REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed Khider – BISKRA

Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie

Département d'informatique

N° d'ordre:

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **Système d'Information Optimisation et Décision**

Optimisation d'une ligne de production dans un système manufacturier reconfigurable utilisant les algorithmes évolutionnaires

---

Par :

BEKIRI ZAINE EL ABIDINE

Soutenu le **** septembre 2020, en présence du jury :

		Président
TORKI Fatima	MAA	Rapporteur
		Examinateur

Remerciement

Tout d'abord je remercie Dieu le tout puissant de m'avoir aidé à atteindre mon but qui est l'élaboration de ce mémoire.

Ensuite, je remercie mes parents à qui je dois respect et obéissance et mes frères et sœurs qui m'ont soutenu.

Je remercie sincèrement mon encadreur Mme Torki Fatima pour ses précieuses aides pour sa patience et ses conseils qui m'ont aidé à réaliser le travail.

Je remercie également les jurys.

Je remercie aussi tous ceux qui ont contribué de prêt ou de loin à la réalisation de mon mémoire.

Dédicace

A mes chers parents, qui ont toujours étaient là pour m'encourager. Quoi que je fasse ou que je dise, je ne saurai vous remercier comme il se doit. Que Dieu vous protège. Et que la réussite soit de mon sort pour que je puisse vous combler de bonheur.

A mes frères et sœurs : sans leur aide morale et matérielle je ne saurais venir à bout de ce travail sans oublier mes oncles et tantes et mes amis.

Résumé

Les systèmes de production connaissent un nouveau paradigme qui est le Système Manufacturier Reconfigurable « Reconfigurable manufacturing system-RMS» qui est caractérisé par les capacités d'adaptation aux changements adéquats, permettant de régler ses structures et ses processus à la demande des différents besoins des clients. Le RMS est constitué d'un ensemble de machines ou de stations en collaboration dans le but d'exécuter un ensemble d'opérations sur une matière première dans le but d'obtenir la forme finale souhaitée du produit demandé et exigé par le client.

Le travail élaboré dans ce mémoire vise à générer des lignes de production optimales afin d'obtenir les meilleurs arrangements des machines reconfigurables de type RMTs minimisant le coût et le temps et maximisant la fiabilité. Tout cela pour obtenir les meilleures solutions en utilisant l'algorithme évolutionnaire nommé Non Dominated Sorting Genetic Algorithm-III « NSGA-III ».

Abstract

The systems of production know a new paradigm which is the Reconfigurable Manufacturing System « RMS» which is characterized by the capacities of adaptation to adequate changes, allowing to rule its structures at the request of different needs. The RMS is composed of a set of machines or of stations in collaboration in order to perform a set of operations on a raw material to get the final desired shape of the product requested and required by the customer.

The work elaborated in this thesis aims to generate optimal production flow lines to obtain the best arrangements of reconfigurable machines type RMTs minimizing cost and time and maximizing reliability. All this to obtain the best solutions using the evolutionary algorithm called Non Dominated Sorting Genetic Algorithm- III « NSGA- III ».

ملخص

عرفت انظمة الإنتاج نموذج جديد والمتمثل في أنظمة الإنتاج القابلة لإعادة التشكيل «RMSs » هذا الأخير يمكن تعريفه على أنه نظام إنتاج لديه القدرة على تسهيل التغييرات المناسبة وضبط بنياته وعملياته استجابة لاحتياجات العملاء المختلفة، ويتكون من مجموعة من الألات أو المحطات بالتعاون لتنفيذ مجموعة من العمليات على المواد الخام من أجل الحصول على الشكل النهائي المطلوب للمنتج الذي طلبه وحدده العميل.

الغرض من هذه المذكرة هو إنشاء خطوط إنتاج مثالية من أجل الحصول على أفضل الترتيبات للآلات القابلة لإعادة التشكيل من نوع RMTs و ذلك للتقليل من تكلفة و وقت الإنتاج بالإضافة الى الزيادة في موثوقية الآلات و الحصول على أفضل الحلول باستخدام الخوارزمية التطورية المسماة Non Dominated Sorting Genetic Algorithm – III «NSGA- III »

Table des matières

Introduction générale 1

Chapitre 1: Optimisation Multi-objectif

1 Introduction 4
2 Problème d'optimisation multi-objectif 4
3 Principaux concepts d'optimisation 5
4 Classification des problèmes d'optimisation multi-objectif 6
5 Notions de base 7
5.1 Multiplicité de solutions 7
5.2 La dominance 7
5.3 La dominance au sens de Pareto 7
5.4 L'optimalité globale au sens de Pareto 8
5.5 L'optimalité locale au sens de Pareto 8
5.6 Front de Pareto 8
5.7 Point Idéal et point Nadir 9
5.8 Convexité 10
6 Approches de résolution 10
6.1 Méthodes a priori 11
6.1.1 Programmation Par but 11
6.1.2 Méthode Lexicographique 11
6.2 Méthodes a posteriori 12
6.2.1 Somme pondérée 12
6.2.2 La méthode ε-Contrainte 12
6.3 Méthodes interactives 12
6.3.1 Méthode de Tchebychev 12
6.3.2 Méthodes de point de référence 13
7 Les algorithmes évolutionnaires 13
8 Les Algorithmes Génétiques 14
8.1 Les techniques Non-élitiste 16
8.1.1 Niched Pareto Genetic Algorithm (NPGA) 16
8.1.2 Multiple Objective Genetic Algorithm (MOGA) 16

8.1.3 Non dominate Sorting Genetic Algorithm (NSGA) 17
8.2 Les techniques élitistes 18
8.2.1 Strength Pareto Evolutionary Algorithm- II (SPEA- II) 18
8.2.2 Non dominated Sorting Genetic Algorithm-II (NSGA-II) 20
8.2.3 Non dominated Sorting Genetic Algorithm-III (NSGA-III) 23
9 Conclusion 25

Chapitre 2: Systemes Manufacturiers Reconfigurables (RMSs)

1 Introduction 26
2 Les systèmes manufacturiers 26
3 L'évolution des systèmes manufacturiers 27
4 Systèmes dédiés et les systèmes flexibles 27
4.1 Système Manufacturiers Dédiés (Dedicated Manufacturing Systems-DMSs) 28
4.2 Système Manufacturiers Dlexibles (Flexible Manufacturing Systems-FMSs) 28
5 RMS : Concepts et définitions 29
5.1 Les différentes caractéristiques du RMS 30
5.2 La famille de produits 31
5.3 Machines Reconfigurables (Reconfigurable Machines-RMs) 31
5.3.1 Machines-outils reconfigurables (Reconfigurable Machines Tools- RMTs) 31
5.4 Lignes de production 32
5.5 La configuration et la reconfiguration du RMS 33
5.5.1 La configuration du système 33
5.5.2 Les déclencheurs de la reconfiguration 33
5.5.3 Le processus de reconfiguration 34
5.5.4 Les types de reconfiguration 35
6 La conception des RMSs 36
6.1 Principes de conception des RMSs 36
6.2 Conception des produits et formations des familles de produits 36
6.3 La conception d'une ligne de production 36
7 Les indicateurs de performance dans les RMSs 36
8 Comparaison entre DMS, FMS, RMS 37
9 Les travaux relatifs 38
10 Conclusion 40

Chapitre 3: Conception du Système

1 Introduction 42
2 L'objectif du système 42
3 Conception du système 43
3.1 Conception Globale 43
3.1.1 Couche 1 43
3.1.2 Couche 2 44
3.1.3 Couche 3 44
3.2 Conception détaillée 44
3.2.1 Données d'entrée du système 45
3.2.1.1 Information sur le produit 45
3.2.1.2 Information sur les machines 45
3.2.1.3 Information sur le temps 46
3.2.1.4 Information sur le coût 46
3.2.2 Structures des données utilisées 46
4 Fonctions Objectifs 48
4.1 Temps total 48
4.1.1 Temps de changements des machines (MCT) 48
4.1.2 Temps de changements d'outils (TCT) 49
4.1.3 Temps de changement des configurations (CCT) 49
4.1.4 Temps de traitement (PT) 49
4.2 Coût total 49
4.2.1 Coût d'utilisation des machines(MUC) 49
4.2.2 Coût de changement de machine (MCC) 50
4.2.3 Coût de changement de configuration(CCC) 50
4.2.4 Coût d'utilisation des outils(TUC) 50
4.2.5 Coût de changement d'outils(TCC) 50
4.3 Fiabilité totale 51
4.3.1 Fiabilité de machine (MF) 51
5 Représentation de l'individu (la solution) 51
6 Codages et décodages 52

6.1 Codages de l'individu 52
6.2 Décodages de l'individu 52
6.2.1 Décodage des machines 52
6.2.2 Décodage des configurations 53
6.2.3 Décodage des outils 53
7 Adaptation de NSGA-III 54
7.1 Fonctionnement 54
7.2 Génération de la population initiale 55
7.3 Evaluation de la population 55
7.4 Fonction de tri rapide non dominée (Fast Non-dominated Sorting) 56
7.5 Mise à jour de la population 56
7.6 La sélection 56
7.7 Le croisement 57
7.8 La mutation 57
8 Conclusion 57

Chapitre 4: Implementation et Résultats

1 Introduction 61
2 Langage de programmation et outils de développement 61
2.1 Langage de programmation Python 61
2.2 L'environnement de programmation Pycharme 61
2.3 Le Package (Tkinter) Tool kit Interface 62
3 Présentation des interfaces de notre système 62
3.1 Interface principale 62
3.1.1 Bouton «Définir de nouvelles données» 63
3.1.1.1 Interface des données de machines 64
3.1.1.2 Interface des données du produit 65
3.1.1.3 Interface des données de temps 67
3.1.1.4 Interface des données de coût 69
3.1.1.5 Interface des données de faiabilité des machines 71
3.1.2 Bouton «Ouvrir les fichiers des données» 72
4 Evaluation des résultats 72
4.1 Etude de cas 73

4.2 Exécution de l'algorithme NSGA-III 79
5 Conclusion 81
Conclusion générale 82
Bibliographie 83

Liste des figures

Figure 1.1: Espace de décision et espace objectif d'un problème d'optimisation multi-objectif. 5
Figure 1.2: Schéma illustrant les différents minimaux. 6
Figure 1.3: Optimalité Locale et globale au sens de Pareto[7] 8
Figure 1.4: Front de Pareto.[7] 9
Figure 1.5: Représentation du point idéal et du point Nadir[7] 10
Figure 1.6: Exemples d'ensemble convexe et d'ensemble non convexe [10] 10
Figure 1.7: Squelette d'un algorithme évolutionnaire.[1] 14
Figure 1.8: Un croisement pour un codage binaire.[16] 15
Figure 1.9: Une mutation pour un codage binaire.[16] 15
Figure 1.10: Illustration du clustering en deux dimensions. 19
Figure 1.11: Schéma de fonctionnement de NSGA-II.[7] 21
Figure 1.12: Illustration du crowding.. 22
Figure 2.1: Schéma classique d'un système manufacturier. 26
Figure 2.2: Système Manufacturier dédié (DMS)[30]. 28
Figure 2.3: Système Mnufacturier flexible (FMS)[30] 28
Figure 2.4: Système MAnufacturier reconfigurable (RMS).[26] 29
Figure 2.5: les caractéristiques du RMS selon.[27] 30
Figure 2.6: Architecture de la famille de produits.[33] 31
Figure 2.7: Exemple d'une machines-outils reconfigurable (RMT).[34] 32
Figure 2.8: Les niveaux de catégories de fabrication. 32
Figure 2.9: Exemple illustratif du changement de configurations d'un RMS pendant sa phase opérationnelle.[30] 34
Figure 2.10: Schéma présentant les différentes phases du processus de reconfiguration. 35
Figure 2.11: Les différents aspects de reconfiguration adaptée. 35
Figure 2.12: Positionnement du RMS par rapport au DMS et FMS.[32] 38
Figure 3.1: Schéma représentant la conception fonctionnelle descendante. 42
Figure 3.2: Architecture globale du système. 43
Figure 3.3: Architecture détaillée du système. 44
Figure 3.4: Architecture des données d'entrée du système. 45
Figure 3.5:Algorithme NSGA- III adapté. 54
Figure 3.6: Croisement à un point. 57
Figure 4.1: Logo de python. 61
Figure 4.2: Logo de PyCharm. 62
Figure 4.3: Logo de TKinter. 62
Figure 4.4: Interface principale. 63
Figure 4.5: Interface des données d'entrée. 63
Figure 4.6: Interface des données de machines. 64
Figure 4.7: Interface nombre de machines. 64
Figure 4.8: Interface de la configuration TAD. 65
Figure 4.9: Interface machine/outil. 65

Figure 4.10: Interface des données de produit. 66
Figure 4.11: Interface pour définir le nombre de composants d'un produit. 66
Figure 4.12: Interface des séquences possibles. 67
Figure 4.13: Interface d'une operation TAD. 67
Figure 4.14: Interface des données de temps. 68
Figure 4.15: Interface de temps de changement d'outils. 68
Figure 4.16: Interface de temps de changement des machines. 68
Figure 4.17: Interface de temps de changement de configuration. 69
Figure 4.18: Interface des données de coût. 69
Figure 4.19: Interface de coût d'utilisation d'une machine. 70
Figure 4.20: Interface de coût de d'utilisation d'outils. 70
Figure 4.21: Interface de coût de changement des machines. 70
Figure 4.22: Interface de coût de changement d'outils. 71
Figure 4.23: Interface de coût de reconfiguration d'une machine. 71
Figure 4.24: Interface fiabilité de machines. 71
Figure 4.25: Interface du gestionnaire des données. 72
Figure 4.26: Interface du paramétrage de NSGA-III. 72
Figure 4.27: Courbe de l'algorithme NSGA-III. 80

Liste des tableaux

Tableau 2.1: Une simple comparaison entre les DMSs et FMSs [30]l. 29
Tableau 2.2: Comparaison entre les DML, FMS et RMS [29]. 37
Tableau 2.3: Comparaison entre notre travail et les travaux connexes 40
Tableau 3.1: Structure de données utilisée 48
Tableau 3.2: Individu sous forme d'une matrice 51
Tableau 3.3: Individu codé en nombre réel 52
Tableau 4.1: Configurations TADs et outils disponibles 73
Tableau 4.2: Séquence d'opérations à réaliser 73
Tableau 4.3: Opérations TAD et outils requis. 74
Tableau 4.4: Temps de traitement 75
Tableau 4.5: Temps de changement d'outils 75
Tableau 4.6: Temps de reconfiguration 76
Tableau 4.7: Temps de changement des machines 76
Tableau 4.8: Coût d'utilisation des machines 76
Tableau 4.9: Coût d'utilisation d'outils 77
Tableau 4.10: Coût de changement des machines 77
Tableau 4.11: Coût de changement des outils 77
Tableau 4.12: Coût de reconfiguration des machines 78
Tableau 4.13: Fiabilité des machines 78
Tableau 4.14: Valeurs du coût total, temps total et fiabilité pour chaque solution 26

Liste des abréviations

AMOSA: Archive Multi-objective Simulated Annealing
CNC: Computer Numerically Controlled machine
DMS: Dedicated Manufacturing Systems
FMS: Flexible Manufacturing System
MOGA: Multi-Objective Genetic Algorithm
NSGA- III: Non-dominated Sorting Genetic Algorithm - III
RMS: Reconfigurable Manufacturing Systems
RMT: Reconfigurable Machine Tool (Machine-outil reconfigurable)
TAD: Tool Approach Direction
TOPSIS: Technique for Order of Preference by Similarity to Ideal Solution

Introduction générale

Contexte

Dans le secteur industriel, la production consiste à transformer les matières premières en produits finis, grâce à des systèmes manufacturiers hautement sophistiqués. Un système manufacturier se compose de l'ensemble des éléments matériels et immatériels nécessaires à la production de produits ayant la forme finale désirée du produit exigé et spécifié par le client. De nos jours, beaucoup d'entreprises pensent à rénover leur pratique de production et leurs philosophies en vue d'une concurrence croissante et continue des systèmes manufacturiers de plus en plus complexes, des services, des produits et des technologies de durées de vies très courtes. Les industriels affirment que le développement de nouvelles approches et outils d'aide à la décision est un besoin de première nécessité pour la bonne décision au bon moment et pour le bon problème.

Avant, il n'existait que deux types principaux de systèmes manufacturiers : les Systèmes de Production Dédiés (Dedicated Manufacturing Systems– DMSs) et les Systèmes de Production Flexibles (Flexible Manufacturing Systems – FMSs). Les Systèmes de Production Dédiés permettent de fabriquer des produits spécifiques en gros volumes mais le principal inconvénient des DMSs est que ces lignes ne sont pas conçues pour être modifiables. Par conséquent, elles ne peuvent pas être converties facilement pour produire de nouveaux produits. Les Systèmes de Production Flexibles mais peuvent produire une variété de produits avec un volume variable sur le même système de production. Dans un FMS, il est difficile d'ajouter une nouvelle machine ou de changer une machine. Autrement dit, la structure du FMS est souvent figée. Plus tard, un nouveau type apparut nommé les Systèmes Manufacturiers Reconfigurables (Reconfigurable Manufacturing Systems - RMS), découvert par le grand pionnier Koren [1] où sa fonction est de combiner le haut débit des DMSs avec la grande flexibilité des FMSs tout en garantissant un haut niveau de réactivité face aux changements. Un RMS est un système manufacturier où les machines, les composants des machines ainsi que le système de manutention peuvent être ajoutés, modifiés, supprimés ou échangés selon les besoins de la production et son principal composant est la machine –outils reconfigurable (Reconfigurable Machine Tool– RMT). Une RMT est une machine possédant une structure qui permet sa reconfiguration pour fournir une fonctionnalité alternative.

La ligne de production est l'affectation de machine, de configuration et d'outil à chaque opération de chaque composant du produit. La capacité des machines de changer de configurations est la principale caractéristique du RMS. D'où une grande nécessité à développer des approches qui prennent en compte cette reconfigurabilité des machines dans la génération des lignes de production. Un problème peut avoir plusieurs objectifs et chaque objectif est modélisé à l'aide d'une fonction mathématique suivant les exigences des clients.

Le processus de résolution de ce problème se heurte à un problème de prise de décision, car il recherche simultanément une solution optimale ou un ensemble de solutions approchées pour chaque fonction objectif et ces fonctions peuvent être en conflit. Pour obtenir la décision optimale, certains compromis entre les objets en conflit sont nécessaires. Ce type de problèmes s'appelle des problèmes d'optimisation multi-objectifs.

La méthode de résolution de ce problème fait face à un problème de décision. Il essaie de rechercher simultanément une solution optimale ou un ensemble de solutions approchées pour chaque fonction objectif. Ces dernières peuvent être en conflit. Ce type de problèmes s'appelle des problèmes d'optimisation multi-objectifs.

Objectif

Notre projet est basée sur les travaux [2], [3]. Donc, notre objectif sert à générer des lignes de production optimale dans un système manufacturier reconfigurable pour appliquer le classement approprié des machines reconfigurables de type RMTs, minimisant le coût, le temps et maximisant la fiabilité. Pour obtenir les solutions optimales (c'est à dire le meilleur Pareto front), on va utiliser et adapter l'algorithme évolutionnaire nommé Non dominated Sorting Genetic Algorithme- III (NSGA-III).

Organisation du mémoire

Ce manuscrit est organisé quatre chapitre à savoir :

Chapitre 1: Le premier chapitre présente le problème d'optimisation multi-objectif, les différentes méthodes d'optimisation et les différentes versions d'algorithmes évolutionnaires.

Chapitre 2: Le second chapitre sera une description de différents types de systèmes manufacturiers, le système manufacturier reconfigurable, de ses caractéristiques, de ses composants et les travaux connexes.

Chapitre 3: Le troisième chapitre sera une description de notre modèle proposé représenté en deux niveaux (la conception globale et détaillé).

Chapitre 4: Le quatrième chapitre est destiné à l'évaluation du modèle proposé en conception. Nous commençons par la description des outils utilisés, ainsi que les expérimentations réalisées et les résultats obtenus.

Conclusion générale: Le présent manuscrit terminera par une conclusion générale et des perspectives attendues.

CHAPITRE 1

Optimisation Multi-objectif

1 Introduction

Au début des années 1990, les méthodes des algorithmes génétiques multi-objectif ont commencé à prendre de l'envergure d'où un intérêt grandissant de plus en plus. Ces dernières sont bien conçues de façon à répondre au problème multi-objectif où on recherche un ensemble de solutions. Les Algorithmes génétiques sont caractérisés par le travail sur une population de solutions, ce qui nous permet de trouver plusieurs solutions potentiellement Pareto optimales. En effet la proche des algorithmes génétiques a la possibilité de résoudre le problème de contrôle multi-objectif. Dans ce chapitre, nous abordons les principaux concepts d'un algorithme génétique multi-objectif.

2 Problème d'optimisation multi-objectif

On peut définir un problème d'optimisation comme la recherche de l'optimum d'une fonction donnée (recherche du minimum ou du maximum). Les problèmes réels déclenchent souvent de multiples mesures de performance ou objectifs, qui doivent être optimisés simultanément. Les objectifs peuvent être conflictuels, ce qui rend l'optimisation impossible à réaliser en pratique. Ces derniers mesurent différents aspects de la qualité de la solution. Dans ce cas, la qualité d'un individu est décrite non pas par un scalaire mais par un vecteur. La performance, la fiabilité et le coût sont des exemples d'objectifs conflictuels [4].

Les objectifs peuvent être conflictuels, ce qui rend l'optimisation impossible à réaliser en pratique

Mathématiquement parlant, un problème d'optimisation se présentera sous la forme suivante Selon [5]:

$$\begin{aligned} &\text{Minimiser} \quad \vec{f}(\vec{x}) (\text{k: fonction à optimiser}) \\ &\text{Avec} \quad \vec{g}(\vec{x}) \leq 0 \ \left(m : \text{contraintes d'inégalité}\right) \\ &\text{Et} \quad \vec{h}(\vec{x}) = 0 \ (p \text{ contraintes d'égalité}) \\ &\text{Ou } \vec{x} \in R^n, \vec{f}(\vec{x}) \in R^K, \vec{g}(\vec{x}) \in R^m \text{ et } \vec{h}(\vec{x}) \in R^p \end{aligned}$$

Ici, les vecteurs, $\vec{g}(\vec{x})$ et $\vec{h}(\vec{x})$ représentent respectivement m contraintes d'inégalité et p contraintes d'égalité. Cet ensemble de contraintes délimite un espace restreint de recherche de la solution optimale. Le $\vec{f}(\vec{x})$ vecteur regroupe k fonctions objectif ou :

$$\vec{f}(\vec{x}) = (f_0(\vec{x}), f_1(\vec{x}), \ldots, f_k(\vec{x}))$$

Tous les énoncés et définitions seront donnés dans le cadre de problèmes de minimisation.

La résolution d'un problème d'optimisation multi-objectif a pour seul but de minimiser «au mieux » les différents objectifs. Dans un problème d'optimisation multicritère, on se heurte fréquemment à des objectifs contradictoires. Deux objectifs sont contradictoires lorsque la diminution d'un objectif entraîne une augmentation de l'autre objectif [5],[6].

En effet un problème de maximisation peut être aisément transformé en un problème de minimisation en considérant l'équivalence suivante :

$$\text{Maximiser } \vec{f}(\vec{x}) \leftrightarrow \text{Minimiser } -\vec{f}(\vec{x})$$

3 Principaux concepts d'optimisation

Tout d'abord nous définissons les notions communes à n'importe quelle méthode d'optimisation multi-objectif [5] :

- **Fonction objectif** : C'est le nom donné à la fonction f (on l'appelle encore fonction de coût ou critère d'optimisation). C'est cette fonction que l'algorithme d'optimisation va devoir "optimiser" (trouver un optimum).
- **Variables de décision** : Elles sont regroupées dans le vecteur $\vec{x}$. C'est en faisant varier ce vecteur que l'on recherche un optimum de la fonction *f*.
- **Vecteur de décision** : Un vecteur de décision est un vecteur correspondant à l'ensemble des variables du problème, il est noté : $\vec{x} = x[x_1, x_2, \ldots\ldots, x_n]^T$ avec : n le nombre de variables ou dimension du problème et x_k la variable sur la dimension K.
- **Critère de décision :** Est un critère sur lequel sont jugés les vecteurs de décision pour déterminer le meilleur vecteur. Un critère peut être une variable du problème ou une combinaison de variables.
- **Espace de recherche :** Représentant l'ensemble des valeurs pouvant être prises par les variables.
- **Espace réalisable :** Représentant l'ensemble des valeurs des variables satisfaisant les contraintes.

La Figure 1.1 présente un exemple avec deux variables de décision et deux fonctions objectifs. Dans cette figure, on peut voir la façon dont la fonction F projette les solutions de l'ensemble réalisable, X, dans l'ensemble réalisable, Z, de l'espace de la fonction objectif.

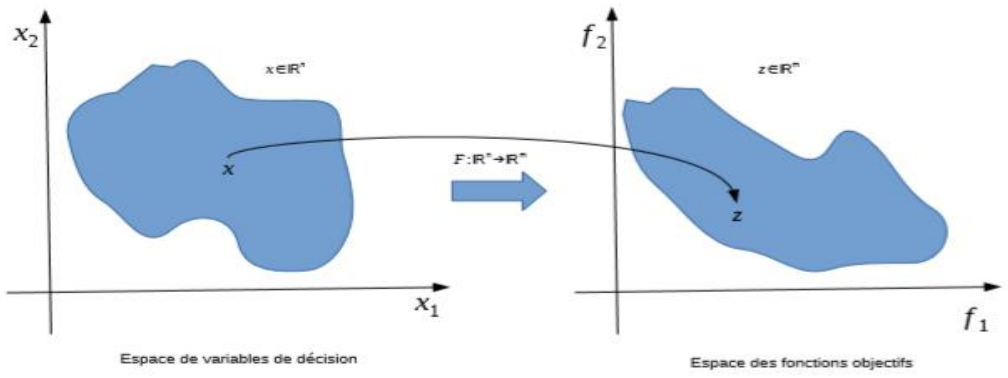


Figure 1.1: Espace de décision et espace objectif d'un problème d'optimisation multi-objectif [7].

- **Minimum global :** Un "point"$\vec{x}^*$est un minimum global de la fonction f si on a : f ($\vec{x}^*$) < f ($\vec{x}$) quel que soit $\vec{x} \rightarrow$ tel que $\vec{x} \neq \vec{x}^*$. Cette définition correspond au point M_3 de la Figure 1.2.
- **Minimum local fort :** Un "point"$\vec{x}^*$ est un minimum local fort de la fonction f si on a : f ($\vec{x}^*$) < f ($\vec{x}$) quel que soit $\vec{x} \in$ V ($\vec{x}^*$) et $\vec{x}^* \neq \vec{x}$, où V ($\vec{x}^*$) définit un "voisinage" de $\vec{x}^*$. Cette définition correspond aux points M_2 et M_4 de la Figure2.
- **Minimum local faible :** Un "point"$\vec{x}^*$ est un minimum local faible de la fonction f si on a : f ($\vec{x}^*$) ≤ f ($\vec{x}$) quel que soit$\vec{x}\in$V ($\vec{x}^*$) et $\vec{x}^* \neq \vec{x}$, où V ($\vec{x}^*$) définit un "voisinage" de $\vec{x}^*$. Cette définition correspond au point M_1 de la Figure 1.2.

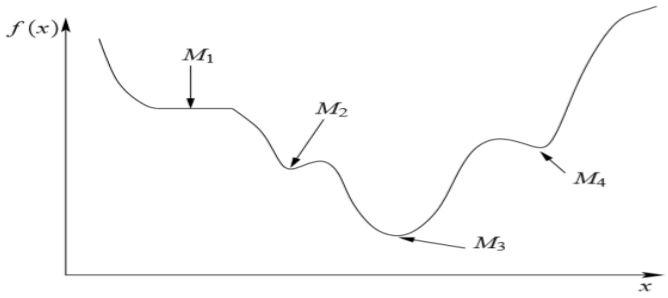


Figure 1.2: Schéma illustrant les différents minimaux.

