation

Les problèmes que nous venons de présenter peuvent aussi se formuler à l'aide de modèles à deux ou plusieurs objectifs (minimisation du coût et maximisation de la fiabilité ou de la disponibilité) sous des contraintes d'espace, de non négativité des variables continues et d'intégralité du nombre des composants x_i dans chaque sous-système. Cette allocation multiobjectif est décrite dans la section suivante (1.6.2).

Dans le cas des systèmes réparables, plusieurs options peuvent être envisagées afin d'améliorer la disponibilité du système telles que la ré-allocation des composants interchangeables, l'application des actions de maintenance préventive et/ou corrective, l'allocation de pièces de rechange, la prise en considération de la dépendance des défaillances des composants en profitant de leurs répartitions des charges, l'allocation des taux de réparation et de défaillance des composants tels que la disponibilité du système soit optimisée, l'implémentation des tests, des opérations de surveillances et/ou diagnostics, etc. [69], [73], [27]. Barabady *et al.* [74] se sont intéressés en 2007 à la mesure de la contribution d'une composante individuelle à la disponibilité globale du système. Ils sont basés sur la mesure de l'indice d'importance de disponibilité d'un composant/sous-système. L'étude révèle que les mesures d'importance de disponibilité pourraient être appliquées dans l'élaboration d'une stratégie d'amélioration de la disponibilité. Le sous-système/composant ayant la plus grande valeur de mesure d'importance a le plus grand effet sur la disponibilité du système.

1.6 Optimisation

Dans ce qui précède, nous avons présenté les différents modèles et techniques d'évaluation de performances utilisés dans le domaine de la SDF. Une fois le modèle établi, il faut choisir la méthode d'optimisation associée qui va explorer l'espace de recherche. Dans nos études, nous nous intéressons aux méthodes de résolution des problèmes d'optimisation combinatoires de la fiabilité/disponibilité des systèmes.

L'optimisation combinatoire vise à minimiser ou maximiser une fonction appelée « fonction objective » d'une ou plusieurs variables de décision soumises à des contraintes. Elle consiste à trouver une meilleure solution à partir d'un ensemble très grand. Cette solution doit être réalisable et

remplit toutes les contraintes posées. Les problèmes d'optimisation combinatoires sont largement rencontrés dans le domaine de la sûreté de fonctionnement. La plupart de ces derniers, dont ceux considérés dans cette thèse, appartiennent à la classe NP-difficile. Cela signifie qu'il est très peu probable d'avoir une solution optimale avec un temps d'exécution polynomial.
Nous présentons dans la suite les principales méthodes appliquées aux problèmes d'optimisation de la fiabilité/disponibilté des systèmes de production. Un état de l'art détaillé concernant l'optimisation de la fiabilité des systèmes est présenté dans la référence [75].

1.6.1 Méthodes de résolution

Il existe deux grandes familles de méthodes de résolution des problèmes d'optimisation : exactes et approchées. La première comprend des algorithmes capables d'obtenir une solution optimale. Alors que la deuxième donne une ou un ensemble de bonnes solutions mais sans garantie d'optimalité. Il est à noter que durant ces dernières années, plusieurs outils d'optimisation dédiés ont fait leurs apparitions sur le marché. Ces outils peuvent être couplés avec autres méthodes de résolution selon le cas étudié.

1.6.1.1 Méthodes exactes

Les principales méthodes exactes utilisées pour résoudre les problèmes d'allocation de fiabilité/disponibilité et de la redondance des systèmes sont la programmation dynamique [76], [70] et la méthode par séparation et évaluation [77], [78].

Programmation dynamique

La méthode de programmation dynamique, appelée aussi optimisation récursive, a été établie par Bellman en 1957 [79]. Elle permet de résoudre des problèmes d'optimisation dont la fonction objective est la somme de plusieurs fonctions. Cette méthode consiste à décomposer un problème donné de dimension n en n sous-problèmes de dimension 1. Le nombre total des étapes à résoudre est alors égal à n. D'après Chevalier [80], la résolution d'un système de n étapes se fait en séquentielle en se basant sur des lois d'évolution et d'une décision. Pour tout processus de décision consécutif, toute sous-politique d'une politique optimale est aussi optimale. La programmation dynamique s'applique lorsqu'une équation récursive existe. Cette dernière permet de décrire la valeur optimale du critère à une étape donnée en fonction de sa valeur à l'étape précédente. Ainsi, pour un problème combinatoire initial, la méthode consiste alors à générer des sous-problèmes, à les résoudre, et à

déterminer la trajectoire optimale. Par contre, cette méthode de programmation dynamique est limitée du fait de la complexité de la génération des relations récursives, de la nature exponentielle due à la génération d'un nombre exponentiel de sous-problèmes. Elle est gourmande en mémoire.
La programmation dynamique a été adoptée par plusieurs auteurs, tels que Misra et Sharma [81], Li [76] et Yalaoui *et al.* [70]. Les auteurs dans la référence [70] se sont intéréssés en 2005 au problème d'allocation de fiabilité et de redondance dans les systèmes série-parallèle. Ils ont montré que ce problème peut être modélisé par un problème de programmation linéaire en nombres entiers. Une décomposition en sous-problèmes a été faite. Chaque sous-problème correspond à un sous-système donné. Il vise à déterminer le nombre de composants de chaque type à utiliser tout en satisfaisant un objectif de fiabilité donné à un coût minimum. Les objectifs de fiabilité pour les sous-systèmes sont déterminés grâce à l'étude du problème global. Ils sont basés sur une analogie entre les sous-problèmes et les problèmes unidimensionnels de sac à dos qui peuvent être résolus par programmation dynamique. Le problème global est ainsi résolu par la programmation dynamique.

Procédure par séparation et évaluation

La Procédure par Séparation et Evaluation (PSE ou *Branch and bound*) est une méthode arborescente qui permet une énumération intelligente de l'espace des solutions. L'énumération est réduite en éliminant toutes les classes de mauvaises solutions par calcul de bornes sur leurs fonctions d'évaluation. Cette procédure se base sur deux composantes principales : la séparation et l'évaluation. La première consiste à subdiviser le problème initial en un certain nombre de sous-problèmes qui ont chacun leur ensemble de solutions faisables. En résolvant tous ces sous-problèmes et en gardant la meilleure solution trouvée, nous garantissons l'optimalité de la solution trouvée pour le problème initial. L'ensemble de solutions et leurs sous-problèmes associés ont une hiérarchie naturelle en arbre. La deuxième vise à déterminer l'optimum de l'ensemble des solutions réalisables associé à un des noeuds de l'arbre ou, au contraire de prouver mathématiquement que cet ensemble ne contient pas de solution intéressante pour la résolution du problème étudié. Cette évaluation se fait grâce à une fonction qui permet de mettre une borne sur certaines solutions pour, soit les exclure, soit les garder comme des solutions potentielles.
Parmi les travaux qui ont appliqué les PSE pour les problèmes de la SDF, nous citons celui de Kuo *et al.* [77] en 1987 pour un problème d'optimisation de fiabilité avec contraintes, celui de Su *et al.* [82] en 1999 pour l'optimisation de la redondance d'un système série avec des contraintes sur les choix multiples.

1.6.1.2 Méthodes approchées

Dans cette partie, nous présentons quelques méthodes d'optimisation approchées qui sont basées essentiellement sur les métaheurisques. Ces dernières sont connues par leur aptitude à résoudre des problèmes NP-difficiles. De ce fait, elles sont largement appliquées dans le domaine fiabiliste. Parmi ces métaheurisques, nous citons les algorithmes génétiques, les colonies des fourmis, le recuit simulé, etc. Même que ces métaheurisques ne garantissent pas l'optimalité des solutions, mais elles présentent de nombreux avantages. Elles s'appliquent à toutes sortes de problèmes combinatoires, et elles peuvent également s'adapter aux problèmes continus. Elles permettent d'obtenir de solutions de bonne qualité pendant un temps de calcul relativement court. D'où leur intérêt dans les applications industrielles où le temps est un facteur essentiel [83].
Dans cette thèse, nous nous intéressons au développement des métaheuristiques pour résoudre les différents problèmes NP-difficiles étudiés concernant l'optimisation de la disponibilté et de la redondance des systèmes. Ces problèmes sont caractérisés par la grande taille de leur espace de recherche. Nous nous sommes basés dans notre choix sur le rôle efficace que peut jouer les métaheuristiques dans la résolution de tels problèmes complexes avec un temps de calcul relativement faible.
Nous décrivons dans la suite quelques métaheuristiques avec des exemples appliqués dans le domaine de la SDF.

Les Algorithmes Génétiques

Les algorithmes génétiques (AGs) font partie des algorithmes évolutionnaires (AEs), sont inspirés de l'évolution biologique des espèces en se basant sur le principe de Darwin. Ils ont été développés par Hollande [84] en 1975 et ont été initialement utilisés par Goldberg [85] en 1989 pour résoudre des problèmes d'optimisation [3].
Le principe peut être décrit comme suit : une population initiale d'individus de taille prédéfine est générée aléatoirement dans l'espace de recherche. Chaque individu de la population est évalué pour mesurer son degré d'adaptation à l'objectif visé. L'algorithme consiste à faire évoluer progressivement, par générations successives, la structure de la population en maintenant sa taille constante. Au cours de ces générations, la population est améliorée en favorisant la survie des individus les plus performants.
L'application des AGs consiste en plusieurs étapes. Elle commence par le codage des solutions du problème. Chaque solution ou un individu de la population est représenté par un chromosome. Le chromosome est formé des gènes. Une liste des gènes pourra être une liste des caractères numériques

(booléens, entiers, réels, etc.). Une évaluation des performances des individus est ensuite établie. Puis, la population est évoluée en passant d'une génération à la suivante. Ce passage se déroule en quatre phases : une phase de sélection, une phase de reproduction, une phase d'évaluation et une phase de remplacement. La phase de sélection consiste à choisir les individus qui participent à la reproduction. Ces individus sélectionnés nommés aussi parents subissent des opérations de croisement et de mutation pour en engendrer de nouveaux individus (enfants). C'est la phase de reproduction. Le croisement consiste à sélectionner aléatoirement des positions dans les chromosomes parents et créer, par une succession de permutations, deux enfants ou plus. Il existe plusieurs types de croisement à savoir : de type 1X en un point ou OX ou LOX en deux points. La mutation consiste à choisir aléatoirement des gènes dans un chromosome enfant et à les modifier. Les performances des nouveaux individus sont ensuite évaluées, durant la phase d'évaluation. Enfin, la phase de remplacement consiste à choisir les membres de la génération suivante. Plusieurs procédures de remplacement existent. La plus courante consiste à remplacer les individus les moins performants par les meilleurs enfants produits. L'algorithme est interrompu selon un critère d'arrêt à préciser. Ce dernier peut être un nombre fixe de générations, un temps de calcul ou un nombre de générations sans amélioration sur la qualité des résultats obtenus. Le principe de l'AG est présenté dans la figure 1.7.

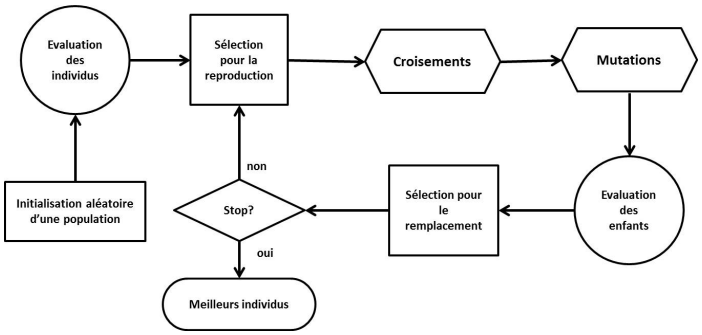


FIGURE 1.7 – Principe d'un algorithme génétique [3]

De nombreux travaux de la littérature ont recouru aux algorithmes génétiques pour résoudre les problèmes d'optimisation de la SDF des systèmes. Nous citons par exemple les travaux de Lisnianski *et al.* [86] en 1996, Elegbede et Adjallah [73] en 2003, Li et Peng [87] en 2014, Habib et al. [30] en 2016, ou encore Kim *et al.* [72] en 2017.

Les colonies de fourmis

Les algorithmes de colonies de fourmis sont des métaheuristques qui ont été introduites par Colorni, Dorigo et Maniezzo [88] en 1991 pour résoudre le problème du voyageur de commerce. Puis, ils ont été l'objet d'améliorations et ont été appliqué avec succès à d'autres problèmes d'optimisation difficile. Ces algorithmes sont inspirés des comportements collectifs de dépôt de nourriture et de suivi de piste observés dans les colonies de fourmis. Lors de l'exploitation des ressources alimentaires, chaque fourmi dépose le long de son chemin, une substance chimique, appelée « phéromone ». Cette substance sera utilisée par les autres membres de la colonie pour orienter leur marche vers le chemin le plus court.

Le principe peut être résumé comme suit. Une colonie de fourmis est un ensemble d'agents dont le déplacement représente une solution partielle du problème. Le mouvement des fourmis est basé sur deux paramètres de décision stochastiques : le taux de phéromones et le taux de désirabilité. Le premier paramètre indique l'efficacité liée à la réalisation d'un mouvement et constitue ainsi une information a posteriori de faire ce mouvement. Alors que le deuxième paramètre indique la tendance a priori suivie par une fourmi de faire ce mouvement. Au fur et à mesure de ces mouvements, chaque fourmi va construire une solution au problème, et une mise à jour et évaluation du niveau de phéromones sont effectuées. Les futures fourmis utilisent alors ces pistes de phéromone pour marquer leur trajet. Afin d'éviter une accumulation illimitée de phéromones à certains endroits, une évaporation naturelle des phéromones au cours du temps est appliquée. De ce fait, seules les bonnes solutions peuvent avoir leurs taux de phéromone en croissance une fois que toutes les fourmis ont fini leurs tours. Ainsi, la colonie de fourmis permet de trouver le plus court chemin (meilleure solution) sans que les individus aient une vision globale du trajet.

La figure 1.8 [4] représente l'expérience de sélection du plus court chemin par une colonie de fourmis. En effet, la première fourmi parcourt un chemin quelconque entre le nid (N) et la source de nourriture (F) en laissant derrière elle une quantité de phéromone, comme l'illustre la partie 1 de cette figure. Puis, les fourmis empruntent indifféremment les quatres chemins possibles. Cela est présenté dans la partie 2. A la fin de l'expérience, il est clair que la majorité des fourmis ont choisi le chemin le plus court comme le montre la partie 3. Ainsi, la quantité des phéromone présentée sur ce trajet est plus importante que celle présentée sur le chemin le plus long.

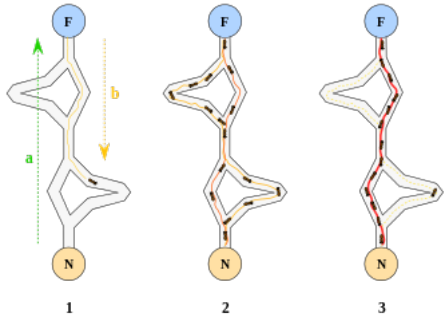


FIGURE 1.8 – Choix des plus courts chemins à parcourir par les fourmis [4]

Les algorithmes d'optimisation par colonies de fourmis ont été appliqués à un grand nombre de problèmes d'optimisation combinatoire, allant de l'affectation quadratique, au routage de véhicules, réseaux de télécommunication ou encore l'ordonnancement [3]. Récemment, ils ont été utilisés pour optimiser les problèmes de fiabilité et de redondance des systèmes sûrs. Parmi ces travaux, nous citons les papiers de Liang et Simth [89] en 2004, Bendjeghaba et Ouahdi [90] en 2008, Sharma et Agarwal [91] en 2009 et Rao *et al.* [92] en 2012.

La recherche tabou

La recherche tabou (*Tabu Search*) est une métaheuristique d'optimisation, a été présentée par Glover [93] en 1986. Elle se base essentiellement sur une exploration non triviale de l'ensemble des solutions en utilisant la notion de voisinage. Elle consiste à mémoriser les solutions ou les régions visitées dans une liste taboue, qui est généralement gérées en FIFO (*First In First Out*). De plus, elle introduit des mécanismes permettant d'interdire à la recherche de retourner trop rapidement vers une solution antérieure. Le passage d'une solution à une autre s'appelle mouvement. A chaque itération, tous les mouvements sont évalués, l'algorithme tabou choisit le meilleur voisin non tabou. Une libération de liste tabou de temps à autre est nécessaire pour assurer la diversification de la recherche. Ces étapes se répètent jusqu'à la satisfaction d'un critère d'arrêt.
Nous trouvons de nombreux travaux qui appliquent les algorithmes de recherche tabou dans le domaine de la SDF. Parmi ces travaux, nous citons ceux de Bland [94] en 1998, Kulturel *et al.* [95] en 2003, Ouzineb *et al.* [96] en 2008 ou encore Cheng *et al.* [97] en 2010.

Le recuit simulé

Le recuit simulé (*Simulated annealing*) est une métaheuristique qui a été inspirée de la physique moléculaire [98]. Elle imite le processus du recuit utilisé en métallurgie pour obtenir un état solide.

Le principe se résume par le fait de porter un métal à une température très élevée, puis le refroidir très lentement en marquant des paliers de température. Cette alternance des cycles de refroidissement et de réchauffage permet la solidification du métal en dépensant moins d'énergies. La solution initiale peut être prise au hasard dans l'espace des solutions possibles. À cette solution correspond une énergie initiale qui est calculée en fonction du critère à optimiser. Une température initiale élevée est également choisie. À chaque itération de l'algorithme, un test est fait pour vérifier si le voisin est meilleur que l'état courant. Dans le cas où il est meilleur, il remplace l'état actuel. Sinon, il devient le nouvel état courant avec une certaine probabilité qui dépend à la fois de la différence des valeurs du critère et de la température courante. Les paramètres utilisés dans le recuit simulé sont la température initiale, un nombre d'itérations de recuit où la température reste fixe, un coefficient positif de décroissance de la température et une température d'arrêt du recuit.
Les algorithmes de recuit simulé sont utilisés dans de nombreux papiers traitant de sujet d'optimisation de la fiabilité et/ou de la redondance des systèmes. Parmi ces papiers, nous citons celui de Suman [99] en 2003, de Kim et al. [100] en 2006, de Chambari et al. [101] en 2013 ou encore celui de Zaretalab et al. [102] en 2015.

1.6.1.3 Outils dédiés

Outre les méthodes de résolution décrites précédemment, il existe plusieurs procédés pour trouver une solution optimale à un problème d'optimisation. Parmi ceux-ci, nous citons les solveurs dédiés comme CPLEX, XPRESS, GAMS, LINDO, LINGO, etc. [103] pour la résolution de modèles de programmation linéaire en nombre entiers. Il est à noter que LINGO, GAMs peuvent être également utilisés pour la résolution des problèmes non linéaires en nombre entiers ou mixtes. Dans ce manuscrit, les modèles d'optimisation développés sont non linéaires. Pour cela, nous avons choisi le solveur LINGO comme une méthode de résolution supplémentaire afin d'évaluer et comparer la qualité des solutions obtenues par les différentes méthodes implémentées. L'intérêt de solveur LINGO pour la résolution des problèmes non linéaires d'optimisation de fiabilité a été montré par le papier de Sarper [104]. Sarper a résolu quatre modèles d'optimisation de fiabilité de logiciels avec des variables de décision binaires. Ces modèles ont été précédemment publiés dans la littérature. Il a montré que la résolution par le logiciel d'optimisation LINGO peut rendre inutile d'utiliser des méthodes plus compliquées telles que la programmation dynamique ou les procédures de Branch-and-Bound.

1.6.2 Optimisation multiobjectif

Dans le contexte industriel réel caractérisé par la forte concurrence économique et technologique, la plupart des problèmes de conception porte souvent sur l'optimisation simultanée de plusieurs critères, et qui sont généralement contradictoires. Nous citons à titre d'exemple, quelques objectifs souvent souhaités dans le domaine de la fiabilité : coûts minimaux, fiabilité/disponibilité maximales, risques minimaux, etc. D'où le rôle de l'optimisation multiobjectif qui consiste à optimiser simultanément plusieurs critères.

1.6.2.1 Formulation

Un problème d'optimisation multiobjectif peut être formulé comme suit [105] :

$$Optimiser\ f = [f_1(x), f_2(x), ..., f_k(x)] \tag{1.58}$$

sous contraintes :

$$g_i(x) = 0, i = 1, ..., p \tag{1.59}$$

$$h_i(x) \leq 0, i = p+1, ..., m \tag{1.60}$$

où f est le vecteur constitué de k fonctions objectives à optimiser. $x = (x_1, .., x_n)$ est le vecteur des variables de décisions à n dimensions. $g_i(x)$ représente les p contraintes d'égalité et $h_i(x)$ représente les $n - p$ contraintes d'inégalité.
Le problème consiste à trouver l'ensemble des solutions nommées « non dominées » groupées dans un front optimal. La notion de dominance est présentée dans la section suivante. Chaque solution du problème est constituée des variables de décisions qui optimisent le vecteur f des fonctions objectives (équation 1.58) tout en satisfaisant toutes les contraintes de (1.59) et (1.60).

1.6.2.2 Notion de dominance

Contrairement aux problèmes d'optimisation monocritère qui cherchent une solution optimale pour une seule fonction objective, dans le cas multicritère, on ne cherche plus à trouver une solution optimale mais un ensemble de solutions de compromis. Pour qu'une solution soit intéressante, il faut qu'il existe une relation de dominance entre la solution considérée et les autres solutions du problème.

La relation de dominance la plus utilisée pour résoudre les problèmes d'optimisation multiobjectif est la notion de dominance au sens de Pareto. Elle est définie comme suit :
Soient A et B deux solutions distinctes du problème multiobjectif de maximisation. On dit que la solution A domine la solution B, si pour tous les objectifs, A présente une performance égale ou meilleure que la performance de B et, au moins pour l'un des objectifs, la performance de A dépasse la performance de B comme le montre l'équation (1.61) [105].

$$f_i(A) \geq f_i(B), \forall i \in \{1, ..., k\} \text{ et } f_i(A) > f_i(B), \text{pour certains } i \tag{1.61}$$

Pour déterminer la relation de dominance pour un problème de minimisation, il suffit de remplacer les signes $\geq$ et $>$ dans (1.61) par $\leq$ et $<$ respectivement.
Les solutions qui dominent les autres mais ne se dominent pas entre elles sont appelées solutions optimales au sens de Pareto (ou solutions non dominées). Elles sont souvent groupées sur un seul front, appelé « front de Pareto. » Aucune de ces solutions ne peut être considérée la meilleure solution du problème [83].
Il est à noter qu'il existe une autre notion de dominance, appelée la dominance de Lorenz. Cette notion a été définie par Kostreva et Ogryczak [106] en 1999 puis étendue en 2004 par Kosterva *et. al* [107]. La dominance de Lorenz a prouvé son efficacité dans pluseurs applications [108, 109]. Dugardin *et al.* [109] ont développé un algorithme génétique à dominance de Lorenz pour l'optimisation multiobjectif des lignes de production. L'application de la dominance de Lorenz requiert plus de calculs pour chaque solution, mais elle permet de réduire considérablement la taille du front des solutions non dominées. Le front optimal de Lorenz est un sous-ensemble du front optimal de Pareto [109].

1.6.2.3 Métriques de performance

L'évaluation de la performance est un problème délicat et important dans l'optimisation multiobjectif. Dans ce cadre, plusieurs métriques ont été proposées dans la littérature. Elles peuvent être classées en deux types : absolues et relatives.
Les métriques absolues permettent de mesurer la qualité d'un ensemble de compromis (un front optimal) sans avoir besoin d'un autre ensemble de référence. Parmi ces métriques, nous citons :

- Le nombre de solutions contenues dans le front optimal n_f ou card(f) [110]
- La distance générationnelle et la déviation standarde de la distance générationnelle [83]

- L'hypervolume [110]
- L'espacement [83]
- La métrique HRS [83]

Les métriques relatives permettent de comparer deux ensembles de compromis. Ainsi, elles peuvent être adoptées pour comparer les fronts de solutions non dominées obtenues par deux algorithmes d'optimisation multiobjectif. Parmi ces métriques, nous citons :

- La métrique de Zitzler [110]
- La métrique de Laumanns [111]
- La distance de Riise [112]
- Le rapport d'erreur [83]

Autres métriques sont aussi présentées dans [83].

Dans ce manuscrit, nous appliquons trois métriques absolues et trois métriques relatives qui sont décrites dans la suite. Les premières sont le nombre de solutions non dominées, l'espacement et la métrique HRS. Les deuxièmes sont la métrique de Zitzler, la distance de Riise et le rapport d'erreur. Ces métriques sont utilisées pour évaluer et comparer les algorithmes d'optimisation multiobjectif que nous appliquons aux problèmes de conception des systèmes sûrs dans le dernier chapitre. Elles ont été appliquées dans plusieurs travaux de la littérature comme ceux d'Amodeo *et al.* [113], Dugardin *et al.* [109], Chehade *et al.* [114], ou encore celui de Alikar *et al.* [115].

Espacement

Cette métrique permet de mesurer l'uniformité de la répartition des solutions dans le plan des objectifs (f_1, f_2). Elle est calculée comme le montre l'équation (1.62) [83].

$$SP = \sqrt{\frac{\sum_{i=1}^{n_F}(d_i - \bar{d})^2}{n_F - 1}} \tag{1.62}$$

$$d_i = min_j(|f_1^i(\vec{x}) - f_1^j(\vec{x})| + |f_2^i(\vec{x}) - f_2^j(\vec{x})|), i, j = 1, ..., n_F \text{ et } i \neq j \tag{1.63}$$

avec n_F est le nombre de solutions du front obtenu et $\bar{d}$ est la moyenne de tous les d_i.

HRS

La métrique HRS (*Hole Relative Size*) permet de mesurer la taille du plus grand trou dans l'espa-

cement des points sur le front obtenu. Elle est définie par l'équation (1.64).

$$HRS = \frac{max_i d_i}{\bar{d}} \tag{1.64}$$

Métrique de Zitzler

Soient F_1 et F_2 deux fronts à comparer. La métrique de Zitzler [110] C_i ($i = 1$ ou 2) est calculée à partir de ces deux fronts, c'est un réel compris dans l'intervalle $[0, 1]$. La valeur C_1 correspond au pourcentage de solutions de front F_1 qui sont dominées par au moins une solution de front F_2. $C_1 = 1$, signifie que toutes les solutions de F_1 sont dominées (ou égales à) par des solutions de F_2. Tandis que, $C_1 = 0$ signifie qu'aucune solution de F_1 n'est dominée par une solution de F_2. D'autre part, il est à noter qu'il est nécessaire de considérer C_2. La valeur C_2 représente le pourcentage de solutions de front F_2 qui sont dominées par au moins une solution de front F_1. En conclusion, $F1$ est de meilleure qualité que $F2$ si $C_1 \leq C_2$.

Distance de Riise

Soient F_1 et F_2 deux fronts à comparer. La distance μ de Riise [112] mesure la distance entre ces deux fronts. Elle est donnée par l'équation 1.65.

$$\mu = \sum_{x=1}^{n_{F_1}} \frac{d_x}{n_{F_1}} \tag{1.65}$$

où d_x est égale à la distance entre une solution x appartenant au front F_1 et sa projection orthogonale sur le front F_2. La distance d_x prend une valeur négative si la solution x est au dessous du front (ou du front extrapolé) F_2. Le signe de μ indique si F_1 est au dessous ($\mu < 0$) ou au dessus ($\mu > 0$) de F_2.

Rapport d'erreur

Ce rapport ERR (*Error Ratio*) permet de déterminer le pourcentage de non-convergence du front obtenu par un algorithme multiobjectif ayant n_F solutions vers un front optimal. Plus cette métrique est proche de 0, plus l'ensemble des solutions a convergé vers les solutions optimales. Il est défini par l'équation (1.66).

$$ERR = \sum_{i=1}^{n_F} \frac{e_i}{n_F} \tag{1.66}$$

avec

$$e_i = \begin{cases} 0, & \text{si la solution } i \text{ appartient au front exact,} \\ 1, & \text{sinon} \end{cases} \tag{1.67}$$

1.6.2.4 Approches de résolution

Plusieurs approches sont proposées dans la littérature pour résoudre les problèmes d'optimisation multiobjectif. Elles se diffèrent notamment par la manière de traiter les fonctions objectives. Dans ce contexte, nous distinguons trois principales classes :

- Les approches aggrégées,
- Les approches non aggrégées et non Pareto,
- Les approches basées sur Pareto.

Les approches aggrégées regroupent les méthodes qui transforment le problème multi-objectif en mono-objectif en donnant ainsi une seule solution. Parmi ces méthodes, nous citons la somme pondérée [83] et la méthode $e-$contrainte [61]. La somme pondérée consiste à associer à chaque fonction objectif un coefficient de pondération, puis à faire une somme linéaire de tous ces objectifs pondérés. Les coefficients de pondération sont supposés connus à priori selon le poids que le décideur souhaite donner à chaque objectif. Alors que, la méthode $e-$contrainte consiste à retenir une fonction objective comme critère à optimiser et transformer les critères restants en contraintes. Elegebde *et al* [73] ont proposé en 2003 une méthodolgie basée sur l'algorithme génétique pour optimiser simultanément la disponibilité et le coût du système réparable parallèle-série. Dans un premier temps, ils ont transformé le problème en un problème d'optimisation mono-objectif en utilisant la technique d'agrégation pondérée. Ensuite, ils ont relaxé les contraintes par une technique de pénalité pour reformuler le problème et trouver une solution.

Les approches non agrégées et non Pareto se caractérisent par un processus de recherche qui traite séparément les objectifs à optimiser comme l'algorithme Vector Evaluated Genetic Algorithm (VEGA). Cet algorithme a été développé par Schaffer [116] en 1985. Il consiste à diviser la population, à chaque génération, en m sous-populations, où m représente le nombre d'objectifs à optimiser et associer à chacune de ces sous-populations un seul objectif. Ensuite, la méthode de sélection ne considère qu'un seul objectif et choisit les meilleurs éléments par rapport à cet objectif.

Les approches basées sur Pareto se servent de la notion de dominance de Pareto dans la sélection des solutions générées. Ces approches ont été l'objet de nombreuses études. Plusieus méthodes ont été développées qui sont basées sur les algorithmes génétiques ou autres métaheuristiques. Nous décrivons brièvement ci-dessous les principales méthodes reportées dans la littérature.

L'algorithme génétique multiobjectif (*Multi-Objective Genetic Algorithm* MOGA) [117] a été proposé par Fonseca et Fleming en 1993 et était le premier algorithme qui applique directement la notion de Pareto. Il utilise une procédure de ranking qui donne à chaque solution non dominée le rang 1 et à chaque solution dominée le rang $n_i + 1$ où n_i est le nombre de solutions qui la dominent. Les individus du même rang ont la même performance. Ensuite, un algorithme génétique avec niche de Pareto (*Niched Pareto Genetic Algorithm* NPGA) [118] a été proposé en 1994 par Horn *et al.*. Cet algorithme se base sur une sélection par tournoi dans laquelle la règle de compétition est liée à la dominance de Pareto. Dans la même année, Srinivas et Deb [119] ont implémenté la première version de l'algorithme génétique NSGA (*Non Dominated Sorting Genetic Algorithm*) qui a été initialement introduit par Goldberg [85]. Cet algorithme associe le rang 1 à tous les individus non-dominés de la population. Puis, ces individus sont enlevés de la population pour identifier l'ensemble suivant des individus non-dominés et leur attribuer le rang 2. Cette procédure est répétée jusqu'à ce que tous les individus de la population possèdent un rang. Par la suite, une nouvelle version élitiste de cet algorithme NSGA-II a été développée par Deb *et al.* [120] en 2002 qui a donné de meilleurs résultats. Dans cet algorithme, la sélection des individus se base sur le calcul de la distance de « crowding », qui estime la densité de solutions autour de chaque individu dans la population. Dans la catégorie des algorithmes élitistes, nous trouvons aussi l'algorithme SPEA (Strength Pareto Evolutionary Algorithm) qui a été introduit en 1999 par Zitzler et Thiele [110]. Il se base sur une archive externe et une technique de clustering. L'archive est utilisée pour garder un nombre limité de solutions non-dominées trouvées depuis le début de l'exécution. La technique de clustering est appliquée pour controller la taille de cet archive. Cet algorithme est ensuite amélioré par Zitzler *et al.* en 2001, menant ainsi à SPEA-II. Cet algorithme se diffère de son prédécesseur par la taille fixe de l'archive. Cette dernière sera complétée par des individus dominés si elle ne contient pas suffisamment des individus non dominés. De plus, SPEA-II tient en compte de la densité des solutions lors de l'évaluation de la performance des individus.

Les applications de ces algorithmes sont nombreuses et variées, que ce soit dans le domaine

de l'ordonnancement [121], la logistique [122] et la SDF des systèmes [123]. Le travail de Almeida [124] présente un état de l'art sur 186 travaux effectués entre 1978 et 2013 dans 16 revues scientifiques concernant les problèmes d'optimisation de la fiabilité et de la maintenance, abordés dans une perspective multicritère. Ghorabaee *et al.* [125] ont étudié en 2015 un problème d'allocation de redondance à deux objectifs d'un système série-k/n. Ils ont présenté quatre métaheuristiques basées sur NSGA-II pour résoudre le problème. Kayedpour *et al.* [126] ont utilisé en 2017 l'algorithme NSGA-II pour optimiser la fiabilité d'un système en considérant la disponibilité instantanée, les composants réparables et la sélection de la stratégie de configuration à appliquer. Dans ce manuscrit, nous proposons dans le dernier chapitre des approches d'optimisation multiobjectif basées sur les deux algorithmes NSGA-II et SPEA-II.

1.6.3 Synthèse sur les méthodes d'optimisation

Il existe différentes méthodes d'optimisation pour résoudre les problèmes d'allocation de fiabilité/redondance en mono et multiobjectif. Malgré que les méthodes exactes donnent de solutions optimales, leur utilisation dans le domaine de la fiabilité reste limitée pour plusieurs raisons. Tout changement minime du système étudié oblige à répéter quasiment toute la méthode. De plus, ces méthodes sont gourmandes en temps de calcul. Pour cela, elles sont souvent appliquées aux problèmes de petite taille. De ce fait, beaucoup de recherches ont été menées aux méthodes approchées et en partuclier les métaheuristiques. Ces dernières peuvent s'adapter facilement à tout changement. Elles sont connues par leur aptitude à résoudre des problèmes NP-difficiles et leur intérêt pour l'optimisation multiobjectif.
L'optimisation multiobjectif connaît un développement important et mérité. Elle permet de traiter des problèmes plus proches de la réalité industrielle. En fait, l'ingénieur fiabiliste se trouve souvent confronté à plusieurs objectifs et cherche à les optimiser simultanémenent. De ce fait, l'allocation multiobjectif est considérée dans notre travail, elle fait le sujet du dernier chapitre.

1.7 Problématique et périmètres d'études

Après avoir décrire le contexte général du sujet de la thèse et les différents modèles et méthodes d'optimisation utilisés en fiabilité, nous passons à présenter la problématique et les périmètres d'études.
La figure 1.9 présente un schéma qui illustre le positionnement de la problématique étudiée dans

ce mémoire. Les thèmes abordés sont encadrés en traits continus, alors que les autres thèmes de la littérature sont encadrés en traits pointillés. Nous nous intéressons dans cette thèse à la concep-

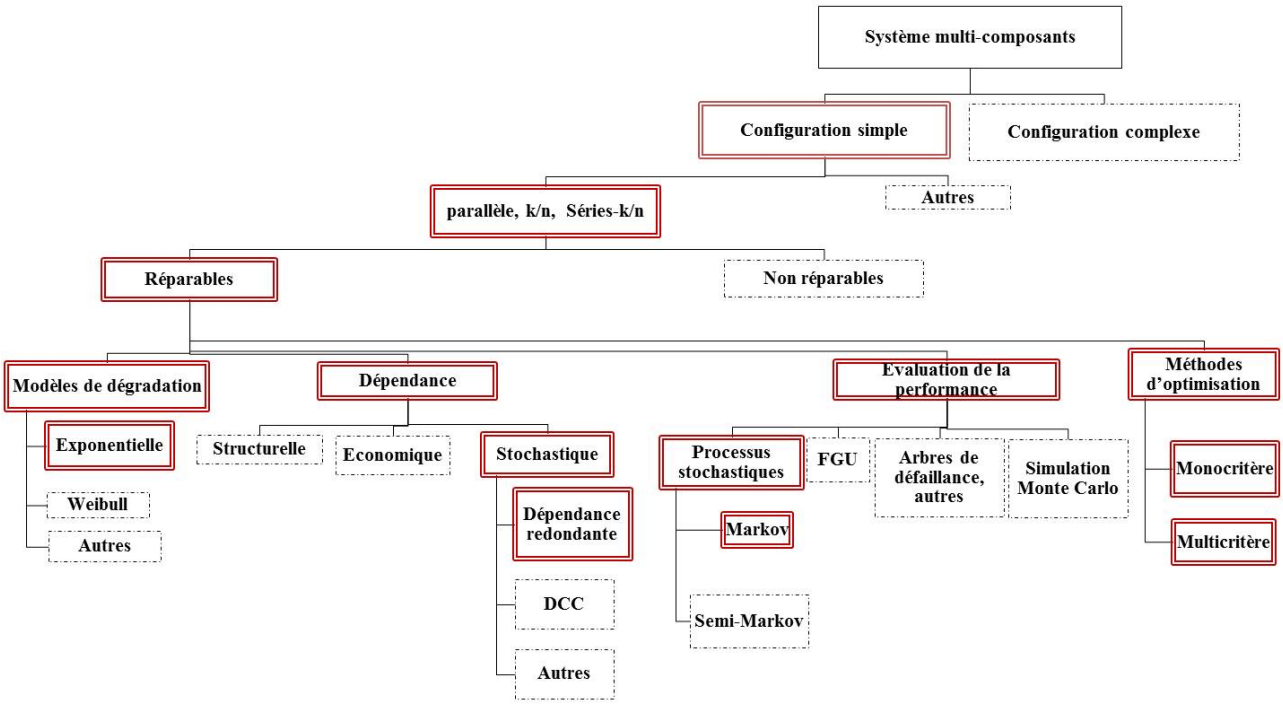


FIGURE 1.9 – Problématique

tion de systèmes de production avec la prise en compte de l'optimisation des performances à long terme. L'objectif est de développer de méthodes et approches permettant d'assurer un niveau requit de maîtrise de systèmes étudiés, en présence de défaillances dépendantes. L'expression de la fiabilité ou de la disponibilité présente une forme très spécifique en fonction des variables de décision. Nous nous sommes plus particulièrement intéréssés à l'optimisation de la disponibilité des systèmes redondants caractérisés par la présence de la dépendance redondante. Cette dépendance de défaillance est à quantifier et à intégrer dans l'évaluation de la performance fiabiliste dès la phase de conception. L'évaluation de performance est faite en utilisant les chaines de Markov à temps continu. Les problèmes considérés sont monocritères dans un premier temps, et multicritères dans un deuxième temps. Les premiers sont ceux de minimisation du coût du système sous contrainte de la disponibilité maximale et son dual, la maximisation de la disponibilité sous contrainte de coût. Les seconds sont ceux de minimisation du coût et de maximisation de la disponibilité sous

contrainte de poids. L'optimisation de la disponibilité présente un intérêt spécifique et différent en fonction des variables de décision. Dans notre étude, on considère de nombreuses variables de décision (discretes, continues). De plus, la phase de modélisation est essentielle pour une bonne formulation du problème étudié. Une fois le modèle établi sous forme d'un problème d'optimisation, il reste toujours à faire le choix d'un algorithme de résolution en tenant compte de la complexité du problème et de la décision associée. Les problèmes traités sont connus par leur nature NP-difficile. Pour les résoudre, nous proposons plusieurs approches d'optimisation basées essentiellement sur le développement de métaheuristiques.

Ainsi, la contribution de cette thèse résulte d'une part dans la modélisation des probèmes d'amélioration de la performance de systèmes, et d'autre part, dans le développement de méthodes et des approches d'optimisation.

1.8 Conclusion

Dans ce chapitre, nous avons présenté le contexte général de l'étude afin de positionner la problématique abordée dans ce mémoire. Les définitions et les notions de base de la SDF des système ont été introduites. Nous distinguons parmi ses grandeurs, la disponibilité. C'est un indicateur essentiel pour mesurer la performance des systèmes réparables à long terme de fonctionnement.

Vu que les systèmes modernes sont dynamiques et dépendants, une classifiaction de principaux types de dépendances a été présentée. La dépendance stochastique revèle un grand intérêt. Elle peut mettre en jeu les défaillances dépendantes des composants.

Comme l'objectif principal de ce travail est de modéliser et de développer des approches permettant d'optimiser la performance du système, il était important de décrire les principales méthodes de modélisation. Les modèles markoviens sont utiles pour représenter l'évolution du système ayant des processus stochastiques. De plus, les différentes stratégies d'amélioration de la SDF ont été aussi décrites en mettant l'accent sur les modèles d'allocation de fiabilité, de redondance et l'allocation combinée. Ces problèmes étant NP-difficles, de nombreuses méthodes approchées furent développées. Nous avons passé en revue les principales méthodes de résolution pour les problèmes d'optimisation mono et multiobjectifs.

Dans le chapitre suivant, nous allons étudier la conception des systèmes parallèles et k/n à composants dépendants du point de vue monoobjectif.

Chapitre 2

Optimisation des systèmes parallèles et k/n à composants dépendants

Résumé :

Dans ce chapitre, nous abordons le problème de conception des systèmes parallèles et des systèmes k sur n à composants dépendants. Nous rappelons d'abord les modèles de dépendances de défaillance existant dans la littérature en mettant l'accent sur la dépendance redondante. Dans la première partie, nous proposons une approche d'optimisation du système parallèle basée sur les algorithmes génétiques et LINGO et en tenant en compte la dépendance redondante. Elle a pour objectif la minimisation du coût du système sous contrainte d'une disponibilité exigée. La deuxième partie est consacrée à l'étude du problème de conception du système k sur n. La dépendance redondante est modélisée et quantifiée pour un tel système. Nous considérons, dans cette partie, deux problèmes d'optimisation : primal et dual. Le premier vise à minimiser le coût du système sous contrainte d'une disponibilité exigée. Alors que le deuxième a pour objectif de maximiser la disponibilité sous contrainte d'un coût. L'efficacité des approches proposées est validée via des applications numériques. Les travaux présentés dans ce chapitre ont donné lieu à deux publications dans les conférences internationales IFAC MIM 2016 *[30] et MMR* 2017 *[127] et une communication dans la conférence internationale LAAS'*16 *[128].*

2.1 Introduction

Les systèmes modernes impliquent des interactions complexes au niveau matériel, logiciel, humain, enviromental, etc. La dépendance entre la durée de vie des composants est inhérente. De plus, elle peut être un élément essentiel pour améliorer la performance des systèmes. Selon Wang *et al.* [129], la description de cette dépendance reste un grand défi pour l'ingénieur fiabiliste. Des efforts considérables sont actuellement déployés pour développer des modèles adaptés qui tiennent en compte des différents types de dépendance : économique, structurelle, stochastique, etc. Dans la section suivante, nous présentons des principaux modèles employés en SDF des systèmes qui permettent de traiter la dépendance de défaillance. Cette dernière fait partie de la dépendance stochastique. Nous nous concentrons dans notre travail sur les modèles qui exploitent directement la dépendance de charges reparties parmi les composants du système, en particulier la dépendance redondante, initiallement introduite par Yu *et al.* [28]. Dans ce contexte, nous présentons, dans ce chapitre, deux études d'optimisation qui considèrent la dépendance redondante parmi les composants du système. La première concerne les systèmes parallèles, elle est présentée dans la partie 1. Alors que la deuxième est plus générale, elle couvre les systèmes k sur n. Elle est le sujet de la partie 2.

