N° Ordre...../FT/UMBB/2021

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE M'HAMED BOUGARA-BOUMERDES

Faculté de Technologie

Thèse de Doctorat

Présentée par :

CHEBOUBA Billal Nazim

En vue de l'obtention du diplôme de **DOCTORAT** LMD en **:**

Filière : Génie Mécanique

Option : Mécatronique

Optimisation Multi-objectif de la Fiabilité des Systèmes Complexes par L'intelligence Artificielle.

Devant le jury composé de :

Mr BENAZZOUZ	Djamel	Professeur	UMBB	Président
Mr ADJERID	Smail	Professeur	UMBB	Directeur
Mr MELLAL	Mohamed Arezki	MCA	UMBB	Co- Directeur
Mr KARA	Redouane	Professeur	UMMTO	Examinateur
Mr BENTARZI	Hamid	Professeur	UMBB	Examinateur

Année Universitaire 2021/2022

Remerciements

Avant tout, tous les remerciements et louanges sont dus à ALLAH (Dieu) le Tout-Puissant pour sa bénédiction qui a rendu ce travail possible.

Je tiens à exprimer mes vifs remerciements au Professeur Smail Adjerid, mon directeur de thèse pour son encadrement continu, pour ses remarques constructives qu'il m'a fournies ainsi que pour ses précieux conseils durant toute la période de ma thèse. Je le remercie également pour la confiance et la grande liberté d'idées et de travail qu'il m'a accordées. En dehors de ses apports scientifiques, je n'oublierai pas aussi de le remercier pour ses qualités humaines, son hospitalité et son soutien permanent qui m'ont permis de mener à bien ma thèse doctorat. À tous leurs doctorants actuels et futurs, vous êtes très chanceux.

Je tiens à exprimer mes remerciements à mon co-directeur de thèse Dr. MELLAL Arezki Mohamed pour m'avoir donné l'opportunité de travailler sur ce sujet intéressant, sans son soutien, disponibilité, et discussions fructueuses, je n'aurais pas atteint les résultats que j'ai obtenus dans ma thèse de doctorat.

Je tiens à exprimer mes sincères remerciements au directeur du Laboratoire Professeur Djamel Benazzouz pour ses conseils et ses précieuses suggestions qui ont été indispensables à ce travail et je le remercie aussi tout particulièrement pour son soutien continu durant toutes les phases de la présente thèse.

Ma profonde gratitude va au Professeur Belkacem Ould-Bouamama et au Dr Othman Lakhal pour la confiance qui m'a été accordée en me donnant l'opportunité de travailler dans son équipe (PERSI) au Centre de Recherche en Informatique, Signal et Automatique de Lille (CRISTAL).

Je remercie messieurs les membres du jury pour la caution qu'ils ont bien voulu apporter à ce travail. J'adresse mes remerciements aux : Monsieur Djamel BENAZZOUZ, professeur à l'université de M'hamed Bougara de Boumerdes pour m'avoir fait l'honneur de bien vouloir participer au jury en tant que président, mais également pour tout l'intérêt qu'il a manifesté sur ce travail de recherche, Monsieur Smail Adjerid, mon directeur de thèse. Monsieur Redouane KARA, professeur à l'Université de Tizi-Ouzou et Monsieur Hamid BENTARZI professeur à l'université de M'hamed Bougara de Boumerdes pour m'avoir fait l'honneur de bien vouloir accepter de

participer au jury en tant qu’examinateurs de mon travail, je les remercie vivement pour l’intérêt qu’ils ont manifesté à mes travaux, pour leur grande qualités humaines.

Je tiens à remercier tout particulièrement, M. Mahdi Ouziala, M. Adel Termeche, Dr Mahdi Boukerdja votre disponibilité et votre aide m'ont été d'une grande aide, et je tiens aussi à remercier tous les membres du laboratoire LMSS et particulièrement à M. Yacine Lounici, qui m'a accompagné dans mon cheminement doctoral.

Dédicace

A ceux qui ont consacré leur vie à me soutenir,

Mon défunt Père et ma Mère,

A ma sœur et mon frère,

A toute ma famille.

Résume

Dans le contexte actuel caractérisé par une forte concurrence économique, les systèmes industriels doivent être les plus fiables possibles pour rester compétitifs, ainsi l'optimisation de la fiabilité des systèmes est devenue un sujet fondamental dans la conception et l'exploitation des systèmes de fabrication à grande échelle.

Afin d'atteindre un haut niveau de fiabilité d'un système de production, trois stratégies fondamentales peuvent être appliquées par les concepteurs : l'allocation de fiabilité, l'allocation de redondance et l'allocation de fiabilité-redondance.

Un compromis entre les variables de décision est nécessaire pour optimiser un ou plusieurs objectifs sans violer l'ensemble des contraintes de conception considéré, ce type de problème est difficile à résoudre en raison de la quantité considérable d'efforts de calcul requis pour trouver les solutions optimales.

Il a été prouvé au cours des deux dernières décennies que les algorithmes inspirés de la nature sont attrayants pour ce genre de problème d'optimisation mathématiques. Les algorithmes bio-inspirés sont efficaces pour arriver à la solution optimale à un problème lorsqu'il existe une myriade de possibilités. Ils sont non déterministes et sont utilisés dans l'analyse de systèmes, leur simplicité et leur parallélisme inhérent sont deux raisons principales de leur popularité et de leur large éventail d'applications. Ils sont flexibles et peuvent être adaptés aux changements de l'environnement, par conséquent les techniques d'optimisation métaheuristiques ont été utilisées comme alternative aux approches mathématiques classiques pour obtenir des solutions optimales globales ou quasi globales en raison de leur grande capacité à détecter des régions prometteuses dans l'espace de recherche et à les explorer à un moment précis.

L'objectif de la thèse est de proposer une nouvelle approche multi-objectif pour l'optimisation simultanée des éléments FMDS+C (Fiabilité, Maintenabilité, Disponibilité, Sécurité et Cout). L'approche développée sera basée sur les méthodes évolutionnaires bio-inspirées et la fusification des incertitudes. Des applications numériques sur des systèmes complexes feront l'objet de la validation de cette approche.

Mots clés : sureté de fonctionnement ; optimisation multi-objectif ; intelligence artificielle.

Abstract

In the current context characterized by strong economic competition, industrial systems must be as reliable as possible to remain competitive, thus optimizing the reliability of systems has become a fundamental subject in the design and operation of manufacturing systems at large scale.

In order to achieve a high level of reliability of a production system, three fundamental strategies can be applied by designers: reliability allocation, redundancy allocation and reliability-redundancy allocation.

A trade-off between decision variables is needed to optimize one or more objectives without violating the set of design constraints under consideration, this type of problem is difficult to solve due to the considerable amount of computational effort required to find the solutions optimal.

It has been proven over the past two decades that nature-inspired algorithms are attractive for this kind of mathematical optimization problem. Bio-inspired algorithms are effective in arriving at the optimal solution to a problem when there are a myriad of possibilities. They are non-deterministic and are used in system analysis, their simplicity and inherent parallelism are two main reasons for their popularity and wide range of applications. They are flexible and can be adapted to changes in the environment, therefore metaheuristic optimization techniques have been used as an alternative to classical mathematical approaches to obtain global or near-global optimal solutions due to their high ability to detect regions promising in the search space and explore them at a specific time.

The objective of the thesis is to propose a new multi-objective approach for the simultaneous optimization of RMAS+C elements (Reliability, Maintainability,

Availability, Security and Cost). The developed approach will be based on bio-inspired evolutionary methods and the fusification of uncertainties. Numerical applications on complex systems will be the subject of the validation of this approach.

Keywords: operational safety; multi-objective optimization; artificial intelligence

ملخص

في السياق الحالي الذي يتميز بمنافسة اقتصادية قوية، يجب أن تكون الأنظمة الصناعية موثوقة قدر الإمكان لتظل قادرة على المنافسة، وبالتالي أصبح تحسين موثوقية الأنظمة موضوعًا أساسيًا في تصميم وتشغيل أنظمة التصنيع على نطاق واسع.

من أجل تحقيق مستوى عالٍ من الموثوقية لنظام الإنتاج، يمكن للمصممين تطبيق ثلاث استراتيجيات أساسية: تخصيص الموثوقية وتخصيص التكرار وتخصيص الموثوقية والتكرار. هناك حاجة إلى المفاضلة بين متغيرات القرار لتحسين هدف واحد أو أكثر دون انتهاك مجموعة قيود التصميم قيد الدراسة، ويصعب حل هذا النوع من المشكلات بسبب المقدار الكبير من الجهد الحسابي المطلوب للعثور على الحلول المثلى.

قد ثبت خلال العقدين الماضيين أن الخوارزميات المستوحاة من الطبيعة جذابة لهذا النوع من مشاكل التحسين الرياضي. تعد الخوارزميات المستوحاة من الطبيعة فعالة في الوصول إلى الحل الأمثل لمشكلة ما عندما يكون هناك عدد لا يحصى من الاحتمالات. إنها غير حتمية وتستخدم في تحليل النظام، وبساطتها وتوازيها المتأصل هما سببان رئيسيان لشعبيتها ومجموعة واسعة من التطبيقات. إنها مرنة ويمكن تكييفها مع التغيرات في البيئة، لذلك تم استخدام "الأدلة العليا" كبديل للنهج الرياضية الكلاسيكية للحصول على حلول مثالية أو شبه مثالية نظرًا لقدرتها العالية على اكتشاف المناطق الواعدة في مساحة البحث واستكشافها في وقت محدد.

الهدف من الأطروحة هو اقتراح نهج جديد متعدد الأهداف للتحسين المتزامن للعناصر (الموثوقية، والاستدامة، والتوافر، والأمن، والتكلفة). سوف يعتمد النهج المطور على الأساليب التطورية المستوحاة من الطبيعة والمنطق الضبابي. ستكون التطبيقات العددية على الأنظمة المعقدة موضوع التحقق من صحة هذا النهج.

الكلمات المفتاحية: سلامة التشغيل؛ متعدد الأهداف، ذكاء اصطناعي

Table des matières

Introduction générale

Chapitre I : Sûreté de fonctionnement

Chapitre II : Approches d'optimisation et de résolution

Chapitre III : **Optimisation mono-objectif : Trois méthodes métaheuristiques pour optimiser la fiabilité des systèmes (PSO, SFS, CS)**

Chapitre IV : Optimisation Multi-Objectif : Allocation de la fiabilité et de la redondance d'un système de protection dans une centrale électrique (NSGA-II, NSGA-III).

Chapitre V : Optimisation floue de la fiabilité des systèmes multi-objectif par des algorithmes génétiques et analyse de clustering

Liste des figures

Liste des tableaux

liste des abréviations

λ	Le taux de défaillance
r_i	La fiabilité du sous-système i
r_{ij}	La fiabilité *j* au niveau *i*
n_i	Le nombre de composants redondants dans le sous-système *i*
n_{ij}	Le nombre de composants redondants dans le sous-système *i* au niveau *j*
m	Nombre de sous-systèmes dans le système
v_{ij}	Le volume de chaque alternative de conception *j* au niveau *i*
w_{ij}	Le poids de chaque alternative de conception *j* au niveau *i*
C_{ij}	Le cout de chaque alternative de conception *j* au niveau *i*
b	Le vecteur de la limitation des ressources
$g_j(\mathrm{x})$	Contraintes d'inégalité
$h_k(x)$	Contraintes d'égalité
V_0	La borne supérieure du volume
w_0	La borne supérieure du poids
C_0	La borne supérieure du coût
$R_s = (.)$	La fonction objectif du problème : fiabilité globale du système
$C_s = (.)$	La fonction objectif du problème : le coût global du système
$V_s = (.)$	La fonction objectif du problème : le volume global du système
d	Dimensions de l'espace de recherche
N	La taille de la population
w_v	Le coefficient d'inertie du PSO
c_1	Le coefficient cognitif du PSO
c_2	Le coefficient social du PSO
R_1 et R_2	Des nombres aléatoires qui introduisent une composante stochastique.
BP	La position du meilleur point dans le groupe
P_i	La position du du *ième* point dans le groupe
$P_i^{'}$	La nouvelle position modifiée de P_i
$\mathrm{P_r}$ et P_t	Des points choisis au hasard dans le groupe
$P_i^{''}$	La nouvelle position modifiée de $P_i^{'}$
ε	Nombre aléatoire choisi dans la distribution uniforme dans l'espace continu
ph	La probabilité que l'oiseau hôte découvre l'œuf de coucou pondu dans son nid
x^*	La solution optimale de Pareto
ϕ_j	Facteur de pénalité ou paramètre de pénalité
F_i	Le front i contient les solutions qui dominent les solutions des fronts précédents
Q_0	Population de progéniture de la même taille que la population parente
P_j	Population créée grâce à l'utilisation d'opérateurs de recombinaison a l'itération *j*
Q_j	Population de progéniture de la même taille que la population parente a l'itération *j*
$\tilde{X}$	La valeur floue des paramètres X,
$\tilde{R}_s(\cdot)$	La fiabilité du système

$\tilde{C}_s(\cdot)$	Le coût du système
$\tilde{g}_j(\cdot)$	L'ensemble des contraintes de conception
$\tilde{r}_i$	La fiabilité floue du sous-système *i*
$\tilde{n}_i$	Le nombre floue de composants redondants dans le sous-système *i*
S	métrique de mesure de performance ‘’ *spacing*’’

Introduction générale

Introduction générale

1. Motivations et objectif

Dans le contexte actuel caractérisé par une forte concurrence économique, les systèmes industriels doivent être les plus fiables possibles pour rester compétitifs, ainsi l'optimisation de la fiabilité des systèmes est devenue un sujet fondamental dans la conception et l'exploitation des systèmes de fabrication à grande échelle. Trois principales méthodes fondamentales peuvent être appliquées par les concepteurs pour but d'augmenter la fiabilité des systèmes : l'allocation de fiabilité, l'allocation de redondance et l'allocation de fiabilité et de la redondance.

Ces méthodes impliquent l'optimisation d'un ou plusieurs objectifs (tels que la fiabilité globale et le coût global du système) sous l'ensemble des contraintes de conception tels que le volume, le poids et le coût. Le problème d'allocation de fiabilité-redondance (RRAP) est un problème NP-Hard, dans lequel les allocations de redondance et de fiabilité sont les variables de décision. Par conséquent, un compromis entre les variables de décision est nécessaire pour optimiser un objectif ou plusieurs objectifs sans violer l'ensemble de contraintes de conception considéré. De nombreuses métaheuristiques ont été utilisées comme approches de solutions pour résoudre le RRAP. La plupart des travaux consacrés à la RRAP ont porté sur des problèmes mono-objectifs.

Les algorithmes évolutionnaires multi-objectifs ont montré leur capacité à résoudre des problèmes d'optimisation multi-objective. Leur particularité est de générer un ensemble de solutions (solutions non dominées) les plus proches possibles des solutions optimales, appelé front de Pareto. Une autre de leur particularité est que ces algorithmes sont capables de maintenir la diversité dans la solution à travers les itérations. Plusieurs algorithmes évolutionnaires multi-objectifs ont été développés au cours des dernières décennies, tels que MOGA, NPGA, NSGA, SPEA, PAES, MOMGA, SPEA2, PEAS-II, NSGA-II, MOEA/D et NSGA-III.

L'objectif de la présente thèse est de proposer une approche de solution permettant de traiter efficacement l'optimisation des objectifs de fiabilité et de coût du système sous des paramètres flous. A cet effet, deux algorithmes sont utilisés : Non dominated Sorting Genetic Algorithm II (NSGA-II) et Non dominated Sorting Genetic Algorithm III (NSGA-III). De même, une méthode de défuzification est mise en œuvre pour convertir les paramètres flous en paramètres précis. Des mesures de performance sont exploitées pour trouver la meilleure méthode d'optimisation multi-objective adaptée à ce type de problème. Enfin, nous sélectionnons les meilleures solutions de compromis du front de Pareto en recourant à une analyse de clustering pour réduire le nombre de solutions (aide à la décision).

2. Contributions

Les travaux de recherche réalisés dans le cadre de cette thèse ont permis la dissémination de nos résultats à travers la publication et les communications suivantes :

Revue internationale :

- Chebouba Billal Nazim, Mohamed Arezki Mellal, and Smail Adjerid. "Fuzzy multiobjective system reliability optimization by genetic algorithms and clustering analysis." *Quality and Reliability Engineering International* (2020).

Conférences internationals :

- CHEBOUBA Billal Nazim, et al. "Multi-objective System Reliability-redundancy Allocation in a Power Plant by Considering Three Targets." *2020 7th International Conference on Control, Decision and Information Technologies (CoDIT)*. Vol. 1. IEEE, 2020.
- CHEBOUBA Billal Nazim, et al. "System reliability and cost optimization under various scenarios using NSGA-III." *2020 International Conference on Electrical Engineering (ICEE)*. IEEE, 2020.
- Mellal Mohamed Arezki, and Billal Nazim Chebouba. "Cost and Availability optimization of Overspeed Protection System in a Power plant." *2019 International Conference on Advanced Electrical Engineering (ICAEE)*. IEEE, 2019.
- CHEBOUBA Billal Nazim, Mohamed Arezki MELLAL, and Smail ADJERID. "Multi-objective system reliability optimization in a power plant." *2018 International Conference on Electrical Sciences and Technologies in Maghreb (CISTEM)*. IEEE, 2018.
- CHEBOUBA Billal Nazim, Mohamed Arezki MELLAL, and Smail ADJERID. "Multi-objective pipeline network reliability optimization by particle swarm optimization." *The 8th International Symposium on Hydrocarbons and Chemistry*, Boumerdes, Algeria.
- CHEBOUBA Billal Nazim, Mohamed Arezki MELLAL, and Smail ADJERID. "Overall reliability optimization of a production system." *International Symposium on Technology & Sustainable Industry Development, ISTSID'2019*, El Oued, Algeria.

- CHEBOUBA Billal Nazim, Mohamed Arezki MELLAL, and Smail ADJERID. "Three computational intelligence methods for system reliability." *The 2nd International Workshop on Signal Processing Applied to Rotating Machinery Diagnostics SIGPROMD'2018*, 29-30 April 2018, Djelfa, Algeria.
- CHEBOUBA Billal Nazim, Mohamed Arezki MELLAL, and Smail ADJERID. "System design optimization under constraint of reliability.*" International Conference on Advanced Mechanics and Renewable Energies CIMAER'2018 28-29 November 2018*, Boumerdes, Algeria.

3. Structure de la thèse

La structure de cette thèse reflète ces contributions avec cinq chapitres suivis d'une conclusion générale :

- **Chapitre I** : Dans ce chapitre les définitions et notions de base de la sûreté de fonctionnement sont présentées ainsi que les attributs, les entraves et les moyens.
- **Chapitre II :** Dans ce chapitre, une introduction à la théorie générale de l'optimisation et à sa formulation mathématique a été relatée. Un aperçu des algorithmes d'optimisation classiques et traditionnels a été donné. La deuxième section du chapitre traite des méthodes d'optimisation classiques et des algorithmes d'optimisation inspirés de la nature, de leurs caractéristiques, ainsi que de leurs avantages et inconvénients. Les différents algorithmes d'intelligence en essaim inspirés de la nature ont été décrits.
- **Chapitre III** : L'objectif principal de ce chapitre est de présenter, implémenter et comparer trois algorithmes métaheuristiques bien connus : Stochastic Fractal Search (SFS), Cuckoo Search (CS) et Particle swarm optimization (PSO), afin de résoudre un problème d'allocation de fiabilité-redondance pour une usine pharmaceutique contenant dix sous-systèmes connectés en séries. Une fonction de pénalité a dû être implémentée dans chaque algorithme pour éliminer les solutions infaisables. Les résultats révèlent la supériorité de SFS sur PSO et CS.
- **Chapitre IV :** Dans ce chapitre, le coût de fiabilité et le volume d'un système de protection contre la survitesse dans une centrale électrique ont été étudiés. Le travail actuel aborde trois scénarios, dans le premier nous avons considéré le problème avec deux objectifs la fiabilité globale et le coût, dans le deuxième scénario nous avons considéré trois objectifs

la fiabilité globale, le coût global et le poids global. Les deux premiers scénarios ont été résolus en utilisant NSGA-II. Dans le troisième et dernier scénario, nous avons considéré le problème avec deux objectifs la fiabilité globale et le coût avec une fiabilité minimale admissible comme contrainte supplémentaire.

- **Chapitre V :** Le but de ce chapitre était d'étudier le problème d'optimisation de la fiabilité d'un système multi-objectif avec des paramètres flous. Une approche de solution basée sur quatre étapes a été proposée : appliquer la procédure de fonction de classement pour défuzzifier les paramètres flous en valeurs nettes, utiliser les NSGA-II et NSGA-III, identifier le meilleur front de Pareto par la méthode d'espacement, et enfin réduire le meilleur front de Pareto par l'analyse de clustering. Une étude de cas numérique traitée dans la littérature comme un problème d'allocation de redondance mono-objectif floue a été étudiée dans le présent chapitre en tant que problème d'allocation de fiabilité-redondance multi-objectif floue.

Chapitre I

Sûreté de fonctionnement

Chapitre I

Sûreté de fonctionnement

I.1 Introduction

La complexité croissante des systèmes industriels, leur implication croissante dans la vie économique et sociale, la nécessité de minimiser les coûts de construction et d'exploitation dans un marché concurrentiel exigent une grande attention dès la phase de conception des systèmes. La sûreté de fonctionnement comprend quatre concepts fondamentaux tels que la fiabilité, la maintenabilité, la disponibilité, la sécurité et le coût [1].

Les systèmes industriels, tels que les systèmes de production, sont souvent constitués de plusieurs sous-systèmes comprenant eux-mêmes des composants dépendants. Les performances des systèmes dépendent des configurations adoptées, ainsi que les liens et dépendances entre chaque composant. La modélisation et l'évaluation des métriques de sûreté de fonctionnement est un vaste domaine d'investigation.

Dans le monde industriel, la phase de conception consiste à étudier les différentes alternatives fonctionnelles et technologiques du système afin de répondre au cahier des charges. Chaque alternative est caractérisée par la sûreté de fonctionnement du système. La conception des systèmes peut alors être reformulée par un problème d'optimisation combinatoire. L'optimisation des performances d'un système en phase de conception est aujourd'hui l'une des principales préoccupations des entreprises car le rendement de ces dernières est vital à l'entreprise pour être compétitive dans le domaine.

I.2 Définitions

La sûreté de fonctionnement repose sur plusieurs concepts de base dont les définitions et les interprétations n'ont cessé d'évoluer et de s'affirmer au fil du temps. Ces concepts sont : Fiabilité, Maintenabilité, Disponibilité, Sécurité et Coût, mais également complétés par d'autres compétences telles que la durabilité, ou une combinaison de ces aptitudes. La sûreté de fonctionnement reflète la confiance qui peut être placée dans un système.

I.3 Attributs

I.3.1 Fiabilité *R (t)*

"La capacité de tout composant, sous-système ou système (*E*) à exécuter une fonction requise, dans des conditions environnementales et opérationnelles données et pendant une période donnée $[0,t]$ " [2]

La fiabilité est définie à partir du taux de défaillance qui varie dans le temps comme le montre la figure 1.

La fiabilité d'un système avec un taux de défaillance constant pendant la période utile, est estimée par [3,4] :

$$\boldsymbol{R(t) = e^{-\lambda t}} \qquad \text{I.1}$$

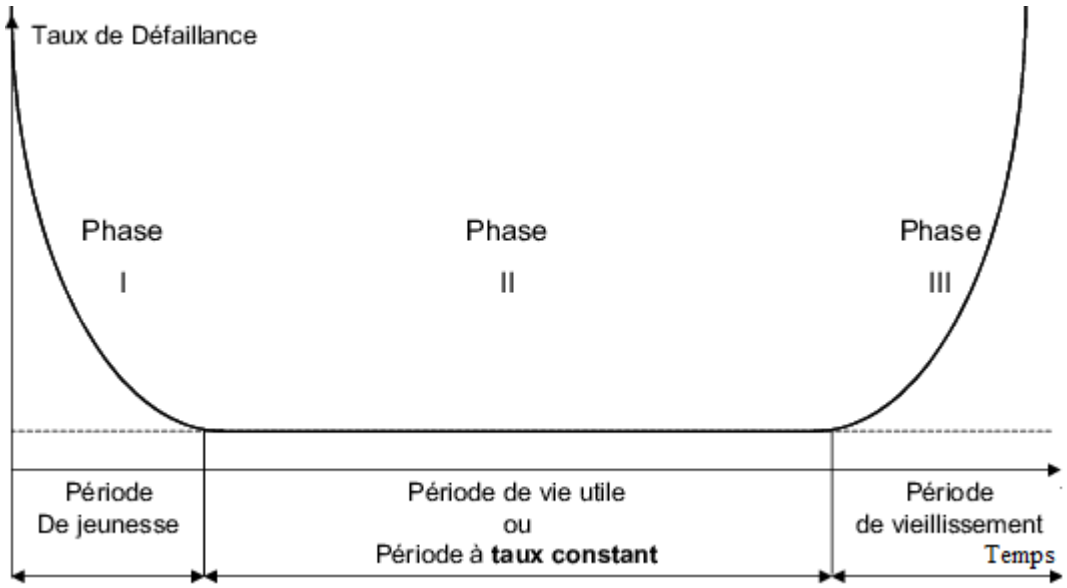

Figure I-1 Courbe en baignoire [5]

I.3.2 Maintenabilité *M (t)*

"La capacité de tout composant, sous-système ou système, dans des conditions d'utilisation spécifiées, à être conservé ou restauré dans un état dans lequel il peut exécuter ses fonctions requises, lorsque la maintenance est effectuée dans des conditions spécifiées et en utilisant des procédures et des ressources prescrites" ***(BS4778).***

En d'autres termes, c'est la probabilité $M(t)$ que la maintenance peut être effectuée à l'instant t pour que le composant, le sous-système ou le système puisse exécuter ses fonctions [1].

I.3.3 Disponibilité *A (t)*

"La capacité d'un élément soit à être en fonctionnement actif, soit à pouvoir fonctionner si nécessaire à un instant donné ou sur une période de temps donnée" (BS4778).

La disponibilité à instant t est [6] :

$$A(t) = Pr(\text{item is functioning at time } t) \quad \text{I.2}$$

La disponibilité moyenne A_{av} désigne la proportion moyenne de temps pendant laquelle l'élément fonctionne. Si nous avons un élément qui est réparé dans un état « comme neuf » à chaque fois qu'il tombe en panne, la disponibilité moyenne est de [7] :

$$A_{av} = \frac{MTTF}{MTTF + MTTR} \quad \text{I.3}$$

Où : MTTF (temps moyen avant la panne) désigne le temps de fonctionnement moyen de l'élément, et MTTR (le temps moyen de réparation) désigne le temps d'arrêt moyen après une panne.

Lorsqu'on considère un système de production, la disponibilité moyenne est appelée la régularité de la production.

I.3.4 La sécurité *S (t)*

La sécurité est la capacité de tout composant, sous-système ou système, dans des conditions d'utilisation spécifiées, à exécuter une fonction requise sans causer la mort, des blessures, des maladies professionnelles, des dommages ou la perte d'équipements [8,9].

I.3.5 Le coût *C (t)*

De nos jours en raison du niveau de compétitivité dans le monde industriel, le coût est également pris en compte en tant qu'attributs de fiabilité.

I.3.6 Autres métriques de sûreté de fonctionnement

Plusieurs métriques de la Sdf peuvent être calculées à partir des mesures de probabilités. Les paramètres suivants caractérisent les durées moyennes [5, 10,11]

- **Temps moyen entre les pannes (MTBF) :** c'est le terme le plus couramment utilisé pour décrire la fiabilité du produit est le temps moyen entre les pannes. (MTBF). Ce terme mesure le taux de défaillance du produit pendant sa durée de vie normale.
- **Temps moyen avant la panne (MTTF) :** est un terme couramment utilisé pour décrire la fiabilité d'un système non réparable. MTTF décrit la durée moyenne d'exécution d'un ensemble de systèmes jusqu'à la défaillance du système. Ce terme est généralement utilisé dans les cas où le produit ne sera pas réparé.

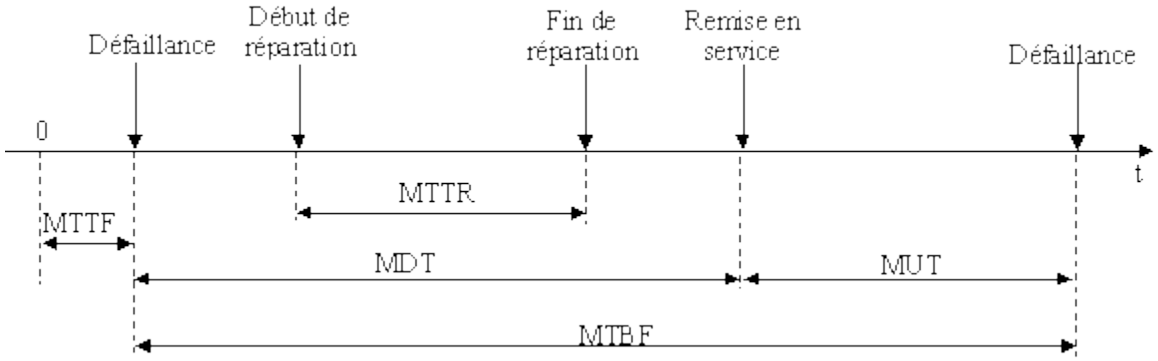

Figure I-2 Métriques de sûreté de fonctionnement

- **Le temps moyen de réparation (MTTR) :** est le terme le plus couramment utilisé pour décrire la maintenabilité d'un système. C'est la somme du temps nécessaire pour réparer toutes les pannes divisée par le nombre total de pannes. Le temps nécessaire pour corriger la panne comprend généralement le dépannage, l'isolation des pannes, la réparation et tout test nécessaire pour vérifier que le problème a été résolu. En termes simples, il s'agit du moment où le client ne peut pas utiliser le produit jusqu'au moment où le client peut l'utiliser.

I.4 Entraves

I.4.1 Fautes

Un service correct est fourni lorsque le service met en œuvre la fonction système. Un échec de service est un événement qui se produit lorsque le service fourni s'écarte du service correct, un

service échoue soit parce qu'il n'est pas conforme à la spécification fonctionnelle, soit parce que cette spécification n'a pas correctement décrit la fonction du système. Une panne de service est une transition d'un service correct à un service incorrect, c'est-à-dire à ne pas mettre en œuvre la fonction du système. La période de livraison d'un service incorrect est une panne de service, la transition d'un service incorrect à un service correct est une restauration de service. L'écart par rapport au service correct peut prendre différentes formes appelées modes de défaillance de service et classés en fonction de la gravité des défaillances [12].

I.4.2 Erreurs

Étant donné qu'un service est une séquence d'états externes du système, une panne de service signifie qu'au moins un (ou plusieurs) état externe du système s'écarte de l'état de service correct. L'écart est appelé une erreur. La cause présumée ou supposée d'une erreur est appelée faute.

I.4.3 Défaillances

Dans la plupart des cas, un défaut provoque d'abord une erreur dans l'état de service d'un composant qui fait partie de l'état interne du système et l'état externe n'est pas immédiatement affecté. Il est important de noter que de nombreuses erreurs n'atteignent pas l'état externe du système et provoquent une défaillance. Un défaut est actif lorsqu'il provoque une erreur, sinon il est dormant.

I.5 Moyens

Telle qu'elle a été développée au cours des trois dernières décennies, la sûreté de fonctionnement est un concept intégrateur qui englobe les moyens suivants [12] :

I.5.1 Préventions des fautes

Moyens pour empêcher l'apparition ou l'introduction de défauts, il est atteint par des techniques de contrôle de la qualité employées lors de la conception et de la fabrication du matériel et des logiciels. Ils incluent la programmation structurée, la dissimulation d'informations, la modularisation, etc..., pour les logiciels, ainsi que des règles de conception rigoureuses pour le matériel : Blindage, durcissement aux radiations, etc.

I.5.2 Tolérance aux fautes

Moyens d'éviter l'arrêt de service en présence de pannes, c'est-à-dire de préserver la fourniture d'un service correct en présence de pannes actives. Il est généralement mis en œuvre par la détection d'erreurs et la réparation ultérieure du système.

I.5.3 Élimination des fautes

Moyens de réduire le nombre et la gravité des défauts, et cela est effectué à la fois pendant la phase de développement et pendant la durée de vie opérationnelle d'un système.