4 Classification des problèmes d'optimisation multi-objectif

On peut classer les différents problèmes d'optimisation en fonction de leurs caractéristiques [5]:

1. Nombre de variables de décision :
   - Une ⇒ mono variable.
   - Plusieurs ⇒multi variable.
2. Type de la variable de décision :
   - Nombre réel continu ⇒continu.
   - Nombre entier⇒ entier ou discret.
   - Permutation sur un ensemble fini de nombres⇒ combinatoire.
3. Type de la fonction objectif :
   - Fonction linéaire des variables de décision⇒ linéaire.
   - Fonction quadratique des variables de décision⇒ quadratique.
   - Fonction non linéaire des variables de décision⇒ non linéaire.
4. Formulation du problème :
   - Avec des contraintes⇒ contraint.
   - Sans contraintes ⇒non contraint.

5 Notions de base

5.1 Multiplicité de solutions

La résolution d'un PMO présente plusieurs solutions possibles. En conséquence il en résulte souvent des objectifs contradictoires (la diminution d'un objectif entraîne l'augmentation de l'autre). Si on prend l'exemple de dimensionnement d'une poutre devant supporter une charge donnée, on voudra obtenir une poutre de section la plus petite possible, produisant la plus petite déformation possible, lorsque la charge repose au milieu de la poutre [8].

L'ensemble des solutions fournies par la résolution d'un PMO n'est pas optimal, vu qu'elles ne minimisent pas toutes les fonctions objectives. Ce sont des solutions de compromis.

5.2 La dominance

Pour détecter les meilleurs compromis, il est nécessaire d'établir une relation d'ordre entre ces éléments. Dans le cas des PMO, cette relation est appelée relation de dominance. Cette dernière nous permet donc de limiter l'ensemble des solutions de compromis. Il existe différents types de relations de dominance, qui facilite suffisamment la liberté du choix de la relation qui reproduit au mieux le comportement d'un décideur [9].

La plus célèbre et la plus utilisée est la dominance au sens de Pareto. Elle est définit par :

Le vecteur $\vec{u}$ domine le vecteur $\vec{v}$ (on note : $\vec{u} \prec \vec{v}$) si :

$\vec{u}$ est au moins aussi bon que $\vec{v}$ dans tous les objectifs, et, $\vec{u}$ rest strictement meilleur que $\vec{v}$ dans au moins un objectif.

5.3 La dominance au sens de Pareto

Afin de définir formellement la notion de dominance au sens de Pareto, les relations =, $\leq$ et < usuelles sont étendues aux vecteurs [7].

Soient $\vec{u}$ et $\vec{v}$ deux vecteurs de même dimension, $\vec{u} = (u_1, u_2, \ldots\ldots, u_n)$ et $\vec{v}(v_1, v_2, \ldots\ldots, v_n)$

$$\vec{u} = \vec{v} \quad si\ et\ seulement\ si \quad \forall i \in \{1,2, \ldots\ldots, m\}\ u_i = v_i$$

$$\vec{u} \leq \vec{v} \quad si\ et\ seulement\ si \quad \forall i \in \{1,2, \ldots\ldots, m\}\ u_i \leq v_i$$

$$\vec{u} > \vec{v} \quad si\ et\ seulement\ si \quad \vec{u} \geq \vec{v} \wedge \vec{u} \neq \vec{v}$$

Les relations > et ≥ sont définies de manière analogue.

On remarque que les relations qu'on vient de définir ne couvrent pas tous les cas possibles. Par exemple les points (ici vecteurs) $\vec{u} = (3,5)\ et\ \vec{v} = (6,1)$ sont incomparables à l'aide de ces relations. Nous définissons maintenant la relation de dominance au sens de Pareto qui permet de comparer tous les vecteurs de décision possible [7].

Soient $\vec{u}$ et $\vec{v}$ deux vecteurs de décision, $\vec{u} = (u_1, u_2, \ldots\ldots, u_n)$ et $\vec{v} = (v_1, v_2, \ldots\ldots, v_n)$

$$\vec{u} \prec \vec{v}\ (\vec{u}\ domine\ \vec{v}\ ) \quad si\ et\ seulement\ si\ \vec{f}(\vec{u}) < \vec{f}(\vec{v})$$

$$\vec{u} \prec \vec{v}\ (\vec{u}\ domine\ faiblement\ \vec{v}\ ) \quad si\ et\ seulement\ si\ \vec{f}(\vec{u}) \leq \vec{f}(\vec{v})$$

$$\vec{u} \sim \vec{v}\ (\vec{u}\ et\ \vec{v}\ \ incomparables\ ou\ non\ dominés) \quad si\ et\ seulement\ si :$$

$$\neg(\vec{f}(\vec{u}) \leq \vec{f}(\vec{v})) \wedge \neg(\vec{f}(\vec{v}) \leq \vec{f}(\vec{u}))$$

5.4 L'optimalité globale au sens de Pareto

Soit F l'image de l'ensemble réalisable X dans l'espace des objectifs. Un vecteur de décision $\vec{u} \in X$ est dit Pareto globalement optimal si et seulement si [10] :

$$\nexists\ \vec{v} \in X,\ \vec{v} \prec \vec{u}.$$

Dans ce cas $\vec{f}(\vec{u}) \in F$ est appelé : solution efficace

5.5 L'optimalité locale au sens de Pareto

Un vecteur de décision $\vec{u} \in X$ est dit Pareto globalement optimal si et seulement si, pour un d >0 fixé [10] :

$$\nexists\ \vec{v} \in X,\ \vec{f}(\vec{v}) \in B\left(\vec{f}(\vec{u}), d\right) et\ \ \vec{v} \prec \vec{u}\ ,$$

Ou $B\left(\vec{f}(\vec{u}), d\right)$ représente une boule de centre $\vec{f}(\vec{u})$ et de rayon d.

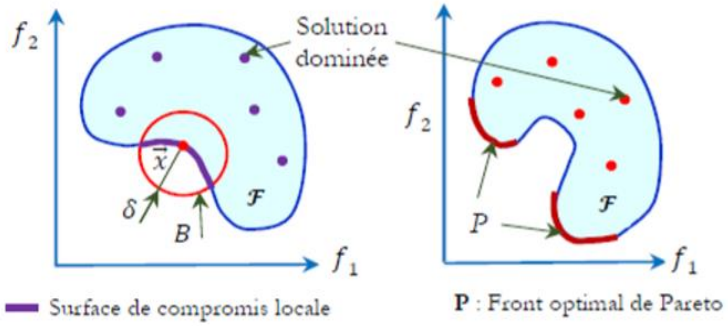


Figure 1.3: Optimalité Locale et globale au sens de Pareto [10].

5.6 Front de Pareto

Soit F l'image de l'ensemble réalisable X dans l'espace des objectifs. Le Front de Pareto $ND(F)\ de\ F$ est défini comme suit [10] :

$$ND(F) = \{\vec{v} \in F | \neg\exists \vec{u} \in F, \vec{u} < \vec{v}\}$$

Le Front de Pareto est aussi appelé l'ensemble des solutions efficaces ou la surface de compromis.

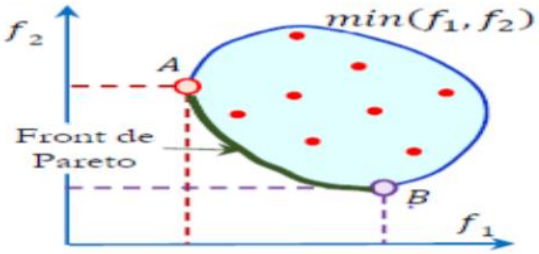


Figure 1.4: Front de Pareto [10].

Le facteur de dominance est déterminé via une assignation de rang ou tri par front. Chaque rang ou front correspond à un niveau de dominance. Ainsi, les solutions de rang 1 (front 1) correspondent à des solutions non-dominées et les solutions de rang 2 sont des solutions dominées seulement par des solutions du front 1. Il en est de même pour les fronts suivants [11].

Algorithme d'Assignation du rang de Pareto[11]

N: Nombre de points de l'ensemble sur lequel on effectue les comparaisons
x_i : Un individu
t : La génération t
$Rang(x_i, t)$ Numéro de la surface de compromis auquel est affecté l'individu X_i à la génération t
RangCourant = 1
$m = N$
While $N \neq 0$ **do**
For $i = 1$ **to** m **do**
If x_i est non dominé
$Rang(x_i, t)$ = RangCourant
End for i
For $i = 1$ **to** m **do**
If $Rang(x_i, t)$ = RangCourant
Ranger x_i dans la population temporaire
$N = N - 1$
End for i
RangCourant = RangCourant +1
$m = N$
End while

5.7 Point Idéal et point Nadir

- **Point Idéal :** Les coordonnées du point idéal (Pi) correspondent aux meilleures valeurs de chaque objectif des points du front de Pareto. les coordonnées de ce point correspondent aussi aux valeurs obtenues en optimisant chaque fonction objective séparément [10].
$$Pi_i = min\{y_i | \vec{y} \in ND(F)\}$$
- **Point Nadir :** Les coordonnées du point Nadir (Pn) correspondent aux pires valeurs de chaque objectif des points du front de Pareto [10].

$$Pn_i = max\{y_i | \vec{y} \in ND(F)\}$$

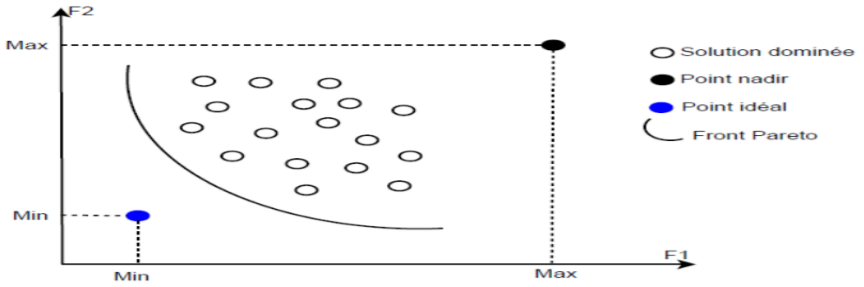


Figure 1.5: Représentation du point idéal et du point Nadir [10].

5.8 Convexité

Parmi les méthodes d'optimisation multi-objectif, certaines demandent le respect de certaines hypothèses. La méthode demande fréquemment le travail sur un espace S des valeurs de $\vec{f}$ qui sont convexes. Un ensemble S est convexe si le segment qui relie deux points quelconques de cet ensemble est contenu dans l'ensemble S. La figure 6 ci-dessous représente un ensemble convexe et d'ensemble non convexe [10].

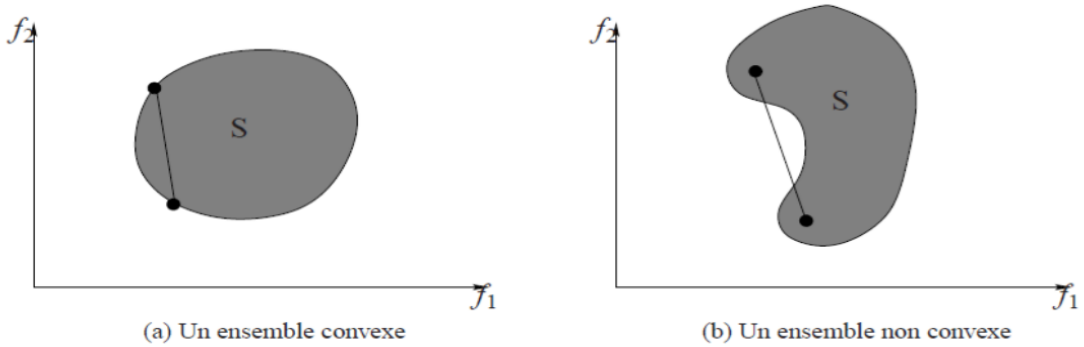


Figure 1.6 : Exemples d'ensemble convexe et d'ensemble non convexe [10].

6 Approches de résolution

La solution d'un problème multi-objectif se compose d'un ensemble de solutions. En effet, pour un problème réel, une seule solution peut être exploitée. Le décideur doit donc faire son choix. Le décideur peut intervenir en amont de la résolution, après celle-ci, ou de manière interactive[12]:

- **Préférence a priori :** le décideur définit ses préférences entre les différents objectifs avant d'utiliser la méthode d'optimisation
- **Préférence progressive :** le décideur affine son choix de compromis au fur et à mesure du déroulement de la méthode d'optimisation.
- **Préférence a posteriori :** le décideur choisit la solution de son choix parmi l'ensemble des solutions fournies par la méthode d'optimisation.

6.1 Méthodes a priori

6.1.1 Programmation Par but

Dans cette méthode, seul le décideur s'engage à fixer le but T_i à atteindre pour chaque objectif f_i. Ces valeurs, considérées comme des contraintes supplémentaires, sont ajoutées au problème. La nouvelle fonction objectif est modifiée de façon à minimiser la somme des écarts entre les résultats et les buts à atteindre :

$$min \sum |f_i(\vec{x}) - T_i|$$

T_i: Représente la valeur à atteindre pour le i^{eme} objectif.

La méthode est facile à mettre en œuvre mais l'efficacité de la méthode dépend de la définition des poids et des objectifs à attendre. Cette méthode à l'avantage de fournir un résultat même si un mauvais choix initial a conduit le décideur a donné un ou plusieurs but(s) T_i non réalisable(s)[13].

6.1.2 Méthode Lexicographique

Forman a suggéré une méthode dans laquelle les objectifs sont d'avance classés par ordre d'importance par le décideur. Ensuite, l'optimum est obtenu en minimisant tout d'abord la fonction objectif la plus importante puis la deuxième et ainsi de suite.

Soient les fonctions objectifs $f_i \; avec \; i = 0, \dots, k$, supposons un ordre tel que [10] :

$$f_1 \succ f_2 \succ \cdots \succ f_k$$

Faut : $\begin{cases} minf_1(x) \\ avecg_j(x) \; satisfait : j = 1, \dots, m \end{cases}$

Soit x_1^*, la meilleure solution trouvée avec : $f_1^* = f_1(x_1^*)$ $f_1$devient alors une nouvelle contrainte.

L'expression du nouveau problème est donc :

$$\begin{cases} minf_2(x) \\ avec \; g_j(x) \; satisfait : j = 1, \dots, m \; et \; f_1(x) = f_1^*, f_2(x) = f_2^*, \dots, f_{i-1}(x) = f_{i-1}^* \end{cases}$$

Soit x_2^* la solution de ce problème, Le i^{eme} problème sera le suivant :

$$\begin{cases} minf_i(x) \\ avec \; g_j(x) \; satisfait : j = 1, \dots, m \; et \; f_1(x) = f_1^*, f_2(x) = f_2^*, \dots, f_{i-1}(x) = f_{i-1}^* \end{cases}$$

La procédure est répétée jusqu'à ce que tous les objectifs soient traités. La solution du problème sera la solution obtenue à l'étape k. Vu la grande importance accordée aux objectif classés en premier, cette importance représente un inconvénient majeur de cette méthode. La meilleure solution f_1^* trouvée pour l'objectif le plus important va faire converger l'algorithme vers une zone restreinte de l'espace d'état et enfermer les points dans une niche (ensemble d'individus situés dans un espace restreint) [13],[7].

6.2 Méthodes a posteriori

6.2.1 Somme pondérée

Dans cette méthode, le principe général est d'abord de faire associer un coefficient de poids pour chaque fonction objective et ensuite de minimiser la somme pondérée de ces objectifs. En conséquence, le problème multi-objectif est changé en un problème mono-objectif. Si les coefficients de poids sont positifs pour tous les objectifs, ou si le problème a une solution unique, nous pouvons avoir une solution optimale au sens de Pareto. La méthode de la somme linéaire des poids peut générer des solutions Pareto optimales quelconque pour un problème d'optimisation multi-objectif convexe (c.-à-d. si toutes les fonctions objectifs et l'espace réalisable sont convexes)[7].

$$min\vec{f} = \sum_{i=0}^{m} w_i f_i(\vec{x}) \qquad w_i \in [0,1]$$

6.2.2 La méthode ε-Contrainte

Ou méthode du compromis. Elle transforme un problème d'optimisation multi-objectif en un problème d'optimisation mono-objectif de la façon suivante [14]:

- Choisir un objectif à optimiser prioritairement.
- Choisir un vecteur de contraintes initiales.
- Transformer le problème en gardant l'objectif prioritaire et en transformant les autres objectifs en contraintes d'inégalités comme suit :

$$\begin{cases} min\ f_1(x) \\ tel\ que\ f_2(x) \leq \varepsilon_2, \ldots., f_m(x) \leq \varepsilon_m \quad x \in X \end{cases}$$

Cette approche a l'avantage par rapport à la précédente dans les problèmes non convexes, mais présente plusieurs inconvénients à savoir :

- La formulation des préférences utilisateur est délicate et nécessite une connaissance approfondie du problème de départ.
- Les contraintes rajoutées compliquent la résolution du problème.

6.3 Méthodes interactives

6.3.1 Méthode de Tchebychev

La mesure de Chebyshev permet de résoudre le problème pondéré pour lequel la fonction objective à minimiser est [7]:

$$Argmin\{max_m(w_m|f_m(x) - z_m|)\}$$

Où les w_m sont les poids de pondération des M objectifs, et z est le vecteur idéal qui représente les bornes inférieures de chaque objectif dans l'espace faisable. Tout comme la méthode de ε-contraintes, cette méthode permet d'accéder aux parties non-convexes du front de Pareto. En effet avec cette méthode, la fonction objective est non différentiable, ce qui présente un problème lors de l'utilisation des méthodes de gradient. De plus, cette méthode requiert l'optimisation de chaque objectif séparément afin de définir le vecteur idéal.

6.3.2 Méthodes de point de référence

D'après Wierzbicki l'approche du point de référence est une technique d'optimisation multi-objectif interactive basée sur la définition d'une fonction de scalarisation à atteindre. L'idée principale de cette technique est comme suit :

D'abord, le décideur est invité à donner un point de référence. Ensuite, les solutions répondant mieux aux niveaux d'aspirations calculés en utilisant la fonction de scalarisation. Si le décideur est satisfait de la solution actuelle, le processus interactif se termine. Sinon, le décideur doit fournir un autre point de référence [7].

7 Les algorithmes évolutionnaires

Un nombre important d'algorithmes dans le domaine de l'optimisation combinatoire a été inspire par Les phénomènes physiques et biologiques. Ces nombreux algorithmes bio-inspirés ont prouvé leur robustesse face à des problèmes complexes. Les algorithmes évolutionnaires sont l'un des exemples d'algorithmes bio-inspirés qui ont la capacité de résoudre des problèmes NP-Difficiles. Les algorithmes évolutionnaires sont basés sur la théorie de l´évolution proposée par Darwin en 1807. L'apparition des espèces adaptées est une conséquence de deux phénomènes principaux [15] :

1) La sélection naturelle (les individus les plus adaptés survivent et se reproduisent).
2) Le matériel génétique des espèces est sujet à des variations qui peuvent se produire. L´évolution se traduit par un processus itératif de recherche de l'optimum dans l'espace de recherche en termes d'optimisation.

D'après M. Yagoubi, on peut dire qu'il existe une multitude de variantes au sein des algorithmes évolutionnaires, qui se sont développées de manière indépendante : La Programmation Evolutionnaire (EP) parue dans les années 60, les Algorithmes Génétiques (GA) introduits par J. Holland en 1975, ces derniers ont été promus par son élève D. E. Goldberg dix ans plus tard (1989), Les Stratégies d'Evolution (SE), et finalement la programmation génétique proposée par J.Koza [4].

Soit F une fonction à optimiser (minimiser ou maximiser), définie sur un espace Ω . Dans un vocabulaire d'EA on notera :

- La fonction F est appelée fonction de fitness ou fonction d'adaptation. Chaque individu dispose de sa propre valeur de fitness ou d'adaptation.
- Les points de l'espace de recherche sont appelés individus et peuvent être représentés par un ou plusieurs chromosomes.
- Un chromosome représente une solution ; il peut être codé par des valeurs binaires, réelles,. . .
- Un ensemble P d'individus est appelé population.
- Chaque itération de l'algorithme produit un nouvel ensemble de solutions appelé nouvelle génération. Les individus de la Nième génération sont appelés parents et ceux de $la\ (N + 1)i^{eme}$ génération sont appelés enfants [4].

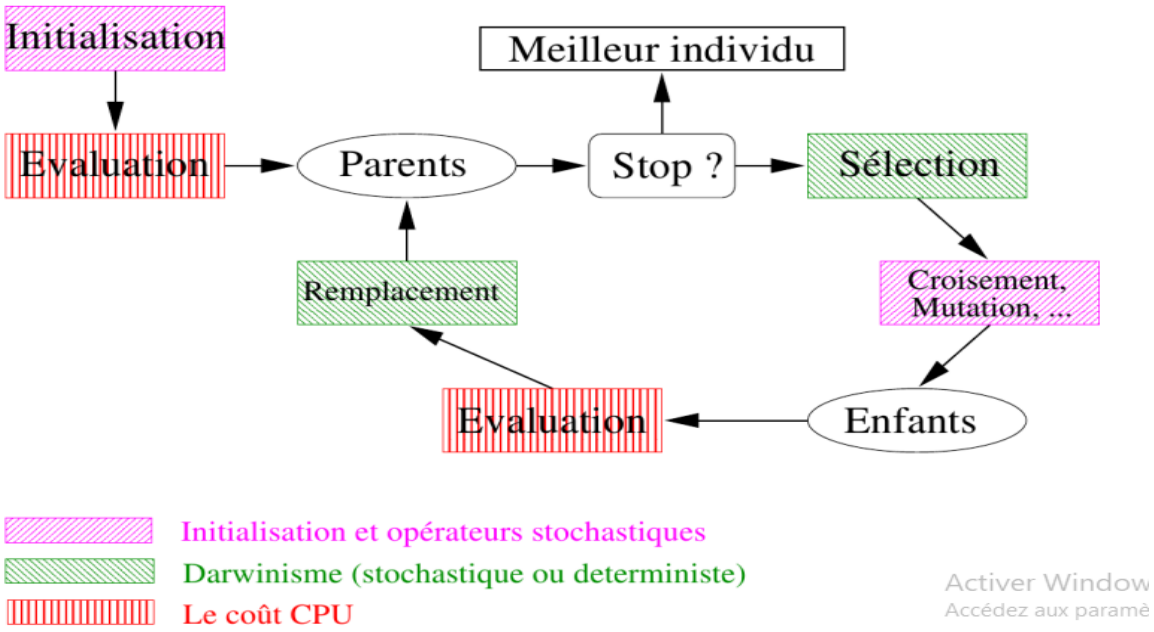


Figure 1.7: Squelette d'un algorithme évolutionnaire [4].

8 Les Algorithmes Génétiques

Les algorithmes génétiques ont pour base la théorie de l'évolution et les règles de la génétique qui démontrent l'habilité des espèces vivantes à s'adapter à leur environnement par la combinaison des mécanismes suivants [16],[17],[18] :

- la sélection naturelle montre que les individus les mieux adaptés à l'environnement ont tendance à survivre plus longtemps et ont donc une plus grande probabilité de se reproduire ;
- la reproduction par croisement fait qu'un individu hérite ses caractéristiques de ses parents, de sorte que le croisement de deux individus bien adaptés à leur environnement aura tendance à créer un nouvel individu bien adapté à l'environnement ;
- la mutation fait que certaines caractéristiques peuvent apparaître ou disparaître de façon aléatoire, permettant ainsi d'introduire de nouvelles capacités d'adaptation à l'environnement, capacités qui pourront se propager grâce aux mécanismes de sélection et de croisement.

Les algorithmes génétiques reprennent ces mécanismes pour définir une méta-heuristique. L'idée est de faire évoluer une population de combinaisons, par sélection, croisement et mutation, la capacité d'adaptation d'une combinaison étant ici évaluée par la fonction objectif à optimiser. L'algorithme 1 décrit ce principe général, dont les principales étapes sont détaillées ci-après.

Algorithme génétique

Initialiser la population avec un ensemble de combinaisons de E
While critères d'arrêt non atteints **do**
Sélectionner des combinaisons de la population
Créer de nouvelles combinaisons par recombinaison et mutation
Mettre à jour la population
End while
return la meilleure combinaison ayant appartenu à la population

- **Sélection :** Cette phase a pour but de sélectionner les combinaisons de la population qui seront ensuite élues pour la recombinaison et la mutation. Il s'agit là de conserver la sélection des meilleures combinaisons, tout en accordant une petite chance aux moins bonnes. Il y a une multitude de méthodes de procéder à cette phase de sélection. [2]
- **Recombinaison (croisement) :** Cette phase consiste à créer de nouveaux individus par un mélange de combinaisons sélectionnées. Le but est de mener la recherche dans une zone nouvelle de l'espace où de meilleures combinaisons peuvent être présentes. Finalement, la recombinaison doit être bien adaptée à la fonction à optimiser et doit être bien capable de transmettre de bonnes propriétés des parents vers la combinaison créée.
  Le principe est de générer au hasard un entier a et le premier enfant va avoir les a premiers gènes du parent1 et les L– a derniers gènes du parent2, et vice versa pour le deuxième enfant.[2]

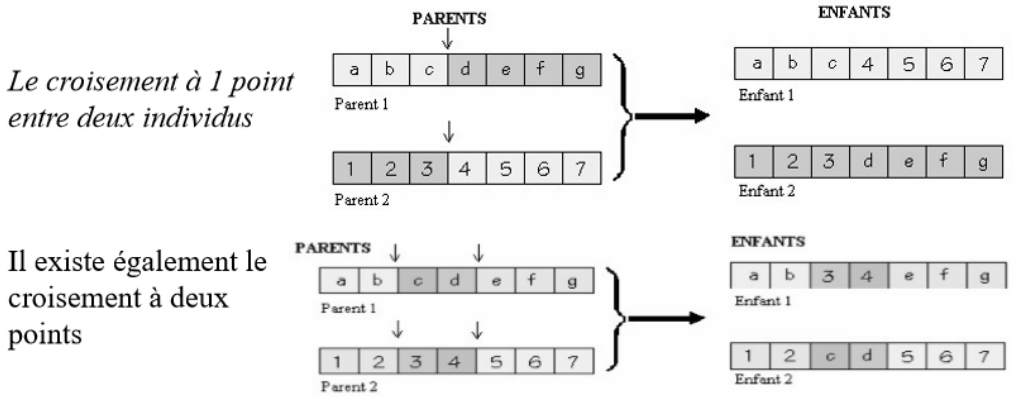


Figure 1.8: Un croisement pour un codage binaire [2].

- **Mutation :** Elle consiste à changer d'une manière aléatoire quelques composants des combinaisons obtenues par croisement.[2]

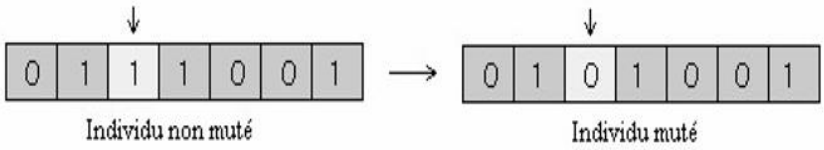


Figure 1.9: Une mutation pour un codage binaire [2].

- **Mise à jour de la population :** Cette phase détermine les nouvelles combinaisons qui doivent devenir membre de la population et les anciennes combinaisons de la population qui doivent être remplacées [19].

  L'objectif de la mise-à-jour est principalement de garder une diversité appropriée de la population, d'éviter une convergence prématurée du processus de recherche, et de donner à l'algorithme l'occasion de découvrir de nouvelles zones prometteuses dans l'espace de recherche. En effet, les décisions sont généralement prises selon des critères liés à la qualité et à la diversité [19].
- **Critères d'arrêt :** Le processus d'évolution est répété, de génération en génération, jusqu'à ce qu'on obtienne un critère d'arrêt. Il peut s'agir d'un nombre maximum de générations, un nombre maximum d'évaluations, un nombre maximum de générations sans améliorer la meilleure solution, une qualité de solution atteinte ou encore une diversité de population inférieure à un seuil donné [19].

8.1 Les techniques Non-élitiste

Les méthodes que nous allons aborder ne gardent pas les individus Pareto optimaux trouvés au cours du temps. Elles conservent difficilement la diversité sur la frontière Pareto. La convergence des solutions vers la frontière de Pareto est lente [20].