2.2 Etat de l'art sur les modèles de dépendances de défaillance

Récemment, une grande attention a été accordée dans la littérature aux systèmes à unités dépendantes. De nombreux travaux ont étudié les systèmes multi-états à défaillances de cause commune [39], [130], les systèmes avec une propagation d'échec à effet sélectif [23], les systèmes à charge répartie [22], [131] ou autres types qui sont introduits conformément aux exigences réelles de l'industrie. Plusieurs modèles ont été proposés pour décrire différents aspects de la dépendance. Vaurio [132] a présenté deux approches qui permettent d'introduire la dépendance de défaillances dans l'analyse du système : les méthodes implicites et les méthodes explicites. Les méthodes implicites utilisent les probabilités jointes pour évaluer la corrélation de probabilité des défaillances à cause commune (DCC). Elles remplacent tous les produits $P(X_i)P(X_j)$ par la probabilité jointe $P(X_i \cap X_j)$ où X_i représente un événement de base correspondant au composant i. La dépendance, en général, induit que la probabilité jointe n'est pas égale au produit des probabilités individuelles de ses événements de base. Cette approche s'applique si les probabilités jointes $P(X_i \cap X_j)$ sont connues ou peuvent être calculées à partir des corrélations ou de probabilités conditionnelles. Les méthodes

explicites cherchent les événements de base mutuellement indépendants, puis les événements des défaillances au niveau du composant sont représentés par l'union d'événements indépendants. Ces méthodes peuvent être gérées facilement en utilisant les arbres de défaillances.

Le modèle du facteur β a été introduit par Fleming en 1974, il est couramment utilisé pour modéliser les DCCs [8]. Considérons par exemple, un système à n composants identiques dont chacun a un taux de défaillance constant λ. La défaillance d'un composant est supposée dûe à l'une des deux causes. La première concerne uniquement le composant indépendemment des autres composants, alors que la deuxième est un événement externe qui provoque la défaillance simultannée de tous les composants. Désignons par λ_I et λ_C les taux de défaillance correspondant respectivement au premier et au deuxième type. En supposant que ces deux types de défaillance sont indépendants, le taux de défaillance total du composant peut être écrit alors sous la forme de la somme de ces deux taux de défaillance.

$$\lambda = \lambda_I + \lambda_C \tag{2.1}$$

Le facteur β représente la proportion relative d'une défaillance de cause commune parmi toutes les défaillances d'un composant. Il est introduit de la manière suivante :

$$\beta = \frac{\lambda_C}{\lambda_I + \lambda_C} = \frac{\lambda_C}{\lambda} \tag{2.2}$$

Ainsi, nous aurons :

$$\lambda_C = \beta\lambda, \text{ and } \lambda_I = (1-\beta)\lambda \tag{2.3}$$

Les travaux de Fricks et Trivedi [17] en 1997 et Guo et Yang [133] en 2008 ont introduit le modèle du facteur β dans les modèles markoviens pour modéliser les DCCs.

Dans l'analyse de la fiabilité des structures, les durées de vie des composants sont généralement dépendantes. Deux composants dans un système peuvent partager la même charge ou être soumis au même stress. Ainsi, les deux variables aléatoires de la durée de vie sont liées entre elles, ou peuvent être positivement dépendantes [134]. Il existe pluseurs notions de la dépendance bivariante dans la littérature [135]. La dépendance en quadrant positif (PDQ *Positive quadrant dependent*) est la plus connue et la plus simple à appliquer. Deux variables aléatoires X et Y sont dites positivement dépendantes en quadrant si elles respectent l'inégalité suivante $P(X \leq x, Y \leq y) \geq P(X \leq x)P(Y \leq y)$ pour tous les x et y. C'est une notion de dépendance plus forte que la corrélation positive. Il est à noter que la plupart des distributions bivariantes dans la

théorie de la fiabilité sont positivement dépendantes.

Kotz *el al.* [136] ont étudié comment le degré de la corrélation entre deux composants en redondance active affecte l'augmentation de la durée de vie moyenne du système dans lequel les deux composants étaient PDQ. Une autre distribution bivariante exponentielle, nommée FGM, a été proposée suite aux travaux de Farlie, Gumbel et Morgenstern. Elle permet de décrire la dépendance de défaillance dans un système. La distribution cumulative de FGM exponentielle est donnée par [137] :

$$F(x,y) = (1-\exp(-\lambda x))(1-\exp(-\mu y))(1+\alpha\exp(-\lambda x-\mu y)) \quad |\alpha| \leq 1 \tag{2.4}$$

X et Y sont considérés comme PDQ si α positif et comme indépendants si α égal à zéro.

Hamadani et Nasrabadi [138] ont étudié en 2007 l'effet d'ajouter un composant redondant à la durée de vie résiduelle moyenne du système. Ils ont considéré deux composants dépendants pour différentes structures du système. La distribution FGM bidimensionnelle a été utilisée pour représenter la distribution conjointe de deux composants dépendants. Une comparaison avec le cas de composants indépendants a été établie. Les résultats ont montré que l'ajout d'un composant en redondance passive ou en série dans un système à composants dépendants va avoir un impact positif sur la durée de vie résiduelle moyenne du système. Cette dernière sera plus grande que celle d'un cas similaire à composants indépendants. De même, lorsque le composant est ajouté en redondance active, la dépendance a un effet positif, sauf pour la durée de vie précoce du système. Cela serait plus clair pour un paramètre de dépendance à valeur plus élevée.

Il est à noter que tous les modèles de dépendances précités sont basés sur des mesures probabilistes obtenues par des approches statistiques. Cependant, il existe autres modèles de dépendances basés sur des mécanismes déterministes relatifs à des processus stochastiques. Récemment, un grand intérêt leur est accordé du fait qu'ils permettent de modéliser le comportement dynamique du système. Dans cette catégorie de modèles, nous trouvons les cas où le taux de défaillance des composants survivants est lié aux règles diverses de la répartition de charges. Une hypothèse courante dans les systèmes multi-composants est que le taux de défaillance d'un composant survivant augmente en raison de l'augmentation de sa charge, induite par la défaillance des autres composants du système [139], [140], [129]. Les systèmes à charges réparties sont généralement trouvés dans les générateurs électriques partageant une charge électrique dans une centrale électrique, dans les unités centrales (CPUs) d'un ordinateur multiprocesseur, dans les câbles d'un pont suspendu [61]. Kvam *et al.* [141] ont modélisé la dépendance entre les composants du système à partir de la pers-

pective des charges réparties. Ils ont considéré deux exemples réels pour lesquels la règle de partage de charge pourrait s'appliquer. Le premier exemple concerne les centrales d'énergie nucléaire où les composants sont ajoutés de manière redondante aux systèmes pour protéger le noyau contre la fusion. L'échec d'un système backup pourrait affecter négativement le fonctionnement du système. Cela peut induire une augmentation significative de la probabilité de fusion. Le deuxième exemple étudie la résistance de fibre par rapport aux fibres composites dans l'industrie textile. Un paquet des fibres soumis à une charge de traction stable peut être considérée comme un système parallèle. Le taux de défaillance des fibres individuelles dépend comment la charge de ce stress global est partagée dans les fibres intactes. Pour un tel système, la règle de partage de charge dépend des propriétés physiques des fibres composites. Dans [142], la fiabilité d'un systèmes multi-d'état a été analysée en considérant la répartition de charges et la propagation d'échec à effet sélectif. Dans [143], la dépendance de défaillance a été considérée lors de l'optimisation d'un système série-parallèle. Dans [144], un système parallèle à charge répartie avec une dépendance de défaillance a été étudié. La théorie de renouvellement Markovienne a été utilisée pour obtenir la disponibilité et le temps moyen du bon fonctionnement du système jusqu'à la première défaillance (MTTF).
Selon Yu *et al.* [28], la défaillance d'un composant dans un système multi-composants peut réduire la fiabilité du système sous deux aspects : la perte de sa contribution à la fiabilité totale du système, et la reconfiguration du système sous forme de la répartition de charges parmi les composants survivants. Cette reconfiguration peut être déclenchée par les défaillances des composants ainsi que par l'ajout de la redondance. C'est dans ce contexte, que la notion de la dépendance redondante a été introduite. Nous présentons ce concept pour les systèmes parallèles dans la partie suivante.

2.3 Partie 1 : les systèmes parallèles

2.3.1 Notions sur la dépendance redondante

Yu et al. [28] ont défini la dépendance redondante dans un système en redondance active de la manière suivante : « *Une dépendance parmi des composants est appelée dépendance redondante si chacun des composants est une redondance pour les autres composants* ». Ils ont considéré un système à composants identiques en redondance active. Le système fonctionne si au moins un composant fonctionne. Lorsqu'il y a un seul composant survivant dans le système, le composant va prendre toute la charge du système. Aucune dépendance existe dans ce cas et le taux de défaillance de ce composant est supposé égal à son taux de défaillance nominal. Lorsqu'il y a plusieurs compo-

sants en fonctionnement, le taux de défaillance du composant change en fonction de la reconfiguration du système et de la répartition de charge [27]. Cette dépendance redondante est quantifiée explicitement par une fonction de dépendance, nommée $g(i)$. Elle dépend du nombre de composants survivants i dans le système. Ainsi, le taux de défaillance de composant, désigné par λ_i, est supposé déterminé par son taux de défaillance nominal λ et la fonction de dépendance $g(i)$ comme le montre l'équation (2.5) [28] :

$$\lambda_i = \frac{\lambda}{g(i)}, i \geq 2, g(1) \equiv 1, g(i) \geq 1 \tag{2.5}$$

Lorsque le système a i composants survivants, son taux de défaillance est égal à la somme des taux des défaillances de ses composants survivants. Il est déterminé par l'équation (2.6).

$$\lambda_s = \sum \lambda_i = i\lambda_i = i\frac{\lambda}{g(i)} \tag{2.6}$$

Ainsi, une relation entre le taux de défaillance du système λ_s et le taux de défaillance nominal peut être déduite (équation (2.7)).

$$\frac{\lambda_s}{\lambda} = \frac{i}{g(i)} \tag{2.7}$$

La dépendance redondante a été classifiée en quatres niveaux : indépendant ($g(i) = 1$), linéaire ($g(i) = 1$), faible ($g(i) < 1$) et fort ($g(i) > 1$). Cette classification a été illustrée par la figure 2.1.

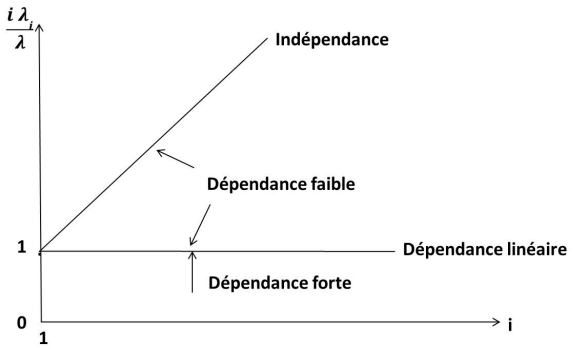


FIGURE 2.1 – Classification de la dépendance redondante pour un système parallèle [5]

2.3.2 Description du problème

L'allocation de redondance est un concept fondamental largement utilisé pour maximiser la fiabilité /disponibilité du système [61], [70], [68]. Cependant, toute amélioration de la performance

nécessite des ressources qui augmentent en général les coûts associés. Par conséquent, il est important d'optimiser la configuration du système à la phase de conception. Il a été démontré que la modélisation de la dépendance entre les composants du système peut être une option efficace pour améliorer la fiabilité du système de manière économique [28]. Mais, la résolution du problème d'optimisation par des méthodes analytiques peut être difficile et coûteuse en termes de temps de calcul, en particulier pour les systèmes complexes et/ou de grande tailles. Dans ce contexte, nous nous intéressons à l'optimisation de la conception du système redondant à composants dépendants. Nous proposons des méthodes de résolution basées sur le solveur LINGO et les algorithmes génétiques (AGs). LINGO permet d'évaluer la qualité des solutions obtenues. L'AG est utilisé pour déterminer la meilleure allocation, en raison de sa flexibilité dans la représentation de variables de décision mixtes (discrètes et continues) et sa capacité de recherche robuste. L'objectif est de déterminer les caractéristiques stochastiques des composants du système dépendant qui permettent de minimser son coût global tout en assurant un niveau requis d'une disponibilité exigée.
Nous considérons un système réparabale à n composants identiques parallèles sous les hypothèses suivantes :

- La défaillance des composants suit une loi exponentielle avec un taux de défaillance nominal λ.
- Le taux de réparation suit une loi exponentielle avec un taux de réparation μ. La réparation est supposée parfaite. Elle est faite par une seule équipe de réparation qui peut s'occuper d'au plus d'un seul composant défaillant à un moment donné.
- Le système fonctionne, si et seulement si, au moins un de ses composants fonctionne. L'évolution du système peut être décrit par les modèles markoviens en temps continu [2], [46]. Le système possède $n+1$ états possibles. Désignons par λ_{n-i} le taux de défaillance du composant lorsqu'il y a i composants défaillants dans le système. Dans ce cas, le système a $(n-i)$ composants en fonctionnement.

Nous nous intéressons à la disponibilité stationnaire du système A_s du fait qu'elle permet d'évaluer la performance du système à long terme. Elle est obtenue comme le montre l'équation (2.8) en utilisant les chaines de Markov :

$$A_s(n,\lambda,\mu) = 1 - \frac{\prod_{i=0}^{n-1} \frac{(n-i)\lambda_{n-i}}{\mu}}{1 + \sum_{j=1}^{n} \prod_{i=0}^{j-1} \frac{(n-i)\lambda_{n-i}}{\mu}} \tag{2.8}$$

Considérons la dépendance redondante entre les composants du système. Le taux de défaillance des composants dépend alors du nombre de composants survivants dans le système et de la fonction de la dépendance redondante g comme le montre l'équation (2.6). En se basant sur les équations (2.6) et (2.8), la disponibilité du système redondant à composants dépendants est formulée selon l'équation (2.9) [5].

$$A_s(n,x) = 1 - \frac{\prod_{i=1}^{n} \frac{ix}{g(i)}}{1 + \sum_{j=1}^{n} \prod_{i=n-j+1}^{n} \frac{ix}{g(i)}} = 1 - \frac{1}{1 + \sum_{k=1}^{n} \prod_{m=1}^{k} \frac{g(m)}{mx}},\ x = \frac{\lambda}{\mu} \tag{2.9}$$

2.3.3 Modèle mathématique

La fonction objective du problème est formulée comme le montre l'équation (2.10). Elle consiste à diminuer le coût total du système, soit C_s. Ce coût comprend le coût d'achat $C(\lambda)$ et le coût de réparation des composants $C_M(\mu)$. La disponibilité du système A_s doit être plus grande ou égale à une disponibilité exigée A_0 comme le présente la contrainte (2.11). Les variables de décision sont le nombre de composants redondants n, le taux de défaillance nominal λ et le taux de réparation des composants μ. Elles sont bornées d'après les contraintes (2.12), (2.13) et (2.14). n_L, λ_L et μ_L sont respectivement leurs valeurs inférieures. n_U, λ_U et μ_U correspondent respectivement à leurs valeurs supérieures.

$$Minimiser\ C_s = nC(\lambda) + C_M(\mu) \tag{2.10}$$

sous contraintes :

$$A_s(n,x) \geq A_0,\ \ x = \frac{\lambda}{\mu} \tag{2.11}$$

$$n_L \leq n \leq n_U \tag{2.12}$$

$$\lambda_L \leq \lambda \leq \lambda_U \tag{2.13}$$

$$\mu_L \leq \mu \leq \mu_U \tag{2.14}$$

Le modèle de coût utilisé dans [5] est adopté. Il est illustré par l'équation (2.15).

$$C(\lambda) = \frac{1}{1 - e^{-\frac{\lambda}{1+p\theta}}} - 1, \quad C_M(\mu) = \mu^q,\ \ p \geq 0, q > 0 \tag{2.15}$$

où p et q permettent de varier respectivement le coût du composant et sa vitesse de réparation. θ représente le paramètre de dépendance. Il est introduit dans le coût du composant. Ce dernier croît en fonction de l'intensité de la dépendance [5].

2.3.4 Méthodes de résolution

Le problème d'optimisation étudié est un problème mixte-entier non linéaire (*Mixed integer non linear problem* MINLP). La fonction objective ainsi que la contrainte de la disponibilité sont non linéaires. En termes de variables de décision, n est entier alors que λ et μ sont réels. Pour résoudre ce problème, nous avons utilisé le solveur LINGO qui permet de résoudre efficacement un modèle d'optimisation non linéaire [25]. De plus, nous avons implémenté les algorithmes génétiques en raison de leur capacité de recherche rapide et de leur flexibilité dans la représentation de variables de décision mixtes.

2.3.4.1 LINGO

LINGO a été conçu pour résoudre efficacement des modèles d'optimisation linéaires, non linéaires et entiers [25]. Il a été mis en place par Ghorabaee et al. [125] pour résoudre de tels problèmes de programmation mathématique. Il est à noter que dans le modèle d'optimisation étudié, la formule non linéaire de la disponibilité comprend une somme basée sur la variable de décision n. Cela exige une reformulation pour que cette contrainte soit intégrée dans LINGO. Un vecteur y d'éléments binaires est alors introduit pour remplacer la variable de décision n. Ainsi, le modèle est reformulé comme suit :

$$Minimiser\ C_s = \left(\sum_{i=1}^{n_U} y(i)\right)\left(\frac{1}{1-e^{\frac{-\lambda}{1+p*\theta}}}-1\right)+\mu^q, \tag{2.16}$$

sous contraintes :

$$\sum_{i=1}^{n_U}\left(y(i)\prod_{m=1}^{i}\frac{g(m)}{mx}\right)\geq\frac{A_0}{1-A_0}, \tag{2.17}$$

$$n_L \leq \sum_{i=1}^{n_U} y(i) \leq n_U \tag{2.18}$$

$$\lambda_L \leq \lambda \leq \lambda_U \tag{2.19}$$

$$\mu_L \leq \mu \leq \mu_U \tag{2.20}$$

$$y(i) \in \{0,1\}\ \forall i \tag{2.21}$$

L'objectif est de minimiser le coût du système comme l'indique l'équation (2.16). Nous remplaçons la variable de décision n par la somme de $y(i)$. La disponibilité du système déjà définie dans (2.9) est reformulée, la somme basée sur n est remplacée par la somme des éléments de vecteur y comme l'illustre la contrainte (2.17). La contrainte (2.18) révèle que la somme des éléments de vecteur y est bornée entre le minimum et le maximum du nombre de composants alloué dans le système. Les contraintes concernant λ (équation (2.19)) et μ (équation (2.20)) restent les mêmes que dans le modèle précédent. La contrainte (2.21) assure que les éléments $y(i)$ de vecteur y sont binaires.

2.3.4.2 Algorithmes Génétiques

Les algorithmes génétiques (AGs) sont l'une des principales métaheuristiques qui ont montré une bonne efficacité pour la résolution des problèmes d'optimisation fiabiliste [123], [68], [72]. Ils n'exigent pas une formulation mathématique rigoureuse. De plus, ils peuvent être adatptés facilement aux problèmes étudiés. De ce fait, nous les adoptons dans ce chapitre. Nous présentons ci-dessous les différentes étapes de l'AG et son application à notre problème.

Codage des solutions

Chaque solution du problème d'optimisation est représentée par un chromosome constitué par des gènes. Ces gènes sont les variables de décision à déterminer : n, λ et μ. Elles sont générées directement dans leur espace de recherche. Ce codage, appelé phénotype, est adapté. Il existe un autre type de codage, le codage binaire qui a été testé aussi. Nous ne l'avons pas choisi comme il a requis un temps de calcul plus important sans apporter une amélioration significative des résultats.

Génération de la population initiale

Une population initiale de taille N_s est générée de manière aléatoire dans l'espace de recherche. Elle constitue les solutions initiales.

Evaluation des solutions

Chaque solution x de la population est évaluée grâce à sa fonction objective, notée par $F(x)$. Cette dernière est égale à $C_s(x)$ si la solution x est faisable, c-à-d, elle satisfait la contrainte sur

la disponibilité requise ($A_s(x) \geq A_0$). Sinon, un grand nombre positif M est ajouté à $F(x)$ pour pénaliser cette solution. $F(x)$ est évaluée comme le montre l'équation (2.22).

$$F(x) = C_s(x) + M \times max\{0, A_0 - A_s(x)\}, M \text{ un grand nombre positif.} \quad (2.22)$$

Reproduction et mutation

Deux chromosomes (Parents) sont sélectionnés aléatoirement et sont soumis ensuite à des opérations de croisement avec une probabilité p_c. Il existe plusieurs types de croisement, simple ou multiple. De nombreuses opérations de croisement ont été réalisées. La meilleure solution a été obtenue par le type 1X. Un point de croisement est choisi aléatoirement. Ensuite, les premières parties de chromosomes des parents sont échangées pour créer deux nouveaux individus (enfants).

Ensuite, ces enfants subissent une mutation avec une probabilité p_m pour créer une diversification dans la population et pour éviter de tomber dans un optimum local. Cette opération consiste à altérer légèrement les valeurs des gènes. Dans ce travail, nous choisissons aléatoirement un gène du chromosome et on le remplace par une valeur aléatoire sélectionnée dans son espace de recherche. Ce processus de sélection, de croisement et de mutation est répété $N_s/2$ fois.

Remplacement et critère d'arrêt

Les enfants récemment obtenus par la mutation sont évalués via leur fonction objective. Ensuite, les mauvais individus de la population sont remplacés par les enfants ayant les meilleures performances. Cela constitue une génération. Nous avons choisi comme critère d'arrêt de l'algorithme un nombre fixe de générations égal à N_g. La procédure décrite ci-dessus se répète alors pour N_g générations. La meilleure solution obtenue au cours de toutes ces générations est la solution du problème.

Réglage des paramètres

Le réglage des paramètres a été fait en se basant sur un plan d'expérience. Différentes valeurs des paramètres ont été considérées : $p_c = (0.7, 0.8, 0.85, 0.9)$, $p_m = (0.03, 0.05, 0.1, 0.15)$, $N_s = (50, 80, 100, 150)$, $N_g = (50, 100, 150, 200)$. Plusieurs tests ont été effectués afin de trouver la meilleure valeur de chaque paramètre. Dans chaque test, la valeur d'un seul paramètre est changée alors que les autres restent fixes. Les valeurs finales sont celles qui assurent un compromis entre la qualité des solutions et le temps de convergence. Elles sont présentées dans le tableau 2.1.

Tableau 2.1 – Paramètres de l'algorithme génétique

Paramètre	Valeur
p_c	0.8
p_m	0.03
N_s	100
N_g	200

2.3.5 Applications numériques

Cette partie présente les résultats des applications numériques en utilisant l'AG et LINGO pour résoudre le problème d'optimisation d'un système parallèle à composants dépendants. Nous avons adopté un exemple de la littérature [28] en augmentant l'espace de recherche afin de tester l'efficacité des méthodes de résolution proposées. Nous avons considéré la fonction de dépendance présentée dans l'équation (2.23) [28] où θ est le paramètre de dépendance. Il représente l'intensité (niveau) de dépendance entre les composants redondants. Les quatres classes de dépendance sont obtenues comme suit [28] : indépendance pour $\theta = 0$, dépendance linéaire pour $\theta = 1$, dépendance faible pour $0 < \theta < 1$ et dépendance forte pour $\theta > 1$.

$$g(i) = \theta.i + (1 - \theta) \tag{2.23}$$

Afin d'évaluer et de comparer les deux méthodes de résolution, nous proposons dans la partie suivante un protocole de tests avec différentes instances.

2.3.5.1 Protocole de tests

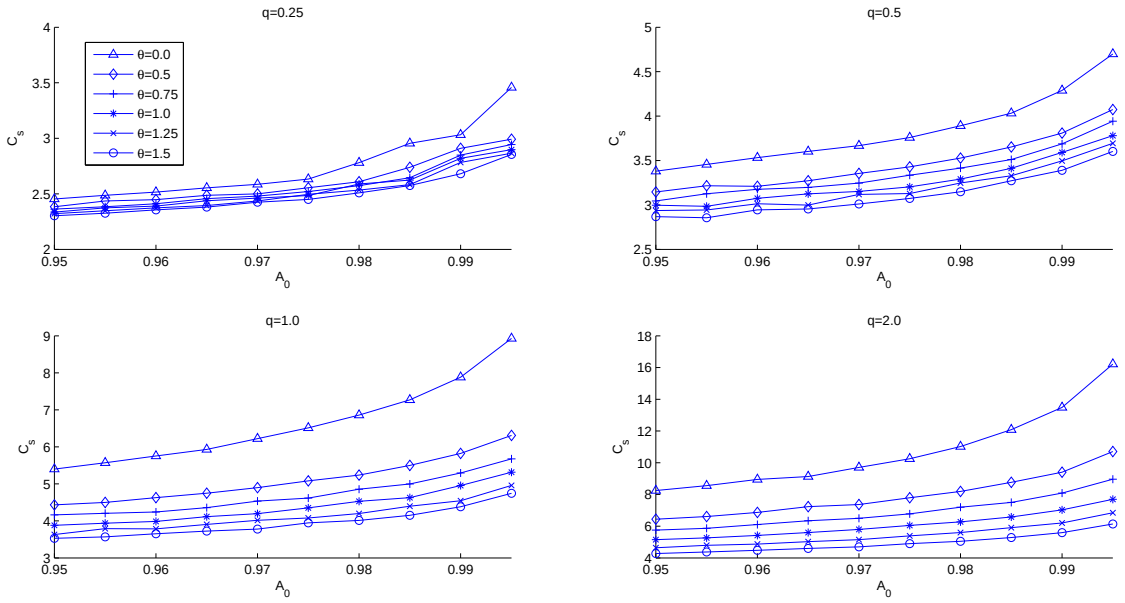
Dans ce protocole, nous effectuons des évaluations sur plusieurs configurations. Chaque configuration diffère de l'autre par un changement de l'un des paramètres du problème étudié. Les paramètres sont p et q qui sont introduits dans la fonction de coût, l'intensité de la dépendance représentée par le paramètre θ et la disponibilité exigée du système A_0. 4 valeurs de p, q et θ sont considérées : $p \in \{0.0, 0.25, 0.5, 1.0\}$, $q \in \{0.25, 0.5, 1.0, 2.0\}$, $\theta \in \{0.0, 0.5, 1.0, 1.5\}$. Pour chaque configuration, le problème d'optimisation est résolu par LINGO et l'AG pour 10 valeurs de la disponibilité exigée A_0, $A_0 \in \{0.95, 0.955, 0.96, 0.965, 0.97, 0.975, 0.98, 0.985, 0.99, 0.995\}$. Ainsi, le nombre total de tests effectués est $4 \times 4 \times 4 \times 10 = 640$. Pour chacun d'eux, le coût total du système obtenu par LINGO, C_{sLINGO}, et l'AG, $C_{s_{AG}}$, est calculé. De plus, leur temps d'exécution T_{LINGO} et T_{AG} sont aussi évalués. Nous calculons, par la suite, le pourcentage de déviation relative de

coût ΔC et de temps d'éxecution ΔT entre ces deux méthodes de résolution comme le montre respectivement les équations (2.24) et (2.25). Nous divisons leur variation en 5 catégories pour une meilleure analyse des résultats comme l'illustre le tableau 2.2. En termes de déviation de coût, le pourcentage de la distribution de solutions obtenues dans chacune des catégories est évaluée. Les tableaux 2.3, 2.4 et 2.5 montrent respectivement cette distribution pour chaque combinaision de (p, q), pour chaque θ et pour la totalité des tests effectués. En termes de déviation de temps d'exécution, la même stratégie d'analyse est appliquée. Les résultats sont illustrés dans les tableaux 2.6, 2.7 et 2.8.

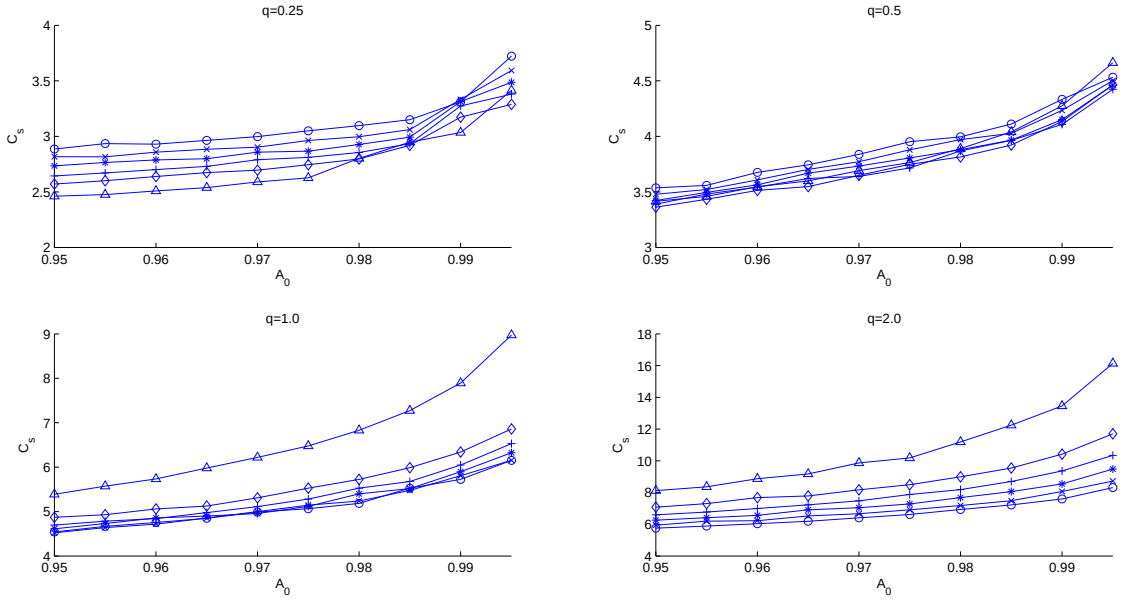
$$\Delta C(\%) = \frac{C_{s_{AG}} - C_{s_{LINGO}}}{C_{s_{LINGO}}} \times 100 \tag{2.24}$$

$$\Delta T = \frac{T_{LINGO}}{T_{AG}} \tag{2.25}$$

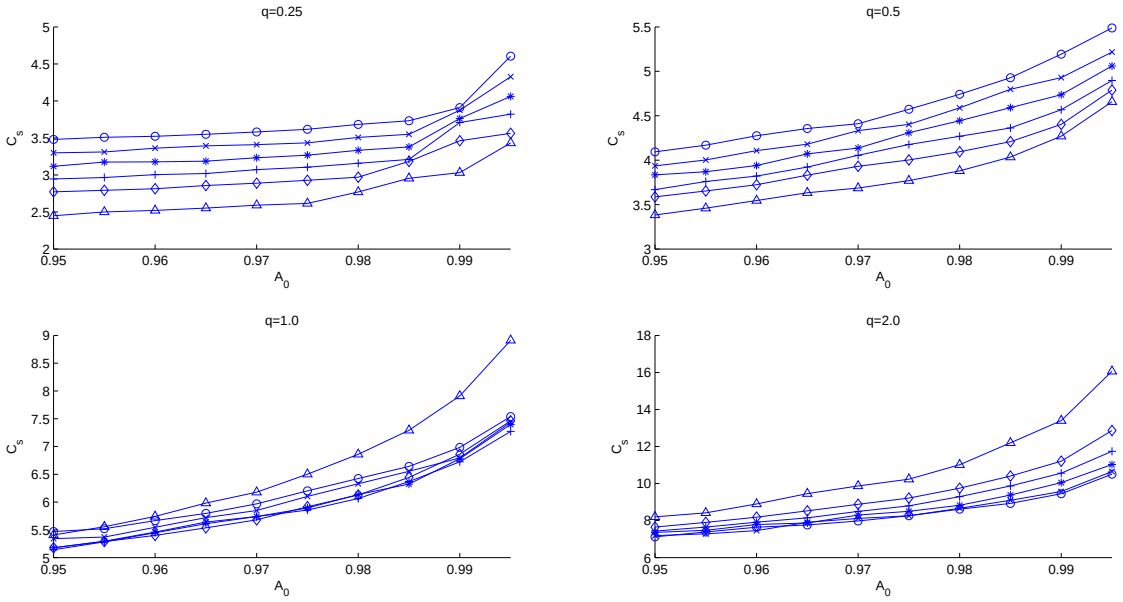
De plus, nous présentons les résultats de l'AG sous forme de graphes comme le montre la figure 2.1 pour les différentes combinaisons de p et q. Afin de mettre l'accent sur la dépendance redondante et son effet sur la performance du système, le problème d'optimisation a été résolu pour 6 valeurs du paramètre de dépendance θ. Les symboles $\{\triangle, \diamondsuit, +, *, \times, \circ\}$ correspondent respectivement aux valeurs $\{\theta = 0.0, 0.5, 0.75, 1.0, 1.25, 1.5\}$.


q=0.25
q=0.5
q=1.0
q=2.0
θ=0.0
θ=0.5
θ=0.75
θ=1.0
θ=1.25
θ=1.5
C_s
A_0

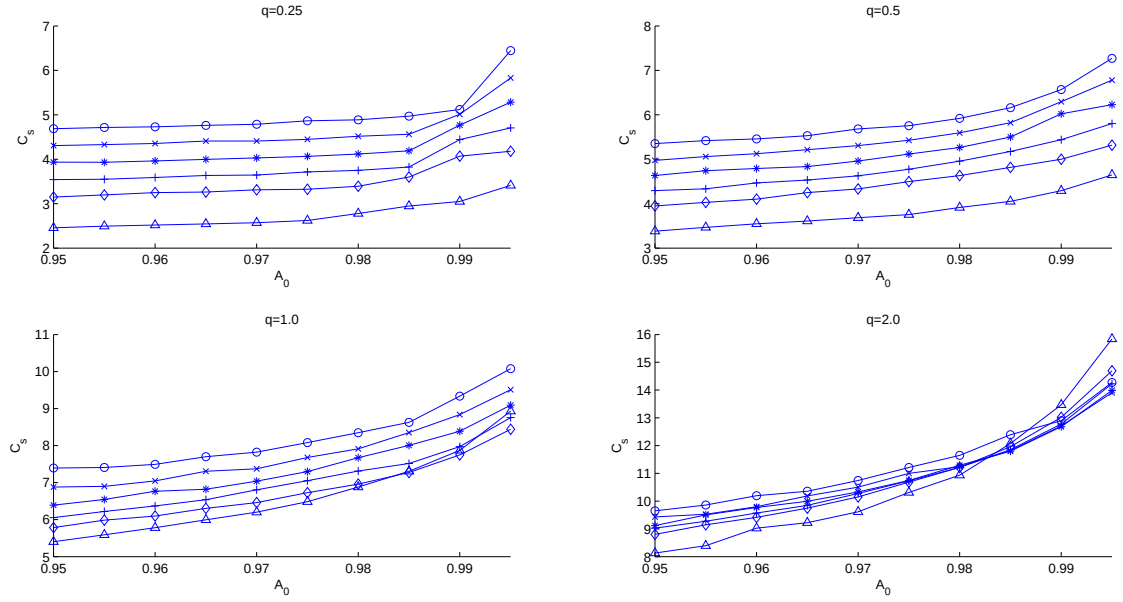
(a) p=0.0


q=0.25
q=0.5
q=1.0
q=2.0
C_s
A_0

(b) p=0.25



(c) p=0.5



(d) p=1.0

FIGURE 2.1 – Coût du système parallèle pour différents niveaux de la dépendance redondante

Tableau 2.2 – Différentes catégories de la variation de ΔC et ΔT

	Catégories	Nomenclature
ΔC	0%	$C1$
	$0\% < \Delta C \leq 1\%$	$C2$
	$1\% < \Delta C \leq 2\%$	$C3$
	$\Delta C > 2\%$	$C4$
	Aucune solution fournie par LINGO	N/A
ΔT	$0 < \Delta T \leq 1$	$T1$
	$1 < \Delta T \leq 5$	$T2$
	$5 < \Delta T \leq 10$	$T3$
	$\Delta T > 10$	$T4$
	Aucune solution fournie par LINGO	N/A

Tableau 2.3 – Distribution de solutions (%) en termes de variation de ΔC pour toutes les combinaisons testées (p, q)

(p, q)	$C1$	$C2$	$C3$	$C4$	N/A
$(0.0, 0.25)$	0	87.5	7.5	5	0
$(0.0, 0.5)$	0	47.5	30	22.5	0
$(0.0, 1.0)$	2.5	42.5	32.5	22.5	0
$(0.0, 2.0)$	0	27.5	15	57.5	0
$(0.25, 0.25)$	5	75	10	10	0
$(0.25, 0.5)$	2.5	87.5	10	0	0
$(0.25, 1.0)$	0	65	32.5	2.5	0
$(0.25, 2.0)$	2.5	37.5	35	25	0
$(0.5, 0.25)$	0	82.5	12.5	0	5
$(0.5, 0.5)$	0	77.5	17.5	5	0
$(0.5, 1.0)$	0	52.5	32.5	15	0
$(0.5, 2.0)$	0	45	20	35	0
$(1.0, 0.25)$	0	80	10	10	0
$(1.0, 0.5)$	2.5	62.5	15	20	0
$(1.0, 1.0)$	2.5	40	27.5	30	0
$(1.0, 2.0)$	2.5	52.5	35	10	0

2.3.5.2 Analyse des résultats

La figure 2.1 révèle que dans le cas $p = 0.0$, où le coût des composants est indépendant de la dépendance redondante, le coût du système diminue avec la croissance de l'intensité de la dépendance redondante entre les composants. Dans les autres cas, lorsque le coût du système prend en compte la dépendance, une étude plus approfondie devrait être réalisée pour déterminer les meilleurs paramètres permettant d'assurer un bon compromis entre le coût du système et la

Tableau 2.4 – Distribution de solutions (%) en termes de variation de ΔC pour les différents niveaux de dépendance représentés par θ

θ	$C1$	$C2$	$C3$	$C4$	N/A
0.0	0	65	18.1	16.9	0
0.5	2.5	70	18.75	9.375	1.25
1.0	2.5	59.375	22.5	15.625	0
1.5	1.25	48.75	26.875	23.125	0

Tableau 2.5 – Distribution de solutions (%) en termes de variation de ΔC

$C1$	$C2$	$C3$	$C4$	N/A
1.25%	60.15625%	21.40625%	16.875%	0.3125%

Tableau 2.6 – Distribution de solutions (%) en termes de variation de ΔT pour toutes les combinaisons testées (p, q)

(p, q)	$T1$	$T2$	$T3$	$T4$	N/A
$(0.0, 0.25)$	12.5	62.5	17.5	7.5	0
$(0.0, 0.5)$	50	42.5	5	2.5	0
$(0.0, 1.0)$	2.5	92.5	5	0	0
$(0.0, 2.0)$	2.5	50	35	12.5	0
$(0.25, 0.25)$	0	65	25	10	0
$(0.25, 0.5)$	0	72.5	17.5	10	0
$(0.25, 1.0)$	0	72.5	20	7.5	0
$(0.25, 2.0)$	5	60	35	0	0
$(0.5, 0.25)$	22.5	60	7.5	5	5
$(0.5, 0.5)$	2.5	65	22.5	10	0
$(0.5, 1.0)$	2.5	75	22.5	0	0
$(0.5, 2.0)$	0	35	35	30	0
$(1.0, 0.25)$	12.5	62.5	22.5	2.5	0
$(1.0, 0.5)$	27.5	35	32.5	5	0
$(1.0, 1.0)$	7.5	67.5	25	0	0
$(1.0, 2.0)$	0	37.5	50	12.5	0

Tableau 2.7 – Distribution de solutions (%) en termes de variation de ΔT pour les différentes valeurs de θ

θ	$T1$	$T2$	$T3$	$T4$	N/A
0.0	0.6	32.5	53.8	13.1	0
0.5	9.375	55	23.125	11.25	1.25
1.0	6.25	71.875	18.125	3.75	0
1.5	19.375	75	2.5	3.125	0

Tableau 2.8 – Distribution de solutions (%) en termes de variation de ΔT

$T1$	$T2$	$T3$	$T4$	N/A
9.21875%	59.6875%	23.59375%	7.1875%	0.3125%

disponibilité requise. De plus, il est noté que les composants redondants ayant une dépendance suffisamment forte sont plus recommandés dans les cas où le coût de réparation croît rapidement ($q > p$) ou lorsqu'une grande valeur de la disponibilité du système est exigée. Ces résultats sont cohérents avec ceux obtenus par [5], où une méthode d'optimisation analytique a été appliquée.
Les résultats obtenus dans les différents tableaux permettent de distinguer les cas où la résolution du problème d'optimisation par l'AG est plus avantageuse que l'utilisation de LINGO et vice versa. Par exemple, d'après le tableau 2.3, pour $p = 0.0$ et $q = 0.25$, 87.5% des tests effectués ont donné des solutions ayant une déviation du coût $\Delta C \leq 1\%$ (catégorie C_2) et $\Delta T > 1$ (correspondant aux catégories T_2, T_3 et T_4). Par conséquent, dans ce cas, l'AG domine LINGO en termes de temps d'exécution avec un coût légèrement moins optimal. Alors que, pour $p = 0.0$ et $q = 2.0$, 57.5% des solutions obtenues avaient $\Delta C > 2\%$ (catégorie C_4). Par conséquent, LINGO est plus recommandé ici. Il est à noter que LINGO n'a pas pu traiter toutes les instances. Il existe de nombreux cas où aucune solution n'est trouvée pendant un temps d'exécution prédéfini (10800 secondes). D'après le tableau 2.5, dans environ 0.3% de la totalité des tests, aucune solution n'est donnée par LINGO (catégorie N/A). En outre, à partir des tableaux 2.4 et 2.7, on peut constater que l'AG est plus recommandé que LINGO dans plus de 59% des tests (catégorie C_2 où $\Delta C \leq 1\%$) lorsque les composants du système sont indépendants ($\theta = 0.0$), faiblement dépendants ($\theta = 0.5$) ou linéairement dépendants ($\theta = 1.0$). Pour une dépendance redondante forte ($\theta = 1.5$), les résultats de LINGO peuvent être meilleurs que ceux donnés par l'AG. Les tableaux 2.5 et 2.8 donnent un aperçu général comparatif des résultats obtenus par l'AG et LINGO. Dans 1.25% des tests, l'AG a donné des solutions égales à celles obtenues par LINGO (catégorie C_1). Dans environ 60% des solutions obtenues, la déviation du coût était inférieure ou égale à 1% (catégorie C_2). En outre, dans environ 90% de la totalité des tests, l'AG a trouvé des solutions très proches de celles données par LINGO avec un temps d'exécution plus rapide (catégories T_2, T_3, T_4 et N/A).
L'approche proposée a prouvé l'utilité de considérer la dépendance redondante à la phase de conception. Elle est rapide et peut être efficace pour les applications dans lesquelles le temps d'exécution a une priorité importante. Il serait intéressant de l'appliquer aux systèmes plus générales en termes de configuration et de tester son efficacité. C'est ce qui constitue le sujet de la partie suivante.

2.4 Partie 2 : les systèmes k sur n

Dans cette partie, nous nous intéressons aux systèmes k/n : G à composants dépendants. Cette configuration est largement adoptée pour de nombreux systèmes redondants à tolérance de panne. Ils peuvent être rencontrés dans les systèmes multi-moteurs utilisés dans les avions, dans les systèmes multi-affichage d'un poste de pilotage, dans les services militaires, industriels et les systèmes de communication [145], [22]. Plusieurs travaux de la littérature ont traité les problèmes d'allocation de fiabilité/disponibilité de ces systèmes [22], [29], [24], [40], [11]. Nous décrivons d'abord ces systèmes. Puis, nous présentons une extension, généralisation, du concept de la dépendance redondante aux systèmes k sur n.

2.4.1 Description du système

Un système k/n : G est considéré avec une réparation parfaite à n composants binaires identiques dont la défaillance des composants et la réparation suivent une loi exponentielle. L'équipe de réparation s'occupe d'au plus un composant défaillant s'il y en a un. Le système fonctionne si et seulement si au moins k parmi ces n composants fonctionnent. Il possède $n-k+2$ états possibles. Les états sont nommés selon le nombre des composants survivants du système. Ils comprennent les états opérationnels variant de k à n et l'état de panne $k-1$. Nous supposons que tous les n composants du système sont survivants à l'instant $t=0$. Lorsque le système est en panne, aucune défaillance des composants survivants ne peut y arriver. Notons par λ_{n-i} le taux de défaillance du composant lorsque le système est à l'état $n-i$. μ est le taux de réparation. Le diagramme d'état de transition du système est représenté par la figure 2.2.