I.5.4 Préventions des fautes

Moyens d'estimer le nombre actuel, l'incidence future et les conséquences probables des défauts. L'évaluation a deux aspects :

- **Évaluation qualitative :** qui vise à identifier, classer, hiérarchiser les modes de défaillance, ou les combinaisons d'événements (défaillances de composants ou conditions environnementales) qui conduiraient à des défaillances du système.
- **Évaluation quantitative :** qui vise à évaluer en termes de probabilités dans quelle mesure certains des attributs de sûreté de fonctionnement sont satisfaits ; ces attributs sont alors considérés comme des mesures de fiabilité.

I.6 Fiabilité des systèmes complexes

La majorité des systèmes sont constitués de sous-systèmes et/ou de composants, tels que : système mécanique, système d'alimentation, ou une installation industrielle. Fonctionnellement, ces sous-systèmes ou composants sont relies en série ou en parallèle ou la combinaison des deux (série-parallèle) [13]

Nous présentons dans cette partie les différents modelés de systèmes utilises en sureté de fonctionnement [1,2] [14].

I.6.1 Système en série

Un système en série signifie que chaque composant du système est destiné au fonctionnement du système. En d'autres termes, la défaillance d'un de ses composants provoque la défaillance du

système. Schématiquement, une instruction doit passer par chaque élément du système (figure II-3).

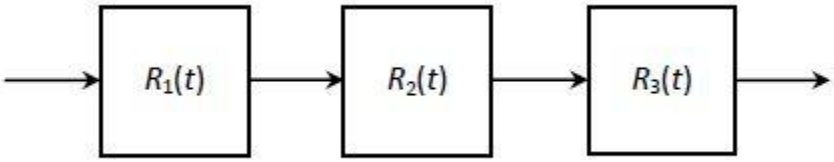

Figure I-3 Système en série.

Chaque composant du système a sa fiabilité et son taux de défaillance qui dépend des autres composants. La fiabilité d'un système série est la probabilité que tous les composants fonctionnent simultanément dans un intervalle de temps donné. Si les défaillances des composants sont indépendantes, alors la probabilité que tous les composants soient fonctionnels est donnée par le produit de leur fiabilité.

$$R_{system}(t) = R_1(t) \times R_2(t) \times R_3(t) \times \ldots \times R_n(t) = \prod_{i=1}^{n} R_i(t) \qquad \text{I.4}$$

I.6.2 Système parallèle

Un système parallèle signifie que des sous-systèmes ou des composants sont montés en parallèle. En d'autres termes, au moins un de ces composants doit fonctionner pour que le système fonctionne. Schématiquement, il existe plusieurs chemins alternatifs pour aller de l'entrée de système à la sortie (figure II-4).

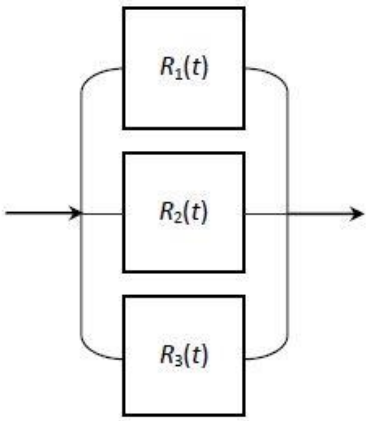

Figure I-4 Système parallèle

Dans les systèmes parallèles, un seul élément est requis pour que le système fonctionne, les autres éléments sont redondants. Cette redondance augmente la fiabilité globale du système, car il faudra que tous les éléments connectés en parallèle tombent en panne pour que le système soit considéré comme défaillant. Si les défaillances sont indépendantes, la probabilité de défaillance du système est égale au produit de toutes les probabilités de défaillance de ces éléments.

$$Q_{system}(t) = Q_1(t) \times Q_2(t) \times Q_1(t) \times \dots Q_n(t) = \prod_{i=1}^{n} Q_i(t) \quad \text{I.5}$$

Si tous les composants ont le même $Q(t)$ alors :

$$Q_{system}(t) = Q^n(t) \quad \text{I.6}$$

Où : $R(t) = 1 - Q(t)$ par conséquent :

$$R_{system}(t) = 1 - \prod_{i=1}^{n} \big(1 - R_i(t)\big) \quad \text{I.7}$$

I.7 Stratégies d'amélioration de la fiabilité

Dans le domaine industriel, les systèmes doivent effectuer un ensemble d'opérations identiques dans une période de temps donnée de manière aussi fiable que possible. Ces systèmes sont constitués de sous-systèmes connectés en série ou en parallèle avec des composants redondants. Différentes alternatives de conception peuvent être atteintes grâce à l'utilisation des composants non-identiques en parallèle à chaque étape. Dans le domaine de la fiabilité des systèmes, plusieurs approches sont utilisées afin d'améliorer les performances des systèmes. Kuo et al [15] ont cité les trois principales méthodes utilisées pour améliorer la fiabilité du système :

- Augmentation de la fiabilité des composants (allocation de fiabilité).
- Ajout de composants redondants (allocation de redondance).
- Augmentation de la fiabilité des composants et ajout de composants redondants en même temps (allocation de fiabilité et de redondance).

Les premières études sur la SDF se sont focalisées sur l'allocation de la fiabilité. Puis très vite les chercheurs ont commencé à traiter de l'allocation de la redondance et de la combinaison de ces deux approches [16, 17]. Il est à noter que le terme allocation signifie allocation par optimisation. Nous donnons ci-après une brève description de la notion d'allocation.

I.8 Conclusion

Dans ce chapitre, sont présentées les notions de base de la sûreté de fonctionnement portant sur ses attributs, ses entraves et ses moyens. L'ingénierie de la sûreté de fonctionnement de ces trois dernières décennies montre que la sûreté de fonctionnement est un ensemble de concepts intégrateur englobant des moyens telles que la prévention et la suppression des pannes, la prévision et la tolérance aux pannes visant à maintenir le fonctionnement efficace des installations industrielles, quelle que soit la source d'un événement redouté.

D’autre part, le cycle de vie des composants des systèmes industriels est censé suivre une « courbe en cuve ». Cependant, la réalité est différente. Les composants, selon leur association et leur comportement influent différemment sur les performances des installations. Ainsi, l’étude des performances des installations constitue un des aspects essentiels de nos travaux. A cet effet, nous présentons les principales méthodes de modélisation mathématique des systèmes considérés. Ensuite, nous décrivons les principales méthodes pour augmenter la fiabilité globale des systèmes

en augmentant soit la fiabilité, la redondance des composants ou en procédant à la combinaison des deux approches en même temps.

Chapitre II

Approches d'optimisation et de résolution

Chapitre II

Approches d'optimisation et de résolution

II.1 Introduction

Selon le dictionnaire anglais de Cambridge, l'optimisation est "l'acte de faire quelque chose d'aussi bon que possible". Mathématiquement parlant, l'optimisation consiste à identifier la solution, c'est-à-dire une ou plusieurs variables de décision, en minimisant ou en maximisant une fonction donnée ou un ensemble de fonctions tout en remplissant un ensemble de contraintes. En conséquence, deux types de problèmes d'optimisation sont distingués dans la littérature : les problèmes d'optimisation à objectif unique et multi-objectif.

Dans le passé, les techniques d'optimisation les plus couramment utilisées étaient des méthodes basées sur le gradient qui utilisait des informations du gradient pour rechercher un espace de solutions près d'un point de départ initial [20]. Les méthodes basées sur le gradient ne peuvent pas résoudre facilement les problèmes d'optimisation non convexes. D'autres types de méthodes d'optimisation, appelées algorithmes métaheuristiques, sont des méthodes stochastiques. Ces méthodes conviennent à la recherche universelle car elles peuvent explorer et trouver des domaines prometteurs dans l'espace de recherche avec un temps de calcul raisonnable [21, 22].

Ces algorithmes ont tendance à bien fonctionner pour la plupart des applications de problèmes d'optimisation dans différents domaines, à savoir les mathématiques appliquées, l'ingénierie, la médecine, l'économie et d'autres sciences. Ces méthodes sont largement utilisées pour concevoir divers systèmes en génie civil, mécanique, électrique et industriel [23, 24].

Dans la section 2 de ce chapitre, nous présentons quelques définitions de base liées à un problème d'optimisation. Dans la section suivante, nous présentons les principales méthodes pour résoudre les problèmes d'optimisation. Dans la section 4, nous passons en revue certains des algorithmes évolutionnaires conçus pour résoudre les problèmes d'optimisation à objectif unique et multi-objectifs.

II.2 Méthodes d'optimisation classiques

II.2.1 Introduction

L'objectif principal de l'optimisation est de trouver le meilleur résultat possible à partir des ressources disponibles sans violer un ensemble de contraintes. Le résultat pourrait être la maximisation ou la minimisation de la fonction objective. La maximisation et la minimisation sont

interchangeables puisque l'une peut être convertie en l'autre et vice versa. Plusieurs algorithmes d'optimisation ont été développés et chaque algorithme est adapté à une classe particulière de problèmes. L'efficacité de l'algorithme est déterminée en termes de complexité de calcul, de temps et d'espace. Le taux de convergence de l'algorithme, les ressources de calcul et l'optimalité de la solution obtenue pour le problème utilisé jouent aussi un rôle majeur dans l'efficacité de l'algorithme. Cela nécessite à son tour des techniques mathématiques pour résoudre les équations qui pourraient avoir des conditions initiales et des limites ou des contraintes sur les variables.

La recherche opérationnelle est la branche de l'optimisation concernée par les techniques d'analyse mathématique nécessaires pour produire le résultat optimal [25]. Cette discipline traite des méthodes analytiques qui aident à la prise de décision complexe lors de la résolution de problèmes. La recherche opérationnelle implique la construction de modèles mathématiques dans lesquels des programmes informatiques peuvent être écrits pour résoudre des problèmes qui ont été modélisés dans un cadre mathématique. Lorsque les problèmes impliquent la stochasticité, des techniques de programmation statistique doivent être employées. Les distributions de probabilité de ces variables stochastiques doivent être connues à l'avance afin d'ajuster correctement les modèles. Les techniques de programmation numérique sont les méthodes classiques d'optimisation qui ont été développées depuis le début des années 1940. Les premiers développements ont été réalisés par Cauchy, Newton et Lagrange [26], et depuis lors plusieurs techniques ont été proposées. Les avancées ont été rendues possibles par le développement parallèle de la technologie de calcul à grande vitesse sur laquelle ces algorithmes ont été programmés. Les méthodes discutées dans ce chapitre sont très populaires et ont été utilisées avec succès pour résoudre des problèmes d'optimisation avec ou sans contraintes.

En complément de ces méthodes traditionnelles, les algorithmes d'optimisation évolutive ont été développés dans les années 1960. Le développement pionnier dans la classe des algorithmes évolutionnaires a été fait par l'invention de l'algorithme génétique, et depuis lors, cette catégorie d'algorithmes s'est développée à pas de géant. L'application de l'intelligence informatique dans ces algorithmes a d'abord été réalisée avec l'invention des réseaux de neurones [27]. La création d'algorithmes évolutionnaires a ouvert la voie au développement d'algorithmes de recherche utilisant des populations d'agents qui recherchent l'optimum en parallèle. Ce parallélisme inhérent à ces algorithmes les a rendus plus efficaces que leurs homologues classiques. Une différence

majeure est que les algorithmes classiques sont déterministes alors que les algorithmes de recherche évolutionnaire sont non déterministes. Le caractère aléatoire incorporé dans la recherche améliore la diversité de l'algorithme et permet à l'algorithme de sortir des optima locaux. Cela augmente l'efficacité des algorithmes dans la recherche, et leur taux de convergence sur l'optimum global est plus élevé.

II.2.2 Modèle mathématique d'optimisation

La solution au problème d'optimisation commence par la définition de l'énoncé du problème avec des objectifs clairement définis. Les équations mathématiques pour le(s) objectif(s) et les contraintes du problème doivent être formulées. Les valeurs initiales des fonctions, le cas échéant, et les limites des paramètres doivent être identifiées. La valeur de la fonction objectif est une indication de la qualité de la solution atteinte qui peut être cadrée pour la maximisation ou la minimisation. Une fonction objectif de maximisation peut être convertie en une fonction de minimisation en prenant le négatif de la fonction et vice versa. De même, les contraintes pourraient être écrites sous forme d'équations du type égalité ou inégalité. Selon le nombre de variables dont dépend la fonction objectif, la fonction devient multidimensionnelle ; généralement la dimension est *d*.

Soit la fonction objectif donnée par :

$$f(X) = f(x_1, x_2, ..., x_d) \qquad \text{II.1}$$

Soit les contraintes du problème données par :

$$\begin{aligned} &g_i(X) = 0, \ \text{i=1,2, ... P} \\ &h_j(X) \geq 0, \ \text{j=1,2, ... Q} \\ &h_j(X) \leq 0, \ \text{j=1,2, ... Q} \end{aligned} \qquad \text{II.2}$$

Dans le cas de problèmes d'optimisation multi-objectifs, le nombre de fonctions objectifs est supposé égal à *K* et les fonctions objectifs multiples sont données par :

$$f(X_k) = f(x_{k1}, x_{k2}, ..., x_{kd}), \ \text{k=1,2, ... k} \qquad \text{II.3}$$

Ces fonctions pourraient être de différents types : continus ou discrets, déterministes ou non déterministes avec des paramètres aléatoires qui leur sont inculqués. Les algorithmes traditionnels sont généralement applicables si la fonction est continue et dérivable. Lorsque la fonction est contrainte, il s'agit d'un problème de programmation avec contrainte contrairement à un problème de programmation sans contrainte. S'il n'y a qu'un seul objectif à optimiser, c'est un problème d'optimisation à objectif unique, alors que s'il y a plusieurs objectifs, c'est un problème d'optimisation multi-objectifs. La fonction peut être concave ou convexe selon les contours du tracé de surface de la fonction. Les fonctions concaves ont un maximum alors que les fonctions convexes ont un minimum. Un problème est dit réalisable s'il existe au moins un ensemble de variables qui satisfont la fonction objectif et les contraintes. Un problème est irréalisable s'il n'est pas possible de trouver au moins un ensemble de variables de conception qui satisfont le(s) objectif(s) et les contraintes. Si le problème est irréalisable, l'ensemble de solutions sera vide. Pour un problème illimité, il n'y a pas de solution optimale car il est toujours possible de trouver une solution meilleure que la solution existante au problème.

II.2.3 Techniques de résolution classiques

Sur la base de la nature et des caractéristiques de la fonction objectif, des contraintes, des variables de conception et de tout autre paramètre associé au problème, les techniques de résolution des problèmes d'optimisation, dont certaines sont applicables à la fois aux problèmes d'optimisation linéaires et non linéaires classiques, sont décrites ci-dessous.

II.2.3.1 Programmation linéaire

La programmation linéaire est une technique de programmation numérique conçue pour résoudre des problèmes d'optimisation où la fonction objectif et les contraintes (égalité et inégalité) sont exprimées sous forme d'équations linéaires (fonctions). Les méthodes de programmation linéaire les plus connues : méthode révisée du simplexe, Algorithme de Karmarkar, principe de dualité, Théorie du transport. Un exemple de modélisation mathématique d'un problème de programmation linéaire est :

$$\text{Minimize } f(X) = 2x_1 + 4x_2 - 5x_3 + x_4 \qquad \text{II.4}$$

Sous contraintes :

$$g_1(X): x_1 + x_2 = 6$$
$$g_2(X): 2x_1 - x_4 = 8 \qquad \text{II.5}$$
$$g_3(X) = x_3 + x_2 < 25$$

En général, il y a un nombre de variables *d* et un nombre *m* d'équations (contraintes) dans ces variables avec $d \geq m$. Les contraintes d'inégalité peuvent être converties en contraintes d'égalité par l'inclusion de termes supplémentaires dans les équations de contrainte.

Habituellement, dans les problèmes de programmation linéaire, les variables de décision sont des nombres positifs. L'espace des solutions réalisables sera convexe avec un optimum global (minimum). Lorsque le nombre de variables de décision est supérieur à deux, l'espace des solutions devient un hyperplan en *d* dimensions [29].

II.2.3.2 Programmation non-linéaire

Il s'agit d'une technique d'optimisation dans laquelle la fonction objectif et tout ou partie des contraintes sont des équations non linéaires. Plusieurs algorithmes ont été proposés pour la résolution de tels problèmes de programmation non linéaire, dont les plus connues sont : Optimisation quadratique, Géométrie algorithmique, Programmation dynamique, programmation stochastique, Multiplicateur de Lagrange.

III.2.3.2.1Programmation dynamique

La programmation dynamique est une méthode pour résoudre des problèmes d'optimisation nécessitant des décisions prises en tant que flux séquentiel à travers le problème. Les décisions pourraient être requises à différents niveaux du système. Cette technique de résolution de tels problèmes a été développée et présentée par Richard Bellman [29] en 1954. Les systèmes physiques peuvent être modélisés comme des machines à états finis où l'ensemble des paramètres du système détermine l'état du système. Le système est modélisé ou conçu comme passant d'un état à un autre dans une séquence temporelle. Le système change d'état en fonction d'une décision prise à cet instant. Enfin, une fonction qui dépend de ces paramètres, atteint une valeur déterminée par l'enchaînement des décisions prises et donc de changement d'état (transformation de variables

d'état) du système. Cette valeur atteinte par la fonction peut être maximale ou minimale mais elle doit être optimale.

Considérant l'ensemble de toutes les décisions possibles qui pourraient être prises à différents moments (en séquence temporelle), le changement d'état pour chacune de ces décisions affecte les états consécutifs. Cela affecte à son tour la valeur finale de la fonction objectif ou une sortie équivalente à la valeur de la fonction. Lorsque toutes ces possibilités sont prises en compte, l'espace de recherche de la solution optimale devient énorme. Pour réduire la dimension du problème afin que la solution puisse être obtenue raisonnablement en un temps fini, elle est réduite à une séquence de problèmes de décision sans prendre en compte l'effet de chaque décision à un instant ultérieur. Par conséquent, le problème à *K* étapes est décomposé en *K* problèmes à une étape qui peuvent être résolus beaucoup plus efficacement que le problème original à *K* étapes. La solution globale doit être la même que celle consistant à combiner les *K* solutions différentes.

III.2.3.2.2Optimisation linéaire en nombres entiers

Lors de la résolution de problèmes de conception technique, les variables de la fonction peuvent atteindre des valeurs réelles qui peuvent être positives ou négatives avec des parties entières et fractionnaires [30, 31]. Si la variable représente la longueur ou le poids, alors les valeurs fractionnaires telles que 4,89 m ou 78,98 kg ont un sens. Si la variable représente des personnes ou des objets, les valeurs fractionnaires telles que 5,7 ou 0,77 n'ont pas de sens. Les valeurs pourraient être arrondies, mais cela pourrait affecter d'autres valeurs et, par conséquent, les contraintes du problème pourraient ne pas être satisfaites. La valeur de la fonction objectif pourrait également être différente de celle de l'optimum en raison de l'arrondi aux valeurs entières les plus proches des variables de conception.

Exemple :

$$f(X) = 3x_1 + x_2^2 - 6x_3 \qquad \text{II.6}$$

Sous contraintes :

$$x_1 \geq 0, \qquad \text{II.7}$$
$$-5 \leq x_2 \leq 5$$

$\boldsymbol{x_1, x_2, x_3}$ sont nécessairement entiers.

Lorsque toutes les variables d'un problème sont restreintes à des valeurs entières, le problème d'optimisation devient un problème de programmation en nombres entiers. Dans le cas où certaines variables ne sont pas des valeurs entières alors le problème devient mixte. Le problème de programmation en nombres entiers peut être linéaire ou non linéaire. Plusieurs méthodes ont été développées pour résoudre le problème de programmation en nombres entiers, mais les performances de la méthode dépendent du problème ou de l'application.

III.2.3.2.3 Programmation Stochastique

La programmation stochastique est une méthode d'optimisation où certaines des variables associées à la fonction objectif sont des variables aléatoires avec une distribution de probabilité définie. Dans les problèmes de conception technique, il pourrait y avoir une limite minimale et une limite maximale pour les variables de conception associées et la valeur réelle pourrait être placée de manière aléatoire dans ces limites. Chaque fois que des variables aléatoires sont impliquées dans le problème, cela devient un problème de programmation stochastique. Même si les variables associées au problème sont de nature aléatoire, les équations mathématiques liées au problème le rendent linéaire ou non linéaire, géométrique ou dynamique, et ces problèmes peuvent être résolus en utilisant les techniques standards existantes.

Exemple : maximiser ou minimiser

$$f(X) = w_1 x_1 + w_2 x_2 + \ldots + w_d x_d \qquad \text{II.8}$$

Sous contraintes :

$$\begin{aligned} a x_1 - b x_4 &> 0 \\ \textstyle\sum_{i=1}^{d} x_i &= 0 \end{aligned} \qquad \text{II.9}$$

Où : w_j, $j = 1, 2, \ldots, d$ sont des coefficients et a et b sont des variables aléatoires avec une distribution de probabilité uniforme dans l'intervalle $[0, 1]$. C'est un problème de programmation linéaire stochastique.

III.2.3.2.4 Multiplicateur de Lagrange

Lorsque la fonction objectif est continue et dérivable avec des contraintes d'égalité, la méthode du multiplicateur de Lagrange peut être utilisée.$f(X) = f(x_1, x_2)$ est la fonction objectif

à minimiser ou à maximiser (fonction à deux variables de conception) et la contrainte$g(X) = g(x_1, x_2) = 0$. La fonction de Lagrange est formulée comme :

$$L(x_1, x_2, \lambda) = f(x_1, x_2) + \lambda g\left(x_{1,} x_2\right) \qquad \text{II.10}$$

Où : x_1^* et x_2^* représentent les points extrêmes sur la surface de la fonction où se produisent des maxima ou des minima et λ est le multiplicateur de Lagrange. Les conditions nécessaires pour x_1^* et x_2^* sont dérivées de l'équation II.10 en différenciant la fonction *L* en respectantx_1, x_2 et λ . Ces conditions sont données par les équations II.11 ci-dessous :

$$\begin{aligned} \left(\frac{\partial f}{\partial x_1} + \lambda \frac{\partial g}{\partial x_1}\right)_{x_1^* x_2^*} &= 0 \\ \left(\frac{\partial f}{\partial x_2} + \lambda \frac{\partial g}{\partial x_2}\right)_{x_1^* x_2^*} &= 0 \\ \left(g(x_1, x_2)\right)_{x_1^* x_2^*} &= 0 \end{aligned} \qquad \text{II.11}$$

Les trois équations ci-dessus sont les conditions nécessaires de x_1^* et x_2^* pour être les points extrêmes. La solution de ces équations produit la valeur maximale ou minimale de la fonction $f(X)$ en respectant $g(X)$. Cette technique peut être étendue aux problèmes à contraintes multiples en introduisant un multiplicateur de Lagrange pour chaque contrainte.

II.3 Algorithmes inspirés de la nature

II.3.1 Introduction

Les algorithmes d'optimisation inspirés de la nature sont des algorithmes métaheuristiques développés à partir des principes de l'évolution biologique, du comportement des essaims et des processus physiques et chimiques [32]. Les algorithmes d'optimisation inspirés de la nature sont des techniques d'intelligence computationnelle bio inspirées car ils intègrent l'intelligence dans les algorithmes. La recherche sur ces algorithmes a progressé considérablement au cours des deux

dernières décennies. La première percée s'est produite dans les années 1960 avec le développement pionnier de l'algorithme génétique évolutif (AG) par John Holland et ses collègues de l'Université du Michigan [33]. Depuis lors, plusieurs algorithmes évolutifs ont été proposés, dont de nombreuses variantes et hybrides de l'AG. Les algorithmes évolutionnaires sont basés sur l'évolution biologique, et AG est l'un des exemples classiques de cette catégorie. Les algorithmes d'intelligence en essaim sont une autre catégorie d'algorithmes bio inspirés qui s'inspirent du comportement des essaims dans la nature, tels que les vols d'oiseaux, la traînée de fourmis, la formation de poissons, etc. L'utilisation de populations d'agents de recherche combinée à des heuristiques a un effet profond sur les solutions à des problèmes de conception technique complexes. Les algorithmes inspirés de la nature sont relativement novateurs pour obtenir facilement des solutions efficaces avec le moins de ressources de calcul. Le partage d'informations et l'interaction sociale entre les membres de leur propre espèce ainsi qu'avec l'environnement par des agents biologiques tels que les fourmis, les abeilles, les corbeaux, les chauves-souris, les coucous, etc. a conduit à l'essor de l'intelligence collective. Ils s'adaptent à l'environnement et utilisent au mieux les ressources disponibles, qu'il s'agisse de partager de la nourriture ou de toute tâche à accomplir, avec la coopération de leur groupe. Par conséquent, tout algorithme modélisé sur leur comportement peut facilement trouver des solutions à des problèmes complexes.

La majorité des algorithmes inspirés de la nature sont largement classés sous les algorithmes évolutifs (EA) et les algorithmes d'intelligence en essaim (SI). La troisième catégorie d'algorithmes inspirés de la nature est basée sur des processus physiques et chimiques, et le recuit simulé (SA) est un algorithme célèbre de cette classe [34].

Figure II-1. Essaim de poissons dans un mouvement collectif.

L'étude de la nature, de la flore, de la faune et de l'écosystème en général a été à l'origine du développement d'algorithmes d'optimisation inspirés de la nature et des centaines d'algorithmes inspirés de la nature, leurs variantes et hybrides ont été proposés. La figure II.1 montre une image d'un essaim de poissons en mouvement collectif dans l'océan.

II.3.2 Algorithmes traditionnels versus algorithmes inspirés de la nature

Les algorithmes d'optimisation constituent une large classe d'algorithmes avec une base mathématique et ont été conçus pour trouver la solution optimale sous contraintes. Les algorithmes traditionnels ne garantissent pas toujours la solution optimale globale puisque la solution finale dépend des conditions initiales. Si ces algorithmes débutent au même point initial, ils déboucheront inévitablement à la même solution finale puisque les algorithmes traditionnels sont déterministes [35]. Tout problème qui semble extrêmement complexe ou difficile à résoudre à l'aide de méthodes traditionnelles peut être résolu en s'inspirant de la nature. La motivation peut être acquise en étudiant la nature et la manière dont ces problèmes sont traités chez les espèces biologiques. Les algorithmes inspirés de la nature ne nécessitent pas de calcul de dérivées, par conséquent, ils sont sans gradient et ne sont pas spécifiques à un problème. Même si l'algorithme démarre au même point initial pour des exécutions répétées, il ne se retrouvera pas avec la même solution. Il y a une

certaine stochasticité intégrée dans l'algorithme, avec des processus de Lévy et des marches aléatoires.

L'une des approches majeures de l'optimisation technique consiste à rechercher parmi toutes les solutions réalisables, l'optimum global ou la meilleure solution qui correspond au problème et à ses contraintes. Fondamentalement, les algorithmes inspirés de la nature sont des algorithmes de recherche métaheuristiques qui recherchent l'optimum en parallèle, avec une population d'agents [36]. Les algorithmes inspirés de la nature génèrent des solutions proches de l'optimum (sinon exactement de l'optimum global) en un temps fini raisonnable, contrairement aux algorithmes traditionnels qui sont insolubles pour les problèmes NP-difficile. Même s'ils restent bloqués dans les optima locaux, le caractère aléatoire intégré leur permet de sortir de l'optimum local. Ils peuvent résoudre des problèmes linéaires et non linéaires qu'ils soient unimodaux ou multimodaux. Les heuristiques et les métaheuristiques intègrent une certaine forme approximative et aléatoire dans les algorithmes, ont une mémoire pour stocker l'historique qui pourrait être la meilleure solution atteinte jusqu'à présent, et elles apprennent également des succès passés. L'inconvénient est le grand nombre d'itérations nécessaires et le manque de cohérence dans les solutions atteintes à chaque itération. Le compromis entre les algorithmes traditionnels et métaheuristiques est que les algorithmes traditionnels ont de bonnes propriétés d'exploitation alors que les algorithmes stochastiques ont de bonnes capacités d'exploration.

II.3.3 Algorithmes bio-inspirés

Le calcul bio-inspiré est une branche de l'intelligence computationnelle, et les différents algorithmes de cette catégorie sont basés sur les caractéristiques des systèmes biologiques, l'informatique évolutive et l'intelligence en essaim [37]. L'informatique bio inspirée a de nombreuses applications dans tous les domaines de l'ingénierie, en particulier l'économie, l'informatique, la conception mécanique et de nombreux autres domaines d'application réels. Ils sont plus adaptés aux problèmes complexes en calculs et gourmands en données et qui se révèlent difficiles à résoudre à l'aide des algorithmes traditionnels.

Les algorithmes bio-inspirés sont efficaces pour arriver à la solution optimale à un problème lorsqu'il existe une myriade de possibilités. Ils sont non déterministes et sont utilisés dans l'analyse de systèmes. Leur simplicité et leur parallélisme inhérent sont deux raisons principales de leur

popularité et de leur large éventail d'applications [38]. Ils sont flexibles et peuvent être adaptés aux changements de l'environnement.

Les algorithmes bio-inspirés pourraient être basés sur une trajectoire ou sur une population. Dans les algorithmes basés sur des trajectoires tel que le recuit simulé, la recherche de la solution optimale commence à partir d'un seul point initial et atteint progressivement l'optimum. Dans les algorithmes basés sur la population tel que l'algorithme génétique où l'optimisation des essaims de particules à la recherche de l'optimum a lieu en parallèle avec une population de particules ou d'agents dans l'espace de recherche. La recherche est un compromis entre une recherche globale à grande échelle (diversification) et une recherche locale intense (intensification). Un bon équilibre entre ces deux éléments est essentiel pour trouver la solution optimale globale en un minimum de temps [39].

Les problèmes de conception technique complexes ont souvent des contraintes non linéaires telles que des limites sur certains paramètres, des relations entre deux ou trois paramètres conduisant à une représentation mathématique d'une contrainte, et le fait que le paysage de la fonction objectif peut être unimodal ou multimodal. Les problèmes d'optimisation multi-objectifs ont souvent des objectifs et des contraintes contradictoires. Il n'y a pas de meilleure solution, mais plusieurs solutions non dominées se trouvant sur le front optimal de Pareto.

La plupart des algorithmes d'optimisation inspirés de la nature sont heuristiques, c'est-à-dire qu'ils trouvent la meilleure approximation qui pourrait ne pas être la solution exacte au problème. Mais ces algorithmes produisent la solution approchée en temps fini, par quelques hypothèses simplificatrices qui ne sont pas toujours vraies pour les algorithmes déterministes. Les algorithmes métaheuristiques fonctionnent à un niveau plus élevé que les algorithmes heuristiques et présentent un compromis entre la recherche exhaustive directe et le caractère aléatoire. La plupart des algorithmes métaheuristiques intègrent un paramètre aléatoire avec une distribution de probabilité connue, pour accélérer la recherche de la solution optimale. Il y a toujours des essais et des erreurs impliqués dans la recherche [40].

II.3.3.1 Intelligence en essaim

Les algorithmes d'intelligence en essaim intègrent les stratégies de recherche de nourriture d'organismes biologiques tels que les animaux, les oiseaux et les insectes [41]. L'agent ayant la

meilleure stratégie de recherche de nourriture survit dans l'environnement puisque la nourriture est essentielle à la survie. Cela implique un processus de recherche, et les agents avec la stratégie de recherche la plus efficace réussissent rapidement par rapport aux autres dans l'environnement concurrentiel. Il y a plusieurs facteurs impliqués dans ces activités de recherche de nourriture, les caractéristiques et la taille de l'agent, son intelligence, son comportement social, l'emplacement et la quantité de nourriture, et l'effort requis pour trouver la nourriture. De plus, étant donné que l'environnement est dynamique, la qualité, la quantité et l'emplacement des aliments ne cessent de changer avec le temps en raison de la consommation et d'autres changements qui se produisent sur une période de temps. Cela nécessite que l'organisme s'adapte et soit polyvalent aux conditions changeantes. Le retour d'informations sur les succès passés du troupeau et le partage d'informations entre les membres du troupeau concernant la qualité et l'emplacement de la nourriture facilitent la recherche de nourriture [42]. Plusieurs espèces présentent une recherche de nourriture sociale qui augmente leurs chances de réussir à trouver de la nourriture et, par conséquent, conduit finalement à de meilleures chances de survie.

Le comportement en essaim des espèces biologiques sur lequel les algorithmes SI sont construits utilise des règles simples, et il ne semble pas y avoir de contrôle centralisé. Lorsqu'il n'y a pas de contrôle centralisé, le comportement individuel fait preuve d'auto-organisation et de contrôle. La figure II.2 montre un troupeau de Stariques montrant un comportement en essaim sans contrôle centralisé évident.

Figure II-2. Troupeau de Stariques présentant un comportement en essaim sans contrôle centralisé évident.