8.1.1 Niched Pareto Genetic Algorithm (NPGA)

Cette approche suggérée par Horn et Napfliotis utilise un tournoi basé sur la notion de dominance de Pareto. Elle compare deux individus sélectionnés arbitrairement avec une sous-population de taille t_{dom} choisie au hasard aussi. Si l'un de ces deux individus domine le sous-groupe, il prend alors position dans la population suivante. La fonction de sharing est appliquée pour choisir l'individu dans les cas restant. Le paramètre t_{dom} permet d'exercer une pression variable sur la population et ainsi d'augmenter ou de diminuer la convergence de l'algorithme. A travers leurs expérimentations Horn et Napfliotis établissent le constat suivant [21] :

- Si t_{dom} » 1% de N, il y a trop de solutions dominées.
- Si t_{dom} » 10% de N, une bonne distribution des individus est obtenue.
- Si t_{dom} >> 20% de N, il y a une convergence prématurée, car la pression est trop importante lors de la sélection.

En plus des paramètres de sharing, l'emploi de cette approche ajoute un paramètre supplémentaire à fixer par l'utilisateur t_{dom} . Ce dernier a la possibilité d'exercer facilement la pression sur la population. Les solutions trouvées sont généralement de bonne qualité [21].

8.1.2 Multiple Objective Genetic Algorithm (MOGA)

Fonesca et Fleming ont présenté une méthode dans laquelle le rang d'un individu est proportionnel au nombre d'individus le dominant : $r_i = 1 + Dom(i)$. Tous les individus non dominés sont de rang 1. Pour obtenir la fitness de chaque individu, on utilise l'application d'une fonction de scaling sur la valeur de son rang. Cette fonction est en général linéaire. La population a tendance à être répartie autour d'un même optimum en utilisant la sélection par rang. Mais cela n'est pas dans l'intérêt du décideur car cette méthode ne lui propose qu'une seule solution. Pour

contourner ce méfait, les auteurs appliquent la fonction de sharing. Ils souhaitent donc de répartir la population sur l'ensemble de la frontière de Pareto [22].

Le sharing utilisé dans cette méthode agit sur l'espace des objectifs. On suppose que deux actions ayant le même résultat dans l'espace des objectifs ne peuvent pas être présentés dans la population. Grace à cette méthode qui a une implémentation facile, on obtient des solutions de bonne qualité. Les performances sont dépendantes de la valeur du paramètre Sshare utilisé dans le sharing [22].

Algorithme MOGA.

Initialisation de la population
Evaluation des fonctions objectifs
Assignation d'un rang basé sur la dominance
Assignation d'une efficacité à partir du rang
For i=1 **to** G
    Sélection aléatoire proportionnelle à l'efficacité
    Croisement
    Mutation
    Evaluation des fonctions objectif
    Assignation d'un rang basé sur la dominance
    Assignation d'une efficacité à partir du rang
End For

8.1.3 Non dominate Sorting Genetic Algorithm (NSGA)

D'après la méthode de Srinivas [23], on obtient le calcul de la fitness en divisant la population en plusieurs groupes, ceci, en fonction du degré de domination de chaque individu au sens de Pareto[10].

- Premièrement on recherche les individus non dominés dans la population entière. Ces derniers constituent la première frontière de Pareto.
- Cette catégorie d'individus est attribuée une fitness factice. Grace à cette dernière, ces individus ont une chance égale de reproduction. L'application d'une fonction de sharing sur cette valeur est nécessaire, cela dans le but de garder la diversité dans la population.
- Ce premier groupe d'individus est supprimé de la population
- Cette procédure est répétée afin de déterminer la seconde frontière de Pareto. La valeur factice de fitness accordée à ce second groupe est inférieure à la plus petite fitness après avoir appliqué la fonction de sharing sur le premier groupe. La procédure est répétée jusqu'à ce qu'on ait traité tous les individus de la population.
- L'algorithme se déroule ensuite comme un algorithme génétique classique. Le temps de calcul de la notation (trie de la population et sharing) est primordial, il rend cette méthode moins efficace en temps de calcul que la méthode MOGA. En effet cette procédure semble plus apte à maintenir une grande diversité de la population et à dispatcher plus efficacement les solutions sur la frontière de Pareto.

Trois critiques ont été soulevées pour cette méthode [24] :

- ✓ Sa complexité de calcul de $O(k, N3)$ avec k le nombre d'objectifs et N la taille de la population, essentiellement due au processus de trie de la population et d'application de l'heuristique de partage
- ✓ Son approche non élitiste.
- ✓ La nécessité de spécifier un paramètre de sharing.

8.2 Les techniques élitistes

Dans le domaine de l'optimisation multi-objectif, la sélection élitiste consiste à maintenir une seconde population appelée archive, contenant les solutions non dominées trouvées au cours des différentes générations de l'algorithme évolutionniste. Les individus de cette population participent avec une certaine probabilité à l'étape de sélection et donc à la reproduction de nouveaux individus. En outre des techniques de regroupements ou clustering sont employées pour limiter la taille de cette archive. Cette technique est utilisée pour résoudre les difficultés des méthodes non-élitistes.

8.2.1 Strength Pareto Evolutionary Algorithm-II (SPEA-II)

SPEA-II algorithme proposé par Zitzler et Thiele [25], il implante trois techniques communes à d'autres approches qui sont :

- L'utilisation de la dominance au sens de Pareto pour évaluer et sélectionner les individus.
- L'utilisation d'une population secondaire (archive externe) pour stocker les solutions non-dominées.
- L'utilisation du clustering pour maintenir la diversité basée sur la notion de niche.

Principe

- ✓ Pour commencer, SPEA-II crée aléatoirement une population initiale P0 à partir de laquelle il remplit l'archive externe P par les individus non dominés dans P0.
- ✓ A chaque itération, il calcul la valeur fitness des individus des deux populations.
- ✓ Les individus sont triés pour les opérations de croisement et mutation dans le but de générer de nouveaux individus suivant les valeurs de fitness obtenues.
- ✓ Avant de commencer une nouvelle génération, les individus non dominés nouvellement produits procèdent à la mise à jour de l'archive. Si après cette dernière, des individus s'avèrent dominés, ils seront automatiquement supprimés de l'archive.
- ✓ Si la taille de l'archive après insertion, excède la taille permise, une réduction par clustering est alors effectuée [25].

Algorithme : SPEA-II

Initialiser la population P_0 et créer l'archive externe vide P
Mise à jour de P à partir des individus non dominés de P_0
While critère d'arrêt non rencontré **do**
Calcul de la valeur de fitness pour tous les individus de $P_t + P$
Sélection dans $P_t + P$ en fonction de la valeur de fitness
Croisement
Mutation Mise à jour de P à partir des individus non dominés de P_t
End while

- **Calcul de la valeur de fitness**

Le calcul de la valeur de fitness s'effectue en deux étapes de la manière suivante :

La première étape consiste à affecter une valeur, appelée valeur de force($S \in [0,1]$), aux individus de l'archive P. Cette valeur est déduite à partir du nombre d'individus qu'un élément $i \in P$ domine faiblement dans la population courante P_t :

$$S(i) = \frac{Nombre\ des\ domines\ par\ i}{|p_t| + 1}$$

La valeur de fitness d'un individu $i \in P$ est égale à sa valeur de Force : $F(i) = S(i)$.

La valeur de fitness est calculée en additionnant les valeurs de force des individus dominant j plus 1, pour un individu j de la population courante P_t

Le chiffre 1 est ajouté au total de la somme pour éviter que des individus de P aient une valeur de fitness plus grande que certains individus de P_t. Plus un individu est dominé par les individus de P et plus sa valeur d'adaptation décroît, diminuant donc ses chances d'être sélectionné [26].

- **Clustering**

L'utilisation de la technique de clustering consiste à réduire la taille de l'archive.

Au départ de la méthode, chaque individu forme son propre cluster (classe), puis les clusters les plus proches en termes de distance sont réunis deux à deux. Cette étape est itérée jusqu'à obtenir le nombre désiré de clusters.

Après avoir déterminé les clusters, on choisit un représentant par cluster. Ce représentant peut être sélectionné de plusieurs manières, par exemple en prenant le barycentre du cluster. C'est ce représentant qui sera maintenu, les autres éléments étant tout simplement éliminés. Les phases de la méthode de clustering sont illustrées par la figure ci-dessous :[27]

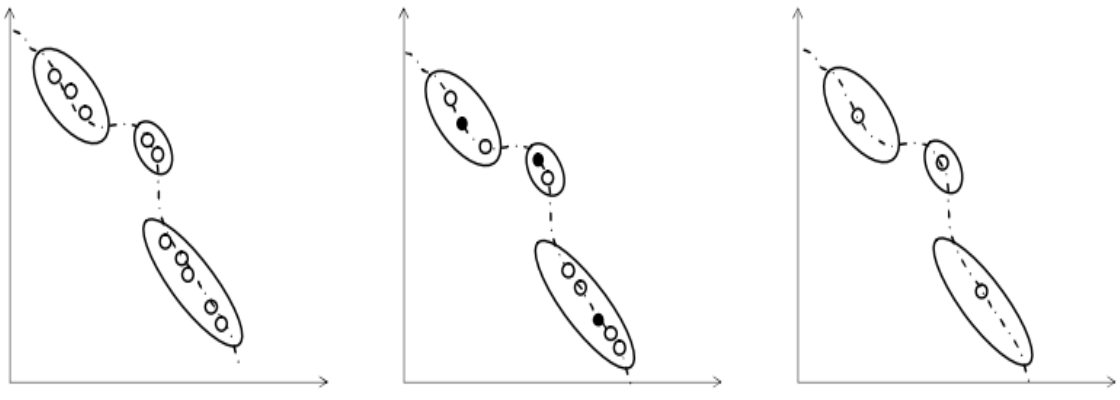


Figure 1.10: Illustration du clustering en deux dimensions.

- **Sélection par tournoi binaire**

Une solution i gagne un tournoi avec une autre solution j si la solution i à un meilleur fitness : $fitness_i < fitness_j$

8.2.2 Non dominated Sorting Genetic Algorithm-II (NSGA-II)

Le NSGA-II est une nouvelle version de l'algorithme NSGA qui est jugée plus fonctionnelle que son prédécesseur. C'est un algorithme élitiste sans archive externe pour stocker l'élite. Pour diriger l'élitisme, NSGA-II garantit à chaque nouvelle génération les meilleurs individus rencontrés et qui seront conservés pour la génération suivante [13]:

- ✓ Il utilise une procédure de tri basée sur la non-dominance, plus rapide.
- ✓ Il ne nécessite aucun réglage de paramètre.
- ✓ Il utilise un opérateur de comparaison basé sur un calcul de la distance de crowding.

- **Principe**
  - ✓ Dans cet algorithme, on utilise une population de taille (N).
  - ✓ La population est ensuite triée selon un critère de non-dominance pour identifier les différents fronts $F1, F2..,$.
  - ✓ Les meilleurs individus vont se retrouver dans le ou les premiers fronts.
  - ✓ Une nouvelle population (P_{t+1}) est formée en ajoutant les premiers fronts au complet (premier front $F1$, second front $F2$, etc.) tant que ceux-ci ne dépassent pas $N/2$.
  - ✓ Si le nombre d'individus présents dans (P_{t+1}) est inférieur à $(N/2)$, une procédure de crowding est appliquée sur le premier front suivant (Fi), non inclus dans (P_{t+1}).
  - ✓ Le but de cet opérateur est d'insérer les $(N/2 - |P_{t+1}|)$ meilleurs individus qui manquent dans la population(P_{t+1}).
  - ✓ Une fois que la première moitié des individus appartenant à la population (P_{t+1}) est identifiée.
  - ✓ La moitié restante de la population (P_{t+1}) est créée par sélection, croisement et mutation.

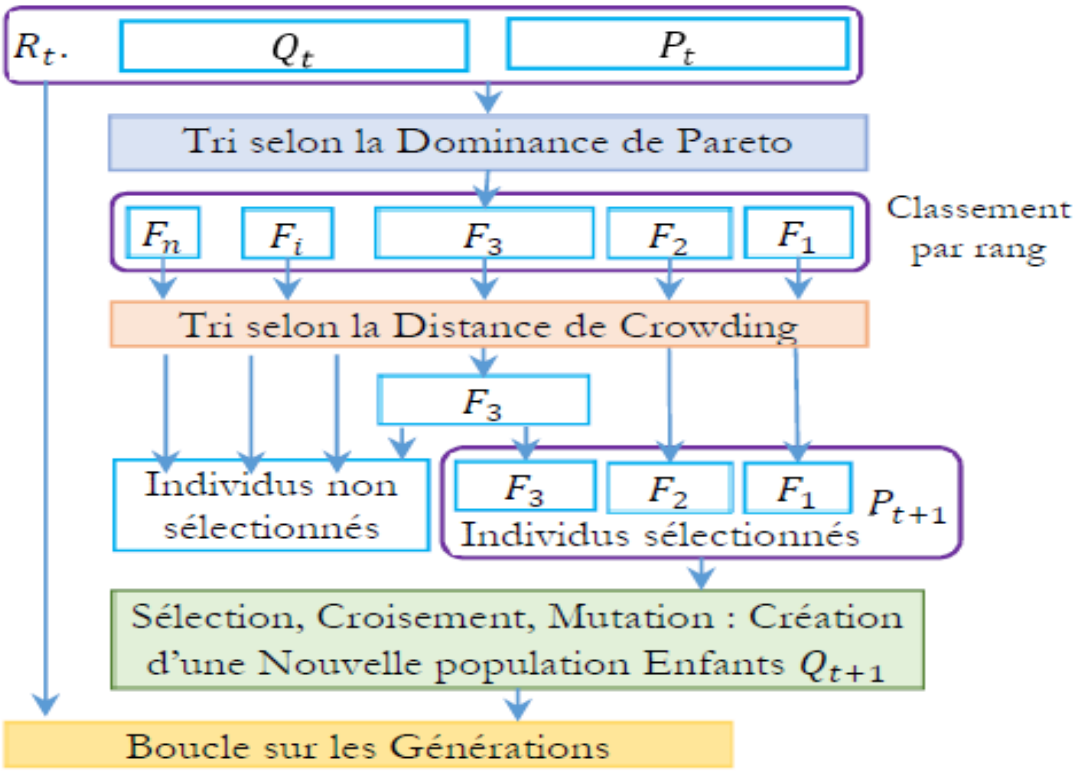


Figure 1.11: Schéma de fonctionnement de NSGA-II [10].

Algorithme : NSGA-II

Initialiser les populations P_0 et Q_0 de taille$N$
While critère d'arrêt non rencontré **do**
Création de $R_t = P_t \cup Q_t$
Calcul des différents fronts Fi de la population R_t par un algorithme de ranking
Mettre $P_{t+1} = \emptyset \; et \; i = 0$
While $|Pt + 1| + |Fi| < N$ **do**
$P_{t+1} = P_t \cup Fi$
$i = i + 1$
End while
Inclure dans P_{t+1} les $(N - |P_{t+1}|)$ individus de Fi les mieux répartis au sens de la distance de crowding
Sélection dans Pt+1 et création de Qt+1 par application des opérateurs de croisement et de mutation
End while

- **Le calcul de la valeur d'adaptation pour NSGA-II**

Il n'est pas utilisé seulement pour le choix des opérateurs de croisement et de mutation, mais aussi il a un rôle dans la sélection des individus à inclure dans P_{t+1} (la population contenant les élites). C'est une phase importante c'est pourquoi en lui a créé une méthode particulière qui est la distance de crowding [10].

- **Crowding**

La distance de crowding d'une solution (S_i) se calcule en fonction du périmètre de l'hyper-cube formé par les points les plus proches de (S_i) sur chaque objectif, Le calcul de la distance de crowding nécessite [27] :

- Le tri des solutions selon chaque objectif, dans un ordre ascendant.
- Ensuite les individus ayant les valeurs limites (la plus petite et la plus grande valeur de fonction objectif $(S_1)\ et\ (S_l)$) se trouvent associés une distance infinie (∞)pour chaque objectif.
- On calcule une distance de crowding égale à la différence normalisée des valeurs des fonctions objectif de deux solutions adjacentes pour les autres solutions intermédiaires
- La distance de crowding d'une solution est calculée en sommant les distances correspondantes à chaque objectif.

Cette distance de crowding se rapporte cependant à la mesure de sa proximité à ses plus proches voisines, rapportée à la taille du front.

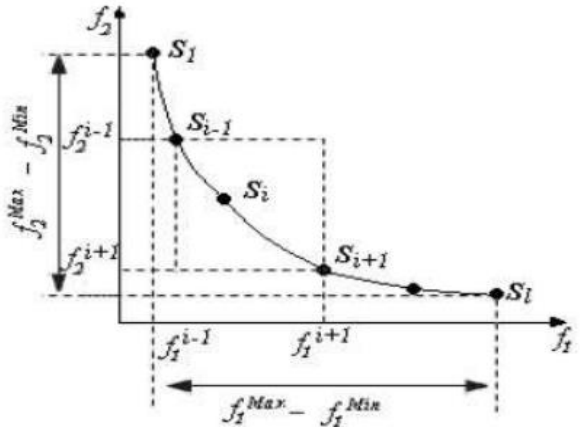


Figure 1.12: Illustration du crowding [10].

- **Sélection par tournoi de crowding**

Une solution i gagne un tournoi avec une autre solution j si l'une des conditions suivantes est vraie [10] :

✓ Si la solution i a un meilleur rang;

$$rang_i < rang_j$$

- ✓ Si elles ont le même rang mais la solution i a une meilleure distance de crowding que la solution j;

$$rang_i = rang_j \; et \; dist_{crowding}(i) > dist_{crowding}(j)$$

- **Avantage et inconvénient**

Avantage: Pour le maintien de la diversité, cette nouvelle version de NSGA ne demande aucun paramètre à fixer. Elle a réduit la complexité de l'algorithme.

Inconvénient : Le mécanisme d'évolution de NSGA-II est comme suit:

A partir d'une certaine génération, toute la population est contenue dans le premier front Pareto. A cette étape, des solutions localisées dans des régions surpeuplées peuvent être supprimées en laissant la place à des solutions non dominées dans la population courante mais qui ne sont pas optimales [26].

8.2.3 Non dominated Sorting Genetic Algorithm-III (NSGA-III)

NSGA-III est un nouvel algorithme génétique de tri non dominée basée sur des points de référence proposée par Deb et Jain [41]. Le cadre de base de NSGA-III est similaire à l'ancienne version NSGA-II qui est populaire dans la résolution des problèmes d'optimisations multi-objectifs. Les deux algorithme (NSGA-III NSGA-II) appliquent un opérateur de croisement et de mutation pour générer une population d'enfants et utilisent une approche de tri rapide non dominée pour déterminer le rang non dominé des individus. Pendant ce temps, ces deux algorithmes utilisent une stratégie de préservation d'élite pour sélectionner la nouvelle génération parmi les parents et les enfants. Mais différant de la crowding distance de NSGA-II, NSGA-III suggère un opérateur de sélection basé sur la mise à jour de nombreux points de référence qui peuvent rendre les fronts optimaux de Pareto bien répartis et améliorer la diversité de la population [42].

Après la génération de la population P_t de taille N Comme NSGA-II, le classement de la population est fait par différents niveaux de solutions non-dominées (fronts, *F1, F2,* etc.). L'étape suivante de l'algorithme NSGA-III est de générer la prochaine génération P_{t+1} (basée sur $F1$, $F2$,…). À partir de F_1, les individus dans les niveaux de non-dominance les plus élevés sont ajoutés à S_t jusqu'à ce que sa taille atteigne N ou dépasse à N pour la première fois, en supposant le niveau de non-dominance l. Les individus (solutions) dans les niveaux de non-dominance supérieurs à l sont simplement écartés, et $St \setminus Fl$ sont sélectionnés comme la génération suivante P_{t+1}. Si la taille de la génération suivante P_{t+1} est= N, l'algorithme répète alors l'étape précédente dans l'itération suivante en générant la population enfant Q_t (si la condition d'arrêt de l'algorithme n'est pas remplie), sinon, les autres individus (solutions) K=N - | Pt + 1 | sont choisis parmi F_l en fonction de points de référence [43], [44].

Puisque les objectifs peuvent être basés sur des échelles différentes, ils sont normalisés et les points de référence sont générés dans l'espace normalisé. Ensuite, chaque solution (individu) est affectée à un point de référence. Chaque individu dans St / F_l est alors affecté au point de référence le plus proche et le reste individus K= N - | Pt + 1 | dans F_l sont choisis de telle sorte que leur point de référence associé n'ait aucun individu associé dans $St \setminus F_l$. La sélection des individus K=N - | P_{t+1} | dans F_l pour le cas de NSGA-II est basé sur la crowding distance avec

des valeurs de crowding distance les plus grandes, par contre, le NSGA-III remplace ce critère par les approches qui sont expliquées suivant [43] :

- **Normalisation des membres de la population**

Pour chaque individu deS_t, chaque valeur objectif f_i est normalisée en soustrayant f_i de z_i^{min}du point idéal de S_t. Le point idéal est un vecteur z =$(z_1^{min}, z_2^{min}, \ldots, z_m^{min})$. , tel que z_i^{min} correspond à la valeur minimale du i[ème] objectif dansS_t, et m est le nombre d'objectifs [44].

- **Génération des points de référence**

Z est un ensemble de points de référence qui sont générés et placés sur un hyperplan normalisé (le nombre d'axes dans l'hyperplan normalisé correspond au nombre total d'objectifs) basé sur la technique de *Das et Dennis* [45]. Le nombre de points de référence *H* tel que $H = |Z|$, il est identifié selon l'équation, où *m* représente le nombre d'objectifs, et *p* correspond au nombre de divisions par axe objectif [44].

$$H = \binom{m+p-1}{p}$$

- **Association entre points de référence et solutions**

Pour chaque point de référence, une ligne de référence est définie en reliant le point de référence à l'origine de l'hyperplan. Après cela, chaque membre *s* dans $S_t$est associé à un point de référence de *Z* basé sur la distance perpendiculaire minimale entre chaque ligne de référence et *s* [43].

- **Opération de préservation de la niche**

Certains points de référence peuvent avoir un ou plusieurs membres associés et certains peuvent ne pas avoir de membres associés. Les membres de S_t/ F_l, associés au point de référence $j \in \Re$, sont comptés et représentés par le compteur de niche ρ_j. Afin d'obtenir une population diversifiée, la procédure suivante est utilisée:
Premièrement, les points de référence avec le minimum ρ_j sont identifiés; s'il y a plus d'un point de référence, l'égalité est rompue au hasard. $\rho_j = 0$ indique qu'aucun membre de S_t/ F_l n'est associé au point de référence *j*. Dans ce cas, il existe deux situations possibles:

- il y a des membres associés à ce point de référence dans F_l, par conséquent, attribuer au membre la distance perpendiculaire minimale à la génération suivante, P_{t+1}, et poser $\rho_j = \rho_j + 1$;
- Il n'y a aucun membre associé à ce point de référence dans F_l, par conséquent, aller simplement à un autre point de référence avec le minimum ρ_j. Si $\rho_j \geq 1$, le point de référence *j* a plus d'un membre associé, et dans ce cas, s'il y a un membre dans F_l associé au point de référence *j*, ajouter l'individu avec la distance perpendiculaire minimale à la génération suivante, $P_{t+1,}$ et définir $\rho_j = \rho_j + 1$. La procédure ci-dessus est répétée jusqu'à ce que la taille de la population atteigne *N* [43].

Algorithme : NSGA-III

Entrées : P_0, N, t, It_{max}
Résultat : $Pt + 1$
While $t \leq It_{max}$ **do**
Création de Q_t
Création de $R_t = P_t \cup Q_t$
Calcul des différents fronts F_i de la population R_t par un algorithme de ranking
Mettre $S_t = \emptyset \; et \; i = 1$
While $|S_t| \leq N$ **do**
$S_t = S_t \cup Fi$
$i = i + 1$
End While
$P_{t+1} = \emptyset$
If $|S_t| = N$ **then**
P_{t+1}=S_t
Else
P_{t+1}= $\cup_{j=1}^{l-1} F_j$
Normalisation des membres de la populationS_t.
Génération des points de références$Z$.
Association chaque solution de S_t avec un point de référence a partir de Z.
Inclure dans P_{t+1} les $K = (N - |P_{t+1}|)$ individus de Fl pour remplir la population P_{t+1}par l'application la procédure préservation de la niche.

End while
$t = t + 1$
End

9 Conclusion

Ce chapitre a pour objectif de présenter dans un premier temps les principales définitions nécessaires à la présentation des problèmes d'optimisation multi-objectif. On a abordé la classification des méthodes de résolution d'un problème d'optimisation multi-objectif selon l'intervention du décideur dans le processus de décision. On s'est penché sur les méthodes utilisées dans le domaine de l'optimisation multi objectif essentiellement les méta-heuristiques. Parmi elles, on trouve le NSGA-III que nous allons exploiter pour optimiser le système de production reconfigurable en terme de coût et de temps.

CHAPITRE 2

Systèmes Manufacturiers Reconfigurables (RMSs)

1 Introduction

De nos jours les entreprises combattent pour survivre. La mondialisation entraîne des variations du marché qui ont entraîné une grave instabilité dans de nombreuses entreprises manufacturières qui luttent pour rester compétitives. Les usines doivent s'adapter aux changements rapides du nouvel environnement économique mondial. Pour cela, la technologie a inventé le RMS qui est un nouveau système rapidement modifiable et rapidement reconfigurable pour permettre aux usines pour être à jour et suivre les changements du marché. La conception du RMS a été le défi principal de cette approche. Ainsi, la réactivité de l'entreprise est essentielle pour rester rentable dans un monde en évolution rapide. Dans ce chapitre on va

2 Les systèmes manufacturiers

Un système est un ensemble composite constitué de personnels, de matériels, de logiciels et de procédures. Tous ces éléments sont en interaction mutuelle dans un environnement donné et sont organisés pour répondre à un besoin. Chaque système est déterminé par la nature de ses éléments constitutifs, les interactions entre ces derniers et le critère d'appartenance au système.

Dans le système de production, il y a les ressources humaines et les ressources physiques. En d'autres termes, on peut le définir comme un ensemble d'opérateurs et de postes de travail et/ou de machines qui sont intégrés. Le rôle de ces machines est d'effectuer une série contrôlée d'opérations répétitives sur les matières premières pour obtenir une forme finale souhaitée ou encore pour assembler un ensemble de pièces afin d'obtenir un produit final. Les postes de travail et/ou les machines sont connectés par l'intermédiaire d'un mécanisme de transfert qui peut être un convoyeur, un robot mobile (AGV : Automated Guided Vehicle) ou tout simplement un opérateur.[28]

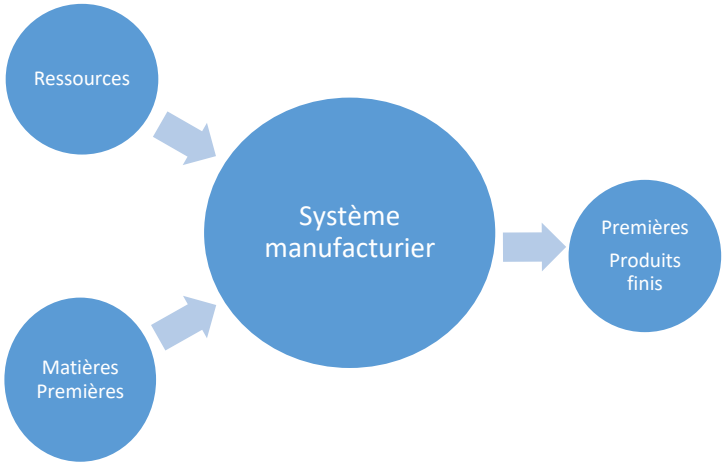


Figure 2.1: schéma classique d'un système manufacturier.

3 L'évolution des systèmes manufacturiers

Le monde de la fabrication a changé considérablement durant le dernier cycle pour répandre aux circonstances économiques et sociales. Dirigé par diverses exigences dans des périodes variées. Les nouvelles technologies de fabrication et les nouveaux paradigmes ont été introduits pour relever les défis économiques et répondre aux besoins sociaux. Faisant face au besoin de la rentabilité, Henry Ford a inventé le système du travail à la chaine en 1913, date à laquelle commença le paradigme de la production de masse.