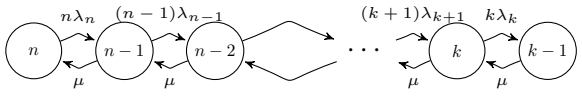


FIGURE 2.2 – Diagramme d'état de transition du système k/n

Dans cette étude, nous nous intéressons à la disponibilité stationnaire afin d'étudier la performance du système à long terme. Notons par π_s la distribution stationnaire du système. Les équations du système ainsi que sa matrice du transition, notée par M, sont obtenues en se basant

sur les modèles markoviens. Elles sont illustrées par l'équation (2.26).

$$\begin{cases} M \bullet \pi_s = 0 \\ \sum_{i=k-1}^{n} \pi_s^i = 1 \end{cases}, M = \begin{bmatrix} -n\lambda_n & \mu & & & \\ n\lambda_n & -(n-1)\lambda_{n-1} - \mu & \mu & & \\ & \cdots & \cdots & \cdots & \\ & & (k+1)\lambda_{k+1} & -k\lambda_k - \mu & \mu \\ & & & k\lambda_k & -\mu \end{bmatrix} \tag{2.26}$$

En se basant sur l'équation (2.26), la probabilité d'état stationnaire du système est dérivée comme le montre l'équation (2.27).

$$\begin{aligned} \pi_s \quad = \quad & \left(1 + \sum_{j=1}^{n-k+1} \prod_{i=0}^{j-1} \frac{(n-i)\lambda_{n-i}}{\mu}\right)^{-1} \left[1, \frac{n\lambda_n}{\mu}, \frac{n(n-1)\lambda_n\lambda_{n-1}}{\mu^2}, ..., \right. \\ & \left. \frac{n(n-1)..(k+1)\lambda_n...\lambda_{k+1}}{\mu^{n-k}}, \frac{n!\lambda_n...\lambda_k}{(k-1)!\mu^{n-k+1}}\right]^T \end{aligned} \tag{2.27}$$

L'expression explicite de la disponibilité stationnaire A_s du système est obtenue par l'équation (2.28).

$$A_s = \sum_{i=k}^{n} \pi_s^i = 1 - \pi_s^{k-1} = 1 - \frac{\prod_{i=0}^{n-k} \frac{(n-i)\lambda_{n-i}}{\mu}}{1 + \sum_{j=1}^{n-k+1} \prod_{i=0}^{j-1} \frac{(n-i)\lambda_{n-i}}{\mu}} \tag{2.28}$$

2.4.2 Modélisation de la dépendance redondante

La dépendance redondante du système $k/n : G$ est étudiée en considérant la répartition de charges parmi ses composants. Le système se trouve dans la condition de fonctionnement la plus sévère lorqu'il possède uniquement k composants opérationnels. Dans ce cas, nous désignons le taux de défaillance du composant par le taux de défaillance nominal λ. Aucune dépendance n'est considérée dans le système. Une telle situation est appelée indépendance. Notons par L la charge totale du système. Elle est répartie également parmi les composants identiques survivants. Dans le cas où k composants fonctionnent, la charge de chaque composant est L/k.

Lorsque le nombre de composants survivants, soit i, varie et dépasse k ($k < i \leq n$), la charge de chaque composant sera L/i. Par conséquent, par rapport au premier cas ayant k composants survivants, la charge supportée par chaque composant est alors diminuée d'un facteur égal à $\frac{L/i}{L/k} = \frac{i}{k}$. Ainsi, le taux de défaillance du composant λ_i change en fonction de la reconfiguration du système et de la charge répartie. Le concept de la dépendance redondante est alors impliqué. Une fonction de dépendance g est introduite pour quantifier cette dépendance comme le montre l'équation (2.29).

$$\lambda_i = \frac{\lambda}{g(i)}, k < i \leq n, g(k) \equiv 1, g(i) > 1 \tag{2.29}$$

Ainsi, le taux de défaillance d'un composant dans un système à composants dépendants est supposé inférieur à celui dans un système à composants indépendants ($g(i) > 1$). Lorqu'il est proportionnel à la charge, la dépendance est dite linéaire. La fonction de la dépendance redondante indique l'intensité de la dépendance entre les composants. $g(k) = 1$ désigne l'indépendance, $g(i) = \frac{i}{k}$ désigne la dépendance linéaire. En plus de ces deux classes, une dépendance faible et une dépendance forte peuvent être identifiées comme le montre le tableau 2.9. La dépendance faible a une valeur de $g(.)$ comprise entre celles de l'indépendance et de la dépendance linéaire. Pour la dépendance forte, la valeur de $g(.)$ est supérieure à celle de la dépendance linéaire. Cette classification est cohérente avec celle donnée par *Yu et al.* [28] pour un système parallèle.

Tableau 2.9 – Classification de la dépendance redondante

Type	fonction de dépendance $g(.)$
Indépendance	$g(i) = 1, k \leq i \leq n$
Dépendance faible	$g(k) = 1, 1 < g(i) < \frac{i}{k}, k+1 \leq i \leq n$
Dépendance linéaire	$g(i) = \frac{i}{k}, k \leq i \leq n$
Dépendance forte	$g(k) = 1, g(i) > \frac{i}{k}, k+1 \leq i \leq n$

En se basant sur l'équation (2.29), la disponibilité du système définie par l'équation (2.28) sera reformulée comme suit :

$$\begin{aligned} A_s(n,\lambda,\mu) = A_s(n,x) &= 1 - \frac{\prod_{i=0}^{n-k} \frac{(n-i)\lambda_{n-i}}{\mu}}{1+\sum_{j=1}^{n-k+1} \prod_{i=0}^{j-1} \frac{(n-i)\lambda_{n-i}}{\mu}} \\ &= 1 - \frac{\prod_{i=k}^{n} \frac{ix}{g(i)}}{1+\sum_{j=1}^{n-k+1} \prod_{i=n-j+1}^{n} \frac{ix}{g(i)}} = 1 - \frac{1}{1+\sum_{j=k}^{n} \prod_{i=k}^{j} \frac{g(i)}{ix}} \\ \text{où } x = \frac{\lambda}{\mu}, \quad & \prod_{i=0}^{j-1} \frac{(n-i)\lambda_{n-i}}{\mu} = \prod_{i=0}^{j-1} \frac{(n-i)x}{g(n-i)} = \prod_{i=n-j+1}^{n} \frac{ix}{g(i)} \end{aligned} \tag{2.30}$$

Le taux de défaillance du système ($\lambda_s = i\lambda_i$) et le taux de défaillance du composant λ_i sont liés par la relation décrite dans l'équation (2.31).

$$\frac{\lambda_s}{\lambda} = i\frac{\lambda_i}{\lambda} = \frac{i}{g(i)} = h(i) \tag{2.31}$$

En se basant sur la classification de la dépendance redondante présentée dans le tableau 2.9, la fonction $h(i)$ peut être illustée graphiquement comme le montre la figure 2.3. Il est à noter que deux cas particuliers peuvent être détectés à partir de cette figure. Ils sont représentés par la zone

hachurée. Le premier correspond à la zone ayant $i < k$ qui représente l'état de panne du système. Le deuxième correspond au secteur hachuré situé au dessus de la ligne d'indépendance, dans lequel la fonction de dépendance est $0 < g(i) < 1$. Cela implique que l'ajout d'un composant redondant dans le système accélérera l'échec des autres composants. Par conséquent, cette dépendance, appelée « dépendance négative » a un effet négatif sur la performance du système. Ce cas n'est pas considéré dans notre étude ($g(i)$ est supposé être supérieur ou égal à 1).
Ce modèle proposé pour la dépendance redondante du système k/n étend le modèle de la dépendance redondante présenté dans [28]. Ce dernier a été décrit dans la Partie 1 pour le système parallèle. Il peut être obtenu en mettant $k = 1$ dans le modèle proposé.

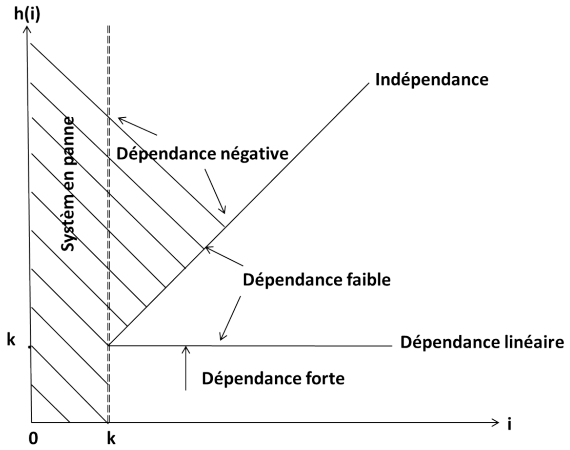


FIGURE 2.3 – Variation de $h(i)$ pour les différentes classes de la dépendance redondante

2.4.3 Description du problème d'optimisation

Dans la famille de modèles d'allocation de fiabilité du système, nous distinguons principalement les méthodes reposant sur la maximisation de fiabilité sous contrainte de coût [146], [147], les méthodes reposant sur la minimisation des coûts sous contrainte de fiabilité [145], [71] et les méthodes bi-objectives qui cherchent à maximiser la fiabilité et minimiser les coûts simultanément [125]. Ces méthodes s'appliquent également pour l'allocation de disponibilité. Dans cette partie, nous nous intéressons aux deux problèmes suivant en tenant en compte de la dépendance redondante :

– PPR : c'est le problème primal. Il vise à minimiser le coût total du système C_s sous contrainte de disponibilité exigée A_0.

– PDU : c'est le problème dual. Son objectif est la maximisation de la disponibilité stationnaire du système A_s en ayant comme contrainte un budget maximum à ne pas dépasser C_{max}.

Pour les deux problèmes, les variables de décisions sont le nombre de composants à mettre en parallèle n, leur taux de défaillance λ et leur taux de réparation μ qui interviennent dans l'évaluation de la disponibilité et du coût du système. Les expressions du coût et de la disponibilité sont représentées respectivement par les équations (2.15) et (2.30). La dépendance redondante est considérée dans ces deux expressions. Les fonctions objectives sont non linéaires et dépendent de variables continues (taux de défaillance et taux de réparation) et discrètes (nombre de composants). Le modèle d'optimisation est non linéaire à variables de décision mixte (MINLP). Nous proposons des méthodes de résolution basées sur l'AG et le solveur LINGO. Nous avons appliqué ces méthodes dans la Partie 1 pour résoudre le problème d'optimisation de la disponibilité du système parallèle à composants dépendants. Les résultats obtenus par l'AG étaient efficaces en termes de qualité de solutions et de rapidité par rapport à celles données par LINGO. Pour cela, il serait intéressant de tester leur efficacité dans la résolution des problèmes d'optimisation étudiés du système k sur n. L'AG a été appliqué dans de nombreux travaux relatifs à ce système [125], [40].

2.4.4 Problème primal PPR

2.4.4.1 Modèle mathématique

La fonction objective pour le problème PPR est la minimisation du coût. Elle est présentée par l'équation (2.32). Les contraintes sur la disponibilité exigée et les variables de décision sont illustrées par les équations (2.33)-(2.36).

$$Minimiser\, C_s(n, \lambda, \mu) = nC(\lambda) + C_M(\mu) \tag{2.32}$$

sous contraintes :

$$A_s(n, \lambda, \mu) \geq A_0 \tag{2.33}$$

$$k \leq n \leq n_U, n \in \mathbb{N} \tag{2.34}$$

$$\lambda_L \leq \lambda \leq \lambda_U, \lambda \in \mathbb{R} \tag{2.35}$$

$$\mu_L \leq \mu \leq \mu_U, \mu \in \mathbb{R} \tag{2.36}$$

2.4.4.2 Méthodes de résolution

2.4.4.2.1 Résolution par LINGO

Comme nous l'avons mentionné dans la Partie 1, pour implémenter le modèle sous LINGO, une reformulation doit se faire pour contourner le problème de présence de la variable de décision n dans l'opération de somme dans les expressions de C_s et A_s. Le modèle est mis à jour en introduisant le vecteur binaire y comme suit :

$$Minimiser\, C_s = \left(\sum_{i=k}^{n_U} y(i)\right)\left(\frac{1}{1-e^{\frac{-\lambda}{1+p*\theta}}}-1\right)+\mu^q, \tag{2.37}$$

sous contraintes :

$$\sum_{i=k}^{n_U}\left(y(i)\prod_{m=k}^{i}\frac{g(m)}{mx}\right) \geq \frac{A_0}{1-A_0}, \tag{2.38}$$

$$y(i) \in \{0,1\}\, \forall i \tag{2.39}$$

$$k \leq \sum_{i=k}^{n_U} y(i) \leq n_U \tag{2.40}$$

$$\lambda_L \leq \lambda \leq \lambda_U, \lambda \in \mathbb{R} \tag{2.41}$$

$$\mu_L \leq \mu \leq \mu_U, \mu \in \mathbb{R} \tag{2.42}$$

2.4.4.2.2 Résolution par l'algorithme génétique

Nous appliquons le même algorithme génétique décrit dans la Partie 1 qui a été implémenté pour optimiser la conception du système parallèle ayant comme fonction objective la minimisation du coût du système (voir la sous-section 2.3.4.2).

2.4.5 Problème dual PDU

2.4.5.1 Modèle mathématique

Pour le problème PDU, la fonction objective est la maximisation de la disponibilité du système. Elle est présentée par l'équation (2.43). La contrainte du coût est illustrée par l'équation (2.44). Les contraintes des variables de décision sont les mêmes que le problème PRR. Elles sont montrées par les équations (2.45)-(2.47).

$$Maximiser\ A_s(n, \lambda, \mu) \tag{2.43}$$

sous contraintes :

$$C_s(n, \lambda, \mu) \leq C_{max} \tag{2.44}$$

$$k \leq n \leq n_U, n \in \mathbb{N} \tag{2.45}$$

$$\lambda_L \leq \lambda \leq \lambda_U, \lambda \in \mathbb{R} \tag{2.46}$$

$$\mu_L \leq \mu \leq \mu_U, \mu \in \mathbb{R} \tag{2.47}$$

2.4.5.2 Méthodes de résolution

2.4.5.2.1 Résolution par LINGO

Le modèle d'optimisation pour le problème PDU est formulé comme suit :

$$Maximiser\ A_s = 1 - \frac{1}{1 + \sum_{i=k}^{n_U} \left(y(i) \prod_{m=k}^{i} \frac{g(m)}{mx} \right)} \tag{2.48}$$

sous contraintes :

$$\left(\sum_{i=k}^{n_U} y(i) \right) \left(\frac{1}{1 - e^{\frac{-\lambda}{1+p*\theta}}} - 1 \right) + \mu^q \leq C_{max} \tag{2.49}$$

$$y(i) \in \{0, 1\}\ \forall i \tag{2.50}$$

$$k \leq \sum_{i=k}^{n_U} y(i) \leq n_U \tag{2.51}$$

$$\lambda_L \leq \lambda \leq \lambda_U, \lambda \in \mathbb{R} \tag{2.52}$$

$$\mu_L \leq \mu \leq \mu_U, \mu \in \mathbb{R} \tag{2.53}$$

2.4.5.2.2 Résolution par l'algorithme génétique

Pour ce problème, l'AG décrit précédemment dans la sous-section 2.3.4.2 est adopté avec une modification de la fonction objective $F(x)$ comme le montre l'équation (2.54). $F(x)$ est égale à $A_s(x)$ si la solution x est faisable, c-à-d, elle satisfait la contrainte sur le coût du système ($C_s(x) \leq C_{max}$). Sinon, un grand nombre négatif N est ajouté à $F(x)$ pour pénaliser cette solution.

$$F(x) = A_s(x) + N \times max\{0, C_s(x) - C_{max}\}, N \text{ un grand nombre négatif.} \tag{2.54}$$

2.4.6 Applications numériques

Les modèles d'optimisation présentés des deux problèmes PPR et PDU sont résolus en utilisant LINGO et l'AG. Une forme particulière de la fonction de dépendance g est adoptée comme le montre l'équation (2.55). Le paramètre de dépendance θ permet de varier l'intensité de la dépendance redondante de la manière suivante : indépendance pour $\theta = 0.0$, dépendance faible pour $\theta < \frac{1}{k}$ (par exemple $\theta = \frac{1}{2k}$), dépendance linéaire pour $\theta = \frac{1}{k}$ et dépendance forte pour $\theta > \frac{1}{k}$ (par exemple $\theta = \frac{3}{k}$). Les valeurs de paramètres considérées pour l'application numérique sont illustrées par l'équation (2.56). L'objectif est de déterminer le nombre redondant n, le taux de défaillance λ et le taux de réparation μ des composants qui minimisent (respectivement maximisent) le coût (respectivement la disponibilité) du système k/n sous contraintes de ressources et une disponibilité exigée (respectivement un budget maximum alloué) pour le problème PPR (respectivement PDU) en considérant la dépendance redondante. La variation du coût (respectivement de la disponbilité) pour le problème PPR (respectivement PDU) est étudiée par rapport aux différents niveaux de la dépendance redondante. Différentes combinaisons des paramètres des fonctions de coût p et q, ainsi que plusieurs valeurs de contrainte sur la disponibilité exigée A_0 (respectivement sur le coût maximum C_{max}) sont considérées pour le problème PPR (respectivement PDU). Les figures 2.4 et 2.5 présentent respectivement les résultats obtenus par l'AG pour les problèmes PPR et PDU. Les symboles $\{\triangle, +, *, \circ\}$ correspondent respectivement aux classes indépendance, dépendance faible, dépendance linéaire et dépendance forte. De plus, une comparaison entre des solutions obtenues

par LINGO et l'AG est illustrée par les tableaux 2.10-2.17 pour la configuration primale et par les tableaux 2.18-2.19 pour la configuration duale. Nous avons fixé le temps d'exécution maximum alloué à $T_{max} = 86400$ secondes (24 h). Si aucune solution n'est obtenue par LINGO après T_{max}, nous interrompons son exécution et nous affichons les solutions. Dans ce cas, le temps correspondant T est marqué par le symbole $\ll - \gg$ dans les tableaux mentionnés ci-haut. Le pourcentage de déviation relative de coût ΔC et de disponibilité ΔA_s entre les deux méthodes de résolution est calculé respectivement pour les problèmes PPR (équation (2.24)) et PDU (équation (2.57)).

$$g(i) = \theta(i-k) + 1 \tag{2.55}$$

$$\begin{cases} k = 10, n_U = 100, \\ \lambda_L = 0.05, \lambda_U = 1.2, \\ \mu_L = 1, \mu_U = 20, \\ \theta \in \{0.0, \frac{1}{2k}, \frac{1}{k}, \frac{3}{k}\}. \end{cases} \tag{2.56}$$

$$\Delta A_s(\%) = \frac{A_{s_{LINGO}} - A_{s_{AG}}}{A_{s_{LINGO}}} \times 100 \tag{2.57}$$

2.4.6.1 Analyse des résultats du problème PPR

D'après la figure 2.4, nous pouvons constater que pour la même valeur de contrainte A_0, le coût C_s du système k/n varie avec l'intensité de la dépendance redondante, créant ainsi différentes options pour la conception du système. Dans le cas $p = 0.0$ (Figure 2.4a), où le coût des composants est indépendant de niveau de la dépendance redondante, le coût C_s du système diminue avec l'augmentation de l'intensité de la dépendance entre les composants (la courbe correspondante à la dépendance forte ($\circ$) est la plus basse). Alors que, dans les autres cas où la dépendance a été considérée ($p \neq 0$), une relation intéressante entre le coût de réparation, le coût des composants et le niveau de dépendance redondant peut être détectée. Le coût des composants était positivement lié à l'intensité de la dépendance, son augmentation (p élevé) qui entraîne un haut niveau de dépendance, peut perturber l'effort d'avoir un système économique comme le montre les cas ($p = 1.0, q = 0.25$), ($p = 1.0, q = 0.5$). Il est utile à noter que lorsque l'exigence sur la disponibilité du système est relativement élevée ($A_0 > 0.99$), l'utilisation des composants dépendants sera plus recommandée.

D'autre part, lorsque le coût de réparation augmente rapidement ($q \geq p$), nous pouvons observer qu'il est plus avantageux d'utiliser des composants dépendants comme l'illustre la figure 2.4b et les figures correspondants aux cas ($p = 0.5, q = 1.0$), ($p = 0.5, q = 2.0$), ($p = 1.0, q = 1.0$) et ($p = 1.0, q = 2.0$).

Par conséquent, une telle étude permet au concepteur du système de trouver les meilleurs paramètres (p, q, θ, A_0) permettant d'assurer l'équilibre souhaité entre un minimum coût de conception et une maximale performance pour un long terme de fonctionnement du système.

Une analyse des résultats des tableaux 2.10-2.17 révèle que l'AG était beaucoup plus rapide que LINGO. Le temps d'exécution de l'AG était inférieur à 5 secondes. Alors que LINGO avait pris plus de temps pour trouver les meilleures solutions. Son temps d'exécution consommé peut varier de plusieurs heures à un jour ou même plus. Il est à noter qu'il existe de nombreux cas où LINGO n'avait pas trouvé de solutions pendant T_{max}, pour cela il était interrompu et les solutions obtenues à T_{max} sont affichées (pour ces solutions, T est marqué par $\ll - \gg$). En termes de qualité de solutions, l'AG a donné des solutions qui sont très proches de celles fournies par LINGO avec un temps de calcul beaucoup plus rapide. La déviation du coût était inférieur à 2 %. Par conséquent, la résolution de problème d'optimisation par l'AG, et en particluier, pour les systèmes à grande échelle peut être plus avantageuse.

2.4.6.2 Analyse des résultats du problème PDU

La figure 2.5 montre que l'utilisation des composants ayant un haut niveau de dépendance redondante peut aboutir à une amélioration significative de la disponibilité du système, surtout lorsque le coût de la réparation est relativement élevé ($q \geq p$). La dépendance redondante peut aider donc à concevoir un système performant sous contrainte d'un budget limité. Il est à noter que ces résultats sont cohérents avec ceux obtenus pour le problème primal (partie 2.4.6.1).

De plus, les tableaux 2.18-2.19 montrent que les solutions de l'AG sont fournies pendant un temps d'éxecution rapide par rapport à celles données par LINGO avec une déviation de la disponibilité ΔA_s inférieure à 1%. En conclusion, nous pouvons déduire que l'AG est efficace pour résoudre le problème d'optimisation considérant la dépendance redondante pour les deux configurations primale et duale.

Tableau 2.10 – Comparaison LINGO/AG pour le problème PPR ayant $p = 0$ et $q = 1$

		LINGO					AG					
A_0	Type	n	λ	μ	A_s	C_s	$T(s)$	n	λ	μ	A_s	C_s	$T(s)$	$\Delta C(\%)$
0.95	Indépendant	16	0.987972	14.93937	0.95	24.43051	17878.79	16	0.983391	14.90746	0.950525	24.46823	3.495897879	0.2%
	Faible	17	1.083163	14.24628	0.95	22.94635	18253.82	18	1.065785	13.62832	0.950052	23.0865	3.401941588	0.6%
	Linéaire	18	1.182635	13.56379	0.95	21.51797	17371.8	17	1.195073	14.23866	0.950119	21.61778	3.391921714	0.5%
	Fort	19	1.2	9.310035	0.95	17.49928	1488.47	19	1.179677	9.289544	0.955082	17.72154	3.433834254	1.3%
0.98	Indépendant	17	0.945444	16.6994	0.98	27.50022	49148.22	17	0.938803	16.62254	0.980263	27.54153	3.459814767	0.2%
	Faible	18	1.047486	15.69535	0.98	25.42258	–	19	1.066291	15.49271	0.980084	25.46862	3.45713486	0.2%
	Linéaire	20	1.198429	14.82678	0.98	23.46645	49400.29	19	1.13873	14.58473	0.980246	23.53517	3.46078074	0.3%
	Fort	21	1.2	9.469941	0.98	18.52121	9693.88	20	1.183806	10.02989	0.98377	18.85296	3.488391527	1.8%
0.995	Indépendant	18	0.889527	18.95235	0.995	31.50483	14473.92	18	0.910401	19.4115	0.995026	31.5301	3.517572876	0.1%
	Faible	20	1.040824	17.57091	0.995	28.49064	86443.85	21	1.083204	17.7585	0.995008	28.50497	3.465063553	0.1%
	Linéaire	23	1.2	15.85435	0.995	25.76765	16989.43	22	1.134247	15.40771	0.995012	25.84002	3.465401302	0.3%
	Fort	23	1.2	9.843772	0.995	19.75707	979.48	25	1.170891	8.940832	0.995361	20.17747	3.479742241	2.1%

Tableau 2.11 – Comparaison LINGO/AG pour le problème PPR ayant $p = 0$ et $q = 2$

		LINGO					AG					
A_0	Type	n	λ	μ	A_s	C_s	$T(s)$	n	λ	μ	A_s	C_s	$T(s)$	$\Delta C(\%)$
0.95	Indépendant	15	0.309735	4.901184	0.95	65.33668	22080.7	15	0.306924	4.976165	0.954482	66.51733	3.646996	1.8%
	Faible	16	0.34831	4.768359	0.95	61.13682	71537.61	16	0.317134	4.356832	0.950773	61.8559	3.206041	1.2%
	Linéaire	16	0.369499	4.62669	0.95	57.19965	–	16	0.364037	4.667097	0.955219	58.21771	3.179026	1.8%
	Fort	21	0.585035	4.176646	0.95	43.85767	–	22	0.576523	4.009995	0.953468	44.29101	3.178875	1.0%
0.98	Indépendant	16	0.285727	5.273433	0.98	76.18716	–	17	0.313713	5.560692	0.98038	77.05472	3.247742	1.1%
	Faible	17	0.325541	5.087341	0.98	70.06223	–	17	0.349029	5.492862	0.980793	70.87151	3.178883	1.2%
	Linéaire	18	0.368742	4.912498	0.98	64.49913	16929.55	19	0.384829	4.977673	0.981608	65.25762	3.164832	1.2%
	Fort	24	0.615425	4.346812	0.98	47.11536	–	23	0.546105	3.996027	0.980597	47.62625	3.175048	1.1%
0.995	Indépendant	17	0.256796	5.737505	0.995	90.98289	–	17	0.265827	5.980013	0.995211	91.58815	3.395139	0.7%
	Faible	19	0.313102	5.485885	0.995	81.77306	–	19	0.304245	5.333215	0.995018	81.87376	3.344347	0.1%
	Linéaire	20	0.359602	5.245072	0.995	73.72587	50508.54	21	0.351192	4.975436	0.995452	74.6647	3.382813	1.3%
	Fort	28	0.654338	4.544046	0.995	50.95573	64331.19	27	0.625025	4.532155	0.996132	51.63595	3.379194	1.3%

Tableau 2.12 – Comparaison LINGO/AG pour le problème PPR ayant $p = 0.25$ et $q = 1$

		LINGO						AG						
A_0	Type	n	λ	μ	A_s	C_s	$T(s)$	n	λ	μ	A_s	C_s	$T(s)$	$\Delta C(\%)$
0.95	Indépendant	16	0.987972	14.93937	0.95	24.43051	39509.79	15	0.979778	15.55387	0.950614	24.56903	3.216955	0.6%
	Faible	17	1.090501	14.3428	0.95	23.12392	50496.96	16	1.090392	14.97741	0.950736	23.24334	3.191658	0.5%
	Linéaire	18	1.198825	13.74947	0.95	21.85519	38552.41	17	1.123591	13.3845	0.950071	21.9155	3.200565	0.3%
	Fort	19	1.2	9.310035	0.95	18.56266	881.86	18	1.19689	9.904103	0.953702	18.70757	3.167823	0.8%
0.98	Indépendant	17	0.945444	16.6994	0.98	27.50022	21974.95	17	1.02887	18.18263	0.980058	27.63809	3.244755	0.5%
	Faible	18	1.05455	15.80119	0.98	25.61821	59250.4	19	1.093522	15.9516	0.980614	25.72152	3.153427	0.4%
	Linéaire	20	1.2	14.84622	0.98	23.83761	17173.65	20	1.15991	14.38929	0.98042	23.91005	3.155437	0.3%
	Fort	20	1.2	9.932202	0.98	19.67181	1391.57	20	1.16749	9.800513	0.982363	19.99154	3.173559	1.6%
0.995	Indépendant	18	0.889527	18.95235	0.995	31.50483	35444.81	19	0.957883	19.75035	0.995033	31.57972	3.259602	0.2%
	Faible	20	1.047836	17.68929	0.995	28.70965	–	20	1.061017	17.94768	0.995085	28.74851	3.207879	0.1%
	Linéaire	22	1.2	16.29724	0.995	26.18777	38137.24	21	1.177321	16.52924	0.995034	26.27948	3.20469	0.4%
	Fort	22	1.2	10.30503	0.995	21.0186	2826.03	21	1.187421	10.87871	0.995621	21.28531	3.20128	1.3%

Tableau 2.13 – Comparaison LINGO/AG pour le problème PPR ayant $p = 0.25$ et $q = 2$

		LINGO						AG						
A_0	Type	n	λ	μ	A_s	C_s	$T(s)$	n	λ	μ	A_s	C_s	$T(s)$	$\Delta C(\%)$
0.95	Indépendant	15	0.309735	4.901184	0.95	65.33668	51815.57	15	0.289527	4.681647	0.954008	66.58783	3.220825	1.9%
	Faible	16	0.349774	4.788407	0.95	61.7041	83364.05	16	0.359391	4.936718	0.950745	61.91973	3.164825	0.3%
	Linéaire	16	0.372599	4.665497	0.95	58.26569	–	15	0.352852	4.762689	0.951166	59.18613	3.187605	1.6%
	Fort	21	0.599822	4.282214	0.95	46.44493	–	19	0.516703	4.020911	0.95107	46.95533	3.159547	1.1%
0.98	Indépendant	16	0.285727	5.273433	0.98	76.18716	–	17	0.33089	5.855349	0.9802	77.62963	3.223413	1.9%
	Faible	17	0.326907	5.108696	0.98	70.70795	–	18	0.37067	5.602141	0.981057	72.09966	3.211985	2.0%
	Linéaire	18	0.371835	4.953699	0.98	65.70094	–	18	0.349279	4.654029	0.980023	65.99319	3.278032	0.4%
	Fort	22	0.586066	4.436724	0.98	50.03294	73283.18	25	0.632536	4.379563	0.982319	50.38713	3.214641	0.7%
0.995	Indépendant	17	0.256796	5.737505	0.995	90.98289	75187.45	16	0.233672	5.600329	0.995063	92.14701	3.582526	1.3%
	Faible	19	0.314415	5.508893	0.995	82.52391	–	19	0.29983	5.328376	0.995532	83.52104	3.497322	1.2%
	Linéaire	20	0.362616	5.289031	0.995	75.0959	55098.83	20	0.369516	5.474526	0.995655	76.04798	3.473474	1.3%
	Fort	27	0.651455	4.648402	0.995	54.01699	21727.59	28	0.690209	4.79677	0.995063	54.10692	3.517456	0.2%

Tableau 2.14 – Comparaison LINGO/AG pour le problème PPR ayant $p = 0.5$ et $q = 1$

		LINGO					AG					
A0	Type	n	λ	μ	A_s	C_s	$T(s)$	n	λ	μ	A_s	C_s	$T(s)$	$\Delta C(\%)$
0.95	Indépendant	16	0.987972	14.93937	0.95	24.43051	11172.47	16	1.018843	15.49165	0.951152	24.53126	3.22941	0.4%
	Faible	17	1.097793	14.4387	0.95	23.3005	14193.67	17	1.077061	14.20737	0.950703	23.34758	3.189034	0.2%
	Linéaire	18	1.2	13.76295	0.95	22.19104	10026.93	17	1.195087	14.23355	0.950024	22.24835	3.178592	0.3%
	Fort	18	1.2	9.812809	0.95	19.60034	652.09	19	1.194256	9.302327	0.9514	19.71369	3.177591	0.6%
0.98	Indépendant	17	0.945444	16.6994	0.98	27.50022	43375.18	18	0.959356	16.4553	0.980003	27.63533	2.927587	0.5%
	Faible	18	1.061567	15.90635	0.98	25.81274	50192.2	18	1.102323	16.5844	0.980505	25.90467	2.910126	0.4%
	Linéaire	20	1.2	14.84622	0.98	24.21077	17704.3	19	1.117181	14.416	0.981286	24.42712	2.87987	0.9%
	Fort	20	1.2	9.932202	0.98	20.80723	6796.1	19	1.18394	10.49474	0.982017	21.05204	2.882491	1.2%
0.995	Indépendant	18	0.889527	18.95235	0.995	31.50483	19727.94	19	0.908104	18.71478	0.995015	31.55595	3.226838	0.2%
	Faible	20	1.054803	17.80691	0.995	28.92742	58475.05	20	1.060453	17.95312	0.99512	28.9788	3.116308	0.2%
	Linéaire	22	1.2	16.29724	0.995	26.59824	30998.98	23	1.188455	15.74614	0.995154	26.69109	3.114412	0.3%
	Fort	22	1.2	10.30503	0.995	22.26757	948.76	22	1.197288	10.33209	0.995271	22.33828	3.174377	0.3%

Tableau 2.15 – Comparaison LINGO/AG pour le problème PPR ayant $p = 0.5$ et $q = 2$

		LINGO					AG					
A_0	Type	n	λ	μ	A_s	C_s	$T(s)$	n	λ	μ	A_s	C_s	$T(s)$	$\Delta C(\%)$
0.95	Indépendant	15	0.309735	4.901184	0.95	65.33668	22383.17	15	0.319553	5.059513	0.950112	65.43801	3.211656	0.2%
	Faible	16	0.351227	4.808289	0.95	62.26912	71323.88	15	0.320663	4.707309	0.952254	62.99671	3.145368	1.2%
	Linéaire	16	0.375647	4.703664	0.95	59.32334	55519.2	16	0.374829	4.731357	0.951832	59.68114	3.156284	0.6%
	Fort	21	0.613906	4.382762	0.95	48.97665	76387.55	20	0.578199	4.312344	0.951996	49.20945	3.152388	0.5%
0.98	Indépendant	16	0.285727	5.273433	0.98	76.18716	–	15	0.257593	5.153517	0.981195	77.61183	3.284613	1.9%
	Faible	17	0.328262	5.129872	0.98	71.35107	–	17	0.325112	5.109899	0.98065	71.65657	3.19822	0.4%
	Linéaire	18	0.374876	4.994222	0.98	66.89329	–	19	0.404113	5.167324	0.980011	67.17655	3.173106	0.4%
	Fort	23	0.623451	4.549791	0.98	52.6598	–	23	0.628175	4.603988	0.980949	52.8446	3.195588	0.4%
0.995	Indépendant	17	0.256796	5.737505	0.995	90.98289	81806.76	17	0.236921	5.310672	0.995102	91.7925	3.520236	0.9%
	Faible	19	0.315717	5.531709	0.995	83.27174	81268.84	17	0.271169	5.360681	0.995339	84.87007	3.447722	1.9%
	Linéaire	20	0.36558	5.332268	0.995	76.45514	70444.55	20	0.384265	5.685925	0.995606	77.58817	3.443019	1.5%
	Fort	27	0.666837	4.758156	0.995	57.0006	67958.47	27	0.683059	4.897875	0.995375	57.2751	3.450429	0.5%

Tableau 2.16 – Comparaison LINGO/AG pour le problème PPR ayant $p = 1$ et $q = 1$

		LINGO						AG						
A_0	Type	n	λ	μ	A_s	C_s	$T(s)$	n	λ	μ	A_s	C_s	$T(s)$	$\Delta C(\%)$
0.95	Indépendant	16	0.987972	14.93937	0.95	24.43051	43274.23	16	0.973277	14.73449	0.950247	24.45147	2.953547	0.1%
	Faible	17	1.112237	14.62868	0.95	23.65072	38139.41	17	1.150627	15.15069	0.950273	23.6862	2.894679	0.2%
	Linéaire	18	1.2	13.76295	0.95	22.86775	337.31	17	1.182815	14.09869	0.950229	22.90319	2.905299	0.2%
	Fort	17	1.2	10.45452	0.95	21.66068	439.58	19	1.19549	9.284583	0.950366	21.8815	2.903356	1.0%
0.98	Indépendant	17	0.945444	16.6994	0.98	27.50022	72660.03	17	0.945908	16.71277	0.980034	27.50539	4.698134	0.0%
	Faible	18	1.075471	16.11467	0.98	26.19855	78090.89	18	1.061171	15.9222	0.980171	26.22347	4.686344	0.1%
	Linéaire	19	1.2	15.34308	0.98	24.95371	48902.61	20	1.185952	14.71243	0.980421	25.02596	4.6575	0.3%
	Fort	19	1.2	10.49829	0.98	23.02282	2057.01	19	1.199398	10.76318	0.983734	23.29733	4.715676	1.2%
0.995	Indépendant	18	0.889527	18.95235	0.995	31.50483	36669.95	19	0.947513	19.66254	0.995265	31.69328	3.206258	0.6%
	Faible	20	1.068606	18.03993	0.995	29.35932	–	21	1.11782	18.41513	0.995225	29.46978	3.160886	0.4%
	Linéaire	22	1.2	16.29724	0.995	27.42533	15123.24	21	1.175412	16.53059	0.995115	27.51861	3.147397	0.3%
	Fort	21	1.2	10.85662	0.995	24.69952	1263.3	21	1.186656	10.93167	0.995869	25.01316	3.157524	1.3%

Tableau 2.17 – Comparaison LINGO/AG pour le problème PPR ayant $p = 1$ et $q = 2$

		LINGO						AG						
A_0	Type	n	λ	μ	A_s	C_s	$T(s)$	n	λ	μ	A_s	C_s	$T(s)$	$\Delta C(\%)$
0.95	Indépendant	15	0.309735	4.901184	0.95	65.33668	64391.28	16	0.330786	5.015843	0.950582	65.9685	3.249023	1.0%
	Faible	16	0.354096	4.847563	0.95	63.39249	81000.63	16	0.334804	4.597283	0.950664	63.73812	3.191112	0.5%
	Linéaire	16	0.381598	4.778183	0.95	61.4145	87303.56 –	17	0.369541	4.444448	0.952475	62.33149	3.15817	1.5%
	Fort	20	0.614283	4.557792	0.95	53.88387	–	22	0.662828	4.582749	0.950789	54.08074	3.16572	0.4%
0.98	Indépendant	16	0.285727	5.273433	0.98	76.18716	80939.28	16	0.291498	5.408237	0.980515	76.52603	3.389323	0.4%
	Faible	17	0.330939	5.171707	0.98	72.62971	99450.45	17	0.320933	5.023341	0.980183	72.78544	3.331021	0.2%
	Linéaire	18	0.380815	5.07334	0.98	69.25078	–	18	0.367399	4.924603	0.980792	69.64418	3.30512	0.6%
	Fort	23	0.650243	4.745314	0.98	57.95555	–	24	0.694292	4.960605	0.982619	58.60854	3.290686	1.1%
0.995	Indépendant	17	0.256796	5.737505	0.995	90.98289	–	17	0.259975	5.813015	0.995024	91.05004	2.980286	0.1%
	Faible	19	0.31829	5.576784	0.995	84.75853	–	18	0.302902	5.568984	0.995	84.84205	2.949889	0.1%
	Linéaire	20	0.371368	5.416686	0.995	79.14255	–	19	0.348831	5.369222	0.995265	79.74418	2.964435	0.8%
	Fort	27	0.695612	4.96348	0.995	62.79354	–	28	0.73044	5.09103	0.995297	63.0558	3.597217	0.4%

Tableau 2.18 – Comparaison LINGO/AG pour le problème PDU ayant $p = 1$ et $q = 1$

		LINGO					AG					
C_{max}	Type	n	λ	μ	A_s	C_s	$T(s)$	n	λ	μ	A_s	C_s	$T(s)$	$\Delta A_s(\%)$
22	Indépendant	15	1.027982	13.6454	0.90702	22	244.61	15	1.05196	13.93875	0.906382	21.98917	2.871472	0.1%
	Faible	16	1.126475	13.68248	0.917843	22	290.35	16	1.102059	13.35344	0.916966	21.97203	2.840086	0.1%
	Linéaire	17	1.2	13.40102	0.931384	22	212.69	16	1.166387	13.51282	0.929393	21.99015	2.848782	0.2%
	Fort	18	1.2	10.13466	0.959543	22	198.93	16	1.19949	11.41978	0.952564	21.97361	2.822308	0.7%
27	Indépendant	17	0.958104	16.41964	0.976582	27	352.97	17	0.98845	16.91064	0.976272	26.98731	2.82808	0.0%
	Faible	19	1.094856	16.65661	0.98553	27	415.24	18	1.088576	17.09768	0.98521	26.98776	2.83097	0.0%
	Linéaire	21	1.2	16.37773	0.993487	27	282.57	20	1.096215	15.26876	0.992491	26.97189	2.894914	0.1%
	Fort	23	1.2	11.83873	0.99952	27	194.49	20	1.191815	13.6769	0.999143	26.99936	2.837633	0.0%
30	Indépendant	18	0.935893	18.38327	0.991446	30	394.83	18	0.94666	18.53823	0.991194	29.95166	2.86538	0.0%
	Faible	21	1.090562	18.49532	0.996355	30	267.06	20	1.01803	17.78028	0.996273	29.9995	2.846178	0.0%
	Linéaire	25	1.2	17.35444	0.999156	30	423.49	20	1.104685	18.38128	0.998523	29.94276	2.848041	0.1%
	Fort	27	1.2	12.20199	0.999987	30	277.99	20	1.150114	15.84762	0.999869	29.90971	2.847941	0.0%

Tableau 2.19 – Comparaison LINGO/AG pour le problème PDU ayant $p = 1$ et $q = 2$

		LINGO					AG					
C_{max}	Type	n	λ	μ	A_s	C_s	$T(s)$	n	λ	μ	A_s	C_s	$T(s)$	$\Delta A_s(\%)$
58	Indépendant	14	0.324374	4.632727	0.912274	58	263.65	14	0.340533	4.794985	0.908428	57.50039	2.838017	0.4%
	Faible	15	0.363411	4.661417	0.920478	58	329.63	15	0.35557	4.442673	0.912714	56.95497	2.84109	0.8%
	Linéaire	16	0.401889	4.660556	0.929752	58	258.2	16	0.403795	4.678964	0.929452	57.96751	2.836052	0.0%
	Fort	23	0.649808	4.746827	0.980278	58	849.62	23	0.626172	4.54338	0.978658	57.8124	2.842456	0.2%
70	Indépendant	15	0.291007	5.058805	0.965581	70	322.87	15	0.323033	5.497804	0.962473	69.56387	2.861717	0.3%
	Faible	17	0.341871	5.082016	0.97359	70	780.95	15	0.315276	5.204312	0.970805	69.91591	2.834188	0.3%
	Linéaire	18	0.377962	5.108759	0.981929	70	884.47	19	0.395407	5.08687	0.98087	69.80115	2.830807	0.1%
	Fort	31	0.712954	5.250226	0.999615	70	1155.6	20	0.490922	4.983879	0.99708	68.42852	2.844054	0.3%
86	Indépendant	17	0.27039	5.589745	0.991915	86	261.29	17	0.284309	5.851518	0.991695	85.93652	2.843741	0.0%
	Faible	19	0.309832	5.535686	0.995809	86	692.87	19	0.336367	5.944015	0.995327	85.64782	2.841442	0.0%
	Linéaire	22	0.377091	5.674255	0.998431	86	767.63	24	0.418839	5.842332	0.998068	85.9239	2.860465	0.0%
	Fort	32	0.615568	5.758657	0.999999	86	249.6	32	0.695712	6.335015	0.999998	85.34761	2.83265	0.0%

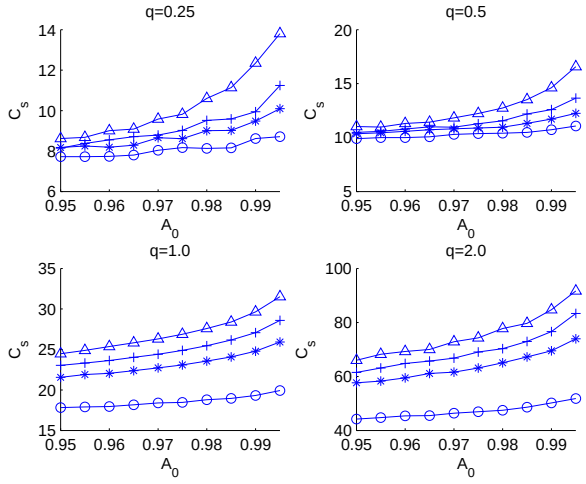
2.5 Conclusion

Le problème de conception des systèmes de type parallèle et k/n est étudié dans ce chapitre en tenant en considération la dépendance de défaillance entre les composants. Ce problème est l'un des moins étudiés en optimisation de la fiabilité/disponibilité bien qu'il est rencontré fréquemment dans de nombreux cas pratiques comme les systèmes de production ou d'alimentation électrique. Ainsi, dans un premier temps, nous nous sommes concentrés sur le problème d'optimisation du système parallèle. C'était un problème non linéaire à variables de décision mixtes. L'objectif était de trouver la meilleure configuration du système à composants dépendants, qui minimise le coût sous contrainte d'une disponibilité exigée. Nous avons utilisé le solveur LINGO qui permet de résoudre efficacement les problèmes non linéaires. De plus, nous avons implémenté l'algorithme génétique en raison de sa rapidité d'exécution et sa flexibilité à représenter des variables de décision mixtes. Une comparaison des résultats obtenus par LINGO et l'AG a été établie. Dans un deuxième temps, nous avons considéré les systèmes k/n qui sont des systèmes plus générales que les systèmes parallèles. Nous avons proposé un modèle pour quantifier la dépendance redondante entre les composants de tels systèmes. De nombreuses classes ont été identifiées : indépendance, faible, linéaire et forte. De plus, nous avons distingué deux configurations d'optimisation : primale et duale. Pour chacune d'elles, un modèle mathématique non linéaire est formulé. Comme l'AG a donné de très bons résultats dans la première partie concernant l'optimisation des systèmes parallèles, nous avons choisi de tester son efficacité pour l'optimisation des systèmes k/n. Le solveur LINGO est aussi utilisé pour évaluer la qualité des solutions obtenues.