La recherche d'une vaste zone en parallèle avec de nombreux agents et le partage des informations collectées lors de la recherche est la clé du succès pour trouver la solution optimale de manière efficace et efficiente. La recherche est lancée de manière aléatoire, les agents se voyant attribuer des positions initiales de manière aléatoire. La recherche se déroule également de manière aléatoire et, au fil du temps, les informations sur les résultats sont partagées par les autres membres pour soit procéder dans la même direction, soit changer de zone ou de direction de recherche. Toute la zone où il y a une possibilité de trouver une solution doit être recherchée, et pour cela une bonne exploration avec diversification est nécessaire. De telles stratégies de recherche simples peuvent obtenir des résultats de manière efficace ; c'est la raison de leur popularité, et de nombreuses recherches ont été consacrées au développement de tels algorithmes métaheuristiques inspirés de la nature. Les multiples agents de ces algorithmes interagissent les uns avec les autres. C'est le principe sous-jacent fondamental de tous ces algorithmes basés sur la population.

Mais de nombreuses espèces tels que les éléphants et les loups gris ont une hiérarchie dans le troupeau ou la meute, et elles adhèrent aux règles fixées par le chef du groupe. L'auto-organisation nécessite de la mémoire pour se souvenir du passé (succès ou autre).

Un point important concernant les algorithmes inspirés de la nature est qu'il n'y a aucune garantie que la solution trouvée sera l'optimum global. Il peut s'agir d'un optimum local ou proche de l'optimum global, voire de la solution exacte. Le paysage de la fonction objectif est utilisé pour améliorer les solutions candidates existantes afin que de meilleures solutions évoluent au fur et à mesure que les itérations progressent. Les nouvelles solutions sont évaluées sur le paysage fonctionnel, et si une nouvelle solution est meilleure que celles existantes, elle est incluse dans la population. Lorsque de nouvelles solutions sont incluses, les plus faibles ou les moins adaptées sont écartées afin que la taille de la population reste constante. Certaines informations sur le paysage des fonctions de fitness peuvent aider à guider la recherche afin que la convergence se produise plus rapidement. La diversité de la population doit être maintenue afin d'apporter des solutions de qualité et éviter que l'algorithme ne se bloque dans les optima locaux.

Les opérateurs stochastiques employés dans les algorithmes métaheuristiques sont responsables de la diversité de la recherche et de la convergence plus rapide dans les espaces de recherche de grande dimension [43]. Les algorithmes déterministes recherchent également le même espace, et même si le point initial est le même, l'algorithme stochastique est plus rapide et ne converge pas aux optima locaux. Les algorithmes métaheuristiques sont plus susceptibles de trouver l'optimum global que les algorithmes déterministes classiques. La principale différence et l'avantage des algorithmes métaheuristiques par rapport à leurs homologues classiques est le caractère aléatoire de la recherche et de l'inclusion de la diversité avec une population d'agents de recherche. Le point de départ des algorithmes métaheuristiques est une population de solutions initialement générées aléatoirement. Ces solutions sont améliorées à chaque itération jusqu'à ce qu'un critère d'arrêt soit satisfait ou que le nombre maximum d'itérations soit atteint ou que la solution optimale globale soit atteinte. Les algorithmes inspirés de la nature sont simples à mettre en œuvre et capables de résoudre efficacement les problèmes NP-difficile non déterministes.

II.3.3.2 Les métaheuristiques

L'heuristique est une stratégie utilisée lorsqu'il n'est pas possible d'obtenir une solution exacte en résolvant un problème en temps fini. L'application d'heuristiques donne une solution approximative satisfaisante au problème dans un délai pratiquement raisonnable, mais ce n'est peut-être pas la solution précise. Les heuristiques sont utiles pour résoudre des problèmes qui nécessitent

des approximations. Les métaheuristiques sont des heuristiques de niveau supérieur [44] utilisées pour résoudre des problèmes d'optimisation, en particulier ceux qui ont des données incomplètes telles que celles de l'intelligence artificielle et de l'apprentissage automatique. Dans certains des problèmes, lorsque l'ensemble de solutions est trop grand pour être complètement testé, des métaheuristiques peuvent être appliquées. Les métaheuristiques peuvent être implémentées dans l'optimisation stochastique, et dans l'optimisation combinatoire, elles recherchent un grand ensemble discret de solutions réalisables. Elles nécessitent moins de calculs par rapport aux méthodes régulières. La métaheuristique est une stratégie générale appliquée à la mise en œuvre d'un large éventail d'algorithmes d'optimisation [45]. Les algorithmes métaheuristiques fonctionnent toujours sur n'importe quel problème et trouvent une solution même si ce n'est peut-être pas la meilleure ou la solution exacte au problème.

La plupart des algorithmes métaheuristiques sont des algorithmes de recherche, mais il est impossible de rechercher toutes les solutions candidates possibles dans l'espace de recherche, donc certaines heuristiques sont nécessaires. L'heuristique étant impliquée dans la recherche, il n'y a aucune garantie que la solution sera la meilleure ou l'optimum global. La solution pourrait être la meilleure pour le problème (globalement optimale), ou elle pourrait être proche de l'optimum (bonne approximation). L'une des caractéristiques importantes de tels algorithmes est l'amélioration des solutions existantes à chaque itération. Les nouvelles solutions sont meilleures que celles existantes, et les plus pauvres sont rejetées.

II.3.4 Diversification et intensification

La principale caractéristique des algorithmes métaheuristiques est que la recherche se déroule en deux phases : l'exploration et l'exploitation. La capacité d'exploration est la recherche de la solution optimale dans une vaste zone de recherche jusqu'alors inexplorée afin d'atteindre l'optimum global. L'exploration conduit à la diversité des solutions et à la capacité de sortir des optima locaux, le cas échéant. La phase d'exploitation sert à intensifier la recherche dans une zone plus petite où il est possible de trouver la solution optimale. Un bon algorithme doit correctement équilibrer ces deux phases de manière optimale pour des performances efficaces et efficientes [46]. Les algorithmes inspirés de la nature sont basés sur la survie du plus apte et l'adaptation à l'environnement. Cela conduit à deux concepts cruciaux : la diversification et l'intensification. La diversification est la capacité de rechercher efficacement des zones inexplorées dans l'ensemble de

l'espace de recherche, tandis que l'intensification exploite les régions locales en recherchant autour d'une meilleure solution actuelle. Certaines techniques pourraient utiliser des gradients pour une recherche locale aussi intense. Le bon équilibre entre exploration (diversification) et exploitation (intensification) est la clé du succès des algorithmes d'optimisation métaheuristiques inspirés de la nature [47].

Certaines espèces d'insectes et d'oiseaux présentent un comportement de vol de Levy qui consiste en des trajectoires de vol rectilignes ponctuées de virages serrés à 90°. Cette trajectoire de vol Levy peut être utile pour l'exploration globale de l'espace de recherche et donc dans la diversification de la recherche. Les algorithmes de recherche utilisent principalement des tailles de pas variables ou des vols de prélèvement pour équilibrer la diversification et l'intensification. Ce juste équilibre pourrait être atteint en choisissant des valeurs appropriées pour les paramètres associés au problème. Plus d'exploration conduit à une convergence plus lente mais augmente la possibilité de trouver l'optimum global tandis que plus d'exploitation conduit à une convergence plus rapide mais la possibilité de trouver l'optimum global est réduite et la probabilité que l'algorithme soit piégé dans l'optimum local est augmentée. Cet équilibre est l'un des facteurs distinctifs les plus importants parmi les algorithmes métaheuristiques et se reflète en termes de performances pour diverses applications.

Les solutions ont tendance à se déplacer dans l'espace de recherche. Ceci est possible par le mouvement des particules dont l'évaluation en tout point du paysage de la fonction objectif est la valeur de fitness ou la solution à ce point. Ces agents se déplacent à chaque itération vers des régions où ils auront une valeur de fitness plus élevée. Ainsi, la valeur de fitness moyenne de la population augmente. Ceci est similaire aux lucioles qui sont attirées par d'autres lucioles qui ont une luminosité plus élevée qu'elles-mêmes. Le mouvement brownien et la diffusion de tout liquide telles que l'encre, la peinture ou l'aquarelle sur un morceau de tissu sont équivalents au mouvement aléatoire des particules en exploration. La diffusion est similaire à une série d'étapes finies tel que le vol de Levy. Le mouvement de diffusion pourrait suivre une distribution gaussienne ou uniforme dans la plupart des cas. D'autres distributions de probabilité sont également possibles dans les marches aléatoires, mais ces deux distributions sont couramment utilisées. En exploitation où la recherche locale est intensifiée, le caractère aléatoire pourrait être réduit et l'algorithme tend vers des déplacements déterministes. Le concept d'attraction a été étudié par le comportement des

lucioles et la façon dont elles se rassemblent à un endroit en raison de l'attraction de leurs lumières clignotantes. Cela intensifie la recherche autour de la région où les lucioles se sont rassemblées et conduit à une convergence plus rapide de l'algorithme. L'exploration recherche de nouvelles solutions dans l'espace de recherche qui pourraient éventuellement être meilleures que celles existantes.

II.3.5 Théorème “No Free Lunch”

L'un des concepts les plus importants dans les algorithmes inspirés de la nature basés sur les métaheuristiques est le théorème No Free Lunch. Le théorème No Free Lunch [48,49] a été proposé par David Wolpert et William Macready en 1997 et déclarent en effet que tous les algorithmes d'optimisation sont comparables en termes de performances lorsqu'ils sont appliqués à un large éventail de problèmes sur l'ensemble du spectre de l'optimisation technique. Selon le théorème, il n'y a pas d'algorithme d'optimisation meilleur que tout autre algorithme ; chacun est le mieux adapté à une classe particulière de problèmes d'optimisation. Les meilleures performances d'un algorithme d'optimisation sur une classe de problèmes sont compensées par les performances sur une autre classe de problèmes. Un algorithme d'optimisation qui donne un résultat optimal pour un problème peut ne pas produire une solution optimale lorsqu'il est appliqué à un autre problème, alors qu'un autre algorithme d'optimisation peut produire le résultat optimal pour le deuxième problème. Pour le dire autrement, chaque algorithme est le mieux adapté à une classe particulière de problèmes, mais il peut ne pas fonctionner aussi bien pour tous les types de problèmes.

II.3.6 Réglage et contrôle des paramètres

La performance des algorithmes d'optimisation inspirés de la nature dépend du réglage des paramètres associés au problème. Le choix initial des valeurs appropriées pour les paramètres et leur maintien tout au long de l'exécution de l'algorithme constituent le réglage des paramètres. Au fur et à mesure que le nombre d'itérations augmente, les paramètres peuvent être maintenus constants ou ils peuvent être modifiés de manière adaptative. Si les valeurs des paramètres sont modifiées au fur et à mesure des itérations, il s'agit du contrôle des paramètres. Dans certaines applications, la variation des paramètres au fur et à mesure que l'algorithme progresse conduit à une convergence plus rapide et à l'atteinte de l'optimum global. C'est ce qu'on appelle le réglage et le contrôle des paramètres [50].

Dans la plupart des algorithmes de la littérature, les paramètres ont été ajustés sur la base d'observations et de résultats expérimentaux. Les algorithmes ont été exécutés sur plusieurs problèmes de test différents et sur des fonctions de références standard et expérimentalement les paramètres ont été ajustés. L'équilibre entre l'intensification et la diversification, une convergence plus rapide, un temps de calcul réduit, le dépassement des optima locaux et l'approche de l'optimum global peuvent être affinés grâce au réglage et au contrôle des paramètres. Les valeurs initiales des paramètres pourraient être choisies au hasard, et au fur et à mesure que les itérations progressent, selon le paysage de l'espace de recherche, les paramètres pourraient être modifiés dynamiquement. Le retour d'information sur les performances de l'algorithme, probablement en termes de valeurs de fonction de fitness, pourrait être utilisé pour le contrôle des paramètres. La taille de la population est l'un des paramètres importants pour les performances des algorithmes en termes de complexité de la recherche, d'interaction, de partage social d'informations et de taux de convergence. La fonction de fitness pourrait également être modifiée dynamiquement dans certains algorithmes. Un exemple typique est l'optimisation multi-objectifs, où la fonction objectif est formulée comme une combinaison pondérée de plusieurs objectifs, dans laquelle les poids pourraient être modifiés au fur et à mesure que les itérations progressent. Dans certains des problèmes d'optimisation sous contraintes, les équations des contraintes ont généralement des constantes ou des poids ou des valeurs limites. Ces poids pourraient également être éventuellement modifiés dynamiquement.

II.3.7 Les algorithmes inspirés de la nature

Soit S représentant l'ensemble des solutions possibles pour $f(X)$. Nous commençons par un sous-ensemble initial de solutions de l'ensemble et cherchons dans leur voisinage de manière itérative jusqu'à ce que la solution réalisable soit trouvée. C'est la recherche locale. Si la solution réalisable ou optimale n'est pas trouvée au voisinage des solutions initiales choisies, la recherche s'étend sur l'espace. C'est la recherche globale. Les itérations se poursuivent jusqu'à ce qu'un critère d'arrêt soit atteint ou que le nombre maximum d'itérations soit atteint. A chaque itération, la recherche se poursuit vers un objectif, par exemple un coût minimisé. Si le coût est inférieur à celui de l'itération précédente, les nouvelles solutions remplacent les anciennes ; sinon, il est rejeté et la recherche continue. La recherche étant initialement dans le voisinage local du sous-ensemble de solutions choisi, il doit y avoir des moyens pour faire sortir l'algorithme des optima locaux et diversifier la recherche. La solution finale atteinte pourrait être l'optimum global ou l'optimum

local, selon l'efficacité de l'algorithme. Ci-dessous le pseudo code des algorithmes inspirés de la nature.

Pseudo-code d'algorithmes inspirés de la nature

Initialization
Population size N
Objective Function $f(X)$
Constraints $g(X)$ and $h(X)$
Randomly position the members of the population in the search space
Define stopping criteria, if any
Maximum number of iterations $MaxIter$
$iter = 1$
while ($iter \leq MaxIter$) **do**
Execute the algorithm on the population N
Evaluate the fitness of the population
Choose the fittest N members and discard the weaker ones
if stopping criteria met, exit,
otherwise continue
$\boldsymbol{iter = iter + 1}$
end while
Highest fitness value is the global optimum solution

II.4 Problème d'optimisation mono et multi-objectif

Dans les applications d'optimisation du monde réel, il est souvent nécessaire d'optimiser simultanément un ou plusieurs objectifs dans un même problème. Ces problèmes sont généralement appelés problèmes d'optimisation à objectif unique (SOP) ou problèmes d'optimisation multi-objectif (MO). Les deux types sont présentés comme suit:

II.4.1 Problème d'optimisation à objectif unique

Le modèle général d'un problème d'optimisation mono-objectif (SOP) est défini comme la minimisation (ou la maximisation) d'une fonction objectif *f*. Le SOP peut être défini comme suit [51,52] :

$$\min_{x\in\Omega} f(x) \qquad \text{II.12}$$

Sous-contraintes :

$$\begin{aligned} g_j(x) &\le 0, \text{ pou } j=1,\ldots, l \\ h_k(x) &\le 0, \text{ pour } k=1,\ldots, p \\ x_{li} \le x_i &\le x_{ui} \text{ pou } i=1,\ldots,n_x \end{aligned} \qquad \text{II.13}$$

Où :n_x représente la dimension de l'espace de décision, chaque variable de décision x_i, x_{li}, x_{ui} est délimitée respectivement par des limites inférieure et supérieure, $g_j(\mathrm{x})$ sont les l contraintes d'inégalité et $h_k(x)$ sont les p contraintes d'égalité.

- **Définition :** Optimisation minimale globale pour un problème mono-objectif :

La fonction f admet un minimum global si et seulement si :

$$\forall x \in \Omega: -\infty < f(x^*) \le f(x) \qquad \text{II.14}$$

$\boldsymbol{x}^*$ est par définition la solution minimale globale, *f* est la fonction objectif.

La figure II.3 illustre le cas d'un problème d'optimisation à objectif unique pour l'optimum global et l'optimum local.

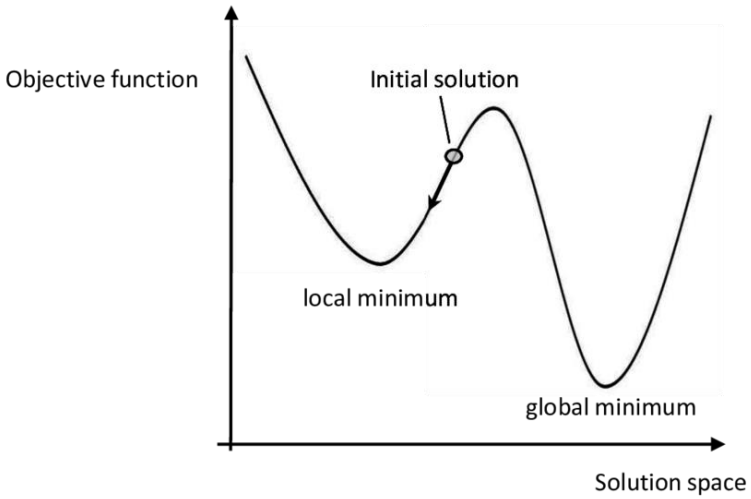

Figure II-3 Optimum global et optimum local.

II.4.2 Problème d'optimisation multi-objectifs

Dans la réalité, la plupart des problèmes d'optimisation implique deux ou plusieurs objectifs de conception, souvent contradictoires. Mathématiquement parlant, un problème d'optimisation multi-objectifs peut être formulé comme suit [52,53,54] :

$$\min_{x\in\Omega} F(x) = \left(f_1(x),..., f_m(x) \right)^T \qquad \text{II.15}$$

Sous-contraintes :

$$\begin{aligned} g_j(x) &\le 0, \text{ pour } j=1,..., l \\ h_k(x) &\le 0, \text{ pour } k=1,..., p \\ x_{li} \le x_i &\le x_{ui} \text{ pour } i = 1,..., n_x \end{aligned} \qquad \text{II.16}$$

Où : F est le vecteur de fonctions objectifs à optimiser, m est le nombre de fonctions objectifs, n_x est la dimension de l'espace de décision, chaque variable de décision x_i est délimitée par des limites inférieure et supérieure $x_{li} \le x_i \le x_{ui}$ pour $i = 1, \ldots, n_x$. $g_j(x)$ sont les l contraintes d'inégalité et $h_k(x)$ sont les p contraintes d'égalité.

La figure II.4 illustre le cas d'un problème d'optimisation multi-objectifs impliquant deux fonctions objectifs.

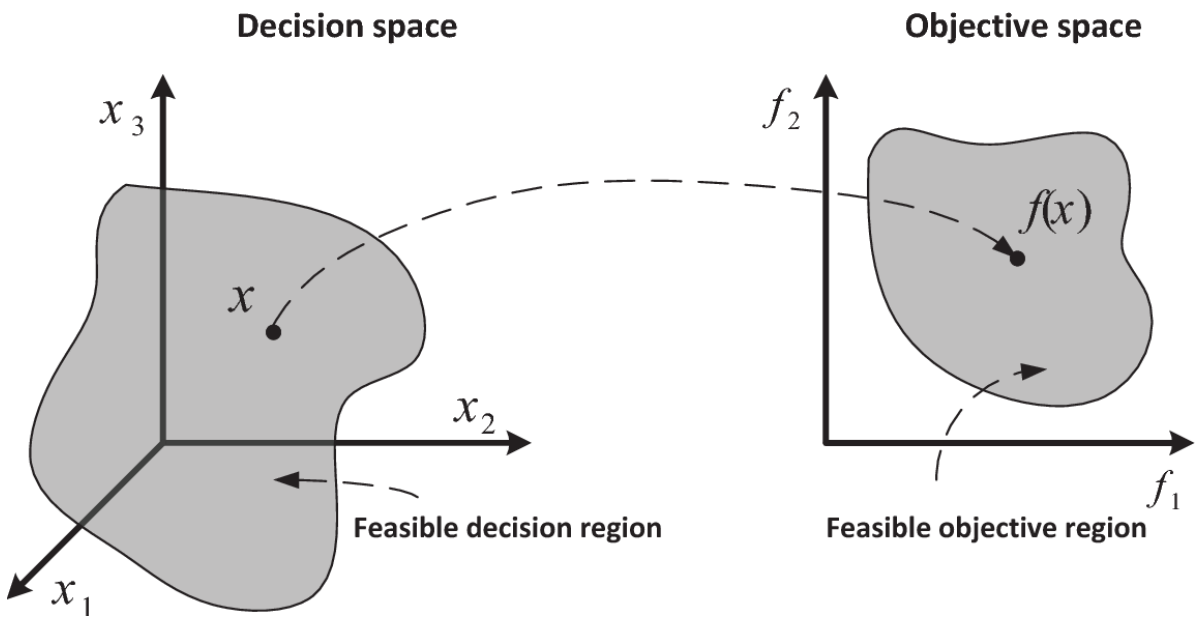

Figure II-4 Espace de décision et d'objectif pour un problème d'optimisation multi-objectifs.

La solution de tels problèmes est très difficile par rapport à l'optimisation à objectif unique. En effet, pour les problèmes d'optimisation multi-objectifs (MO), il n'y a pas une solution optimale mais un ensemble de solutions de compromis [55].

II.4.3 Dominance et optimalité

L'objectif principal de l'optimisation multi-objectifs est de trouver une solution optimale pour le problème décrit en II.3. Pour résoudre ce problème, la plupart des algorithmes d'optimisation multi-objectifs modernes utilisent le concept de dominance de Pareto. Le principe de dominance est de comparer deux solutions afin de déterminer si l'une domine l'autre ou non. La dominance de Pareto est définie comme suit:

- **Pareto optimal** : un vecteur de solution $x^* \in X$ est une solution optimale si et seulement si :

$$\neg\exists\, x \in X: f_i\,(x) \leq f_i(x^*) \wedge f(x) \neq f(x^*);\;\; \forall i \in \{1, 2, \dots, N\}, \qquad \text{II.17}$$

Ces solutions sont aussi appelées ''true Pareto solutions''.

- ***Pareto dominance*** : une solution x^1 domine x^2 notée $x^1 \succ x^2$ si :

$$f_i\ (x^1) \leq f_i(x^2) \wedge \exists j: f_j(x^1) \leq f_j(x^2);\ \ i,j \in\ \ \{1,2,\dots,N\}, \qquad \text{II.18}$$

S'il n'y a pas de solutions qui dominent x^1 alors x^1 est non dominée.

- **Pareto set** : un ensemble de solutions non-dominées est un ‘’Pareto set’’.

$$\{x* \mid \neg\ \exists\ x\ : x \succ x*\} \qquad \text{II.19}$$

- **Pareto front** : c’est l'ensemble des vecteurs dans l'espace objectif qui sont l’image d'un ensemble de Pareto

$$\{F(x^*) \mid \neg\ \exists\ x : x \succ x*\} \qquad \text{II.20}$$

II.4.4 Méthodes de gestion des contraintes

L'espace de recherche d'un problème d’optimisation sous contraintes peut être formulé comme suit :

$$\zeta = \begin{cases} g_j(x) \leq 0, \text{ for } j=1,\dots, l \\ h_k(x) \leq 0, \text{ for } k=1,\dots, p \\ x_{li} \leq x_i \leq x_{ui} \text{ for } i=1,\dots,n_x \end{cases} \qquad \text{II.21}$$

Où : $g_j(x)$ sont les l contraintes d'inégalité est $h_k(x)$ sont les p contraintes d'égalité.

Le fait que les problèmes d'optimisation de fiabilité contiennent des fonctions de contraintes non linéaires, une méthode de gestion a besoin d’être mise en œuvre afin de maintenir la recherche de solution dans l'espace des possibles [56, 57]. Différentes méthodes de gestion des contraintes peuvent être trouvées dans la littérature telles que :

- **Penalty method**

Lorsqu'une solution sort de l'espace de recherche, une pénalité est ajoutée à la valeur des deux fonctions de fitness, de sorte que les solutions qui conduisent à la violation de la contrainte non linéaire sont éliminées et appliquent la sélection des solutions réalisables [58]. Cette dernière méthode est définie comme suit :

$$fitness\ value = -R_s(r_1, r_2, \ldots, r_m, n_1, n_2, \ldots, n_m) + \psi(r_1, r_2, \ldots, r_m, n_1, n_2, \ldots, n_m) \quad \text{II.22}$$

Où : $\psi(r_1, r_2, \ldots, r_m, n_1, n_2, \ldots, n_m)$ est la fonction de pénalité, calculée comme suit :

$$\psi(r_1, r_2, \ldots, r_m, n_1, n_2, \ldots, n_m) = \sum_{j=1}^{M} \phi_j \,.\, max(\,0, g_j(r_1, r_2, \ldots, r_m, n_1, n_2, \ldots, n_m))^2 \quad \text{II.23}$$

Où : ϕ_j correspond au facteur de pénalité ou paramètre de pénalité, une constante positive.

- **Méthode de dominance contrainte :**

Cette méthode proposée dans [59], pour résoudre l'optimisation multi-objectifs sous contraintes, est basée sur le concept de domination sous contraintes, également appelée supériorité de la solution réalisable. En présence de contraintes, chaque solution peut être soit réalisable, soit irréalisable. Une solution x_a est dite domine une solution x_b si l'une des conditions suivantes est vraie :

- x_a est réalisable et x_b est irréalisable.
- x_a et x_b sont irréalisables et x_a a une valeur de violation de contrainte plus petite.
- x_a et x_b sont réalisables et x_a domine x_b avec le principe de dominance habituel.

II.4.5 Prise de décision

Dans la plupart des problèmes d'optimisation multi-objectif, l'ensemble de Pareto optimal se compose d'un nombre important voire infini de solutions. Néanmoins, en pratique, une seule solution doit être sélectionnée dans l'ensemble de Pareto optimal. La personne qui a pour tâche de choisir la solution la plus pratique dans l'ensemble de Pareto optimal est appelée le décideur (DM).

Les préférences du DM sont incorporées pour induire un ordre total entre les éléments de l'ensemble de Pareto. Il existe plusieurs approches dans lesquelles les préférences du décideur peuvent être incorporées. Par exemple, le DM peut classer l'ensemble des objectifs selon leur importance. Une autre possibilité est d'obtenir un échantillon du front de Pareto, puis de sélectionner une solution à partir de cet échantillon. Une classification courante des techniques de résolution de problèmes MO est basée sur le moment où le DM est tenu de fournir ses informations de préférence.

Pour cette raison, différentes techniques d'aide au marquage de décision ont été développées pour introduire des préférences du DM dans le processus d'optimisation [60] cette classification est la suivante :

- **Méthodes a priori :** utilisées pour leur simplicité de mise en œuvre et leur grande généricité. En effet, les objectifs du problème d'optimisation sont transformés en une seule fonction objectif. Le processus d'optimisation mono-objectif est alors lancé pour déterminer la solution optimale.
- **Méthodes a posteriori :** Dans les méthodes a posteriori, on cherche à trouver les solutions de Pareto optimales sans que le DM n'intervienne au cours du processus de résolution. Par conséquent, la méthode doit fournir au DM la totalité du front de Pareto trouvé, puis le DM choisit les solutions les plus adaptées pour lui. Cependant, le grand nombre de solutions obtenues peut rendre l'ensemble de Pareto optimal difficile à analyser pour le DM.
- **Méthodes interactives** : Les méthodes interactives visent à impliquer le DM tout au long du processus de recherche. Par conséquent, tout au long du processus de résolution, le DM interagit avec la méthode de résolution.

Les algorithmes d'optimisation multi-objectifs, contrairement aux algorithmes d'optimisation unique, visent à assurer la convergence et à maintenir la diversité au sein de l'ensemble Pareto-optimal simultanément. Des métriques sont nécessaires pour évaluer adéquatement de telles performances. Dans la section suivante, nous présentons quelques métriques.

II.4.5.1 Generational distance

La *Generational Distance* (*GD*) proposée dans [61], est une mesure de la distance entre les solutions du front Pareto-optimal $\hat{F}$ est le vrai Pareto front F^*. La formulation mathématique de *GD* est la suivante :

$$GD(F^*, \hat{F}) = \frac{1}{|\hat{F}|}\sqrt{\sum_{p\in\hat{F}} d(p, F^*)^2} \qquad \text{II.24}$$

Où : $d(x, F^*)$ est la distance euclidienne minimale entre p et des points de F^*.

II.4.5.2 Inverted generational distance

Inverted generational distance (*IGD*), selon [62], c'est une variante de la *Generational Distance*. Elle mesure les distances entre chaque solution composant le vrai front de Pareto optimal F^* et l'approximation calculée $\hat{F}$. *IGD* est calculée comme suit :

$$IGD(F^*, \hat{F}) = \frac{1}{|\hat{F}|}\sqrt{\sum_{p^*\in F^*} d(p^*, F)^2} \qquad \text{II.25}$$

Où : $d(p^*, \hat{F})$ est la distance euclidienne minimale entre F^* et les points dans $\hat{F}$.

II.4.5.3 Spacing metric

Le *Spacing metric* (*S*), proposé par [63] mesure l'uniformité de la distribution des solutions obtenues dans l'espace objectif. La formulation mathématique de *S* est la suivante :

$$S(\hat{F}) = \frac{1}{|\hat{F}|}\sqrt{\sum_{i=1}^{|\hat{F}|}(\bar{d} - d_i)^2} \qquad \text{II.26}$$

Où : $\bar{d}$ est la valeur moyenne de tous les d_i:

$$di = \min_{j=1}^{|\hat{F}|}\left[\sum_{k=1}^{m}|f_k(x_i) - f_k(x_j)|\right] \qquad \text{II.27}$$

Pour $i \neq j$ est $i = 1, \ldots, \left|\hat{F}\right|$

II.4.5.4 Hypervolume Metric

Le *Hyper-volume* (*Hv*), est une mesure la performance [62, 64], est calculé dans l'espace objectif multidimensionnel délimité par l'ensemble approximatif $\hat{X}$. Mathématiquement, pour chaque solution $\hat{x} \notin \hat{X}$, un hyper-cube v_i est construit avec un point de référence r_p, et la solution $\hat{x}$ comme les coins diagonaux de ce l'hyper-cube. Par la suite, une union de tous les hyper-cubes est trouvée et sa *Hv* est calculée :

$$Hv = volume\left(\bigcup_{i=1}^{n} \hat{X} v_i\right) \qquad \text{II.28}$$

II.5 Conclusion

Dans ce chapitre, une introduction à la théorie générale de l'optimisation et à sa formulation mathématique a été décrite. Un aperçu des algorithmes d'optimisation classiques et traditionnels a été donné. La suite du chapitre traite des méthodes d'optimisation classiques et des algorithmes d'optimisation inspirés de la nature, de leurs caractéristiques, ainsi que de leurs avantages et inconvénients. Les différents algorithmes d'intelligence en essaim inspirés de la nature ont été exposés.

Ensuite, nous avons présentés les bases des problèmes d'optimisation à objectif unique et multi-objectifs. Nous avons ainsi présenté les principales différences entre ces deux problèmes d'optimisation du point de vue de l'unicité ou de la diversité des solutions possibles. Le concept de dominance de Pareto utilisé pour l'optimisation multi-objectifs a été également relaté. L'étape de prise de décision où les approches d'intégration des préférences des décideurs dans le processus de résolution ont été discutées.

Chapitre III

Optimisation mono-objectif : Trois méthodes métaheuristiques pour optimiser la fiabilité des systèmes (PSO, SFS, CS)

Chapitre III

Optimisation mono-objectif : Trois méthodes métaheuristiques pour optimiser la fiabilité des systèmes (PSO, SFS, CS)

III.1 Introduction

De nos jours, la haute performance dans le domaine industriel est devenue plus importante que jamais. Le niveau compétitif peut être atteint en améliorant la fiabilité globale du système. Le concepteur peut pratiquer trois méthodes principales pour atteindre l'optimum : l'utilisation de composants redondants (allocation de redondance), l'augmentation de la fiabilité des composants (allocation de fiabilité) ou les deux (allocation de fiabilité-redondance) [65].

Ces méthodes sont appliquées pour maximiser la fiabilité sans violer les contraintes de conception considérées tels que le coût, le poids et le volume. L'objectif est de déterminer le nombre optimal de composants redondants, la fiabilité de chacun ou les deux [66, 67].

Les techniques d'optimisation métaheuristiques ont été utilisées comme alternative aux approches mathématiques classiques pour obtenir des solutions optimales globales ou quasi globales. En raison de leur grande capacité à détecter des régions prometteuses dans l'espace de recherche et à les explorer à un moment précis. Il a été prouvé au cours des deux dernières décennies que les algorithmes inspirés de la nature sont attrayants, car ils n'appliquent pas d'hypothèse mathématique aux problèmes d'optimisation et ont de meilleures capacités de recherche globale que les algorithmes d'optimisation conventionnels [68].