Dans les années 1970, l'industrie manufacturière japonaise a commencé à formuler des principes de fabrication économique, depuis lors, la qualité constante des produits a été un point imminent. A la fin des années 1970, le développement du contrôle numérique par ordinateur(Computer Numerically Controlled machine-CNC) a facilité l'invention des systèmes manufacturiers flexibles(FMS), qui permettent de produire une variété de produits sur le même système de fabrication.

En 1990, la globalisation a commencé à transformer le paysage concurrentiel. Les sociétés manufacturières font face aux changements imprévus des marchés, sans oublier une demande de produits variant rapidement et l'introduction fréquente de nouveaux produits. Ce qui fait de la conception des systèmes manufacturiers pour les nouvelles usines un défi de grande importance car cela affecte les performances de l'usine pendant de nombreuses années après la conception de l'usine.

Il est indispensable que les nouvelles usines possèdent un nouveau type de système manufacturier, un système conçu pour une réactivité rapide aux poussées imprévues du marché et aux changements imprévus des produits. En réponse à ce défi, une conception d'usines a été proposée avec un nouveau système architectural appelé système manufacturier reconfigurable (Reconfigurable Manufacturing System-RMS).

Le RMS a une architecture de système ouverte qui permet d'ajouter des machines aux systèmes opérationnels existants très rapidement dans le but de répondre rapidement et économiquement aux poussées inattendues de la demande du marché.[29]

4 Systèmes dédiés et les systèmes flexibles

A la fin du 20ème siècle on distinguait deux types principaux de systèmes manufacturiers DMS et FMS, auxquels viennent de s'ajouter les systèmes manufacturiers reconfigurables (Reconfigurable Manufacturing Systems - RMS) au début de ce nouveau millénaire [30]:

4.1 Systèmes Manufacturiers Dédiés (Dedicated Manufacturing Systems- DMSs)

Les systèmes manufacturiers dédiés ont pour but de fabriquer une grande capacité d'un seul produit (Production en masse), la possibilité de changement de produit ou d'apporter une modification sur le système est presque impossible ou très coûteuse [30].

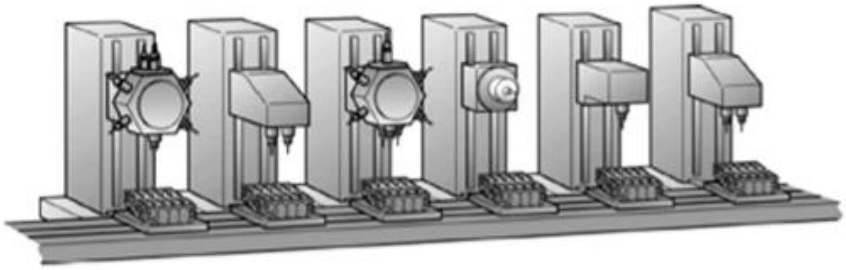


Figure 2.2: Système Manufacturier Dédié (DMS) [30].

4.2 Systèmes Manufacturiers Flexibles (Flexible Manufacturing Systems-FMSs)

Les systèmes manufacturiers flexibles se caractérisent principalement par leurs capacités de fabriquer une variété de produits mais avec une capacité de production très limitée [30].

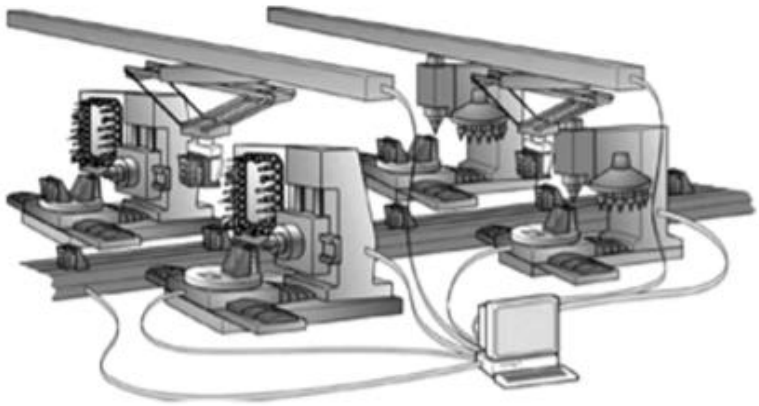


Figure 2.3: Système Manufacturier Flexible (FMS) [30].

Les deux systèmes DMS et FMS sont différenciés par les caractéristiques relatives à la capacité, aux fonctionnalités et aux coûts. Pour les DMS et FMS, les fonctionnalités et les capacités sont généralement fixées. Les DMSs consistent à fournir une grande quantité d'un produit identique de très bonne qualité d'un coût réduit dans la mesure où le taux de demande est élevé mais ce n'est généralement pas le cas dans le marché actuel. Contrairement aux DMSs où chaque machine effectue quelques opérations simples et le procédé est répétitif et bien contrôlé, les FMSs utilisent des machines capables d'effectuer diverses opérations, peuvent produire une large gamme de produits différents [31].

Les FMSs sont adaptés à la production d'une quantité réduite de produits divers. Les principaux inconvénients d'un FMS sont le coût, la complexité et les taux de production faibles.

DMS	FMS
Avantages: • Non coûteux • Rapide - fonctionnement multi-outils	**Avantages:** • convertible • Capacité évolutive
Limitations: • Non flexible - pour une seule pièce • Capacité fixe - non évolutif	**Limitations:** • coûteux • Lent - fonctionnement avec un seul outil

Tableau 2.1: Une simple comparaison entre les DMSs et FMSs [30].

5 RMS : concepts et définitions

L'idée initiale derrière les RMS a été proposée par Lile et Huff. Le RMS est un système de production qui peut utiliser d'une façon intelligente sa configuration dans but de répondre à la demande de production [32].

On peut définir le RMS comme un ensemble de machines à outils reconfigurables (RMTs) et des contrôles numériques par ordinateur (CNC). Il a la capacité de fabriquer au même moment plusieurs types de produits appartenant à une même famille et en quantités irrégulières. Le système peut être reconfiguré physiquement (hard) et/ou logiquement (soft).La suppression et/ou la réorganisation des composants et des fonctions du RMS d'une façon rentable est interprétée par la reconfigurabilité. Le RMS a combiné la caractéristique du taux de production élevé du DMS et la caractéristique de flexibilité du FMS [33].

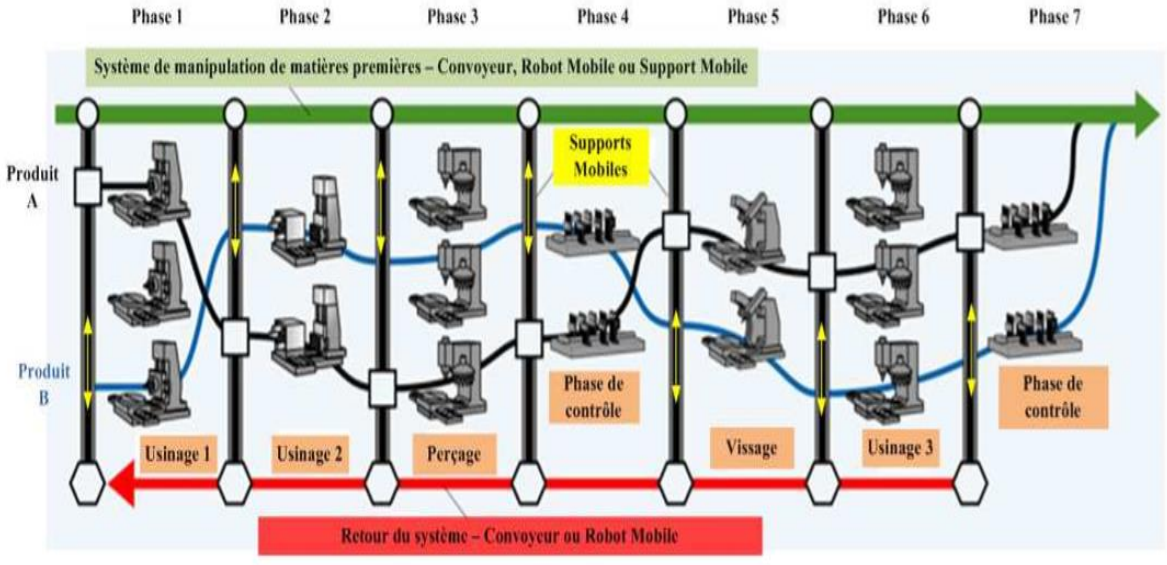


Figure 2.4: Système de production reconfigurable(RMS) [28].

5.1 Les différentes caractéristiques du RMS

On dit qu'un système est reconfigurable si et seulement s'il possède les caractéristiques de base. Ces dernières devraient être implémentées dans le système reconfigurable dans la phase de la conception [1].

- **Modularité :** Les principaux composants du RMS doivent être modulaires et sont constitués d'éléments structurels, des axes, des contrôles, des logiciels et de l'outillage.
- **Intégrabilité :** La capacité d'intégration des modules rapidement et précisément par un ensemble d'interfaces mécaniques, informationnelles, et de contrôle permet l'intégration et la communication.
- **Diagnostabilité :** la lecture automatique de l'état actuel d'un système et les contrôles afin de découvrir et de déterminer les raisons profondes des défauts, et puis les corriger.
- **Evolutivité :** Le changement proportionnel et facile de la capacité de production existante est réalisé par la réorganisation d'un système de production existant, et / ou par le changement de la capacité de production de composants reconfigurables auprès de ce système.
- **Convertibilité :** La capacité de transformer facilement la fonctionnalité des systèmes existants, des machines et des contrôles en fonction de nouvelles exigences de production.
- **Personnalisation :** Pour répondre aux nouveaux besoins auprès d'une famille de produits identiques, cela demande l'adaptation d'une flexibilité personnalisée des systèmes de production et des machines.

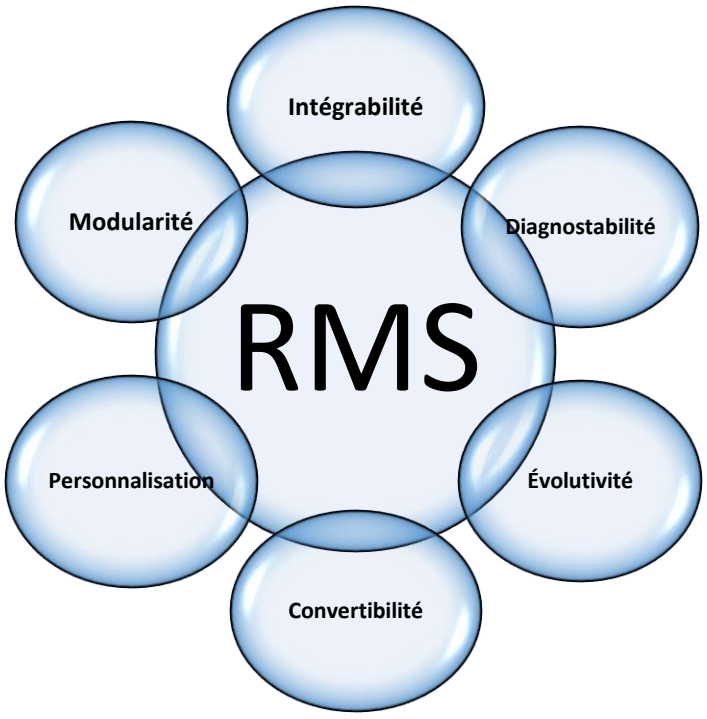


Figure 2.5: les caractéristiques du RMS [1].

5.2 La famille de produits

La famille de produits : C'est un ensemble de produits identiques ayant quelque partie, de composants et/ou de modules communs.

Le fabricant est capable d'élaborer une stratégie de famille de produits où certains modules fonctionnels sont partagés alors que d'autres sont attribués avec plusieurs variantes chacun. La combinaison de ces derniers fournira une grande variété dans les produits finaux.

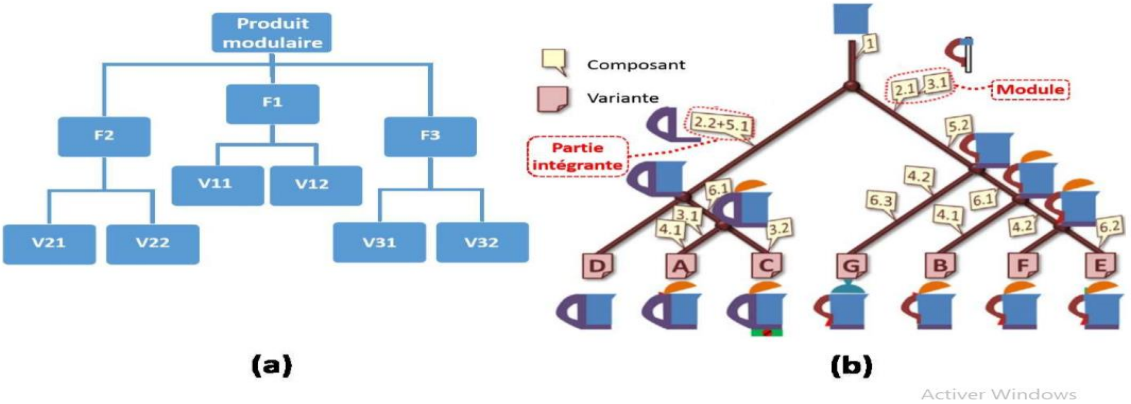


Figure 2.6: Architecture de la famille de produits [34].

5.3 Machines Reconfigurables (Reconfigurable Machines-RMs)

Les machines reconfigurables (reconfigurable machine -RMs) sont des machines dont les structures peuvent être changées pour fournir une fonctionnalité alternative et/ou la capacité de la mise à jour selon la demande. Elles sont toujours conçues en fonction des caractéristiques communes de la famille de produits [35].

5.3.1 Machines-outils reconfigurables (Reconfigurable Machines Tools-RMTs)

La machine reconfigurable « Reconfigurable Machine Tool- RMT» est le principal composant d'un RMS. C'est un nouveau type de machine ayant une structure modulaire permettant sa reconfiguration. Elle possède une structure qui peut être reconfigurée pour fournir soit une fonctionnalité alternative ou assurer une augmentation progressive de son taux de production pour satisfaire les exigences de la demande du client. Notons que, un RMT peut toujours revenir à son état original en cas de besoin [33].

La conception des RMTs se repose donc sur deux objectifs fondamentaux :

1. Adapter le fonctionnement de la machine pour répondre aux différents besoins en termes de fonctionnalités.

2. Augmenter le taux de production de la machine en ajoutant des ressources supplémentaires.

Dans la Figure 2.7 un exemple d'un RMT. La pièce à manufacturer est située sur un support qui peut se déplacer simultanément sur les deux axes X et Y. Les broches Z1et Z2 peuvent se déplacer linéairement sur leurs axes respectifs, situés sur le même support. La broche Z3 peut usiner la pièce sur un axe horizontal avec des différents angles. Les broches peuvent être

rapidement ajoutées ou enlevées en fonction des besoins. Dans la deuxième configuration, on a déplacé le support horizontal du slot B au slot C et le support horizontal a également changé de position. Cette machine, avec ces différentes configurations possibles, a accès à plus de points d'usinage par rapport à une machine classique, et une productivité plus importante par rapport à une machine CNC ayant un outil unique [27].

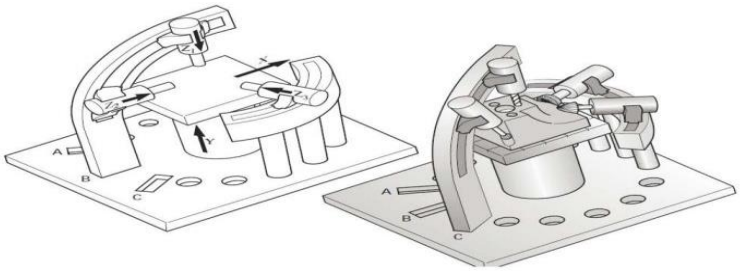


Figure 2.7: Exemple d'une machines-outils reconfigurable (RMT).[27]

5.4 Lignes de production

Détermine les composants nécessaires, les machines et leur ordonnancement pour effectuer toutes les opérations d'un produit particulier en peu de temps et à faible coût. En d'autres termes, la ligne de production assigne à chaque machine un ensemble d'opérations qui peuvent être effectuées sur cette machine pour obtenir le produit souhaité. La figure ci-dessous représente un exemple illustratif d'une ligne de production de 4 opérations. La deuxième colonne représente l'ensemble de machines avec leurs configurations et l'outil disponible (Machine 2(M2)-Configiration 2(C2)-Outil 2(T2)), donc le problème est parmi lesquelles elle a la capacité de réaliser l'opération 2. Donc, pour atteindre le produit final il faut passer à travers une ligne de production commençant par la matière première.

Solutions Réalisable no.	Configuration de la machine réalisable pour la séquence d'opération 2→5→7→15→8→16					
	2	**5**	**7**	**15**	**8**	**16**
1	M2-C2-T2	M1-C2-T3	M1-C3-T4	M2-C2-T4	M1-C1-T3	M1-C1-T3
2	M2-C4-T4	M2-C4-T2	M2-C4-T1	M3-C2-T1	M2-C3-T2	M1-C3-T4
3	M1-C3-T2	M5-C2-T2	M5-C1-T1	M5-C3-T5	M3-C2-T1	M4-C3-T5
4	M4-C3-T3	-	M5-C4-T5	-	M4-C3-T1	M5-C2-T2

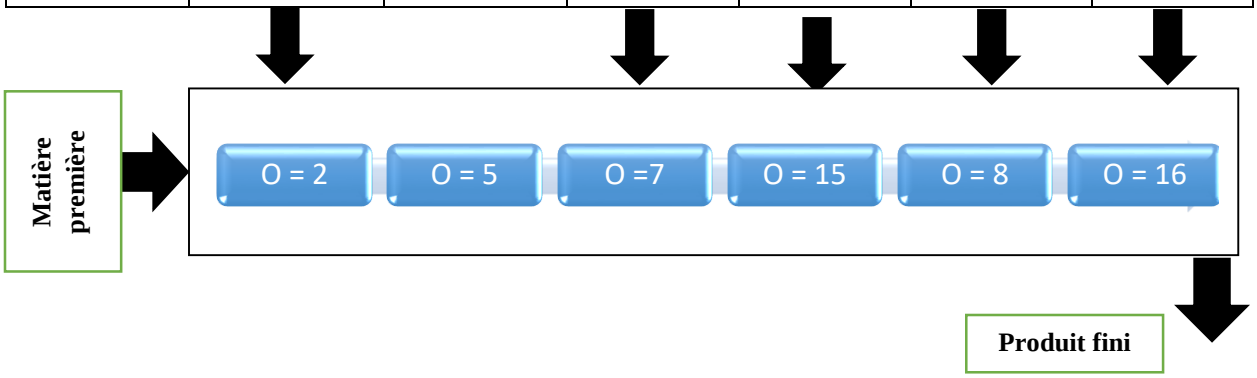


Figure 2.8: Exemple d'une ligne de production [3].

Enfin pour réaliser cette sequence 2→5→7→15→8→16, on trouve plusieurs séquences de machines par exemple :

5.5 La configuration et la reconfiguration du RMS

5.5.1 La configuration du système

La configuration est un ensemble paramètres, elle est mise en place pour satisfaire l'objectif fixé par le système. On peut dire que la configuration est l'interprétation de la définition de l'objectif du système en une organisation de ses éléments [31].

5.5.2 Les déclencheurs de la reconfiguration

L'occurrence d'un évènement déclencheur de la reconfiguration (DR) nécessite la réalisation de la reconfiguration du système. Par exemple, le changement de produit est un DR qui nécessite la reconfiguration du système pour pouvoir fabriquer le nouveau produit. Généralement, il y a deux types d'DRs [31]:

- Les DRs extrinsèques au système : ils font partie du contexte du système manufacturier et ils n'ont aucune relation avec le système. On peut citer comme DR extrinsèque, les variations de la demande du client (augmentée, ou diminuée).
- Les DRs intrinsèques au système : ce sont généralement les pannes. Le système, grâce à sa caractéristique de reconfiguration peut négliger la panne et continuer de satisfaire la demande du client.

En résumé, si un déclencheur survient quelque soit son type, le système est obligé de procéder à la reconfiguration pour s'adapter aux changements de contexte provoqués par cette occurrence. La reconfiguration du système nécessite généralement l'intervention d'opérateurs humains pour faire les changements nécessaires, par exemple ajouter une machine, ajouter une ligne d'assemblage, etc.

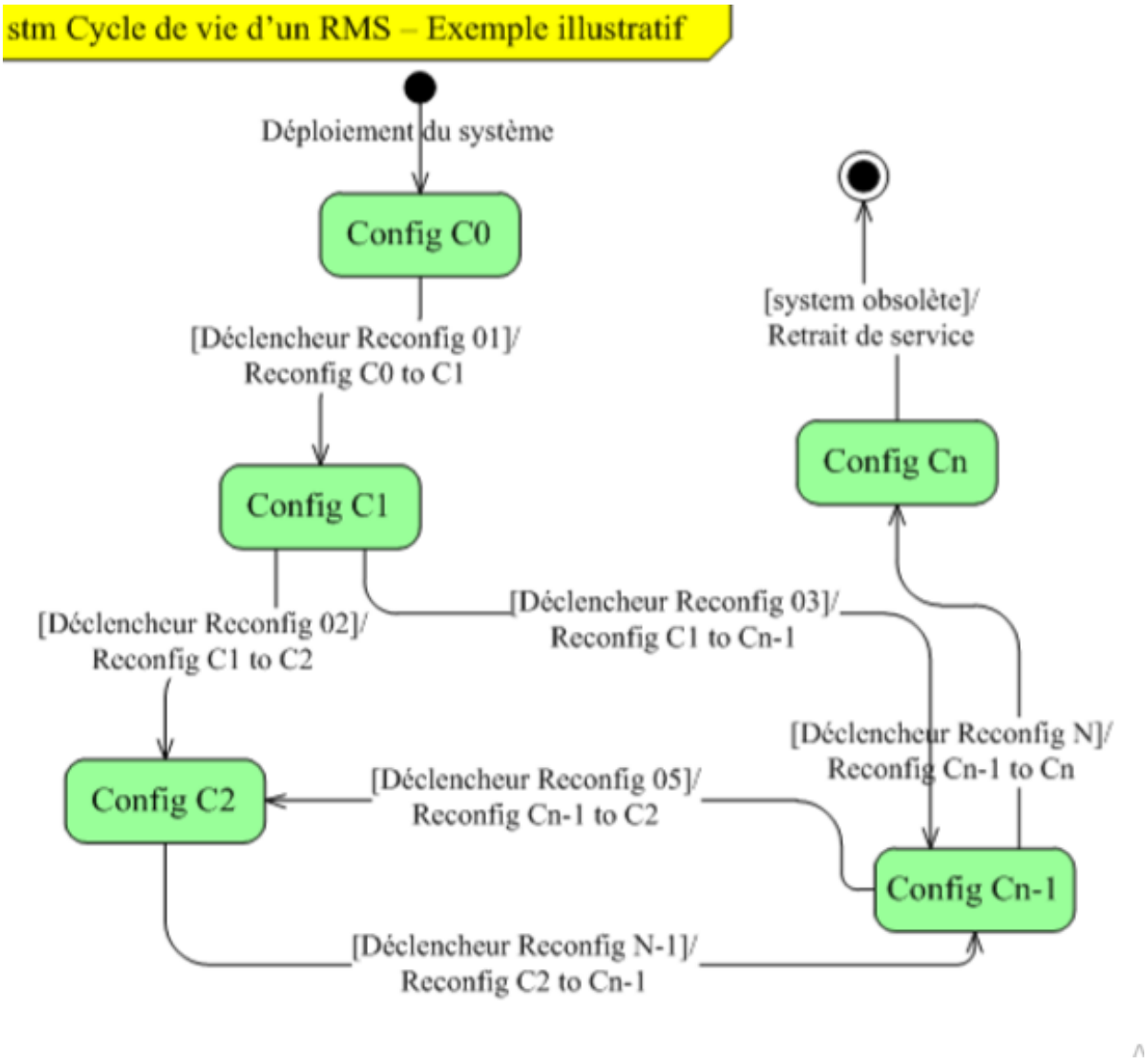


Figure 2.9: Exemple illustratif du changement de configurations d'un RMS pendant sa phase opérationnelle [31].

5.5.3 Processus de reconfiguration

Le processus de reconfiguration d'un système reconfigurable fonctionne quand l'objectif déterminé pour la configuration actuelle est satisfait ou quand la configuration actuelle du système ne répond plus à ces mêmes objectifs. Le processus a lieu en plusieurs phases une fois la décision de lancer le processus de reconfiguration est prise [36] :

- Une nouvelle configuration répondant aux objectifs fixés est recherchée.
- Un cheminement doit être déterminé pour aboutir à cette configuration.
- Placement du système dans un état acceptable pour la nouvelle configuration.
- Mise en œuvre de la reconfiguration pour arriver au nouveau fonctionnement répondant aux objectifs fixés.

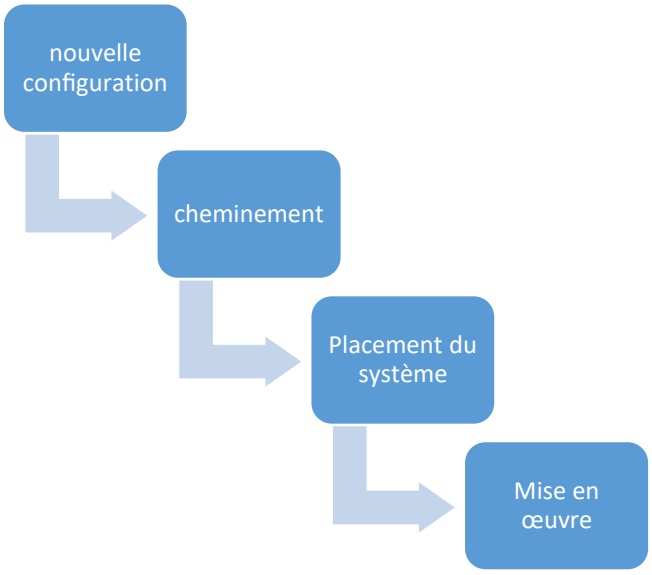


Figure 2.10: schéma présentant les différentes phases du processus de reconfiguration.

5.5.4 Les types de reconfiguration

La figure ci-dessous illustre les différents types de reconfiguration qui concernent un RMS, peuvent être récapitulés comme suit :

- Dans la structure du système : réorganiser les machines suivant une configuration série, parallèle ou hybride.
- Une reconfiguration logicielle du système.
- Une reconfiguration dans le système de contrôle associé au système.
- Une reconfiguration au sein d'une machine du système : par exemple ajouter ou supprimer des modules, changement de quelques composants…
- Une reconfiguration dans le processus de fabrication du système [33].

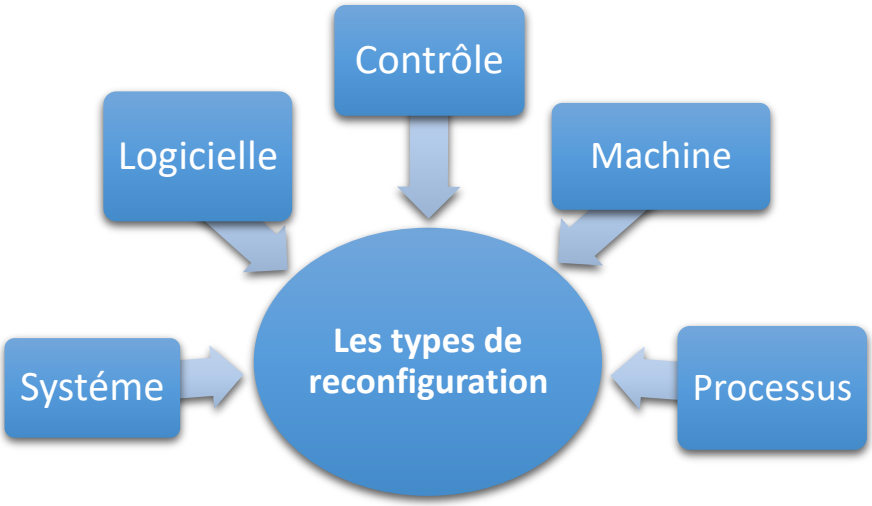


Figure 2.11: les différents aspects de reconfiguration adaptée.