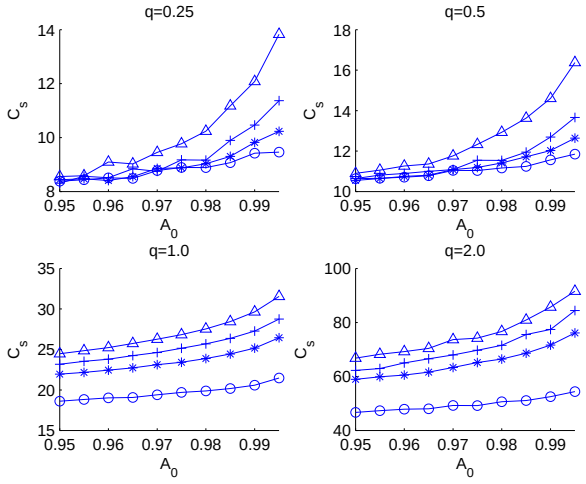
Dans le but de bien vérifier l'efficacité et de mieux comparer les méthodes de résolution, des protocoles de tests ont été appliqués pour les différents niveaux de la dépendance redondante. Une comparaison en termes de qualité de solutions et de temps d'exécution a été établie. Les solutions données par l'AG étaient très proches de celles de LINGO et obtenues durant un temps d'exécution plus rapide. Les approches proposées pour les deux problèmes primal et dual ont présenté des résultats intéressants. Elles soulignent l'importance de considérer la dépendance redondante dès la phase de conception. Les composants redondants ayant une dépendance suffisamment forte étaient plus recommandés dans les cas où le coût de réparation croît rapidement ($q > p$) ou lorsqu'une grande valeur de la disponibilité du système est exigée. Cette dépendance peut améliorer la performance du système sans avoir besoin d'ajouter des composants redondants. Elle a aboutit à une augmentation de la disponibilité du système et une diminution de son coût. Ces travaux ont fait

l'objet de plusieurs publications : un article publié dans la conférence internationale IFAC MIM 2016 [30], un article accepté dans la conférence internationale MMR 2017 [127] et une communication dans la conférence internationale LAAS'16 [128].

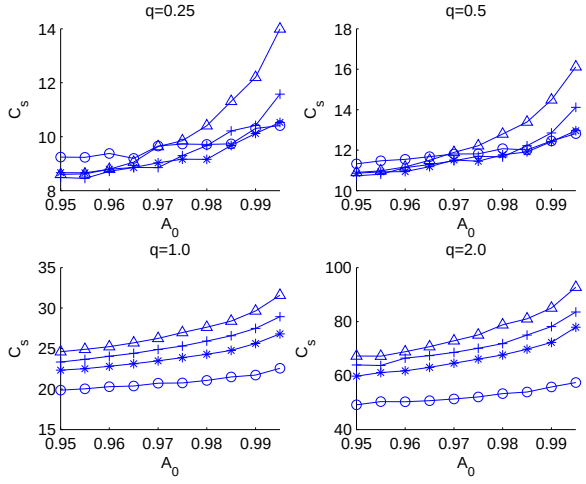
Dans le chapitre suivant, nous allons étendre l'approche proposée pour optimiser la conception des systèmes séries k/n à composants dépendants. Plusieurs méthodes de résolution seront aussi présentées et comparées.



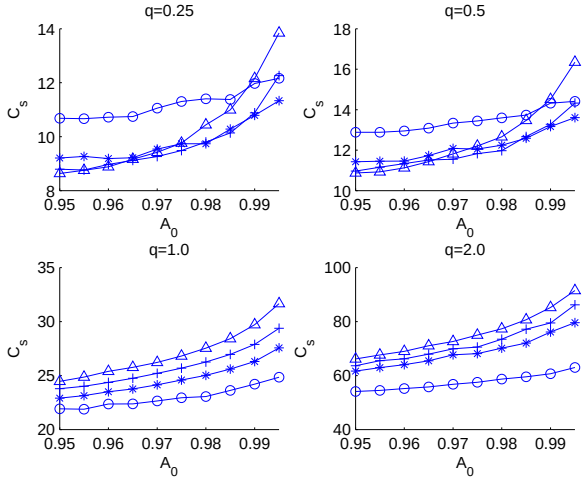
(a) p=0.0



(b) p=0.25

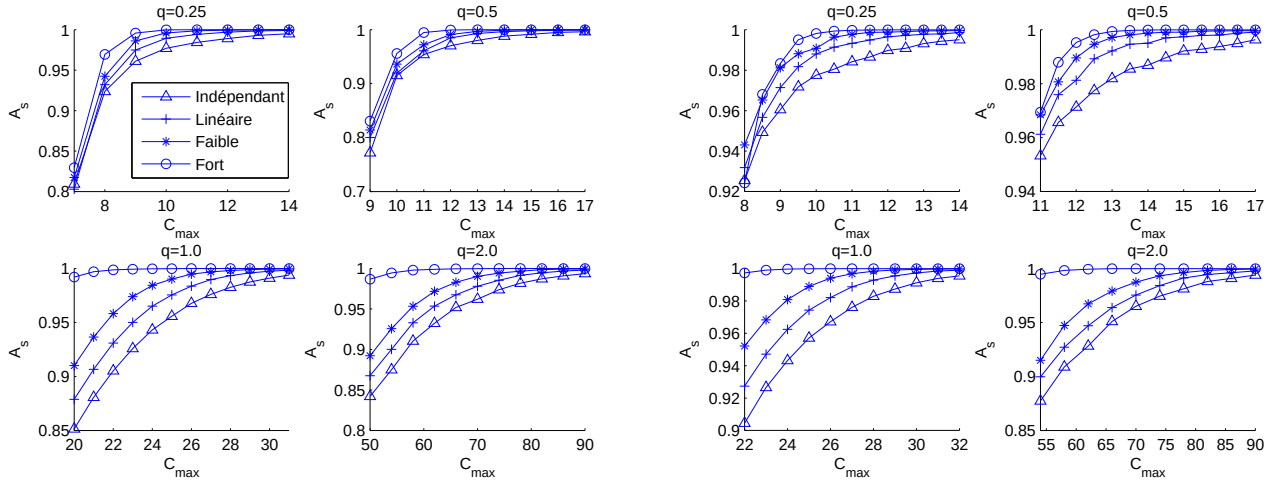


(c) p=0.5



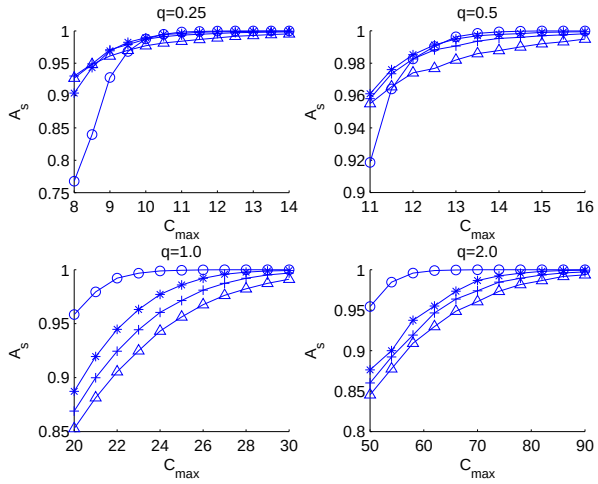
(d) p=1.0

FIGURE 2.4 – Coût du système k/n par rapport au niveau de la dépendance redondante pour le problème primal

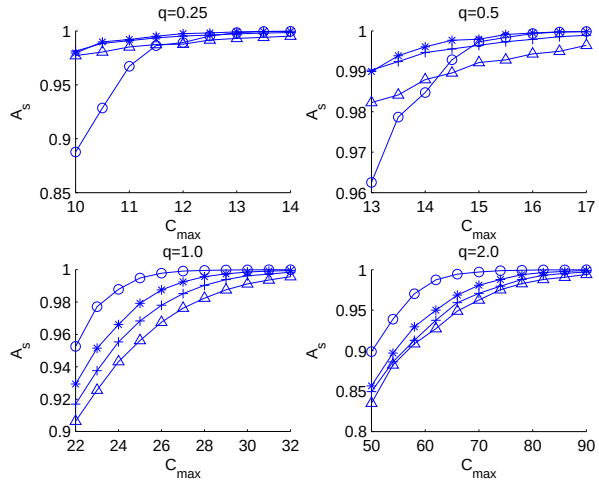


(a) p=0

(b) p=0.25



(c) p=0.5



(d) p=1.0

FIGURE 2.5 – Disponibilité du système k/n par rapport aux différents niveaux de la dépendance redondante pour le problème dual

Chapitre 3

Optimisation des systèmes séries k sur n à composants dépendants

Résumé :

Ce chapitre présente une extension de l'étude faite au chapitre 3 *pour évaluer les performances d'un système série k/n à composants dépendants. Vu que la dépendance de défaillance et les ressources de réparation affectent significativement la disponibilité du système, elles sont considérées lors de la phase de conception. Un problème d'optimisation non linéaire est ensuite formulé en monocritère tenant en compte l'intensité de la dépendance redondante. Il vise à déterminer la meilleure configuration en termes de nombre de composants et nombre des équipes de réparation relatifs à chaque sous-système, tout en minimisant le coût total sous contrainte d'une disponibilité exigée. Etant donné que le problème est NP-difficile, des méthodes de résolution basées sur les algorithmes génétiques, les algorithmes d'optimisation par colonies de fourmis, et leur hybridation avec une recherche locale sont développées. Des problèmes de différentes tailles sont testés afin d'évaluer l'efficacité des approches d'optimisation proposées. Les solutions des meilleurs algorithmes sont comparées à celles obtenues par une méthode exacte et par le solveur LINGO. De plus, nous abordons, par la suite, le problème en configuration duale, afin de maximiser la disponibilité pour ce type des systèmes. Les résultats obtenus sont intéressants et ont fait l'objet de deux articles publiés dans les conférences internationales IFAC AMEST* 2016 *[31] et IMCET* 2016 *[148].*

3.1 Introduction

Les systèmes séries-k/n sont constitués de plusieurs sous-systèmes k/n reliés en série. Chaque sous-système comprend des composants en redondance et fonctionne si au moins k parmi ses composants fonctionnent. De telles configurations peuvent être utilisées pour modéliser les stations à micro-ondes d'un réseau de télécommunications, de systèmes d'oléoduc, de systèmes de tolérance aux pannes, des chaines de production, etc. [149]. Les systèmes séries-k/n, ainsi que leurs cas particuliers série-parallèle ($k = 1$) sont ceux qui ont attiré le plus d'intérêt en optimisation de l'allocation de redondance et de la fiabilité/disponibilité.

Comme nous l'avons mentionné dans le chapitre 2 concernant l'optimisation des systèmes parallèles et des systèmes k/n, la dépendance redondante entre les composants peut avoir un impact important pour l'optimisation de la performance du système. Dans ce chapitre, nous étendons cette étude. Nous nous intéressons aux problèmes de conception des systèmes réparables séries-k/n avec dépendance redondante dans les deux configurations primale et duale. De nombreuses équipes de réparation sont allouées pour chaque sous-système. En fait, les ressources de réparation, que ce soit humaines (comme les équipes de réparation) ou matérielles (comme les pièces de rechange) sont limitées en général, et affectent la disponibilité du système. Pour cela, il serait utile de les considérer lors de la phase de conception. Nour el Fath et Ait-Kadi [150] ont étudié en 2007 un système multi-états réparable série-parallèle avec des ressources de maintenance limitées. Un ensemble de personnel a été attribué à chaque module (sous-système) pour les actions de maintenance. Une approche d'optimisation a été proposée et résolue via une heuristique. L'objectif était de trouver, sous contrainte de fiabilité, la meilleure configuration et les coûts de maintenance du système pour lequel le nombre des équipes de maintenance est supposé inférieur au nombre des composants réparables.

Le problème d'allocation combinée de redondance et de fiabilité a été étudié par Chern en 1992 [151] pour les systèmes séries-parallèles. Il a montré que ce problème, aussi bien pour la minimisation de coût de certaines ressources sous contrainte d'une fiabilité exigée que son dual, est NP-difficile [151]. Pour cela, de nombreuses études ont été menées dans ce domaine conduisant à l'élaboration de méthodes approchées, et en particulier, les métaheuristiques. Parmi les travaux qui ont appliqué les métaheuristiques dans des problèmes de conception de systèmes séries-k/n, nous citons celui de Sooktip *et al.* [152] en 2011 qui utilisent les AGs pour maximiser la fiabilité du système à composants hétérogènes en respectant des contraintes de coût et de poids, celui de

Boddu et Xing [153] en 2013 qui proposent une méthodologie basée sur l'AG avec une pénalité guidée pour le problème d'allocation de redondance d'un système ayant des composants à redondance mixte : active et passive. Nous citons également les papiers de Liang et Smith [89] en 2004, qui ont appliqué pour la première fois, l'optimisation par colonies de fourmis (ACO) pour le problème d'allocation de fiabilité en conception du système. Les composants étaient indépendants et binaires. La fonction objectif était la fiabilité, et les contraintes étaient le coût et le poids du système. Une recherche locale a été également appliquée pour améliorer les solutions fournies par l'ACO. Plus tard, Ebrahimipur *et al.* [154] ont appliqué en 2009, la métaheuristique ACO pour optimiser la fiabilité du système multi-états sous contraintes de coût et de poids. Les variables de décision étaient le nombre et les versions des composants utilisés dans chaque sous-système. La méthode de la fonction génératrice universelle a été utilisée pour évaluer la fiabilité du système tout entier. Toutes ces études illustrent l'efficacité de l'utilisation des métaheuristiques afin d'optimiser les problèmes d'allocation de redondance et de fiabilité des systèmes séries-k/n. Pour cela, nous proposons, dans la suite, des approches d'optimisation basées sur l'AG, les algorithmes ACO, et leur hybridation avec une recherche locale tout en considérant la dépendance redondante. Il est à noter que c'est la première application de l'ACO pour tels problèmes.

3.2 Description du système

Un système série-k/n est considéré comme le montre la figure 3.1. Il est constitué de l sous-systèmes connectés en série, dont chacun comprend n_i composants et fonctionne si au moins k_i composants fonctionnent ($i = 1, ..., l$). Les hypothèses adoptées sont les suivantes, elles ont été appliquées dans de nombreux travaux de la littérature.

- Les composants dans chaque sous-système sont identiques et possèdent deux états possibles : en fonctionnement ou en panne [143].
- Lorsque le système tombe en panne, aucune défaillance des composants survivants ne peut y arriver [155].
- Il existe r_i équipes de réparation attribuées à chaque sous-système i ($1 \leq r_i \leq n_i - k_i + 1$) où $n_i - k_i + 1$ est le nombre maximum possible des composants défaillants [156], [157]. Chaque équipe de réparation peut fixer, à un temps donné, un composant défaillant. La réparation est supposée parfaite [150].
- Le taux de défaillance et de réparation suivent une distribution exponentielle. Les modèles

Markoviens peuvent être alors utilisés pour évaluer la performance du système [5].

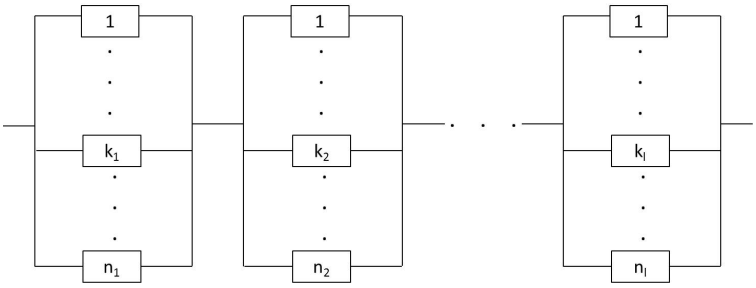


FIGURE 3.1 – Structure du système séries-k/n

Notons par m le nombre des composants survivants dans le sous-système i ($i = 1, ..., l$). Le taux de défaillance de ce sous-système lorsqu'il est à l'état m ($m = k_i - 1, k_i, ..., n_i - 1, n_i$) est désigné par $m\lambda^i_m$, où λ^i_m représente le taux de défaillance du composant dans le sous-système i à cet état m. Quant au taux de réparation, noté par μ^i_m, du sous-système i, il est défini comme le montre l'équation (3.1)[157].

$$\mu^i_m = \begin{cases} r_i\mu^i, & k_i - 1 \leq m \leq n_i - r_i \\ (n_i - m)\mu^i, & n_i - r_i + 1 \leq m \leq n_i - 1 \\ 0 & \text{sinon.} \end{cases} \tag{3.1}$$

Le diagramme de transition d'un sous-système i (k_i/n_i) est illustré par la figure 3.2. Les notations adoptées dans ce chapitre sont présentées dans le tableau 3.1.

3.3 Concept de la dépendance redondante et évaluation de la disponibilité

Dans cette section, le concept de la dépendance redondante proposé dans le chapitre précédent est appliqué pour la configuration k_i/n_i. La disponibilité du système à composants dépendants est

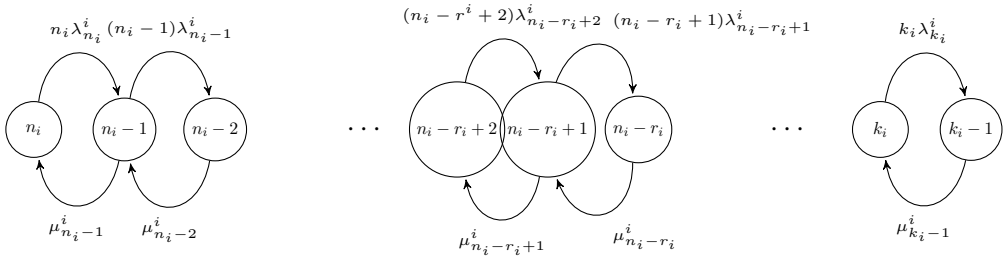


FIGURE 3.2 – Diagramme de transition d'état de sous-système i (k_i/n_i)

Tableau 3.1 – Notations

l	nombre de sous-systèmes
i	indice de sous-système ($i = 1, ..., l$)
n_i	nombre des composants dans le sous-système i
k_i	nombre minimum des composants requis pour le sous-système i
n_i^{max}	nombre maximum de n_i
r_i	nombre des équipes de réparation pour le sous-système i
a	niveau de l'intensité de la dépendance entre les composants redondants
λ^i, μ^i	taux de défaillance nominal et taux de réparation des composants du sous-système i
C_i^c, C_i^r	coût d'acquisition des composants et coût des équipes de réparation utilisés dans le sous-système i
$g(.)$	fonction de dépendance
A_i	disponibilité du sous-système i
A_s	disponibilité du système
A_0	valeur minimale de disponibilité exigée

ensuite déterminée.

Dans le cas des composants indépendants, la disponibilité A_i du sous-système i (k_i/n_i) ayant r_i équipes de réparation, est donnée par l'équation (3.2).

$$A_i = \frac{\prod_{m=k_i}^{n_i} \frac{m\lambda_m^i}{\mu_{m-1}^i}}{1 + \sum_{j=k_i}^{n_i} \prod_{m=j}^{n_i} \frac{m\lambda_m^i}{\mu_{m-1}^i}} = 1 - \frac{1}{1 + \sum_{j=k_i}^{n_i} \prod_{m=k_i}^{j} \frac{\mu_{m-1}^i}{m\lambda_m^i}} \tag{3.2}$$

Dans le cas où la dépendance de défaillance entre les composants est considérée, une reformulation de la disponibilité A_i doit être effectuée.

Rappelons le principe de la dépendance redondante : le taux de défaillance d'un composant en fonctionnement dans un sous-système i ayant m composants opérationnels dépend de son taux de défaillance nominal λ et de la fonction de dépendance, notée par $g(m)$, comme le montre l'équation (3.3). Dans ce cas, le taux de défaillance de ce sous-système i est alors déterminé comme l'illustre l'équation (3.4).

$$\frac{\lambda^i}{g(m)},\ k_i < m \leq n_i,\ g(k_i) \equiv 1,\ g(m) \geq 1 \tag{3.3}$$

$$\lambda_m^i = m\frac{\lambda^i}{g(m)},\ k_i \leq m \leq n_i \tag{3.4}$$

En se basant sur les équations (3.1), (3.3) et (3.4), la disponibilité A_i correspondante au sous-système i ($i = 1, 2, ..l$) tenant en compte de la dépendance redondante peut être obtenue comme

le montre l'équation (3.5).

$$A_i = 1 - (1 + A_{i1} + A_{i2})^{-1} \tag{3.5}$$

où

$$A_{i1} = \sum_{j=k_i}^{n_i - r_i + 1} \frac{(k_i - 1)! r_i^{j-k_i+1} \prod_{m=k_i}^{j} g(m)}{j!} (\frac{\mu^i}{\lambda^i})^{j-k_i+1} \tag{3.6}$$

$$A_{i2} = \sum_{j=n_i-r_i+2}^{n_i} \frac{(k_i - 1)! r_i^{n_i-r_i-k_i+2} (r_i - 1)! \prod_{m=k_i}^{j} g(m)}{j!(n_i - j)!} (\frac{\mu^i}{\lambda^i})^{j-k_i+1} \tag{3.7}$$

Il est à noter que si nous remplaçons k_i par la valeur 1 dans les équations (3.6) et (3.7), l'équation (3.5) va correspondre à l'expression de la disponibilité du système parallèle avec dépendance redondante, ayant r_i équipes de réparation. Elle est égale à celle rapportée dans la littérature [143]. Quant à la classification de la dépendance redondante, nous avons identifié quatres classes dans le chapitre 2 (partie 2.4.2, tableau 2.9) comme suit : indépendance ($g(m) = 1$), dépendance faible ($g(m) < \frac{m}{k_i}$), dépendance linéaire ($g(m) = \frac{m}{k_i}$) et dépendance forte ($g(m) > \frac{m}{k_i}$). En se basant sur cette classification, nous adoptons, dans ce chapitre, une forme paramétrique spécifique de la fonction $g(.)$, comme le montre l'équation (3.8), elle peut satisfaire les différentes relations ci-haut en variant le paramètre de dépendance a. Ce dernier permet alors de caractériser l'intensité de la dépendance. Autres formes de la fonction $g(.)$ peuvent être choisies, elles dépendent de la loi de répartition des charges, et aussi du comportement de défaillance du composant lorsqu'il est chargé.

$$g(m) = \left(\frac{m}{k}\right)^a \tag{3.8}$$

En utilisant l'équation (3.8), le terme $\prod_{m=k_i}^{j} g(m)$ peut être reformulé comme l'illustre l'équation (3.9).

$$\prod_{m=k_i}^{j} g(m) = \prod_{m=k_i}^{j} \left(\frac{m}{k_i}\right)^a = \left(\frac{j!}{(k_i - 1)!}\right)^a \frac{1}{(k_i^a)^{j-k_i+1}} = \frac{\prod_{m=k_i}^{j} m^a}{(k_i^a)^{j-k_i+1}} \tag{3.9}$$

En se basant sur l'équation (3.9), les expressions de A_{i1} et A_{i2} sont mises à jour. Par conséquent, la disponibilité du sous-système i est obtenue comme le montre l'équation (3.10) :

$$A_i = 1 - \left(1 + \sum_{j=k_i}^{n_i-r_i+1} \left(\frac{j!}{(k_i-1)!}\right)^{a-1} \left(\frac{r_i \mu^i}{k_i^a \lambda^i}\right)^{j-k_i+1} + \sum_{j=n_i-r_i+2}^{n_i} \frac{(j!)^{a-1} r_i^{n_i-r_i-k_i+2} (r_i-1)!}{((k_i-1)!)^{a-1} k_i^{a(j-k_i+1)} (n_i-j)!} (\frac{\mu^i}{\lambda^i})^{j-k_i+1}\right)^{-1} \tag{3.10}$$

Evaluation de la disponibilité du système

Le système étudié est composé de l sous-systèmes k_i/n_i $(i = 1, ..., l)$ connectés en série. Il tombe en panne, si au moins l'un de ses sous-systèmes est défaillant. Ainsi, la disponibilité du système A_s

est donnée alors par l'équation (3.11).

$$A_s = \prod_{i=1}^{l} A_i \tag{3.11}$$

Nous présentons ci-dessous l'expression explicite de la disponibilité du système pour les différents niveaux de la dépendance redondante.

Cas de dépendance faible et forte : $g(m) = \left(\frac{m}{k}\right)^a$. Les valeurs $0 < a < 1$ et $a > 1$ correspondent respectivement aux cas de la dépendance faible et de la dépendance forte.

$$A_s = \prod_{i=1}^{l} \left\{ 1 - \left(1 + \sum_{j=k_i}^{n_i - r_i + 1} \frac{(j!)^{a-1} r_i^{j-k_i+1}}{((k_i-1)!)^{a-1} k_i^{a(j-k_i+1)}} (\frac{\mu^i}{\lambda^i})^{j-k_i+1} + \sum_{j=n_i-r_i+2}^{n_i} \frac{(j!)^{a-1} r_i^{n_i-r_i-k_i+2} (r_i-1)!}{((k_i-1)!)^{a-1} k_i^{a(j-k_i+1)} (n_i-j)!} (\frac{\mu^i}{\lambda^i})^{j-k_i+1} \right)^{-1} \right\} \tag{3.12}$$

Cas de l'indépendance : $a = 0$, $g(m) = 1$

$$A_s = \prod_{i=1}^{l} \left\{ 1 - \left(1 + \sum_{j=k_i}^{n_i - r_i + 1} \frac{(k_i-1)! r_i^{j-k_i+1}}{j!} (\frac{\mu^i}{\lambda^i})^{j-k_i+1} + \sum_{j=n_i-r_i+2}^{n_i} \frac{(k_i-1)! r_i^{n_i-r_i-k_i+2} (r_i-1)!}{j!(n_i-j)!} (\frac{\mu^i}{\lambda^i})^{j-k_i+1} \right)^{-1} \right\} \tag{3.13}$$

Cas de la dépendance linéaire : $a = 1$, $g(m) = \frac{m}{k_i}$

$$A_s = \prod_{i=1}^{l} \left\{ 1 - \left(1 + \sum_{j=k_i}^{n_i - r_i + 1} (\frac{r_i}{k_i})^{j-k_i+1} (\frac{\mu^i}{\lambda^i})^{j-k_i+1} + \sum_{j=n_i-r_i+2}^{n_i} \frac{(r_i^{n_i-r_i-k_i+2} (r_i-1)!}{k_i^{j-k_i+1} (n_i-j)!} (\frac{\mu^i}{\lambda^i})^{j-k_i+1} \right)^{-1} \right\} \tag{3.14}$$

3.4 Description du problème d'optimisation

L'objectif de notre étude est de choisir une configuration optimale du système série-k/n en termes de nombre de composants et des équipes de réparation à attribuer à chaque sous-système. Les composants sont caractérisés par leur taux de défaillance, leur taux de réparation et les coûts

unitaires d'acquisition et de réparation. La dépendance de défaillance entre les composants redondants de chaque sous-système est à considérer dès la phase de conception. L'intensité de la dépendance redondante est intégrée dans l'expression de la disponibilité du système. Nous étudions pour cela, deux problèmes d'optimisation non linéaire à variabes de décision mixte : PPR et PDU. Le premier est celui qui nécessite le moindre coût tout en respectant la contrainte d'une disponibilité exigée. Quant au problème (PDU), la configuration optimale possède la maximale disponibilité sous la contrainte d'un coût maximum à ne pas dépasser.

3.5 Problème PPR

3.5.1 Modèle mathématique

La formulation mathématique du modèle d'optimisation est illustrée par les équations (3.15)-(3.18).

$$Minimiser\, C_s = \sum_{i=1}^{l} (n_i C_i^c + r_i C_i^r) \tag{3.15}$$

sous contraintes :

$$A_s = \prod_{i=1}^{l} \left\{ 1 - \left(1 + \sum_{j=k_i}^{n_i - r_i + 1} \frac{(k_i-1)! r_i^{j-k_i+1} \prod_{m=k_i}^{j} g(m)}{j!} \left(\frac{\mu^i}{\lambda^i} \right)^{j-k_i+1} + \sum_{j=n_i-r_i+2}^{n_i} \frac{(k_i-1)! r_i^{n_i-r_i-k_i+2} (r_i-1)!}{j!(n_i-j)!} \prod_{m=k_i}^{j} g(m) \left(\frac{\mu^i}{\lambda^i} \right)^{j-k_i+1} \right)^{-1} \right\} \geq A_0 \tag{3.16}$$

$$k_i \leq n_i \leq n_i^{max}, n_i \in \mathbb{N}^*, \;\; i = 1, 2, ..., l \tag{3.17}$$

$$1 \leq r_i \leq n_i - k_i + 1, r_i \in \mathbb{N}^* \;\; i = 1, 2, ..., l \tag{3.18}$$

L'objectif est de minimiser le coût du système C_s tel qu'indiqué par l'équation (3.15). Il comprend la somme des coûts unitaires des composants C_i^c et des équipes de réparation C_i^r de chaque sous-système i ($i = 1, ..., l$). La contrainte (3.16) révèle que la disponibilité du système A_s ne doit pas être inférieure à une valeur requise A_0. La contrainte (3.17) indique que le nombre de composants n_i de chaque sous-système i devrait être compris entre k_i et n_i^{max}, où k_i est le nombre minimum de composants requis pour avoir le sous-système i opérationnel et n_i^{max} est le nombre maximum de composants alloué pour ce sous-système. En outre, la contrainte (3.18) signifie que le

nombre des équipes de réparation relatif à chaque sous-système i est compris entre 1 et le nombre maximum possible de composants défaillants $n_i - k_i + 1$.

3.5.2 Méthodes de résolution

Dans cette partie, nous proposons plusieurs méthodes de résolution du problème étudié. Vu que le modèle d'optimisation décrit est NP-difficile, nous choisissons d'appliquer des métaheuristiques, ainsi que leur hybridation, dû à leur efficacité pour la résolution de tels problèmes. Les solutions des meilleurs algorithmes sont comparées ensuite à celles obtenues par une méthode exacte et par le solveur LINGO.

Nous présentons d'abord le modèle reformulé utilisé sous LINGO. Ensuite, nous décrivons l'algorithme génétique (AG) et l'algorithme d'optimisation par colonies de fourmis (ACO), avec leur adaptation au problème de conception du système étudié. Nous montrons la méthode de recherche locale que nous avons appliquée à notre problème pour les deux algorithmes.

3.5.2.1 LINGO

En raison de la complexité de l'expression de la disponibilité (équation 3.16) intégrée dans le modèle d'optimisation, une reformulation est alors exigée pour faire la résolution en utilisant LINGO. Cette expression comprend une somme sur les variables de décision, ainsi que des opérations factorielles. De telles opérations sont mises à jour comme suit :

Dans l'expression A_{i1}, la somme $\sum_{j=k_i}^{n_i-r_i+1}$ peut être remplacée par $\sum_{j=1}^{n_i^{max}} D(i,j)$ où $D(i,j)$ est un vecteur binaire défini par l'équation (3.25). Le terme $\frac{j!}{(k_i-1)!}$ provenant de $\prod_{m=k_i}^{j} g(m)$ (voir équation (3.9) peut être substitué par $\prod_{f=1}^{j} E(i,f)$ où $E(i,f)$ est un vecteur entier comme l'introduit l'équation (3.26). De ce fait, l'expression de A_{i1} peut être reformulée comme le présente l'équation (3.19) :

$$A_{i1} = \sum_{j=1}^{n_i^{max}} \frac{D(i,j)\prod_{f=1}^{j}(E(i,f))^{a-1}}{k_i^{a(j-k_i+1)}} \left(\frac{r_i\mu^i}{\lambda^i}\right)^{j-k_i+1} \tag{3.19}$$

De même, des vecteurs binaires et entiers, que nous les appelons respectivement S et T sont introduits pour contourner la limitation du LINGO lors de l'implémentation de A_{i2}. Ils sont représentés respectivement par les équations (3.27) et (3.28). A_{i2} est alors reformulé comme l'illustre l'équation (3.20) :

$$A_{i2} = \sum_{j=1}^{n_i^{max}} S(i,j)\frac{(\prod_{l=1}^{k_i-1} l) r_i^{n_i-r_i-k_i+2}(\prod_{m=k_i}^{j} m^a)(\prod_{q=1}^{j} T(i,q))}{(\prod_{f=1}^{j} f)(k_i^a)^{j-k_i+1}}(\frac{\mu^i}{\lambda^i})^{j-k_i+1} \tag{3.20}$$

Par conséquent, le modèle d'optimisation à implémenter sur LINGO est comme suit :

$$Minimiser\, C_s = \sum_{i=1}^{l}(n_i C_i^c + r_i C_i^r) \tag{3.21}$$

sous contraintes :

$$k_i \leq n_i \leq n_i^{max}, n_i \in \mathbb{N}^*, \qquad i = 1, 2, ..., l \tag{3.22}$$

$$1 \leq r_i \leq n_i - k_i + 1, r_i \in \mathbb{N}^*, \;\; i = 1, 2, ..., l \tag{3.23}$$

$$\begin{aligned} A_i = 1 - \Bigg(& 1 + \sum_{j=1}^{n_i^{max}} \frac{D(i,j)\prod_{f=1}^{j}(E(i,f))^{a-1}}{k_i^{a(j-k_i+1)}} \left(\frac{r_i \mu^i}{\lambda^i}\right)^{j-k_i+1} \\ & + \sum_{j=1}^{n_i^{max}} \frac{S(i,j)(\prod_{l=1}^{k_i-1} l) r_i^{n_i-r_i-k_i+2} (\prod_{m=k_i}^{j} m^a)(\prod_{q=1}^{j} T(i,q))}{(\prod_{f=1}^{j} f)(k_i^a)^{j-k_i+1}} (\frac{\mu^i}{\lambda^i})^{j-k_i+1} \Bigg)^{-1}, \;\; i = 1, 2, ..., l \end{aligned} \tag{3.24}$$

$$\forall i \in [1,l], \forall j \in [1, n_i^{max}], D(i,j) \in \{0,1\} = \begin{cases} 1, & \text{if } k_i \leq j \leq n_i - r_i + 1, \\ 0, & \text{otherwise} \end{cases} \tag{3.25}$$

$$E(i,j) = \begin{cases} 1, & \text{if } D(i,j) = 0, \\ j, & \text{otherwise} \end{cases} \tag{3.26}$$

$$\forall i \in [1,l], \forall j \in [1, n_i^{max}], S(i,j) \in \{0,1\} = \begin{cases} 1, & \text{if } n_i - r_i + 2 \leq j \leq n_i, \\ 0, & \text{otherwise} \end{cases} \tag{3.27}$$

$$T(i,j) = \begin{cases} 1, & \text{if } S(i,j) = 0, \\ n_i - j + 1, & \text{otherwise} \end{cases} \tag{3.28}$$

$$\prod_{i=1}^{l} A_i \geq A_0 \tag{3.29}$$

L'équation (3.21) présente la fonction objective. Les contraintes (3.22) et (3.23) montrent respectivement l'intervalle de variation des variables de décision n_i et r_i. L'équation (3.24) illustre l'expression de la disponibilité du système après la reformulation. Les contraintes (3.25)-(3.28) représentent les vecteurs introduits pour implémenter A_i sur LINGO. La dernière équation (3.29) présente la constrainte sur la disponibilité du système.

3.5.2.2 Algorithme Génétique

L'AG est appliqué comme le montre l'algorithme 1. Les étapes sont les suivantes :

Codage

Nous avons adopté le codage suivant. Chaque chromosome est un vecteur de $2l$ gènes comme indiqué ci-dessous où l est le nombre total de sous-systèmes. n_i et r_i représentent respectivement le nombre de composants et le nombre des équipes de réparation dans le sous-système i $(i = 1, ..., l)$. Ce sont les variables de décision à déterminer.

$$\underbrace{[n_1 \cdots n_i \cdots n_l}_{\text{nombre des composants}} \quad \underbrace{r_1 \cdots r_i \cdots r_l]}_{\text{nombre des équipes de réparation}}$$

Génération de la population initiale

Pour créer la population initiale, un ensemble de chromosomes est généré de manière aléatoire. La taille de la population est fixée à Pop. Les gènes des chromosomes prennent des valeurs entières et sont générés dans leur espace de recherche.

Evaluation des solutions

La fonction objective $F(x)$ de chaque solution x de la population est évaluée comme le montre l'équation (3.47).

$$F(x) = C_s(x) + M \times max\{0, A_0 - A_s(x)\}, M \text{ un grand nombre positif.} \tag{3.30}$$

Opération de croisement

Les individus de la population courante sont triés par ordre croissant en fonction des valeurs de leurs fonctions objectives. Les deux premiers individus (ayant des coûts minimums) sont sélectionnés comme parents. L'idée derrière cette sélection est de donner la préférence à de meilleurs individus, ce qui leur permet de transmettre leurs gènes à la génération suivante par l'intermédiaire d'un opérateur de croisement [158]. Le croisement en un et deux points a été testé avec une probabilité P_c. Le premier est adopté comme il a donné de meilleurs résultats. Ainsi, deux enfants sont créés en échangeant les gènes des deux parents selon le point de croisement choisi aléatoirement. Après l'opération de croisement, la faisabilité de la contrainte concernant les équipes de réparation est

vérifiée comme l'indique l'équation (3.31).

$$r_i = \begin{cases} r_i - k_i + 1, & \text{if } r_i > n_i - k_i + 1, \\ r_i, & \text{sinon} \end{cases} \quad pour\ i = 1, ..., l \tag{3.31}$$

Mutation

Dans cet algorithme, la mutation des nouveaux enfants obtenus se fait avec une probabilité P_m pour plus de diversification des solutions. Elle consiste à sélectionner de façon aléatoire un sous-système, soit i, et à diminuer la valeur de n_i et r_i par 1. Cela permet de réduire la fonction objective représentée par le coût du système. Les contraintes sur n_i et r_i sont ainsi vérifiées comme le montre l'équation (3.48).

$$\begin{cases} n_i = k_i, & \text{if } n_i < k_i. \\ r_i = 1, & \text{if } r_i < 1 \end{cases} \tag{3.32}$$

Remplacement et critère d'arrêt

Le remplacement élitiste est appliqué. Il consiste à remplacer les mauvais individus de la population par les meilleurs chromosomes obtenus. Ce processus de sélection, croisement, mutation et remplacement est répété pour $Pop/2$.

Les étapes décrites ci-dessus forme un cycle. Elles sont répétées jusqu'un critère d'arrêt est atteint, qui est égal au nombre maximum de cycles N_c. La meilleure solution obtenue au cours de tous ces cycles, que nous l'appelons $BestSol$, est sauvegardée.

3.5.2.2.1 Recherche locale pour l'AG hybride

Dans le cas de l'AG hybride, une recherche locale (LS) est appliquée dans le voisinage de la meilleure solution trouvée $BestSol$ selon que les contraintes du problème sont respectées ou pas. La démarche est présentée par l'algorithme 2. Elle est comme suit :

Si la disponibilié du système A_s donnée par cette solution est supérieure ou égale à la disponibilité exigée A_0, nous essayons alors de minimiser le coût de la configuration en diminuant le nombre de composants et des équipes de réparation. Pour ce faire, nous cherchons le sous-système, soit i, ayant le nombre maximum de composants n_i et satisfaisant $n_i > k_i$. Nous diminuons n_i par la valeur 1. En termes des équipes de réparation, si ce sous-système i possède $1 < r_i \leq n_i - k_i + 2$, nous diminuons aussi r_i par la valeur 1. Ainsi, le coût du système sera réduit. Par la suite, nous

Algorithme 1 Algorithme AG

Initialiser les paramètres
pour chaque cycle **faire**
  Générer aléatoirement une population de taille Pop
  Evaluer les chromosomes en calculant leur fonction objective selon l'équation (3.47)
  pour $i = 1$ à $Pop/2$ **faire**
    Sélectionner aléatoirement deux chromosomes
    Appliquer un croisement avec une probabilité P_c sur les chromosomes sélectionnés
    Vérifier les contraintes selon l'équation (3.31)
    Appliquer une mutation sur les chromosomes obtenus avec une probabilité P_m
    Vérifier les contraintes selon l'équation (3.48)
    Evaluer les nouveaux chromosomes obtenus
    Remplacer les mauvais individus de la population par les meilleurs individus obtenus
  fin pour
  Sauvegarder la meilleure solution
fin pour
Afficher la meilleure solution de tous les cycles, appelée $BestSol$
Lancer la recherche locale dans le cas de l'algorithme hybride

évaluons la disponibilité du système pour cette nouvelle configuration. Si la contrainte sur A_0 est satisfaite, la nouvelle solution ayant cette configuration est enregistrée, elle remplace l'ancienne valeur de $BestSol$. Ce processus est répété pour l fois (l est le nombre de sous-systèmes) et tant que la contrainte de disponibilité est remplie. De cette façon, nous augmentons la performance de la configuration tout en limitant l'augmentation des coûts avec la minimisation du nombre de composants et des équipes de réparation. Cette recherche locale permet aussi d'avoir une répartition presque égale des composants entre les sous-systèmes.

3.5.2.3 Algorithme d'optimisation par colonies des fourmis (ACO)

Codage des solutions et phase d'initialisation

Comme nous l'avons déjà mentionné que le système considéré est constitué de l sous-systèmes en série. Chaque sous-système $i(i = 1, ..., l)$ comprend n_i composants identiques et r_i équipes de réparation, avec $n_i \in [k_i, n_i^{max}]$ et $r_i \in [1, n_i - k_i + 1]$. Afin d'appliquer l'algorithme ACO à notre problème, nous avons adopté le codage présenté dans le tableau 3.2. Une configuration possible d'un sous-système $i(i = 1, ..., l)$ peut être représentée par le couple des variables (n_i, r_i) satisfaisant la condition $r_i \leq n_i - k_i + 1$. (n_i, r_i) peut être sélectionné à partir de l'ensemble des cas possibles variant de $(k_i, 1)$ à $(n_i^{max}, n_i^{max} - k_i + 1)$. Ensuite, pour chaque configuration de sous-système (n_i, r_i), la disponibilité A_i et le coût C_i correspondants sont évalués. A_i est calculée selon l'équation (3.5).

Algorithme 2 Procédure de la recherche locale LS pour l'AG hybride

Considérer la solution $BestSol$ obtenue comme la solution $NewSol$
pour $v = 1$ au nombre de sous-systèmes l **faire**
  si $A_s(NewSol) \geq A_0$ **alors**
    Trouver pour la solution $NewSol$ le sous-système i ayant le maximum nombre de composants n_i avec $n_i > k_i$
    Décrémenter n_i par 1 ($n_i = n_i - 1$)
    si le nombre des équipes de réparation r_i pour ce sous-système i vérifie $1 < r_i \leq n_i - k_i + 2$ **alors**
      Décrémenter r_i par 1 ($r_i = r_i - 1$)
    fin si
    Evaluer la solution $NewSol$ avec la configuration modifiée
    si $C_s(NewSol) < C_s(BestSol)$ et $A_s(NewSol) \geq A_0$ **alors**
      Remplacer $BestSol$ par $NewSol$
    fin si
  sinon
    Sortir de la procédure LS
  fin si
fin pour

Tableau 3.2 – Codage appliqué au problème étudié

Sous-système 1	$(k_1,1)$	$(k_1+1,1)$	$(k_1+1,2)$	$\cdots$	$(n_1^{max},1)$	$(n_1^{max},2)$	$\cdots$	$(n_1^{max}, n_1^{max}-k_1+1)$
Sous-système 2	$(k_2,1)$	$(k_2+1,1)$	$(k_2+1,2)$	$\cdots$	$(n_2^{max},1)$	$(n_2^{max},2)$	$\cdots$	$(n_2^{max}, n_2^{max}-k_2+1)$
$\vdots$			$\vdots$					
Sous-système i	$(k_i,1)$	$(k_i+1,1)$	$(k_i+1,2)$	$\cdots$	$(n_i^{max},1)$	$(n_i^{max},2)$	$\cdots$	$(n_i^{max}, n_i^{max}-k_i+1)$
$\vdots$			$\vdots$					
Sous-système l	$(k_l,1)$	$(k_l+1,1)$	$(k_l+1,2)$	$\cdots$	$(n_l^{max},1)$	$(n_l^{max},2)$	$\cdots$	$(n_l^{max}, n_l^{max}-k_l+1)$

C_i est évalué selon l'équation (3.33).

$$C_i = n_i C_i^c + r_i C_i^r \tag{3.33}$$

Un modèle basé sur un graphe orienté est appliqué afin de calculer les paramètres de l'ACO comme l'illustre la figure 3.3. Chaque colonne $i(i = 1, ..., l)$ de ce graphe regroupe toutes les configurations possibles (n_i, r_i) variant de $(k_i, 1)$ à $(n_i^{max}, n_i^{max} - k_i + 1)$ relatives au sous-système i. Des arcs sont utilisés pour connecter des noeuds de sous-systèmes consécutifs et sont caractérisés par deux métriques de désirabilité.