L'objectif de ce chapitre est de présenter, implémenter et comparer trois algorithmes métaheuristiques d'optimisation de la la fiabilité globale d'une usine pharmaceutique en appliquant trois algorithmes puissants, Particle Swarm Optimization (PSO), Stochastic Fractal Search (SFS), et Cuckoo Search algorithm (CS).

Le présent chapitre est organisé comme suit : la section 2 présente les différents types de problèmes d'optimisation de fiabilité, la section 3 donne quelques définitions, les principes de base et les pseudo-codes des algorithmes appliqués, la section 4 présente l'étude de cas numérique et les résultats avec une discussion sont présentés dans la section 5. Enfin, des conclusions avec des suggestions pour d'autres travaux sont données dans la dernière section.

III.2 Formulations mathématiques du problème (Mono-objectif)

III.2.1 Problème d'allocation de fiabilité

La formulation mathématique générale du problème d'allocation de fiabilité est donnée comme suit [69,70] :

$$Maximize\ R_S(r) = \prod_{i=1}^{m}\left[1-\prod_{j=1}^{li}(1-r_{ij})^{n_{ij}}\right] \qquad \text{III.1}$$

Sous-contraintes:

$$\sum_{i=1}^{n}\sum_{j=1}^{l_i} c_{ij} n_{ij} \le C_0 \qquad \text{III.2}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{l_i} w_{ij} n_{ij} \le W_0$$

$$\sum_{j=1}^{l_i} n_{ij} \ge 1;\ i = 1,2,...,m$$

$$0 \le r_{ij} \le 1;\ r_{ij} \in [0,1] \subset R^{+}$$

$$1 \le i \le m,\ 1 \le j \le l_i$$

Dans le problème d'allocation de fiabilité, la structure du système est fixe et les variables de décision sont les valeurs de fiabilité des composants, où $R_s = (.)$ est la fonction objectif du problème (fiabilité globale du système), m est le nombre de sous-systèmes dans le système, r_{ij} est la fiabilité j au niveau i, c_{ij} et w_{ij} sont respectivement le coût et le poids de chaque alternative de conception j au niveau i, C_0 et W_0 sont les bornes supérieures du coût et du poids des ressources.

III.2.2 Problème d'allocation de redondance

La formulation mathématique générale du problème d'allocation de redondance est donnée comme suit [69, 71] :

$$Maximize\ R_S(n) = \prod_{i=1}^{m}\left[1 - \prod_{j=1}^{li}(1 - r_{ij})^{n_{ij}}\right] \quad \text{III.3}$$

Sous-contraintes :

$$\sum_{i=1}^{n}\sum_{j=1}^{l_i} c_{ij} n_{ij} \le C_0 \quad \text{III.4}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{l_i} w_{ij} n_{ij} \le W_0$$

$$\sum_{j=1}^{l_i} n_{ij} \ge 1;\ i = 1, 2, ..., m$$

$$n_{ij} \ge 0;\ n_{ij} \in \mathrm{Z}^{+}$$

$$1 \le i \le m,\ 1 \le j \le l_i$$

Dans le problème d'allocation de redondance, les variables de décision sont les valeurs de redondance des composants, où $R_s = (.)$ est la fonction objectif du problème (fiabilité globale du système). m est le nombre de sous-systèmes dans le système, n_{ij} est le nombre de composants redondants dans le sous-système j au niveau i. c_{ij} et w_{ij} sont respectivement le coût et le poids de chaque alternative de conception j au niveau i, C_0 et W_0 sont les bornes supérieures du coût et du poids des ressources.

III.2.3 Problème d'allocation de fiabilité-redondance

La formulation mathématique générale du problème d'allocation de fiabilité-redondance est donnée comme suit [69, 72, 73] :

$$Maximize\ R_S(r,n) = \prod_{i=1}^{m}\left[1-\prod_{j=1}^{li}(1-r_{ij})^{n_{ij}}\right] \quad \text{III.5}$$

Sous-contraintes :

$$\sum_{i=1}^{n}\sum_{j=1}^{l_i} c_{ij}n_{ij} \le C_0 \quad \text{III.6}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{l_i} w_{ij}n_{ij} \le W_0$$

$$\sum_{j=1}^{l_i} n_{ij} \ge 1;\ i = 1,2,...,m$$

$$0 \le r_{ij} \le 1;\ r_{ij} \in [0,1] \subset R^+;\ n_{ij} \ge 0;\ n_{ij} \in Z^+$$

$$1 \le i \le m,\ 1 \le j \le l_i$$

Les valeurs de fiabilité des composants et leur redondance sont considérées comme des variables de décision dans le problème d'allocation de fiabilité-redondance, où $R_s = (.)$ est la fonction objectif du problème (fiabilité globale du système). m est le nombre de sous-systèmes dans le système, r_{ij} est la fiabilité j au niveau i, n_{ij} est le nombre de composants redondants dans le sous-système j au niveau i. c_{ij} et w_{ij} sont respectivement le coût et le poids de chaque alternative de conception j au niveau i, C_0 et W_0 sont les bornes supérieures du coût et du poids des ressources.

III.3 Techniques d'optimisation métaheuristiques

Les approches de solutions retenues dans le cadre de cette thèse sont au nombre de trois décrites ci-dessous.

III.3.1 Optimisation par essaims particulaires -PSO

III.3.1.1 Introduction

James Kennedy et Russell C. Eberhart ont développé l'optimisation par essaims particulaires (PSO) à l'Université Purdue (USA) en 1995. La PSO est enracinée dans le comportement des volées d'oiseaux et combinée aux principes de l'évolution [74]. La figure III.1 montre une volée d'oiseaux présentant un comportement collectif.

Figure III-1 Volée d'oiseaux présentant un comportement collectif.

PSO est le premier algorithme d'intelligence d'essaim basé sur l'activité de vols d'oiseaux. Plusieurs scientifiques ont tenté de modéliser l'activité de vols chez les oiseaux en fonction de leur comportement consistant à rester (voler) ensemble, à se disperser, à changer de direction sans entrer en collision, et tout cela se produit en même temps. Puisqu'il y a plusieurs agents ou particules qui se recherchent dans l'espace en parallèle, l'algorithme a un parallélisme inhérent qui le rend efficace.

PSO est un algorithme d'optimisation proposé pour les problèmes d'optimisation linéaires et non linéaires, avec ou sans contraintes. Ce dernier ne nécessite pas le calcul de dérivées et convient aux problèmes d'optimisation combinatoire tant continus que discrets. L'algorithme PSO

peut être appliqué à des espaces de recherche continus, discrets ou mixtes contenant des optima simples et multiples, ce qui le rend adapté aux fonctions unimodales et multimodales.

PSO est une technique basée sur la population où des essaims de particules se déplacent dans l'espace de recherche. Un objectif ou une fonction de fitness doit être défini de manière appropriée pour le problème dont la valeur est obtenue en évaluant à différentes positions dans l'espace de recherche. Chaque particule dans l'espace de recherche représente une solution potentielle au problème. C'est un algorithme itératif ; par conséquent, un nombre maximum d'itérations ou un critère d'arrêt doit être défini, selon le cas. L'algorithme est relativement rapide et ne nécessite pas de mémoire importante.

Dans PSO, la population de particules reste la même à chaque itération tout au long de l'exécution de l'algorithme. La population exacte porte la première à la dernière itération ; par conséquent, il n'y a pas de concept de survie du plus apte, contrairement à l'algorithme génétique (AG). Avec un mélange de membres plus âgés et de nouveaux descendants, la population change à chaque génération créée tandis que la taille de la population reste constante. PSO est un algorithme lié à l'AG utilisant une population de particules et basé sur des principes évolutifs puisque les solutions évoluent de manière itérative.

PSO est simple à mettre en œuvre et efficace pour différents types de fonctions objectif sur une large gamme [75]. L'essaim présente un dynamisme de groupe et un comportement de flocage. Il imite également le comportement social humain, dans lequel les individus interagissent les uns avec les autres et se mettent à jour en fonction des interactions sociales [76].

L'intelligence de l'essaim collectif est similaire à celle des humains, ils apprennent tous les deux de leur propre expérience et de l'expérience des autres. La PSO combine la recherche locale et globale, qui peut être modélisée comme une intensification pour l'exploitation et une diversification pour l'exploration.

Le caractère unique de PSO réside dans la modélisation de l'algorithme en tant que vol de particules à travers l'espace de recherche [77], qui est un hyperespace de dimension *d*. Les particules atteignent la meilleure position globale en acquérant des vitesses et en accélérant vers de meilleures positions à chaque itération.

Dans l'espace de recherche, PSO a plusieurs solutions candidates représentées par des particules ou des oiseaux volant à travers le même espace de recherche. Ce qui permet de rechercher des régions connues aussi bien qu'inconnues dans l'espace ; les valeurs de fitness de chaque candidat sont calculées à chaque itération en fonction de leurs positions dans l'espace de recherche.

La fonction objectif prend les vecteurs de position des particules comme entrées pour évaluer les solutions. Les positions initiales des particules (sur la base desquelles la fitness est calculée) sont choisies au hasard et la taille de la population est ajustée en fonction de la complexité du problème considéré.

La position de l'oiseau ou de la particule dans l'espace de recherche détermine sa valeur de fitness ; la position et la vitesse actuelles déterminent la position suivante et la nouvelle vitesse de cette même particule. Si la vitesse est trop élevée, la particule peut dépasser la solution optimale, et si elle est trop faible, elle peut converger vers un optimum local ; par conséquent, il devrait s'agir d'un compromis entre être trop élevé et trop bas. La particule ayant la meilleure position atteinte jusqu'à présent (meilleure valeur de fitness) est appelée record personnel (P) ; la mémoire est nécessaire pour stocker les meilleures informations de position pour chaque itération. Chaque particule essaie de se déplacer vers le meilleur global (G), qui est la meilleure position parmi toutes les particules de l'essaim. L'algorithme est itératif ; ainsi, chaque position de particule sera mise à jour à chaque itération jusqu'à ce que le critère d'arrêt soit atteint ou que le nombre maximum d'itérations soit atteint.

III.3.1.2 Comportement de l'essaim

Le comportement en essaim fait référence au comportement collectif manifesté par les animaux, les oiseaux ou les insectes lorsqu'ils s'impliquent dans une activité collective [78]. Les entités de l'essaim se déplacent ensemble, se nourrissent ou migrent vers une direction spécifique de manière bien organisée. Ce comportement d'essaimage a été utilisé pour développer de multiples algorithmes d'optimisation inspirés de la nature dont la PSO est le développement pionnier basé sur le vol d'oiseaux. Une volée de flamants roses volant ensemble est illustrée à la figure III.2

Figure III-2 Une volée de flamants roses volant ensemble.

Les oiseaux se comportent collectivement de manière auto-organisée et le troupeau est décentralisé. Ils volent à l'unisson, mais il y a une composante aléatoire, qui est plus apte à modéliser leur comportement en tant que troupeau. Ce caractère aléatoire le rend réaliste. Les membres de la population interagissent entre eux et avec l'environnement, ce qui conduit à un comportement global qui n'est pas écrit en règle générale .

La dynamique de groupe sous-jacente des oiseaux du troupeau est basée sur trois règles :

- Tous les oiseaux font face à la même direction,
- Les oiseaux restent proches les uns des autres,
- Les oiseaux du troupeau ne se heurtent à aucun autre.

Ces règles ont été encadrées par Reynolds dans son article de 1987 [79] comme des règles simples du modèle de flocage :

- Évitement des collisions - les membres du troupeau ne se heurtent pas les uns aux autres.
- Correspondance de vitesse - tous les oiseaux volent à la même vitesse.
- Centrage du troupeau - les membres du troupeau essaye de se déplacer vers le centre du troupeau.

Les membres du troupeau bénéficient de l'expérience des autres et de leur propre expérience dans la recherche de nourriture. Il y a à la fois des avantages et des inconvénients de la recherche de nourriture collective. Les avantages l'emportent sur les inconvénients lorsque la nourriture est disponible de manière dispersée. Le partage social de l'information entre les co-espèces est un avantage évolutif, et c'est le principe fondamental sous-jacent de l'optimisation des essaims de particules. Le comportement social humain est similaire mais pas le même que celui des oiseaux et des animaux. Les oiseaux et les animaux restent ensemble pour éviter les prédateurs, trouver de la nourriture et chercher des partenaires. Les humains ont une composante cognitive et ils n'agissent pas à l'unisson, mais leurs attitudes et leurs croyances sont en accord (conformité) avec celles de leurs pairs. Le changement du comportement social humain est équivalent au mouvement du comportement des oiseaux.

Les éléments de l'essaim ont leur propre manifestation personnelle (comportement cognitif) et la manifestation de l'essaim (comportement social). Les membres de l'essaim se comportent en fonction de leur propre passé et de celui de l'ensemble de l'essaim. Les oiseaux se déplacent ensemble dans la même direction. La position et la vitesse changent en fonction de leur comportement passé et du comportement social collectif de l'essaim. Ils se déplacent dans l'espace de recherche, qui est aussi l'espace de solution. Chaque particule/oiseau a une position et une vitesse qui varient avec le temps. Cependant, lorsque les oiseaux volent ensemble en troupeau, leurs vitesses doivent être ajustées pour être identiques ou proches de celles de leurs voisins. Le mouvement doit être synchrone et dans le même sens avec des positions différentes au sein du même troupeau. Cela peut être modélisé avec un certain caractère aléatoire introduit dans le mouvement des oiseaux. L'essaim doit avoir une mémoire pour qu'il se souvienne de sa meilleure position précédente.

Les principales caractéristiques ou principes sur lesquels le comportement des essaims est modélisé sont [80] : proximité, qualité, diversité, stabilité et adaptabilité.

Le paradigme sous-jacent de PSO est constitué de calculs effectués dans un espace de recherche de dimensions d sur une séquence d'intervalles de temps, ce qui est le premier principe de l'intelligence en essaim. Les membres de la population répondent aux meilleurs facteurs de qualités personnelles et globales ; ceci est conforme au deuxième principe. La diversité spécifiée dans la troisième caractéristique se produit en raison de la répartition de la réponse entre le meilleur

personnel et le meilleur global. La population ne change d'état que lorsque le meilleur global change ; cela le rend stable. La population change lorsque les meilleurs changements la rendent adaptative, ce qui est conforme aux quatrième et cinquième principes énoncés ci-dessus.

III.3.1.3 L'algorithme d'optimisation par essaims particulaires

L'algorithme commence par un énoncé du problème et les contraintes associées si le problème considéré est contraint. La taille de la population *N* et la fonction objectif sont définies en fonction des critères à optimiser. La population de particules est distribuée uniformément dans tout l'espace de recherche, avec leurs positions et vitesses choisies au hasard dans les limites définies. L'espace de recherche est supposé être de dimension *d*, de sorte que chaque position et vitesse de particule est représentée par un vecteur de dimension *d*. La vitesse des particules peut être initialisée à zéro ou à une autre valeur à l'intérieur des limites définies. Les valeurs de fitness pour l'essaim de particules à leurs positions actuelles sont calculées, et il s'agit de leur record personnel puisqu'il s'agit de la valeur de fitness initiale de l'essaim. La valeur de fitness maximale parmi l'ensemble de l'essaim est la meilleure globale. Les particules sont accélérées vers la meilleure position globale de l'essaim puisque leurs positions sont les meilleures personnelles actuelles. Les nouvelles valeurs de fitness des particules sont calculées en fonction de leurs nouvelles positions. Le record personnel et le record global de l'essaim sont mis à jour parmi toutes les positions atteintes par les particules jusqu'à présent. Au cours de l'exploration spatiale de recherche, les particules trouvent des solutions qui peuvent correspondre à des optima locaux ou globaux. Ce processus est répété de manière itérative jusqu'à ce que la solution optimale globale soit atteinte, que le nombre maximum d'itérations soit atteint ou que le critère d'arrêt soit satisfait [81].

- **Algorithme**

$X_i^{iter} = \left[x_{i1}^{iter}, x_{i2}^{iter}, ..., x_{id}^{iter} \right]$ est la particule *i* dans l'espace de recherche de dimension *d* en itération indexé par la variable *iter*, et *N* est la taille de la population, où $i = 1,2,\ldots,N$. $P_i^{iter} = \left[p_{i1}^{iter}, p_{i2}^{iter}, ..., p_{id}^{iter} \right]$ est la meilleure *position personnelle* de la particule *i*, $G_i^{iter} = \left[g_{i1}^{iter}, g_{i2}^{iter}, ..., g_{id}^{iter} \right]$ est la *meilleure position globale* de l'essaim, et $f\left(X_i^{iter}\right)$ est la fonction objectif ou la fonction fitness évaluée pour la particule *i* en itération *iter*. Chaque particule a une valeur de fitness basée sur sa position dans l'espace de recherche obtenue en évaluant la fonction

de fitness. La valeur de fitness la plus élevée atteinte par la particule jusqu'à présent dans une certaine position de l'espace de recherche est la *meilleure position personnelle* de la particule. La valeur de fitness la plus élevée atteinte jusqu'à présent dans une certaine position de l'espace de recherche parmi toutes les particules de l'essaim est la *meilleure position globale* de l'essaim. Les équations III.7 et III.8 donnent la mise à jour de la vitesse et de la position pour les particules.

$$V_i^{iter+1} = w_v V_i^{iter} + c_1 R_1 \left(P_i^{iter} - X_i^{iter}\right) + c_2 R_2 \left(G_i^{iter} - X_i^{iter}\right) \quad \text{III.7}$$

$$X_i^{iter+1} = X_i^{iter} + V_i^{iter+1} \quad \text{III.8}$$

Où : $V_i^{iter} = \left[v_{i1}^{iter}, v_{i2}^{iter}, \ldots, v_{id}^{iter}\right]$, $R_1 = [r_{11}, r_{12}, \ldots, r_{1d}]$, $R_2 = [r_{21}, r_{22}, \ldots, r_{2d}]$.

Étant donné que l'espace de recherche est de dimension *d*, la position et la vitesse de chaque particule sont des vecteurs de dimension *d* aussi, et ils doivent être mis à jour pour chaque itération. Les équations de mise à jour de la vitesse et de la position pour la $i_{ème}$ particule sont données par les équation III.9 et III.10 respectivement.

$$v_{i,j}^{iter+1} = w_v v_{i,j}^{iter} + c_1 r_{1j} \left(p_{i,j}^{iter} - x_{i,j}^{iter}\right) + c_2 r_{2j} \left(g_j^{iter} - x_{i,j}^{iter}\right) \quad \text{III.9}$$

$$x_{i,j}^{iter+1} = x_{i,j}^{iter} + v_{i,j}^{iter+1} \quad \text{III.10}$$

$i = 1,2,\ldots,N; j = 1,2,\ldots,d$

Les vecteurs dans les équations ci-dessus sont de dimension *d* puisque l'espace de recherche est supposé être un hyperespace de dimension *d*. Le nombre maximum d'itérations pour l'algorithme est représenté par *MaxIter*, un critère d'arrêt pour l'algorithme pourrait également être défini, comme un seuil ε. La variable w_v est le coefficient d'inertie, et une valeur inférieure pour w_v accélère le mouvement des particules alors qu'une valeur plus élevée pour w_v amortit le mouvement. Le coefficient d'inertie est responsable du mouvement de la particule dans n'importe quelle direction dans l'espace de recherche. Si les particules se déplacent plus rapidement, cela entraînera une convergence plus rapide de l'algorithme, et si les particules se déplacent lentement, cela entraînera une convergence plus lente et une plus grande exploration de l'espace de recherche.

Le deuxième terme de l'équation de mise à jour de la vitesse $c_1 R_1\left(P_i^{iter} - X_i^{iter}\right)$ est la composante cognitive qui amène la *ième* particule à se déplacer vers les meilleures positions trouvées par elle-même jusqu'à présent. Cela conduit intrinsèquement à l'inclusion de la mémoire dans la particule afin qu'elle puisse revenir à ses meilleures positions trouvées dans le passé. Les constantes c_1 et c_2 influencent la taille de pas maximale que la particule peut prendre dans la direction de la *meilleure position personnelle* et la *meilleure position globale* dans chaque itération, on les appelle donc constantes ou coefficients d'accélération. Le coefficient c_1 est le coefficient cognitif, et sa valeur détermine la taille du pas de la particule prise vers sa meilleure *position personnelle* position. Le troisième terme de l'équation de mise à jour de la vitesse $c_2 R_2\left(P_i^{iter} - X_i^{iter}\right)$ est la composante sociale et détermine le mouvement de la *ième* particule vers les meilleures positions trouvées par l'essaim jusqu'à présent. Le coefficient c_2 est le coefficient social, et sa valeur détermine la taille de pas que la particule prend vers la *meilleure position globale* trouvée par l'essaim jusqu'à présent. Les coefficients R_1 et R_2 sont des nombres aléatoires qui introduisent une composante stochastique dans le mouvement des particules de l'essaim. Cela fait apparaître que les particules se déplacent de manière pseudo-aléatoire vers la *meilleure position personnelle* et *meilleure position globale*. Si seul le terme cognitif est inclus, les performances seront moins bonnes car il n'y a pas d'interaction entre les particules. Si seul le terme social est inclus, la performance sera soit supérieure, soit inférieure à la performance avec les termes cognitifs et sociaux inclus, selon le problème auquel il est appliqué. Les paramètres w_v , c_1 et c_2 pourraient être dans l'intervalle $0.8 \leq w_v \leq 1.2, 0 \leq c_1 \leq 2, 0 \leq c_2 \leq 2$ qui s'est avéré satisfaisant pour la plupart des applications. Les valeurs réelles sont choisies en fonction du problème à résoudre. Les valeurs de r_{1j} et $r_{2j} (j = 1,2,\ldots,d)$ sont choisies au hasard dans l'intervalle $\left(0 \leq r_{ij} \leq 1 \text{ et } 0 \leq r_{2j} \leq 1\right)$, et elles sont régénérées chaque fois que la vitesse est mise à jour. Cela introduit la composante stochastique dans l'algorithme où la composante aléatoire est introduite dans la trajectoire de la particule alors qu'elle vole vers la *meilleure position personnelle* et la *meilleure position globale*.

Soit l'espace de recherche délimité par$[-Xmax; +Xmax]$. Les particules doivent se déplacer dans l'espace de recherche à l'intérieur de cette plage et ne pas dépasser cette dernière. Ainsi une technique appelée *speed clamping* est proposée pour limiter la vitesse maximale de chaque particule. Les limites de la vitesse des particules sont $[-Vmax; +Vmax]$, ou *Vmax = k·Xmax*, *k*

est le *velocity-clamping* facteur prenant des valeurs dans la plage $0.1 \leq k \leq 1.0$. Pour de tels problèmes, la vitesse maximale est donnée par $Vmax = k\cdot(Xmax - Xmin)/2$. Comme mentionné précédemment, la vitesse des particules est limitée aux limites [$-Vmax$ a $+Vmax$] ou $Vmax = k\cdot Xmax$. Dans l'équation de mise à jour de la vitesse si l'amplitude de la nouvelle vitesse V_i^{iter+1} est inférieure à V_{max} alors cette valeur sera la nouvelle vitesse ; sinon si elle dépassera cette limite et sera bridée à $\pm V$max. Si la vitesse n'est pas limitée dans les limites, les particules s'envoleront hors de l'espace de recherche.

La vitesse et la position des particules sont mises à jour selon l'équation III.9 et l'équation III.10 à chaque itération jusqu'à ce que l'algorithme converge. Ce processus est répété jusqu'à ce que le critère d'arrêt soit atteint ou que le nombre maximum d'itérations soit atteint. A la fin des itérations, le courant *global best* position (fonction de fitness évaluée au *global best* position) est la solution optimale globale au problème. Certains des critères d'arrêt utilisés sont le nombre maximal d'itérations atteint, la valeur de fitness cible de la fonction objectif est atteinte, aucune amélioration n'est observée sur un certain nombre d'itérations, le rayon d'essaim normalisé est proche de zéro, etc. La solution optimale est atteinte lorsque le *global best* est l'optimum global. Lorsque l'optimum global n'est pas atteint dans un nombre prédéfini d'itérations ou que l'essaim diverge, l'algorithme est considéré comme ayant échoué. La divergence de l'essaim est contrôlée par le paramètre *Vmax*. Le paramètre d'inertie w_v doit être sélectionné avec soin et peut être diminué au fur et à mesure que les itérations progressent. Cela fait passer l'algorithme du mode exploration (diversification) au mode exploitation (intensification). Si le poids d'inertie est grand, la recherche est globale, et si le poids d'inertie est petit, la recherche est locale. Les vitesses des particules doivent être fixées à un maximum de *Vmax.* Si la vitesse est trop grande, les particules survoleront la solution optimale ; si la vitesse est trop petite, les particules peuvent rester bloquées dans les optima locaux. L'algorithme PSO est stable, car il change d'état uniquement lorsque les meilleures positions personnelles ou globales changent et adaptatif car il change d'état lorsque le *global best* change.

III.3.1.4 Pseudo-code

Le Pseudo-code de Particle Swarm Optimization :
Initialization
Population (swarm) size N
Define objective function $F(X)$ of dimension d
Initial positions and velocities of the particles
Compute fitness values of the particles
Initial personal best and global best
Parameters: inertia weight wv, coefficients c_1 and c_2
Random parameters R_1 and R_2
Maximum number of iterations MaxIter
Stopping criteria, if any
$iter = 1$
for $iter = 1$ to *MaxIter* do
Update velocity and position of the particles
Calculate fitness values for all the particles
Update personal best and global best
if stopping criteria met then exit, otherwise continue
end for
global best is the optimum solution

III.3.2 Recherche fractale

III.3.2.1 Introduction

Il existe d'innombrables applications d'optimisation dans notre monde. De nos jours, de nombreuses entreprises sont confrontées à des problèmes nécessitant une optimisation. En effet, il existe de nombreux problèmes difficiles dans l'industrie et la science qu'il est vraiment nécessaire de résoudre. Ils peuvent être formulés comme des problèmes d'optimisation.

Nous avons besoin d'optimisation pour minimiser le temps, les coûts ou les risques, et maximiser les bénéfices, la qualité ou l'efficacité. De nombreux problèmes complexes d'optimisation de la vie réelle ont émergé dans de nombreux domaines scientifiques tels que l'ingénierie, l'économie et les affaires [82], qui ne peuvent pas être résolus dans un délai raisonnable et donnent entre-temps une réponse précise. En effet, de tels problèmes sont souvent fortement non linéaires. De plus, beaucoup d'entre eux incluent de nombreuses variables différentes et agissent sous des contraintes complexes. Ces contraintes sont soit sous la forme de bornes simples telles que des plages de propriétés des matériaux, soit sous la forme de relations non linéaires telles que la contrainte maximale, la flèche maximale, la capacité de charge minimale ou la configuration géométrique [83]. D'autre part, étant donné que la taille de l'espace de recherche augmente considérablement tout en résolvant des problèmes d'optimisation de grande dimension, les algorithmes d'optimisation classiques comme la recherche exhaustive ne fournissent pas une solution appropriée. Par conséquent, la principale alternative pour résoudre ce type de problème est d'utiliser des algorithmes approximatifs.

III.3.2.2 Fractales

La propriété d'un objet ou d'une quantité qui explique l'auto-similitude à toutes les échelles, dans un sens quelque peu technique, est appelée fractale. Le terme de "fractal" vient du mot latin fracactus qui signifie "cassé" ou "fracturé", et il a été utilisé pour la première fois par Benoit Mandelbrot en 1975. Mandelbrot a également essayé d'utiliser le concept de théories fractales pour décrire des motifs géométriques dans la nature [84].

Développant la recherche dans ce domaine, la liste d'exemples de fractales comprenant des structures allant des agrégats microscopiques à l'amas de galaxies est devenue très longue. Les phénomènes de croissance loin de l'équilibre sont un domaine important engagées dans de nombreux domaines de la science et de la technologie. Certains exemples de tels processus incluent la solidification dendritique dans un milieu sous-refroidi, le doigté visqueux qui est observé lorsqu'un fluide visqueux est injecté dans un fluide plus visqueux et l'électrodéposition d'ions sur une électrode [85].

En règle générale, afin de générer une forme fractale, certaines méthodes standard telles que : Iterated function systems [86], Strange attractors [87], L-systems [88], Finite subdivision rules [89] et Random fractals [90] peuvent être utilisées.

- **Fractales aléatoires**

Des fractales aléatoires peuvent être générées en modifiant le processus d'itération via des règles stochastiques tels que : Levy flight, Gaussian walks, percolation clusters, self-avoiding walks, fractal landscapes, trajectories of Brownian motion, et the Brownian tree [85].

Certaines fractales aléatoires, tels que les clusters décrivant une colonie bactérienne, peuvent être générées par un modèle à motivation physique appelé « Diffusion Limited Aggregation » (DLA) [91]. Pour simplifier, nous envisageons de former un tel amas sur un plan, avec la particule initiale (graine) située à l'origine. D'autres particules sont alors générées aléatoirement autour du point d'origine et provoquent une diffusion. Un algorithme mathématique, comme la " marche aléatoire ", peut être utilisé pour simuler le processus de diffusion. La particule diffusante adhère à la particule de graine qui en est constituée (Figure III.3).

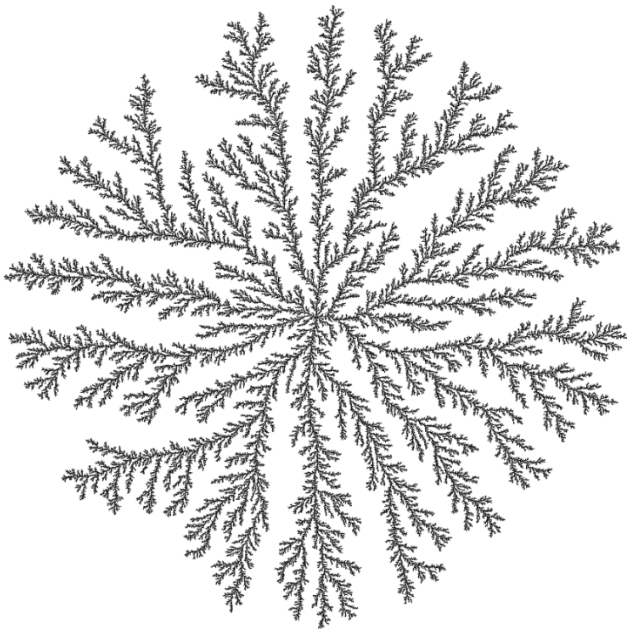

Figure III-3 Croissance fractale en point de flocon de neige.

- **Dielectric breakdown**

Les ramifications de décharge étroites qui sont fréquemment observées dans la nature sont appelées claquages diélectriques. Une étude sur les propriétés de claquage diélectrique montre que la tendance à la ramification peut être modélisée en motifs stochastiques complexes. Les exemples sont l'éclairage, les décharges de surface et l'arborescence dans les polymères. La structure globale

des salves ramifiées montre souvent une similitude structurelle étroite au sein d'une grande classe de types de décharges. Pourtant, à l'heure actuelle, même une classification qualitative de ces structures fait défaut. Niemeyer et al. [92] ont montré que les décharges branchées suivaient des propriétés fractales et ont proposé un nouveau modèle stochastique pour décrire le modèle de décharge d'un claquage diélectrique. Leur modèle est à peu près similaire à DLA.

III.3.2.3 L'algorithme Stochastic Fractal Search (SFS)

Bien que Fractal Search réussisse bien à trouver la solution, cette approche souffre de certains inconvénients. Le principal inconvénient est d'avoir de nombreux paramètres qui doivent être bien réglés, et le deuxième inconvénient principal est que l'échange d'informations ne se produit pas entre les particules. L'échange d'informations entre tous les points participants du groupe est une tentative d'accélérer la convergence au minimum. Une phase appelée processus de mise à jour est ajoutée à l'algorithme de recherche Fractal pour effectuer la recherche de manière indépendante, ce qui est essentiel. D'autre part, étant donné que Fractal Search est un algorithme dynamique car le nombre d'agents dans l'algorithme est modifié, un compromis entre la précision et la consommation de temps est confronté. Par conséquent, pour résoudre les problèmes mentionnés, une autre version de la recherche fractale (FS) appelée recherche fractale stochastique (SFS) est introduite.

Les deux principaux processus qui produisent dans l'algorithme SFS sont le processus de diffusion et le processus de mise à jour. Le premier processus est similaire à la recherche fractale, et chaque particule diffuse autour de sa position actuelle pour satisfaire la propriété d'intensification (exploitation). Ce processus évite d'être piégé dans les minima locaux et augmente les chances de trouver les minima globaux. L'algorithme simule la façon dont un point du groupe met à jour sa position en fonction d'autres points du groupe dans le deuxième processus. Contrairement à la phase de diffusion dans FS, qui provoque une augmentation spectaculaire du nombre de points participants, un processus de diffusion statique pour SFS est considéré. Cela signifie que la particule la mieux générée par le processus de diffusion est la seule particule prise en compte et que le reste des particules est rejeté. En plus d'une exploration efficace de l'espace du problème, SFS utilise des méthodes aléatoires comme processus de mise à jour.