6 La conception des RMSs

6.1 Principes de conception des RMSs

La conception d'un RMS demande l'application d'une vision à long terme du système de production, pour assurer la faisabilité économique de plusieurs générations de produits et de situations de marché. En d'autres termes.

La conception des RMSs doit être faite selon les principes de la reconfiguration. Plus ces principes sont applicables à un système de production, plus ce système est reconfigurable. L'intégration de ces principes dans la conception des RMSs mène à la réalisation de l'objectif final. Ce dernier a pour but de créer une usine dynamique qui est capable d'adapter rapidement sa capacité de production tout en préservant des niveaux élevés de qualité des produits.

Ces principes sont :

- Un RMS doit fournir des ressources de production réglables pour répondre aux changements imprévisibles du marché et les évènements intrinsèques du système.
- Un RMS doit être conçu autour d'une famille de produits, avec juste suffisamment de flexibilité personnalisée pour produire tous les membres de la famille.
- Les caractéristiques de base d'un RMS devraient être intégrées dans le système dans son ensemble, ainsi que dans ses composants (physiques et logiques).
- La capacité d'un RMS doit pouvoir être rapidement ajustée (incrémenté ou décrémenté) par petits incréments.
- La fonctionnalité d'un RMS doit pouvoir être adaptée rapidement à de nouveaux produits.

Les capacités d'ajustement intégrées d'un RMS doivent faciliter une réponse rapide à des défaillances imprévues du matériel [31].

6.2 Conception des produits et formations des familles de produits

Un RMS est conçu autour d'une famille de produits avec suffisamment de flexibilité pour pouvoir les fabriquer tous. Ces produits sont regroupés en familles en fonction de certaines caractéristiques partagées telles que la modularité, la séquence d'opérations de fabrication, etc. Chaque famille de produits nécessite une configuration du système permettant de les produire, et le passage d'un produit à l'autre ou plus généralement d'une famille de produits à l'autre nécessite la reconfiguration du système [31].

6.3 La conception d'une ligne de production

La ligne de production constitue le rapport entre le produit (la conception) et le système de production (ressources du système de production). Cette ligne est attachée au produit à travers le lien entre les caractéristiques du produit et les opérations (chaque caractéristique du produit est réalisée en effectuant une ou plusieurs opérations à des machines dans cette ligne).

7 Les indicateurs de performance dans les RMSs

On trouve différentes mesures de performances utilisées pour évaluer la performance d'un système manufacturier reconfigurable « RMS » telles que le temps de production, le coût de

production, la fiabilité du système, la disponibilité, la maintenance, le taux de productivité, les délais, le temps de reconfiguration, etc. Des recherches sont basées sur les mesures de performance et la façon de trouver la meilleure configuration pour les RMS. Dans la section travaux conexe, on a trouvé qu'il existe plusieurs travaux optimise le RMS grâce à l'utilisation des indicateurs de performance qui sont représentés comme des fonctions objectifs.

8 Comparaison entre DMS, FMS, RMS

Dans cette partie de chapitre, nous allons procéder à la comparaison des RMSs par rapport aux autres types de systèmes manufacturiers.

L'environnement concurrentiel mondial demande des systèmes fiables, performants et non-couteux, et les DMSs et FMSs sont incapable de satisfaire leurs demandes. La notion de système de production de type RMS englobe les caractéristiques positives des DML et des FMS. Ajoutons à cela l'amélioration des points indésirables comme il est montré dans le Tableau 2.2[28].

	DMS	RMS	FMS
Structure du système	Fixe	Modifiable	Modifiable
Structure de la machine	Fixe	Modifiable	Fixe
Réglage du système	Partie	Famille composants	Machine
Flexibilité	Non	Personnalisée	Générale
Évolutivité	Non	Oui	Oui
Exploitation des outils Simultanément	Oui	Oui	Non
Productivité	Elevée	Elevée	Faible
Coût	Faible	Moyen	Onéreux

Tableau 2.2: Comparaison entre les DMS, FMS et RMS [29].

Enfin, il est possible de comparer les trois types de système de production DMS, FMS et RMS en se basant principalement par rapport à quelques critères (capacité, fonctionnalité et coût).

On définit la capacité comme la cadence nécessaire pour répondre à la demande du client (en terme de volume de production) dans les délais convenus.

La fonctionnalité renseigne sur l'aptitude du système à exécuter l'ensemble des opérations nécessaires à la fabrication d'un ou de plusieurs produits (familles de produits).

Les DMSs et les FMSs sont limités en capacité et fonctionnalité par rapport aux RMSs ou ces critères sont changeants au fil du temps en fonction du marché. On déduit qu'un RMS balance entre un DMS et un FMS en termes de capacité et de fonctionnalité. Il a des capacités et des fonctionnalités variables.

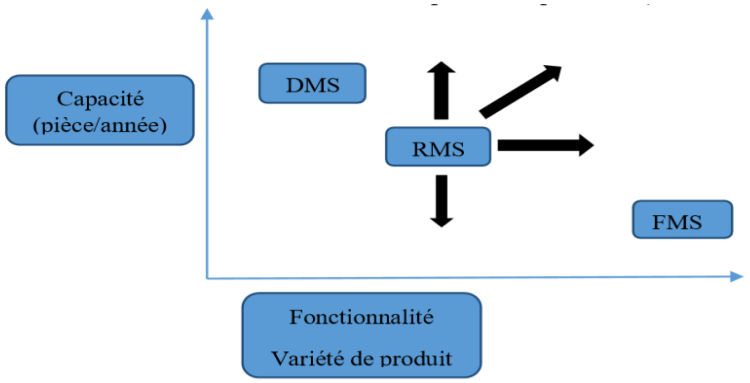


Figure 2.12: Positionnement du RMS par rapport au DMS et FMS.[33]

9 Les travaux connexes

Cette partie du travail abord brièvement quelques problèmes d'optimisation de la reconfigurabilité des RMS résolus par des méta-heuristiques. Voici quelques études :

Bensmain Abderrahman [2] ont focalisé sur le problème de génération des processus de fabrication optimales dans un RMS, utilisant au maximum les hauts degrés de reconfigurabilité des RMTs pour obtenir des plans efficaces. La fonction de génération des plannings de fabrication comporte trois problèmes :

1) La génération des processus de fabrication dans un cas unitaire où ils ont adapté des techniques d'optimisation multicritère (NSGA-II et AMOSA).
2) La génération des processus de fabrication dans le cas multi unité où une optimisation basée sur la simulation a été adaptée.
3) L'intégration des fonctions de génération des processus de fabrication avec l'ordonnancement où ils ont développé une nouvelle heuristique permettant d'effectuer cette intégration.

Les expériences numériques démontrant l'applicabilité et l'efficacité des approches proposées.

Musharavati & Hamouda [40] ont réussi à obtenir en termes de résultats obtenus et de l'effort de calcul une amélioration satisfaisante des performances de la méta-heuristique. Ils ont procédé à la combinaison du recuit simulé (Simulated Annealing -SA) avec l'exploitation des connaissances et l'architecture parallèle dans le cadre de générer des processus de fabrication dans un RMS en minimisant les coûts d'exploitation et en maximisant la productivité.

Hichem Haddou Bendrbal [33], a travaillé sur la problématique de conception des systèmes manufacturiers reconfigurables (RMSs). Il a pour but de concevoir des systèmes réactifs en se basant sur leurs capacités en fonction de reconfigurabilité. Le travail a été élaboré sur trois niveaux :

1) le niveau des composantes relatif aux modules des machines reconfigurables.

2) le niveau des machines et leurs interactions ainsi que l'impact de ces interactions sur le système.
3) le niveau de l'atelier composé de l'ensemble des machines reconfigurables.

Afin d'assurer les meilleurs performances du système conçu telles que l'indicateur de modularité, l'indicateur de flexibilité, l'indicateur de robustesse et l'effort d'évolution d'un système reconfigurable, des indicateurs de performance ont été développés pour chaque niveau. Ils ont développé des modèles d'optimisation multicritère, résolus à travers des méta-heuristiques multicritères (AMOSA) et (NSGA-II) pour l'ensemble des problèmes étudiés. De nombreuses expériences numériques et analyses ont été réalisées afin de démontrer l'applicabilité des approches.

Faisal Hasan, et al [39] ont fait des recherches sur l'optimisation des performances des systèmes manufacturiers reconfigurables (RMSs) en optimisant leur productivité. Ils ont appliqué l'algorithme génétique pour l'affectation de la configuration optimale de la machine pour une ligne de production reconfigurable, avec la minimisation du temps de cycle comme fonction objectif.

Chaube et al [37] suggèrent une nouvelle approche pour générer des processus de fabrication dynamique dans un système manufacturier reconfigurable. Les exigences des pièces / produits sont évaluées puis comparées à la fonctionnalité offerte par les machines comprenant un système de fabrication. Un processus de fabrication optimal est généré si la production est réalisable. Dans le cas contraire, un message d'erreur est affiché. Cette approche se base sur la méta-heuristique NSGA-II utilisée pour résoudre le problème multi-objectif dont la réduction du coût et du temps de fabrication sont des fonctions objectifs.

Ashraf et al [3] ont résolu problème d'optimisation multi-objectif dans l'environnement RMS considérant quatre fonctions objectif : le coût, reconfigurabilité, capacité opérationnelle et fiabilité. Ce problème est résolu par l'utilisation de l'algorithme NSGA-II et dans la ligne de production les opérations à fabriquer sont associés à une RMT adéquate. Les ensembles de solutions non dominées ainsi obtenus pour l'allocation RMT la ligne de production à partir de l'application de NSGA-II sont nombreux. Par conséquent, ces solutions sont ensuite classées par l'application d'une technique de tri de solution, nommé TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution).

Après l'analyse les travaux cités au-dessus, nous avons vus que la majorité des travaux utilise le NSGA-II comme une métha-heuristique utilisée pour générer leurs solutions optimales. Vu que les meilleures performances de la nouvelle génération d'un algorithme évolutionnaire nommé NSGA-III dans plusieurs problèmes traitent plus de deux fonctions d'objectifs. Pour cette raison, on va l'adapter pour générer des lignes des productions optimales selon trois objectifs : le temps de productions, le cout de production, le taux de la fiabilité. Ces derniers ont inspiré de [2], [3].

Voilà un tableau comparatif entre notre travail et les travaux connexes :

Travaux	Méta-heuristiques Utilisées	Fonctions objectif évaluées
[2]	AMOSA/ NSGA-II	Minimise le cout/le temps
[3]	NSGA-II /TOPSIS	Minimise le coût, reconfigurabilité, Maximise la capacité opérationnelle et fiabilité
[33]	AMOSA/ NSGA-II	Plusieurs indicateurs
[37]	NSGA-II	Minimise le temps de cycle/le cout
[39]	GA	Minimise le temps de cycle de production
[40]	SA	Minimise le cout/maximise la productivité
Notre travail	NSGA- III	Minimiser le cout/le temps et maximise la fiabilité

Tableau 2.3 : Comparaison entre notre travail et les travaux connexes.

10 Conclusion

Dans ce chapitre nous allons aborder le système manufacturier reconfigurable « RMS » qui est considéré comme une nouvelle génération dans les systèmes manufacturiers. Nous allons présenter une idée générale sur les systèmes manufacturiers traditionnels «DMS et FMS» et une idée approfondie sur les RMSs. Nous avons achevé le travail avec une comparaison entre les trois paradigmes « DMS, FMS et RMS ».

Dans le chapitre suivant nous avons présenté la conception globale et détaillée de notre système. Nous avons expliqué en détail l'algorithme génétique NSGA-III.

CHAPITRE 3

Conception du Système

1 Introduction

Quand il s'agit de la conception du système, on parle d'une activité itérative multi-niveau ; c'est l'escale la plus importante dans le processus de développement des logiciels. Ce dernier consiste à représenter les fonctionnalités du système d'une manière permettant l'obtention rapide d'un ou plusieurs programmes réalisant ces fonctionnalités. Celui-ci ne concerne que l'étape d'analyse et de définition des besoins. Ainsi, on vise le design du produit logiciel souhaité par l'utilisateur dans le but d'atteindre une résolution meilleure du problème posé.

Dans notre système, nous allons utiliser la conception fonctionnelle descendante comme suit:

- Le problème à résoudre.
- La construction du schéma général du système et de ses principales unités.
- La conception détaillée du système en question.

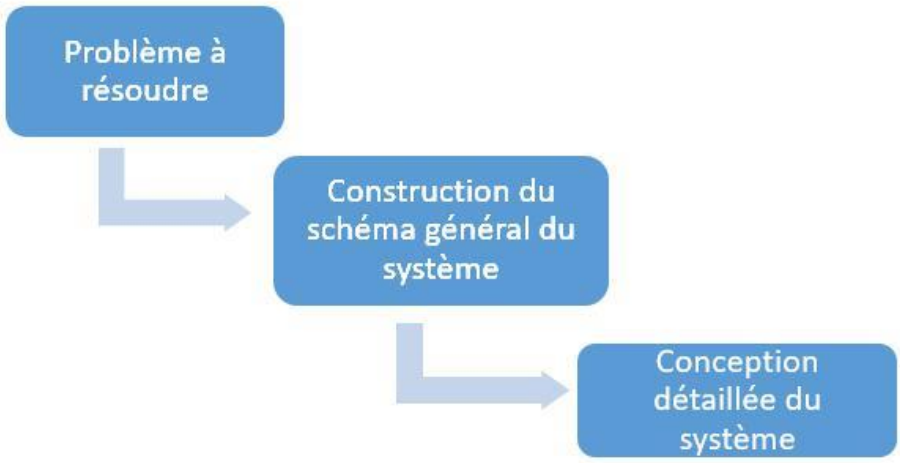


Figure 3.1: schéma représentant la conception fonctionnelle descendante.

2 L'objectif du système

La recherche de l'ensemble optimal des machines pour fabriquer un produit, tel est l'objectif majeur de notre projet qui repose sur deux principales tâches : donner une séquence d'opérations et les affecter aux machines (rechercher des meilleures séquences des machines avec leurs configurations et leurs outils pour fabriquer le produit désiré). Donc, on cherche à minimiser le temps (temps de changement des machines, le temps de changement d'outils, le temps de changement de configurations, temps de traitement), le coût (coût d'utilisation des machines/outils, coût de changement des machines/outils, coût de reconfiguration) et maximiser la fiabilité des machines. Ce problème appartient donc à la catégorie des problèmes d'optimisation multi-objectif.

Dans cette recherche on s'est inspiré des travaux qui proposent les mêmes modèles mathématiques utilisés pour calculer le coût, le temps et la fiabilité totale de la production [2],[3].

Chaque produit a un ensemble de composants (F). On se basant sur ces composants et parmi un ensemble de machines reconfigurables disponibles (M), un ensemble de machines candidates est sélectionné selon ses fonctionnalités. Par la suite, nous décidons quelles sont les machines les plus optimales dans la ligne de production.

Généralement, des algorithmes évolutionnaires (EA) sont utilisés pour résoudre ce type de problème. Dans notre cas, nous utiliserons l'algorithme évolutionnaire (NSGA-III). Ce qui consiste à essayer d'optimiser la qualité des solutions obtenues et de converger vers les solutions réalisées par cet algorithme.

3 Conception du système

3.1 Conception Globale

L'objectif de la conception globale est de présenter notre modèle en plusieurs composants plus simples. La figure suivante illustre l'architecture globale proposée pour notre système.

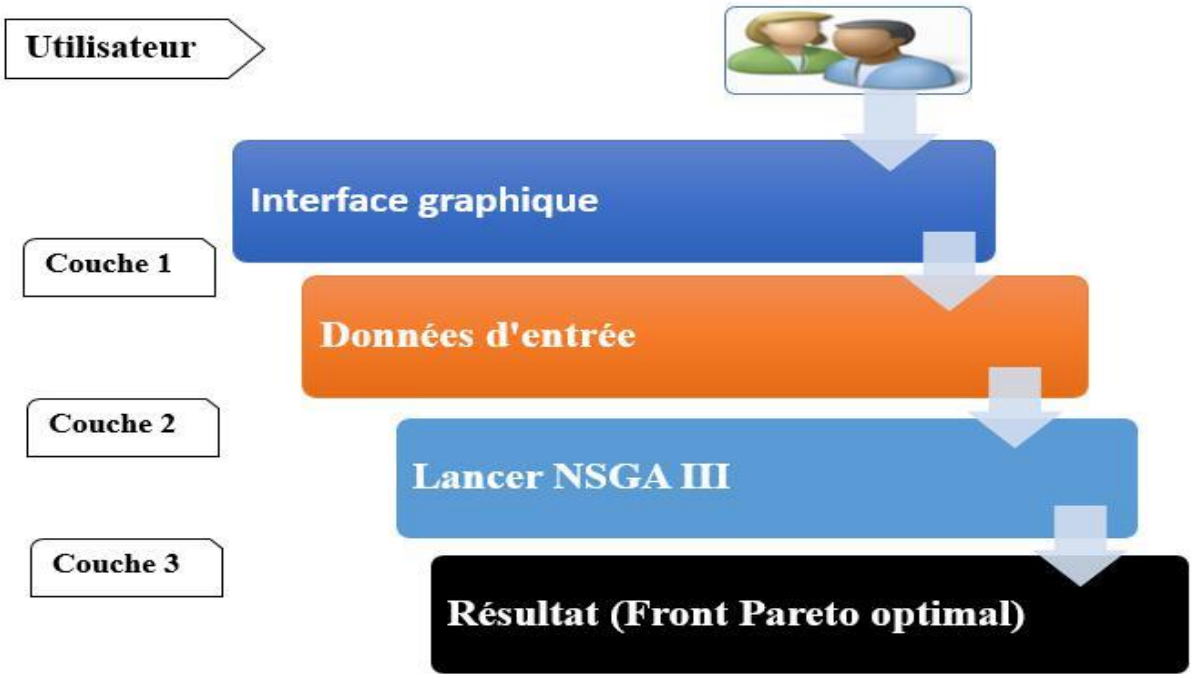


Figure 3.2: Architecture globale du système.

3.1.1 Couche 1

Elle est le niveau supérieur de l'application, Il s'agit de l'interface graphique visible par l'utilisateur lors du chargement de l'application ainsi que ses paramètres et ses données. En conséquence, cette couche assure l'interactivité entre l'utilisateur et la machine en lui offrant les moyens pour manipuler l'exemple à exécuter.

3.1.2 Couche 2

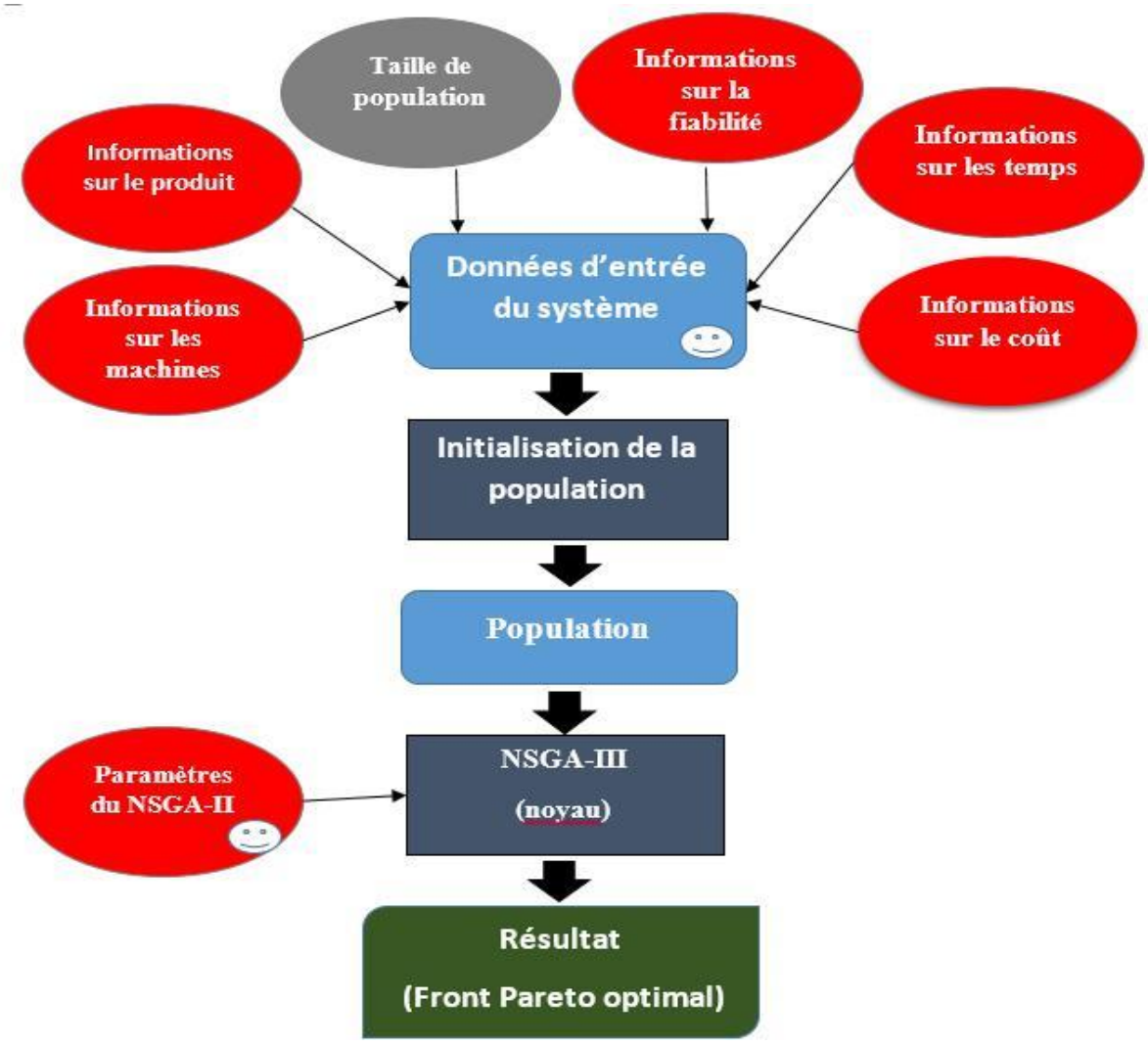
Le rôle de cette couche est d'établir la relation entre les deux autres couches (les couches 1 et 3) et d'assurer le stockage des données afin que ces dernières puissent être utilisées dans la couche noyau. Donc, elle est la liaison entre l'interface et le code source.

3.1.3 Couche 3

C'est le module principal du système (le moteur). Il traite les données chargées pour obtenir des résultats. Ce module est l'implémentation du mécanisme de l'algorithme NSGA-III.

3.2 Conception détaillée

Dans cette section, l'architecture détaillée du système est proposée pour être utilisée au développement et à la réalisation du système.



: C'est-à-dire, Cette information est entrée par l'utilisateur.

Figure 3.3: Architecture détaillée du système.

3.2.1 Données d'entrée du système

Dans cette section, nous présentons les données d'entrées nécessaires. Ces données sont réparties en quatre groupes (informations sur le produit, informations sur les machines, informations sur le coût et informations sur le temps).

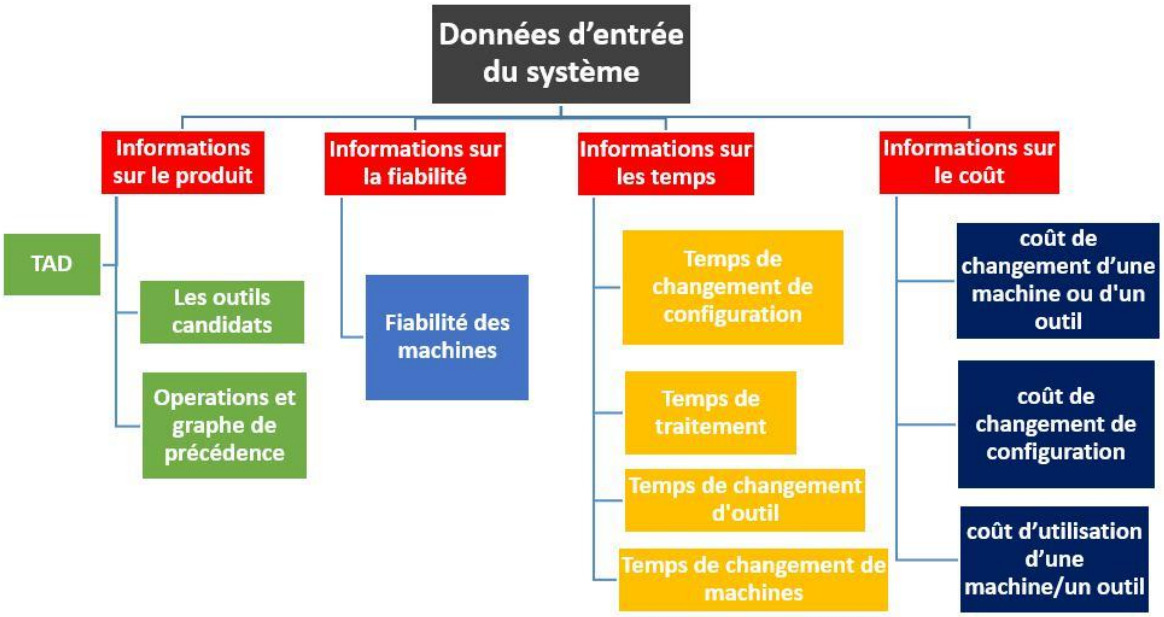


Figure 3.4: Architecture des données d'entrée du système.

3.2.1.1 Informations sur le produit

Le produit étant défini comme un ensemble de composants qui, chacune d'entre elles, nécessitent un ensemble d'opérations.

- **La séquence d'Operations**: La séquence d'opérations les contraintes de précédence qui exige un ordre précis de succession des opérations.
- **TAD requis :** Pour un produit, un TAD désigne la direction suivant laquelle une opération doit être effectuée. Un TAD est défini sur le repère tridimensionnel X, Y et Z.
- **Les outils candidats :** Ce genre d'information spécifie les outils pouvant effectuer une opération particulière. Et chaque outil à ses propres composants en termes de précision et coût d'utilisation.

3.2.1.2 Informations sur les machines

Les machines représentent l'environnement duquel le produit va être sorti. Les informations sur ces machines concernent essentiellement leurs configurations possibles et les capacités dont elles disposent dans chacune de leurs configurations, ainsi que les outils nécessaires pour la réalisation de leurs opérations.

3.2.1.3 Informations sur le temps

L'information sur le temps concerne la durée nécessaire pour les différentes machines pour accomplir toutes les opérations ayant pour résultat le produit fini. Ce temps se compose de quatre phases:

- Le temps de traitement.
- Le temps de changement de configuration.
- Le temps de changement d'outil.
- le temps de changement de machine.

3.2.1.4 Informations sur le coût

L'information sur les coûts se subdivise en cinq catégories :

- Le coût d'utilisation d'une machine/un outil.
- Le coût d'utilisation d'un outil.
- Le coût de changement d'une machine ou d'un outil.
- Le coût de changement d'un outil.
- Le coût de changement de configuration.