Soient $o \equiv (n_i, r_i)$ et $p \equiv (n_j, r_j)$ deux noeuds appartenant respectivement aux sous-systèmes consécutifs i et j. L'arc reliant les deux noeuds est caractérisé par une distance $d_{o,p}$ qui est exprimée

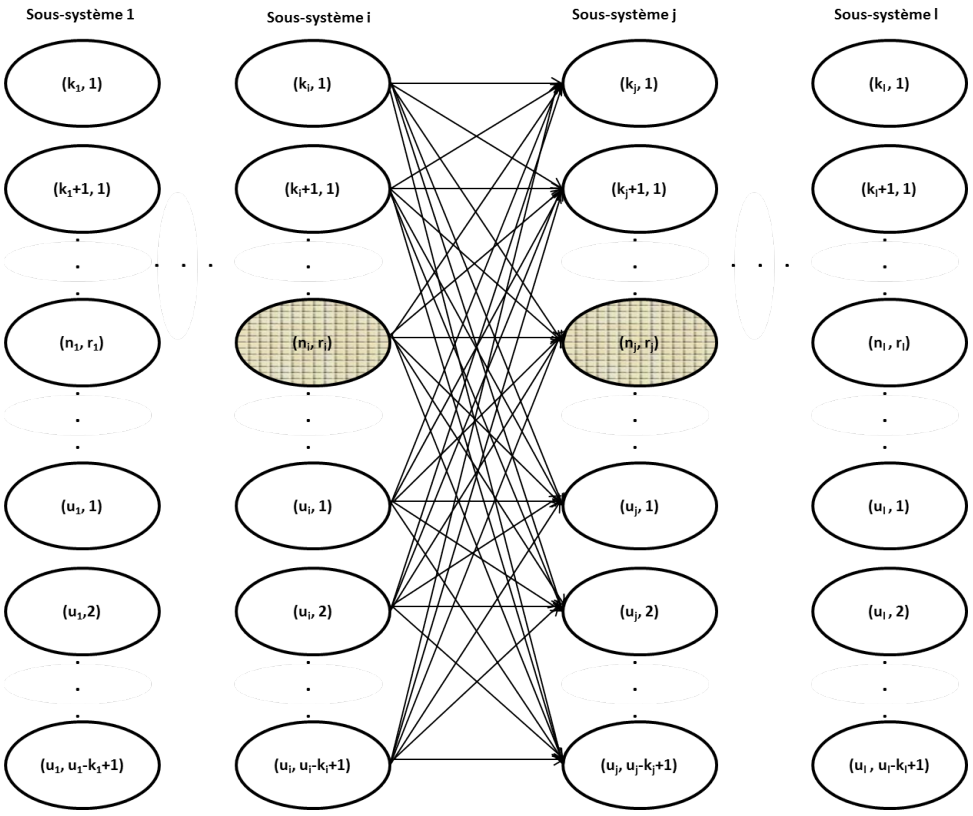


FIGURE 3.3 – Graphe orienté décrivant le probème RAP (u_i équivalent à n_i^{max})

en termes de coût comme le présente l'équation (3.34).

$$d_{o,p} = C_i + C_j \tag{3.34}$$

De plus, les deux noeuds liés par l'arc (o, p) contribuent en une partie de la disponibilité du système. Cette contribution désignée par $A_{o,p}$ est calculée selon l'équation (3.35).

$$A_{o,p} = A_i * A_j \tag{3.35}$$

Durant leur mouvements, les fourmis déposent sur les arcs visités des substances chimiques, appelées phéromones. Ainsi deux métriques caractérisent les arcs qui sont : $\tau_{o,p}$ et $\eta_{o,p}$. La première est la quantité de phéromone entre les points o et p. Initialement, elle est uniformément répartie selon l'équation (3.36).

$$\tau_0 = \frac{1}{N_t * mean(d)} \tag{3.36}$$

où les paramètres N_t et $mean(d)$ représentent respectivement le nombre total de noeuds et la valeur

moyenne des distances de tous les arcs présents dans le graphe (figure 3.3). Alors que la deuxième métrique représente la visibilité de l'arc. Elle dépend d'une heuristique spécifique au problème. Dans notre cas, nous considérons que la visibilité d'un arc augmente avec la disponibilité et diminue avec le coût associé, puisque nous essayons de réduire le coût de la configuration tout en satisfaisant la contrainte de la disponibilité. Par conséquent, elle est évaluée comme l'indique l'équation (3.37).

$$\eta_{o,p} = \frac{A_{o,p}}{d_{o,p}} \tag{3.37}$$

L'algorithme ACO est répété pour N_c cycles ou itérations dont chacun utilise N_a fourmis. Chaque fourmi construit son tour en choisissant le nouveau point à visiter en se basant sur les deux métriques de désirabilité. A la fin de son tour, elle construit une solution qui représente une configuration possible du système étudié.

Construction des chemins des fourmis

Dans un premier temps, chaque fourmi est déposée aléatoirement sur un point (noeud) de la première colonne du graphe, correspondant à la configuration du premier sous-système. Ensuite, la fourmi construit son chemin en choisissant le prochain point à visiter sur la colonne suivante. Chaque point du chemin construit représente une configuration d'un sous-système. Une fourmi A étant sur un point, soit r, choisit le prochain point à visiter, soit s, en appliquant une politique de décision stochastique. Cette dernière est basée sur l'information heuristique spécifique $\eta_{o,p}$ et la quantité de phéromone $\tau_{o,p}$ comme le présente l'équation (3.38).

$$p = \begin{cases} argmax_{v \in J_A(o)}([\tau_{o,v}]^\alpha [\eta_{o,v}]^\beta) & \text{si } q \leq q_0 \\ S & \text{sinon.} \end{cases} \tag{3.38}$$

S est une variable aléatoire sélectionnée selon une probabilité présentée par l'équation (3.39).

$$S = \begin{cases} \frac{([\tau_{o,p}]^\alpha [\eta_{o,p}]^\beta)}{\sum_{v \in J_A(o)} [\tau_{o,v}]^\alpha [\eta_{o,v}]^\beta} & \text{si } p \in J_A(o) \\ 0 & \text{sinon.} \end{cases} \tag{3.39}$$

où les paramètres α et β sont utilisés pour donner plus d'importance soit à la visibilité soit à la quantité de phéromones. $J_A(o)$ est l'ensemble des points non encore visités par la fourmi A. q est une valeur aléatoire générée entre 0 et 1. $q_0 (0 \leq q_0 \leq 1)$ est un paramètre qui donne plus

d'importance soit à l'exploitation (intensification) soit à l'exploration (diversification).

Mises à jour locale et globale

En construisant son chemin, une fourmi change la quantité de phéromones en appliquant deux types de mise à jours : local et global. Le premier se fait sur chaque arc visité en modifiant sa quantité de phéromone selon l'équation (3.40). Cela permet d'éviter d'avoir une convergence prématurée et de décourager la fourmi suivante de choisir le même arc à la même itération.

$$\tau_{o,p} = (1-\rho)\tau_{o,p} + \rho\tau_0 \tag{3.40}$$

où $\rho \in (0,1)$ et $(1-\rho)$ est le taux d'évaporation de phéromones. τ_0 est la quantité initiale de phéromone calculée selon l'équation (3.36).

Concernant la mise à jour globale, une fois que toutes les fourmis ont fini leurs tours, la quantité de phéromones sur chaque arc est modifiée selon une fonction de mise à jour globale comme le montre l'équation (3.41).

$$\tau_{o,p} = (1-\rho)\tau_{o,p} + \rho\Delta\tau_{o,p} \tag{3.41}$$

où $\Delta\tau_{o,p}$ est un facteur qui consiste à favoriser les meilleures solutions trouvées jusqu'à présent. Il est calculé selon l'équation (3.42)

$$\Delta\tau_{o,p} = \begin{cases} \frac{1}{C_{best}} & \text{si } o,p \text{ appartiennent à la meilleure solution} \\ 0 & \text{sinon.} \end{cases} \tag{3.42}$$

où C_{best} est le coût du système correspondant à la meilleure solution trouvée depuis le début des essais. Avec ces mises à jour des phéromones, les arcs qui disposent des plus grandes quantités de phéromone attirent plus de fourmis à la prochaine étape. L'algorithme 3 présente la structure globale de l'algorithme.

3.5.2.3.1 Recherche locale pour l'ACO hybride

Dans le cas de l'algorithme ACO hybride, une recherche locale (LS) est appliquée comme le montre l'algorithme 4. Elle est comme suit :

Une fois que la fourmi a fini son tour et avant de réaliser une mise à jour globale des phéromones, nous appliquons une recherche locale qui est basée sur la recherche en voisinage de la solution

Algorithme 3 Algorithme ACO

Initialiser les paramètres et les phéromones
Placer aléatoirement les N_a fourmis sur les points représentant les configurations du premier sous-système
pour chaque sous-système $i = 2$ à l **faire**
  Choisir les prochaines points à visiter en se basant sur les équations (3.38) et (3.39)
fin pour
Retour des fourmis aux points de départ
Mise à jour locale des phéromones selon l'équation (3.40)
pour les N_a fourmis **faire**
  Evaluer la solution trouvée par la fourmi
  Lancer la recherche locale dans le cas de l'algorithme hybride
fin pour
Retenir la meilleure solution
Mise à jour globale des phéromones selon l'équation (3.41)
si le critère d'arrêt est satisfait **alors**
  Arrêter l'algorithme et sortir
sinon
  Retour des fourmis aux points de départ et reconstruction de nouveaux tours
fin si

actuelle trouvée, appelée $CurrentSol$ selon que les contraintes du problème sont respectées ou pas. En effet, si la solution donnée par cette fourmi assure une disponibilié du système A_s qui est supérieure ou égale à la disponibilité requise A_0, nous appliquons la même stratégie de LS décrite dans la section (3.5.2.2.1) pour l'AG.

Le sous-système i ayant le maximum n_i avec $n_i > k_i$ va perdre un composant. La variable r_i correspondante est également diminuée de la valeur 1 si $1 < r_i \leq n_i - k_i + 2$. Alors que les configurations des autres sous-systèmes restent les mêmes. Une nouvelle solution $NewSol$ est ainsi obtenue. Elle aura évidemment un coût moindre que le coût de la solution actuelle. Si $NewSol$ respecte la contrainte de disponibilité du système, la fourmi remplace sa solution $CurrentSol$ par $NewSol$.

Ce processus est répété jusqu'à l fois et tant que la contrainte sur la disponibilité du système est satisfaite.

3.5.2.4 Méthode exacte

La méthode d'énumération complète (FEM) est implémentée. Elle consiste à énumérer toutes les solutions possibles et à trouver la solution optimale du problème qui vérifie les différentes contraintes.

Algorithme 4 Procédure de la recherche locale LS pour l'ACO hybride

Considérer la solution $CurrentSol$ proposée par la fourmi comme la solution $NewSol$
pour $v = 1$ au nombre de sous-systèmes l **faire**
  si $A_s(NewSol) \geq A_0$ **alors**
    Trouver pour la solution $NewSol$ le sous-système i ayant le maximum nombre de composants n_i avec $n_i > k_i$
    Décrémenter n_i par 1 ($n_i = n_i - 1$)
    si le nombre des équipes de réparation r_i pour ce sous-système i vérifie $1 < r_i \leq n_i - k_i + 2$ **alors**
      Décrémenter r_i par 1 ($r_i = r_i - 1$)
    fin si
    Evaluer la solution $NewSol$ avec la configuration modifiée
    si $C_s(NewSol) < C_s(CurrentSol)$ et $A_s(NewSol) \geq A_0$ **alors**
      Remplacer $CurrentSol$ par $NewSol$
    fin si
  sinon
    Sortir de la procédure LS
  fin si
fin pour

3.5.3 Réglage des paramètres

Le règlage des paramètres est un problème essentiel pour les métaheuristiques qui sont connues par leur nature stochastique. Cela peut affecter les performances algorithmiques. Par conséquent, un plan expérimental devrait être réalisé pour trouver les valeurs de paramètres les plus adéquates pour le processus d'optimisation [159],[71]. Dans cette étude, un test représente l'exécution de l'algorithme avec une combinaison possible des valeurs des paramètres. Nous avons sélectionné le nombre de cycles comme un critère d'arrêt. Tous les tests sont effectués pour $N_c = 500$ et sont répétés sur dix essais. Il est à noter qu'un nombre élevé de cycles peut être coûteux en termes de temps d'exécution. De plus, les essais répétés peuvent fournir des informations statistiques utiles concernant l'évolution moyenne de l'algorithme. De ce fait, nous répétons chaque test 10 fois. Nous enregistrons la valeur minimale (Min), la valeur moyenne (Moyenne), l'écart-type (Ecart-type) et la valeur maximale (Max) du coût du système obtenu pour les 10 simulations. Le test donnant la plus petite valeur de coût sera choisi. Dans le cas où plusieurs tests ont la même valeur minimale de coût, nous choisissons celui qui a pris le minimum temps d'exécution.

En général, lorsque la métaheuristique est appliquée avec une recherche locale, un nouveau réglage des paramètres est exigé. Dans cette partie, nous appliquons deux versions de l'AG (avec et sans LS) ainsi que deux versions de l'ACO (avec et sans LS). Pour chacun de ces algorithmes, nous considérons v paramètres, appelés aussi facteurs dont la variation peut affecter les performances

du processus d'optimisation. Nous testons pour chaque paramètre deux niveaux indiqués par L et H qui correspondent respectivement à une valeur basse et haute. Par conséquent, nous avons 2^v différents tests à faire.

3.5.3.1 Réglage des paramètres de l'AG

Les facteurs concernés sont la taille de la population Pop, la probabilité de croisement P_c et la probabilité de mutation P_m comme ce sont les paramètres les plus sensibles de l'algorithme génétique [71]. Les valeurs correspondantes à tester sont illustrées dans le tableau 3.3. Ces valeurs sont considérées dans de nombreuses recherches de la littérature [30], [160].

Tableau 3.3 – Facteurs de plan d'expérience de l'AG

Paramètres	L	H
$F_1 = Pop$	50	90
$F_2 = P_c$	0.5	0.8
$F_3 = P_m$	0.05	0.1

3.5.3.2 Réglage des paramètres de l'ACO

Concernant les paramètres de l'ACO, nous avons fixé $q_0 = 0.5$ pour accorder une même importance à l'exploitation et à l'exploration comme dans le travail de Sharma et Agarwal [91]. De plus, un équilibre approprié entre α et β est souhaité, c-à-d $\alpha + \beta = 1$ comme dans [161]. Parsuite, pour une valeur fixe de α, le paramètre β peut être obtenu facilement par $\beta = 1 - \alpha$. Dans cette étude, nous considérons deux valeurs pour α qui sont $0,3$ et $0,7$. La première valeur donne plus de poids à l'heuristique spécifique du problème qu'à la quantité de phéromone, tandis que la seconde valeur montre le cas inverse. En termes de paramètre d'évaporation ρ, deux valeurs sont testées : 0.4 et 0.9. En termes de nombre de fourmis, les valeurs suivantes sont proposées : 50 et 100. Il est à noter que les valeurs choisies ci-haut sont basées sur plusieurs travaux de la littérature [162],[91]. Le tableau 3.4 recapitulate les facteurs considérés pour l'ACO.

En se basant sur la description ci-dessus, trois facteurs $(F_1, F2, F3)$ sont considérés pour le réglage des paramètres pour chaque algorithme. Chaque facteur a deux valeurs possibles. Par conséquent, il existe 8 tests à faire. La matrice des tests est présentée dans le tableau 3.5 où $(F_1, F2, F3)$ représente (Pop, P_c, P_m) pour l'AG et (α, ρ, N_a) pour l'ACO. Par exemple, selon le plan d'expérience utilisé,

Tableau 3.4 – Facteurs de plan d'expérience de l'ACO

Paramètres	L	H
$F_1 = \alpha$	0.3	0.7
$F_2 = \rho$	0.4	0.9
$F_3 = N_a$	50	100

Tableau 3.5 – Matrice des tests effectués

Exp. Num	F_1	F_2	F_3
1	L	L	L
2	H	L	L
3	L	H	L
4	H	H	L
5	L	L	H
6	H	L	H
7	L	H	H
8	H	H	H

le test numéro 1 consiste à exécuter l'AG avec les paramètres $(Pop, P_c, P_m) = (50, 0.5, 0.05)$ et l'ACO avec les paramètres $(\alpha, \rho, N_a) = (0.3, 0.4, 50)$.

Tableau 3.6 – Ensemble de données des exemples testés

Sous-système i	λ^i	μ^i	C_i^c	C_i^r
1	0.03	0.10	40	15
2	0.04	0.13	50	20
3	0.05	0.14	30	10
4	0.06	0.20	70	30
5	0.07	0.18	65	25
6	0.09	0.27	80	35

3.6 Applications numériques

Afin d'évaluer la performance de l'approche proposée, nous considérons une reformulation de l'exemple utilisé dans [143] pour répondre au problème étudié dans ce chapitre. Nous appliquons les métaheuristiques : l'AG et l'ACO dans les deux cas sans et avec LS pour différentes tailles de problèmes. Dans notre étude, la taille du problème dépend principalement des facteurs suivants :

Tableau 3.7 – Meilleures solutions obtenues pour les exemples $E1$, $E2$ et $E3$ par les différentes métaheuristiques avec les détails statistiques

		AG							AG + recherche locale						
	a	Meilleur test	$(n_1 \ldots n_l, r_1 \ldots r_l)$	A_s	Min	Moyenne	Ecart-type	Max	Meilleur test	$(n_1 \ldots n_l, r_1 \ldots r_l)$	A_s	Min	Moyenne	Ecart-type	Max
$E1$	0	1	(2, 2)	0.9467	**110**	110	0	110	1	(2, 2)	0.9467	**110**	110	0	110
	0.5	1	(2, 1)	0.918	**95**	95	0	95	1	(2, 1)	0.918	**95**	95	0	95
	1	1	(2, 1)	0.9353	**95**	95	0	95	1	(2, 1)	0.9353	**95**	95	0	95
	1.5	1	(2, 1)	0.9501	**95**	95	0	95	1	(2, 1)	0.9501	**95**	95	0	95
		ACO							ACO + recherche locale						
	a	Meilleur test	$(n_1 \ldots n_l, r_1 \ldots r_l)$	A_s	Min	Moyenne	Ecart-type	Max	Meilleur test	$(n_1 \ldots n_l, r_1 \ldots r_l)$	A_s	Min	Moyenne	Ecart-type	Max
	0	1	(2, 2)	0.9467	**110**	110	0	110	1	(2, 2)	0.9467	**110**	110	0	110
	0.5	1	(2, 1)	0.918	**95**	95	0	95	1	(2, 1)	0.918	**95**	95	0	95
	1	1	(2, 1)	0.9353	**95**	95	0	95	1	(2, 1)	0.9353	**95**	95	0	95
	1.5	1	(2, 1)	0.9501	**95**	95	0	95	1	(2, 1)	0.9501	**95**	95	0	95
		AG							AG + recherche locale						
	a	Meilleur test	$(n_1 \ldots n_l, r_1 \ldots r_l)$	A_s	Min	Moyenne	Ecart-type	Max	Meilleur test	$(n_1 \ldots n_l, r_1 \ldots r_l)$	A_s	Min	Moyenne	Ecart-type	Max
$E2$	0	7	(3 3 4 3 3 3, 3 2 3 2 3 2)	0.9025	**1355**	1369.5	10.9163	1390	7	(3 3 4 3 3 3, 3 2 3 2 3 2)	0.9025	**1355**	1369.5	10.9163	1390
	0.5	8	(3 3 3 2 3 3, 3 2 3 2 2 2)	0.9012	**1230**	1243.5	**9.1439**	1255	8	(3 3 3 2 3 3, 3 2 3 2 2 2)	0.9012	**1230**	1234.5	**5.9861**	1250
	1	6	(3 3 3 2 3 2, 2 2 2 2 2 2)	0.902	**1125**	1138	**9.1894**	1155	2	(3 3 3 2 3 2, 2 2 2 2 2 2)	0.902	**1125**	1146	**8.756**	1155
	1.5	5	(3 2 3 2 3 2, 1 2 2 2 2 2)	0.9031	**1060**	1072	**5.3748**	1080	4	(3 2 3 2 3 2, 1 2 2 2 2 2)	0.9031	**1060**	1068	**4.8305**	1075
		ACO							ACO + recherche locale						
	a	Meilleur test	$(n_1 \ldots n_l, r_1 \ldots r_l)$	A_s	Min	Moyenne	Ecart-type	Max	Meilleur test	$(n_1 \ldots n_l, r_1 \ldots r_l)$	A_s	Min	Moyenne	Ecart-type	Max
	0	7	(3 3 4 3 3 3, 3 2 3 2 3 2)	0.9025	**1355**	1402.5	**21.7626**	1430	8	(3 3 4 3 3 3, 3 2 3 2 3 2)	0.9025	**1355**	1370	**7.0711**	1375
	0.5	5	(3 3 3 2 3 3, 3 2 3 2 2 2)	0.9012	**1230**	1275.5	**20.3374**	1295	2	(3 3 3 2 3 3, 3 2 3 2 2 2)	0.9012	**1230**	1233.5	**2.4152**	1235
	1	1	(3 3 3 2 3 2, 2 2 2 2 2 2)	0.902	**1125**	1168	**28.8868**	1205	6	(3 3 3 2 3 2, 2 2 2 2 2 2)	0.902	**1125**	1134	**6.9921**	1145
	1.5	7	(3 2 3 2 3 2, 1 2 2 2 2 2)	0.9031	**1060**	1085.5	**12.7911**	1100	7	(3 2 3 2 3 2, 1 2 2 2 2 2)	0.9031	**1060**	1064.5	**4.378**	1070
		AG							AG + recherche locale						
	a	Meilleur test	$(n_1 \ldots n_l, r_1 \ldots r_l)$	A_s	Min	Moyenne	Ecart-type	Max	Meilleur test	$(n_1 \ldots n_l, r_1 \ldots r_l)$	A_s	Min	Moyenne	Ecart-type	Max
$E3$	**0**	8	(5 5 6 5 5 5, 3 4 3 3 3 3)	0.9024	**2130**	2190.5	26.3997	2225	8	(5 5 6 5 5 5, 3 4 3 3 3 3)	0.9024	**2130**	2190.5	26.3997	2225
	0.5	5	(5 5 5 4 5 5 , 2 2 4 3 3 3)	0.901	**1985**	2032.4	30.8115	2065	2	(5 5 5 4 5 4, 4 4 4 3 3 3)	0.9003	**1975**	2000	17.7951	2025
	1	4	(4 5 5 4 5 4, 3 2 3 3 3 2)	0.9007	**1835**	1870.5	**17.3925**	1895	4	(4 5 5 4 5 4, 3 2 3 3 3 2)	0.9007	**1835**	1862.5	**12.304**	1880
	1.5	6	(4 4 5 4 4 4, 3 3 4 2 3 2)	0.9022	**1720**	1759	19.6921	1780	6	(4 4 5 4 5 4, 2 2 3 2 2 2)	**0.90341715**	1751	15.0555	1765	
		ACO							ACO + recherche locale						
	a	Meilleur test	$(n_1 \ldots n_l, r_1 \ldots r_l)$	A_s	Min	Moyenne	Ecart-type	Max	Meilleur test	$(n_1 \ldots n_l, r_1 \ldots r_l)$	A_s	Min	Moyenne	Ecart-type	Max
	0	1	(5 5 6 5 5 5, 3 3 4 3 4 3)	0.9103	**2145**	2171	14.4914	2195	5	(5 5 6 5 5 5, 4 3 3 3 3 3)	0.9022	**2125**	2169.5	17.3925	2185
	0.5	1	(5 5 6 4 5 5, 3 2 5 3 3 2)	0.9051	**2005**	2083.5	56.1768	2165	5	(5 5 6 4 5 4, 3 3 4 3 3 3)	0.9032	**1970**	2019	24.2441	2045
	1	7	(4 5 5 4 5 4, 3 3 3 2 3 3)	0.908	**1860**	1947	**47.4459**	2000	7	(5 5 5 4 5 4,2 2 3 2 3 2)	0.902	**1830**	1859	**14.4914**	1885
	1.5	5	(4 4 4 4 6 4, 3 2 3 2 2 2)	0.9003	**1765**	1838.5	49.4441	1890	3	(5 4 5 4 4 4, 2 2 3 2 3 2)	**0.90361715**	1740	15.9861	1765	

Tableau 3.8 – Meilleures solutions obtenues pour l'exemple $E4$ par les différentes métaheuristiques avec les détails statistiques

		AG					AG + recherche locale					
	a	Meilleur test	$(n_1...n_l, r_1...r_l)$	A_s	Min	Moyenne	Ecart-type	Max	Meilleur test	$(n_1...n_l, r_1...r_l)$	A_s	Min	Moyenne	Ecart-type	Max
$E4$	0	7	(7 7 7 7 7 7 7, 4 6 6 4 6 5)	0.9904	**3030**	3117.5	55.6402	3190	2	(7 7 8 7 7 7 7, 6 4 6 4 6 4)	0.9906	**3015**	3089.5	38.7621	3135
	0.5	8	(7 6 7 6 6 6 6, 5 4 5 4 4 4)	0.9901	**2645**	2676	21.9596	2720	3	(7 6 7 6 7 6 7, 3 4 4 4 4 3)	0.9902	**2635**	2707.8	34.6189	2743.3
	1	6	(6 6 7 5 6 6 6, 3 3 3 3 4 3)	0.9914	**2400**	2432	17.8263	2450	6	(6 6 6 5 6 6 6,4 4 3 3 3)	0.9905	**2390**	2430	18.2574	2450
	1.5	6	(6 5 6 5 6 5 5, 3 3 4 2 3 3)	0.9903	**2195**	2216	11.005	2230	3	(5 6 6 5 6 5 5, 3 2 4 3 2 3)	0.9905	**2190**	2237	24.6306	2275
		ACO							ACO + recherche locale						
	a	Meilleur test	$(n_1...n_l, r_1...r_l)$	A_s	Min	Moyenne	Ecart-type	Max	Meilleur test	$(n_1...n_l, r_1...r_l)$	A_s	Min	Moyenne	Ecart-type	Max
	0	5	(7 7 8 7 7 7 7, 5 4 6 4 5 5)	0.9908	**3010**	3101	40.1248	3140	7	(7 7 8 7 7 7,5 4 5 4 6 4)	0.9902	**2990**	3035.5	29.1976	3080
	0.5	2	(6 6 6 6 8 6, 5 4 5 4 3 4)	0.9901	**2680**	2759	41.7532	2815	8	(6 6 7 6 6 6, 5 4 5 4 5 4)	0.99	**2630**	2653	15.4919	2685
	1	8	(7 6 6 5 6 6, 3 3 5 3 3 3)	0.9906	**2405**	2482	45.3505	2545	2	(6 6 6 5 6 6, 4 3 4 3 3 3)	0.9902	**2370**	2427.5	23.8339	2455
	1.5	6	(6 5 6 5 6 6, 3 3 2 2 3 2)	0.9904	**2220**	2298	37.0585	2340	5	(5 5 6 5 6 5, 3 3 3 3 3 3)	0.9907	**2175**	2228.5	24.5006	2260

Tableau 3.9 – Résultats numériques des métaheuristiques hybrides, FEM et LINGO

		FEM		LINGO		AG + recherche locale				ACO + recherche locale			
	a	C_s	T	C_s	T	C_s	T	RDP^b	RDP^c	C_s	T	RDP^b	RDP^c
$E1$	0	110	0.0119 s	110	1.11 s	110	3.23 s	0	0	110	2.60 s	0	0
	0.5	95	0.0115 s	95	1.01 s	95	3.11 s	0	0	95	2.71 s	0	0
	1	95	0.0079 s	95	0.39 s	95	1.80 s	0	0	95	2.70 s	0	0
	1.5	95	0.0121 s	95	0.28 s	95	3.25 s	0	0	95	2.70 s	0	0
$E2$	0	1355	3.47 h	1380$^\times$	24 h	1355	14.21 s	0	−1.81	1355	6.91 s	0	−1.81
	0.5	1230	2.1052 h	1435$^\times$	24 h	1230	14.26 s	0	−14.29	1230	14.26 s	0	−14.29
	1	1125	1.4141 h	1310$^\times$	24 h	1125	4.62 s	0	−14.12	1125	11.50 s	0	−14.12
	1.5	1060	2.9167 h	1150$^\times$	24 h	1060	12.17 s	0	−7.83	1060	11.71 s	0	−7.83
$E3$	0	2115	11.4312 jours	4095$^\times$	24 h	2130	12.91 s	0.70	−47.99	2125	10.16 s	0.47	−48.11
	0.5	1960	11.4787 jours	3380$^\times$	24 h	1975	11.41 s	0.76	−41.57	1970	9.21 s	0.51	−41.72
	1	1830	10.5256 jours	4025$^\times$	24 h	1835	7.28 s	0.27	−54.41	1830	5.29 s	0	−54.53
	1.5	1710	11.5681 jours	4095$^\times$	24 h	1715	13.16 s	0.29	−58.12	1715	11.87 s	0.29	−58.12
$E4$	0	2980	15.9112 jours	N/A	N/A	3015	21.93 s	1.17	N/A	2990	8.20 s	0.33	N/A
	0.5	2615	16.1123 jours	N/A	N/A	2635	22.47 s	0.76	N/A	2630	8.31 s	0.57	N/A
	1	2370	13.9132 jours	N/A	N/A	2390	7.88 s	0.84	N/A	2370	7.89 s	0	N/A
	1.5	2175	17.3507 jours	N/A	N/A	2190	20.13 s	0.68	N/A	2175	12.38 s	0	N/A

$^\times$ signifie qu'aucune solution n'est donnée par LINGO pendant 24 h d'exécution. LINGO est interrompu et la solution est affichée.

N/A signifie qu'aucune solution faisable n'est donnée par LINGO jusqu'à 24 h d'exécution.

RDP^b Pourcentage de déviation relative entre le coût de la métaheuristique et celui de FEM $\frac{C_s(métaheuristique)-C_s(FEM)}{C_s(FEM)}100\%$

RDP^c Pourcentage de déviation relative entre le coût de la métaheuristique et LINGO $\frac{C_s(métaheuristique)-C_s(LINGO)}{C_s(LINGO)}100\%$

- Le nombre l de sous-systèmes qui constituent le système entier.
- La taille de l'espace de recherche des composants $[k, n^{max}]$ et la contrainte de la disponibilité requise A_0.

Pour cela, nous testons 4 exemples de tailles différentes (l, k, n^{max}, A_0) comme suit :

- Exemple $E1 = (1, 1, 5, 0.9)$
- Exemple $E2 = (6, 1, 5, 0.9)$
- Exemple $E3 = (6, 2, 9, 0.9)$
- Exemple $E4 = (6, 2, 9, 0.99)$

Sans perdre de généralité, nous supposons que le nombre minimum de composants requis dans chaque sous-système est égal à k $(k_i = k, i = 1, ..., l)$. De même, le nombre maximum de composants autorisés dans chaque sous-système est égal à n^{max} $(n_i^{max} = n^{max}, i = 1, ..., l)$. Lorsque le système est constitué de l sous-systèmes, les données d'entrée correspondantes sont tirées des premiers l sous-systèmes présentés dans le tableau 3.6. Ce tableau présente le taux de défaillance nominal, le taux de réparation et le coût unitaire des composants et des équipes de réparation pour chaque sous-système. Nous nous intéressons à l'optimisation de la conception du système série-k/n en tenant compte de la dépendance redondante. L'intensité de la dépendance varie avec le paramètre de dépendance a comme décrit dans la deuxième section. Par conséquent, 4 valeurs de a sont considérées : 0 (indépendance), 0.5 (dépendance faible), 1 (dépendance linéaire) et 1.5 (dépendance forte) pour chaque instance testée.

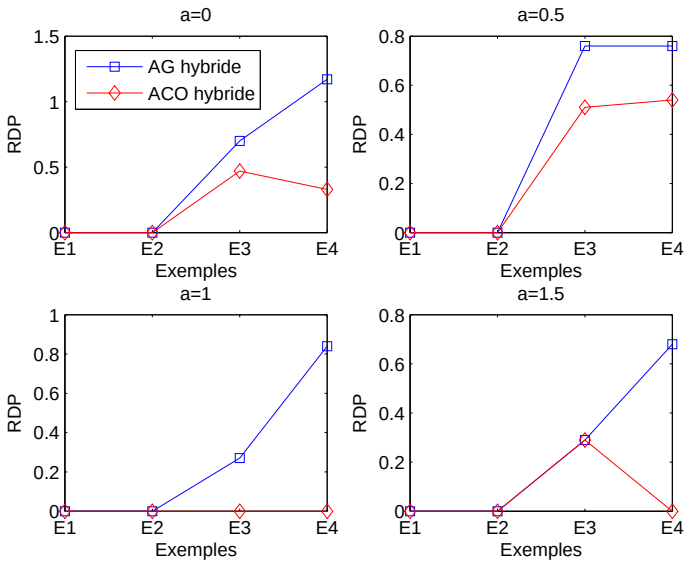


FIGURE 3.4 – Pourcentage de déviation relative entre le coût de l'AG hybride et FEM et entre l'ACO hybride et FEM pour les 4 exemples testés

La performance des algorithmes proposés est comparée aux résultats obtenus par LINGO et la méthode exacte FEM. Cette dernière est basée sur une énumération complète de toutes les alternatives possibles de l'espace de recherche. Pour les exemples $E1$ et $E2$, il y a respectivement 15 et 11390625 solutions possibles du problème. Alors que chacun des exemples $E3$ et $E4$ a 2.1768×10^9 solutions possibles. La simulation est implémentée en utilisant MATLAB R2011a et le solveur LINGO 16 sur un ordinateur avec un processeur Intel Core $i5-4590$ de 3.3 GHz et 8 GB de RAM. Nous avons fixé le temps d'exécution maximum autorisé pour LINGO à 24 h. Si ce temps s'est écoulé sans solution donnée, nous interrompons l'exécution et nous vérifions les résultats (présence d'une solution faisable ou pas).

Pour chacun des exemples ($E1, E2, E3$ et $E4$) et pour chaque niveau de dépendance $a (a \in \{0, 0.5, 1, 1.5\})$, nous avons appliqué le plan expérimental décrit ci-haut. Les tableaux 3.7 et 3.8 présentent le meilleur test obtenu (Meilleur test) et les meilleures solutions données par toutes les instances testées. Le nombre des composants redondants n_i et des équipes de réparation r_i dans chaque sous-système i $(i = 1, ..., l)$, la disponibilité du système A_s, la valeur minimale (Min), la valeur moyenne (Moyenne), l'écart-type (Ecart-type) et la valeur maximale (Max) du coût du système obtenus au cours de 10 simulations du meilleur test sont affichés.

Selon les résultats obtenus, nous pouvons noter que pour les exemples $E1$ et $E2$, tous les algorithmes ont donné la même solution, c'est la solution optimale. Tandis que, lorsque la taille

du problème augmente comme dans les exemples $E3$ et $E4$, il est clair que chacun de l'AG et de l'ACO a donné de meilleures solutions (coût égal ou plus petit) lorsqu'il est couplé avec une recherche locale. Dans le cas d'un coût égal, les algorithmes hybrides ont présenté de meilleurs résultats statistiques. Leurs écart-types sont inférieurs à ceux donnés par les algorithmes sans LS. Ces détails sont marqués en gras. En outre, nous pouvons déduire que plusieurs configurations de conception et par conséquent, différentes valeurs du coût et de la disponibilité du système peuvent être associées aux différents niveaux de dépendance redondante. Par exemple, dans le cas de $E4$ où le système considéré est composé de six sous-systèmes 2/9 en série avec une disponibilité requise égale à $A_0 = 0,99$. Les solutions ayant le minimum coût du système ont été données par l'ACO hybride. Nous pouvons constater que dans le cas des composants indépendants ($a = 0$), le système comprend 43 composants et 28 équipes de réparation donnant un coût du système égal à 2990. Alors que, lorsque l'intensité de la dépendance redondante passe de faible, à linéaire puis à forte, le nombre de composants correspondant sont respectivement 37, 35 et 32. Les équipes de réparation sont respectivement 27, 20 et 18. Le coût du système est égal à 2630 pour la dépendance faible, à 2370 pour la dépendance linéaire et à 2175 pour la dépendance forte. Par conséquent, nous pouvons conclure qu'un niveau suffisamment élevé de dépendance peut aider à concevoir des systèmes plus économiques. Il nécessite moins de composants et des équipes de réparation. Cette conclusion est également vraie pour les autres exemples testés ($E2$, $E3$ et $E4$).

Les résultats des meilleurs algorithmes obtenus sont ensuite comparés à ceux de LINGO et de FEM comme indiqué dans le tableau 3.9 pour 16 instances. La performance est évaluée en fonction du coût du système C_s, du temps d'exécution pris T, du pourcentage de déviation relative par rapport à FEM (RDP^b) et à LINGO (RDP^c). Une analyse des résultats révèle que l'AG hybride et l'ACO hybride ont réussi à obtenir les solutions optimales (ayant $RDP^b = 0$) pour 8 et 11 instances respectivement pendant un temps d'exécution rapide. Concernant les cas restants, le pourcentage de déviation de coût par rapport à FEM (RDP^b) était aussi très faible. D'autre part, LINGO a donné des solutions optimales uniquement pour le problème de petite taille $E1$. Pour les problèmes $E2$ et $E3$, il a donné des solutions faisables mais qui ne sont pas optimales après 24 h d'exécution. Tandis que, pour le problème $E4$ ayant une taille relativement grande, aucune solution faisable n'a été trouvée.

Concernant la méthode exacte FEM, elle a atteint les solutions optimales, mais son temps d'exécution était très long pour les problèmes de grande taille ($E3$ et $E4$). Par conséquent, les métaheuristiques hybrides proposées sont capables de résoudre efficacement le problème décrit.

De plus, le pourcentage de déviation relative de coût de chacun de ces algorithmes hybrides par rapport à FEM est présenté dans la figure 3.4 pour les différents niveaux de dépendance et pour les quatre exemples traités. Nous pouvons en déduire que pour tous ces tests, l'ACO hybride présente une performance supérieure ou égale à celle de l'AG hybride.

3.7 Problème PDU

3.7.1 Modèle mathématique

Dans cette partie, le problème décrit ci-haut pour le système série k/n est étudié en configuration duale. Le même concept de la dépendance redondante ainsi que les mêmes expressions de la disponibilité A_s, du coût C_s du système et de la fonction de dépendance redondante $g(m)$ (équation (3.8)) sont adoptées. La différence existe dans le modèle d'optimisation où la maximisation de la disponibilité est la fonction objective, alors que le coût du système est considéré parmi les contraintes. Ce dernier doit respecter une contrainte de budget C_{max}. Le modèle mathématique est alors le suivant :

$$\text{Maximiser } A_s = \prod_{i=1}^{l} \left\{ 1 - \left(1 + \sum_{j=k_i}^{n_i - r_i + 1} \frac{(k_i-1)! r_i^{j-k_i+1} \prod_{m=k_i}^{j} g(m)}{j!} \left(\frac{\mu^i}{\lambda^i} \right)^{j-k_i+1} + \sum_{j=n_i-r_i+2}^{n_i} \frac{(k_i-1)! r_i^{n_i-r_i-k_i+2} (r_i-1)!}{j!(n_i-j)!} \prod_{m=k_i}^{j} g(m) \left(\frac{\mu^i}{\lambda^i} \right)^{j-k_i+1} \right)^{-1} \right\} \quad (3.43)$$

sous contraintes :

$$C_s = \sum_{i=1}^{l} (n_i C_i^c + r_i C_i^r) \leq C^{max} \quad (3.44)$$

$$k_i \leq n_i \leq n_i^{max}, n_i \in \mathbb{N}^*, \qquad i = 1, 2, ..., l \quad (3.45)$$

$$1 \leq r_i \leq n_i - k_i + 1, r_i \in \mathbb{N}^*, \ \ i = 1, 2, ..., l \quad (3.46)$$

3.7.2 Méthodes de résolution

3.7.2.1 LINGO

Le problème PDU est résolu par LINGO. La même reformulation du modèle d'optimisation présentée pour le problème PPR dans la partie 3.5.2.1 est utilisée.

3.7.2.2 Algorithme Génétique

L'algorithme génétique est appliqué pour la résolution du problème dual. Le même codage décrit dans la partie 3.5.2.2 est adopté. Les différentes étapes de l'AG implémenté sont les suivantes.

- Générer aléatoirement une population initiale de taille Pop.
- Évaluer les solutions en calculant le coût total du système (équation (3.44)), la disponibilité du système (équation (3.43)) et par conséquent leur fonction objective illustrée par l'équation (3.47).

$$F = A_s - N * max(0, C_s - C_0) \text{ } N \text{ est un grand nombre positif} \tag{3.47}$$

- Sélectionner aléatoirement deux solutions et créer deux nouveaux chromosomes (enfants) par croisement à un seul point avec une probabilité P_c.
- Appliquer une mutation sur les nouveaux chromosomes obtenus avec une probabilité P_m. Dans ce travail, la mutation consiste à augmenter le nombre de composants et le nombre des équipes de réparation d'un sous-système choisi de manière aléatoire par la valeur 1. Cette opération peut augmenter la disponibilité du système qui est la fonction objective. Les variables de décision sont mises à jour si leurs contraintes correspondantes sont dépassées comme le montre l'équation (3.48).

$$\begin{cases} n_i = n_i^{max}, & \text{si } n_i > n_i^{max}. \\ r_i = n_i - k_i + 1, & \text{si } r_i > n_i - k_i + 1 \end{cases} \tag{3.48}$$

- Comparer successivement la valeur de la fonction objective de chaque individu obtenu (enfant) avec la plus mauvaise solution de la population (ayant la valeur minimale de F). La meilleure solution joint la population et l'autre est rejetée.
- Répéter les étapes 2 à 5 $POP/2$ fois.
- Répéter le processus global pour N_c cycles. À la fin de chaque cycle, sauvegarder la meilleure solution ayant la fonction objective maximale. Lorsque le critère d'arrêt, qui est dans ce cas le nombre maximum de cycles N_c, est atteint, la meilleure solution de tous les cycles est sélectionnée comme la solution du problème décrit.

  Pour trouver les meilleurs paramètres de l'AG, un plan expérimental est appliqué où plusieurs valeurs sont testées. Les meilleures solutions sont obtenues avec les valeurs $Pop = 200$, $P_c = 0.8$, $P_m = 0.1$ et $N_c = 500$.

3.7.3 Applications numériques

Cette partie présente les résultats des applications numériques pour la configuration duale en appliquant l'approche proposée aux 4 exemples de différentes tailles. Les données numériques sont illustrées dans le tableau 3.10. Les caractéristiques d'un système constitué de l sous-systèmes sont données par les l premières lignes du tableau 3.10. Nous supposons que les sous-systèmes possèdent le même nombre minimal (k) et maximal (n_{max}) des composants. Le problème d'optimisation est résolu par l'AG et LINGO pour 4 niveaux de dépendance représentés par les différentes valeurs de a : indépendance ($a = 0$), dépendance faible ($a = 0.5$), dépendance linéaire ($a = 1$) et dépendance forte ($a = 1.5$). Le temps maximum d'exécution de LINGO est fixé à $14400s$ (4 heures). Les résultats obtenus sont comparés dans le tableau 3.11.