Afin de créer de nouvelles particules à partir de la procédure de diffusion, deux méthodes statistiques appelées vol gaussien et de Lévy sont étudiées. Des études préliminaires sur l'utilisation

séparée des distributions de Lévy et de Gauss montrent que bien que le vol de Lévy converge plus rapidement que la marche gaussienne en quelques générations, la marche gaussienne est plus prometteuse que le vol de Lévy pour trouver les minima globaux. Par conséquent, contrairement à Fractal Search, qui utilise la distribution de vol de Levy, la distribution gaussienne est la seule "marche aléatoire" utilisée dans le processus de croissance DLA de SFS. Généralement, une série de marches gaussiennes participant au processus de diffusion ont été répertoriées dans les équations suivantes :

$$GW_1 = Gaussian(\mu_{BP}, \sigma) + (\varepsilon \times BP \times P_i) \quad \text{III.11}$$

$$GW_2 = Gaussian(\mu_{BP}, \sigma) \quad \text{III.12}$$

Où : ε et ε' sont des nombres aléatoires uniformément distribués limités à [0, 1], *BP* et P_i sont désignés comme la position du meilleur point et du *ième* point dans le groupe, respectivement. Les deux premiers paramètres gaussiens sont μ_{BP} et σ où μ_{BP} est exactement égal à *BP*. Les deux derniers paramètres sont μ_P et σ où μ_P est égal à P_i. En tenant compte des paramètres gaussiens, l'écart type est calculé par l'équation III.13 :

$$\sigma = \left| \frac{log(g)}{g} \times (P_i - BP) \right| \quad \text{III.13}$$

Pour encourager une recherche plus localisée en tant qu'individus, et se rapprocher de la solution, le terme $\frac{log(g)}{g}$ est utilisé afin de diminuer la taille des sauts gaussiens, lorsque le nombre de générations augmente.

Supposons un problème d'optimisation globale avec dimension *d*. Par conséquent, chaque individu noté considéré pour résoudre le problème a été construit sur la base d'un vecteur de dimension *d*. Au cours du processus d'initialisation, chaque point est initialisé de manière aléatoire en fonction des contraintes du problème en prescrivant des limites minimales et maximales. L'équation d'initialisation du point j, P_j est abordé comme suit :

$$P_j = LB + \varepsilon \times (UB - LB) \quad \text{III.14}$$

Où : ε est un nombre aléatoire uniformément distribué qui est limité à [0, 1], *LB* et *UB* sont respectivement les vecteurs contraints inférieur et supérieur du problème, comme indiqué dans les équations précédentes.

Après avoir initialisé tous les points, la fonction de fitness de chaque point est calculée pour atteindre le meilleur point (*BP*) parmi tous les points. Selon la propriété d'exploitation dans la procédure de diffusion, tous les points ont parcouru leur position actuelle pour exploiter l'espace de recherche du problème. D'autre part, deux stratégies statistiques visant à augmenter l'exploration spatiale sont envisagées en raison de la propriété d'exploration. La première procédure statistique s'exécute sur chaque indice vectoriel, et la deuxième méthode statistique est ensuite appliquée à tous les points.

Pour la première procédure statistique d'abord, tous les points sont classés en fonction de la valeur de la fonction de fitness. Chaque point *i* du groupe reçoit alors une valeur de probabilité qui obéit à une distribution uniforme simple comme l'équation suivante :

$$Pa_i = \frac{rank(P_i)}{N} \quad \text{III.15}$$

Où : $rank(P_i)$ est considéré comme le rang du point P_i parmi les autres points du groupe, et *N* est utilisé comme nombre de tous les points du groupe.

En réalité, l'équation III.15 veut affirmer que plus le point est bon, plus la probabilité est élevée. Cette équation est utilisée pour augmenter les chances de changer la position des points qui n'ont pas obtenu une bonne solution. D'autre part, la chance de passer de bonnes solutions dans la prochaine génération augmentera. Pour chaque point P_i en groupe, selon que la condition $Pa_i < \varepsilon$ est satisfaite, et où ε est un nombre aléatoire uniforme appartenant à $[0, 1]$, le composant *j* de P_i , est mis à jour selon l'équation III.16, sinon il reste inchangé.

$$P_i'(j) = P(j)(P_t(j) - P_i(j))_r \quad \text{III.16}$$

Où : $P_i^{'}$ est la nouvelle position modifiée de P_i, $\mathrm{P_r}$ et P_t sont des points choisis au hasard dans le groupe, ε est le nombre aléatoire choisi dans la distribution uniforme dans l'espace continu [0, 1].

En ce qui concerne la première procédure statistique qui est effectuée sur les composantes des points, la deuxième modification statistique vise à changer la position d'un point en considérant la position d'autres points dans le groupe. Cette propriété améliore la qualité de l'exploration et satisfait la propriété de diversification. Avant de commencer la deuxième procédure, encore une fois, tous les points obtenus à partir de la première procédure statistique sont classés en fonction de l'équation III.15, similaire au premier processus statistique, si la condition $Pa_i < \varepsilon$ est tenue pour un nouveau point P_{0i}, la position actuelle de P_{0i} est modifiée selon les équations III.17 et III.18, sinon aucune mise à jour ne se produira. Les équations III.17 et III.18 se présentent comme suit :

$$P_i^{''} = P_i^{'} - \hat{\varepsilon} \times \left(P_t^{'} - BP\right) \left| \varepsilon^{'} \leq 0.5 \right. \qquad \text{III.17}$$

$$P_i^{''} = P_i^{'} + \hat{\varepsilon} \times \left(P_t^{'} - P_r^{'}\right) \left| \varepsilon^{'} > 0.5 \right. \qquad \text{III.18}$$

Où : $P_r^{'}$ et $P_t^{'}$ sont des points choisis au hasard, obtenus à partir de la première procédure. Le nouveau point $P_i^{''}$ est remplacé par $P_i^{'}$ si sa valeur de fonction de fitness est meilleure que $P_i^{'}$.

III.3.2.4 Pseudo-code

Pseudo-code de l'algorithme Stochastic Fractal Search
Initialize a population of N points
While g < maximum generation or (stop criterion) **Do**
For each Point P_i in the system **Do**
Call Diffusion Process with the following process:
q = (maximum considered number of diffusion).
For $j = 1 \; to \; q$ Do
If (user uses Gaussian walk to solve the problem)

Create a new point.

Else (user uses another Gaussian Walks to solve the problem)

Create a new point.

Call Updating Process with the following process

First Updating Process:

First, all points are ranked.

For each Point P_i in the system **Do**

For each component j in P_i **Do**

If rand $[0, 1] \geq P_{ai}$

Update the component j in P_i .

Else

Do nothing.

Second Updating Process:

Once again, all points obtained by the first

Update Process are ranked.

For each new point P_i' in the system Do

If $\left(\text{rand } [0.1] \geq P_i'\right)$

Update the position.

Else

Do nothing.

III.3.3 Algorithme de recherche de Coucou

III.3.3.1 Introduction

Les algorithmes métaheuristiques sont très puissants pour résoudre des problèmes complexes de la vie réelle car ils s'inspirent de phénomènes naturels. Les organismes biologiques dans la nature ont leurs propres processus évolutifs et mécanismes de survie dans des environnements hostiles. Les principales caractéristiques des algorithmes métaheuristiques d'intensification et de diversification sont responsables de la performance efficace dans la résolution de problèmes

d'ingénierie complexes et de problèmes NP-difficile en informatique. L'intensification et la diversification conduisent à de bonnes propriétés d'exploitation et d'exploration de l'algorithme qui sont nécessaires pour une recherche locale intense et une recherche globale diversifiée. L'inculcation du comportement de vol de Levy dans l'algorithme d'optimisation métaheuristique améliore les performances d'exploitation et d'exploration.

Le Cuckoo search (CS) est un algorithme d'optimisation qui a été développé sur la base du comportement parasite de la couvée des coucous. C'est un algorithme métaheuristique proposé par Xin-She Yang et Suash Deb en 2009 [93, 94]. Certaines espèces de coucous construisent leurs nids et élèvent leurs petits, tandis que de nombreuses autres espèces sont des parasites de la couvée. Les traits principaux de l'algorithme Cuckoo search sont le parasitisme du couvain et le comportement de vol de Levy. Chaque espèce dans la nature, y compris le coucou, suit instinctivement la théorie de la survie du plus fort de Darwin. Ils s'adaptent aux conditions environnementales changeantes et présentent une intelligence et des propriétés d'essaim comme manifestations de cette adaptation. L'algorithme Cuckoo search a été construit sur le comportement d'essaim et l'intelligence des oiseaux de l'espèce coucou qui sont décrits dans les sections suivantes.

III.3.3.2 Comportement du coucou

Le coucou appartient à la famille des oiseaux appelés Cuculidae. Ils ont des habitats étendus. Certaines espèces de coucous se trouvent dans les régions tropicales, en particulier dans les forêts tropicales, tandis que d'autres sont migratrices. Les coucous sont de taille moyenne, élancés et vivent dans les arbres ou sur le sol. Les oiseaux migrateurs se déplacent pendant l'hiver. La plupart d'entre eux vivent en solitaire et se nourrissent d'insectes et de fruits. La figure III.4 montre un coucou mâle.

Figure III-4 Un oiseau coucou mâle.

Les coucous sont des parasites de la couvée et les Cuculinae sont la sous-famille [95]. Les coucous parasites sont des parents appauvris. Ils utilisent plus de 100 espèces d'autres oiseaux comme hôtes. Les coucous parasites pondent leurs œufs dans le nid d'autres oiseaux. Les différentes espèces de coucous pondent des œufs blancs, gris ou colorés, comme le vert, le rouge et le jaune, et ils peuvent être unis ou tachetés. Ces variétés d'œufs peuvent correspondre aux œufs de n'importe quel oiseau hôte. Habituellement, le coucou attend près du nid d'un oiseau hôte en observant l'occasion d'entrer et de pondre ses œufs. Ils essaient d'imiter les œufs de l'oiseau hôte afin que leurs œufs soient conservés et éclos par l'oiseau hôte. Les coucous veillent à ce que leurs œufs éclosent avant ceux de l'oiseau hôte. Les coucous femelles visitent différents nids hôtes, poussent l'un des œufs de l'oiseau hôte et pondent leurs œufs. Les œufs correspondent presque aux œufs hôtes en termes de couleur et de taille, ce que l'on appelle le mimétisme des œufs. Lorsque les poussins de coucou ont dépassé le nid, ils s'envolent.

Les poussins de coucou ressemblent à ceux de l'oiseau hôte de sorte que leurs petits sont nourris par l'hôte. Par exemple, chez certaines espèces de coucous, l'oiseau hôte peut être un corbeau, et donc le poussin de coucou est noir et ressemble à un jeune corbeau. Certains d'entre eux habitent des roselières où les fauvettes roseaux ont leurs nids, et ils deviennent l'hôte des coucous. Le coucou pond ses œufs dans le nid de la fauvette roseau, et une fois qu'il est éclos, le

poussin de coucou essaie de pousser l'œuf de l'oiseau hôte. Ceci est fait pour s'assurer que le poussin de coucou est nourri par l'oiseau paruline roseau hôte.

III.3.3.3 Levy Flights

Les vols Levy portent le nom du mathématicien français Paul Levy. Certains oiseaux et insectes, en particulier les mouches des fruits, présentent un comportement de vol Levy. Ils empruntent des trajectoires rectilignes ponctuées de virages serrés à 90°. Les processus stochastiques sont régis par des trajectoires aléatoires communément appelées "marche aléatoire" Le vol de prélèvement est une classe particulière de marche aléatoire, longueur du pas à une distribution de probabilité à queue lourde. La figure III.5 montre un exemple de mouvement de vol Levy.

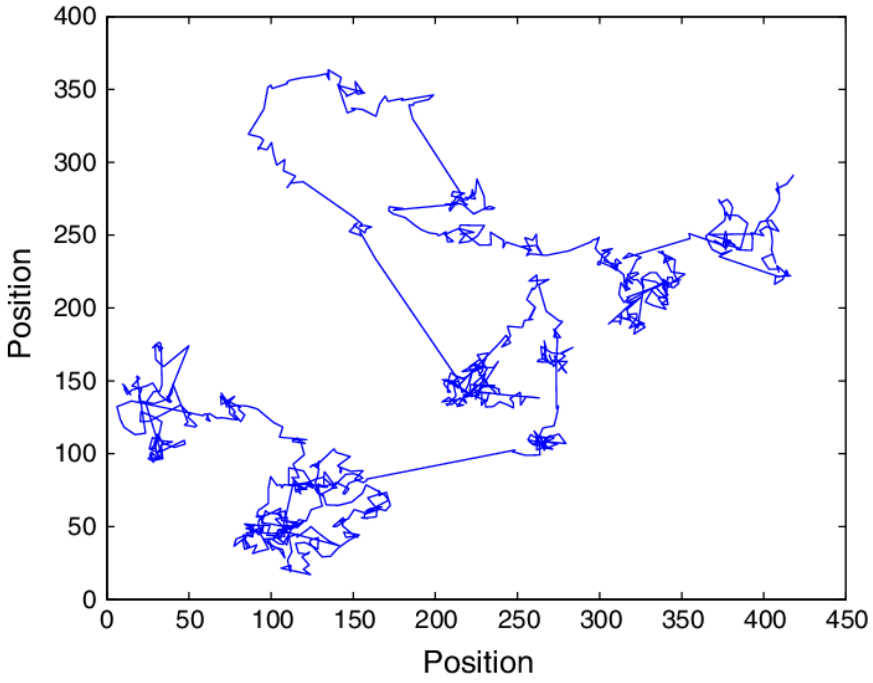

Figure III-5 Mouvement de Levy flight.

La fonction de densité de probabilité (*pdf*) de la distribution de Levy concerne les variables aléatoires continues qui ne peuvent prendre que des valeurs positives, elle est donnée par l'équation III.19 :

$$f(x,\mu,c) = \sqrt{\frac{c}{2\pi}} \frac{e^{\frac{-c}{2(x-\mu)}}}{(x-\mu)^{\frac{3}{2}}} \qquad \text{III.19}$$

Où : μ est un paramètre de position qui décale la courbe et c est un paramètre d'échelle. Comme $x \to \infty$, le pdf est approximativement donné par :

$$f(x,\mu,c) \approx \sqrt{\frac{c}{2\pi}} x^{-\frac{2}{3}} \qquad \text{III.20}$$

Le *pdf* est donné par l'équation III.20. La distribution de Levy a une moyenne et une variance infinies.

Pour les grandes valeurs d'une variable aléatoire *x* (peut être la somme de plusieurs variables aléatoires) qui suivent la distribution de Levy, la fonction de densité de probabilité est approximativement donnée par :

$$f(x) \approx x^{-(1+\mu)}, 0 < \mu < 2 \qquad \text{III.21}$$

Les variables aléatoires à variance infinie présentent une distribution de loi de puissance avec des queues lourdes et grasses. Les vols de prélèvement et le mouvement brownien décrivent le mouvement de plusieurs animaux et oiseaux lors de la recherche de nourriture. Habituellement, les animaux et les oiseaux recherchent de la nourriture en marchant au hasard. Étant donné que les marches aléatoires peuvent être modélisées statistiquement, l'emplacement et la probabilité de transition déterminent la taille du pas et la direction du mouvement. Les vols de prélèvement ont de nombreuses applications qui incluent le comportement à la lumière, le comportement de recherche de nourriture des animaux, la marche humaine, la description des tremblements de terre, etc. Il est montré qu'il est possible de développer des matériaux optiques qui présentent un

comportement de vol de Levy pour la lumière. Il s'agit d'un travail pionnier qui a permis d'étudier le comportement de vol de Levy dans des conditions contrôlées.

III.3.3.4 Cuckoo Search Optimization

Cuckoo search (CS) est un algorithme d'optimisation métaheuristique basé sur le comportement de reproduction parasite obligatoire de certaines espèces de coucous. Il inculque également le mouvement de vol de Levy présenté par certains animaux, oiseaux et insectes, y compris les coucous. L'algorithm Cuckoo search a été développé sur la base de trois hypothèses :

- Le nombre de nids hôtes dans lesquels un coucou peut pondre est fixe, représenté par la taille de la population N. $ph \in [0,1]$ est la probabilité que l'oiseau hôte découvre l'œuf de coucou pondu dans son nid. Une fois découvert, l'oiseau hôte évacue l'œuf ou abandonne le nid et en construit un nouveau. Par conséquent, ph est la fraction de N nids abandonnés ou remplacés par de nouveaux par l'oiseau hôte.
- Un coucou ne pond qu'un seul œuf dans un nid d'hôte choisi au hasard à tout moment.
- Les nids avec des œufs de haute qualité ont des valeurs de fitness plus élevées et survivent donc jusqu'à la prochaine génération.

Chaque œuf dans un nid hôte (en supposant qu'un seul œuf est présent) représente une solution au problème. L'œuf de coucou parasite représente une nouvelle solution qui pourrait remplacer la solution existante si elle s'avère meilleure.

Une fonction objectif est formulée comme une expression mathématique basée sur le problème à résoudre. L'évaluation de la fonction objectif avec un ensemble de variables d'entrée donne une solution possible au problème. L'algorithme place aléatoirement les N nids d'hôtes dans l'espace de recherche. Pour simplifier, on suppose qu'il n'y a qu'un seul œuf hôte dans le nid et que le coucou parasite pond un œuf dans le nid hôte. Ceci est répété pour tous les nids dans l'espace de recherche. Les œufs représentent des solutions possibles au problème, ce qui signifie qu'ils sont les valeurs de fonction de fitness évaluées. Si l'aptitude de l'œuf de coucou s'avère meilleure que l'aptitude de l'œuf hôte, l'œuf de coucou remplace l'œuf hôte dans le nid. Sinon, l'œuf de coucou est jeté hors du nid. Ce processus est répété pour tous les nids dans l'espace de recherche. À chaque itération, les nids sont mis à jour en fonction de leurs valeurs de fitness. A la fin du nombre

maximum d'itérations fixé pour l'algorithme, les nids avec les pires valeurs de fitness sont remplacés par de nouveaux.

- **Algorithme**

Soit X_k^{iter} représentant la position du kth nid des oiseaux hôtes situés au hasard dans l'espace de recherche.

$$\boldsymbol{X_k^{iter} = \left[x_{k1}^{iter}, x_{k1}^{iter}, \ldots, x_{kd}^{iter}\right] k = 1, 2, \ldots, N} \quad \text{III.22}$$

L'espace de recherche est supposé de dimension *d*, indexé par la variable j avec $j = 1, 2, \ldots, d$. donc l'objectif ou la fonction de fitness définie pour le problème est de dimension *d* donné par :

$$\boldsymbol{f(X) = f(x_1, x_2, \ldots, x_d)} \quad \text{III.23}$$

Les itérations sont indexées par la variable *iter*, le nombre maximum d'itérations étant représenté par *MaxIter*. Si nécessaire, un critère d'arrêt pourrait également être défini pour le problème.

Soit X_c^{iter} est la position du cth Cuckoo en itération $\boldsymbol{iter}$ et X_c^{iter+1} est sa nouvelle position dans la prochaine itération. La position du cth Cuckoo dans l'itération suivante est régie par sa position actuelle et le mouvement de vol de Levy comme indiqué ci-dessous :

$$\boldsymbol{X_c^{iter+1} = X_c^{iter} + s \oplus Levy(u)} \quad \text{III.24}$$

Où : *s* est la taille du pas qui est toujours positive et sa valeur dépend du problème ; généralement, il peut être choisi comme 1. $Levy(u)$ est la distribution de Levy qui modélise la probabilité de transition, faisant ainsi dépendre la position suivante de la position actuelle et de la probabilité de transition.

$$\boldsymbol{Levy(u) \approx t^{-u}, 1 \leq u \leq 3} \quad \text{III.25}$$

Cette marche aléatoire utilisant le vol Levy est efficace dans la recherche de l'optimum puisque la taille du pas peut être rallongée ou raccourcie. Les nouvelles solutions sont générées par la marche aléatoire avec un pas important pour une meilleure exploration de l'espace de recherche, et un pas court pour une meilleure exploitation des régions locales dans l'espace de recherche. Lorsque la taille du pas est grande, l'algorithme peut sauter hors de l'optimum local et réduit la possibilité d'être piégé dans des solutions localement optimales. Une séquence de telles marches aléatoires modélisées à l'aide du comportement de vol de Levy devient une chaîne de Markov. Les vols de prélèvement sont destinés à la recherche mondiale, au maintien de la diversité de la population et aux "promenades aléatoires" pour intensifier la recherche. Les vols Levy ont de longues trajectoires pour la recherche globale entrecoupées de courts mouvements browniens pour une recherche locale intense. Le paramètre de probabilité ph équilibre la recherche entre local et global. Quand $ph = 0.25$, la recherche est 75% globale et 25% locale. Les tailles de pas dynamiques possibles dans les vols Levy rendent la recherche efficace car elle peut s'adapter entre diversification et intensification.

Le cth Cuckoo et le kth nid d'hôte sont choisis au hasard et leur aptitude évaluée. En comparant leurs valeurs de fitness, si la fitness du coucou est supérieure à celle de l'hôte, l'œuf de coucou remplacera l'œuf de l'oiseau hôte dans le nid choisi. Sinon, l'œuf hôte ne sera pas dérangé. La fraction ph des nids avec les pires valeurs de fitness est abandonnée. Les nids sont classés en fonction de leurs valeurs de fitness, et à la fin du nombre maximum d'itérations, le nid avec le fitness le plus élevé est la solution optimale globale.

III.3.3.5 Pseudo-code

Pseudo-code de Cuckoo Search Optimization

Initialization

Create host nests in the search space randomly with population size *N*

Define *d*-dimensional objective function $f(X), X = \{x1, x2, \ldots, xd\}$

Define stopping criteria, if any

Maximum number of iterations *MaxIter*

$iter = 1$

while ($iter \leq MaxIter$) **do**

Randomly choose a cuckoo Xc and evaluate its fitness value Fc

Randomly choose a host nest k and evaluate its fitness Fk

if $(Fc > Fk)$

replace the host egg with the cuckoo egg

end if

Fraction *ph* of host nests with least fitness values are discarded

Nests with higher fitness are retained and new ones built to replace the discarded ones

All the nests (solutions) are ranked according to their fitness values

Nest with highest fitness chosen as the current best optimal solution to the problem

if *stopping criteria met* **exit**

else *continue*

iter = *iter* + 1

end while

Nest with best fitness value is the global optimum solution

III.4 Etude de cas numérique

III.4.1 Description du système

Les métaheuristiques évoquées dans la section précédente sont illustrées à travers une usine pharmaceutique afin d'optimiser sa fiabilité globale. Ci-dessous, une brève description de l'installation est donnée [96].

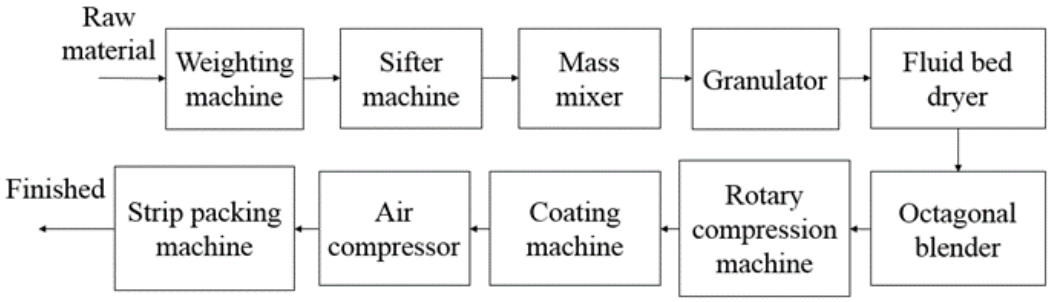

Figure III-6 Usine pharmaceutique.

L'usine pharmaceutique (figure III.6) se compose de dix sous-systèmes connectés en série, une balance, une machine à décalage, un mélangeur de masse, un granulateur, un séchoir à lit fluidisé, un mélangeur octogonal, une machine de compression rotative, une machine d'enrobage, un compresseur d'air et une machine d'emballage en bandes [96]. La matière première est transférée d'un sous-système à un autre chronologiquement jusqu'à la fin de la ligne de production.

Le modèle mathématique non linéaire du système ci-dessus est formulé comme suit [96] :

$$\text{Maximize } \mathrm{R}_s(r,n) = \prod_{i=1}^{10}[1-(1-r_i)^{n_i}] \qquad \text{III.26}$$

Sous-contraintes :

$$g_1(r,n) = \sum_{i=1}^{10} C(r_i)\left(n_i + exp\left(\frac{n_i}{4}\right)\right) \leq C \qquad \text{III.27}$$

$$g_2(r,n) = \sum_{i=1}^{10} v_i\, n_i^2 \leq V$$

$$g_3(r,n) = \sum_{i=1}^{10} w_i\left(n_i \times exp\left(\frac{n_i}{4}\right)\right) \leq W$$

$$0.5 \leq r_i \leq 1-10^{-6}, r_i \in [0,1]$$

$$1 \leq n_i \leq 10, n_i \in Z^+$$

$$0.5 \leq R_s \leq 1-10^{-6}$$

Le tableau III-1 rapporte les données du système.

Table III-1. Données du système

Sous-système i	$10^5\,\alpha_i$	β_i	v_i	w_i	v	c	w	$T(h)$
1	0.611360	1.5	4	9	289	553	483	1000
2	4.032464	1.5	5	7				
3	3.578225	1.5	3	5				
4	3.654303	1.5	2	9				
5	1.163718	1.5	3	9				
6	2.966955	1.5	4	10				
7	2.045865	1.5	1	6				
8	2.649522	1.5	1	5				
9	1.982908	1.5	4	8				
10	3.516724	1.5	4	6				

III.5 Résultats et discussion

Les trois algorithmes implémentés (PSO, CS et SFS) ont été programmés à l'aide de MATLAB et exécutés sur un processeur Intel core i7 avec 6 Go de RAM et (2,20 GHz, Windows 7, 64 bits). Les performances de chaque algorithme sont comparées aux autres pour mettre en évidence la supériorité d'un algorithme sur un autre [97].

Les résultats numériques de l'étude de cas sont présentés dans le tableau 2, il comprend la fiabilité et la redondance de chaque composant, la fiabilité globale, le nombre d'évaluations de fonction et la consommation de ressources.

Table III-2. Résultats pour le système d'usine pharmaceutique

Méthodes	**PSO**		**CS**		**SFS**	
i	r_i	n_i	r_i	n_i	r_i	n_i
1	0.88189	3	0.881380	3	0.8819187	3
2	0.82139	3	0.822142	3	0.8213525	3
3	0.82641	3	0.825369	3	0.8257501	3
4	0.8252	3	0.825310	3	0.8251109	3
5	0.864	3	0.863700	3	0.8639056	3
6	0.83299	3	0.833564	3	0.8330750	3
7	0.84592	3	0.846415	3	0.8460067	3
8	0.83638	3	0.836544	3	0.8370517	3
9	0.84712	3	0.847035	3	0.8472050	3
10	0.8265	3	0.826192	3	0.8266916	3
Slack 1	0.10023274		0.00229444		0.00025481	
Slack 2	13.02599631		13.02599631		13.02599631	
Slack 3	10.00000000		10.00000000		10.00000000	
NFE	4600		16450		**3760**	
R_S	0.95884908315176		0.95886030208529		**0.95886308454060**	

À partir du tableau 2, on peut observer que le SFS a dépassé les CS et PSO, avec une fiabilité optimale du système de 0,958863084540604 et a utilisé 3 760 fonctions d'évaluations ; pendant ce temps, la recherche de coucou et l'optimisation par l'essaim de particules ont atteint respectivement une fiabilité de 0,958860302085295 et 0,958849083151761 avec une utilisation plus élevée des de fonctions d'évaluations 16 450 et 4 600.

Les paramètres spécifiques de chaque algorithme (comme indiqué dans les tableaux 3, 4 et 5) ont été choisis afin que chaque algorithme puisse fonctionner de la manière la plus optimale.

Table III-3. Parameters de PSO

Parameters	valeurs
Taille de la population (taille de l'essaim)	100
Nombre de variables de décision	20
Poids d'inertie	1
Rapport d'amortissement du poids d'inertie	0.99
Coefficient d'apprentissage personnel	1.5
Coefficient d'apprentissage global	2.0

Table III-4. Parameters de SFS

Parameters	valeurs
Taille de la population	40
Nombre de variables de décision	20
Numéro de diffusion maximal	2
diffusion walk	0.25

Table III-5. Parameters de CS

Parameters	valeurs
Taille de la population	25
Nombre de variables de décision	20
Taux de découverte d'oeufs/solutions extraterrestres	0.20

III.6 Conclusion

Les métaheuristiques sont souvent utilisées pour résoudre des problèmes difficiles qui nécessitent d'explorer un espace plus large. L'avantage des algorithmes métaheuristiques est lié à l'exploration efficace de l'espace sans être sensible à la taille de l'espace de recherche. Typiquement, les métaheuristiques sont basées sur trois principaux objectifs : résoudre les problèmes plus rapidement, résoudre les gros problèmes et obtenir des algorithmes robustes. De plus, la facilité de conception et de mise en œuvre ainsi que la flexibilité devraient être les autres caractéristiques de ces algorithmes. Deux caractéristiques importantes des métaheuristiques sont : l'intensification (ou exploitation) et la diversification (ou exploration). La recherche autour des meilleures solutions actuelles et la sélection des meilleurs candidats ou solutions relèvent des propriétés d'intensification, tandis que la diversification étudie l'efficacité de l'algorithme à explorer l'espace de recherche en utilisant souvent la méthode de randomisation.

L'objectif principal de ce chapitre était de présenter, implémenter et comparer trois algorithmes métaheuristiques bien connus (PSO, CS et SFS) afin de résoudre un problème d'allocation de fiabilité-redondance pour une usine pharmaceutique contenant dix sous-systèmes connectés en séries. Une fonction de pénalité a dû être implémentée dans chaque algorithme pour éliminer les solutions infaisables. Les résultats révèlent la supériorité de la recherche fractale stochastique sur la recherche coucou et l'optimisation de l'essaim de particules.

Chapitre IV

Optimisation Multi-Objectif : Allocation de la fiabilité et de la redondance d'un système de protection dans une centrale électrique (NSGA-II, NSGA-III)

Chapitre IV

Optimisation Multi-Objectif : Allocation de la fiabilité et de la redondance d'un système de protection dans une centrale électrique (NSGA-II, NSGA-III)

IV.1 Introduction

Ce chapitre présente l'utilisation de deux algorithmes d'optimisation multi-objectif pour résoudre le problème d'allocation fiabilité-redondance dans le cas d'un système de protection contre la survitesse d'une centrale électrique. Habituellement, ce genre de problème est considéré comme un problème à objectif unique soumis à une ou plusieurs contraintes non linéaires et résolu en utilisant soit des techniques de programmation mathématique, soit plus récemment des métaheuristiques spéciales. Dans le présent chapitre, le problème est considéré comme un problème d'optimisation multi-objectif. Les algorithmes NSGA-II et NSGA-III sont utilisés et démontrent leur capacité à identifier l'ensemble des solutions optimales (front de Pareto) fournissant au décideur l'espace de solutions optimales.

De nos jours, la haute performance dans le domaine industriel est devenue plus cruciale que jamais, comme déjà expliquer dans le chapitre précédent, la conception d'un système hautement fiable est atteinte soit par l'utilisation de composants redondants (allocation de redondance), l'augmentation de la fiabilité des composants (allocation de fiabilité), ou les deux (allocation fiabilité-redondance). L'approche est formulée comme un problème d'optimisation non linéaire [98], [99].

Diverses méthodes métaheuristiques ont été développées afin de résoudre ces problèmes [100], [101]. Ces derniers sont difficiles à résoudre en raison de la quantité considérable d'efforts de calcul requis pour trouver les solutions optimales. D'autre part, la plupart des travaux antérieurs ont résolu le problème en tant que problème d'optimisation unique, c'est-à-dire que l'objectif était de maximiser la fiabilité ou de minimiser le coût, seulement.

L'objectif principal de ce chapitre est de résoudre un problème d'allocation de fiabilité-redondance afin d'obtenir soit le maximum ou le minimum des fonctions objectif considérées sans violer aucune des contraintes de volume, de poids, de coût et en utilisant deux algorithmes d'optimisation multi-objectif, le Non-Sorting Genetic Algorithm II (NSGA-II) et le Non-Sorting Genetic Algorithm III (NSGA-III). L'étude de cas consiste en un système de protection contre la survitesse dans une centrale électrique.