3.2.2 Structures des données utilisées

<table>
<tr><td rowspan="4">Structure de données du produit</td><td colspan="2">Structures</td></tr>
<tr><td><u>Composant</u></td><td>• n : Cette valeur représente le nombre total des composants du produit.
• NOPk : Cette valeur représente le nombre d'opérations du composant k.</td></tr>
<tr><td><u>Opérations</u></td><td>• TNOP : Cette valeur représente le nombre totale des opérations nécessaires pour manufacturier un produit. Où :
$$TNOP = \sum_{k}^{n} NOP_k$$
• PS [1….NPS] : Liste représente la séquence de précédence des opérations.
• NPS : Est le nombre d'opérations possibles.</td></tr>
<tr><td><u>TADs</u></td><td>• OPTAD[1..TNOP] [1..6] : Matrice des opérations TADs. Où chaque opération TAD occupe une ligne dans la matrice OPTAD. Avec :
$$\text{OPTAD}\left[\text{OP}_i^k\right][\text{d}] = \begin{cases} 1, \text{si l'opération } \mathbf{i} \text{ de la composant } \mathbf{k}, \text{ besoin TAD } \boldsymbol{d} \\ 0, Sinon \end{cases}$$
d index représentant le TAD, d=1 .. 6, il prend l'une des positions des axes de direction.
<table>
<tr><td>d =</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr>
<tr><td>TAD dans la direction</td><td>X+</td><td>Y+</td><td>Z+</td><td>X-</td><td>Y-</td><td>Z-</td></tr>
</table>
</td></tr>
</table>

	Les outils requis	• **NT :** cette valeur représente le nombre d'outils disponibles • **OPT [1..TNOP] [1.. NT] :** Matrice des outils requis d'opérations. Chaque ligne dans OPT représente l'ensemble des outils pouvant effectuer une opération. $\text{OPTAD}\left[\text{OP}_i^k\right][\text{t}] = \begin{cases} 1, \text{si l'outil } \mathbf{t} \text{ peut effectuer l'opération } \mathbf{i} \\ \text{du composant } \boldsymbol{k} \\ 0, Sinon \end{cases}$
Structure de données des machines	Machines	• **NM :** Nombre de machines disponibles.
	Configurations	• **NC [1..NM] :** un tableau contient le nombre de configurations disponible pour chaque machine. • **CTAD[1..NM][1-MAX(NC)][6] :** Matrice des TADs disponibles par configuration de machine. $\text{CTAD}[\text{m}][\text{c}][\text{d}] = \begin{cases} 1, \text{Si la configuration } \mathbf{c} \text{ de la machine } \mathbf{m} \\ \text{offre le TAD } \mathbf{d} \\ 0, Sinon \end{cases}$ ❖ m= [1..NM] : index de la machine ❖ c = [1..MAX(NC)] : index de la configuration. ❖ d = [1..TAD] : index du TAD.
	Outils	• **MT[1-NM][1-NT] :** la matrice des outils pour chaque machine. • **NT:** le nombre des outils disponibles. $MT[m][t] = \begin{cases} 1, \text{si la machine } \mathbf{m} \text{ peut utiliser l'outil } \mathbf{t} \\ 0, Sinon \end{cases}$
	Fiabilité	• $\boldsymbol{fiabilit}$é $[\mathbf{1}..\boldsymbol{NM}]$: représente la fiabilité de chaque machine.
Structure de données des coûts	Coût d'utilisation des machines	• $\boldsymbol{CM}$ $[\mathbf{1}..\boldsymbol{NM}]$: représente le coût d'utilisation d'une machine par opération par unité de temps.
	Coût de changement de machines	• $MCCost[1..NM\,][1..NM\,]$: c'est la matrice des coûts de changement de machines $MCCost[a][b] = C$, donc : **C** est le coût de changement d'une machine **a** par la machine **b** pour effectuer les deux opérations consécutives.
	Coût de reconfiguration d'une machine	• **CCCost[1..NM][1..MAX(NC)][1..MAX(NC)]:** Cette matrice représente le coût de changement d'une configuration dans la même machine. $\text{CCCost}[\text{M}][\text{A}][\text{B}] = \text{Y}$, le **Y** représente le coût de passage de la configuration **A** à la configuration **B** pour effectuer deux opérations successives sur la même machine **M**.
	Coût d'utilisation des outils	• $\boldsymbol{CT}[\mathbf{1}..\boldsymbol{NT}]$: Cette matrice représente le coût d'utilisations des outils

	Coût de changement des outils	• $\boldsymbol{TCCost}[\mathbf{1}..\boldsymbol{NT}][\mathbf{1}..\boldsymbol{NT}]$: C'est le coût de passage d'un outil à un autre. $TCCost[A][B] = Y.\mathbf{Y}$ c'est le coût de passage de l'outil **A** à l'outil **B** pour effectuer deux opérations successives sur la même machine.
Structure de données des temps	Temps de changement de configurations	• $\boldsymbol{CCTime}[\mathbf{1}..\boldsymbol{NM}][\mathbf{1}..\boldsymbol{MAX}(\boldsymbol{NC})][\mathbf{1}..\boldsymbol{MAX}(\boldsymbol{NC})]$: C'est le temps nécessaire pour passer de configuration à une autre sur la même machine.
	Temps de changement d'outils	• $\boldsymbol{TCTime}[\mathbf{1}..\boldsymbol{NT}][\mathbf{1}..\boldsymbol{NT}]$: C'est le temps de changement d'outils sur la même machine pour effectuer une opération
	Temps de changement de machines	• $\boldsymbol{MCTime}[\mathbf{1}..\boldsymbol{NM}]$ $[\mathbf{1}..\boldsymbol{NM}]$: C'est le temps nécessaire pour un produit afin de passer de machine à une autre.
	Temps de traitement	• $\boldsymbol{PrTime}[\mathbf{1}..\boldsymbol{n}][\mathbf{1}..\boldsymbol{MAX}(\boldsymbol{NOP})][\mathbf{1}..\boldsymbol{NM}]$: C'est le temps nécessaire pour effectuer une opération particulière sur une machine avec l'une de ses configurations à l'aide d'un outil approprié.

Tableau 3.1: Structure de données utilisée.

4 Fonctions Objectifs

Dans ce projet nous tenteront d'optimiser conjointement trois grands axes : le coût total, la fiabilité et le temps total suivant le modèle de [2],[3]. Un seul modèle multicritère sera par la suite résolu en se basant sur les techniques de méta-heuristiques.

Donc nous fixerons ces trois objectifs :

1. Le temps total (minimiser).
2. Le coût total (minimiser).
3. La fiabilité totale (maximiser).

4.1 Temps total

Le temps total est englobe les différents temps nécessaires pour la réalisation de l'ensemble des opérations donnant lieu au produit fini.

Les composants des temps total sont les suivantes :

4.1.1 Temps emps de changements des machines (MCT)

C'est le temps nécessaire pour le passage d'une machine **j** a la machine **j'**, afin d'effectuer deux opérations successives différentes (u et u').

Où **u** doit être effectué avant **u'**.

Son expression est donnée comme suit :

$$\mathrm{MCT} = \sum_{\mathrm{u}=1}^{\mathrm{TNOP}} \mathrm{MCTime}[Mj\ (\mathrm{u})]\ [\mathrm{Mj}\ '(\mathrm{u}')]$$

4.1.2 Temps de changements d'outils (TCT)

C'est le temps de passage d'un outil à un autre outil, il génère si deux opérations successives sur la même machine besoins deux outils.

Son expression est donnée comme suit :

$$\mathrm{TCT} = \sum_{\mathrm{u}=1}^{\mathrm{TNOP}} \mathrm{TCTime}[\ \mathrm{T}_q^j(\mathrm{u})][\mathrm{T}_{q'}^j(\mathrm{u}')]$$

4.1.3 Temps de changement des configurations (CCT)

C'est le temps de passage d'une configuration à une autre sur la même machine. Il dépend de la machine et les 2 configurations.

Son expression est donnée comme suit :

$$\mathrm{CCT} = \sum_{\mathrm{u}=1}^{\mathrm{TNOP}} \mathrm{CCTimr}[c_i^j\ (\mathrm{u})]\ [c_{i'}^j\ (\mathrm{u}')]$$

4.1.4 Temps de traitement (PT)

C'est le temps nécessaire pour effectuer une opération particulière sur une machine. Il dépend de la machine, la configuration adoptée ainsi que l'outil d'usinage utilisé.

Son expression est donnée comme suit:

$$PT = \mathrm{PrTime}[\ \mathrm{Mj}\ (\mathrm{u})][\ \mathrm{C}_i^j\ (\mathrm{u})][\mathrm{Tj}\ (\mathrm{u})]$$

Objectif 1 : Minimiser le temps total (Total Time)

Le premier objectif est égal à la somme des objectifs ci-dessus.

$$\mathbf{Temps\ total\ =\ CCT\ +\ T\ CT\ +\ MCT\ +\ P\ T}$$

4.2 Coût total

Le calcul du coût total d'une ligne de production nécessite de calculer les cinq coûts suivants:

4.2.1 Coût d'utilisation des machines(MUC)

C'est le coût d'utilisation des machines pour effectuer toutes les opérations d'une production.

Le MUC est exprimé comme suit :

$$\mathrm{MUC} = \sum_{\mathrm{u}=1}^{\mathrm{TNOP}} \mathrm{CM}[\mathrm{Mj}\ (\mathrm{u})]\ \times\ \mathrm{PrTime}[\ \mathrm{Mj}\ (\mathrm{u})][\ \mathrm{C}_i^j\ (\mathrm{u})][\mathrm{Tj}\ (\mathrm{u})]$$

Avec:

- $Mj\ (u)$: La machine Mj effectuant l'opération d'index u

- $CM[Mj\ (u)]$: Coût unitaire de l'utilisation de la machine Mj
- $C_i^j\ (u)$: La configuration de la machine utilisée pour effectuer l'opération d'index u.
- $Tj\ (u)$: L'outil utilisé pour effectuer l'opération d'index u.
- $PrTime\big[\ Mj\ (u)C_i^j\ (u)Tj\ (u)\big]$: Temps requis pour achever l'opération.

4.2.2 Coût de changement de machine (MCC)

Il s'agit du coût du passage d'une machine **j** à la machine **j'**, pour effectuer deux opérations successives différentes (**u** et **u'**), où **u** doit être effectué avant **u'**.

Le MCC est exprimé comme suit :

$$\mathrm{MCC} = \sum_{\mathrm{u=1}}^{\mathrm{TNOP}} \mathrm{MCCOST}[Mj\ (\mathrm{u})]\ [\mathrm{Mj}\ '(\mathrm{u}')]$$

4.2.3 Coût de changement de configuration(CCC)

C'est le coût nécessaire pour changer les configurations des machines.

Son expression est donnée comme suit :

$$\mathrm{CCC} = \sum_{\mathrm{u=1}}^{\mathrm{TNOP}} \mathrm{CCCOST}[c_i^j\ (\mathrm{u})]\ [c_{i'}^j\ (\mathrm{u}')]$$

L'opération **u** doit être effectuée avant **u'** sur la même machine. CCC correspond à la somme du coût de changement de configuration (CCC_i) de chaque machine dans la ligne de production.

4.2.4 Coût d'utilisation des outils(TUC)

L'expression du coût d'utilisation des outils est donnée comme suit :

$$\mathrm{TUC} = \sum_{\mathrm{u=1}}^{\mathrm{TNOP}} \mathrm{CT}[\mathrm{T}(\mathrm{u})]\ \times\ \mathrm{PrTime}[\ \mathrm{Mj}\ (\mathrm{u})][\mathrm{C}_i^j\ (\mathrm{u})][\mathrm{Tj}\ (\mathrm{u})]$$

Avec:

- $T(u)$: L'outil utilisé pour effectuer l'opération d'index u.
- $Mj\ (u)$: La machine Mj effectuant l'opération d'index u.
- $C_i^j(u)$: La configuration d'index **i** de la machine d'index **j** effectuant l'opération d'index u.
- $PrTime[\ Mj\ (u)][C_i^j\ (u)][Tj\ (u)]$: Temps requis pour achever l'opération u.

4.2.5 Coût de changement d'outils(TCC)

C'est le coût de changer d'outil dans une machine particulière pour effectuer les différentes opérations sur la même machine. Puisque les différentes opérations peuvent exiger des outils de différents types.

Son expression est donnée comme suit:

$$\mathrm{TCC} = \sum_{\mathrm{u=1}}^{\mathrm{TNOP}} \mathrm{TCCOST}[\ \mathrm{T}_q^j(\mathrm{u})][\mathrm{T}_{q'}^j(\mathrm{u}')]$$

Objectif 2 : Minimiser le coût total (Total Cost)

Tous les coûts nécessaires sont calculés. Le deuxième objectif relatif au coût est exprimé sous la forme suivante :

$$\textbf{Coût total} = \textbf{MUC} + \textbf{MCC} + \textbf{CCC} + \textbf{TUC} + \textbf{TCC}$$

4.3 Fiabilité totale

Une fiabilité totale est généralement représentée par le pourcentage de la fiabilité d'une ligne de production nécessaire pour la réalisation de l'ensemble des opérations donnant comme résultat le produit fini.

4.3.1 Fiabilité de machine(MF)

Une fiabilité de machine, comme son nom l'indique, est considérée comme le pourcentage de fiabilité pour chaque machine disponible.

$$MF = MachineFiabilité[\ M_i]$$

Objectif 3 : Maximiser la fiabilité totale (Total Reliability)

L'objectif relatif à la fiabilité est exprimé sous la forme suivante :

$$\textbf{Fiabilité totale} = \sum_{i=1}^{NTOP} \textbf{MachineFiabilité}[\ \mathbf{M}_i] \ / \textbf{NBL}$$

Avec :

- NBL : Est le nombre de machines dans la ligne de production.

5 Représentation de l'individu (la solution)

Comme représenté dans le chapitre II dans la section (Ligne de production), c'est la ligne de production qui est responsable d'affecter à chaque opération une machine capable de l'exécuter. Pour permettre la manipulation et l'exécution des lignes de production on doit fournir la séquence d'opérations avec la précision de quel composant. Puis, on doit générer les séquences de machines, configuration et outils sou une forme d'une matrice comme a été proposé dans des travaux précédents [2], [3], où cette matrice est de taille M x N, où M est égale à 3 (Machines, Outils, Configurations), et N dépend du nombre total des opérations.

Cette matrice est interprétée, ci-dessous, de gauche à droite colonne par colonne.

Machine	M2	M1	M1	M2	M2	M3	M1
Configuration	C3	C2	C1	C1	C1	C1	C2
Outil	T2	T1	T3	T2	T3	T1	T4

Tableau 3.2: Individu sous forme d'une matrice.

Dans cet exemple, le produit est caractérisé par trois composants dont F1 a besoin de trois opérations pour le finir, F2 en a besoin de deux pour le finir et F3 en a besoin de deux pour le même but.

La première colonne doit être lue comme suit : l'opération O1 du composant F1 est effectuée sur la machine M2, avec la configuration C3, en utilisant l'outil T2. Ainsi les colonnes suivantes sont interprétées de la même manière.

6 Codages et décodages

Il est impératif que le codage/décodage soit soigneusement sélectionné pour éviter les cas des solutions non faisable.

6.1 Codages du l'individu (la solution)

L'étape clé dans l'implémentation des méta-heuristiques est bien le codage des solutions. Celui-ci consiste à passer de la représentation réelle des solutions vers une représentation codée. Le but du codage des solutions est de rendre les solutions plus maniables par l'algorithme de recherche.

Une solution (individu ou chromosome) générée par l'algorithme représente la ligne de fabrication qui est considérée comme une matrice **M** × **N** de nombres réels compris entre **0** et **1**, comme le propose [2] où **M** est égale à **3** (Machines, , Configuration, Outils), et **N** dépend du nombre total des opérations.

Exemple : une matrice de 3 × 6. Cette solution a 6 opérations pour l'accomplir.

0.25	0.51	0.92	0.21	0.54	0.90
0.63	0.53	0.03	0.39	0.76	0.23
0.50	0.71	0.17	0.44	0.50	0.08

Tableau 3.3: Individu codé en nombre réel.

Une fois les résultats obtenus, il faut passer à l'étape du décodage des solutions codées. Afin d'aboutir à cela, on recourt à des procédures de décodage c'est-à-dire pour : les machines, les configurations et les outils.

6.2 Décodages du l'individu

6.2.1 Décodage des machines

Le décodage des machines est la troisième étape du décodage d'une solution. Ce décodage doit être effectué après celui des composants et des opérations. La procédure est résumée comme suit:

1. Identifier parmi les machines reconfigurables, celles qui possèdent les TAD et les outils requis pour effectuer l'opération de la même colonne.
2. Créer une liste de la taille n des machines reconfigurables identifiées. n est le nombre des machines reconfigurables pouvant réaliser l'opération de la même colonne.
   Par hypothèse, les machines ayant le petit index sont placées dans la liste avant les autres. Par exemple $M1$ avant $M2$ avant $M3$...)
3. Multiplier n par le nombre réel se trouvant dans la solution codée, dans la cellule correspondante à la cellule à décoder. Cette étape renvoie un nombre reel p $(0 < p \leq n)$

4. Arrondir le nombre réel obtenu p au nombre entier immédiatement supérieur, où $1 \leq p \leq n$.
5. La machine de la position p dans la liste des machines candidates est placée dans la cellule en cours de décodage.
   Ce processus est ensuite répété jusqu'à ce que tous les éléments de la troisième ligne soient décodés.

6.2.2 Décodage des configurations

Le décodage des configurations est la quatrième étape du décodage d'une solution. Ce décodage est effectué après celui des machines. La procédure est résumée comme suit :

1. Identifier, parmi les machines déjà décodées par la procédure du décodage des machines, les configurations pouvant effectuer l'opération concernée en termes de $TADs$ et d'outil.
2. Ces configurations sont classées dans une liste de taille n. Où n est le nombre de configurations identifiées.
   Par exemple, si parmi 7 configurations de la machine, les configurations $C1, C2\ et\ C5$ sont capables d'effectuer l'opération concernée, alors la liste comprend $C1- C2 - C5$ en tenant compte que $n = 3$. Par hypothèse, les configurations ayant le petit index sont placées dans la liste avant les autres. Par exemple ($C2$ avant $C3$).
3. Multiplier n par le nombre réel se trouvant dans la solution codée, dans la cellule correspondante à la cellule à décoder. Cette étape renvoie un nombre reel $p\ (0 < p \leq n)$
4. Arrondir le nombre réel obtenu p au nombre entier immédiatement supérieur, où $1 \leq p \leq n$.
5. La configuration de la position p dans la liste des configurations candidates est placée dans la cellule en cours de décodage. Ce processus est ensuite répété jusqu'à ce que tous les éléments de la quatrième ligne soient décodés.

6.2.3 Décodage des outils

Le décodage des outils est la dernière étape du décodage d'une solution. Après cette étape, on peut obtenir un individu (une solution) qui pourrait être évalué.

Le décodage des outils est réalisé à travers les étapes suivantes :

1. Identifier parmi les outils :
   - ceux qui sont disponible pour la machine concernée.
   - ceux qui ont la capacité d'effectuer l'opération à réaliser.
2. Ces outils sont classés dans une liste de taille n. Où n est le nombre des outils identifiés. Par hypothèse, les outils ayant le petit index sont placés, dans la liste, avant les autres.
   Par exemple (T1 avant T2).
3. Multiplier n par le nombre réel se trouvant dans la solution codée, dans la cellule correspondante à la cellule à décoder. Cette étape renvoie un nombre réel $p\ (0 < p \leq n)$
4. Arrondir le nombre réel obtenu p au nombre entier immédiatement supérieur, où

$$1 \leq p \leq n.$$

5. L'outil de la position p dans la liste des outils candidates est placé dans la cellule en cours de décodage.
   Ce processus est ensuite répété jusqu'à ce que tous les éléments de la cinquième ligne soient décodés.

7 Adaptation de NSGA-III

7.1 Fonctionnement

L'algorithme génétique de tri non dominé III (NSGA- III) a été conçu pour résoudre une multitude de problèmes multi-objectifs. Dans notre travaille on va l'adapter dans le but de se procurer le meilleur arrangement des machines (RMTs) et générer, par la suite, des lignes de productions optimales dans l'environnement des systèmes de manufacturiers reconfigurables. Afin d'atteindre cet objectif, on a adapté le NSGA- III comme un algorithme évolutionnaire.

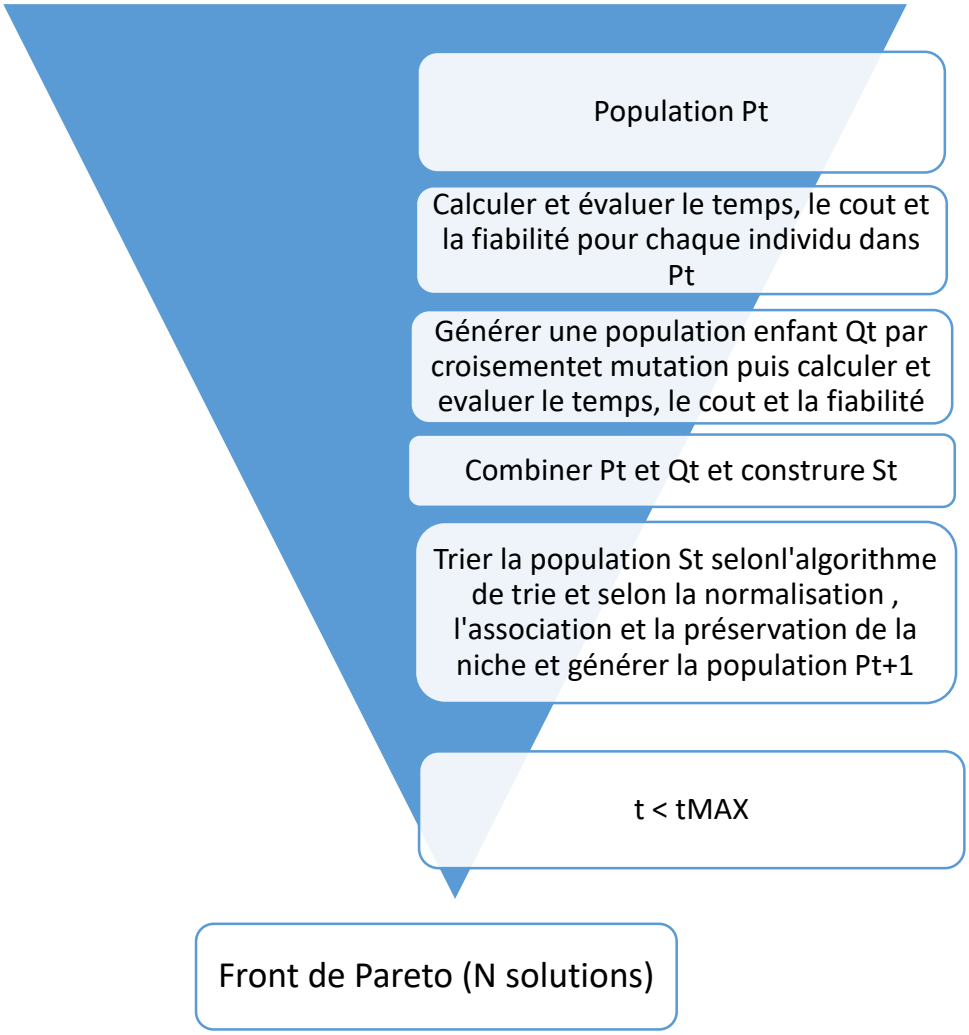


Figure 3.5: Algorithme NSGA-III proposé.

Dans cette section, on va expliquer en détail le déroulement de cette méta-heuristique proposée ci-dessus.

- L'étape préliminaire de l'algorithme NSGA- III est la génération aléatoire de la population P de N individus.
- À toute génération t, la population enfants (par exemple, Q_t) est d'abord créée en utilisant la population parent (par exemple, P_t) et les opérateurs génétiques habituels.
- Puis, les deux populations sont combinées pour former une nouvelle population (par exemple, R_t) de taille $2N$. Cette population (R_t) subit ensuite un tri en utilisant le concept du non dominance de Pareto.
- Les individus de la population (R_t) sont regroupés dans des fronts de non-dominance tels que $F1$ représente les individus de rang *1,* $F2$ les individus de rang 2, … etc. L'objectif est de réduire le nombre d'individus de *2N* dans la population R_t pour obtenir une population R_{t+1} de taille N.
- Si la taille de $F1$ est inférieure à N, alors tous les individus de $F1$ sont conservés. Il en est de même pour les autres fronts tant que le nombre d'individus conservés ne dépasse pas la taille N. Par exemple on a les fronts $F1$ et $F2$ sont intégralement conservés mais la conservation du front $F3$ va entraîner un dépassement de la taille N de la population P_{t+1}. Dans ce cas l'algorithme NSGA-III fait intervenir un mécanisme de préservation basé sue les point de référence comme illustré dans le chapitre I à condition que tous les fronts qui dépassent à N sont alors supprimés.
  Ces étapes sont répétées jusqu'à ce que le critère de terminaison soit satisfait.
- Lorsque le nombre de générations est satisfaisant, on peut obtenir de la dernière génération le front de Pareto qui regroupe les meilleurs individus de la population.

Dans les sections suivantes on va détailler beaucoup plus les étapes de l'algorithme NSGA- III

7.2 Génération de la population initiale

On génère une population initiale d'une façon aléatoire selon le nombre d'individus codés par des valeurs réels] 0-1[définis soit disant N. Ensuite, Ces individus codés doivent être décodés pour le même individu pour permettre l'évaluation. C'est à dire pour calculer le temps total (f1) et le coût total (f2) et la fiabilité totale (f3).

7.3 Evaluation de la population

Le calcul des fonctions fitness (f1, f2 et f3), en ce qui nous concerne: la minimisation du temps total de production (temps total), la minimisation du coût total de production (Coût total) et la maximisation de la fiabilité totale, nous exige de suivre les différentes phases de décodages citées ultérieurement et qui respecte les contraintes d'affectation des opérations aux machines. Les valeurs f1, f2 et f3, obtenues par la réalisation des différents calculs, vont déterminer la qualité des solutions obtenues et donneront accès à l'ensemble des solutions à prendre en compte parmi les différents fronts de Pareto.

7.4 Fonction de tri rapide non dominée (Fast Non-dominated Sorting)

Cette technique renvoie une population classée (F) avec plusieurs fronts $(F1, F2, \dots..)$ en fonction des trois objectifs précédemment fixés, minimiser le coût, minimiser le temps et maximiser la fiabilité.

Conséquemment, pour concevoir cette population classée, c'est à dire savoir quelles solutions feront parties de chaque front de Pareto, il faut réaliser un classement des résultats obtenus par chaque séquence de chromosomes générés. Chaque solution doit être comparée à chaque solution de la population pour s'assurer qu'elle est dominée. Dans la première étape, nous calculons deux entités : 1) Compteur de domination Np, qui représente le nombre de solutions qui dominent la solution p et 2) Sp qui est l'ensemble de solutions que la solution p domine.

Toutes les solutions qui font parties du premier front non-dominé doivent avoir leur compteur de domination égal à zéro. Partant de là, pour chaque solution p avec $Np = 0$, nous visitons chaque membre (q) dans son ensemble Sp et nous réduisons le compteur de domination. Pendant cette réduction, si un membre q du compteur de domination devient zéro, nous l'ajoutons dans une liste Q. Ces membres appartiennent au deuxième front non dominé.

Cette procédure décrite est faite avec chaque membre de Q et le troisième front est identifié suivant la même approche. Cette démarche continue jusqu'à l'identification de tous les fronts. Pour chaque solution p dans le deuxième ou le plus haut niveau de non-domination, le compteur de domination Np peut être égal au moins à $N - 1$ ($N = taille\ de\ la\ population$). Alors chaque solution p sera visitée au moins $N - 1$ fois avant que son compteur de domination soit égal à zéro. A ce moment-là, la solution est affectée à un niveau de non-domination et elle ne sera pas de nouveau visitée [44].

7.5 Mise à jour de la population

Dans cette étape, avant de commencer la création de la population enfants (Q_t), la population parent (P_t) doit être évaluée (calcul du coût total et du temps total et la fiabilité totale de chaque individu dans la population P_t). Après cela, il faut identifier ses fronts non dominés (F) c'est-à-dire $F1, F2, \dots . ect$.

Une fois la population enfants (Q_t) est générée. On applique la sélection de tournois binaires, le croisement et la mutation pour avoir une population totale (R_t). Cette dernière sera classée en tenant compte les différents concepts déjà mentionnés en termes de fronts , génération de point de référence, la normalisation, l'association, et la préservation de la niche, sont des opérateurs qui ont été détaillé dans le chapitre I dans le but de conserver un nombre constant d'individus pour chaque génération (notre taille de la population $= N$). Le processus continu d'une génération à la suivante jusqu'à ce que le nombre maximal de générations est atteint.