Une analyse de ces résultats montre que dans l'exemple 1 dans lequel le système était simple, composé de trois sous-systèmes, LINGO et l'AG ont donné les mêmes résultats. LINGO était plus rapide en termes de temps d'exécution. Tandis que, dans les autres exemples, lorsque la taille du système augmente, l'AG était beaucoup plus rapide que LINGO. Concernant l'exemple 2, pour les cas ($a = 0$) et ($a = 1$), l'AG a donné les mêmes solutions fournies par LINGO pendant un temps de calcul plus rapide. Pour $a = 0.5$, la qualité de la solution donnée par l'AG était meilleure que celle de LINGO ($\Delta A_s < 0$). Pour $a = 1.5$, l'AG a donné une solution très proche de celle obtenue par LINGO ($\Delta A_s = 0.16\%$). Pour l'exemple 3, les solutions de l'AG étaient meilleures que les solutions de LINGO (LINGO a été interrompu) avec un temps de calcul plus rapide. Le pourcentage de déviation de la disponibilité du système ($\Delta A_s(\%)$) était négative. Pour l'exemple 4 dans lequel le système a une taille plus grande, aucune solution n'a été donnée par LINGO dans les trois premiers cas. Par conséquent, nous pouvons conclure que l'AG proposé est plus recommandé pour résoudre des problèmes de grande taille, il peut fournir des solutions de très bonne qualité en un temps d'exécution rapide.

En outre, par rapport à l'effet de la dépendance redondante, nous pouvons noter que, la disponibilité du système est améliorée avec l'intensité de la dépendance redondante pour une même valeur de contrainte C_{max}. Prenons l'exemple 4, les résultats de l'AG montrent clairement que lorsque le niveau de la dépendance redondante augmente de l'indépendance, à la dépendance faible, linéaire puis forte, la disponibilité du système augmente respectivement de 0.8770, 0.9189, 0.9601 à 0.9825. Par conséquent, la dépendance redondante peut aider à construire des systèmes plus fiables. Cette remarque est également applicable aux autres exemples testés.

Tableau 3.10 – Données des composants

Sous-système i	λ^i	μ^i	C_i^c	C_i^r
1	0.04	0.1	30	20
2	0.05	0.12	40	25
3	0.06	0.14	20	15
4	0.07	0.20	70	30
5	0.08	0.17	60	25

3.8 Conclusion

Nous avons étudié, dans ce chapitre, le problème monocritère de conception de systèmes réparables séries k/n tout en modélisant la dépendance redondante entre les composants. Ce problème de conception consiste à affecter à chaque sous-système un nombre de composants redondants et des équipes de réparation tout en considérant le concept de la dépendance de défaillance. La disponibilité du système est évaluée pour chaque niveau de dépendance. Nous avons présenté diverses modélisations et méthodes que nous avons appliquées pour résoudre efficacement ce problème.

Dans un premier temps, le modèle d'optimisation est traité en configuration primale pour laquelle l'objectif était de diminuer le coût du système sous contrainte d'une disponibilité exigée. Les méthodes développées pour ce problème sont basées sur les algorithmes génétiques et les colonies des fourmis ainsi leur hybridation avec une recherche locale. Il est à noter que c'était la première application de l'ACO à ce problème de conception des systèmes. Nous avons comparé les résultats des meilleures méthodes avec les solutions optimales identifées par l'énumération complète et avec les solutions fournies par le solveur LINGO. Les résultats obtenus sont particulièrement intéressants. Les algorithmes hybrides ont atteint les solutions optimales pour la majorité des instances testées dans un temps de calcul raisonnable.

Dans un deuxième temps, le modèle d'optimisation est étudié en configuration duale pour laquelle l'objectif était de maximiser la disponibilité du système sous contrainte d'un coût maximum. Nous avons résolu le problème avec l'AG et LINGO. L'AG a donné de bons résultats par comparaison avec LINGO, en particulier, pour les exemples testés de grande taille. Il est à noter qu'il serait intéressant aussi de tester autres méthodes de résolution comme les ACOs et les algorithmes hy-

Tableau 3.11 – Résultats numériques de l'algorithme génétique et LINGO pour le problème dual

Exemple 1 : $l=3$, $k=2$, $n^{max}=5$, $C^{max}=700$									
	LINGO				AG				
a	$(n_1...n_l, r_1...r_l)$	A_s	C_s	$T(s)$	$(n_1...n_l, r_1...r_l)$	A_s	C_s	$T(s)$	$\Delta A_s(\%)$
0	(5 5 5, 4 4 4)	0.9233	690	3.02	(5 5 5, 4 4 4)	0.9233	690	23.513	0
0.5	(5 5 5, 4 4 4)	0.9528	690	2.28	(5 5 5, 4 4 4)	0.9528	690	23.8293	0
1	(5 5 5, 4 4 4)	0.9742	690	2.08	(5 5 5, 4 4 4)	0.9742	690	10.0346	0
1.5	(5 5 5, 4 4 4)	0.9874	690	1.38	(5 5 5, 4 4 4)	0.9874	690	24.4521	0
Exemple 2 : $l=5$, $k=1$, $n^{max}=5$, $C^{max}=1000$									
	LINGO				AG				
a	$(n_1...n_l, r_1...r_l)$	A_s	C_s	$T(s)$	$(n_1...n_l, r_1...r_l)$	A_s	C_s	$T(s)$	$\Delta A_s(\%)$
0	(4 3 4 3 3, 3 2 3 2 3)	0.8889	1000	14361.06	(4 3 4 3 3, 3 2 3 2 3)	0.8889	1000	42.3297	0
0.5	(4 3 4 3 3, 3 2 3 2 3)$^{\times}$	0.9370$^{\times}$	1000$^{\times}$	$\times$	(4 4 4 3 3, 2 2 3 2 2)	0.9375	995	42.1689	-0.053
1	(4 3 4 3 4, 2 2 2 2 2)	0.9722	1000	11010.52	(4 3 4 3 4, 2 2 2 2 2)	0.9722	1000	15.7997	0
1.5	(4 3 4 3 4, 2 2 2 2 2)	0.9885	1000	10921.33	(4 4 4 3 3, 3 2 2 2 2)	0.9869	1000	41.5233	0.16
Exemple 3 : $l=5$, $k=3$, $n^{max}=9$, $C^{max}=2000$									
	LINGO				AG				
a	$(n_1...n_l, r_1...r_l)$	A_s	C_s	$T(s)$	$(n_1...n_l, r_1...r_l)$	A_s	C_s	$T(s)$	$\Delta A_s(\%)$
0	(8 5 6 9 7, 3 3 4 4 3)$^{-}$	0.7347^{-}	2000^{-}	$-$	(9 7 8 6 7, 4 4 5 3 4)	0.8674	1995	54.8982	-18.06
0.5	(9 9 7 6 7, 4 4 3 2 3)$^{\times}$	0.8525$^{\times}$	1970$^{\times}$	$\times$	(8 7 9 6 7, 4 4 4 3 5)	0.9187	1995	59.4653	-7.76
1	(8 8 8 5 6, 3 3 4 3 4)$^{\times}$	0.8793$^{\times}$	1815$^{\times}$	$\times$	(7 7 8 6 8, 4 3 5 4 4)	0.9522	2000	20.2264	-8.29
1.5	(9 8 6 8 7, 3 3 3 1 4)$^{\times}$	0.8860$^{\times}$	2000$^{\times}$	$\times$	(7 7 8 7 7, 4 4 4 3 4)	0.9808	1990	56.3164	-10.69
Exemple 4 : $l=5$, $k=5$, $n^{max}=15$, $C^{max}=3000$									
	LINGO				AG				
a	$(n_1...n_l, r_1...r_l)$	A_s	C_s	$T(s)$	$(n_1...n_l, r_1...r_l)$	A_s	C_s	$T(s)$	$\Delta A_s(\%)$
0	N/A	N/A	N/A	N/A	(10 10 13 10 11, 6 5 5 6 7)	0.8770	2995	80.6525	
0.5	N/A	N/A	N/A	N/A	(11 11 11 10 10, 6 7 7 5 5)	0.9189	2965	82.9506	
1	N/A	N/A	N/A	N/A	(10 11 14 10 12, 5 6 4 4 5)	0.9601	2995	29.5974	
1.5	(10 8 13 5 8, 6 4 4 1 4)$^{\times}$	0.2918$^{\times}$	2120$^{\times}$	$\times$	(11 11 12 10 11, 5 5 7 4 5)	0.9825	2945	81.4637	-236.70

$^{\times}$ signifie qu'aucune solution n'est donnée par LINGO pendant 4 h d'exécution. LINGO est interrompu et la solution est affichée.
N/A signifie qu'aucune solution faisable n'est donnée par LINGO après 4 h d'exécution.
$\Delta A_s(\%)$ Pourcentage de déviation relative entre la disponibilité de la solution donnée par LINGO et celle de l'AG $\frac{A_s(LINGO)-A_s(AG)}{A_s(LINGO)}100\%$

brides.

Les deux problèmes étudiés (primal et dual) ont souligné l'importance de l'introduction de la dépendance redondante. Ce concept qui permet de décrire des aspects dynamiques dans les systèmes paraît intéressant à considérer dès la phase de conception. Les composants dépendants étaient plus favorables pour concevoir des systèmes économiques et fiables. D'autre part, les méthodes proposées peuvent être appliquées à des problèmes de grandes tailles et pour lesquelles les solutions optimales ne peuvent pas être identifiées avec une énumération complète ou avec LINGO. Ces travaux ont fait l'objet de deux articles publiés dans les conférences internationales IFAC AMEST 2016 [31] et IMCET 2016 [148].

La deuxième thématique de cette thèse concerne le problème d'optimisation multiobjective de conception des systèmes séries-k/n à composants dépendants. Le chapitre suivant décrit en détails ce problème ainsi que les différentes approches de résolution adoptées.

Chapitre 4

Optimisation multiobjective des systèmes séries k sur n à composants dépendants

Résumé :

Dans ce chapitre, nous abordons un nouveau aspect de la conception des systèmes séries-k/n à composants dépendants, à savoir le point de vue multiobjectif. Nous commençons par une description et une modélisation mathématique du problème étudié en nous concentrant sur deux objectifs contradictoires sous contrainte du poids donné : maximisation de la disponibilité et minimisation du coût du système. Les composants appartenant au même sous-système sont supposés identiques et peuvent être dépendants. Les variables de décision à déterminer pour chaque sous-système sont le nombre de composants redondants, le nombre des équipes de réparation, le type de composants et leur niveau de dépendance. Nous présentons ensuite de méthodes d'optimisation qui sont basées sur la deuxième version de l'algorithme génétique par dominance de Pareto (NSGA-II), l'algorithme SPEA-II et une méthode exacte. Plusieurs techniques de gestion des contraintes sont considérées et de nombreuses métriques de performance sont appliquées. Les résultats obtenus sont intéressants et ont fait l'objet d'un article publié dans le journal international International Journal of Production Research (IJPR) [163] .

4.1 Introduction

Nous étudions dans ce chapitre le problème multiobjectif de conception de systèmes séries-k/n qui constitue le deuxième axe de cette thèse. En effet, un grand intérêt est accordé au problème d'allocation de la redondance en multiobjectif (*Multi-objective Redundancy Allocation Problem MORAP*) au cours des dernières décennies [164],[165]. C'est un problème d'optimisation combinatoire NP-difficile qui consiste à sélectionner la version et le nombre des composants à mettre en parallèle dans chaque sous-système tout en satisfaisant simultanément plusieurs objectifs sous différentes contraintes [166]. De nombreux travaux de la littérature ont étudié différents types de ces problèmes d'allocation pour diverses structures de système et ont proposé des méthodes de résolution basées essentiellement sur les métaheuristiques. Nous citons dans ce cadre quelques travaux récents.

Lins et Droguett ont présenté en 2011 [123] une approche multiobjective qui couplait l'algorithme génétique (AG) avec une simulation à événements discrets pour résoudre les RAPs dans des systèmes soumis à des réparations imparfaites. L'algorithme proposé a fourni un ensemble de solutions de compromis qui comprenaient non seulement les configurations du système, mais aussi le nombre des équipes de maintenance. La disponibilité du système et le coût du système devaient être maximisés et minimisés respectivement. Mousavi *et al.* ont étudié en 2015 [167] le problème MORAP d'un système série-parallèle multi-états ayant des composants homogènes non réparables dans un environnement flou. Une résolution basée sur les AGs a été développée. Ghorabaee *et al.* ont abordé en 2015 aussi [125] un RAP bi-objectif pour un système constitué de s sous-systèmes k/n indépendants en série avec des composants non identiques sous contrainte d'un poids prédéfini. Quatre méta-heuristiques basées sur NSGA-II ont été présentées pour optimiser le problème considéré. Plus tard, Attar *et al.* ont proposé en 2017 [168] un modèle multiobjectif pour le problème conjoint d'allocation de disponibilité-redondance (JARAP) dans un système série-parallèle considérant la disponibilité et le coût total comme fonctions objectives. La disponibilité du système a été estimée à l'aide d'un modèle de simulation par ordinateur. Les variables de décision étaient le nombre des composants redondants, ainsi que leurs versions et les actions techniques et organisationnelles à attribuer à chaque sous-système. Le problème a été résolu en utilisant les algorithmes NSGA-II et SPEA-II. Dans la même année, Kayedpour *et al.* [126] ont proposé en 2017 un algorithme intégré basé sur les processus markoviens et l'algorithme NSGA-II pour résoudre le problème de la conception de la fiabilité en tenant compte de la disponibilité

instantanée, des composants réparables et des stratégies de configurations possibles.

Il est à noter que toutes ces études multiobjectives ont supposé que les composants sont indépendants. Cependant, les systèmes réelles utilisent souvent des composants ayant différents types de dépendance, en particulier la dépendance de défaillance. Nous pouvons trouver des systèmes à composants dépendants dans les centrales électriques [33], dans les aircrafts [169], dans le domaine de production [170], etc. D'autre part, la plupart des travaux de la littérature qui s'intéressent à la dépendance de défaillance lors de la conception des systèmes ne prennent en considération qu'un seul objectif à optimiser [29], [143], [5]. Pour cela, ce travail tente de surmonter cette limitation en étudiant le problème d'allocation de redondance de systèmes avec la notion de la dépendance redondante du point de vue multiobjectif en raison de son importance pour les concepteurs de systèmes. Notons qu'à notre connaissance, cela représente la première étude de ce type. A cet effet, nous proposons une modélisation mathématique bi-objective du problème et de méthodes de résolution basées sur les algorithmes NSGA-II, SPEA-II et la méthode d'énumération complète. Cette dernière est appliquée pour résoudre les problèmes de petite et moyenne taille. Plusieurs scénarios de tests sont appliqués. Sept mesures sont utilisées pour comparer les performances des différents algorithmes.

4.2 Formulation du problème

Dans cette section, nous présentons la description du système étudié, les notations utilisées et le modèle mathématique proposé.

4.2.1 Description du système

Nous considérons un système série-k/n ayant l sous-systèmes connectés en série sous les hypothèses suivantes :

- Chaque sous-système comprend n_i composants identiques en parallèle et fonctionne si au moins k_i parmi ses composants sont opérationnels. Les composants ont deux états possibles : en marche ou en arrêt [171], [172].
- Chaque sous-système possède différents types de composants disponibles dans le marché. Les composants d'un même sous-système appartiennent au même type [166], [173]. Chaque type est caractérisé par des attributs déjà connus (tels que le taux de défaillance, le taux de réparation, le poids et le coût d'achat) [174].

– Dans chaque sous-système, le taux de défaillance des composants opérationnels augmente avec le nombre des autres composants défaillants [143]. r_i équipes de réparation sont attribuées à chaque sous-système, dont chacune peut fixer, à un temps donné, un composant défaillant avec une réparation parfaite [29].

4.2.2 Notations

l	nombre de sous-systèmes
i	indice de sous-système ($i = 1, ..., l$)
n_i	nombre des composants dans le sous-système i
k_i	nombre minimum des composants requis pour le sous-système i
n_i^{max}	borne supérieure de n_i
n	ensemble de n_i, ($n = n_1, ..., n_l$)
r_i	nombre des équipes de réparation pour le sous-système i
r	ensemble de r_i, ($r = r_1, ..., r_l$)
h_i	nombre de choix disponibles (types) pour les composants du sous-système i
z_i	indice de type des composants utilisé dans le sous-système i, $z_i \in \{1, ..., h_i\}$
z	ensemble de z_i, $z = (z_1, ..., z_l)$
$g(.)$	fonction de la dépendance redondante
a_i	niveau de l'intensité de la dépendance redondante entre les composants du sous-système i
a	ensemble de a_i, $a = (a_1, ..., a_l)$
λ_{i,z_i}	taux de défaillance nominal des composants de type z_i utilisés dans le sous-système i
μ_{i,z_i}	taux de réparation pour les composants de type z_i utilisés dans le sous-système i
c^c_{i,z_i}, w_{i,z_i}	coût d'achat et poids des composants de type z_i utilisés dans le sous-système i
c_i^{rt}	coût de l'équipe de réparation pour le sous-système i
$\theta_i, \alpha_i, \gamma_i$	paramètres constants associés respectivement au coût d'interconnexion, au coût de dépendance et au poids des composants parallèles dans le sous-système i
A_i	disponibilité de sous-système i
A_s	disponibilité du système
W_s	poids du système
W_c	borne supérieure de poids du système

4.2.3 Objectives et analyse des contraintes

Dans cette étude, l'objectif est de maximiser la disponibilité et de minimiser le coût du système simultanément sous la contrainte de poids et en tenant compte de la dépendance de défaillance. Pour reformuler le MORAP, l'analyse suivante est effectuée.

4.2.3.1 Analyse de la disponibilité du système considérant la dépendance redondante

Comme nous nous intéressons à la structure réparable k_i/n_i $(i = 1, ..., l)$, nous adoptons le modèle de la dépendance redondante décrit dans le chapitre précédent pour une telle configuration. Une fonction de la dépendance redondante $g(m)$ a été considérée qui dépend de nombre de composants survivants m dans un sous-système i. Elle est présentée par l'équation (4.1). Le paramètre a_i de cette équation correspond à l'intensité de la dépendance entre les composants redondants dans le sous-système i. Selon la valeur de a_i, les quatre classes de dépendance peuvent être obtenues comme suit : indépendance $(a_i = 0)$, faible $(0 < a_i < 1)$, linéaire $(a_i = 1)$ et forte $(a_i > 1)$.

$$g(m) = \left(\frac{m}{k_i}\right)^{a_i} \tag{4.1}$$

Comme nous l'avons déjà montré que la dépendance redondante affecte la performance du système. Elle est introduite lors de l'évaluation de la disponibilité du système.

Dans ce travail, nous supposons que chaque sous-système i $(i = 1, ..., l)$ a h_i types de composants qui sont fonctionnellement équivalents. Les composants d'un même sous-système sont du même type. Cette situation peut être trouvée en pratique. Les composants peuvent accomplir la même tâche et peuvent avoir différents attributs (coût, poids, etc.) en raison de problèmes liés à la production, à la fabrication ou à la fourniture. L'indice relatif au type des composants z_i utilisé dans le sous-système i (k_i/n_i) est introduit dans le modèle. Par conséquent, la disponibilité A_i du sous-système dépendant i est formulée comme l'indique l'équation (4.2). Nous supposons que les composants du sous-système i sont de même type z_i at ayant le niveau de dépendance a_i.

$$A_i = 1 - \left(1 + \sum_{j=k_i}^{n_i - r_i + 1} \left(\frac{j!}{(k_i - 1)!}\right)^{a_i - 1} \left(\frac{r_i \mu_{i,z_i}}{k_i^{a_i} \lambda_{i,z_i}}\right)^{j - k_i + 1} + \sum_{j=n_i - r_i + 2}^{n_i} \frac{(j!)^{a_i - 1} r_i^{n_i - r_i - k_i + 2} (r_i - 1)!}{((k_i - 1)!)^{a_i - 1} k_i^{a_i (j - k_i + 1)} (n_i - j)!} (\frac{\mu_{i,z_i}}{\lambda_{i,z_i}})^{j - k_i + 1}\right)^{-1} \tag{4.2}$$

La disponibilité du système sera égale au produit de la disponibilité de tous ses sous-systèmes.

4.2.3.2 Coût et poids du système

Le coût du système C_s se compose de trois parties évaluées comme suit :
La première partie représente le coût de conception comme le montre l'équation (4.3).

$$C_1 = \sum_{i=1}^{l} c^c_{i,z_i}(n_i + exp(\theta_i n_i)) \tag{4.3}$$

où c^c_{i,z_i} est le coût d'achat du composant appartenant au type z_i dans le sous-système i. $exp(\theta_i n_i)$ est un coût supplémentaire représentant les interconnexions entre les composants parallèles [175] et θ_i est un paramètre constant pour le sous-système i.
La deuxième partie traite du coût des équipes de réparation comme l'indique l'équation (4.4).

$$C_2 = \sum_{i=1}^{l} c^{rt}_i r_i \tag{4.4}$$

où c^{rt}_i est le coût de l'équipe de réparation correspondante au sous-système i. Nous supposons que ce coût est le même pour tous les types de composants disponibles pour le sous-système i.
La troisième partie est le coût de dépendance. Il est intégré dans le coût du sous-système comme l'indique l'équation (4.5).

$$C_3 = \sum_{i=1}^{l} (exp(\alpha_i n_i a_i) - 1) \tag{4.5}$$

où α_i est un paramètre constant relatif au sous-système i et a_i est le niveau de dépendance entre les composants redondants dans ce sous-système. L'équation (4.5) révèle que le coût du sous-système augmente si l'intensité de la dépendance entre ses composants augmente. Cette relation de proportionnalité directe a été également considérée dans [5].
Le poids du système est obtenu comme le montre l'équation (4.6).

$$W_s = \sum_{i=1}^{l} w_{i,z_i}(n_i + exp(\gamma_i n_i)) \tag{4.6}$$

où γ_i est un paramètre constant pour le sous-système i et $exp(\gamma_i n_i)$ est une valeur supplémentaire qui pénalise les interconnexions entre les composants parallèles [175].

4.2.4 Modèle mathématique

Nous proposons la formulation mathématique présentée par les équations (4.7)-(4.13) du problème d'optimisation étudié.

$$Maximiser\ A_s(a,z,n,r) = \prod_{i=1}^{l} A_i \tag{4.7}$$

$$Minimiser\ C_s(a,z,n,r) = \sum_{i=1}^{l} c_{i,z_i}^{c}(n_i + exp(\theta_i n_i)) + \sum_{i=1}^{l} c_i^{rt} r_i + \sum_{i=1}^{l}(exp(\alpha_i n_i a_i) - 1) \tag{4.8}$$

sous contraintes :

$$W_s = \sum_{i=1}^{l} w_{i,z_i}(n_i + exp(\gamma_i n_i)) \leq W_c \tag{4.9}$$

$$a_i \geq 0, \qquad i = 1,2,...,l \tag{4.10}$$

$$1 \leq z_i \leq h_i, z_i \in \mathbb{N}^*, \qquad i = 1,2,...,l \tag{4.11}$$

$$k_i \leq n_i \leq n_i^{max}, n_i \in \mathbb{N}^*, \qquad i = 1,2,...,l \tag{4.12}$$

$$1 \leq r_i \leq n_i - k_i + 1, r_i \in \mathbb{N}^*,\ \ i = 1,2,...,l \tag{4.13}$$

L'objectif est de déterminer la meilleure allocation de redondance du système sous de multiples contraintes et de satisfaire deux objectifs. Les équations (4.7) et (4.8) représentent les fonctions objectives qui sont respectivement : la minimisation du coût du système C_s et la maximisation de la disponibilité du système A_s. La contrainte (4.9) révèle que le poids du système W_s ne doit pas dépasser une valeur prédéfinie W_c. Une description de A_s, C_s et W_s a été donnée dans la partie (4.2.3.2). Les variables de décision à déterminer sont l'intensité de la dépendance $a = (a_1, ..., a_l)$, le type de composants $z = (z_1, ..., z_l)$, le niveau de redondance $n = (n_1, ..., n_l)$ et le nombre des équipes de réparation $r = (r_1, ..., r_l)$ alloué aux sous-systèmes $1, ..., l$. Elles sont représentées respectivement par les contraintes (4.10)-(4.13). La contrainte (4.10) concerne le paramètre de l'intensité de la dépendance. Les composants redondants dans un sous-système i peuvent être indépendants ($a_i = 0$), faiblement dépendants ($0 < a_i < 1$), linéairement dépendants ($a_i = 1$) ou fortement dépendants

($a_i > 1$). Les trois dernières contraintes limitent les variables z_i, n_i et r_i entre des bornes minimales et maximales prédéfinies pour chaque sous-système i.

4.3 Méthodes de résolution

Comme nous l'avons mentionné précédement, cette étude concerne le problème d'allocation de redondance multiobjectif (MORAP) qui est connu par sa nature NP-difficile. Les métaheuristiques sont souvent utilisées pour résoudre ce type de problèmes. Le travail de Branke et Deb [176] résume les principales méthodes telles que l'algorithme génétique multiobjectif (MOGA), l'algorithme évolutionnaire de Pareto SPEA (*Strength Pareto Evolutionary Algorithm*), SPEA-II, la première version de l'algorithme génétique NSGA (*Non Dominated Sorting Genetic Algorithm*), NSGA-II, etc. Il est à noter que NSGA-II a été largement utilisé ces dernières années pour résoudre différents types de MORAP [115], [123], [126], [177]. Il était performant et a donné de meilleurs résultats [166], [115], [126]. De ce fait, nous avons choisi de l'implémenter dans notre travail. Cependant, seulement quelques travaux avaient appliqué SPEA-II pour résoudre les problèmes MORAP [178], [168]. Il serait intéressant alors d'évaluer son efficacité pour le problème étudié. Ainsi, nous avons appliqué dans ce travail SPEA-II et deux versions modifiées de NSGA-II, appelées respectivement NSGA-II avec pénalité noté par NSGA-II-P et NSGA-II avec une domination de contrainte, noté par NSGA-II-CD (*NSGA-II with constraint domination*). Ces algorithmes multiobjectifs sont évolutionnaires approximatifs et sont basés sur le principe de Pareto. Ils peuvent être adaptés généralement à tout type de problème. Leur procédure commence avec une population initiale générée aléatoirement qui s'améliore au cours des générations successives. L'arrêt des algorithmes est géré par un critère prédéfini. Dans notre cas, il est supposé égal à un nombre fixe de générations N_g.

Afin d'appliquer ces algorithmes à notre problème, nous avons adapté un codage spécifique, des processus de sélection, de croisement et de mutation appropriés. En outre, nous avons parti de la même population initiale et nous avons utilisé la même stratégie de croisement et de mutation pour les trois métaheuristiques pour mieux les comparer. Ces parties communes sont décrites comme suit :

Codage de solutions

Chaque solution possible du problème d'optimisation correspond à un chromosome. Il s'agit d'un ensemble de $4l$ gènes regroupés dans un vecteur où l désigne le nombre de sous-systèmes. Ils

constituent les variables des décision du modèle. La représentation phénotypique est adoptée. Les premiers l gènes représentent le niveau de la dépendance redondante, les deuxièmes l gènes correspondent aux types de composants sélectionnés, les troisièmes l sont le nombre de composants et les derniers l sont le nombre des équipes de réparation à allouer à chaque sous-système i $(i = 1, ..., l)$. Ce codage est représenté par la figure 4.1.

$$\underbrace{[a_1 \cdots a_i \cdots a_l}_{\text{Intensité de la dépendance}} \quad \underbrace{z_1 \cdots z_i \cdots z_l}_{\text{type des components}} \quad \underbrace{n_1 \cdots n_i \cdots n_l}_{\text{nombre des components}} \quad \underbrace{r_1 \cdots r_i \cdots r_l]}_{\text{nombre des équipes de réparation}}$$

FIGURE 4.1 – Codage appliqué

Génération initiale de la population

Une population initiale de taille N_p est générée aléatoirement. Chaque chromosome (solution) de la population se compose de gènes sélectionnés aléatoirement dans leur espace de recherche. La population initiale peut contenir des solutions non faisables (la contrainte de poids n'est pas satisfaite). Elle sera améliorée au cours des générations successives.

Croisement

Pour chaque algorithme proposé, un processus de sélection adéquat est fait pour former un ensemble des parents possibles S_{par}. L'opération suivante est alors de créer deux enfants, désignés par $Enfant\,1$ et $Enfant\,2$.

Deux parents appelés $Parent\,1$ et $Parent\,2$ sont choisis aléatoirement parmi l'ensemble prédéfini S_{par} et un nombre aléatoire est généré. Si ces parents ne sont pas égaux (ayant au moins un gène différent) et le nombre généré est inférieur à la probabilité de croisement p_c, les deux parents vont participer à la procédure de croisement pour former deux enfants. Sinon, $Enfant\,1$ et $Enfant\,2$ seront égaux respectivement à $Parent\,1$ et $Parent\,2$. Pour effectuer le croisement, nous générons un vecteur aléatoire binaire de base B_v ayant l bits (l est le nombre de sous-systèmes connectés en série). Ensuite, nous formons un masque binaire appelé $Mask$ en concaténant quatre fois le vecteur de base V. Le nombre 4 représente le nombre de variables de décision à déterminer pour chaque sous-système i (a_i, z_i, n_i, r_i). Par conséquent, le masque est de même taille que le chromosome qui est égale à $4l$. En se basant sur le masque généré, $Parent\,1$ et $Parent\,2$ seront croisés pour créer deux nouveaux individus. A titre exemple, nous illustrons une application de l'opération de croisement sur un système constitué de 4 sous-systèmes ayant chacun au moins 3 composants comme le montre le tableau 4.1. Le croisement peut se produire en plusieurs points. Les gènes des parents seront échangés dans les positions où le masque a un bit égal à 1. L'avantage d'un tel

croisement est de garder les variables de décision dans leur espace de recherche. Il substitue les gènes correspondant au même sous-système.

Mutation

Une fois les nouveaux chromosomes (enfants) obtenus à la suite du croisement, ils sont soumis au processus de mutation avec une probabilité p_m. Dans ce travail, la mutation consiste à choisir aléatoirement un sous-système, soit x et remplacer ses gènes correspondants a_x, z_x, n_x et r_x par de nouvelles valeurs sélectionnées aléatoirement de leur espace de recherche. Ainsi, une modification dans la structure de chromosome (enfant) est effectuée. Cela accroît la diversité dans la population et évite la convergence vers un optimum local. Cette technique est présentée par le tableau 4.2. Nous supposons dans cet exemple que $x = 3$ pour chaque enfant.

Tableau 4.1 – Exemple de croisement pour un système composé de 4 sous-systèmes ($l = 4$) ayant un nombre minimum de composants requis $k = 3$ et B_v est le vecteur de base binaire aléatoirement généré

B_v :	[1	0	1	0]														
⇓																		
Mask 1 :	[1	0	1	0	⋮	1	0	1	0	⋮	1	0	1	0	⋮	1	0	1	0]
Parent 1 :	[**1**	0.5	**1.5**	0	⋮	**4**	2	**3**	1	⋮	**3**	5	**6**	4	⋮	**1**	2	**4**	2]
Parent 2 :	[**0**	1	**1**	0.5	⋮	**3**	2	**1**	3	⋮	**4**	3	**5**	6	⋮	**2**	1	**1**	3]
⇓																		
Enfant 1 :	[0	0.5	1	0	⋮	3	2	1	1	⋮	4	5	5	4	⋮	2	2	1	2]
Enfant 2 :	[1	1	1.5	0.5	⋮	4	2	3	3	⋮	3	3	6	6	⋮	1	1	4	3]

Tableau 4.2 – Exemple de mutation pour un système composé de 4 sous-systèmes ($l = 4$) ayant un nombre minimum de composants requis $k = 3$ et $x = 3$ représente l'indice du sous-système qui va subir la mutation.

Enfant 1 :	[0	0.5	**1**	0	⋮	3	2	**1**	1	⋮	4	5	**5**	4	⋮	2	2	**1**	2]
Enfant 2 :	[1	1	**1.5**	0.5	⋮	4	2	**3**	3	⋮	3	3	**6**	6	⋮	1	1	**4**	3]
⇓																		
Enfant 1 muté :	[0	0.5	**1.5**	0	⋮	3	2	**2**	1	⋮	4	5	**4**	4	⋮	2	2	**2**	2]
Enfant 2 muté :	[1	1	**0**	0.5	⋮	4	2	**2**	3	⋮	3	3	**5**	6	⋮	1	1	**2**	3]

4.3.1 Algorithme NSGA-II

L'algorithme génétique élitiste de tri non dominé (NSGA-II) est la version améliorée de l'algorithme NSGA. Il a été proposé par Deb [179]. Cet algorithme manipule une population de solutions et utilise un mécanisme explicite de préservation de la diversité. Les individus (solutions) d'une population sont triés sur une base de non-dominance et auxquels sont attribués un rang et une crowding distance. Cette dernière mesure la dispersion des solutions, elle est basée sur le calcul de la distance moyenne aux deux points de part et d'autre de l'individu considéré selon les deux objectifs [179]. Les individus non dominés constituent le front $F1$ ayant un rang 1. Les autres fronts Fi sont ensuite définis récursivement en ignorant les solutions des fronts précédemment trouvés. Nous adoptons l'algorithme NSGA-II tel qu'il est présenté dans l'algorithme 5.

Algorithme 5 Algorithm NSGA-II [179]

Générer une population initiale P_1 de taille N_p
Évaluer les solutions en se basant sur les fonctions objectives
Trier les solutions pour obtenir les fronts non dominés en appliquant la fonction fast-non-dominated-sort function(P_g, N_p) //les individus sont attribués un rang et une crowding distance
pour $g = 1, ..., N_g$ **faire** //g est le numéro de génération courante
  Créer la population des parents S_{par} en utilisant la fonction Parent-selection1 (P_g, N_p)
  Créer la population fille Q_g en se basant sur les opérateurs de croisement et de mutation
  Evaluer les solutions de Q_g
  Fusionner les populations parent P_g et fille Q_g pour former la population $R_g = P_g \cup Q_g$
  Trier les solutions de R_g pour obtenir l'ensemble des fronts $F = (F_1, F_2, ...)$ en utilisant la fonction fast-non-dominated-sort(R_g)
  Initialiser la population de la génération suivante $P_{g+1} \longleftarrow \varnothing$ and $i \longleftarrow 1$
  tantque $|P_{g+1}| + |F_i| < N_p$ **faire**
    Ajouter le front numéro i à P_{g+1}
    $i \longleftarrow i + 1$
  fin tantque
  Trier les individus du front F_i en se basant sur leur crowding distance
  Compléter P_{g+1} avec les premiers $N_p - |P_{g+1}|$ éléments de F_i //les individus restants de P_{g+1} sont sélectionnés de ce front en se basant sur leur crowding distance
fin pour
Les individus de la dernière population (de la dernière génération) sont les solutions non dominées du problème

4.3.1.1 NSGA-II avec pénalité (NSGA-II-P)

L'une des approches efficaces utilisées pour étudier un problème d'optimisation avec contraintes est la technique de pénalisation. Cette dernière permet de guider la recherche vers la région faisable [166]. De ce fait, nous l'appliquons dans ce travail comme suit.

Algorithme 6 Fonction fast-non-dominated-sort(P)

pour $p = 1, ..., size(P)$ **faire** //pour chaque individu p de la population
  $S_p \longleftarrow$ solutions dominées par p //S_p est l'ensemble des solutions qui sont dominées par p
  $n_p \longleftarrow$ nombre de solutions qui dominent p.
  $F_1 \longleftarrow$ solutions ayant n_p=0 //les solutions non dominées sont ajoutées sur le front F_1 ayant le rang 1

  $i \longleftarrow 1$ //initialiser le compteur i de front à 1
  tantque F_i n'est pas vide **faire**
    $Q \longleftarrow \varnothing$ //Q initialisée à vide, est utilisée pour sauvegarder les individus du front $(i+1)$ suivant
    pour chaque individu p dans F_i **faire**
      pour chaque individu q dans S_p **faire**
        $n_q \longleftarrow n_q - 1$
        si $n_q = 0$ **alors** //Aucune des solutions dans les fronts suivants ne dominerait q
          $q_{rank} \longleftarrow i + 1$
          $Q \longleftarrow Q \cup q$ //ajouter l'individu q à l'ensemble Q
        fin si
      fin pour
    fin pour
    $i \longleftarrow i + 1$ //incrémenter le nombre de front
    $F_i \longleftarrow Q$ //Q sera le front suivant
  fin tantque
fin pour
$F \longleftarrow$ l'ensemble trié de tous les fronts non dominés //$F = (F_1, F_2, ..)$
pour $i = 1, ..., size(F)$ **faire** //pour chaque front F_i
  pour $p = 1, ..., size(F_i)$ **faire** //chaque individu p de front F_i
    $d_i \longleftarrow$ crowding distance
  fin pour
fin pour

Algorithme 7 Fonction Parent-selection1(P, N_p)

$S_{par} \longleftarrow \varnothing$ //initialiser l'ensemble des parents
Créer une matrice C_{ind} de N_p lignes et 2 colonnes //matrice des indices des parents
Remplir chaque colonne de C_{ind} avec les entiers obtenus par une permutation aléatoire de 1 à N_p
pour $i = 1, ..., N_p$ **faire**
  Sélectionner aléatoirement une ligne de C_{ind} qui contient les indices des deux parents candidats
    Obtenir les parents correspondants de la population
    Comparer les deux parents selon leur rang
  si les rangs des deux parents sont différents **alors**
    Trouver le parent avec le rang minimum et ajouter-le à S_{par}
  sinon //Dans le cas des rangs égaux, comparer leur crowding distance
    si les crowding distances des deux parents sont différentes **alors**
      Trouver le parent avec la valeur maximale de crowding distance et ajouter-le à S_{par}
    sinon
      Sélectionner aléatoirement l'un des deux parents et ajouter-le à S_{par}
    fin si
  fin si
fin pour

Afin d'évaluer une solution, la fonction Fitness est calculée. Elle se compose de la fonction objective et d'une fonction de pénalité dynamique. Cette dernière représente le degré de non faisabilité (degré de violation de la contrainte) [166]. En outre, étant donné que les fonctions objectives sont conflictuelles (minimiser le coût du système et maximiser la disponibilité du système) et afin d'avoir un problème d'optimisation homogène pour le codage, nous avons proposé les fonctions Fitness suivantes FF_1 et FF_2 présentées respectivement par les équations (4.14) et (4.15).

$$FF_1 = \begin{cases} -A_s & \text{si } W_s \leq W_c, \\ -A_s \times \left(\frac{W_s - W_c}{W_c}\right) & \text{sinon} \end{cases} \tag{4.14}$$

$$FF_2 = \begin{cases} C_s & \text{si } W_s \leq W_c, \\ C_s + (W_s - W_c) & \text{sinon} \end{cases} \tag{4.15}$$

Dans cet algorithme (NSGA-II-P), une solution A domine une autre solution B si en comparant leurs valeurs de fonctions Fitness, A a une performance au moins aussi bonne que celle de B et, au moins pour l'une des fonctions Fitness, la performance de A est meilleure que celle de B.

Mathématiquement, cette notion de dominance peut être formulée par :

$$A \text{ domine } B \text{ si } FF_i(A) \leq FF_i(B) \forall i \in \{1,2\} \text{ et } \exists j \in \{1,2\} \text{ tel que } FF_j(A) < FF_j(B) \quad (4.16)$$

4.3.1.2 NSGA-II avec une domination de contrainte (NSGA-II-CD)

Pour résoudre efficacement un problème multiobjectif avec contraintes, Deb [120] a proposé en 2002 une version modifiée de la définition de dominance. Dans ce concept, une solution A domine une autre solution B si l'une des conditions suivantes est remplie :

- Solution A est faisable, et la solution B n'est pas faisable.
- Les deux solutions A et B sont faisables, et A domine B selon l'équation (4.16).
- Les deux solutions A et B ne sont pas faisables, mais A a un degré de violation de contrainte inférieur à celui de B.

Selon cette définition, nous avons utilisé FF_1 et FF_2 comme l'indique respectivement les équations (4.17) et (4.18). Ainsi les fonction Fitness représentent directement les fonctions objectives. En outre, nous définissons une fonction supplémentaire FF_3 (équation (4.19)) pour évaluer la violation de contrainte.

$$FF_1 = -A_s \quad (4.17)$$

$$FF_2 = C_s \quad (4.18)$$

$$FF_3 = \begin{cases} 0 & \text{si } W_s \leq W_c, \\ W_s - W_c & \text{sinon} \end{cases} \quad (4.19)$$

4.3.2 Algorithme SPEA-II

L'algorithme évolutionnaire SPEA-II (*Strength Pareto Evolutionary Algorithm version* 2) est la deuxième version de SPEA qui a été introduite par [180]. Il utilise une archive de Pareto qui stocke toutes les solutions non dominées trouvées lors des générations. Les différentes étapes du SPEA-II adoptées sont décrites dans l'algorithme 8.

L'algorithme consiste à créer une population P_g de taille N_p et une archive Q_g de taille N qui comporte les solutions non dominées à chaque génération. La dominance d'une solution sur une autre est déterminée en calculant la valeur de sa fonction fitness. La démarche est la suivante :

Calcul de la fonction Fitness

Pour chaque individu i, de l'ensemble $P_g \cup Q_g$ (uniquement sur P_g pour la première génération), nous calculons sa valeur de force, nommée $S(i)$ (équation 4.20) qui représente le nombre de solutions j dominées par cette solution i.

$$S(i) = |j/j \in P_g + Q_g \wedge i \succ j| \tag{4.20}$$

où :

$|.|$ signifie le cardinal d'un ensemble,

$+$ désigne l'union de deux ensembles,

$\succ$ signifie que i domine j selon la relation de dominance de Pareto présentée par l'équation (4.16) avec $FF_1 = -A_s$ et $FF_2 = C_s$.

Ensuite, en se basant sur $S(i)$, une deuxième valeur $R(i)$ est calculée selon l'équation (4.21). Elle représente l'ensemble de solutions j qui dominent i. Si $R(i)$ est égale à 0, cela signifie que la solution correspondante est non dominée.

$$R(i) = \sum_{j \in P_g + Q_g \wedge j \succ i} S(j) \tag{4.21}$$

Afin de déterminer la diversité de la population autour de chaque individu, SPEA2 a adopté la méthode du $k$ième voisin pour différencier entre les individus ayant les mêmes valeurs de $R(i)$.

Pour chaque individu i, les distances (dans l'espace objectif) qui le séparent de tous les individus j de l'archive et de la population sont calculées et stockées dans une liste. Après avoir trié la liste en ordre croissant, nous trouvons la distance σ_i^k donnée par le $k$ième élément, où k est calculé selon l'équation (4.22).

$$k = \sqrt{(N_p + N)} \tag{4.22}$$

où N_p est la taille de la population et N est la taille de l'archive. Ensuite, la densité $D(i)$ correspondante à l'individu i est calculée selon l'équation (4.23).

$$D(i) = \frac{1}{\sigma_i^k + 2} \tag{4.23}$$

Dans le dénominateur, le chiffre 2 est ajouté pour s'assurer que la valeur du dénominateur est plus grande que 0 et que $D(i)$ est inférieur à 1. Parsuite, la fonction Fitness correspondante à la solution i est calculée comme le présente l'équation (4.24), elle est égale à la somme des deux valeurs $R(i)$ et $D(i)$.

$$F(i) = R(i) + D(i) \tag{4.24}$$

Les solutions non dominées sont les solutions ayant une valeur de Fitness inférieure à 1. Ces solutions sont ensuite transférées vers l'archive de la génération suivante.

Sélection des solutions pour l'archive

Lors de cette phase, trois cas peuvent se présenter comme suit :

Le premier cas est lorsque le nombre de solutions non dominées est égal à la taille de l'archive. La phase de sélection peut alors s'effectuer directement et n'exige aucune procédure particulière.

Le deuxième cas est lorsque le nombre de solutions non dominées est inférieur à la taille de l'archive. L'archive sera complétée alors avec les meilleurs individus dominés de la population et de l'archive précédente.

Le troisième cas est lorsque le nombre des individus non dominés dépasse la taille de l'archive. Parsuite, l'opérateur de troncature est appliqué. Les individus qui ont les plus petites distances par rapport à d'autres individus sont éliminés.

Nouvelle population et critère d'arrêt

La nouvelle archive sera utilisée pour sélectionner l'ensemble des parents candidats qui subissent des opérations de croisement et de mutation. La population qui en résulte est la population de la génération suivante.

Le processus se répète jusqu'au critère d'arrêt qui est dans notre cas le nombre de générations.