Le reste du chapitre est organisé comme suit : la section 2 présente les différents types de problèmes d'optimisation de fiabilité mixte, une vue d'ensemble du front de Pareto et les méthodes de gestion des contraintes ; la section 3 présente les scénarios envisagés pour la résolution du problème et les pseudo-codes de NSGA-II et NSGA-III implémentés. Les sections 4 et 5 présentent respectivement l'étude de cas numérique et les résultats obtenus avec une discussion. Enfin, la dernière section est dédiée à la conclusion.

IV.2 Approche de résolution multi-objectif

Le concept d'optimalité ne peut pas être appliqué dans des problèmes d'optimisation à objectifs multiples, comme dans un problème à objectif unique (chapitre III), par conséquent on fait appel à d'autres outils plus adaptés à ce genre de problème d'optimisation. Dans la partie suivante on présente le problème d'optimisation d'allocation de la fiabilité et de la redondance sous forme multi-objectif, ensuite les algorithmes adaptés à ce genre de problèmes sont présentés dans la 3eme section du chapitre.

IV.2.1 Description du problème Multi-objectif

Les problèmes d'optimisation de fiabilité multi-objectif existent principalement de trois manières : l'allocation de fiabilité (affecter la fiabilité aux différents composants), l'allocation de redondance (affecter la redondance aux différents composants) et l'affectation de fiabilité-redondance (affecter à la fois la fiabilité et la redondance à chaque composant).

IV.2.1.1 Problème d'allocation de fiabilité

La formulation mathématique générale du problème d'allocation de fiabilité multi-objectif est donnée comme suit :

$$Maximize\ R_s(r) = R_s(r_1, r_2, ..., r_m) \qquad \text{IV.1}$$

$$\text{Minimize } C_s(r) = C_s(r_1,\ r_2, ...,\ r_m)$$

Sous-contraintes :

$$g_j(r_1, r_2, ..., r_m) \le b \qquad \text{IV.2}$$

$$0 \le r_i \le 1;\ i = 1,2,...,m$$

$$r_i \in [0,1] \subset R^+$$

Dans le problème d'allocation de la fiabilité, la structure du système est fixe et les variables de décision sont les valeurs de fiabilité des composants, ou $R_s = (.)$ est la première fonction objectif du problème (fiabilité globale du système), et $C_s = (.)$ est la deuxième fonction objectif qui est le coût global du système, *m* est le nombre de sous-systèmes dans le système, r_i est la fiabilité du sous-système *i*, *b* est le vecteur de la limitation des ressources.

IV.2.1.2 Problème d'allocation de redondance

La formulation mathématique générale du problème d'allocation de redondance multi-objectif est donnée comme suit :

$$Maximize\ R_s(n) = R_s(n_1, n_2, ..., n_m) \qquad \text{IV.3}$$

$$\text{Minimize } C_s(n) = C_s(n_1,\ n_2, ...,\ n_m)$$

Sous-contraintes :

$$g_j(n_1, n_2, \ldots, n_m) \le b$$

$$1 \le n_i \le n_{imax} \qquad \text{IV.4}$$

$$n_i \in Z^+$$

Où : $R_s = (.)$ et $C_s = (.)$ sont les fonctions objectif du problème (fiabilité globale du système, coût global), *m* est le nombre de sous-systèmes dans le système, n_i est le nombre de composants redondants dans le sous-système *i*, *b* est le vecteur de la limitation des ressources.

IV.2.1.3 Problème d'allocation de fiabilité-redondance (problème mixte)

La formulation mathématique générale du problème d'allocation de fiabilité-redondance multi-objectif est donnée comme suit :

$$Maximize\ R_s(r,n) = R_s(r_1, r_2, ..., r_m; n_1, n_2, ..., n_m) \qquad \text{IV.5}$$

$$\text{Minimize } \mathrm{C}_s\left(r,n\right)=C_s\left(r_1,\ r_2,...,\ r_m;n_1,n_2,...,n_m\right)$$

Sous-contraintes :

$$g_j\left(r_1,r_2,...,r_m;n_1,n_2,...,n_m\right)\le b \qquad \text{IV.6}$$

$$1\le n_i\ \le n_{i\max};\ 0\le r_i\ \le 1;i=1,2,...,m$$

$$r_i\in\left[0,1\right]\subset \mathrm{R}^+;n_i\in Z^+$$

Où : $R_S=(.)$ et $C_S=(.)$ sont les fonctions objectif du problème (fiabilité globale du système, coût global), m est le nombre de sous-systèmes dans le système, n_i est le nombre de composants redondants dans le ième sous-système, r_i est la fiabilité du sous-système i, b est le vecteur de la limitation des ressources.

IV.2.2 Front de Pareto

L'optimalité de Pareto est utilisée, ce qui nous donne une classification des solutions selon les définitions suivantes. En termes de minimisation [55] :

Définition 1. *Pareto optimal* : Un vecteur de solution $x^* \in X$ est la solution optimale de Pareto si et seulement si:

$$\neg\exists\, x\in X:\ f_i\ (x)\le f_i(x^*)\wedge f(x)\ne f(x^*);\ \forall i\in\{1,\ 2,\ \ldots,\ N\}, \qquad \text{IV.7}$$

Ces solutions sont aussi appelées *True Pareto solutions*

Définition 2. *Pareto dominance* : Une solution x^1 domine x^2 désignée comme $x^1 \succ x^2$ si et seulement si :

$$f_i\ (x^1)\le f_i(x^2)\wedge\exists j:\ f_j(x^1)\le f_j(x^2);\ i,j\in\{1,\ 2,\ \ldots,\ N\}, \qquad \text{IV.8}$$

S'il n'y a pas de solutions qui dominent x^1 alors x^1 est non-dominée.

Définition 3. *Pareto set* : Un ensemble de solutions non-dominées est dit *Pareto set* :

$$\{x^* \mid \neg \exists\, x \;:\; x \succ x^*\} \qquad \text{IV.9}$$

Définition 4. *Pareto front* : l'ensemble des vecteurs dans l'espace objectif qui sont l'image d'un ensemble de Pareto :

$$\{F(x^*) \mid \neg \exists\, x :\; x \succ x^*\} \qquad \text{IV.10}$$

IV.2.3 Technique de gestion des contraintes

Comme dans le chapitre précèdent et afin de garder les solutions à l'intérieur de l'espace de recherche une méthode de gestion de contraintes a été implémentée puisque le problème considéré est contraint.

Dans le présent problème, la méthode de la fonction de pénalité est utilisée, lorsqu'une solution sort de l'espace de recherche, une pénalité est ajoutée à chaque valeur de la fonction objectif, de sorte que les solutions qui conduisent à la violation de la contrainte non linéaire sont éliminées et ne seront pas portées à l'itération suivante [58]. Cette dernière méthode est définie comme suit :

$$\textit{fitness value} = -R_s(r_1, r_2, \ldots, r_m, n_1, n_2, \ldots, n_m) + \psi(r_1, r_2, \ldots, r_m, n_1, n_2, \ldots, n_m) \qquad \text{IV.11}$$

Où : $\psi(r_1, r_2, \ldots, r_m, n_1, n_2, \ldots, n_m)$ représentent la fonction de pénalité, calculée comme suit :

$$\psi(r_1, r_2, \ldots, r_m, n_1, n_2, \ldots, n_m) = \sum_{j=1}^{M} \phi_j \,.\, max(\,0, g_j(r_1, r_2, \ldots, r_m, n_1, n_2, \ldots, n_m))^2 \qquad \text{IV.12}$$

Où : ϕ_j correspondent au facteur de pénalité ou paramètre de pénalité, une constante positive.

IV.3 Approches de solutions (scénarios)

Dans ce chapitre pour résoudre le problème d'optimisation multi-objectif deux algorithmes sont implémentés afin de trouver les solutions optimales à l'intérieur de l'espace de recherche.

IV.3.1 L'Algorithme Génétique de tri Non-dominé II (NSGA-II)

IV.3.1.1 Introduction

Le NSGA-II est un algorithme d'optimisation multi-objectif bien connu basé sur la population et a été introduit par Deb et al [102]. Le NSGA-II est une version améliorée de l'algorithme original Non-Sorting Genetic Algorithm (NSGA), ce dernier algorithme exigeait que l'utilisateur définisse et implémente un paramètre de partage afin de maintenir une diversité admissible dans l'ensemble de solutions, mais l'utilisation du paramètre de partage prédéterminé a rendu le processus d'optimisation plus complexe, et les résultats largement dépendants de la valeur de ce paramètre.

Dans la deuxième version de l'algorithme (NSGA-II), les deux inconvénients de l'algorithme d'origine ont été éliminés en implémentant le *Crowding Distance procedure* (CD) à la place du *sharing parameter*, la nouvelle approche ne nécessite pas de paramètre prédéfini afin d'éviter un ensemble agrégé de solutions et a également rendu l'algorithme moins complexe en termes de calcul.

IV.3.1.2 Préservation de la diversité

Comme mentionné précédemment, afin de maintenir une bonne répartition des solutions dans l'ensemble de solutions obtenues, le NSGA-II utilise un *Crowded-Comparison* approche plutôt que la *sharing parameter* σ_{share} qui rend l'algorithme moins complexe en termes de calcul et ne dépend pas d'un paramètre défini par l'utilisateur [103].

$$d_{ij} = \frac{\left| f_1^k - f_1^j \right|}{f_1^{\max} - f_1^{\min}},\ d_{ik} = \frac{\left| f_2^k - f_2^j \right|}{f_2^{\max} - f_2^{\min}},\ CDi = d_{ij} + d_{ik} \qquad \text{IV.13}$$

Tout d'abord, la distance moyenne de deux points de chaque côté d'une solution particulière est calculée le long des deux objectifs, pour obtenir une estimation de la densité de solutions entourant une solution particulière dans la population comme le montre la figure IV-1, ensuite *Crowded-Comparison Operator* est mis en application, afin d'éviter un ensemble de solutions agrégé ou vide, et d'assurer un front de Pareto bien épargné dans l'espace de solutions.

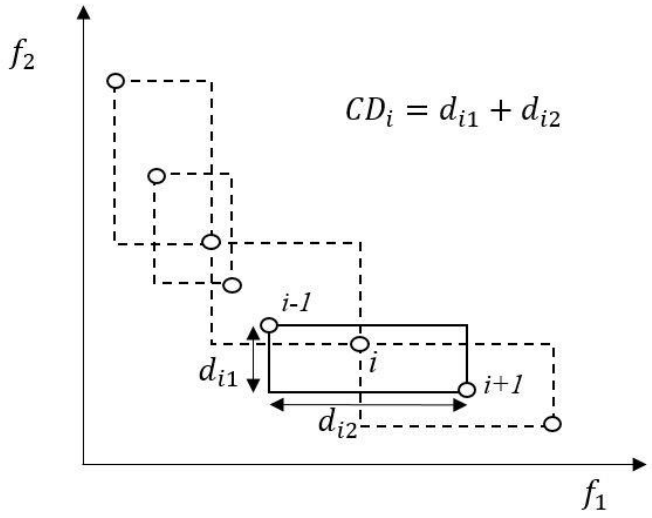

Figure IV-1 Calculs de *Crowding distance*.

IV.3.1.3 Complexité de calcul d'une génération avec NSGA-II

La partie *fast non-dominated sorting* est la partie la plus complexe de calcul de NSGA-II. Pour trouver si une solution est dominée, cette dernière est comparée à toutes les autres solutions de la population avec une valeur, désignée par *O(NM)*, nécessitant un calcul informatique complexe. Toutes les autres solutions sont comparées de la sorte, induisant un nombre considérable de calculs ce qui explique sa complexité dans le NSGA-II et qui aboutit à une nouvelle valeur désignée par $O(N^2M)$.

IV.3.1.4 Boucle principale

Premièrement, le NSGA-II commence par générer une population parente aléatoire P_0 de taille N, puis un niveau de non-domination est attribué à chaque solution en fonction de leur *Pareto dominance*. La procédure de tri non dominé classe chaque solution en fonction de sa valeur de fitness et les attribue à plusieurs fronts F_i ; le premier front F_1 contient les solutions qui dominent les solutions des autres fronts, Q_0 une population de progéniture de la même taille que la population parente P_0 est créée grâce à l'utilisation d'opérateurs de recombinaison, de tournoi binaire et de mutation. Ces étapes sont répétées jusqu'à ce que la création d'arrêt soit atteinte.

La procédure du NSGA-II est comme suit :

- R_j de taille 2 N est créé par la combinaison de la population parente et de la progéniture $R_j = P_j \cup Q_j$.
- Le tri non-dominé est effectué sur R_j afin d'identifier F_i membres (les membres de F_1 sont une solution non dominée et doivent être soulignés et portés à travers l'algorithme).
- Les membres non-dominés des fronts $(F_1, F_2, \ldots, F_i)$ sont sélectionnés dans l'ordre de leur classement pour créer P_{j+1} de taille N .
- Nouvelle population Q_{j+1} de taille N est créée en effectuant une mutation, un croisement et une sélection sur la population P_{j+1}
- Les étapes précédentes sont répétées jusqu'à ce qu'un critère d'arrêt soit atteint.

Algorithme 1. Pseudo-code de NSGA-II.

- Generation of a random a parent population P_0 of size N .
- Allocate ranks according to non-dominated level and crowded comparison operator.
- **While** j < maximum number of iteration **do**
  - Creating an offspring population Q_j of size N applying reproduction, crossover and mutation
  - Combining via $R_j = P_j \cup Q_j$
  - Sorting on R_j and classifying them into non-dominated fronts (pareto-front) PF_i, $i = 1,2,\ldots,$ etc.
  - Setting a new population $P_{j+1} = \phi$ and $i = 1$.
    - **While** the parent population $|P_{k+1}| + |PF_i| < N$ **do**
    - Calculation the crowding distance of PF_i.
    - the *i*th non-dominated front PF_i is added to the parent population P_{j+1}
    - $i = i + 1$
    - **End while**
  - Sorting the PF_i using the crowding-based comparison operator.
  - Filling the parent population P_{j+1} with the first $N - |P_{j+1}|$ solutions of PF_i
  - Generating the offspring population Q_{j+1}.

- Setting $j = j+1$

- End while

- Collecting the non-dominated solutions in the vector *P*

IV.3.2 L'Algorithme Génétique de tri Non-dominé (NSGA-III)

IV.3.2.1 Introduction

Le NSGA-III a été proposé par Deb et Jain [104], même s'il est similaire au NSGA-II [105], le NSGA-III possède des modifications notables dans son opérateur de sélection, et contrairement au NSGA-II, le maintien de la diversité parmi les membres de la population est assisté par la fourniture et la mise à jour adaptative d'un certain nombre de points de référence uniformément répartis. Et la diversité est assurée par l'utilisation de points de référence différents à ceux du NSGA-II où l'approche *Crowding distance* est appliquée.

IV.3.2.2 Classification de la population en niveaux non dominés

Toute la population du front non-dominé des niveaux de 1 à l sont d'abord inclus dans S_t. Si $|S_t| = N$; aucune autre opération n'est nécessaire et la génération suivante est lancée avec $P_{t+1} = S_t$. Par contre, si $|S_t| > N$, les membres de 1 à $(l-1)$ fronts sont sélectionnées, i.e., $P_{t+1} = \cup_{i=1}^{l-1} F_i$ et le reste des membres de la population $(K = N - |P_{t+1}|)$ constitue le dernier Front F_l.

IV.3.2.3 Détermination des points de référence sur un hyperplan

Comme mentionné précédemment, afin d'assurer la diversité des solutions optimales obtenues, le NSGA-III utilise un ensemble de points de référence prédéterminés. Le nombre de ces points de référence est défini par :

$$H = \begin{pmatrix} C+g-1 \\ g \end{pmatrix} \qquad \text{IV.14}$$

Où : *H* est le nombre total de points de référence, *C* le nombre d'objectifs, et g le nombre de divisions nécessaires sur chaque axe objectif.

IV.3.2.4 Normalisation adaptative des membres de la population

D'abord la valeur minimale pour chaque fonction objectif z_i^{min} doit être identifiée, en déterminant le point idéal de la population désigné par S_j, ensuite chaque fonction objectif sera déplacée comme suit :

$$f_i^{'}(x) = f_i(x) - z_i^{min} \qquad \text{IV.15}$$

IV.3.2.5 Fonctionnement de l'association

Une fois la normalisation des fonctions objectif effectuée, chaque membre de la population doit être associé à une ligne référence. Cette ligne de référence par un point de référence le point d'origine du repère. Ensuite, pour chaque point (membre de la population), la distance le séparant de la ligne de référence est évaluée. Et, enfin, le point le plus proche de la ligne de référence est conservé.

IV.3.2.6 Complexité de calcul d'une génération avec NSGA-III

Comme dans le NSGA-II le *fast non-dominated sorting* est la partie la plus complexe en termes de calcul de NSGA-III. En conséquence, la complexité de calcul de cette partie de l'algorithme est $O(N^2M)$ dans le pire des cas.

IV.3.2.7 Boucle principale

Comme dans le NSGA-II, l'algorithme NSGA-III ne nécessite pas le réglage de nouveaux paramètres autres que les paramètres GA habituels tels que la taille de la population, le croisement et les probabilités de mutation et leurs paramètres associés, La population de descendants a été générée en utilisant les mêmes opérateurs génétiques que dans l'algorithme NSGA-II.

Algorithm 2. Pseudo-code de NSGA-III.
-Calculate the number of reference points *H* (input), -Initializing randomly a parent population P_j of size *N*. - **While** *j*≤maximum number of iterations **Do** - Initial non-dominated front $S_j = \phi,\ i = 1.$

- Q_j of size N an offspring population is created by applying reproduction, crossover and mutation

-Combining via $R_j = P_j \cup Q_j$

- Sorting on R_j and classifying them into non-dominated fronts (Pareto front) F_i, i=1,2, …

- **Repeat**

$S_j = S_j \cup F_i$ and $i = i + 1$

- **Until** $|S_t| \geq N$

- Last front to be included : $F_l = F_i$

- **if** $|S_j| = N$ then

$P_{j+1} = S_j$, break

- **else**

$P_{j+1} = \cup_{j=1}^{l-1} F_k$

- Points to be chosen from F_l: $K = N - |P_{j+1}|$

- Normalize the population members, and create reference set Z^r

- Associate each population member S of S_j with the reference point.

- Apply the niche counter (preservation) $k \in Z^r$.

- Keep the niche obtained solutions for the next generation.

- **End if**

- **End while**

IV.4 Étude de cas numérique

Le système de protection considéré (Figure IV-2) permet de protéger la turbine à gaz d'une survitesse en coupant l'alimentation air-carburant. Il contient quatre vannes (sous-systèmes) connectées en parallèle. Le but de ce travail est de trouver les valeurs optimales des variables de décision pour maximiser la fiabilité sans violer les contraintes. Les parties électriques et mécaniques du système assurent la détection de survitesse en continu.

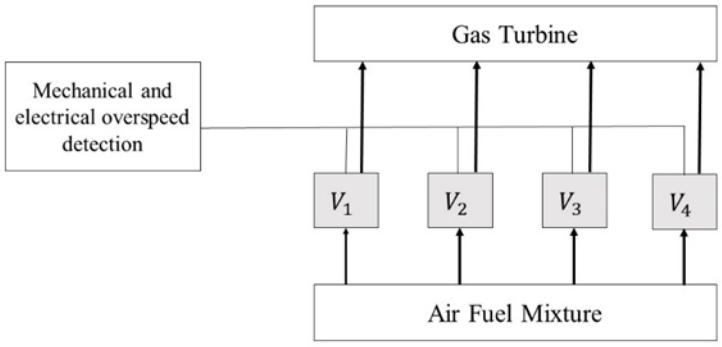

Figure IV-2 Système de protection contre la survitesse.

Dans la littérature, le présent problème a été précédemment résolu en le considérant comme un problème d'optimisation mono-objectif (fiabilité globale) [106]. Les données du système sont rapportées dans le tableau IV-1.

Dans cette application trois scenarios sont pris en considération : dans le premier scenario [107] le problème optimisation multi-objectif d'allocation de la fiabilité et de la redondance est considéré comme un problème a deux objectifs, le premier la fiabilité et le deuxième le coût. Dans le deuxième scenario [108] trois objectifs sont pris en considération en plus de la fiabilité et du coût nous avons ajouté le poids comme un troisième objectif. Enfin dans le dernier scenario [109] on a les deux objectifs de la fiabilité et le coût comme ceux du premier scenario mais une contrainte d'un seuil minimum de fiabilité est ajoutée avec trois valeurs (sous- scénario) différentes (0.7, 0.8 et 0.9).

Table IV-1 Données du système.

Sous-systèmes ***i***	$10^5 a_i$	b_i	v_i	w_i	***V***	***C***	***W***	***T*(h)**
1	1.0	1.5	1	6	250	400	500	1000
2	2.3	1.5	2	6				
3	0.3	1.5	3	8				
4	2.3	1.5	2	7				

IV.5 Résultats et discussion

Les NSGA-II et NSGA-III implémentés ont été programmés à l'aide du logiciel MATLAB et exécutés sur un PC Intel Core I7 avec 6 Go de RAM et 2,20 GHz (Windows 7, 64 bits).

IV.5.1 Premier scénario

Le modèle mathématique non linéaire du système (Figure IV-2) dans le premier scénario est formulé comme suit :

$$Maximize\ R_s = \prod_{i=1}^{4}\left[1-\left(1-r_i\right)^{n_i}\right] \qquad \text{IV.16}$$

$$Minimize\ C_s = \sum_{i=1}^{4} C\left(r_i\right)\left(n_i + \exp\left(\frac{n_i}{4}\right)\right)$$

Sous-contraintes :

$$g_1\left(r,n\right) = \sum_{i=1}^{4} C\left(r_i\right)\left(n_i + \exp\left(\frac{n_i}{4}\right)\right) \le C$$

$$g_2\left(r,n\right) = \sum_{i=1}^{4} v_i {n_i}^2 \le V \qquad \text{IV.17}$$

$$g_3\left(r,n\right) = \sum_{i=1}^{4} w_i\left(n_i \times \exp\left(\frac{n_i}{4}\right)\right) \le W$$

$$0.5 \le r_i \le 1\ ;\ r_i \in [0,1]$$

$$1 \le n_i \le 10\ ;\ n_i \in Z^+$$

$$i = 1,2,\ldots,4$$

Sachant que les paramètres spécifiques de l'algorithme (Tableau IV-2) ont été choisis afin que l'algorithme puisse fonctionner de la manière la plus optimale.

Table IV-2 Paramètres et règles du NSGA-II implémentés pour le scénario 1.

Paramètres et règles	**Valeurs**
Population	100
Crossover Percentage	0.7
Number of Parents (Offspring's)	2*round(pCrossover*nPop/2)
Mutation Percentage	0.4
Number of Mutants	round(pMutation*nPop)
Mutation Rate	0.02
Mutation Step Size	0.1*(VarMax-VarMin)

Le front de Pareto pour la fiabilité et le coût du système est illustré dans la figure IV-3. Il illustre un compromis important entre les deux objectifs, ce qui signifie qu'une fiabilité élevée entraîne un coût élevé. En d'autres termes, augmenter la fiabilité du système augmente le coût (plus cher), ce qui est en concordance avec la revue de la littérature sur ce système.

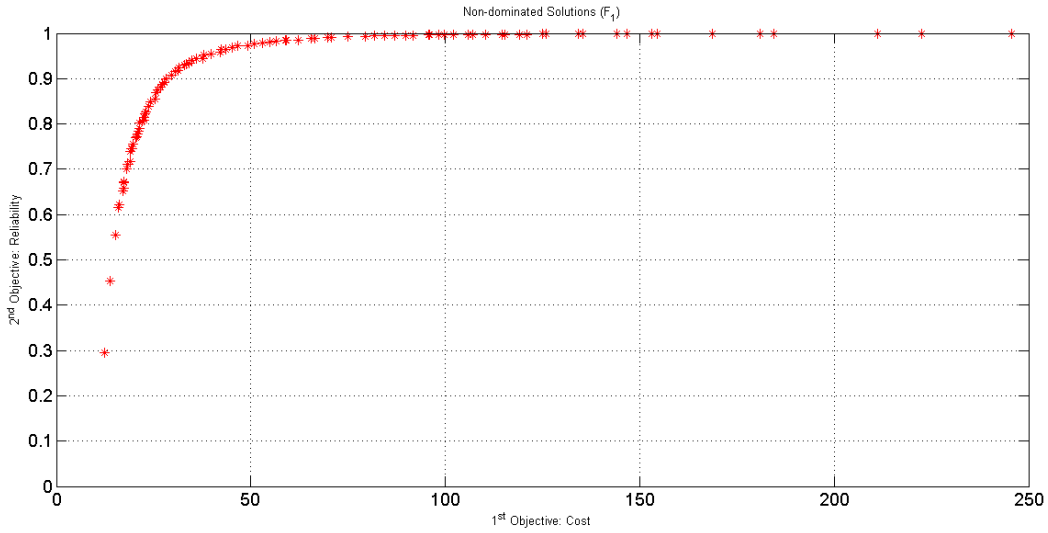

Figure IV-3 Front de Pareto Fiabilité-Coût du système.

Sachant que dans la pratique la valeur de la fiabilitè ne sera jamais égale à comme l'indique en détail la figure IV-4.

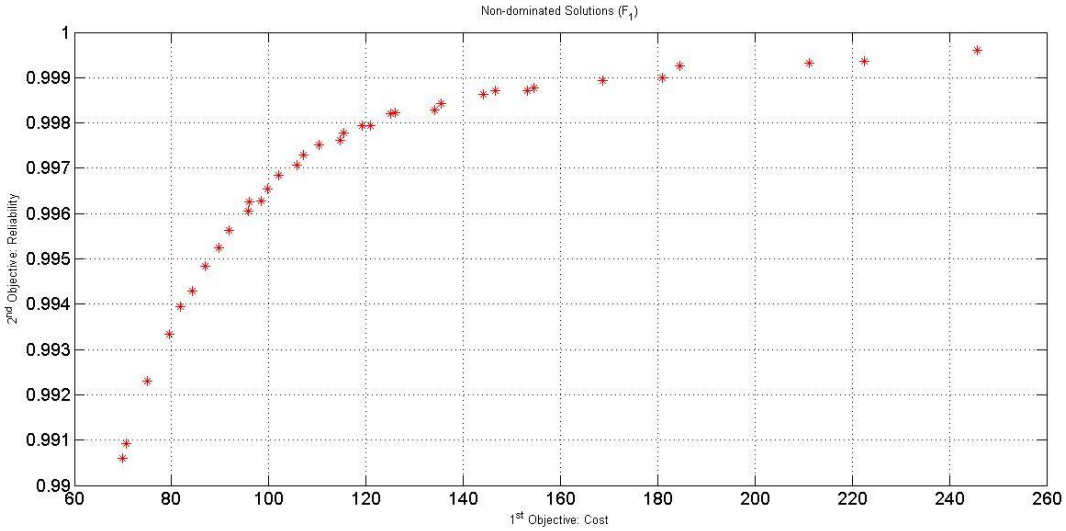

Figure IV-4 Front de Pareto Fiabilité-Coût ($R_s \geq 0.99$)

Les solutions le long du front de Pareto sont également optimales, cependant, le décideur (ou l'utilisateur) peut choisir une solution particulière plutôt qu'une autre, en fonction de l'importance de chaque objectif et de la disponibilité des ressources.

IV.5.2 Deuxième scénario

Le modèle mathématique non linéaire du système (Figure IV-2) dans le scénario deux est formulé comme suit :

$$Maximize\, R_S = \prod_{i=1}^{4}\left[1-\left(1-r_i\right)^{n_i}\right] \qquad \text{IV.18}$$

$$Minimize\ C_S = \sum_{i=1}^{4} C\left(r_i\right)\left(n_i + exp\left(\frac{n_i}{4}\right)\right)$$

$$Minimize\ V_S = \sum_{i=1}^{4} v_i n_i^{\,2}$$

Sous-contraintes :

$$g_1(r,n) = \sum_{i=1}^{4} C(r_i)(n_i + exp(\frac{n_i}{4}) \le C$$

$$g_2\left(r,n\right) = \sum_{i=1}^{4} v_i n_i^{\,2} \le V$$

$$g_3\left(r,n\right) = \sum_{i=1}^{4} w_i (n_i \exp\left(\frac{n_i}{4}\right)) \le W \qquad \text{IV.19}$$

$$0.5 \le r_i \le 1, r_i \in [0,1]$$

$$1 \le n_i \le 10, n_i \in Z^+$$

$$i = 1,2,\ldots,4$$

Dans le présent scénario on considère trois objectifs contradictoires qui sont la fiabilité, le poids et le coût. Dans la littérature, le problème actuel a été précédemment résolu en considérant soit un objectif (fiabilité globale) ou deux (fiabilité globale et coût global). Les données de l'algorithme appliqué sont rapportées dans le tableau IV-3. Une population de 40 chromosomes a été utilisée sur 50 générations.

Table IV-3 Paramètres et règles du NSGA-II implémentés pour le scénario 2.

Paramètres et règles	Valeurs
Population	40
Crossover percentage	0.8
Number of parents	2*round(pCrossover*nPop/2)
Mutation percentage	0.2
Number of mutants	round(pMutation*nPop)
Mutation rate	0.02
Mutation step size	0.1*(VarMax-VarMin)

L'ensemble généré de solutions non-dominées pour la fiabilité, le coût et le volume du système est illustré dans la figure IV-5 et au tableau IV-4. Ces derniers montrent une corrélation notable entre les trois objectifs, en d'autres termes, une fiabilité élevée entraîne un coût et un volume élevés

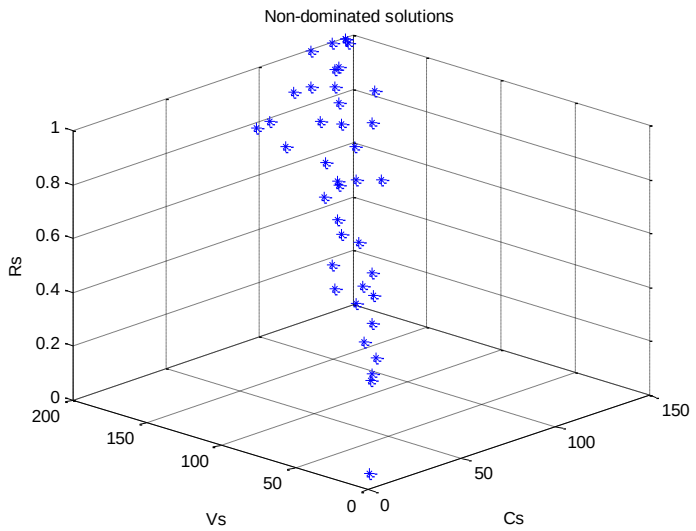

Figure IV-5 Front de Pareto obtenu en 3D du scénario 2.

Table IV-4 Solutions optimales.

Solutions	**[Rs; Cs; Vs]**
1	[0.9980; 145.865; 200]
2	[0.9979; 138.736; 200]
3	[0.3254; 18.5898; 22.0]
4	[0.8664; 39.0163; 79.0]
5	[0.9807; 86.6628; 107]
6	[0.9975; 143.3339; 195]
7	[0.9949; 118.7766; 173]
8	[0.9632; 66.3668; 93.0]
9	[0.9925; 104.9654; 155]
10	[0.7667; 31.7843; 47.0]
11	[0.9837; 77.1278; 116]
12	[0.5177; 24.3765; 28.0]
13	[0.4581; 19.6422; 28.0]
14	[0.8188; 33.9549; 64.0]
15	[0.9769; 75.6770; 128]
16	[0.6756; 27.0580; 59.0]
17	[0.9917; 111.472; 137]
18	[0.9913; 98.7680; 164]
19	[0.9160; 49.9465; 71.0]
20	[0.8977; 45.5916; 78.0]
21	[0.9976; 127.182; 200]
22	[0.9416; 50.6257; 93.0]
23	[0.4027; 21.7755; 22.0]

24	[0.9768; 62.5323; 146]
25	[0.9467; 53.1944; 123]
26	[0.3484; 19.1418; 22.0]
27	[0.5806; 22.5116; 37.0]
28	[0.9901; 92.2691; 137]
29	[0.9966; 121.3334; 173]
30	[0.7819; 30.5318; 57.0]
31	[0.6668; 32.0410; 38.0]
32	[0.6059; 23.5406; 52.0]
33	[0.9084; 57.4735; 64.0]
34	[0.5990; 28.9411; 33.0]
35	[0.9901; 90.0338; 164]
36	[0.9693; 55.3834; 146]
37	[0.6353; 26.0802; 37.0]
38	[0.9068; 45.9731; 79.0]

Les figures IV-6 et IV-7 illustrent un affichage 2D de la même exécution que la 3D afin de montrer la relation étroite fiabilité vs volume et fiabilité vs coût, respectivement.