7.6 La sélection

La sélection détermine quelles solutions de la population actuelle seront préservées pour faire le croisement. Dans cette méthode, la sélection est faite selon la méthode de tournoi, où l'on confronte de deux individus de la population initiale qui sont choisis au hasard afin de stocker les individus qui appartiennent au meilleur front de Pareto. Dans le cas où les deux solutions

confrontées seraient membres du même front, la décision sera prise à partir du critère de la crowding distance [38].

7.7 Le croisement

L'opérateur du croisement produit deux enfants en prenant deux parents et un point de croisement comme paramètre. Chaque parent est coupé en deux fragments au même point (point de croisement), puis chaque enfant prend le fragment avant le point de croisement d'un parent, et le fragment après le point de croisement de l'autre parent. De cette manière la qualité des enfants (les fonctions d'objectifs) sera déterminée en grande partie de la qualité des parents.

Dans notre méta-heuristique, pour générer les enfants on utilise un type du croisement à un point avec une probabilité Pc comme illustré dans la figure suivante :

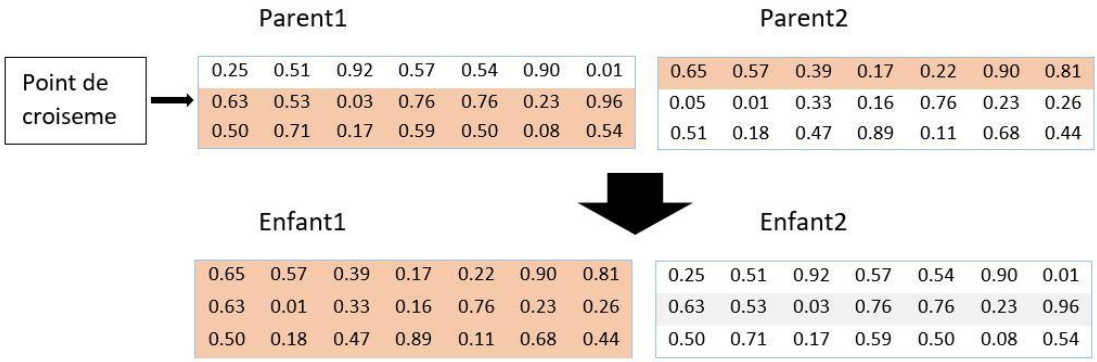


Figure 3.6: Croisement à un point.

7.8 La mutation

La mutation vise à perturber les enfants générés avec une probabilité P_m en modifiant ses gènes pour participer à la population d'enfants.

8 Conclusion

Dans ce chapitre nous avons élaboré la conception de notre projet qui s'articule autour de deux grands axes (la conception globale et la conception détaillée). Ainsi, l'énumération de nos principaux objectifs nous a facilité l'identification et la présentation du problème de génération lié à la ligne de production dans le cadre des systèmes reconfigurables. Pour parvenir à résoudre ce genre de problème la méta-heuristique multicritère NSGA-III a été proposée puis exécutée. Cela nous a permis par la suite de rendre faisable le modèle adopté.

CHAPITRE 4
Implémentation et Résultats

1 Introduction

Après avoir expliqué et détailler la conception générale de notre projet dans le chapitre précèdent, le passage à l'étape suivante dite implémentation nous est obligatoire pour aboutir à nos objectifs.

Dans ce chapitre final, nous allons présenter les différents outils de développement ainsi que les langages appris et exploités dans la réalisation de notre projet. Après, nous réaliserons un exemple illustratif sur notre application pour l'algorithme traité ultérieurement (NSGA- III). A la fin, nous exposerons et discuterons quelques résultats expérimentaux.

2 Langage de programmation et outils de développement

Pour pouvoir réaliser notre projet nous avons fait recours à des outils et des langages adéquats qui étaient choisis soigneusement. Ces derniers sont énumérés avec une brève définition, pour chacun d'eux, dans ce qui suit :

2.1 Langage de programmation Python

Python est un langage de programmation, créé par Guido van Rossumont, dont la première version est apparue en 1991. Python est un langage puissant, à la fois facile à maîtriser et riche en possibilités. Dès son installation sur ordinateur par exemple, l'utilisateur dispose d'une multitude de fonctionnalités intégrées au langage.

Ce langage de programmation bénéficie d'une licence libre. Il fonctionne sur la plupart des plates-formes informatiques, des smartphones aux ordinateurs. Il est disponible pour tous ces systèmes d'exploitation (Windows, LINUX, Mac OS). Il est conçu pour optimiser la productivité des programmeurs en offrant des outils de haut niveau et une syntaxe simple à utiliser.

Figure 4.1: Logo de python.

2.2 L'environnement de programmation Pycharme

PyCharm est un environnement de développement intégré utilisé pour programmer en Python.

Il permet l'analyse de code et contient un débogueur graphique. Il permet également la gestion des tests unitaires, l'intégration du logiciel de gestion des versions, et supporte le développement web avec Django.

Développé par l'entreprise tchèque JetBrains, c'est un logiciel multi-plateforme qui fonctionne sous Windows, Mac OS X et Linux. Il est décliné en édition professionnelle.

Figure 4.2: Logo de PyCharm.

2.3 Le Package (Tkinter) Tool kit Interface

Le paquet tkinter (interface Tk) est l'interface Python standard de la boîte à outils d'IUG Tk. Tk et tkinter sont disponibles sur la plupart des plates-formes Unix, ainsi que sur les systèmes Windows.

Figure 4.3: Logo de TKinter.

3 Présentation des interfaces de notre système

Notre système contient un ensemble d'interfaces facilitant à l'utilisateur la saisie des données. Ces interfaces sont détaillées dans le passage suivant :

3.1 Interface principale

Nous avons développé de nombreuses interfaces graphiques pour faciliter l'utilisation de l'application.

Notre Interface principale offre de nombreuses fonctionnalités à l'utilisateur :

- Créer un nouveau problème (informations sur les machines et le produit).
- Sauvegarder les différentes données saisies au format CSV qui sont des fichiers lisibles tout en offrant la possibilité de les consulter.
- Sans oublier la possibilité de lancement de l'algorithme NSGA-III.

La figure suivante illustre l'interface d'accueil de notre application.

Figure 4.4: Interface principale.

3.1.1 Bouton «Définir de nouvelles données»

Avec la commande “Définir de nouvelles données” l’utilisateur peut créer de nouvelles machines reconfigurables et par conséquent un produit en sélectionnant leurs configurations conformément à la formulation expliquée dans le chapitre III.

Les données d'entrée sont saisies dans la fenêtre illustrée par la figure suivante:

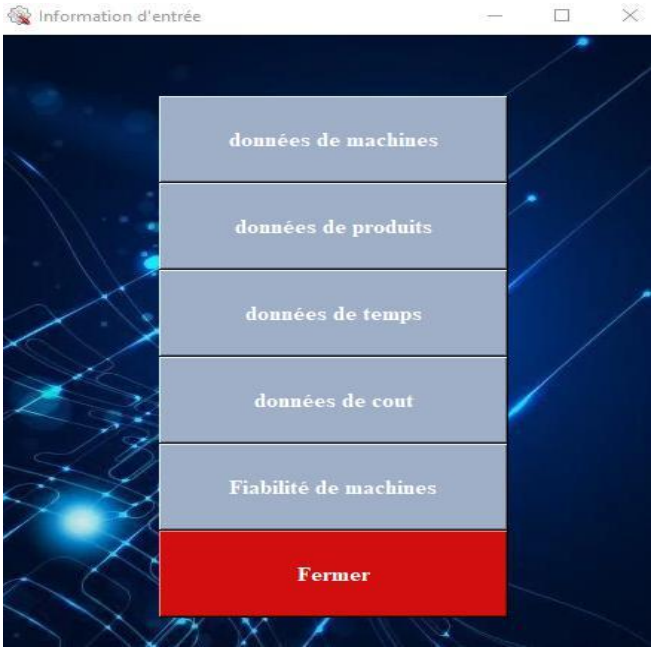


Figure 4.5: Interface des données d'entrée.

3.1.1.1 Interfaces des données de machines

La première étape des données d'entrée consiste à saisir les données de machines. Pour le faire, il suffit de cliquer sur le bouton «données de machines» et la nouvelle fenêtre (Figure 38) s'affiche pour permettre l'entrée de ces données.

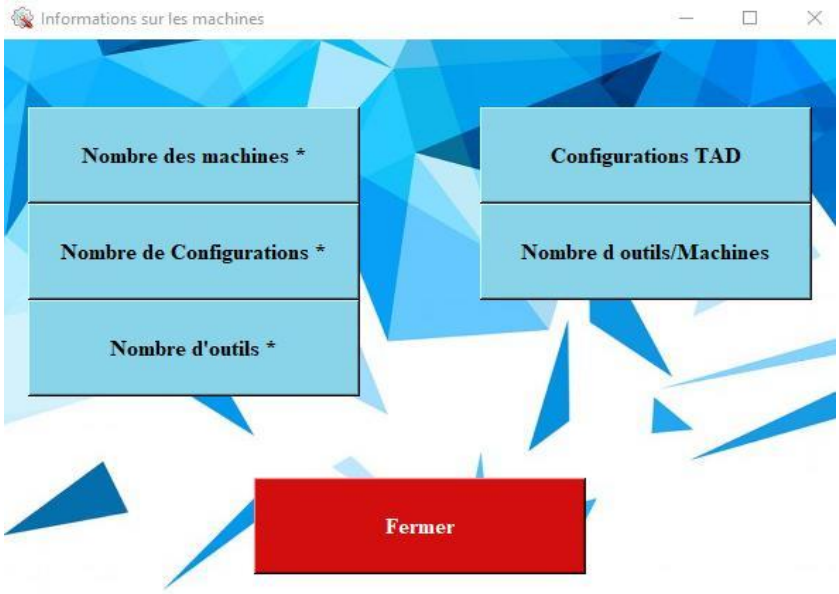


Figure 4.6: Interface des données de machines.

A partir de cette fenêtre l'utilisateur doit identifier le nombre de machines, les configurations, les outils, outils/machines, les outils disponibles et les valeurs des axes de mouvement pour chaque configuration d'une machine.

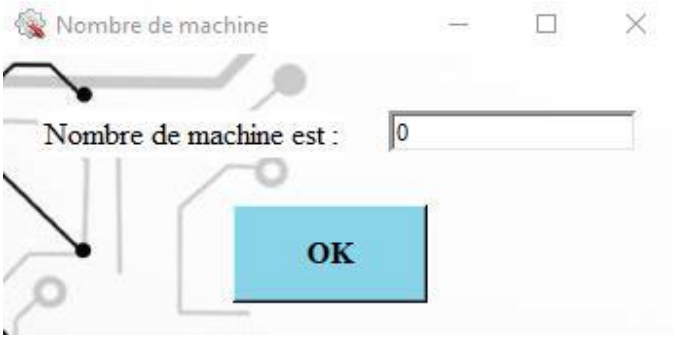


Figure 4.7: Interface nombre de machines.

Pour la configuration TAD « CTAD » l'utilisateur doit identifier les axes « X, Y, Z » de chaque configuration d'une machine. Voir la figure suivante.

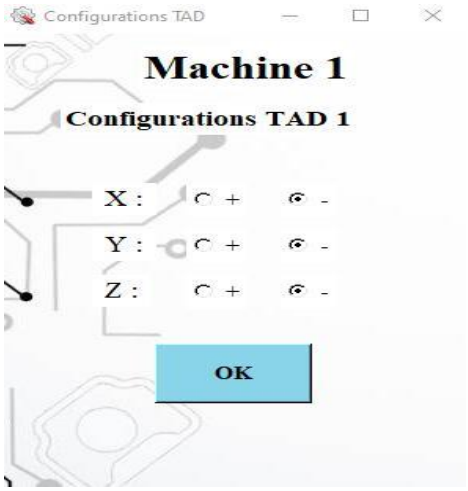


Figure 4.8: Interface de la configuration TAD.

L'interface Nombre d'outil / Machine permet à l'utilisateur de sélectionner l'ensemble d'outils disponibles de chaque machine. Voir la figure suivante.

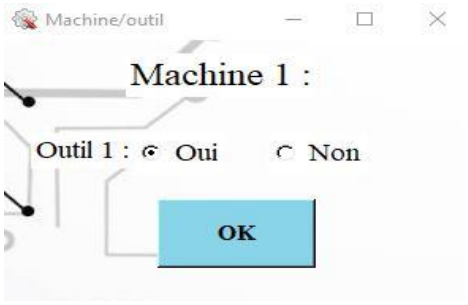


Figure 4.9: Interface machine/outil.

3.1.1.2 Interfaces des données du produit

La deuxième étape de la saisie des données d'entrée consiste à entrer les données du produit. En cliquant sur le bouton «données du produit» une nouvelle fenêtre (Figure 4.10) s'affiche permettant la sélection des choix appropriés au produit défini.

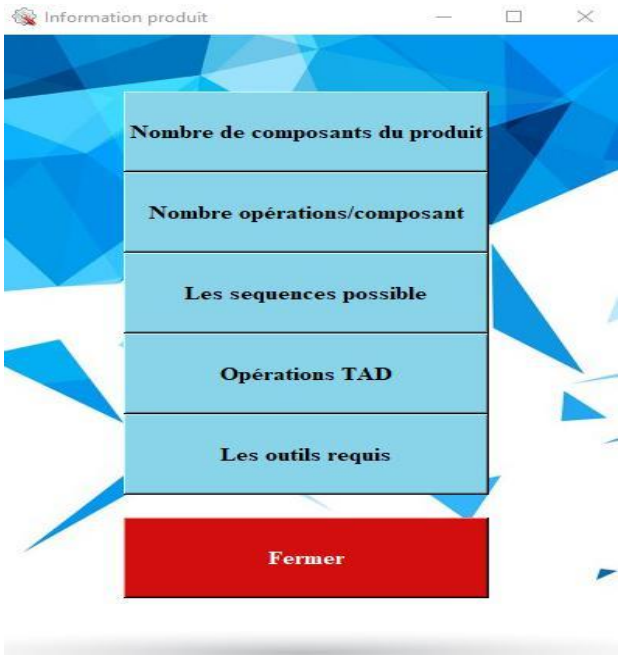


Figure 4.10: Interface des données de produit.

A partir de cette fenêtre l'utilisateur doit identifier le nombre des composants « composants » d'un produit, le nombre d'opérations de chaque composant « NOPC », les séquences d'opérations possibles et les opérations TAD.

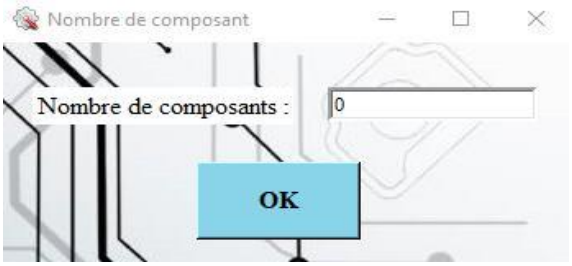


Figure 4.11: Interface pour définir le nombre de composants d'un produit.

A partir du bouton «Les séquences possibles», l'utilisateur doit saisir les séquences possibles pour les opérations d'un composant spécifié. La figure ci-dessous illustre une séquence possible du composant 1 qui possède 4 opérations.

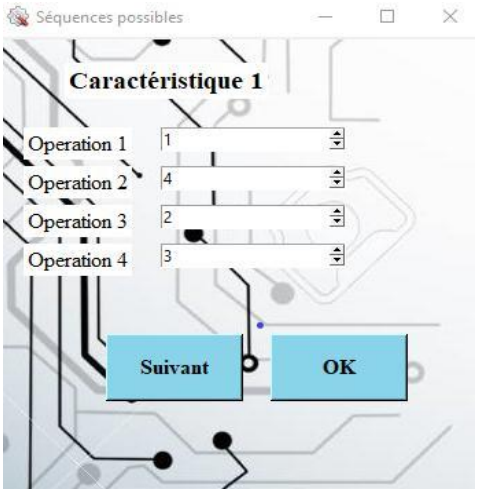


Figure 4.12: Interface des séquences possibles.

Ainsi le bouton «Opération TAD», offre à l'utilisateur la manipulation des opérations TAD. La figure 45 illustre le TAD de l'opération 2 du composant 2 dont les valeurs dimensionnelles successives sont : [0, 1, 0, 1, 0, 1].

Figure 4.13: Interface d'une operation TAD.

3.1.1.3 Interfaces des données de temps

La troisième étape permet à l'utilisateur de saisir toutes les données de temps grâce au bouton «données de temps».

Figure 4.14: Interface des données de temps.

- **Temps de changement d'outils**

Le temps total de changement d'outils correspond à la durée prise par l'ensemble des changements d'outils nécessaires à la réalisation d'un produit.

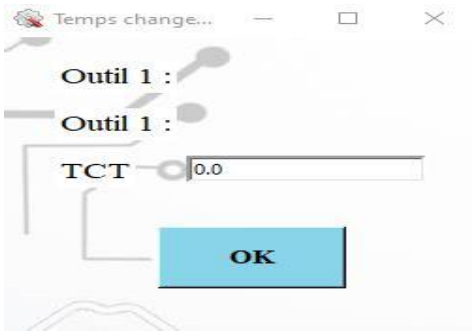


Figure 4.15: Interface de temps de changement d'outils.

- **Temps de changement des machines**

Le temps total de transfert est considéré comme la durée nécessaire pour transporter le produit d'une machine vers une autre jusqu'à l'obtention du produit fini.

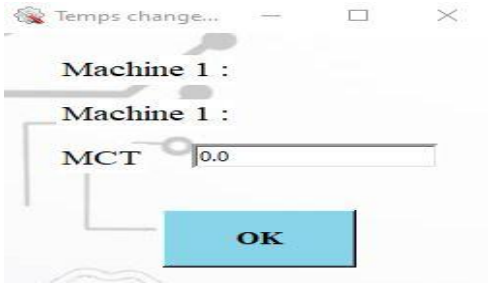


Figure 4.16: Interface de temps de changement des machines.

- **Temps de changement de configuration**

Le temps total de changement de configuration désigne la somme de toutes les durées consommées lors des changements individuels des configurations des machines dans le but de réaliser le produit défini.

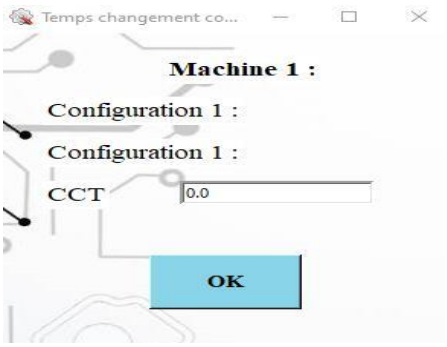


Figure 4.17: Interface de temps de changement de configuration.

3.1.1.4 Interfaces des données de coût

La quatrième étape permet à l'utilisateur de saisir toutes les données de coût grâce au bouton de «données de coût».

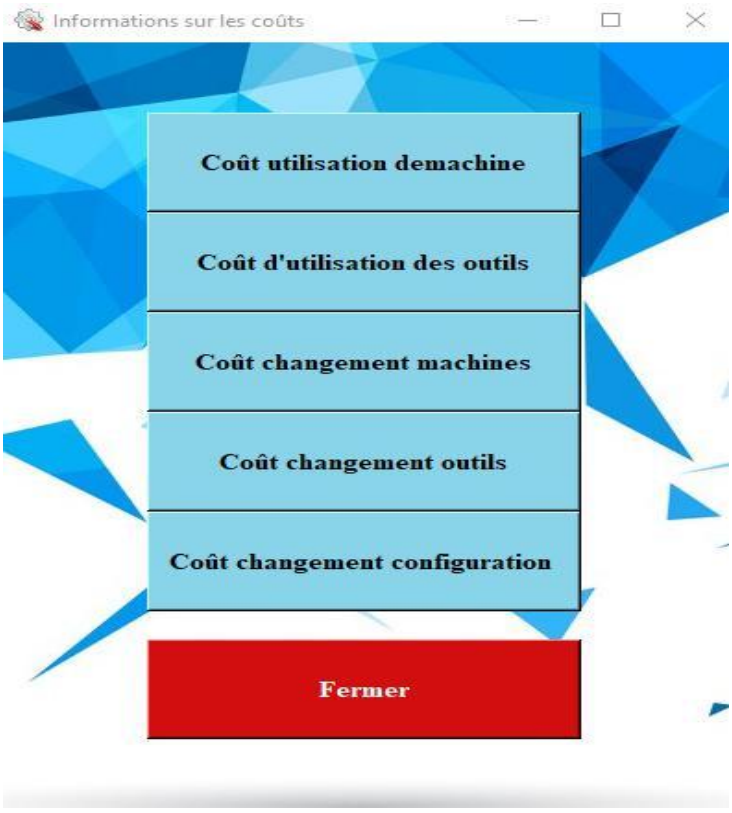


Figure 4.18: Interface des données de coût.

- **Coût d'utilisation de la machine**

La fenêtre (Figure 4.19) permet à l'utilisateur de saisir le coût d'utilisation de chaque machine.

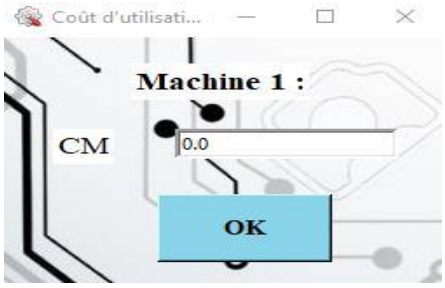


Figure 4.19: Interface de coût d'utilisation d'une machine.

- **Coût d'utilisation des outils**

Le coût total d'utilisation des outils inclus la somme des coûts consommés lors de l'utilisation de chaque outil jusqu'à l'obtention du produit fini. La fenêtre ci-dessous (figure 4.20) permet à l'utilisateur de saisir le coût d'utilisation de chaque outil.

Figure 4.20: Interface de coût de d'utilisation d'outils.

- **Coût de changement des machines**

Le coût total de changement de machines représente l'ensemble des coûts nécessaires pour transporter le produit d'une machine vers une autre. La fenêtre ci-dessous (Figure 4.21) permet à l'utilisateur de saisir les coûts de changement relatifs à chaque machine.

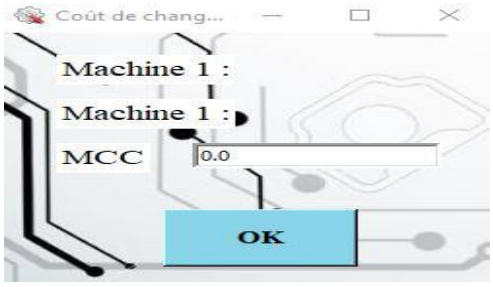


Figure 4.21: Interface de coût de changement des machines.

- **Coût de changements des outils**

Le coût total de changement d'outils correspond à la somme des coûts de changements d'outils nécessaires à la réalisation du produit. La fenêtre ci-dessous (Figure 4.22) permet à l'utilisateur de choisir les coûts de changement correspondant à chaque outil.

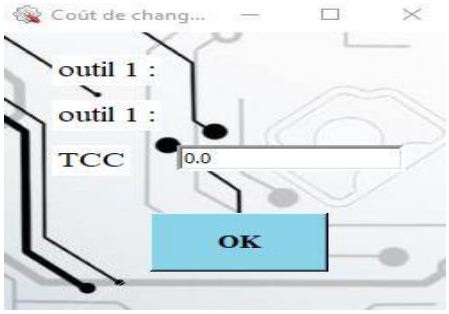


Figure 4.22: Interface de coût de changement d'outils.

- **Coût de reconfiguration d'une machine**

Le coût total de changements de configurations désigne l'ensemble des coûts générés par les actions de reconfiguration des machines.

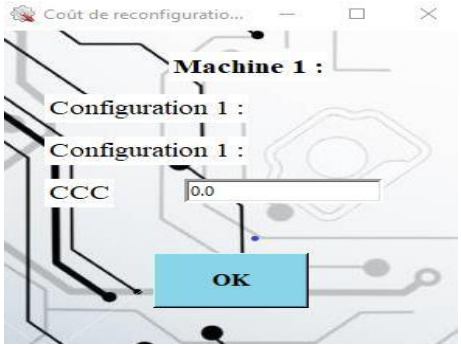


Figure 4.23: Interface de coût de reconfiguration d'une machine.

3.1.1.5 Interface de fiabilité des machines

La cinquième étape permet à l'utilisateur de saisir les données de fiabilité grâce au bouton de «Fiabilité de machine».

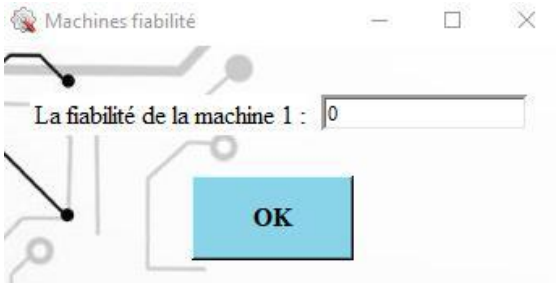


Figure 4.24: Interface fiabilité de machines.

3.1.2 Bouton «Ouvrir les fichiers des données»

La commande «Ouvrir les fichiers des données» permet à l'utilisateur de consulter les fichiers où les données sont stockées.

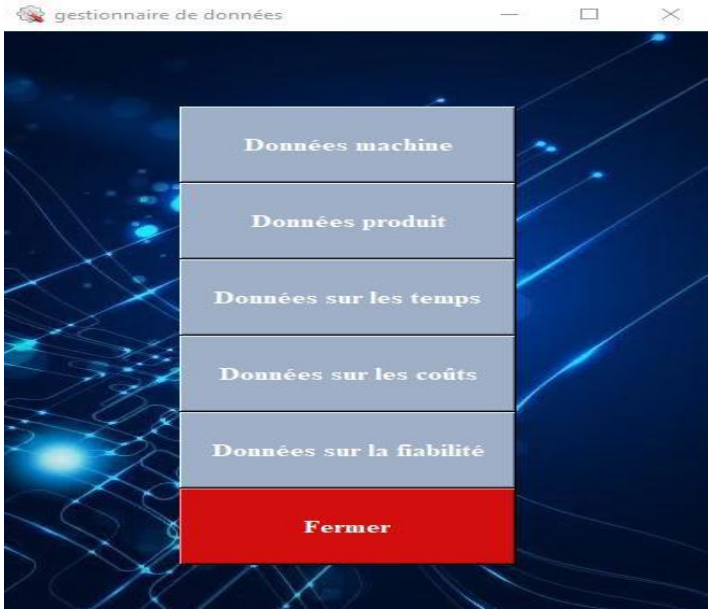


Figure 4.25: Interface du gestionnaire des données.

La commande NSGA-III permet à l'utilisateur de saisir les paramètres de NSGA-III et de l'exécuter.

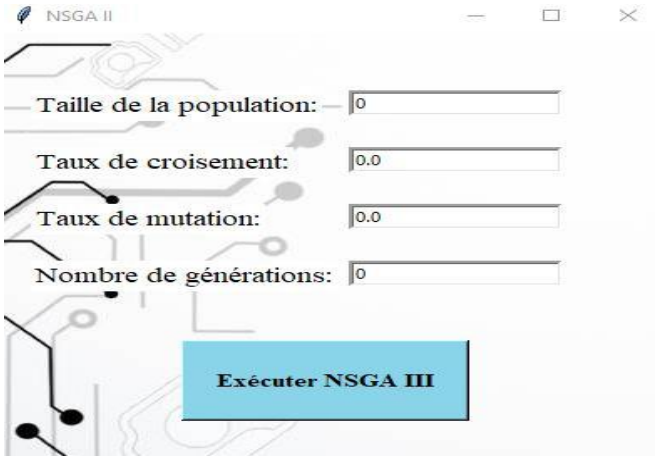


Figure 4.26: Interface du paramétrage de NSGA-III.

4 Evaluation des résultats

Dans notre cas d'étude, nous prenons un exemple illustratif pour tester le fonctionnement de notre application et évaluer les performances de l'algorithme implémentés. Cet exemple est inspiré des travaux de [33].

4.1 Etude de cas

Les données d'entrée de machines se composent de huit machines (M1, M2, M3, M4, M5, M6, M7, M8) qui partagent un ensemble de six outils (T1, T2, T3, T4, T5, T6). Le tableau 4.1, ci-dessous illustre les données d'entrée des machines qui incluent les configurations TAD et l'ensemble des outils disponibles pour chaque machine.