4.3.3 Réglage des paramètres

Afin de choisir les meilleurs paramètres des algorithmes ci-haut, nous avons effectué plusieurs tests expérimentaux sur l'application numérique présentée dans la sous-section 4.4.2.1. Nous avons testé le problème avec une probabilité de croisement $p_c = \{0.6, 0.9\}$ et une probabilité de mutation $p_m = \{\frac{1}{2D}, \frac{1}{D}\}$, où D représente le nombre total de variables de décision. Il est égal à $4l$ dans ce travail où l représente le nombre de sous-systèmes. Plusieurs fronts de Pareto ont été obtenus

Algorithme 8 Algorithme SPEA-II [180]

Générer la première population P_1 de taille N_p, fixer la taille de l'archive de Pareto à N et le paramètre de cluster à k
Créer une archive de Pareto vide $Q_g \longleftarrow \varnothing$
pour $g = 1, ..., N_g$ **faire** //g est le numéro de la population courante
  Calculer la fonction Fitness de toutes les solutions de P_g et Q_g
  Copier toutes les solutions non dominées de P_g et Q_g à Q_{g+1} //Q_{g+1} est l'archive de la génération suivante
  si $|Q_{g+1}| < N$ **alors**
    Copier les meilleures $N - |Q_{g+1}|$ solutions dominées de l'archive précédente et de la population à la nouvelle archive Q_{g+1}
  sinon si $|Q_{g+1}| > N$ **alors**
    Pour chaque solution i dans Q_{g+1}, calculer la distance σ_i de son $k$ième plus proche voisin dans Q_{g+1}.
    Appliquer une technique de troncature d'archive pour supprimer de façon itérative les solutions avec un ordre décroissant de distance σ de Q_{g+1} jusqu'au $|Q_{g+1}| = N$
  fin si
  Créer la population des parents S_{par} en utilisant la fonction SPEA-II-Parent-selection(Q_{g+1}, N)
  Créer la progéniture P_{g+1} en fonction des opérateurs de croisement et de mutation
fin pour
Les solutions du problème sont les individus non dominés dans l'archive Q_{g+1}

Algorithme 9 Fonction SPEA-II-Parent-selection(P, N_p)

$S_{par} \longleftarrow \varnothing$ //initialiser l'ensemble des parents
Créer une matrice C_{ind} de N_p lignes et 2 colonnes //matrice des indices des parents candidats
Remplir chaque colonne de C_{ind} avec les entiers obtenus par une permutation aléatoire de 1 à N_p
pour $i = 1, ..., N_p$ **faire**
  Sélectionner aléatoirement une ligne de C_{ind} qui contient les indices des deux parents candidats
  Trouver les parents correspondants de la population P
  Comparer les deux parents en se basant sur leur fonction Fitness
  Trouver le parent ayant la valeur minimale de Fitness et ajouter-le à S_{par}
fin pour

pour chaque algorithme sur 10 essais en utilisant une population de 100 individus. Les meilleurs résultats ont été obtenus avec $p_c = 0.9$ et $p_m = \frac{1}{D}$. La taille d'archive de SPEA-II a été fixée à 100 après plusieurs tests. Il est à noter que les valeurs choisies ont été également utilisées pour différents problèmes multiobjectifs dans la littérature [178], [166]. Dans le travail de [178], NSGA-II et SPEA-II ont utilisé une population de 100 individus. L'archive SPEA-II était également de même taille. Dans le travail de [166], les probabilités de croisement et de mutation de NSGA-II ont été fixées respectivement à 0.9 et à $\frac{1}{D}$.

4.3.4 Méthode exacte

En plus des métaheuristiques précitées, la méthode d'énumération complète (FEM) est implémentée dans ce travail. Elle est appliquée uniquement sur les petites et moyennes instances en raison du temps de calcul important que le problème étudié nécessite. Elle consiste à énumérer toutes les solutions réalisables et à trouver parmi elles les solutions non dominées qui forment le front Pareto réel (PF*).

4.4 Exemples illustratives et analyse de performances

4.4.1 Critères de comparaison

Afin de comparer et d'évaluer la performance des différentes méthodes développées (NSGA-II-P, NSGA-II-CD, SPEA-II et FEM), nous utilisons 7 critères qui sont les suivants :

- Le nombre de solutions non dominées du front Pareto, noté par NS.
- Le temps CPU en secondes, c'est le temps d'exécution pris par l'algorithme pour trouver les solutions.
- L'espacement, noté par SP (*Spacing*), qui permet de mesurer l'uniformité de la répartition des solutions dans le plan des objectifs [115].
- La métrique HRS (*Hole Relative Size*) qui permet de mesurer la taille du plus grand trou dans l'espacement des points sur le front Pareto [83].
- Le rapport d'erreur, noté par ERR (*Error Ratio*), qui permet de déterminer le pourcentage de non-convergence de l'algorithme d'optimisation vers la méthode exacte [168].
- La distance de Riise μ, qui permet de comparer deux fronts obtenus avec deux algorithmes d'optimisation multiobjective en se basant sur la distance entre eux [112], [113].

– La mesure de Zitzler qui permet aussi de comparer deux fronts Pareto en calculant le pourcentage de solutions dans un front qui sont dominées par au moins une solution de l'autre front [110], [109].

4.4.2 Exemples et résultats

Dans cette partie, nous traitons deux scénarios de tests. Le premier scénario vise à évaluer la performance de l'approche proposée sur des problèmes de grande taille. Pour ce faire, les trois méta-heuristiques développées (NSGA-II-P, NSGA-II-CD et SPEA-II) sont appliquées sur un problème de grandes instances, noté $P1$ et une comparaison des résultats est établie. Le second scénario vise à évaluer la qualité des solutions obtenues par ces algorithmes par rapport à la méthode exacte FEM. Par conséquent, les méthodes de résolution sont appliquées sur des problèmes de petites et moyennes instances, nommés respectivement $P2$ et $P3$. Les résultats obtenus sont comparés à ceux obtenus par FEM.

En raison de la nature stochastique des méta-heuristiques, 10 exécutions sont effectuées pour chaque algorithme et sont répétées avec un nombre différent de générations $N_g = 250$, 1000 et 5000. Ces valeurs sont sélectionnées après de nombreux tests en fonction de la qualité des solutions finales et du temps de convergence nécessaire. Pour chaque exécution, nous comparons les fronts Pareto obtenus en utilisant les différentes métriques de performance. Nous calculons les moyennes Av. (*Average*) et les écarts types S.D. (*Standard deviation*) de ces métriques au cours de dix exécutions et pour les trois valeurs mentionnées de N_g comme indiqué plus tard dans les tableaux 4.5, 4.6, 4.10 et 4.11. Les figures ci-après illustrent les fronts Pareto obtenus lors de la dixième exécution des algorithmes.

4.4.2.1 Scenario 1

Dans ce scénario, nous considérons une version modifiée de l'exemple décrit dans [174] pour répondre au problème étudié. Un système de 14 sous-systèmes connectés en série est considéré pour le problème $P1$. Chaque sous-système possède trois ou quatre types disponibles des composants. Chaque type est caractérisé par le taux de défaillance nominal, le coût d'achat et le poids, comme l'illustre le tableau 4.3. Nous ajoutons également le taux de réparation de chaque type dans le tableau 4.3. En ce qui concerne les sous-systèmes, les paramètres correspondants sont affichés dans le tableau 4.4. Nous supposons que le nombre de composants utilisés dans le sous-système $i(i = 1, ..., 14)$ varie entre une valeur minimale requise $k_i = 3$ et une valeur maximale allouée

$n_i^{max} = 10$. De plus, l'un des quatres niveaux de dépendance est sélectionné pour les composants redondants d'un même sous-système i : indépendant ($a_i = 0$), faible ($a_i = 0.5$), linéaire ($a_i = 1$) ou fort ($a_i = 1.5$). L'objectif est de maximiser la disponibilité et de minimiser le coût du système sous une contrainte de poids $W_s \leq 600$.

La figure 4.2 présente les fronts de Pareto obtenus par chaque algorithme lors de la dixième exécution du problème $P1$ avec $N_g = 250$. Les résultats des métriques de performance appliquées à ce problème sont illustrés dans les tableaux 4.5 et 4.6. Nous pouvons constater, à partir du tableau 4.5, la supériorité des algorithmes NSGA-II-P et NSGA-II-CD en termes de nombre de solutions (NS) et de temps CPU (CT) vis-à-vis SPEA-II. En termes des métriques SP et HRS, NSGA-II-CD présente les meilleures valeurs (valeurs inférieures) par rapport à NSGA-II-P et SPEA-II. En outre, trois comparaisons sont présentées dans le tableau 4.6 basées sur la distance de Riise (μ) et la mesure de Zitzler (C_1, C_2). Dans un premier lieu, NSGA-II-P est comparé à SPEA-II. Les résultats révèlent que la moyenne de μ est négative, ce qui signifie que le front de Pareto de NSGA-II-P se trouve sous le front de Pareto de SPEA-II. De plus, la moyenne de C_1 est inférieure à celle de C_2. Ce qui montre que les solutions NSGA-II-P dominent les solutions SPEA-II. Par conséquent, NSGA-II-P surpasse SPEA-II en termes de convergence.

Dans un deuxième lieu, NSGA-II-CD est également comparé à SPEA-II. De même, il a montré plus d'avantages que SPEA-II comme Av. $\mu < 0$ et Av. $C_1 <$ Av. C_2.

Finalement, NSGA-II-P et NSGA-II-CD sont comparés ensemble comme ils ont présenté une meilleure performance que SPEA-II. La petite valeur de Av. μ indique que les fronts Pareto de ces deux algorithmes étaient très proches. De plus, la valeur était négative. En termes de la mesure de Zitzler, NSGA-II-P domine NSGA-II-CD.

Parsuite, nous pouvons conclure que pour toutes les valeurs testées de N_g, les résultats étaient en faveur de NSGA-II-P et NSGA-II-CD. Ce dernier peut fournir une meilleure uniformité de la dispersion des solutions, tandis que NSGA-II-P peut apporter une meilleure convergence. La figure 4.3a montre le front Pareto obtenu lors de la dixième exécution de l'algorithme avec $N_g = 5000$.

Nous avons sélectionné plusieurs solutions du front de Pareto de NSGA-II-P illustrées par la figure 4.2a pour être analysées. Table 4.7 présente deux solutions ayant une valeur de disponibilité relativement élevée. Leur configuration du système implique des composants dépendants (faibles, linéaires et forts). L'utilisation de composants dépendants dans les sous-systèmes peut aider à fournir une disponibilité élevée du système.

Or toutes les solutions non dominées sont équivalentes dans l'approche multiobjectif, il est impor-

tant pour le concepteur du système d'évaluer la performance des solutions en termes de retour d'investissement (ROI). Ce dernier représente le gain de la disponibilité par rapport à l'investissement requis dans la conception du système en passant d'une solution, soit i à une autre solution, soit j. Il est calculé selon l'équation (4.25) [123] où A_i et A_j représentent respectivement la disponibilité des deux solutions i et j. C_i et C_j sont leurs coûts respectifs.

$$ROI = \frac{A_i - A_j}{C_i - C_j}, i \neq j \tag{4.25}$$

Pour faire une application, 4 solutions A, B, C et D sont sélectionnées du front de Pareto montré par la figure 4.2a. Le ROI correspondant est évalué dans le tableau 4.8. Nous pouvons constater qu'une augmentation de la disponibilité du système de 0.015 (de la solution A à la solution B) exige un investissement égal à 4.6648. Alors qu'un gain de disponibilité du système d'environ 0.0009 (de la solution C à la solution D) nécessite un investissement coûteux égal à 19.1402. Par conséquent, son ROI correspondant est pire que celui obtenu par le premier cas. Cela révèle qu'un énorme investissement ne donne pas toujours un gain de disponibilité du système. Une étude approfondie doit se faire pour choisir les solutions adéquates du problème étudié.

Tableau 4.3 – Données d'entrée des composants pour les applications numériques

i	Choix 1 (z_i=1)				Choix 2 (z_i = 2)				Choix 3 (z_i=3)				Choix 4 (z_i=4)			
	λ_{i1}	μ_{i1}	c^c_{i1}	w_{i1}	λ_{i2}	μ_{i2}	c^c_{i2}	w_{i2}	λ_{i3}	μ_{i3}	c^c_{i3}	w_{i3}	λ_{i4}	μ_{i4}	c^c_{i4}	w_{i4}
1	0.00532	0.0532	1	3	0.000726	0.0121	1	4	0.00499	0.0907	2	2	0.00818	0.0909	2	5
2	0.008180	0.1169	2	8	0.000619	0.0031	1	10	0.00431	0.0287	1	9	–	–	–	–
3	0.013300	0.1400	2	7	0.011000	0.1375	3	5	0.01240	0.0954	1	6	0.004660	0.0932	4	4
4	0.007410	0.0674	3	5	0.012400	0.1378	4	6	0.00683	0.0854	5	4	–	–	–	–
5	0.00619	0.0696	2	4	0.00431	0.0539	2	3	0.00818	0.1363	3	5	–	–	–	–
6	0.004360	0.0623	3	5	0.005670	0.0872	3	4	0.00268	0.0298	2	5	0.000408	0.0051	2	4
7	0.010500	0.2100	4	7	0.004660	0.1165	4	8	0.00394	0.1313	5	9	–	–	–	–
8	0.015000	0.0600	3	4	0.001050	0.0064	5	7	0.01050	0.1062	6	6	–	–	–	–
9	0.002680	0.0063	2	8	0.000101	0.0003	3	9	0.000408	0.0017	4	7	0.000943	0.0038	3	8
10	0.014100	0.1141	4	6	0.006830	0.0615	4	5	0.001050	0.0121	5	6	–	–	–	–
11	0.003940	0.0208	3	5	0.003550	0.0201	4	6	0.003140	0.0210	5	6	–	–	–	–
12	0.002360	0.1115	2	4	0.007690	0.0436	3	5	0.013300	0.0890	4	6	0.011000	0.0756	5	7
13	0.002150	0.0174	2	5	0.004360	0.0186	3	5	0.006650	0.0377	2	6	–	–	–	–
14	0.011000	0.0501	4	6	0.008340	0.0751	4	7	0.003550	0.0359	5	6	0.004360	0.0501	6	9

Tableau 4.4 – Données d'entrée des sous-systèmes

Sous-système i	θ_i	γ_i	α_i	c_i^{rt}
1	0.25	0.03	0.1	2
2	0.25	0.03	0.1	2
3	0.25	0.03	0.1	3
4	0.25	0.03	0.1	4
5	0.25	0.03	0.1	2
6	0.25	0.03	0.1	3
7	0.25	0.03	0.1	4
8	0.25	0.03	0.1	4
9	0.25	0.03	0.1	2
10	0.25	0.03	0.1	4
11	0.25	0.03	0.1	3
12	0.25	0.03	0.1	2
13	0.25	0.03	0.1	2
14	0.25	0.03	0.1	3

Tableau 4.5 – Résultats expérimentaux pour le problème $P1$ en termes de NS, SP, HRS et CT.

		NSGA-II-P				NSGA-II-CD				SPEA-II			
		NS	SP	HRS	CT(s)	NS	SP	HRS	CT(s)	NS	SP	HRS	CT(s)
$N_g = 250$	Av.	100	5.14	12.68	43.5	100	1.47	4.44	29.54	61.6	1.76	5.58	269.71
	S.D.	0	6.91	14.86	1.06	0	0.68	3.12	0.61	15.7	0.49	2.6	18.09
$N_g = 1000$	Av.	100	1.89	5.78	164.18	100	1.72	4.64	108.48	86.6	2.5	15.12	1040.02
	S.D.	0	0.93	4.41	4.55	0	0.38	2.72	3.82	13.34	1.6	12.85	16.02
$N_g = 5000$	Av.	100	3.1	7.2	837.93	100	2.11	4.05	538.47	86.7	1.92	13.37	5483.44
	S.D.	0	1.78	7.53	27.23	0	0.55	1.82	12.58	6.97	1.34	7.57	531.96

Tableau 4.6 – Comparaison des fronts Pareto obtenus pour le problème $P1$ en termes de μ, C_1 et C_2.

		NSGA-II-P/SPEA-II			NSGA-II-CD/SPEA-II			NSGA-II-P/NSGA-II-CD		
		μ	C_1	C_2	μ	C_1	C_2	μ	C_1	C_2
$N_g = 250$	Av.	-0.229	10.5	77.47	-0.1684	14.7	67.06	-0.0164	28.9	55
	S.D.	0.258	8.04	14.65	0.119	12.41	20.93	1.97E-02	17.19	17.75
$N_g = 1000$	Av.	-5.92E-02	23	46.18	-3.58E-02	26.5	50.05	-8.60E-03	30.4	52
	S.D.	7.20E-02	12.04	30.26	6.78E-02	15.43	30.14	5.89E-03	13.94	13.22
$N_g = 5000$	Av.	-2.33E-02	12.5	39.8	-0.0205	16.3	23.28	-3.26E-03	29.8	45
	S.D.	6.63E-02	8.47	35.48	6.39E-02	7.87	31.6	6.31E-03	14.05	15.65

4.4.2.2 Scenario 2

Pour ce scénario, nous considérons deux problèmes $P2$ et $P3$ avec respectivement deux ($l = 2$) et trois ($l = 3$) sous-systèmes connectés en série. Le nombre de composants dans tout sous-système

Tableau 4.7 – Deux solutions sélectionnées du front Pareto de NSGA-II-P lors de la dixième exécution de problème $P1$ avec $N_g = 250$.

	i	Intensité de dépendance (a_i)	z_i	n_i	r_i	A_s	C_s	W_s
solution C	1	Linéaire	3	5	2	0.9842	594.9791	588.54
	2	Linéaire	3	7	3			
	3	Linéaire	3	7	2			
	4	Linéaire	1	6	3			
	5	Linéaire	2	6	3			
	6	Forte	4	5	2			
	7	Linéaire	2	5	1			
	8	Forte	1	8	3			
	9	Linéaire	1	8	4			
	10	Linéaire	2	7	3			
	11	Faible	1	6	3			
	12	Faible	2	7	5			
	13	Forte	1	6	2			
	14	Forte	3	8	4			
solution D	1	Forte	3	8	3	0.9851	614.1193	594.7589
	2	Linéaire	3	7	3			
	3	Linéaire	3	7	2			
	4	Linéaire	1	6	3			
	5	Linéaire	2	6	3			
	6	Forte	4	5	2			
	7	Linéaire	2	5	1			
	8	Forte	1	8	3			
	9	Linéaire	1	8	4			
	10	Linéaire	2	7	3			
	11	Faible	1	6	3			
	12	Faible	2	7	5			
	13	Forte	1	6	2			
	14	Forte	3	8	4			

Tableau 4.8 – Retour de l'investissement de certaines solutions sélectionnées du front Pareto de NSGA-II-P de la dixième exécution de problème $P1$ avec $N_g = 250$.

Solution	Disponibilité	Coût	ROI
A	0.8083	398.2974	3.2155×10^{-3}
B	0.8233	402.9622	
C	0.9842	594.9791	4.7021×10^{-5}
D	0.9851	614.1193	

i ($i = 1, ..., l$) est supposé entre 3 et 5. De plus, le niveau de dépendance a_i est sélectionné de l'ensemble $\{0, 0.5, 1, 1.5\}$ comme dans le cas de $P1$. La contrainte de poids est égale à 70. Pour les deux problèmes, les données d'entrée concernant les composants et les sous-systèmes sont prises respectivement à partir des l premières lignes des tableaux 4.3 et 4.4.
Pour le problème $P2$, l'espace de recherche se compose de 6912 solutions possibles, dont 3856 sont réalisables. Tandis que, pour le problème $P3$, il y ait 663552 solutions possibles dans l'espace de recherche, dont 9344 sont réalisables. FEM a été appliquée sur les deux problèmes et les résultats correspondants sont illustrés dans le tableau 4.9. D'après ce tableau, nous pouvons voir que le front Pareto réel de FEM contient 88 solutions non dominées obtenues pendant un temps de calcul rapide (109.42 secondes) pour le problème $P2$. Alors que dans le cas de $P3$, FEM a exigé un temps de calcul assez important ($CT = 31.85$ heures) pour trouver le front Pareto optimal de 87 solutions.

Le tableau 4.10 illustre les résultats obtenus en termes de cinq métriques des trois algorithmes proposés pour la résolution des problèmes $P2$ et $P3$ avec différentes valeurs de N_g. Considérons le cas $N_g = 250$ de $P2$, nous pouvons constater que NSGA-II-P et NSGA-II-CD présentent une supériorité significative par rapport à SPEA-II, et cela couvre toutes les mesures effectuées. NSGA-II-P a fourni le plus grand nombre de solutions non dominées (Av. $NS = 82.7$) avec le minimum taux d'erreur (Av. $ERR = 0.1194$). NSGA-II-CD avait des résultats légèrement meilleurs que ceux de NSGA-II-P en termes de SP, HRS et CT. Dans les cas où le nombre maximal de générations N_g est augmenté à 1000 et 5000, la qualité des fronts Pareto des trois algorithmes est significativement améliorée. Le rapport d'erreur moyen par rapport aux solutions exactes est clairement diminué. Par exemple, pour $N_g = 1000$, il était égal à 0.008, 0.0159 et 0.1625 pour NSGA-II-P, NSGA-II-CD et SPEA-II respectivement. Cependant, le temps de calcul CT est évidemment augmenté. Il est à noter que des résultats intéressants sont obtenus pour $N_g = 5000$. Les deux algorithmes NSGA-II-P et NSGA-II-CD ont convergé vers le front optimal PF*. Ils ont trouvé toutes les solutions exactes non dominées ($NS = 88$ et $ERR = 0$). De même, nous aboutissons à la même conclusion en analysant les résultats de $P3$. Le tableau 4.11 compare les fronts Pareto obtenus des trois algorithmes proposés pour $P2$ et $P3$ par rapport au front optimal de FEM en termes de métriques μ, C_1 et C_2. Les résultats montrent que NSGA-II-P a la meilleure convergence pour les cas $N_g = 250$ et $N_g = 1000$ des deux problèmes. Le front de Pareto correspondant est le plus proche du PF* de FEM et domine les fronts Pareto des autres algorithmes. Par exemple, comme déjà vu dans le problème $P2$ avec $Ng = 1000$, le PF* optimal est sous les fronts Pareto approximatifs (Av. $\mu < 0$). NSGA-II-P possède le front le plus proche du front optimal (Av. $\mu = -6.5E - 06$). NSGA-II-P,

NSGA-II-CD et SPEA-II ont en moyenne respectivement 0.8%, 1.59% et 16.5% de solutions qui sont dominées par au moins une solution du PF*. En outre, NSGA-II-P domine NSGA-II-CD (Av. $\mu < 0$ et Av. $C_1 <$ Av. C_2). Pour $N_g = 5000$, NSGA-II-P et NSGA-II-CD ont atteint le PF* optimal pour les dix exécutions (Av. $\mu = 0$, Av. $C_1 = 0$ et Av. $C_2 = 0$). Ce qui prouve les résultats obtenus précédemment dans le tableau 4.10. Alors que, SPEA-II peut avoir besoin de plus de générations et d'une plus grande taille d'archive pour obtenir les solutions optimales. Ainsi, le temps de calcul exigé sera plus important. Les figures 4.3b et 4.3c comparent les fronts obtenus à la dixième exécution contre le front optimal de FEM pour les problèmes $P2$ et $P3$.

Tableau 4.9 – Résultats de la méthode exacte FEM

	NS	CT (s)
Problème $P2$	88	109.42
Problème $P3$	87	114660

Tableau 4.10 – Résultats expérimentaux obtenus par les algorithmes proposés pour les problèmes $P2$ et $P3$.

		NSGA-II-P					NSGA-II-CD					SPEA-II				
		NS	ERR	SP	HRS	CT(s)	NS	ERR	SP	HRS	CT(s)	NS	ERR	SP	HRS	CT(s)
Problème $P2$																
$N_g = 250$	Av.	82.7	0.1194	0.31	5.87	12.4	80.2	0.1602	0.30	5.64	4.21	50	0.275	0.63	7.65	222.70
	S.D.	2.94	0.0463	0.025	0.54	0.58	1.75	0.0745	0.01	0.25	0.41	6.79	0.08	0.35	4.28	1.55
$N_g = 1000$	Av.	87.3	0.008	0.31	5.67	49.19	87.2	0.0159	0.31	5.68	18.79	49.5	0.1625	0.58	8.25	853.34
	S.D.	0.82	0.0153	0.016	0.29	1.18	0.63	0.0341	0.015	0.35	0.3	8.99	0.064	0.36	4.78	1.32
$N_g = 5000$	Av.	88	0	0.3	5.56	253.43	88	0	0.3	5.56	95.17	51.1	0.0951	0.5	8.6	4110.21
	S.D.	0	0	0	0	5.7	0	0	0	0	1.58	5.38	0.0427	0.421	5.028	75.92
Problème $P3$																
$N_g = 250$	Av.	74.1	0.1926	0.24	8.42	13.93	73.9	0.358	0.22	5.19	4.79	30.8	0.5319	0.98	5.74	222.69
	S.D.	3.07	0.0453	0.06	8.83	0.58	2.33	0.1454	0.05	1.80	0.51	9.23	0.0971	0.62	2.24	0.59
$N_g = 1000$	Av.	84.7	0.0371	0.20	4.64	57.98	84.7	0.0556	0.20	4.86	18.87	46.2	0.4122	0.69	9.01	865.06
	S.D.	1.76	0.0373	0.02	1.46	2.62	1.70	0.0347	0.03	2.03	0.72	10.5	0.1757	0.29	3.26	4.90
$N_g = 5000$	Av.	87	0	0.19	3.66	285.74	87	0	0.19	3.66	96.67	44.6	0.3416	0.51	7.78	4313.26
	S.D.	0	0	0	0	8.88	0	0	0	0	0.26	10.47	0.1774	0.21	3.56	218.03

4.5 Conclusion

Nous avons étudié dans ce chapitre le problème multiobjectif de conception des systèmes à composants dépendants en considérant la dépendance redondante. Malgré que l'optimisation multiobjective d'allocation de redondance a été largement discutée dans la littérature, aucun travail n'avait appliqué l'optimisation multiobjective au type spécial de problème de conception étudié dans cette thèse. Ainsi, nous avons proposé une nouvelle méthodologie d'optimisation multiobjective pour trouver la meilleure allocation de redondance de plusieurs sous-systèmes réparables k/n

Tableau 4.11 – Comparaison des fronts Pareto de FEM et des algorithmes proposés pour les problèmes $P2$ et $P3$

		FEM/NSGA-II-P			FEM/NSGA-II-CD			FEM/SPEA-II			NSGA-II-P/NSGA-II-CD		
		μ	C_1	C_2	μ	C_1	C_2	μ	C_1	C_2	μ	C_1	C_2
Problème $P2$													
$N_g = 250$	Av.	-2.47E-04	0	11.94	-3.47E-04	0	16.01	-1.10E-03	0	27.5	-4.51E-04	3.67	11.83
	S.D.	2.36E-04	0	4.64	2.21E-04	0	7.45	8.84E-04	0	8.59	5.72E-04	2.42	7.66
$N_g = 1000$	Av.	-6.50E-06	0	0.8	-1.01E-05	0	1.59	-1.42E-03	0	16.25	-2.27E-06	0.8	1.59
	S.D.	1.08E-05	0	1.53	1.58E-05	0	3.4	1.60E-03	0	6.4	2.06E-05	1.53	3.4
$N_g = 5000$	Av.	0	0	0	0	0	0	-8.44E-04	0	9.51	0	0	0
	S.D.	0	0	0	0	0	0	1.11E-03	0	4.27	0	0	0
Problème $P3$													
$N_g = 250$	Av.	-2.21E-03	0	19.26	-2.57E-03	0	35.8	-1.82E-02	0	53.19	-9.47E-04	5.95	23.99
	S.D.	1.79E-03	0	4.53	2.04E-03	0	14.54	6.70E-03	0	9.71	2.15E-03	3.02	16.52
$N_g = 1000$	Av.	-1.17E-04	0	3.71	-2.05E-04	0	5.56	-1.27E-02	0	41.22	-2.28E-05	2.75	4.49
	S.D.	1.15E-04	0	3.73	1.66E-04	0	3.47	7.65E-03	0	17.57	2.28E-04	3.21	3.45
$N_g = 5000$	Av.	0	0	0	0	0	0	-3.76E-02	0	34.16	0	0	0
	S.D.	0	0	0	0	0	0	6.83E-02	0	17.73	0	0	0

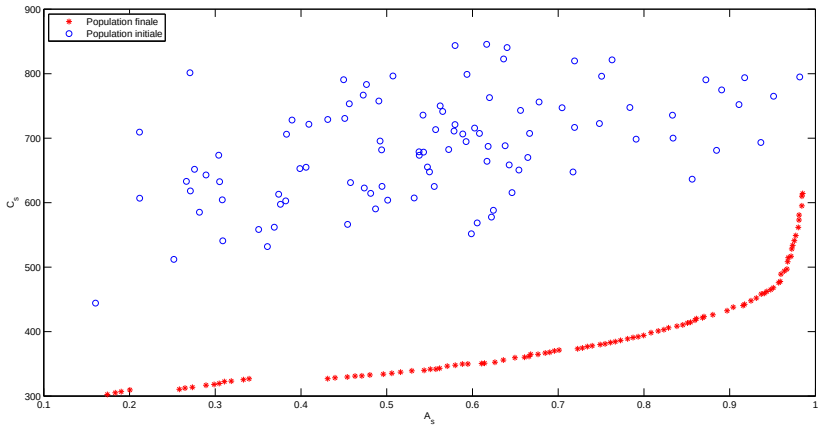
dépendants connectés en série. La dépendance redondante entre les composants d'un sous-système était considérée. Deux objectifs étaient pris en compte sous contrainte de poids : maximisation de la disponibilité du système et minimisation du coût total du système. Ce dernier comportait de nombreux facteurs tels que le coût de conception, le coût de réparation et le coût de dépendance. En plus du nombre de redondance et des équipes de réparation, le choix des composants et le niveau de dépendance ont été déterminés.

Étant donné que le problème décrit ci-haut est NP-difficile, deux variantes de l'algorithme NSGA-II appelés respectivement NSGA-II avec pénalité et NSGA-II avec une domination de contrainte ont été proposées en plus de SPEA-II. Dans le but de bien vérifier l'efficacité et de mieux comparer ces méthodes de résolution, des scénarios de tests ont été appliqués. Dans ces scénarios, nous avons procédé aux changements de la taille du système et une série de tests a été réalisée. Pour les problèmes de petite et moyenne taille, nous avons appliqué une méthode supplémentaire, la méthode d'énumération complète. Les trois algorithmes développés (NSGA-II-P, NSGA-II-CD et SPEA-II) ont été répétés 10 fois pour différents nombres de générations $N_g = 250$, 1000 et 5000. Les fronts Pareto ont été obtenus et comparés en fonction de plusieurs métriques de performance. Les résultats ont montré que NSGA-II-P et NSGA-II-CD peuvent fournir une meilleure qualité de solutions pendant un temps de calcul rapide par rapport à SPEA-II. Le premier algorithme a pu trouver des solutions exactes ou proches de celles du front Pareto réel. Il a surpassé les autres algorithmes en termes de convergence et de dominance tandis que le second avait une meilleure répartition des solutions non dominées. Les deux ont réussi à obtenir toutes les solutions exactes

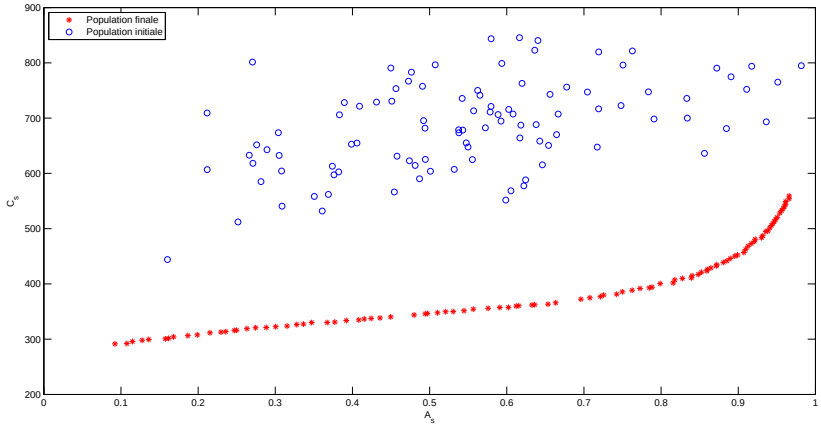
non dominées pour les problèmes de petite et moyenne taille lorsque le nombre de générations N_g a été augmenté à 5000. Cette valeur peut être considérée comme la plus appropriée pour toute autre utilisation de tels problèmes. En outre, nous avons constaté que la disponibilité élevée du système était généralement assurée par des solutions non dominées qui intègrent une dépendance linéaire et/ou forte. Les composants dépendants peuvent aider à concevoir des systèmes plus fiables.

L'approche proposée peut être un outil d'aide à la décision pour le concepteur de système comme elle fournit un ensemble de solutions non dominées au lieu d'une seule solution. Elle permet de trouver la meilleure configuration du système répondant à de multiples objectifs et contraintes.

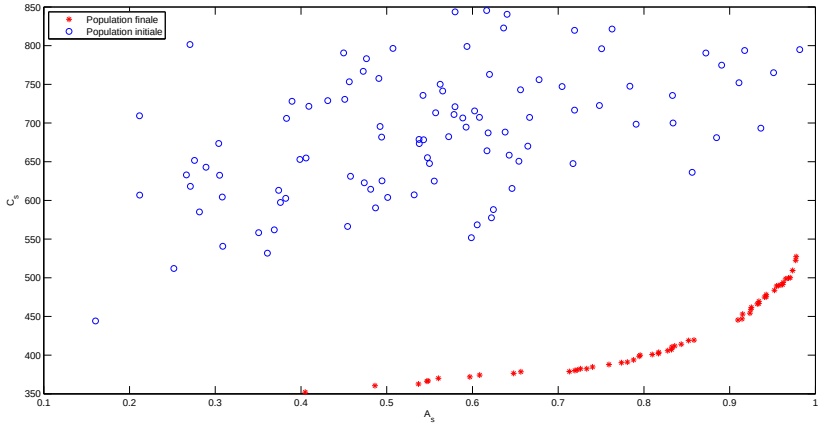
Ce travail a fait l'objet d'une publication dans le journal international International Journal of Production Research (IJPR) [163].



(a) Meilleures solutions obtenues par NSGA-II-P

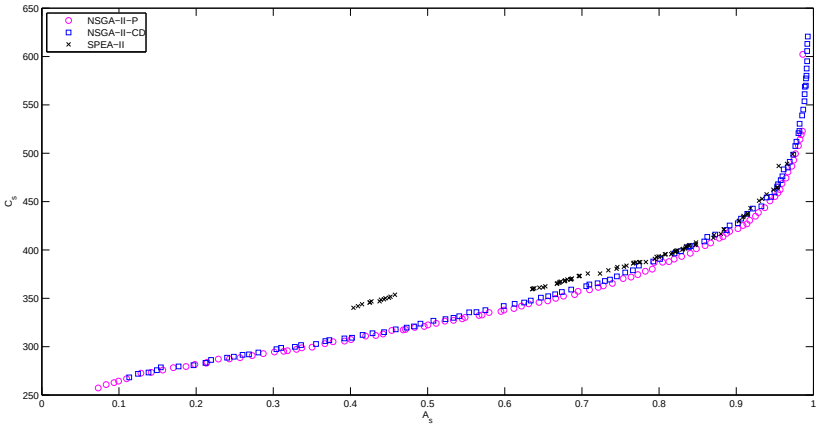


(b) Meilleures solutions obtenues par NSGA-II-CD

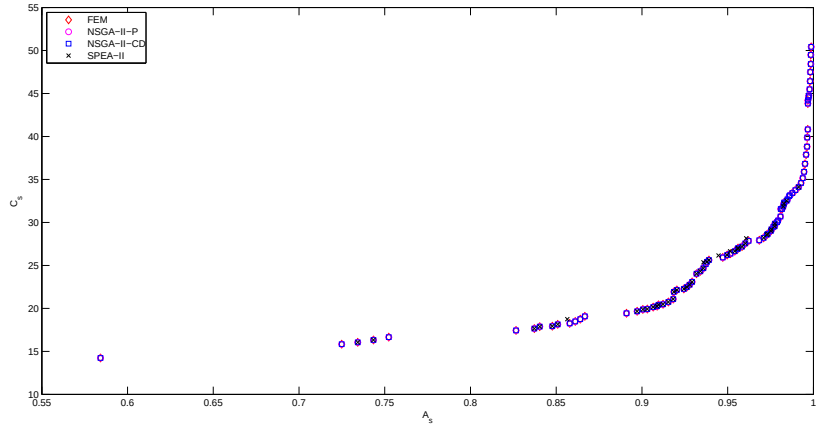


(c) Meilleures solutions obtenues par SPEA-II

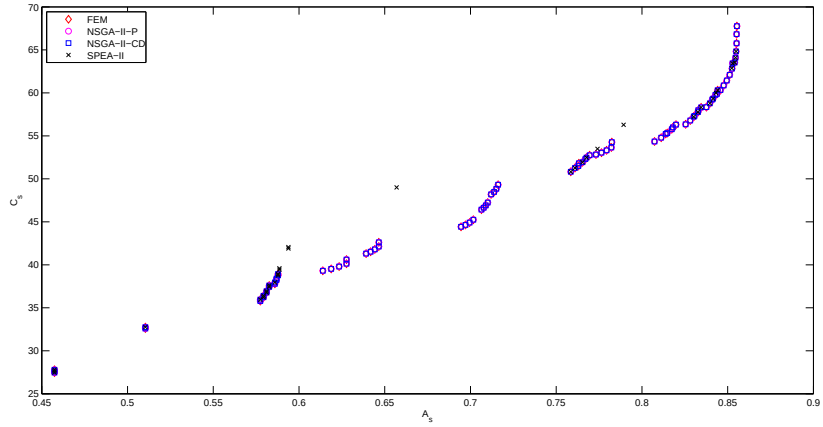
FIGURE 4.2 – Fronts Pareto obtenus par les trois algorithmes pour les problèmes $P1$ avec $N_g = 250$



(a) Problème P1



(b) Problème P2



(c) Problème P3

FIGURE 4.3 – Comparaison des fronts Pareto obtenus par les différents algorithmes pour les problèmes $P1$, $P2$ et $P3$ ayant $N_g = 5000$

Conclusion générale et perspectives

Plusieurs alternatives fonctionnelles et technologiques permettant de répondre au cahier des charges doivent être étudiées lors de la conception d'un système industriel. Chaque alternative est caractérisée par sa fiabilité, sa maintenabilité, sa productivité, le taux de rejet de substances polluantes, la durée de vie des composants et leur recyclage, l'ergonomie, et l'effet sur les opérateurs, etc. Une étude poussée doit donc être menée de manière à choisir les composants à utiliser pour respecter un ensemble de contraintes (coût d'investissement, fiabilité du processus, productivité, pollution du système, dépense d'énergie, gestion de la maintenance, etc.) et ceci pour atteindre une conception optimale du système en question. De très nombreux travaux ont été menés dans cette direction. La diversité de structures et des options existantes pour l'amélioration de performances des systèmes, ont aboutit à différentes classifications et modèles d'optimisation. Les systèmes sont généralement constitués de plusieurs sous-systèmes multi-composants. Dans la plupart des modèles traités dans la littératute, les composants sont supposés stochastiquement indépendants, ce qui est rarement valide dans la réalité industrielle. En effet, la défaillance d'un composant affecte généralement la performance des autres composants survivants dans le système. Dans ce contexte, nous nous sommes intéressés dans cette thèse au développement de méthodes et approches permettant d'assurer un niveau requis de maîtrise de processus industriels dans leurs phases de conception tout en considérant la dépendance stochastique. Nous avons considéré des systèmes réparables multi-composants opérant selon une structure parallèle, k/n et série k/n. Nous avons proposé un modèle pour la dépendance redondante pour la configuration k/n, qui est ensuite utilisé pour la configuration série k/n. Ce concept a permis de quantifier la dépendance de défaillance entre les composants en se basant sur la notion de la répartition de charges. Il a été intégré dans l'évaluation de la performance fiabiliste du système et a été considéré dans les modèles d'optimisation développés. Ainsi, l'originalité de notre travail était basée donc sur le fait de coupler différents types d'objectifs et de contraintes liés à différents aspects : économie,

performances, fiabilité, développement durable, etc. tout en tenant en compte de la dépendance redondante.

Ce manuscrit a été organisé en quatre chapitres. Afin de situer précisément le sujet étudié et nos contributions, nous avons dédié le premier chapitre à la présentation du contexte général de l'étude et à la présentation d'une synthèse bibliographique concernant les différents thèmes abordés. Le problème monocritère de conception de systèmes a été traité dans les chapitres 2 et 3 pour de nombreuses configurations. Le problème multicritère a été abordé dans le dernier chapitre.

Dans le premier chapitre, nous avons introduit le sujet de la thèse. Nous avons rappelé les différents concepts de base de la sûreté de fonctionnement et précisé le type de dépendances considérées dans notre travail. Ensuite, nous avons présenté les principaux modèles d'allocation, les techniques d'évaluation et d'amélioration de la performance des systèmes existants dans la littérature en plus des approches d'optimisation. Ce chapitre nous a montré le grand intérêt de la dépendance stochastique qui peut mettre en jeu les défaillances dépendantes des composants. De plus, l'importance des modèles markoviens a été souligné pour représenter l'évolution du système ayant des processus stochastiques. Quant aux méthodes de résolution, nous avons révélé l'intérèt des métaheuriqtiues pour la résolution des problèmes d'allocation de la fiabilité/disponibilité. Ces derniers sont connus par leur nature NP-difficile. Ainsi, nous avons positionné dans ce chapitre la problématique étudiée et nos contributions par rapport à d'autres travaux de la littérature.

Dans le deuxième chapitre, un état de l'art sur les modèles de dépendances de défaillance a été réalisé en focalisant ensuite sur la dépendance redondante entre les composants du système. Cette dernière fait partie des modèles basés sur des mécanismes déterministes relatifs à des processus stochastiques. Récemment, un grand intérêt leur est accordé du fait qu'ils permettent de modéliser le comportement dynamique du système. Nous avons étudié, dans ce chapitre, la conception des systèmes de type parallèle et k/n à composants dépendants. Notre première contribution consistait à proposer des nouvelles approches d'optimisation basées sur le solveur LINGO et l'algorithme génétique pour trouver le nombre des composants redondants à utiliser dans le système et leurs caractéristiques stochastiques en termes de taux de défaillance et taux de réparation. Dans le cas de systèmes parallèles, nous avons adopté le modèle de la dépendance redondante proposé par Yu *et al.* [5]. Dans le cas des systèmes k/n, nous avons proposé un modèle plus général pour quantifier la dépendance redondante entre les composants du système. Cela a été fait à l'aide d'une fonction de dépendance. De plus, la disponibilité stationnaire du système a été évaluée en utilisant les chaines de Markov tout en introduisant cette notion de dépendance. Nous nous sommes intéréssés

à deux configurations d'optimisation : primale et duale. La première vise à minimiser le coût du système sous contrainte d'une disponibilité exigée. Alors que la deuxième avait pour objectif de maximiser la disponibilité sous contrainte d'un coût maximal prédéfini. Nous avons analysé l'effet de la dépendance redondante, ainsi que les paramètres stochastiques des composants sur la performance du système. Les résultats que nous avons obtenus, constituent, pour nous, une contribution fondamentale, permettant par la suite d'étudier de systèmes et de problématiques plus complexes.

Dans le troisième chapitre, nous avons considéré des systèmes plus généraux : les systèmes réparables séries-k/n à composants dépendants. Nous avons abordé le problème d'allocation de redondance avec dépendance redondante sous l'hypothèse que les composants et les équipes de réparation assignés à chaque sous-système étaient limités. Nous avons trouvé les expressions de la disponibilité stationnaire de ce système, relatives aux différents niveaux de dépendance. Pour le problème de la minimisation du coût sous contrainte de disponibilité, nous avons développé des méthodes de résolution basées sur l'algorithme génétique, les algorithmes d'optimisation par colonies de fourmis (ACO), et leur hybridation avec une recherche locale. Il est à noter que c'était la première application des algorithmes d'optimisation par colonies de fourmis pour ce type de problèmes. Un plan expérimental a été implémenté pour régler les paramètres de différentes métaheuristiques appliquées. De plus, une résolution par le solveur LINGO et une méthode exacte (méthode d'énumération complète) a été également faite. Les solutions obtenues par les algorithmes hybrides ont convergé vers l'optimum pour la majorité des instances testées dans un temps de calcul raisonnable. Pour le cas dual, la résolution était faite en utilisant les algorithmes génétiques et LINGO. Les résultats obtenus pour les deux problèmes étaient intéressants. Ils ont souligné l'importance de considérer la dépendance redondante dès la phase de conception. Cette dépendance a permis d'améliorer la performance du système sans avoir besoin d'ajouter de la redondance. Les systèmes à composants dépendants ont exigé un nombre des composants et des équipes de réparation inférieur à celui requis par les systèmes à composants indépendants.