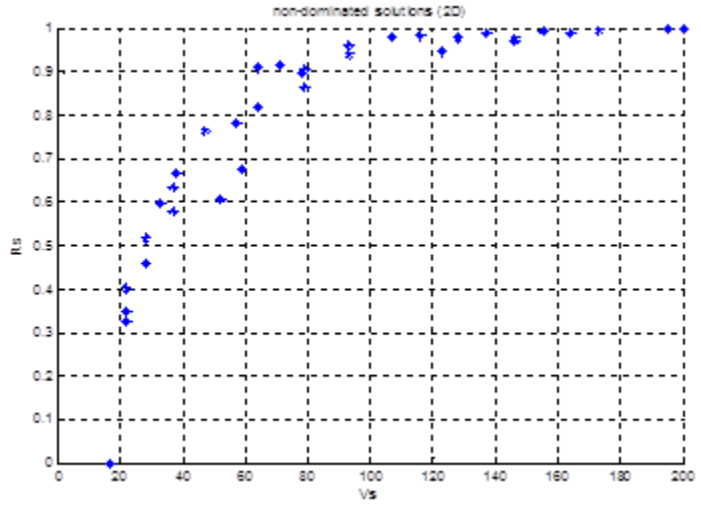

Figure IV-6 Front de Pareto en 2D (Rs vs. Vs) du scénario 2.

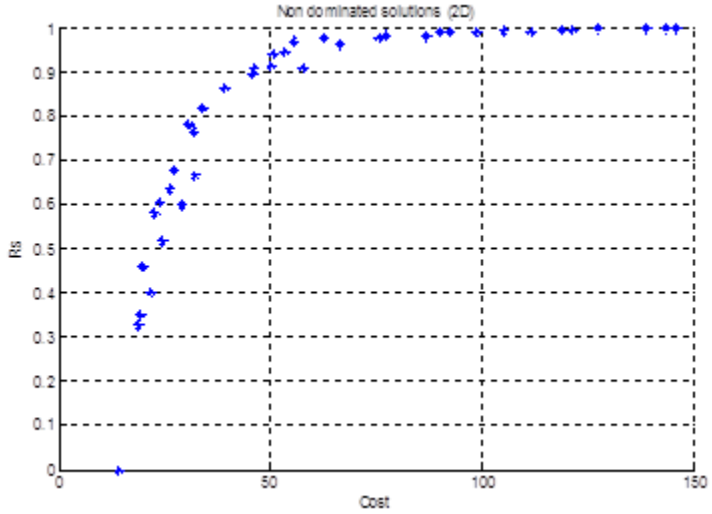

Figure IV-7 Front de Pareto en 2D (Rs vs. Cs) du scénario 2.

A titre d'exemple, le tableau IV-5 montre les variables de décision de la solution (#5) de Front de Pareto : [0.9807 ; 86.6628 ; 107,0].

Table IV-5 Exemple de variables de décision.

i	**1**	**2**	**3**	**4**
n_i	4	4	3	4
r_i	0.7209	0.7968	0.8541	0.6960

IV.5.3 Troisième scénario

La formulation mathématique du système dans le scénario trois est illustrée comme suit:

$$Maximize\, R_S = \prod_{i=1}^{4}\left[1-\left(1-r_i\right)^{n_i}\right] \qquad \text{IV.20}$$

$$Minimize\ C_S = \sum_{i=1}^{4} C\left(r_i\right)\left(n_i + exp\left(\frac{n_i}{4}\right)\right)$$

Sous-contraintes : IV.21

$$g_1(r,n) = \sum_{i=1}^{4} C(r_i)\left(n_i + exp\left(\frac{n_i}{4}\right)\right) \le C$$

$$g_2\left(r,n\right) = \sum_{i=1}^{4} v_i {n_i}^2 \le V$$

$$g_3\left(r,n\right) = \sum_{i=1}^{4} w_i\left(n_i \exp\left(\frac{n_i}{4}\right)\right) \le W$$

$$g_4(r,n) = \prod_{i=1}^{4}\left[1-\left(1-r_i\right)^{n_i}\right] \ge R_m$$

$$0.5 \le r_i \le 1,\ r_i \in [0,1]$$

$$1 \le n_i \le 10,\ n_i \in \mathbb{Z}^+$$

$$i = 1, 2, \ldots, 4$$

Pour garantir les meilleures performances de l'algorithme NSGA-III, ce dernier a été exécuté en dix fois indépendamment et les paramètres indiqués dans le tableau IV-6 ont été ajustés sur la base des expériences précédentes et des données épuisées de la littérature.

Table IV-6 Paramètres et règles du NSGA-III implémenté pour le scénario 3.

Paramètres et règles	Valeurs
Population	30
Crossover Percentage	0.8
Number of Parents	2*round(pCrossover*nPop/2)
Mutation Percentage	0.2
Number of Mutants	round(pMutation*nPop)
Mutation Rate	0.02
Mutation Step Size	0.1*(VarMax-VarMin)

La figure IV-8 illustre le front de Pareto du problème d'optimisation considéré dans ce scénario 3.1 où la fiabilité minimale admissible est supérieure ou égale à $R_m \geq 0.7$.

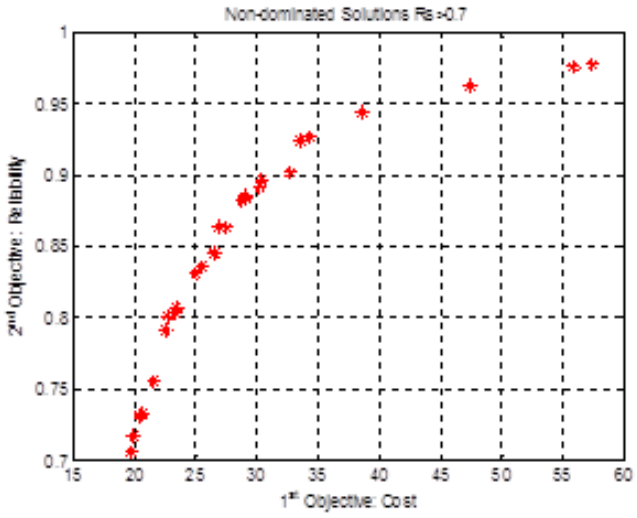

Figure IV-8 Front de Pareto dans le scénario 3.1 ($R_s \geq 0.7$)

Le tableau IV-7 fournit les variables de décision d'une des solutions optimales [0,8634 ; 27.5432] du front de Pareto dans le scénario 3.1. Le tableau IV-8 représente les valeurs des deux fonctions objectif, le temps CPU nécessaire était de 11,4138 secondes.

Table IV-7 Exemple de variables de décision dans le scénario 3.1.

i	**1**	**2**	**3**	**4**
n_i	4	5	4	4
r_i	0.5989	0.4047	0.7042	0.5684

Table IV-8 Résultats dans le scénario 3.1.

Solution No	**[R_s; C_s]**
1	[0.9775; 57.4000]
2	[0.8359; 25.4827]
3	[0.7559; 21.5633]
4	[0.9759; 55.8407]
5	[0.8309; 24.9933]
6	[0.7321; 20.6046]
7	[0.7056; 19.6824]
8	[0.9242; 33.4961]
9	[0.8960; 30.3921]
10	[0.8449; 26.4854]
11	[0.7174; 19.9044]
12	[0.9266; 34.2210]
13	[0.9016; 32.6813]
14	[0.8842; 29.0011]
15	[0.8634 ; 27.5432]

16	[0.7174; 19.9044]
17	[0.9431; 38.5613]
18	[0.8920; 30.2230]
19	[0.8852; 29.0910]
20	[0.8456; 26.5329]
21	[0.8038; 23.3018]
22	[0.8012; 22.6870]
23	[0.7176; 19.9672]
24	[0.9625; 47.3799]
25	[0.9016; 32.6813]
26	[0.8822; 28.7277]
27	[0.8630; 26.8675]
28	[0.8069; 23.4049]
29	[0.7914; 22.5525]
30	[0.7320; 20.4299]

La figure IV-9 illustre le front de Pareto du problème d'optimisation considéré dans ce scénario 3.2 où la fiabilité minimale admissible est supérieure ou égale à $R_m \geq 0.8$.

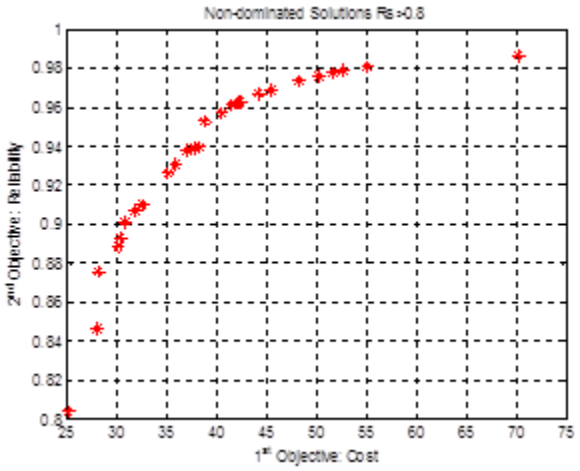

Figure IV-9 Front de Pareto dans le scénario 3.2 ($R_s \geq 0.8$)

Le temps CPU requis était de 12.0793 secondes pour le scénario 3.2 ($R_m = 0.8$), on peut observer que l'exécution de ce dernier a consommé plus de temps CPU que le premier, ceci est dû à la taille de l'espace de recherche, ce dernier est devenu plus petit dans le deuxième scénario, ce qui rend plus difficile la recherche de solutions optimales ou quasi optimales.

Le tableau **IV-9** fournit les variables de décision d'une des solutions optimales [81.3346; 0.9696] du front de Pareto dans le scénario 3.2. Le tableau IV-10 représente les valeurs des deux fonctions objectif.

Table IV-9 Exemple de variables de décision dans le scénario 3.2.

i	1	2	3	4
n_i	5	5	5	4
r_i	0.6352	0.5370	0.7043	0.5506

Table IV-10 Résultats dans le scénario 3.2.

Solution No	$[R_s; C_s]$
1	[0.9809; 54.9735]
2	[0.8038; 25.1292]
3	[0.9866; 70.0702]
4	[0.8467; 28.0192]
5	[0.9763; 50.1290]
6	[0.9778; 51.5201]
7	[0.9669; 44.1762]
8	[0.9763; 50.1290]
9	[0.9620; 42.0191]
10	[0.9398; 38.3351]
11	[0.9382; 37.0109]

12	[0.9790; 52.5576]
13	[0.9623; 42.1879]
14	[0.9526; 38.7655]
15	[0.9306; 35.8562]
16	[0.8757; 28.1283]
17	[0.9629; 42.3260]
18	[0.9393; 37.8494]
19	[0.9259; 35.1243]
20	[0.8888; 30.1637]
21	[0.9621; 42.0608]
22	[0.9571; 40.4700]
23	[0.9070; 31.9061]
24	[0.8929; 30.3398]
25	[0.9691; 45.4568]
26	[0.9609; 41.4566]
27	[0.9102; 32.5990]
28	[0.9010; 30.7811]
29	[0.9736; 48.2306]
30	[0.9609; 41.4566]

Le front de Pareto du dernier scénario 3.3 ($R_m = 0.9$) est présenté dans la figure IV-10, où une fois de plus nous pouvons observer un compromis sérieux entre le coût et la fiabilité, plus le système est fiable et plus la conception devient plus chère

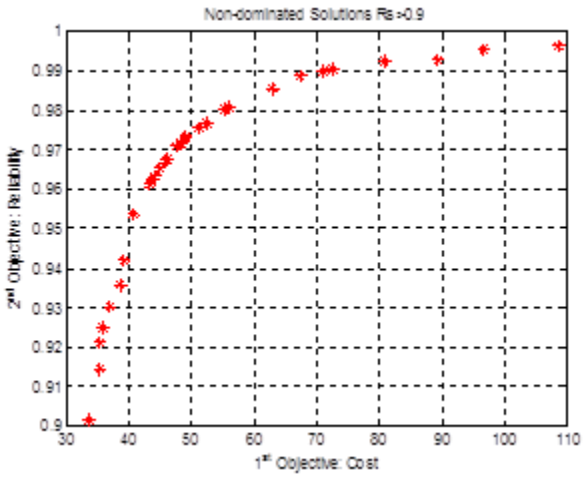

Figure IV-10 Front de Pareto dans le scénario 3.3 ($R_s \geq 0.9$)

Le tableau IV-11 présente les huit variables de décision réelles et entières d'une des solutions non dominées [0.9420 ; 39.0881]. Le tableau IV-12 rapporte les solutions non dominées. Le temps CPU requis était de 12,4503 secondes.

Table IV-11 Exemple de variables de décision dans le scénario 3.3.

i	1	2	3	4
n_i	5	5	3	5
r_i	0.6661	0.5394	0.7872	0.5232

Table IV-12 Résultats dans le scénario 3.3.

Solution No	$[R_s; C_s]$
1	[0.9963; 108.708]
2	[0.9956; 96.6096]
3	[0.9535; 40.6121]
4	[0.9898; 71.0390]
5	[0.9768; 52.3875]

6	[0.9303; 36.8210]
7	[0.9249; 35.8153]
8	[0.9927; 89.3079]
9	[0.9803; 55.0948]
10	[0.9358; 38.4941]
11	[0.9017; 33.5716]
12	[0.9905; 72.5534]
13	[0.9855; 62.8539]
14	[0.9653; 44.8141]
15	[0.9420; 39.0881]
16	[0.9211; 35.2746]
17	[0.9926; 80.7568]
18	[0.9886; 67.2674]
19	[0.9611; 43.2111]
20	[0.9145; 35.1975]
21	[0.9729; 48.9607]
22	[0.9675; 45.7567]
23	[0.9632; 44.0424]
24	[0.9736; 49.0168]
25	[0.9710; 47.6239]
26	[0.9624; 43.6374]
27	[0.9718; 48.2624]
28	[0.9676; 46.1015]
29	[0.9807; 55.9130]
30	[0.9757; 51.0314]

VI. Conclusion

Dans ce chapitre, le problème d'optimisation multi-objectif d'allocation de la fiabilité et de la redondance d'un système de protection dans une centrale électrique a été traité.

Le travail actuel aborde trois scénarios, dans le premier nous avons considéré le problème avec deux objectifs la fiabilité globale et le coût, dans le deuxième scénario nous avons considéré trois objectifs la fiabilité globale, le coût global et le poids global. Les deux premiers scénarios ont été résolus en utilisant NSGA-II.

Dans le troisième et dernier scénario, nous avons considéré le problème avec deux objectifs la fiabilité globale et le coût avec une contrainte supplémentaire à savoir ; une fiabilité minimale admissible, le problème a été résolu en utilisant l'algorithme génétique de tri non dominé III (NSGA-III).

L'approche multi-objectif offre au décideur un nombre considérable de solutions optimales (un ensemble de solutions non dominées (front de Pareto)), à l'inverse de l'approche mono-objectif où une solution unique est générée pas les outils de résolution. Dans le chapitre suivant nous allons voir comment on peut réduire le nombre de solutions optimales, afin de faciliter la tâche au décideur et de l'aider à choisir une solution optimale en fonction de la cible de conception et de la limitation des ressources.

Chapitre V

Optimisation floue de la fiabilité des systèmes multi-objectif par des algorithmes génétiques et analyse de clustering

Chapitre V

Optimisation floue de la fiabilité des systèmes multi-objectif par des algorithmes génétiques et analyse de clustering

V.1 Introduction

De nos jours, les installations industrielles doivent être aussi fiables que possible en raison de la compétitivité de l'industrie. Afin d'atteindre un haut niveau de fiabilité d'un système de production, comme déjà vu précédemment, trois stratégies fondamentales peuvent être appliquées par les concepteurs : l'allocation de fiabilité, l'allocation de redondance et l'allocation de fiabilité-redondance. Ces stratégies impliquent l'optimisation d'un ou plusieurs objectifs (tels que la fiabilité globale du système et le coût global du système) sous l'ensemble des contraintes de conception (tels que le volume, le poids et le coût) [111-121]. Le problème d'allocation de fiabilité-redondance (RRAP) est un problème NP-Hard [122], dans lequel les allocations de redondance et de fiabilité sont les variables de décision. Par conséquent, un compromis entre les variables de décision est nécessaire pour optimiser un ou plusieurs objectifs sans violer l'ensemble des contraintes de conception considéré. De nombreuses métaheuristiques ont été utilisées comme approches de solutions pour résoudre le RRAP et peuvent être résumées dans les références [104,103-125]. La plupart des travaux consacrés au problème d'allocations de fiabilité (RRAP) traitent des problèmes mono-objectif.

Les algorithmes évolutionnaires multi-objectif ont montré leur capacité à résoudre des problèmes d'optimisation multi-objectif. Leur particularité est de générer un ensemble de solutions (solutions non dominées) les plus proches possible des solutions optimales, appelé Front de Pareto. Une autre particularité est qu'ils sont capables de maintenir la diversité dans la solution à travers les itérations. Plusieurs algorithmes évolutionnaires multi-objectif ont été développés au cours des dernières décennies, tels que MOGA, NPGA, NSGA, SPEA, PAES, MOMGA, SPEA2, PEAS-II, NSGA-II, MOEA/D et NSGA-III [126-136].

Dans [137,138], les problèmes d'allocation de fiabilité-redondance multi-objectif (MORRAP) ont fait l'objet d'un examen à l'aide NSGA-II. Un système série-parallèle composé de "*k-out-of-n*" sous-systèmes avec trois objectifs ont été résolu en utilisant NSGA-II dans [140]. Les objectifs de fiabilité, de coût, de poids et de volume ont été convertis en un problème mono-objectif dans [141,142] et résolu à l'aide d'une optimisation adaptative d'essaim de particules (PSO). Quelques approches ont été proposées dans la littérature pour considérer les données floues. Dans [143], les incertitudes sur l'intervalle de temps ont été étudiées aussi. Les paramètres du système flou ont été pris en compte dans [144] pour le problème d'allocation de redondance uniquement.

D'autre part, le problème a été considéré comme un problème d'optimisation mono-objectif et les paramètres flous ont été défuzzifiés en utilisant trois méthodes.

Le but de ce chapitre est de proposer une approche de solution permettant de traiter efficacement l'optimisation des objectifs de fiabilité et de coût du système sous des paramètres flous du système. Le Non-Sorting Genetic Algorithm II (NSGA-II) et le Non-Sorting Genetic Algorithm III (NSGA-III) sont tous deux utilisés, une méthode de défuzification est mise en œuvre pour convertir les paramètres flous en paramètres nets. Des métriques de mesure de performance sont utilisées pour trouver la meilleure méthode d'optimisation multi-objectif adaptée à ce type de problème, puis les meilleures solutions de compromis du Front de Pareto sont sélectionnées en recourant à une analyse de clustering pour aider le décideur. Le reste du chapitre est organisé comme suit : la section 2 décrit les problèmes, la section 3 présente l'approche de solution proposée, une étude de cas numérique est présentée dans la section 4 afin de mettre en évidence l'applicabilité de l'approche proposée. Les résultats avec une discussion sont donnés dans la section 5. Enfin, la dernière section illustre les conclusions de ce travail.

V.2 Optimisation multi-objectif floue de la fiabilité du système

Les problèmes d'optimisation de la fiabilité du système peuvent être caractérisés par le nombre d'objectifs, c'est-à-dire l'optimisation de la fiabilité du système mono-objectif ou multi-objectif. Un aperçu de ces méthodes avec des paramètres exacts est donné dans [111, 112, 145, 146 ,113-118, 138]. Dans [144], l'optimisation floue de la fiabilité du système mono-objectif a été étudiée. Le présent chapitre considère la manière floue multi-objectif.

Allocation floue multi-objectif de la fiabilité :

$$\begin{aligned} &\text{Maximize} \quad \tilde{R}_s(\tilde{r}) = \tilde{R}_s(\tilde{r}_1, \tilde{r}_2, ..., \tilde{r}_m) \\ &\text{Minimize} \quad \tilde{C}_s(\tilde{r}) = \tilde{C}_s(\tilde{r}_1, \tilde{r}_2, ..., \tilde{r}_m) \end{aligned} \tag{V.1}$$

Sous-contraintes :

$$\begin{aligned} &\tilde{g}_j(\tilde{r}_1, \tilde{r}_2, ..., \tilde{r}_m) \le \tilde{b} \\ &0 \le \tilde{r}_i \le 1; \;\; i = 1, 2, ..., m \\ &\tilde{r}_i \in [0,1] \subset R^+ \end{aligned} \tag{V.2}$$

Allocation floue multi-objectif de la redondance :

$$\begin{aligned} &\text{Maximize} \quad \tilde{R}_s(\tilde{n}) = \tilde{R}_s(\tilde{n}_1, \tilde{n}_2, ..., \tilde{n}_m) \\ &\text{Minimize} \quad \tilde{C}_s(\tilde{n}) = \tilde{C}_s(\tilde{n}_1, \tilde{n}_2, ..., \tilde{n}_m) \end{aligned} \qquad \text{V.3}$$

Sous-contraintes :

$$\begin{aligned} &\tilde{g}_j(\tilde{n}_1, \tilde{n}_2, ..., \tilde{n}_m) \le \tilde{b} \\ &1 \le \tilde{n}_i \le \tilde{n}_{i\max}, \ i = 1, 2, ..., m; \\ &\tilde{n}_i \in \mathbb{Z}^+ \end{aligned} \qquad \text{V.4}$$

Allocation floue multi-objectif de la fiabilité-redondance :

$$\begin{aligned} &\text{Maximize} \quad \tilde{R}_s(\tilde{r}, \tilde{n}) = \tilde{R}_s(\tilde{r}_1, \tilde{r}_2, ..., \tilde{r}_m;\ \tilde{n}_1, \tilde{n}_2, ..., \tilde{n}_m) \\ &\text{Minimize} \quad \tilde{C}_s(\tilde{r}, \tilde{n}) = \tilde{C}_s(\tilde{r}_1, \tilde{r}_2, ..., \tilde{r}_m;\ \tilde{n}_1, \tilde{n}_2, ..., \tilde{n}_m) \end{aligned} \qquad \text{V.5}$$

Sous-contraintes :

$$\begin{aligned} &\tilde{g}_j(\tilde{r}_1, \tilde{r}_2, ..., \tilde{r}_m;\ \tilde{n}_1, \tilde{n}_2, ..., \tilde{n}_m) \le \tilde{b} \\ &0 \le \tilde{r}_i \le 1;\ 1 \le \tilde{n}_i \le \tilde{n}_{i\max}, \ i = 1, 2, ..., m; \\ &\tilde{r}_i \in [0,1] \subset R^+;\ \tilde{n}_i \in \mathbb{Z}^+ \end{aligned} \qquad \text{V.6}$$

Où : $\tilde{X}$ désigne la valeur floue des paramètres X, $\tilde{R}_s(\cdot)$ est la fiabilité du système, $\tilde{C}_s(\cdot)$ est le coût du système, $\tilde{g}_j(\cdot)$ est l'ensemble des contraintes de conception (tels que le coût, le poids et le volume). $\tilde{r}_i$ est la fiabilité du sous-système *i*. $\tilde{n}_i$ est le nombre de composants redondants dans le sous-système *i*. *m* est le nombre total de sous-systèmes dans le système, $\tilde{n}_{i\max}$ est le nombre maximal autorisé de composants redondants dans le sous-système *i*, et $\tilde{b}$ est le vecteur de la limitation des ressources.

V.3 Approche de la solution proposée

L'approche de solution proposée dans ce travail, comme le montre la figure V-1 est basée sur les principales étapes suivantes :

- Défuzzification des paramètres flous.
- Générez les fronts de Pareto à l'aide des NSGA-II et NSGA-III avec la gestion des contraintes.
- Identifiez le meilleur Front de Pareto.
- Réduire le front de Pareto par une analyse de clustering.

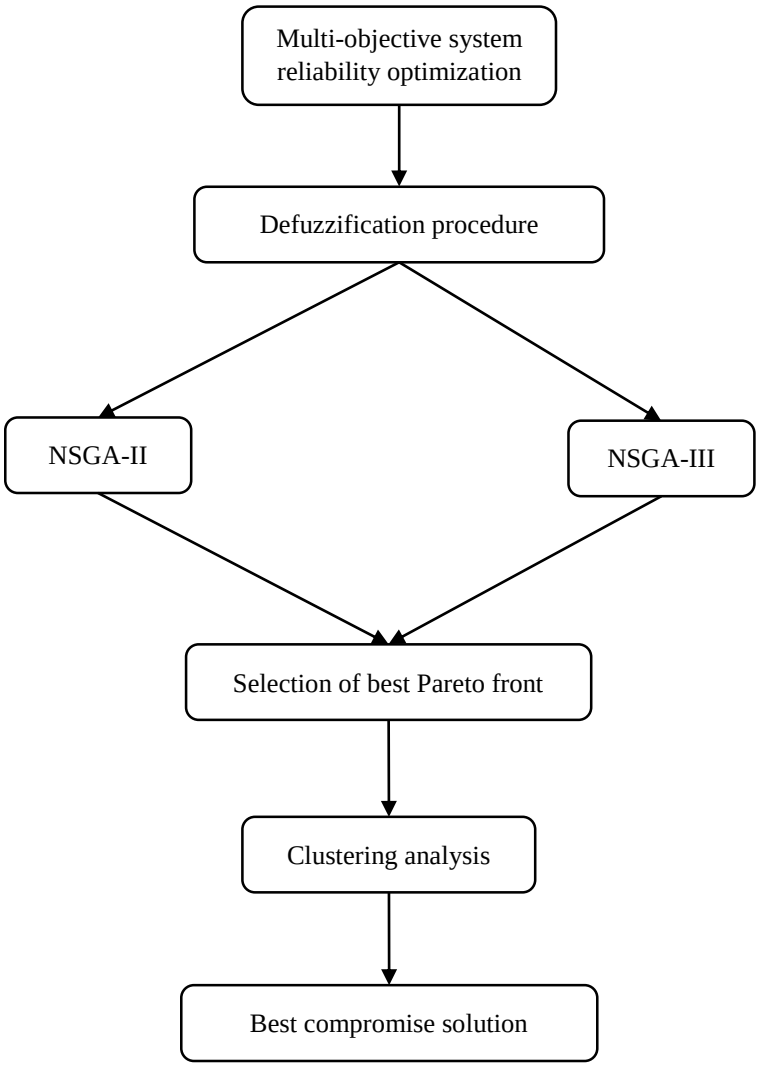

Figure V-1 Organigramme de l'approche de la solution proposée.

La complexité de calcul de l'approche de la solution proposée est donnée dans l'annexe 1 [128, 135, 148]. Étant donné que le pire des cas est toujours pris en compte lors de l'étude de la complexité donc, la complexité de calcul globale du pire cas de l'approche de solution proposée est $O(N^2M)$.

V.3.1 Procédure de défuzzification

Les paramètres flous sont introduits lorsque les valeurs sont inexactes ou inconnues à l'avance. Les définitions et les opérations arithmétiques des nombres flous peuvent être trouvées dans [159, 160]. Plusieurs procédures de défuzzification ont été proposées dans la littérature. La ranking *function procédure* est l'une des méthodes les plus connues [144, 149, 150]. L'approche proposée adopte cette méthode et elle est définie comme suit [144, 150] :

Un nombre flou triangulaire représentant un paramètre (noté $\tilde{A} = (a, a', a'')$, Où : a est la valeur inférieure, a' est la valeur modale et a'' est la valeur supérieure, est défuzzifié (converti en une valeur nette) par la formule suivante :

$$\Re(\tilde{A}) = \frac{a + 2a' + a''}{4} \qquad \text{V.7}$$

V.3.2 Génération des fronts de Pareto

Comme déjà vu dans le chapitre précèdent, le NSGA-II et le NSGA-III sont implémenté dans le présent travail.

V.3.2.1 Gestion des contraintes

Comme il a été montré dans le chapitre précédent, diverses méthodes de gestion des contraintes peuvent être trouvées dans la littérature. Dans le présent travail, la méthode de la fonction de pénalité est utilisée :

$$\begin{aligned} \text{Fitness value} &= -R_S(\cdot) + \psi(\cdot) \\ \text{Fitness value} &= C_S(\cdot) + \psi(\cdot) \end{aligned} \qquad \text{V.8}$$

Où : $\psi(\cdot)$ représente la fonction de pénalité, calculée comme suit :

$$\psi(\cdot) = \sum_{j=1}^{M} \phi_j . \max(0, g_j(\cdot))^2 \qquad \text{V.9}$$

Où : ϕ_j est le facteur de pénalité.

V.3.2.2 Meilleur Front de Pareto

Une métrique de mesure de performance est utilisée pour comparer les fronts générés par chaque algorithme. Dans ce travail, le *spacing* est appliquée [151,154]. Cette métrique décrit comment les solutions sont distribuées le long de Fronts de Pareto. Le nombre de solutions non-dominées est également pris en compte pour observer quel algorithme génère le plus de solutions non-dominées. Le *spacing* est calculé comme suit [-] :

$$S = \sqrt{\frac{1}{|n_{PF} - 1|} \sum_{w=1}^{n_{PF}} (\overline{d} - d_w)^2} \qquad \text{V.10}$$

Où : n_{PF} est le nombre de solutions non-dominées.

$$d_w = \min_{w,w \neq l} \sum_{p=1}^{q} \left| f_p^w - f_p^l \right| \quad w, l = 1,2,\ldots,n_{PF} \qquad \text{V.11}$$

Où : q est le nombre d'objectifs dans le problème et f_p est la valeur de la fonction objectif de l'objectif p_t.

$$\overline{d} = \sum_{w=1}^{n_{PF}} d_w / |n_{PF}| \qquad \text{V.12}$$

Plus S est petit plus le front de Pareto est plus uniformément dispersé dans l'espace des solutions ; Où : $S = 0$ représente la meilleure performance possible.

V.3.3 Analyse de clustering

Le Front de Pareto est un ensemble de solutions, ce qui rend difficile pour le décideur d'identifier les solutions optimales. Les méthodes permettant de faciliter cette tâche sont une partie clé de la solution. Un aperçu de ces méthodes est disponible dans [155-158]. Dans le présent travail, le *k-means algorithm* [156] est utilisé pour identifier les groupes de solutions ayant des caractéristiques similaires dans l'espace des fonctions objectif. Cette méthode a une grande efficacité dans le regroupement des ensembles de données. L'idée principale, est une fois que le centroïde de chaque groupe est calculé, une solution est assignée au groupe avec le centroïde le plus proche. L'algorithme vise à minimiser la distance au carré entre chaque centre et ses points de données alloués.

Les étapes générales du *k-means clustering* sont résumées comme suit [156] :

- Un groupe initial de centroïdes représentés par des points *k* est placé dans l'espace de l'objet.
- Chaque groupe est affecté par les solutions les plus proches de son centre de gravité.
- Les positions des *k* centroïdes sont recalculées une fois que toutes les solutions ont été attribuées.
- Les étapes 2 et 3 sont répétées jusqu'à ce que l'allocation de cluster ne change plus.

L'algorithme *k-means* optimise la fonction objectif suivante :

$$KM(N,C)=\sum_{a}\min_{b\in\{1,\ldots,K\}}\left\|f(x_a)-c_b\right\|^2 \qquad \text{V.13}$$

Où:

$f\left(x_a\right)=ath$ Vecteur de données.

$C_b=bth$ Centroïde du cluster.

N Ensemble de vecteurs de données.

C Ensemble de centroïdes.

V.4 Étude de cas

Un exemple numérique tiré de [144] est considéré pour illustrer l'applicabilité de l'approche de la solution proposée. Il s'agit d'un système en série à 3 étages (voir Figure V-2) avec différents composants redondants, c'est-à-dire des alternatives de conception. Les données nettes et floues du système sont : $C_0 = 30$ et $W_0 = 17;\ \tilde{C}_0 = (26,\ 30,\ 33)$ et $\tilde{W}_0 = (14,\ 17,\ 19)$.

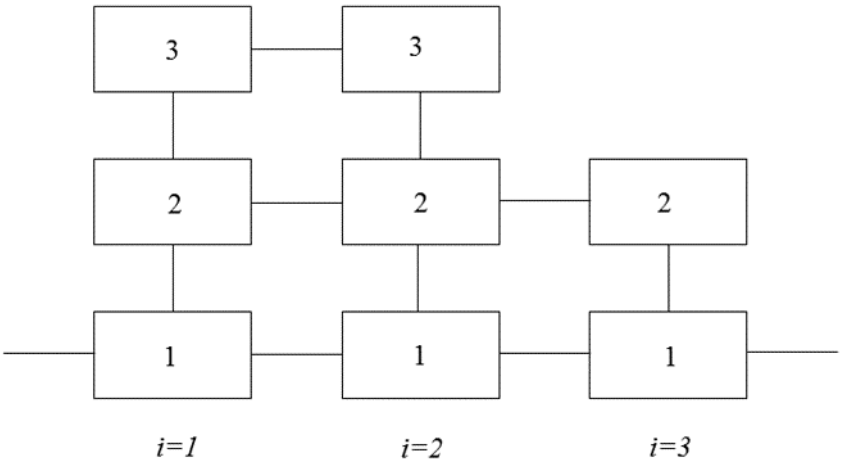

Figure V-2 Étude de cas : Système à 3 niveaux.