Machines	**Configurations**	**TADs**						**Outils requis**
		X+	X-	Y+	Y-	Z+	Z-	
Machine 1	**C1**	1	0	0	1	0	1	**T1, T2, T3, T4, T5**
	C2	1	0	1	0	1	0	
	C3	0	1	0	1	1	0	
Machine 2	**C1**	0	1	1	0	1	0	**T3, T5, T6**
	C2	0	1	0	1	1	0	
	C3	1	0	1	0	1	0	
	C4	1	0	0	1	0	1	
Machine 3	**C1**	1	0	0	1	1	0	**T1, T4, T6**
	C2	0	1	1	0	0	1	
Machine 4	**C1**	1	0	0	1	0	1	**T1, T3**
	C2	0	1	1	0	0	1	
Machine 5	**C1**	1	0	0	1	1	0	**T1, T3, T4, T5, T6**
	C2	0	1	0	1	1	0	
	C3	0	1	0	1	1	0	
Machine 6	**C1**	0	1	1	0	1	0	**T2, T5, T1, T4, T6**
	C2	1	0	1	0	1	0	
	C3	0	1	1	0	0	1	
Machine 7	**C1**	1	0	0	1	0	1	**T1, T6**
	C2	0	1	1	0	1	0	
Machine 8	**C1**	0	1	1	0	1	0	**T2, T4, T5**
	C2	1	0	1	0	1	0	
	C3	1	0	0	1	1	0	
	C4	1	0	1	0	0	1	

Tableau 4.1: Configurations TADs et outils disponibles.

Dans cet exemple, le produit est composé de quatre composants, le premier composant a nécessité trois opérations à effectuer. Le deuxième et quatrième composants ont nécessité quatre opérations à effectuer. Cependant, le troisième a eu besoin de cinq opérations à effectuer. Donc, comme total on a 16 opérations à réaliser.

Et voici la séquence d'opérations et illustre dans ce tableau :

Composant	1	4	2	3	2	3	4	1	3	1	2	4	3	4	2	3
Operation	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	5

Tableau 4.2: séquence d'opérations à réaliser.

Le tableau suivant indique les données d'entrée du produit, à savoir les composants et leurs opérations TADs, ainsi que l'outil requis pour chaque opération.

Composants	**Opérations**	**TADs**						**Outils requis**
		X+	X-	Y+	Y-	Z+	Z-	
Composant 1	**O1**	1	0	0	1	0	1	**T1**
	O2	0	1	1	0	1	0	**T5**
	O3	1	0	1	0	1	0	**T2**
Composant 2	**O1**	1	0	0	1	1	0	**T4**
	O2	0	1	0	1	0	1	**T6**
	O3	0	1	0	1	1	0	**T1**
	O4	1	0	1	0	0	1	**T5**
Composant 3	**O1**	0	1	1	0	0	1	**T1**
	O2	0	1	0	1	1	0	**T5**
	O3	1	0	0	0	0	1	**T1**
	O4	0	1	0	1	1	0	**T5**
	O5	0	0	1	1	0	0	**T2**
Composant 4	**O1**	1	0	0	1	0	1	**T3**
	O2	1	0	0	1	1	0	**T4**
	O3	0	1	0	1	1	0	**T1**
	O4	0	1	1	0	1	0	**T6**

Tableau 4.3: Opérations TAD et outils requis.

Comme on a vu dans le chapitre III, les informations du temps se composent de 4 entités (temps de traitement, temps de changement de machines, temps de changement d'outils, temps de changement de configuration).

Nous présentons les données de chacune d'entre elles dans les tableaux suivants :

- **Le temps de traitement**

Pour savoir le temps de traitement d'une opération sur une machine, on est obligé de prédéfinir le type d'opération, type de configuration et l'outil utilisé. Voir le tableau suivant.

Composants	**Opérations**	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**
Composant 1	**Op1**	580		471			286		
	Op2	795	112	365					324
	Op3		330		465				
Composant 2	**Op1**	312						619	
	Op2	246	222			357			
	Op3		618	910			142		
	Op4	815		333	296			524	

Composant 2	**Op1**	345	112			222			
	Op2	114						111	312
	Op3	609		90			24		
	Op4		535			84	204		
	Op5		449						102
Composant 3	**Op1**	802			416			662	
	Op2	742		123					
	Op3	494	312						601
	Op4	741			395			108	96

Tableau 4.4: Temps de traitement.

- **Le temps de changement d'outils**

Le temps de changement d'outil de notre exemple est résumé dans le tableau suivant.

Outils / Outils	**Outil 1**	**Outil 2**	**Outil 3**	**Outil4**	**Outil5**	**Outil6**
Outil 1	0	59	12	80	70	61
Outil 2	10	0	19	20	15	24
Outil 3	12	18	0	8	15	12
Outil 4	47	15	12	0	20	49
Outil 5	37	10	21	18	0	11
Outil 6	23	18	42	11	29	0

Tableau 4.5: Temps de changement d'outils.

- **Le temps de reconfiguration**

Le temps de changement de configuration des différentes machines de notre exemple est exposé dans le tableau suivant:

	Configurations	**C1**	**C2**	**C3**	**C4**
Machine 1	**C1**	00	32	54	
	C2	25	00	18	
	C3	29	45	00	
Machine 2	**C1**	00	49	75	47
	C2	49	00	34	16
	C3	68	43	00	70
	C4	64	13	11	00
Machine 3	**C1**	00	34		
	C2	09	00		
Machine 4	**C1**	00	48		
	C2	22	00		
Machine 5	**C1**	00	51	12	
	C2	29	00	62	
	C3	36	84	00	
Machine 6	**C1**	00	87	92	

	C2	25	00	33	
	C3	39	14	00	
Machine 7	**C1**	00	55		
	C2	17	00		
Machine 8	**C1**	00	13	17	61
	C2	29	00	31	54
	C3	38	42	00	19
	C4	11	60	23	00

Tableau 4.6: Temps de reconfiguration.

- **Le temps de changement des machines**

Le temps de changement des machines de notre exemple est détaillé dans le tableau12:

Machines	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**
M 1	00	80	62	58	91	52	85	79
M 2	68	00	68	85	75	68	52	58
M 3	62	56	00	66	64	95	91	73
M 4	52	98	73	00	79	45	82	41
M 5	85	57	60	97	00	53	66	77
M 6	79	52	81	59	93	00	73	63
M 7	44	83	58	76	64	53	00	95
M 8	98	65	42	57	53	49	63	00

Tableau 4.7: Temps de changement des machines.

Comme expliqué dans le chapitre III, les informations concernant les coûts se composent de cinq sommes (Coût d'utilisation des machines, coût de changement de machines, coût de reconfiguration d'une machine, coût d'utilisation et de changement des outils).

Nous présentons, dans ce qui suit, la quantité de chaque coût.

- **Coût d'utilisation des machines**

Le coût d'utilisation des machines de notre exemple est détaillé dans le tableau suivant:

Machines	**Coût d'utilisation**
Machine 1	50
Machine 2	20
Machine 3	25
Machine 4	22
Machine 5	30
Machine 6	26
Machine 7	39
Machine 8	36

Tableau 4.8: Coût d'utilisation des machines.

- **Coût d'utilisation d'outils**

Le coût d'utilisation des outils de notre exemple est détaillé dans le tableau suivant :

Outils	Coût d'utilisation
Outil 1	08
Outil 2	14
Outil 3	25
Outil 4	10
Outil 5	15
Outil 6	19

Tableau 4.9 : Coût d'utilisation d'outils.

- **Coût de changement des machines**

Le coût de changement des machines de notre exemple est résumé dans le tableau suivant :

Machines	M1	M2	M3	M4	M5	M6	M7	M8
M1	00	06	12	15	17	19	14	19
M2	07	00	10	04	06	13	14	11
M3	11	19	00	15	07	08	09	15
M4	12	14	10	00	11	16	19	15
M5	12	14	09	21	00	19	18	15
M6	08	05	14	07	11	00	10	16
M7	16	11	08	13	09	10	00	11
M8	06	17	12	15	19	20	14	00

Tableau 4.10 : Coût de changement des machines.

- **Le coût de changement d'outils**

Le coût de changement d'outils de notre exemple est expliqué dans le tableau suivant:

Outils	Outil 1	Outil 2	Outil 3	Outil4	Outil 5	Outil6
Outil 1	00	16	07	05	8.5	08
Outil 2	8.5	00	10	05	09	7.5
Outil 3	2	5.5	00	4.5	16	12
Outil 4	4.5	9.5	07	00	11	10
Outil 5	14	08	05	13	00	06
Outil 6	10	06	17	09	07	15

Tableau 4.11: Coût de changement des outils.

- **Coût de reconfiguration des machines**

Le coût de reconfigurations des machines de notre exemple est résumé dans le tableau suivant:

	Configurations	**C1**	**C2**	**C3**	**C4**
Machine 1	**C1**	00	13	03	
	C2	15	00	11	
	C3	09	19	00	
Machine 2	**C1**	00	09	17	17
	C2	19	00	24	06
	C3	18	13	00	26
	C4	10	13	11	00
Machine 3	**C1**	00	14		
	C2	25	00		
Machine 4	**C1**	00	18		
	C2	22	00		
Machine 5	**C1**	00	15	12	
	C2	22	00	16	
	C3	16	14	00	
Machine 6	**C1**	00	17	12	
	C2	05	00	23	
	C3	19	14	00	
Machine 7	**C1**	00	15		
	C2	17	00		
Machine 8	**C1**	00	13	17	24
	C2	09	00	13	30
	C3	28	12	00	19
	C4	11	08	23	00

Tableau 4.12 : Coût de reconfiguration des machines.

- **La fiabilité de machines**

La fiabilité de machines de notre exemple est expliquée dans le tableau suivant:

Machine	M1	M2	M3	M4	M5	M6	M7	M8
Fiabilité	0.90	0.88	0.94	0.79	0.91	0.84	0.94	0.87

Tableau 4.13: Fiabilité des machines.

4.2 Exécution de l'algorithme NSGA-III

À ce stade, toutes les données des machines, informations sur le produit, le temps, le coût et la faibilité sont identifiés. Pour exécuter l'algorithme (NSGA-III), il est indispensable de déterminer les paramètres de NSGA-III. Les paramètres pris en compte pour notre cas sont les suivants :

- Taille de la population = 30
- Taux de croisement = 0.8
- Taux de mutation = 0.2
- Nombre de générations = 25

Après l'exécution de l'algorithme, on obtient la figure qui représente le front Pareto optimal réalisé par l'algorithme NSGA-III.

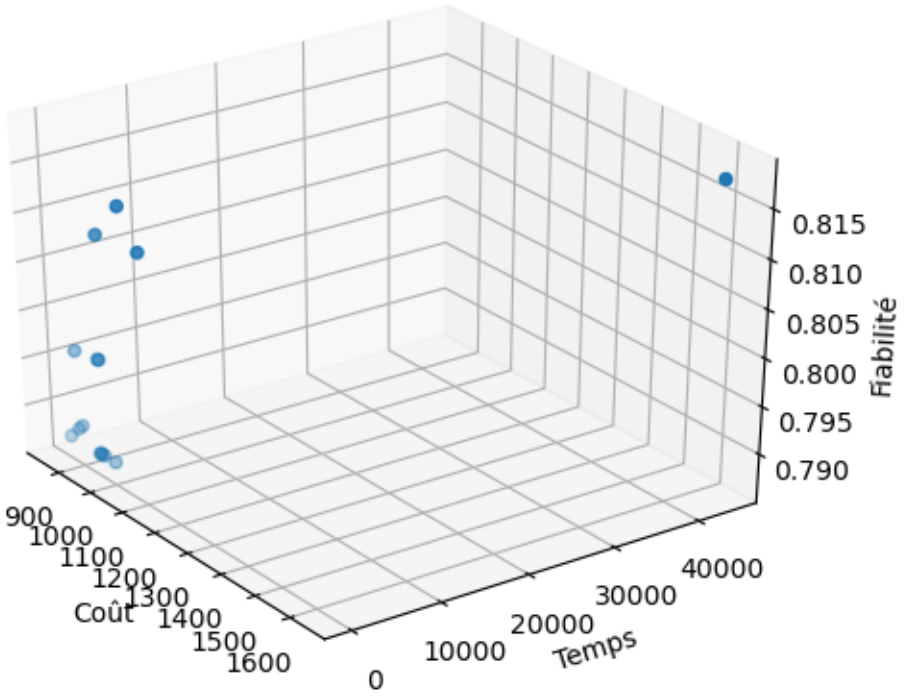


Figure 4.27: courbe de l'algorithme NSGA-III.

Le coût total, le temps total et la fiabilité totale sont résumés dans le tableau suivant :

Coût total	Temps total	Fiabilité
225.879	891	0.789
437.638	1641	0.818
256.338	878	0.788
171.567	948	0.787
177.107	947	0.787
183.767	960	0.787
186.417	952	0.792
207	994	0.782
240.681	1087	0.811
271.656	875	0.796
233.36	959	0.81

Tableau 4.14: Les valeurs du coût total, temps total et fiabilité pour chaque solution.

Et enfin, pour exécuter la séquence d'opérations qui a été représenté dans le tableau 4.2, les meilleures solutions représentants les machines, les configurations et les outils sont résumés comme suivant :

Solution1
Machine : [3, 2, 5, 5, 5, 5, 4, 8, 5, 7, 1, 8, 7, 6, 2, 2]
configuration : [2, 4, 1, 1, 1, 3, 1, 4, 3, 1, 4, 2, 2, 1, 2, 2]
Outil : [6, 6, 6, 4, 6, 6, 1, 5, 1, 1, 4, 2, 1, 6, 5, 6]

Solution 2
Machine : [1, 5, 3, 8, 1, 1, 3, 8, 3, 5, 2, 2, 2, 2, 8, 2]
configuration : [1, 1, 2, 4, 1, 2, 1, 3, 1, 3, 4, 2, 1, 2, 1, 2]
Outil : [2, 6, 4, 5, 3, 1, 6, 5, 6, 5, 5, 6, 6, 5, 4, 5]

Solution 3
Machine : [3, 2, 8, 8, 5, 5, 2, 8, 5, 2, 1, 8, 7, 6, 2, 2]
configuration : [2, 4, 3, 3, 1, 2, 4, 4, 3, 4, 3, 2, 2, 1, 2, 2]
Outil : [4, 6, 5, 4, 6, 6, 5, 5, 6, 5, 4, 2, 1, 5, 5, 6]

Solution 4
Machine : [6, 2, 3, 5, 5, 5, 7, 8, 2, 2, 6, 8, 2, 2, 5, 5]
configuration : [3, 4, 1, 1, 1, 2, 1, 4, 2, 4, 4, 2, 1, 1, 3, 3]
Outil : [6, 6, 6, 6, 6, 5, 1, 5, 6, 5, 5, 2, 6, 6, 5, 5]

Solution 5
Machine : [6, 2, 3, 5, 5, 5, 7, 8, 2, 2, 3, 8, 2, 2, 5, 5]
configuration : [3, 4, 1, 1, 1, 3, 1, 4, 2, 4, 3, 2, 1, 1, 3, 2]
Outil : [6, 6, 6, 4, 6, 5, 1, 5, 6, 5, 5, 2, 6, 6, 5, 5]
Solution 6
Machine : [6, 2, 3, 5, 5, 5, 7, 8, 2, 2, 8, 8, 2, 2, 5, 5]
configuration : [3, 4, 1, 1, 1, 2, 1, 4, 2, 4, 3, 2, 1, 1, 3, 3]
Outil : [4, 3, 6, 6, 6, 5, 1, 5, 6, 5, 5, 2, 6, 6, 5, 5]
Solution 7
Machine : [6, 2, 3, 5, 5, 5, 7, 8, 2, 2, 1, 8, 2, 2, 5, 5]
configuration : [3, 4, 1, 1, 1, 2, 1, 4, 2, 4, 3, 2, 1, 1, 3, 3]
Outil : [6, 6, 6, 4, 4, 6, 1, 2, 6, 5, 4, 2, 6, 6, 4, 6]
Solution 8
Machine : [6, 2, 3, 5, 5, 5, 7, 8, 2, 2, 6, 8, 2, 2, 5, 5]
configuration : [3, 4, 1, 1, 1, 2, 1, 4, 2, 4, 3, 2, 1, 1, 3, 3]
Outil : [4, 3, 6, 6, 6, 5, 1, 5, 6, 5, 5, 2, 6, 6, 4, 6]
Solution 9
Machine : [3, 2, 3, 5, 3, 5, 2, 8, 2, 2, 5, 8, 2, 6, 2, 2]
configuration : [2, 4, 1, 1, 1, 3, 4, 4, 2, 4, 2, 2, 1, 1, 2, 2]
Outil : [6, 6, 6, 4, 6, 6, 5, 2, 6, 5, 5, 2, 6, 5, 5, 5]
Solution 10
Machine : [3, 2, 3, 8, 5, 5, 2, 8, 5, 2, 4, 8, 7, 6, 2, 2]
configuration : [2, 4, 1, 3, 1, 3, 4, 4, 3, 4, 3, 2, 2, 1, 2, 2]
Outil : [4, 3, 6, 2, 6, 6, 5, 5, 6, 5, 5, 2, 6, 6, 5, 6]
Solution 11
Machine : [3, 2, 3, 5, 3, 5, 2, 8, 2, 2, 1, 8, 7, 6, 2, 2]
configuration : [2, 4, 1, 1, 1, 3, 4, 4, 2, 4, 4, 2, 2, 1, 2, 2]
Outil : [4, 6, 6, 6, 6, 5, 5, 2, 6, 5, 4, 2, 6, 5, 5, 5]

5 Conclusion

Pour conclure notre chapitre, l'implémentation a été efficacement réalisée grâce au langage de programmation soigneusement choisi " Python" ainsi que l'environnement de programmation "Pycharme". L'utilisation de ces deux derniers nous a permis d'obtenir les résultats attendus. Ce chapitre a été consacré à la présentation des données et des outils utilisés pour la réalisation de notre application, puis à l'exposition des résultats de l'implémentation sous forme d'interfaces utilisateur graphiques (GUI). En dernière section, on a traité et testé les résultats grâce à l'exemple appliqué sous les paramètres prédéfinis.

Conclusion générale

Durant ces dernières décennies, le Système Manufacturier Reconfigurable « RMS » a connu un intérêt majeur. Il est considéré comme un paradigme manufacturier changeable et le plus adéquat pour faire face aux exigences du marché et du client. Ceci est possible grâce à la conception des RMSs qui intègre dès le départ l'évolutivité, la reconfigurabilité ainsi que les changements aux niveaux des fonctionnalités et/ou de ses capacités.

Au cours de l'élaboration de ce mémoire, nous avons essayé de trouver une solution adéquate aux problèmes de génération d'une ligne de production dans le Système Manufacturier Reconfigurable. De ce fait nous avons résolu le problème d'optimisations multi-objectif dans le but de choisir les meilleures machines reconfigurables pour produire un produit final dans un temps et un cout minimisés et une fiabilité maximisée basant sur l'adaptation d'un algorithme évolutionnaire nommé « NSGA-III».

Durant la réalisation de notre projet nous avons appris des connaissances sur :

- Les problèmes d'optimisation multi objectifs.
- L'algorithme d'optimisation multi objectifs « NSGA-III».
- Les Systèmes Manufacturiers Reconfigurables « RMSs ».

Nous avons implémenté l'algorithme d'optimisations multi-objectif « NSGA-III» pour obtenir le front de Pareto optimal, en se basant sur les travaux de [2], [3].

La conception de telle approche ainsi que l'expérimentation préliminaire du modèle proposé, ont donné lieu à des propositions et à des réflexions définies comme étant des perspectives de recherche, qui sont les suivants :

Le model proposé dans ce projet nous a donné l'occasion à réfléchir sur des perspectives d'amélioration du model. Parmi ces perspectives nous citons quelques unes possible à réaliser :

Faire une comparaison entre NSGA-III et une autre méta-heuristique comme NSGA-II SPEA-II, MOPSO (Multi Objective particle Swarm Optimization), ABC (Artificial Bee Colony),... ect

Ajouter d'autres indicateurs (fonctions objectifs) comme la maintenance, la disponibilité, la reconfigurabilité,…. ect

[1] Y. Koren *et al.*, « Reconfigurable manufacturing systems », *Ann. CIRP*, vol. 48, p. 2, 1999.

[2] « BENSMAÏNE Abderrahmane , thèse de doctorat « Algorithmes évolutionnaires et méthodes approchées multicritères pour la génération des processus de fabrication dans un environnement reconfigurable », 2013 - Recherche Google ».

[3] M. Ashraf, « Some Aspects of Planning for Reconfigurable Manufacturing Systems », PhD Thesis, Aligarh Muslim University, 2018.

[4] M. Yagoubi, « Optimisation évolutionnaire multi-objectif parallèle: application à la combustion Diesel », PhD Thesis, 2012.

[5] Y. Collette et P. Siarry, *Optimisation multiobjectif: Algorithmes*. Editions Eyrolles, 2011.

[6] C. Dhaenens, « Optimisation Combinatoire Multi-Objectif: Apport des méthodes coopératives et contribution à l'extraction de connaissances », PhD Thesis, 2005.

[7] A. Cheriet, « Métaheuristique hybride pour l'optimisation multi-objectif », PhD Thesis, Université Mohamed Khider-Biskra, 2016.

[8] C. A. Coello, « An updated survey of GA-based multiobjective optimization techniques », *ACM Comput. Surv. CSUR*, vol. 32, n° 2, p. 109–143, 2000.

[9] P. Siarry et Y. Collette, « Optimisation multiobjectif », *Groupe Eyrolles*, 2002.

[10] Y. BAHMANI, « Optimisation multicritère de l'ordonnancement des activités de la production et de la maintenance intégrées dans un atelier Job Shop », PhD Thesis, Université de Batna 2, 2017.

[11] F. Mazière, « Modèles de parallélisme pour les métaheuristiques multi-objectifs », PhD Thesis, Université du Québec à Chicoutimi, 2019.

[12] N. Jozefowiez, « Optimisation combinatoire multi-objectif: des méthodes aux problèmes, de la Terre à (presque) la Lune », PhD Thesis, Institut National Polytechnique de Toulouse (INP Toulouse), 2013.

[13] K. Deb, S. Agrawal, A. Pratap, et T. Meyarivan, « A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II », in *International conference on parallel problem solving from nature*, 2000, p. 849–858.

[14] Y. Houam, « Commande multi-objectifs en utilisant les inégalités matricielles linéaires (LMIs) et les algorithmes génétiques », PhD Thesis, Université Mohamed Khider-Biskra, 2013.

[15] M. A. Benatia, « Optimisation multi-objectives d'une infrastructure réseau dédiée aux bâtiments intelligents », PhD Thesis, Rouen, INSA, 2016.

[16] H. John, « Holland. 1975. Adaptation in natural and artificial systems », *Ann Arbor Univ. Mich. Press*, 1975.

[17] D. E. Goldberg, « Genetic algorithms in search », *Optim. Mach.*, 1989.

[18] A. E. Eiben et J. E. Smith, Introduction to evolutionary computing, vol. 53. Springer, 2003.

[19] J.-K. Hao et C. Solnon, Méta-heuristiques et intelligence artificielle. 2014.

[20] A. Berro, « Optimisation multiobjectifs et stratégies d'évolution en environnement dynamique », PhD Thesis, ANRT [diff.], 2001.

[21] J. rey Horn, N. Nafpliotis, et D. E. Goldberg, « Multiobjective optimization using the niched pareto genetic algorithm », *IlliGAL Rep.*, vol. 93005, 1993.

[22] C. M. Fonseca et P. J. Fleming, « An overview of evolutionary algorithms in multiobjective optimization », *Evol. Comput.*, vol. 3, nº 1, p. 1–16, 1995.

[23] N. Srinivas et K. Deb, « Muiltiobjective optimization using nondominated sorting in genetic algorithms », *Evol. Comput.*, vol. 2, nº 3, p. 221–248, 1994.

[24] S. Mahdi, « Optimisation multiobjectif par un nouveau schéma de coopération méta/exacte », 2007.

[25] E. Zitzler, M. Laumanns, et L. Thiele, « SPEA2: Improving the strength Pareto evolutionary algorithm », *TIK-Rep.*, vol. 103, 2001.

[26] K. Deb, A. Pratap, S. Agarwal, et T. Meyarivan, « A fast and elitist multiobjective genetic algorithm: NSGA-II », *IEEE Trans. Evol. Comput.*, vol. 6, nº 2, p. 182–197, 2002.

[27] G. Campos Ciro, « Développement de méthodes d'ordonnancement efficaces et appliquées dans un système de production mécanique », PhD Thesis, Troyes, 2015.

[28] I. Chalfoun, « Conception et déploiement des Systèmes de Production Reconfigurables et Agiles (SPRA) », PhD Thesis, 2014.

[29] Y. Koren, X. Gu, et W. Guo, « Reconfigurable manufacturing systems: Principles, design, and future trends », *Front. Mech. Eng.*, vol. 13, nº 2, p. 121–136, 2018.

[30] G. Zhang, R. Liu, L. Gong, et Q. Huang, « An analytical comparison on cost and performance among DMS, AMS, FMS and RMS », in *Reconfigurable manufacturing systems and transformable factories*, Springer, 2006, p. 659–673.

[31] K. Lameche, « Proposition d'une méthodologie pour la conception des systèmes de production reconfigurables et d'un outil associé d'aide à la décision par simulation de flux », PhD Thesis, Nantes, 2018.

[32] H. Dammak, « Reconfiguration dynamique des systèmes manufacturiers non-fiables », PhD Thesis, Université Laval, 2014.

[33] H. H. Benderbal, « Développement d'une nouvelle famille d'indicateurs de performance pour la conception d'un système manufacturier reconfigurable (RMS): approches évolutionnaires multicritères », PhD Thesis, 2018.

[34] G. Putnik *et al.*, « Scalability in manufacturing systems design and operation: State-of-the-art and future developments roadmap », *CIRP Ann.*, vol. 62, nº 2, p. 751–774, 2013.

[35] Y. Koren, « General RMS characteristics. Comparison with dedicated and flexible systems », in *Reconfigurable manufacturing systems and transformable factories*, Springer, 2006, p. 27–45.

[36] F. Frizon de Lamotte, « Proposition d'une approche haut niveau pour la conception, l'analyse et l'implantation des systèmes reconfigurables », BRETAGNE SUD, 2006.

[37] Chaube, A., Benyoucef, L. & Tiwari, M. K. , 'An adapted NSGA-2 algorithm based dynamic process plan generation for a reconfigurable manufacturing system', Journal of Intelligent Manufacturing, 2012 - Recherche Google ».

[38] Développement de méthodes d'ordonnancement efficaces et appliquées dans un système de production mécanique - Recherche Google ».

[39] Hasan, F.Hasan, S. Jain, & S., Naved Ali, 'Configuration selection for optimal Throughput from a reconfigurable product line using genetic algorithm', conference, 2014

[40] F., Musharavati, A.S., Hamouda 'Enhanced simulated annealing based algorithms and their applications to process planning in reconfigurable manufacturing systems. advanceds in Engineering software, 45(1), PP.80-90.

[41] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints." IEEE Trans. Evolutionary Computation, vol. 18, no. 4, pp. 577–601, 2014.

[42]X., Yuana, H., Tiana, Y., Yuanb et al, 'An extended NSGA-III for solution multi-objective hydro-thermal-wind scheduling considering wind power cost', Energy Conversion and Management, Volume 96, 15 May 2015, Pages 568-578

[43]M.,Tavanaab,Z.LicMohammad et al, 'Multi-Objective Control Chart Design Optimization Using NSGA-III and MOPSO Enhanced with DEA and TOPSIS', Expert Systems with Applications Volume 50, 15 May 2016, Pages 17-39

[44] G., Campos Ciro "dévelopment de methodes d'ordonnacement efficaces et appliquées dans un systeme de production mécanique', these de docorat, 2015.

[45] I. Das and J. E. Dennis, "Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems," SIAM Journal on Optimization, vol. 8, no. 3, pp. 631–657, 1998.