Dans le dernier chapitre, l'optimisation multiobjective a été appliquée à un nouveau problème de conception des systèmes séries-k/n à composants dépendants. C'était une contribution significative par rapport aux travaux de la littérature qui présentent un manque d'étude de ce problème de conception multiobjective. Nous avons pris en compte le cas où les composants de chaque sous-système sont à choisir parmi un ensemble de types disponibles dans le marché. Les composants appartenant au même sous-système ont été supposés identiques et peuvent être dépendants. Le

problème consistait à trouver pour chaque sous-système le nombre de composants redondants et des équipes de réparation, le type de composants et leur niveau de dépendance. L'objectif était d'optimiser simultanément la disponibilité et le coût du système sous une contrainte d'un poids fixé. Nous avons implémenté trois algorithmes multiobjectifs. Deux parmi eux, étaient basés sur la deuxième version de l'algorithme génétique par dominance de Pareto (NSGA-II) ayant différentes techniques de gestion de contrainte. Le troisième était l'algorithme SPEA-II qui est rarement appliqué aux problèmes d'allocation de redondance. De plus, la méthode d'énumération complète a été appliquée aux problèmes de petites et moyennes instances. Un protocole de tests a été élaboré pour déterminer l'avantage d'une méthode de résolution par rapport à une autre en se basant sur plusieurs métriques de performances. Les résultats obtenus étaient très intéressants et ont prouvé l'efficacité des méthodes de résolution proposées en termes de qualité des solutions et temps d'exécution.
L'ensemble des modèles et des méthodes développés dans cette thèse peut former un outil d'aide efficace à la décision pour le concepteur du système afin de concevoir des systèmes performants avec le moindre coût.
Les travaux présentés dans ce mémoire ont donné lieu à plusieurs publications [30], [127], [128], [31], [148], [163].

Les différents travaux présentés dans cette thèse mettent en perspective plusieurs pistes de recherche future. Ces dernières peuvent être résumées en trois axes principaux à savoir : la dépendance stochastique, l'étude des systèmes plus complexes et l'optimisation de la conception.

En ce qui concerne la dépendance stochastique, nous avons considéré le cas de la dépendance positive, en particulier la dépendance redondante. Il serait utile de traiter aussi le cas de la dépendance négative, dans le sens où la mise en fonctionnement d'un composant accélère la défaillance des autres composants. Ainsi, la dépendance négative implique que le système est plus fiable quand il ne contient qu'un seul composant. Au cours de la thèse, nous avons traité les systèmes k sur n : G et les systèmes séries-k sur n : G. Nous avons évalué la disponibilité de ces systèmes tout en introduisant la notion de la dépendance redondante via une fonction de dépendance. La dépendance redondante s'est montrée comme un élément important dans l'analyse de la sûreté de fonctionnement du système. Il serait intéréssant d'explorer de plus en profondeur les aspects pratiques de ce concept. Nous prévoyons la possibilité de réaliser de tests

sur des données réelles issues de collaborations avec des industriels et de trouver autres fonctions dépendantes adaptées à ces problèmes pratiques. Une application pourrait être le problème de lot sizing en production qui vise à satisfaire la demande sur un horizon temporel donné. La prise en compte de la dépendance entre les machines redondants et de la règle de répartition de charges peut affecter la performance fiabiliste de la ligne de production.

En ce qui concerne l'extension de notre étude aux systèmes plus complexes, nous envisageons la possibilité de considérer des systèmes multi-états où les composants peuvent avoir en plus d'un fonctionnement parfait et d'un échec total, des états intermédiares. En effet, dans de nombreux cas, le système peut avoir un fonctionnement partiel lors de sa dégradation entre un meilleur fonctionnement et un échec total. Dans ce cas, la performance du système dépend du nombre de composants qui sont en état de fonctionnement parfait et du nombre de composants qui sont en état dégradé. Le problème de l'explosion des états est une difficulté qui doit être tenir en compte. Les modèles Markoviens peuvent être limités pour une telle étude. Les modèles non Markoviens peuvent être proposés. De plus, les composants peuvent être caractérisés par autres lois de distribution de défaillance et de réparation. De paramètres supplémentaires peuvent aussi être intégrés qui influencent le coût total du système comme le coût d'installation, les coûts de fonctionnement, les coûts de changements d'outils, etc.

En ce qui concerne l'optimisation de la conception, plusieurs points peuvent être approfondis.

- Appliquer de nouvelles heuristiques ou métaheuristiques de résolution pour les problèmes primal et dual.
- Appliquer de méthodes exactes..
- Développer de nouvelles approches d'hybridation à tester comme la technique de la recherche locale guidée.
- Améliorer la performance des algorithmes appliqués en réalisant une série de tests plus approfondies afin de bien régler leurs paramètres.
- Ajouter de nouveaux objectifs à optimiser (comme le volume dans le cas multiobjectif) ou des contraintes pouvant rendre le problème étudié plus réaliste.

Bibliographie

[1] D. J. Smith. *Reliability, maintainability and risk. Practical methods for engineers.* Sixth Edition. Butterworth Heinemann, 2001.

[2] A. Villemeur. *Sûreté de fonctionnement des systèmes industriels : Fiabilité, Facteurs humains, Informatisation.* Paris : Eyrolles, 1988.

[3] J. Dréo, A. Pétrowski, P. Siarry, and E. Taillard. *Métaheuristiques pour l'optimisation difficile.* Eyrolles, 2005.

[4] http ://www.collective-behavior.com/spontaneous-path-formation-in ants.

[5] H. Yu, C. Chu, and E. Châtelet. Availability optimization of a redundant system through dependency modeling. *Applied Mathematical Modelling*, 38(19–20) :4574 – 4585, 2014.

[6] J.C. Laprie. *Guide de la sûreté de fonctionnement.* Cépaduès-Editions, 1996.

[7] A. Pagès and M. Gondran. *Fiabilité des systèmes.* Editions Eyrolles, 1980.

[8] A. hoyland and M. Rausand. *System Reliabiity Theory.* John Wiley & Sons, 1994.

[9] IEEE 90 – Institute of Electrical and IEEE Standard Computer Dictionary Electronics Engineers. *Une compilation des glossaires des normes IEEE relatives à l'informatique.* New York, NY, 1990.

[10] C.E. Ebeling. *An introduction to reliability and maintainability engineering.* McGRAW-HILL companies, Inc., 1997.

[11] S. Carpitella, A. Certa, J. Izquierdo, and C. M. La Fata. k−out−of−n systems : An exact formula for the stationary availability and multi-objective configuration design based on mathematical programming and {TOPSIS}. *Journal of Computational and Applied Mathematics*, pages –, 2017.

[12] J. Fournier. *Fiabilité du logiciel.* Hermes, Paris, 1993.

[13] J. Ringler. *Précis de Probabilités et de Statistiques à l'Usage de la Fiabilité*. OCTARES Editions, 1996.

[14] G. A. P. Castaneda. *Evaluation par simulation de la sûreté de fonctionnement de systèmes en contexte dynamique hybride*. Automatique / Robotique. Institut National Polytechnique de Lorraine - INPL, 2009.

[15] L.C. Thomas. A survey of maintenance and replacement models for maintainability and reliability of multi-item systems. *Reliability Engineering*, 16(4) :297–309, 1986.

[16] K. B. Misra. *New trends in system reliability evaluation, The Netherlands*. Elsevier Science Publishers, Amsterdam, 1993.

[17] R. Fricks and K.S. Trivedi. Modeling failure dependencies in reliability analysis using stochastic petri nets. In *Proc. European Simulation Multi-conference (ESM '97), Istanbul*, 1997.

[18] G. Levitin. A universal generating function approach for the analysis of multi-state systems with dependent elements. *Reliability Engineering & System Safety*, 84(3) :285 – 292, 2004.

[19] NASA. *Fault Tree Handbook with Aerospace Applications*. NASA Office of Safety and Mission Assurance, Washington, DC., 2002.

[20] C. N. Guey and C. D. Heising. Development of a common cause failure analysis method : The inverse stress-strength interference (issi) technique. *Structural Safety*, 4(1) :63 – 77, 1986.

[21] Z. Pan and Y. Nonaka. Importance analysis for the systems with common cause failures. *Reliability Engineering & System Safety*, 50(3) :297 – 300, 1995.

[22] L. Xie, J. Zhou, and C. Hao. System-level load–strength interference based reliability modeling of k−out−of−n system. *Reliability Engineering & System Safety*, 84(3) :311 – 317, 2004.

[23] G. Levitin, L. Xing, S. V. Amari, and Y. Dai. Reliability of non-repairable phased-mission systems with propagated failures. *Reliability Engineering & System Safety*, 119 :218 – 228, 2013.

[24] T. Yuge, M. Maruyama, and S. Yanagi. Reliability of a k−out−of−n system with common-cause failures using multivariate exponential distribution. *Procedia Computer Science*, 96 :968 – 976, 2016.

[25] L. Chang and Z. Wu. Performance and reliability of electrical power grids under cascading failures. *International Journal of Electrical Power & Energy Systems*, 33(8) :1410 – 1419, 2011.

[26] I. Eusgeld, C. Nan, and S. Dietz. "system-of-systems" approach for interdependent critical infrastructures. *Reliability Engineering & System Safety*, 96(6) :679 – 686, 2011. {ESREL} 2009 Special Issue.

[27] H. YU. *Optimisation de la fiabilité et de la disponibilité des systèmes à composants dependants*. PhD thesis, Université de Technologie de Troyes, 2006.

[28] H. Yu, C. Chu, E. Châtelet, and F. Yalaoui. Reliability optimization of a redundant system with failure dependencies. *Reliability Engineering & System Safety*, 92(12) :1627 – 1634, 2007. Special Issue on {ESREL} 2005.

[29] L. Yuan. Reliability analysis for a k−out−of−n :g system with redundant dependency and repairmen having multiple vacations. *Applied Mathematics and Computation*, 218(24) :11959 – 11969, 2012.

[30] **M. Habib**, H. Chehade, F. Yalaoui, N. Chebbo, and I. Jarkass. Availability optimization of a redundant dependent system using genetic algorithm. *IFAC-PapersOnLine*, 49(12) :733 – 738, 2016.

[31] **M. Habib**, F. Yalaoui, H. Chehade, N. Chebbo, and I. Jarkass. Design optimization and redundant dependency study of series k−out−of−n : G repairable systems. *IFAC-PapersOnLine*, 49(28) :126 – 131, 2016.

[32] A. Habib, R. O. Al-Seedy, and T. Radwan. Reliability evaluation of multi-state consecutive k−out−of−r-from−n : G system. *Applied Mathematical Modelling*, 31(11) :2412 – 2423, 2007.

[33] S. Eryilmaz. Parallel and consecutive-k−out−of−n :f systems under stochastic deterioration. *Applied Mathematics and Computation*, 227 :19 – 26, 2014.

[34] C. Franko, M. Ozkut, and C. Kan. Reliability of coherent systems with a single cold standby component. *Journal of Computational and Applied Mathematics*, 281 :230 – 238, 2015.

[35] K. D. Heidtmann. A class of noncoherent systems and their reliability analysis. In *Proceedings 11th Annu. Symp. Fault Tolerant Computing*, pages 96–98, 1981.

[36] H. Pham. Optimal cost effective design of a class of noncoherent systems. In *Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences*, volume i, pages 67–74 vol.1, Jan 1991.

[37] S. J. Upadhyaya and H. Pham. Analysis of noncoherent systems and an architecture for the computation of the system reliability. *IEEE Transactions on Computers*, 42(4) :484–493, Apr 1993.

[38] G. Levitin, L. Xing, and Y. Dai. Reliability of non-coherent warm standby systems with reworking. *IEEE Transactions on Reliability*, 64(1) :444–453, March 2015.

[39] A. Lisnianski and G. Levitin. *Multi-state system reliability, Assessment, Optimization and Applications*. Series on Quality, Reliability & Engineering Statistics, World Scientific Publishing Co. Pte. Ltd.,Singapore, 2003.

[40] J. Li, G. Chen, J. Li, and R. Wang. Availability evaluation and design optimization of multi-state weighted k−out−of−n systems. In *2016 Prognostics and System Health Management Conference (PHM-Chengdu)*, pages 1–6, Oct 2016.

[41] B. S. Dhillon and C. Singh. On fault trees and other reliability evaluation methods. *Microelectronics Reliability*, 19(1–2) :57 – 63, 1979.

[42] J. B. Dugan, S. J. Bavuso, and M. A. Boyd. Fault trees and markov models for reliability analysis of fault-tolerant digital systems. *Reliability Engineering & System Safety*, 39(3) :291 – 307, 1993.

[43] S. Contini, G. G. M. Cojazzi, and G. Renda. On the use of non-coherent fault trees in safety and security studies. *Reliability Engineering & System Safety*, 93(12) :1886 – 1895, 2008. 17th European Safety and Reliability Conference.

[44] S. Kabir. An overview of fault tree analysis and its application in model based dependability analysis. *Expert Systems with Applications*, 77 :114 – 135, 2017.

[45] F. Di Maio, S. Baronchelli, and E. Zio. Hierarchical differential evolution for minimal cut sets identification : Application to nuclear safety systems. *European Journal of Operational Research*, 238(2) :645 – 652, 2014.

[46] J. K. Muppala, R. M. Fricks, and K. S Trivedi. *Techniques for system dependability evaluation*. Springer US, 2000.

[47] A. Villemeur. *Evaluation de la fiabilité, disponibilité et maintenabilité des systèmes réparables : la méthode de l'Espace des Etats*. EDF-DER HT/50/3-juillet, 1987.

[48] H. Goldberg. *Extending the limits of reliability theory*. John Wiley & Sons, 1981.

[49] Ph. Chretienne and R. Faure. *Processus stochastiques, leurs graphes, leurs usages*. Editions Gauthier-Villars, 1974.

[50] E. Niel and E. Craye. *Maîtrise des risques et sûreté de fonctionnement des systèmes de production*. ELavoisier, 2002.

[51] J. H. Chiang and J. Yuan. Optimal maintenance policy for a markovian system under periodic inspection. *Reliability Engineering & System Safety*, 71(2) :165 – 172, 2001.

[52] A. Reibman, R. Smith, and K. Trivedi. Markov and markov reward model transient analysis : An overview of numerical approaches. *European Journal of Operational Research*, 40(2) :257 – 267, 1989.

[53] M. L. Putterman. *Markov Decision Proceses. Discrete stochast & Sons Inc.* 1994.

[54] W. Duan and Z. Wang. Vibration reliability analysis of turbine blade based on ann and monte carlo simulation. In *2010 Sixth International Conference on Natural Computation*, volume 4, pages 1934–1939, Aug 2010.

[55] C. F. DeSieno and L. L. Stine. A probability method for determining the reliability of electric power systems. *IEEE Transactions on Reliability*, R-14(1) :30–35, March 1965.

[56] X. Liang and L. Goel. Distribution system reliability evaluation using the monte carlo simulation method. *Electric Power Systems Research*, 40(2) :75 – 83, 1997.

[57] D. Cheng, D. Zhu, R. P. Broadwater, and S. Lee. A graph trace based reliability analysis of electric power systems with time-varying loads and dependent failures. *Electric Power Systems Research*, 79(9) :1321 – 1328, 2009.

[58] M. A. Drighiciu, G. Manolea, and A. Petrisor. A hybrid petri net approach for manufacturing systems design. {*IFAC*} *Proceedings Volumes*, 40(18) :499 – 504, 2007. 4th {IFAC} Conference on Management and Control of Production and Logistics.

[59] F. Long, P. Zeiler, and B. Bertsche. Modelling the production systems in industry 4.0 and their availability with high-level petri nets. *IFAC-PapersOnLine*, 49(12) :145 – 150, 2016. 8th {IFAC} Conference on Manufacturing Modelling, Management and Control {MIM} 2016Troyes, France, 28—30 June 2016.

[60] E. Cabau. *Introduction à la conception de la sûreté.* Cahier technique numéro 144, Schneider Electric, 1999.

[61] W. Kuo and V. R. Prasad. An annotated overview of system-reliability optimization. *IEEE Transactions on Reliability*, 49(2) :176–187, Jun 2000.

[62] A. Yalaoui. *Allocation de fiabilité et de redondance dans les systèmes parallèle-série et série-parallèle.* PhD thesis, Université de Technologie de Troyes, 2004.

[63] FA. Tillman, CL. Hwang, and W. Kuo. Optimization techniques for system reliability with redundancy-a review. *IEEE Transactions on Reliability*, R-26(3) :148 – 155, 1977.

[64] A. Yalaoui, C. Chu, and E. Châtelet. Reliability allocation problem in a series–parallel system. *Reliability Engineering & System Safety*, 90(1) :55 – 61, 2005.

[65] W. Kuo and R. Wan. Recent advances in optimal reliability allocation. *IEEE Transactions on Systems, Man, and Cybernetics - Part A : Systems and Humans*, 37(2) :143–156, March 2007.

[66] Z. Tian, M. J. Zuo, and H. Huang. Reliability-redundancy allocation for multi-state series-parallel systems. *IEEE Transactions on Reliability*, 57(2) :303–310, June 2008.

[67] F. Yue, G. Zhang, Z. Su, Y. Lu, and T. Zhang. Multi-software reliability allocation in multimedia systems with budget constraints using dempster–shafer theory and improved differential evolution. *Neurocomputing*, 169 :13 – 22, 2015.

[68] J. Zhao, S. Zeng, J. Guo, and C. Yang. Redundancy allocation with non-identical component and uncertainty. In *2015 Annual Reliability and Maintainability Symposium (RAMS)*, pages 1–6, Jan 2015.

[69] K. Misra. On optimal reliability design : a review. *Systems Science*, 12(4) :5–30, 1986.

[70] A. Yalaoui, E. Châtelet, and C. Chu. A new dynamic programming method for reliability redundancy allocation in a parallel-series system. *IEEE Transactions on Reliability*, 54(2) :254–261, June 2005.

[71] A. O. C. Elegbede, C. Chu, K. H. Adjallah, and F. Yalaoui. Reliability allocation through cost minimization. *IEEE Transactions on Reliability*, 52(1) :106–111, March 2003.

[72] H. Kim and P. Kim. Reliability–redundancy allocation problem considering optimal redundancy strategy using parallel genetic algorithm. *Reliability Engineering & System Safety*, 159 :153 – 160, 2017.

[73] C. Elegbede and K. Adjallah. Availability allocation to repairable systems with genetic algorithms : a multi-objective formulation. *Reliability Engineering & System Safety*, 82(3) :319 – 330, 2003.

[74] J. Barabady and U. Kumar. Availability allocation through importance measures. *International Journal of Quality & Reliability Management*, 24(6) :643–657, 2007.

[75] J. Rupe. Review of optimal reliability design, fundamentals and applications. *IEEE Transactions on Reliability*, 56(1) :167–167, March 2007.

[76] D. Li. Iterative parametric dynamic programming and its application in reliability optimization. *Journal of Mathematical Analysis and Applications*, 191(3) :589 – 607, 1995.

[77] W. Kuo, H. H. Lin, Z. Xu, and W. Zhang. Reliability optimization with the lagrange-multiplier and branch-and-bound technique. *IEEE Transactions on Reliability*, R-36(5) :624–630, Dec 1987.

[78] D. Li, X. Sun, and K. McKinnon. An exact solution method for reliability optimization in complex systems. *Annals of Operations Research*, 133(1) :129–148, 2005.

[79] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1957.

[80] A. Chevalier. *La Programmation dynamique*. Dunod, 1957.

[81] K. B. Misra and U. Sharma. An efficient algorithm to solve integer-programming problems arising in system-reliability design. *IEEE Transactions on Reliability*, 40(1) :81–91, Apr 1991.

[82] S. C. Sup and C. Y. Kwon. Branch-and-bound redundancy optimization for a series system with multiple-choice constraints. *IEEE Transactions on Reliability*, 48(2) :108–117, Jun 1999.

[83] Y. Collette and P. Siarry. *Optimisation multiobjectif : Algorithmes*. Eyrolles, 2011.

[84] J. H. Holland. *Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press, 1975.

[85] D. Goldberg. *Genetic algorithms in search, optimization and machine learning*. Kluwer Academic Publishers, 1989.

[86] A. Lisnianski, G. Levitin, H. Ben-Haim, and D. Elmakis. Power system structure optimization subject to reliability constraints. *Electric Power Systems Research*, 39(2) :145 – 152, 1996.

[87] Y.F. Li and R. Peng. Availability modeling and optimization of dynamic multi-state series–parallel systems with random reconfiguration. *Reliability Engineering & System Safety*, 127 :47 – 57, 2014.

[88] A. Colorni, M. Dorigo, and V. Maniezzo. Distributed optimization by ant colonies. In *Proceedings of the First European Conference on Artificial Life*, pages 134–142, 1992.

[89] Y-C Liang and A. E. Smith. An ant colony optimization algorithm for the redundancy allocation problem (rap). *IEEE Transactions on Reliability*, 53(3) :417–423, Sept 2004.

[90] O. Bendjeghaba and D. Ouahdi. Multi-agent ant system for redundancy allocation problem of multi states power system. In *Power and Energy Conference, 2008. PECon 2008. IEEE 2nd International*, pages 1270–1274, Dec 2008.

[91] V. K. Sharma and M. Agarwal. Ant colony optimization approach to heterogeneous redundancy in multi-state systems with multi-state components. In *Reliability, Maintainability and Safety, 2009. ICRMS 2009. 8th International Conference on*, pages 116–121, July 2009.

[92] M. S. Rao, D. S. Roy, P. R. Parro, and D. K. Mohanta. An ant colony metaheuristic approach for optimal reliability assessment of software systems incorporating redundancy. In *Information and Communication Technologies (WICT), 2012 World Congress on*, pages 577–582, Oct 2012.

[93] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5) :533 – 549, 1986.

[94] J. A. BLAND. Structural design optimization with reliability constraints using tabu search. *Engineering Optimization*, 30(1) :55–74, 1998.

[95] S. Kulturel, A. E. Smith, and D. W. Coit. Efficiently solving the redundancy allocation problem using tabu search. *IIE Transactions*, 35(6) :515–526, 2003.

[96] M. Ouzineb, M. Nourelfath, and M. Gendreau. Tabu search for the redundancy allocation problem of homogenous series–parallel multi-state systems. *Reliability Engineering & System Safety*, 93(8) :1257 – 1272, 2008.

[97] G. Cheng, K. Raahemifar, and O. Das. Comparative study of heuristics for reliability optimization of complex systems. In *CCECE 2010*, pages 1–4, May 2010.

[98] S. Kirkpatrick1, C. Gelatt Jr, and M. Vecchi. Optimization by simulated annealing. *Science*, 220(4598) :671–680, 1983.

[99] B. Suman. Simulated annealing-based multiobjective algorithms and their application for system reliability. *Engineering Optimization*, 35(4) :391–416, 2003.

[100] H. Kim, C. Bae, and D. Park. Reliability-redundancy optimization using simulated annealing algorithms. *Journal of Quality in Maintenance Engineering*, 12(4) :354–363, 2006.

[101] A. Chambari, A. Najafi, S. Rahmati, and A. Karimi. An efficient simulated annealing algorithm for the redundancy allocation problem with a choice of redundancy strategies. *Reliability Engineering & System Safety*, 119 :158 – 164, 2013.

[102] A. Zaretalab, V. Hajipour, M. Sharifi, and M. Shahriari. A knowledge-based archive multi-objective simulated annealing algorithm to optimize series–parallel system with choice of redundancy strategies. *Computers & Industrial Engineering*, 80 :33 – 44, 2015.

[103] Appendix ii : Lingo software. In Mahmoud M. El-Halwagi, editor, *Process Integration*, volume 7 of *Process Systems Engineering*, pages 389 – 394. Academic Press, 2006.

[104] H. Sarper. No special schemes are needed for solving software reliability optimization models. *IEEE Transactions on Software Engineering*, 21(8) :701–702, Aug 1995.

[105] I. D. Lins and E. L. Droguett. Multiobjective optimization of availability and cost in repairable systems design via genetic algorithms and discrete event simulation. *Pesquisa Operacional*, 29 :43–66, April 2009.

[106] M. Kostreva and W. Ogryczak. Linear optimization with multiple equitable criteria. *RAIRO-Operations Research*, 33(3) :275–297, 1999.

[107] M. M. Kostreva, W. Ogryczak, and A. Wierzbicki. Equitable aggregations and multiple criteria analysis. *European Journal of Operational Research*, 158(2) :362–377, 2004.

[108] P. Perny, O. Spanjaard, and L-X Storme. A decision-theoretic approach to robust optimization in multivalued graphs. *Annals of Operations Research*, 147(1) :317–341, 2006.

[109] F. Dugardin, F. Yalaoui, and L. Amodeo. Reentrant lines scheduling and lorenz dominance : a comparative study. In *Proceedings of the 8th International FLINS Conference*, pages 707–712, 2008.

[110] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms : a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4) :257–271, Nov 1999.

[111] M. Laumanns, E. Zitzler, and L. Thiele. A unified model for multi-objective evolutionary algorithms with elitism. In *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, volume 1, pages 46–53, 2000.

[112] A. Riise. Comparing genetic algorithms and tabu search for multi-objective optimization. In *In Proceedings of the International Federation of Operational Research Societies conference IFORS, Edinburgh, UK*, page 29, 2002.

[113] L. Amodeo, H. Chen, and A. El Hadji. Multi-objective supply chain optimization : An industrial case study. *Lecture Notes in Computer Science*, 4448 :732–741, 2007.

[114] H. Chehade, F. Yalaoui, L.Amodeo, and P. De Guglielmo. Optimisation multi-objectif pour le problème de dimensionnement de buffers. *Journal of Decision Systems*, 18(2) :257–287, 2009.

[115] N. Alikar, SM. Mousavi, RAR. Ghazilla, M. Tavana, and EU. Olugu. Application of the nsga-ii algorithm to a multi-period inventory-redundancy allocation problem in a series-parallel system. *Reliability Engineering & System Safety*, 160 :1–10, 2017.

[116] J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 93–100, 1985.

[117] C. M. Fonseca and P. J. Fleming. Genetic algorithms for multi-objective optimization : formulation, discussion and generalization. In *The Fifth International Conference on Genetic Algorithms*, pages 416–423, 1993.

[118] J. Horn, N. Nafpliotis, and D. E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, pages 82–87 vol.1, Jun 1994.

[119] N. Srinivas and K. Deb. Multiobjective optimization using non dominated sorting in genetic algorithms. evolutionary computation. *Evolutionary Computation*, 2(3) :221–248, 1994.

[120] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm : Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2) :182–197, Apr 2002.

[121] M. Gen, W. Zhang, L. Lin, and Y. Yun. Recent advances in hybrid evolutionary algorithms for multiobjective manufacturing scheduling. *Computers & Industrial Engineering*, pages –, 2017.

[122] A. B. Arabani, M. Zandieh, and S. F. Ghomi. Multi-objective genetic-based algorithms for a cross-docking scheduling problem. *Applied Soft Computing*, 11(8) :4954 – 4970, 2011.

[123] I. D. Lins and E. L. Droguett. Redundancy allocation problems considering systems with imperfect repairs using multi-objective genetic algorithms and discrete event simulation. *Simulation Modelling Practice and Theory*, 19(1) :362 – 381, 2011. Modeling and Performance Analysis of Networking and Collaborative Systems.

[124] A. de Almeida, R. Ferreira, and C. Cavalcante. A review of the use of multicriteria and multiobjective models in maintenance and reliability. *IMA Journal of Management Mathematics*, 26(3) :249, 2015.

[125] M. K. Ghorabaee, M. Amiri, and P. Azimi. Genetic algorithm for solving bi-objective redundancy allocation problem with k−out−of−n subsystems. *Applied Mathematical Modelling*, 39(20) :6396 – 6409, 2015.

[126] F. Kayedpour, M. Amiri, M. Rafizadeh, and A. S. Nia. Multi-objective redundancy allocation problem for a system with repairable components considering instantaneous availability and strategy selection. *Reliability Engineering & System Safety*, 160 :11 – 20, 2017.

[127] **M. Habib**, F. Yalaoui, H. Chehade, N. Chebbo, and I. Jarkass. Availability analysis and redundant dependency modeling of k-out-of-n : g system. In *10th International conference on mathematical methods in reliability, MMR'17, France*, July 2017.

[128] **M. Habib**, F. Yalaoui, H. Chehade, N. Chebbo, and I. Jarkass. Evaluation et optimisation des performances des systèmes de production à la phase de conception. In *22nd Scientific International Conference, The Social Avenues of the Research, LAAS'16, Lebanon*, April 2016.

[129] L. Wang, X. Jia, and J. Zhang. Reliability evaluation for multi-state markov repairable systems with redundant dependencies. *Quality Technology & Quantitative Management*, 10(3) :277–289, 2013.

[130] B. Jafary and L. Fiondella. A universal generating function-based multi-state system performance model subject to correlated failures. *Reliability Engineering & System Safety*, 152 :16 – 27, 2016.

[131] S. Li and J. Lynch. On a threshold representation for complex load-sharing systems. *Journal of Statistical Planning and Inference*, 141(8) :2811 – 2823, 2011.

[132] J. K. Vaurio. Treatment of general dependencies in system fault-tree and risk analysis. *IEEE Transactions on Reliability*, 51(3) :278–287, Sep 2002.

[133] G. Haitao Guo and X. Yang. Automatic creation of markov models for reliability assessment of safety instrumented systems. *Reliability Engineering & System Safety*, 93(6) :829 – 837, 2008.

[134] C.D. Lai and M. Xie. A new family of positive quadrant dependent bivariate distributions. *Statistics & Probability Letters*, 46(4) :359 – 364, 2000.

[135] R. B. Nelsen. On measures of association as measures of positive dependence. *Statistics & Probability Letters*, 14(4) :269 – 274, 1992.

[136] M. Xie S. Kotz, CD. Lai. On the effect of redundancy for systems with dependent components. *IIE Transactions*, 35(12) :1103–1110, 2003.

[137] B. Güven. Test of independence in the farlie-gumbel-morgenstern distribution. *Communications in Statistics-Theory and Methods*, 32(9) :1753–1765, 2003.

[138] A. Z. Hamadani and A. N. Nasrabadi. The effect of dependency on the mrl function of redundant systems. In *2007 IEEE International Conference on Industrial Engineering and Engineering Management*, pages 1191–1195, Dec 2007.

[139] CE. Ebeling. *An introduction to reliability and maintainability engineering.* Tata McGraw-Hill Education, 2004.

[140] A. Blokus. Reliability analysis of large systems with dependent components. *International Journal of Reliability, Quality and Safety Engineering*, 13(01) :1–14, 2006.

[141] P. H. Kvam and E. A. Pena. Estimating load-sharing properties in a dynamic reliability system. *Journal of the American Statistical Association.*, 100 :262–272, 2005.

[142] I. Maatouk, E. Châtelet, and N. Chebbo. Reliability of multi-states system with load sharing and propagation failure dependence. In *Quality, Reliability, Risk, Maintenance, and Safety Engineering (ICQR2MSE), 2011 International Conference on*, pages 42–46, June 2011.

[143] L. Hu, D. Yue, and J. Li. Availability analysis and design optimization for a repairable series-parallel system with failure dependencies. *International Journal of Innovative Computing, Information and Control*, 8(10) :6693–6705, 2012.

[144] L. Wang, Zhang J, Chen W, and X. Jia. Reliability evaluation of a load-sharing parallel system with failure dependence. *Communications in Statistics-Simulation and Computation*, 45(9) :3094–3113, 2016.

[145] W. Kuo and M. J. Zuo. *Optimal reliability modeling : principles and applications.* John Wiley & Sons, 2003.

[146] G. S. Liu. Availability optimization for repairable parallel-series system by applying tabu-ga combination method. In *IEEE 10th International Conference on Industrial Informatics*, pages 803–808, July 2012.

[147] I. Maatouk, E. Châtelet, and N. Chebbo. Availability maximization and cost study in multi-state systems. In *Reliability and Maintainability Symposium (RAMS), 2013 Proceedings - Annual*, pages 1–6, Jan 2013.

[148] **M. Habib**, H. Chehade, F. Yalaoui, N. Chebbo, and I. Jarkass. Maximization of system availability with failure dependency. In *2016 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET)*, pages 115–119, Nov 2016.

[149] L. Cui, C. Lin, and S. Du. m−consecutive−k , l−out−of−n systems. *IEEE Transactions on Reliability*, 64(1) :386–393, March 2015.

[150] M. Nourelfath and D. Ait-Kadi. Optimization of series–parallel multi-state systems under maintenance policies. *Reliability Engineering & System Safety*, 92(12) :1620 – 1626, 2007. Special Issue on {ESREL} 2005.

[151] M. S. Chern. On the computational complexity of reliability redundancy allocation in a series system. *Operations Research Letters*, 11(5) :309 – 315, 1992.

[152] I. Sooktip, N. Wattanapongsakorn, and D.W. Coit. System reliability optimization with k−out−of−n subsystems and changing k. In *Reliability, Maintainability and Safety (ICRMS), 2011 9th International Conference on*, pages 1382–1387, June 2011.

[153] Prashanthi Boddu and Liudong Xing. Reliability evaluation and optimization of series–parallel systems with k−out−of−n : G subsystems and mixed redundancy types. *Proceedings of the Institution of Mechanical Engineers, Part O : Journal of Risk and Reliability*, 227(2) :187–198, 2013.

[154] V. Ebrahimipur and A. Shabani. Optimization multi -state series weighted k−out−of−n systems by ant colony algorithm. In *2009 IEEE International Conference on Industrial Engineering and Engineering Management*, pages 281–285, Dec 2009.

[155] M. Amiri, F. Ghassemi-Tari, M. R. M. Shahi, J. S. Sadaghiani, and A. Mahtasshami. A methodology for analyzing the transient availability and survivability of a system with every combination of components by using fault tree. *Journal of Applied Sciences*, 9 :1074–1081, 2010.

[156] D. Huffman, R. Bergman, S. V. Amari, and M. J. Zuo. Availability analysis of systems with suspended animation. In *2008 Annual Reliability and Maintainability Symposium*, pages 283–288, Jan 2008.

[157] R. Moghaddass, M.J. Zuo, and Q. Jian. Reliability and availability analysis of a repairable k−out−of−n :g system with R repairmen subject to shut-off rules. *Reliability, IEEE Transactions on*, 60(3) :658–666, Sept 2011.

[158] Eleni I. Vlahogianni, Matthew G. Karlaftis, and John C. Golias. Optimized and meta-optimized neural networks for short-term traffic flow prediction : A genetic approach. *Transportation Research Part C : Emerging Technologies*, 13(3) :211 – 234, 2005.

[159] A. Dey and R. Mukerjee. *Fractional, factorial plans.* In : Series of Probability and Statistics. New York : Wiley, 1999.

[160] Xiaohui Li, Lionel Amodeo, Farouk Yalaoui, and Hicham Chehade. A multiobjective optimization approach to solve a parallel machines scheduling problem. *Advances in Artificial Intelligence*, 2010.

[161] F. Khodadadi, S. J. Mirabedini, and A. Harounabadi. Improve traffic management in the vehicular ad hoc networks by combining ant colony algorithm and fuzzy system. *International Journal of Advanced Computer Science and Applications(ijacsa)*, 7(4), 2016.

[162] M. Samrout, R. Kouta, F. Yalaoui, E. Châtelet, and N. Chebbo. Parameter's setting of the ant colony algorithm applied in preventive maintenance optimization. *Journal of Intelligent Manufacturing*, 18(6) :663–677, 2007.

[163] **M. Habib**, F. Yalaoui, H. Chehade, N. Chebbo, and I. Jarkass. Multi-objective design optimization of repairable k−out−of−n subsystems in series with redundant dependency. *International Journal of Production Research*, pages 1–22, 2017.

[164] M. A. Ardakan, A. Z. Hamadani, and M. Alinaghian. Optimizing bi-objective redundancy allocation problem with a mixed redundancy strategy. {*ISA*} *Transactions*, 55 :116 – 128, 2015.

[165] N. Alikar, SM. Mousavi, M. Tavana, and STA. Niaki. A multi-objective multi-state series-parallel redundancy allocation model using tuned meta-heuristic algorithms. *International Journal of Systems Science : Operations & Logistics*, pages 1–22, 2016.

[166] Jalal Safari. Multi-objective reliability optimization of series-parallel systems with a choice of redundancy strategies. *Reliability Engineering & System Safety*, 108 :10 – 20, 2012.

[167] SM. Mousavi, N. Alikar, STA. Niaki, and A. Bahreininejad. Two tuned multi-objective meta-heuristic algorithms for solving a fuzzy multi-state redundancy allocation problem under discount strategies. *Applied Mathematical Modelling*, 39(22) :6968–6989, 2015.

[168] A. Attar, S. Raissi, and K. Khalili-Damghani. A simulation-based optimization approach for free distributed repairable multi-state availability-redundancy allocation problems. *Reliability Engineering & System Safety*, 157 :177–191, 2017.

[169] S.J Findlay and N.D Harrison. Why aircraft fail. *Materials Today*, 5(11) :18 – 25, 2002.

[170] B. Iung, P. Do, E. Levrat, and A. Voisin. Opportunistic maintenance based on multi-dependent components of manufacturing system. {*CIRP*} *Annals - Manufacturing Technology*, 65(1) :401–404, 2016.

[171] H. Zoulfaghari, A. Z. Hamadani, and M. A. Ardakan. Bi-objective redundancy allocation problem for a system with mixed repairable and non-repairable components. {*ISA*} *Transactions*, 53(1) :17 – 24, 2014.

[172] SM. Mousavi, N. Alikar, and STA. Niaki. An improved fruit fly optimization algorithm to solve the homogeneous fuzzy series-parallel redundancy allocation problem under discount strategies. *Soft Computing*, 20(6) :2281–2307, 2016.

[173] SM. Mousavi, N. Alikar, M. Tavana, and D. Di Caprio. An improved particle swarm optimization model for solving homogeneous discounted series-parallel redundancy allocation problems. *Journal of Intelligent Manufacturing*, pages 1–20, 2017.

[174] David W. Coit. Maximization of system reliability with a choice of redundancy strategies. *IIE Transactions*, 35(6) :535–543, 2003.

[175] Z. Wang, T. Chen, K. Tang, and X. Yao. A multi-objective approach to redundancy allocation problem in parallel-series systems. In *2009 IEEE Congress on Evolutionary Computation*, pages 582–589, May 2009.

[176] J. Branke and K. Deb. *Multi-objective optimization interactive and evolutionary approaches*. Springer, 2008.

[177] N. Alikar, SM. Mousavi, RAR. Ghazilla, M. Tavana, and EU. Olugu. A bi-objective multi-period series-parallel inventory-redundancy allocation problem with time value of money and inflation considerations. *Computers & Industrial Engineering*, 104 :51–67, 2017.

[178] R. Kumar, K. Izui, M. Yoshimura, and S. Nishiwaki. Multi-objective hierarchical genetic algorithms for multilevel redundancy allocation optimization. *Reliability Engineering & System Safety*, 94(4) :891 – 904, 2009.

[179] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization : Nsga-ii. In *Proceedings of the Parallel Problem Solving from Nature VI Conference*, page 849–858, 2000.

[180] E. Zitzler, M. Laumanns, and L. Thiele. Spea2 : Improving the strength pareto evolutionary algorithm. *Eurogen*, 3242(103) :95–100, 2001.

[181] Y. Tao and L. Lu. Risk-informed maintenance for non-coherent systems. In *2013 Proceedings Annual Reliability and Maintainability Symposium (RAMS)*, pages 1–6, Jan 2013.

[182] J. Fournier. *Fiabilité du logiciel*. Hermes, Paris, 1993.

[183] Comission Electronique Internationale. *Liste des termes de base, définitions et mathématiques applicables à la fiabilité*. Publications 271(1974), 271A (1978), 271B(1983), 271C(1985).

[184] AFNOR-Norme. *Statistique et Qualité, Introduction à la fiabilité, X NF 06-501*. X NF 06-501, 1977.

[185] M. A. Geoffrion. Integer programming by implicit enumeration and balas' method. *Siam Review*, 9(2) :178–190, 1967.

[186] A. Berrichi, F. Yalaoui, L. Amodeo, and M. Mezghiche. Bi-objective ant colony optimization approach to optimize production and maintenance scheduling. *Computers & Operations Research*, 37(9) :1584 – 1596, 2010.

[187] A. Suppapitnarm and T. Parks. Simulated annealing : an alternative approach to true multiobjective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999)*, pages 406–407, 1999.

[188] G. Deleuze, N. Brinzei, and N. Villaume. Représentation des défaillances de cause commune d'un système programme de grande taille par réseaux de petri colores, temporises et hiérarchiques. In *19ème Congrès de Maîtrise des Risques et Sûreté de Fonctionnement, Lambda-Mu'2014*, 2014.

[189] J-P Signoret, Y. Dutuit, P-J Cacheux, C. Folleau, S. Collas, and P. Thomas. Make your petri nets understandable : Reliability block diagrams driven petri nets. *Reliability Engineering & System Safety*, 113 :61–75, 2013.

[190] Ni-Bin Chang. *Systems Analysis for Sustainable Engineering : Theory and Applications*. Green Manufacturing & Systems Engineering, 2010.

Milia HABIB

Doctorat : Optimisation et Sûreté des Systèmes

Année 2017

Optimisation de performances et maîtrise de la fiabilité dans la conception de systèmes de production

Cette thèse porte sur les problèmes de conception et d'optimisation de la fiabilité des systèmes avec la prise en compte de la dépendance redondante. Nous nous intéressons d'abord à la conception de systèmes réparables dépendants de type parallèle et k sur n : G. Après avoir rappelé le modèle de la dépendance redondante présenté dans la littérature pour les systèmes parallèles, nous proposons un modèle plus général pour les systèmes k sur n : G. Ce modèle permet de quantifier la dépendance de défaillance entre les composants redondants du système. Nous évaluons également la disponibilité stationnaire du système avec la prise en compte de la dépendance à l'aide des modèles markoviens. Nous étudions ensuite la conception des systèmes réparables séries k sur n en considérant la notion de dépendance redondante. Ces problèmes sont traités sous deux approches d'optimisation : mono et multicritère. Dans l'approche monocritère, nous abordons, dans un premier temps, le problème de minimisation des coûts sous contrainte d'une disponibilité exigée. Nous proposons de le résoudre en utilisant le solveur LINGO et en développant des algorithmes génétiques et des algorithmes d'optimisation par colonies de fourmis. Ces algorithmes sont ensuite améliorés par une recherche locale. Dans un deuxième temps, nous étudions le problème dual de maximisation de la disponibilité que nous le résolvons à l'aide des algorithmes génétiques et LINGO. Dans l'approche multicritère, nous considérons simultanément les deux objectifs. Nous proposons des algorithmes multiobjectifs basés sur NSGA2 et SPEA2.

Mots clés : conception - redondance (ingénierie) - fiabilité - systèmes d'aide à la décision - optimisation mathématique.

Performances Optimization and Reliability Control in the Production System Design

This thesis deals with the design and optimization problems of the reliability of the systems taking into account the redundant dependency. First, we focus on the design of repairable dependent parallel and k out of n: G systems. After recalling the redundant dependency model presented in the literature for parallel systems, we propose a more general model for the k out of n: G systems. This model allows quantifying the failure dependence between the redundant components of the system. We also evaluate the stationary system availability considering the dependence based on the Markov models. Then, we study the design of the series repairable k out of n systems considering the redundant dependency notion. These problems are treated under two optimization approaches: single and multicriteria. In the single criterion approach, we first address the minimization problem of the costs under a required availability constraint. We propose to solve it by using the LINGO solver and by developing genetic algorithms and ant colony optimization algorithms. These algorithms are then improved by a local search. In a second step, we study the dual problem of availability maximization which we solve using the genetic algorithms and LINGO. In the multicriteria approach, we simultaneously consider both objectives. We propose multi-objective algorithms based on the Non-dominated Sorting Genetic Algorithms (NSGA2) and the second version of the Strength Pareto Evolutionary Algorithm (SPEA2).

Keywords: design - redundancy (engineering) - dependability - decision support systems - mathematical optimization.

Thèse réalisée en partenariat entre :

Université Libanaise

utt UNIVERSITÉ DE TECHNOLOGIE TROYES

Ecole Doctorale "Sciences et Technologies"