La fiabilité du système a été optimisée avec l'allocation de redondance en considérant les paramètres flous et la fiabilité globale comme objectif [144]. Dans ce travail, deux scénarios sont considérés pour le système : allocation de redondance et allocation de fiabilité-redondance. Les paramètres sont flous et la fiabilité globale et le coût sont considérés comme des objectifs. Les tableaux V-1 et V-2 rapportent les données des deux scénarios.

Le problème d'allocation de redondance multi-objectif défuzzifié du système est donné comme suit :

$$\text{Maximize } R_S = (1-(1-0.99)^{n_{11}}(1-0.95)^{n_{12}}(1-0.92)^{n_{13}}) \\ (1-(1-0.98)^{n_{21}}(1-0.8)^{n_{22}}(1-0.90)^{n_{23}})(1-(1-0.8)^{n_{31}}(1-0.92)^{n_{32}}) \quad \text{V.14}$$

$$\text{Minimize } C_s = \tilde{4}n_{11} \oplus \tilde{13}n_{12} \oplus \tilde{7}n_{13} \oplus \tilde{8}n_{21} \oplus \tilde{3}n_{22} \oplus \tilde{3}n_{23} \oplus \tilde{11}n_{31} \oplus \tilde{5}n_{32} \quad \text{V.15}$$

Sous-contraintes :

$$\tilde{4}n_{11} \oplus \tilde{13}n_{12} \oplus \tilde{7}n_{13} \oplus \tilde{8}n_{21} \oplus \tilde{3}n_{22} \oplus \tilde{3}n_{23} \oplus \tilde{11}n_{31} \oplus \tilde{5}n_{32} \leq \tilde{30} \quad \text{V.16}$$

$$\tilde{2}n_{11} \oplus \tilde{3}n_{12} \oplus \tilde{5}n_{13} \oplus \tilde{3}n_{21} \oplus \tilde{3}n_{22} \oplus \tilde{9}n_{23} \oplus \tilde{4}n_{31} \oplus \tilde{6}n_{32} \leq \tilde{17} \quad \text{V.17}$$

$$n_{11} + n_{12} + n_{13} + n_{21} + n_{22} + n_{23} + n_{31} + n_{32} \geq 1 \quad \text{V.18}$$

$$n_{11}, n_{12}, n_{13}, n_{21}, n_{22}, n_{23}, n_{31}, n_{32} \geq 0, \text{ entiers} \quad \text{V.19}$$

Tableau V-1 Données du système dans le scénario 1.

j	**Eléments**	**1**	**2**	**3**
1	$\tilde{c}$	(2 ,4, 5)	(6 ,8, 9)	(9 ,11, 12)
	$\tilde{w}$	(1 ,2, 4)	(2 ,3, 5)	(3 ,4, 6)
	r	0.99	0.98	0.98
	n	n_{11}	n_{12}	n_{13}
2	$\tilde{c}$	(11 ,13, 14)	(1 ,2, 4)	(3 ,5, 6)
	$\tilde{w}$	(2 ,3, 5)	(2 ,3, 6)	(5 ,6, 8)
	r	0.95	0.8	0.92
	n	n_{21}	n_{22}	n_{23}
3	$\tilde{c}$	(5 ,7, 8)	(1 ,3, 4)	
	$\tilde{w}$	(4 ,5, 7)	(8 ,9, 11)	
	r	0.92	0.90	
	n	n_{31}	n_{32}	
l_i		3	3	2

Le problème d'allocation de fiabilité-redondance multi-objectif défuzzifié du système est donné comme suit :

$$\text{Maximize } R_s = (1-(1-r_{11})^{n_{11}}(1-r_{12})^{n_{12}}(1-r_{13})^{n_{13}}) \\ (1-(1-r_{21})^{n_{21}}(1-r_{22})^{n_{22}}(1-r_{23})^{n_{23}})(1-(1-r_{31})^{n_{31}}(1-r_{32})^{n_{32}}) \quad \text{V.20}$$

$$\text{Minimize } C_s = \tilde{4}n_{11} \oplus \tilde{13}n_{12} \oplus \tilde{7}n_{13} \oplus \tilde{8}n_{21} \oplus \tilde{3}n_{22} \oplus \tilde{3}n_{23} \oplus \tilde{11}n_{31} \oplus \tilde{5}n_{32} \quad \text{V.21}$$

Sous-contraintes :

$$\tilde{4}n_{11} \oplus \tilde{13}n_{12} \oplus \tilde{7}n_{13} \oplus \tilde{8}n_{21} \oplus \tilde{3}n_{22} \oplus \tilde{3}n_{23} \oplus \tilde{11}n_{31} \oplus \tilde{5}n_{32} \le \tilde{30} \quad \text{V.22}$$

$$\tilde{2}n_{11} \oplus \tilde{3}n_{12} \oplus \tilde{5}n_{13} \oplus \tilde{3}n_{21} \oplus \tilde{3}n_{22} \oplus \tilde{9}n_{23} \oplus \tilde{4}n_{31} \oplus \tilde{6}n_{32} \le \tilde{17} \quad \text{V.23}$$

$$n_{11}+n_{12}+n_{13}+n_{21}+n_{22}+n_{23}+n_{31}+n_{32} \ge 1 \quad \text{V.24}$$

$$n_{11}, n_{12}, n_{13}, n_{21}, n_{22}, n_{23}, n_{31}, n_{32} \ge 0, \text{entiers} \quad \text{V.25}$$

$$0 \le r_{11}, r_{12}, r_{13}, r_{21}, r_{22}, r_{23}, r_{31}, r_{32} \le 1, \text{ réels} \quad \text{V.26}$$

Tableau V-2 Données du système dans le scénario 2

j	**Eléments**	**1**	**2**	**3**
1	$\tilde{c}$	(2 ,4, 5)	(6 ,8, 9)	(9 ,11, 12)
	$\tilde{w}$	(1 ,2, 4)	(2 ,3, 5)	(3 ,4, 6)
	r	r_{11}	r_{12}	r_{13}
	n	n_{11}	n_{12}	n_{13}
2	$\tilde{c}$	(11 ,13, 14)	(1 ,2, 4)	(3 ,5, 6)
	$\tilde{w}$	(2 ,3, 5)	(2 ,3, 6)	(5 ,6, 8)
	r	r_{21}	r_{22}	r_{23}
	n	n_{21}	n_{22}	n_{23}
3	$\tilde{c}$	(5 ,7, 8)	(1 ,3, 4)	
	$\tilde{w}$	(4 ,5, 7)	(8 ,9, 11)	
	r	r_{31}	r_{32}	
	n	n_{31}	n_{32}	
l_i		3	3	2

V.5 Résultats et discussion

Le problème mentionné ci-dessus (Figure V-2) a été implémenté dans les programmes NSGA-II et NSGA-III en utilisant le logiciel MATLAB 2014b et exécuté sur un PC Intel Core I7 2,20 GHz avec 6 Go de RAM. Pour garantir les meilleures performances pour les deux algorithmes et que la comparaison soit juste, les algorithmes ont été exécutés sur dix fois indépendantes et les paramètres du tableau V-3 ont été fixés en appliquant les deux étapes suivantes :

- Une plage d'une largeur raisonnable (réduite) est définie pour chaque paramètre, sur la base des expériences précédentes et de la littérature.
- Un réglage fin est effectué à l'intérieur de la plage définie ; ensuite, la méthode d'essai et d'erreur est exécutée. Un grand nombre d'essais est effectué avec les valeurs des paramètres et comparées entre elles afin de déterminer la relation entre les valeurs des paramètres et leur effet sur les résultats d'optimisation (nombre de solutions non dominées, espacement…). En d'autres termes, trouver des relations de cause à effet.

Tableau V-3 Paramètres implémentés en NSGA-II et NSGA-III.

Parameters	Values
Taille de la population	50
Pourcentage de croisement	0.8
Nombre de parents	2*round(pCrossover*nPop/2)
Pourcentage de mutation	0.2
Nombre de mutants	round(pMutation*nPop)
Taux de mutation	0.02
Taille du pas de mutation	0.1*(VarMax-VarMin)

Comme indiqué dans la section précédente, deux scénarios ont été envisagés. Dans le premier scénario, le problème d'optimisation multi-objectif flou est résolu en tant que problème d'allocation de redondance, les variables de décision sont les valeurs de redondance des composants (8 entiers) et la fiabilité de chaque composant de sous-système est fixe (prédéfinie). Dans le deuxième scénario, le problème d'optimisation multi-objectif flou est résolu en tant que problème d'allocation

de fiabilité-redondance, la fiabilité des composants et leurs valeurs de redondance sont considérées comme des variables de décision dans les deux cas. Cela en fait un problème mixte avec huit variables entières et huit variables réelles.

V.5.1 Scénario 1

Le Front de Pareto est généré par chaque algorithme dans chaque scénario. Les figures V-3 et V-4 représentent le Front de Pareto (solutions non-dominées, R_s vs C_s) généré par l'algorithme NSGA-II pour le problème d'allocation de redondance.

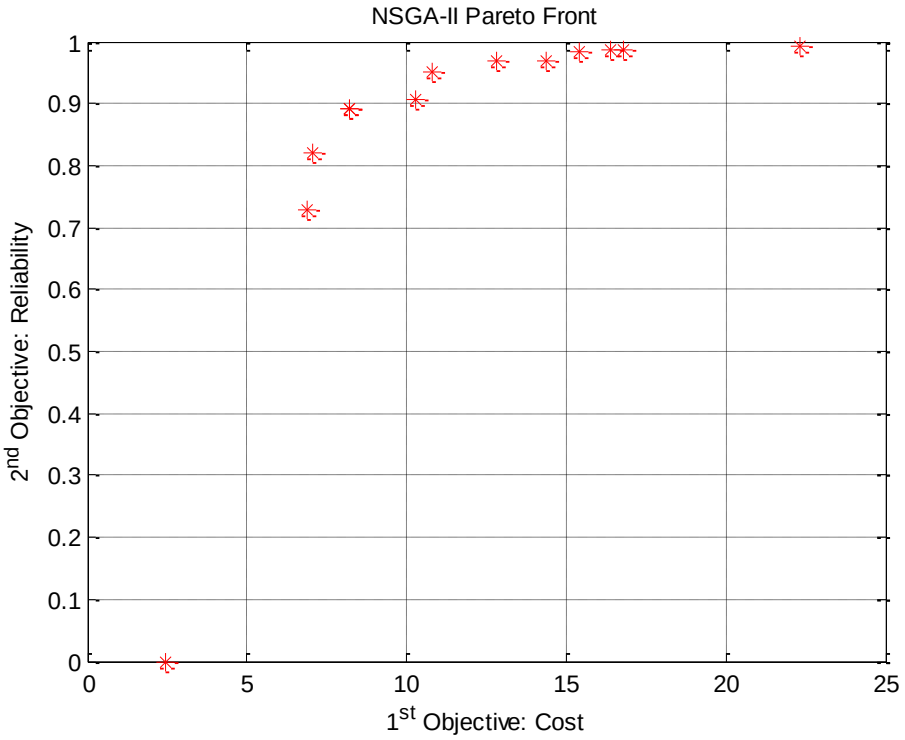

Figure V-3 Pareto Front de NSGA-II pour le scenario 1.

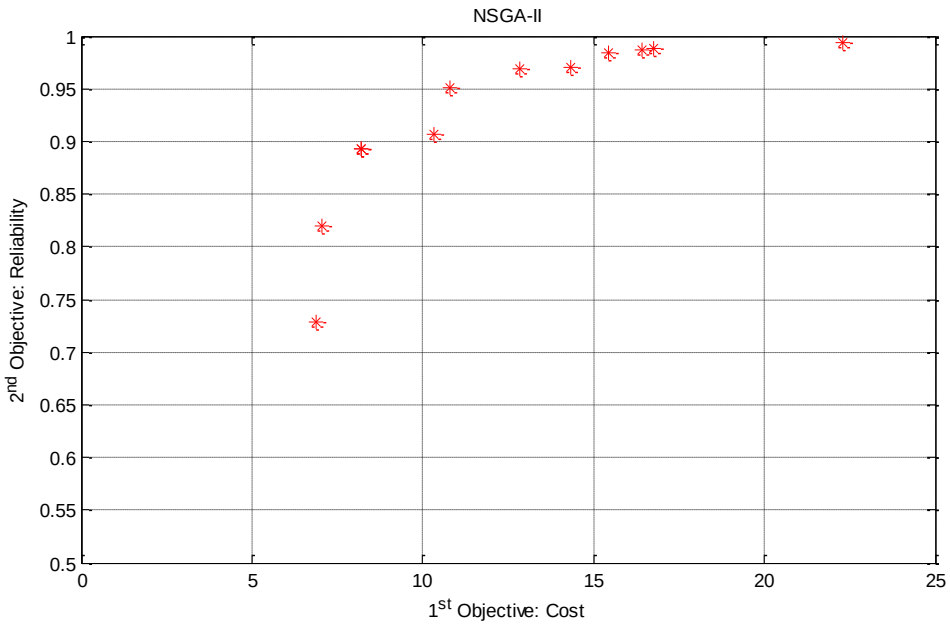

Figure V-4 Pareto Front de NSGA-II pour le scenario 1 ($R_s \geq 0.5$).

Les figures V-5 et V-6 représentent le Front de Pareto généré par NSGA-III pour le même scénario. Les deux fronts présentent un compromis notable entre le coût global du système et la fiabilité globale du système ; en d'autres termes, plus le système est fiable, plus la conception est coûteuse. Il convient de noter que le NSGA-III a généré plus de solutions non-dominées que le NSGA-II

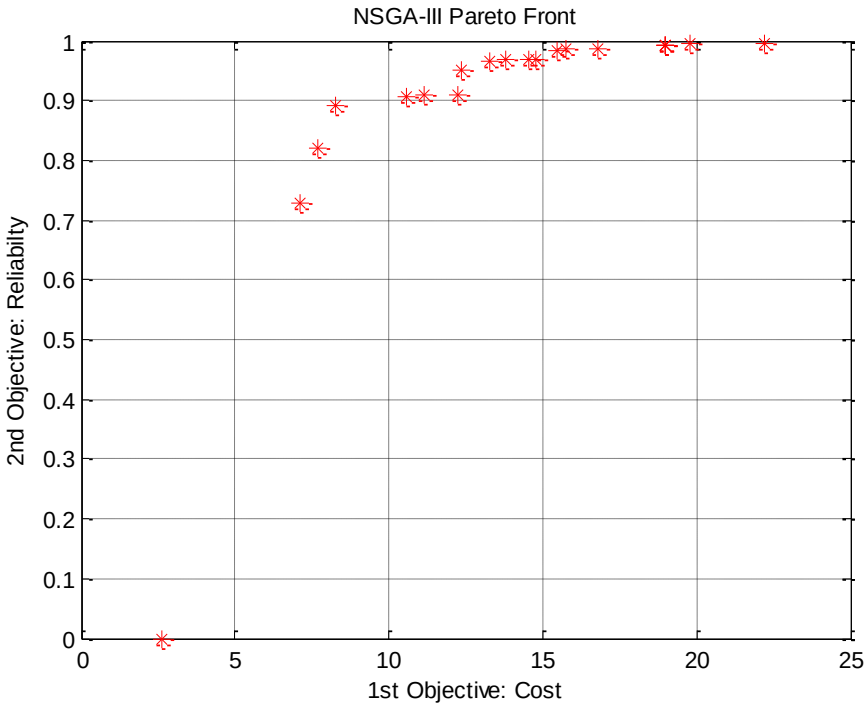

Figure V-5 Pareto Front de NSGA-III pour le scenario 1.

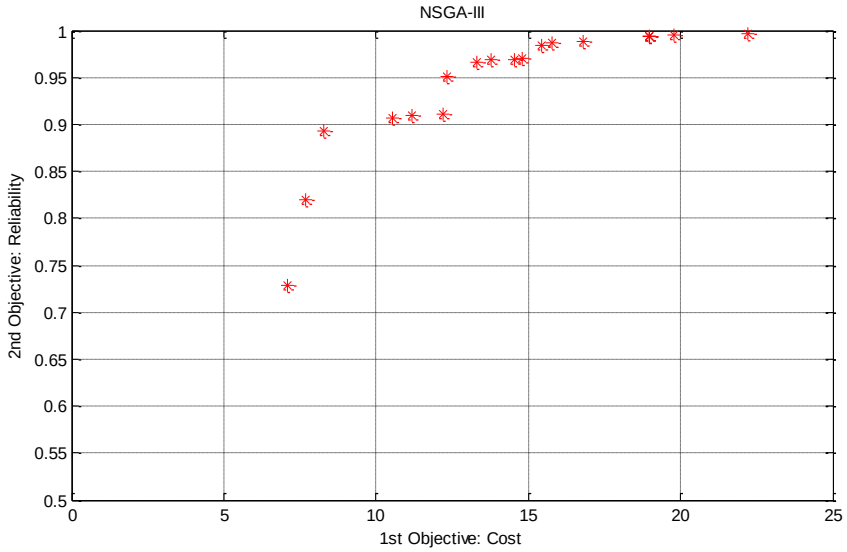

Figure V-6 Pareto Front de NSGA-III pour le scenario 1 ($R_s \geq 0.5$)

L'espacement (mesure de performance métrique) illustré dans le tableau V-4 indique que les solutions proposées par le front généré par le NSGA-III sont plus étalés dans l'espace de solutions. Les valeurs détaillées de l'espacement sont données en annexe

Tableau V-4 Métriques de mesure des performances dans le scénario 1.

	Nombre de solutions non-dominées	Spacing
NSGA-II	12	1.8503
NSGA-III	**18**	**0.5894**

V.5.2 Scénario 2

Pour le deuxième scénario, le problème est considéré comme un problème d'allocation de fiabilité-redondance. Comme il s'agit d'un problème mixte, le problème est plus complexe en termes de calcul. Les figures de V-7 à V-10 illustrent l'ensemble des solutions générées respectivement par le NSGA-II et NSGA-III.

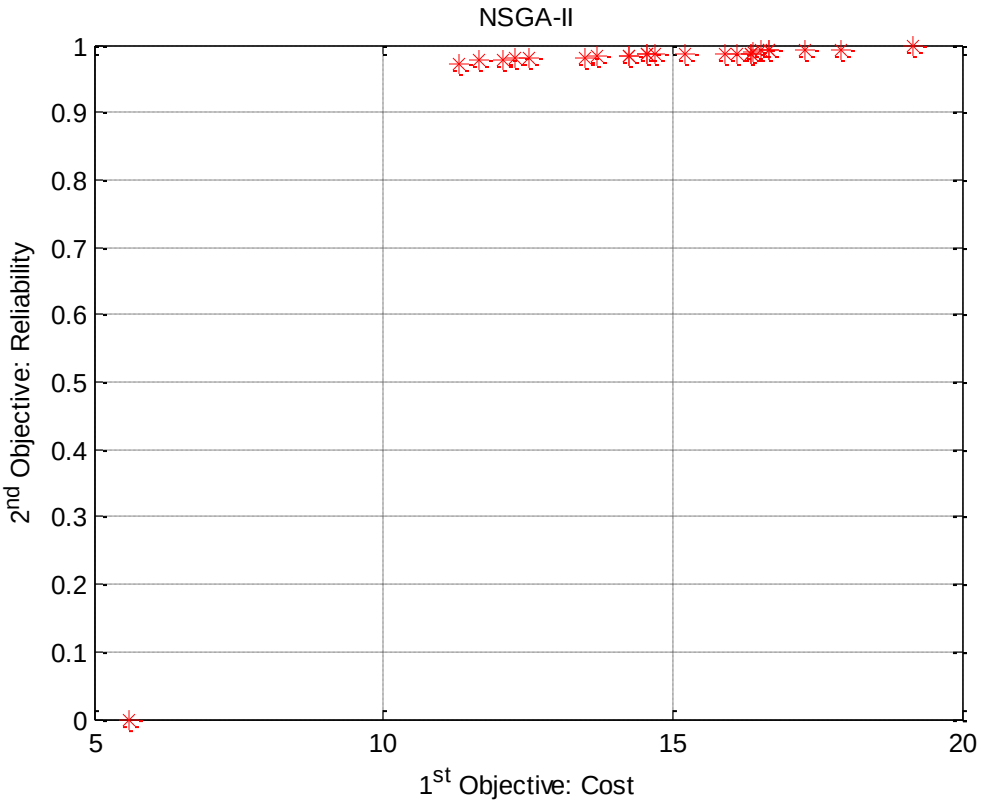

Figure V-7 Pareto Front de NSGA-II pour le scenario 2.

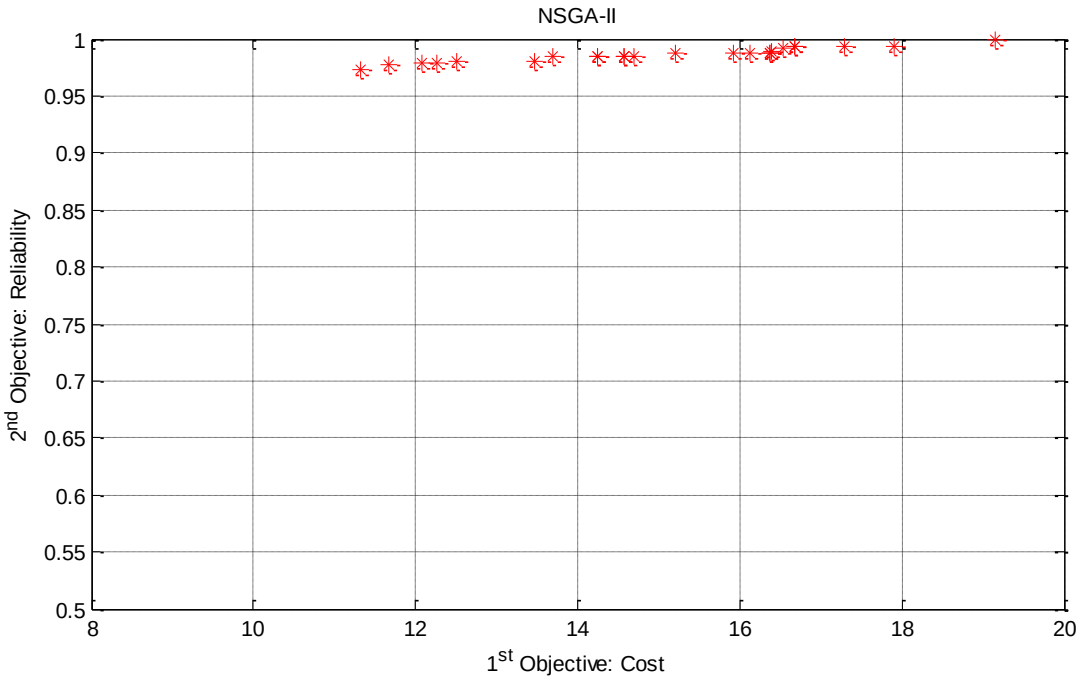

Figure V-8 Pareto Front de NSGA-II pour le scenario 2 ($R_s \geq 0.5$).

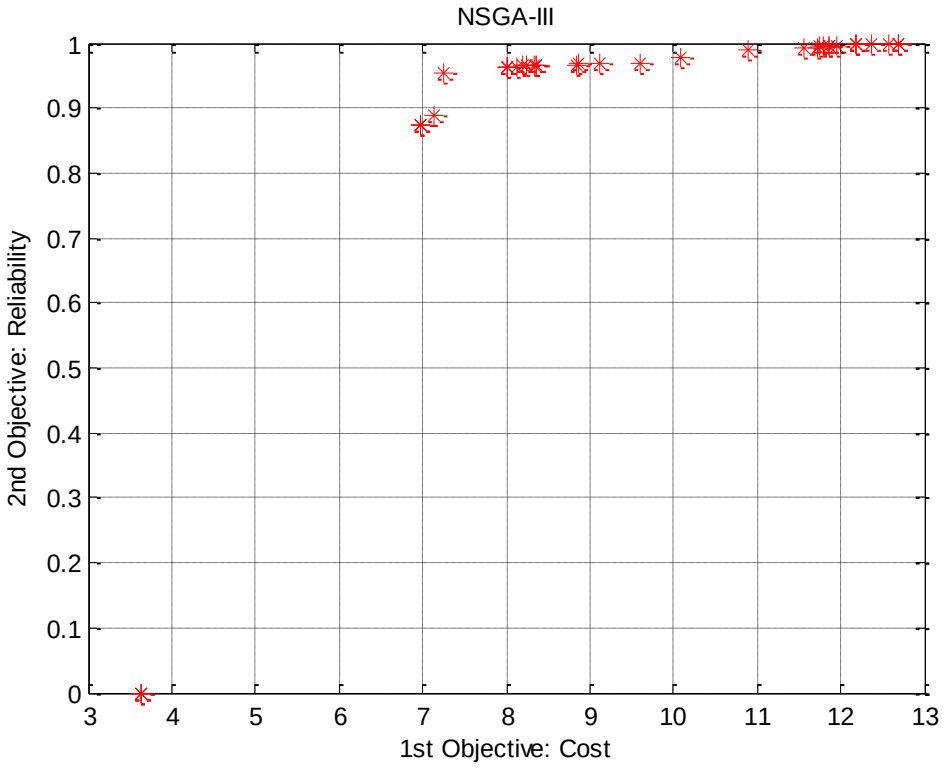

Figure V-9 Pareto Front de NSGA-III pour le scenario 2.

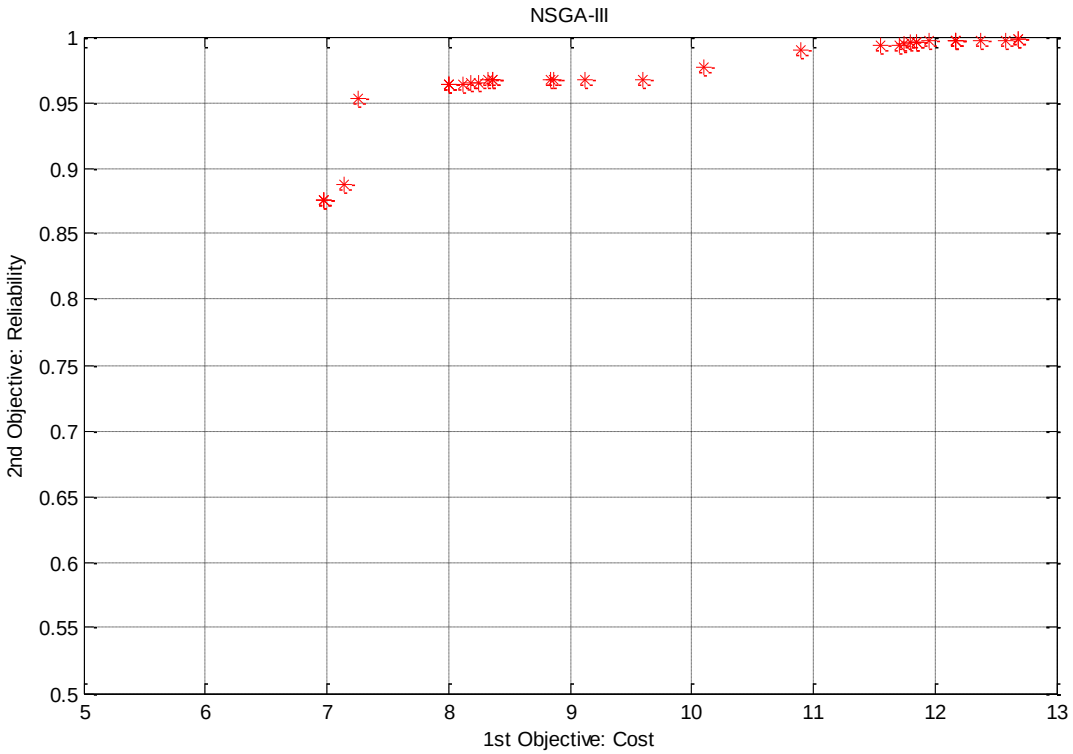

Figure V-10 Pareto Front de NSGA-III pour le scenario 2 ($R_s \geq 0.5$).

Comme dans le premier scénario, le NSGA-III offre un nombre plus important de solutions non-dominées et ces solutions sont plus dispersées dans l'espace des solutions que la solution offerte par le NSGA-II (voir tableau V-5).

Tableau V-5 Métriques de mesure des performances dans le scénario 2.

	Nombre de solutions non-dominées	Spacing
NSGA-II	21	1.4138
NSGA-III	**26**	**0.1142**

A partir des résultats ci-dessus, il est clairement possible d'observer que les solutions obtenues par l'approche d'allocation de fiabilité-redondance sont meilleures que celles offertes par l'approche d'allocation de redondance. En d'autres termes, pour la même fiabilité, les solutions du deuxième scénario sont moins chères que celles du premier scénario et ceci est valable dans les deux algorithmes. La conversion du problème d'optimisation considéré d'un problème d'allocation de redondance (RAP) à un problème d'allocation de fiabilité-redondance (RRAP) a conduit à passer d'un problème à huit variables de décision à un problème à seize variables de décision (mixtes, réelles et entières). Cette conversion rend le problème plus complexe à résoudre, mais augmente considérablement la taille de l'espace de recherche, ce qui explique les résultats supérieurs du deuxième scénario (RRAP) sur le premier (RAP), comme on le voit dans les tableaux V-4 et V-5. Pour le deuxième scénario, les solutions non-dominées générées par l'algorithme NSGA-III sont supérieures à celles générées par le NSGA-II. Le NSGA-III a atteint une fiabilité globale élevée avec un coût global inférieur par rapport à l'algorithme NSGA-II (voir figure V-11).

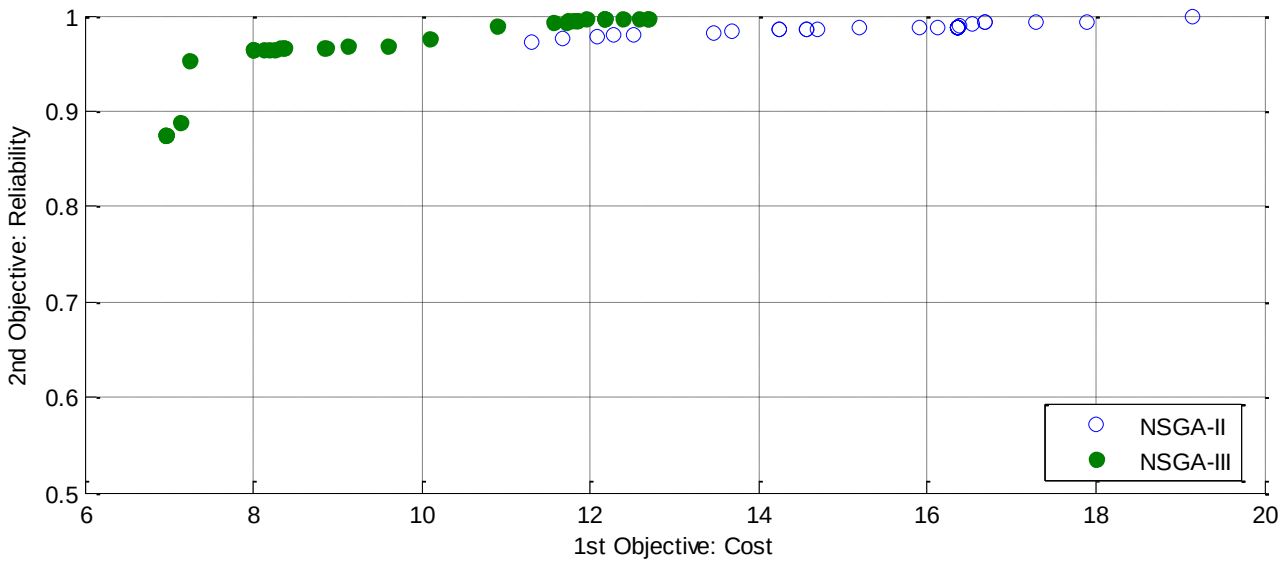

Figure V-11 Pareto Fronts de NSGA-II vs NSGA-III

Cela est dû à une différence notable entre les deux algorithmes dans l'opérateur de sélection, le NSGA-II utilise le *crowding distance*, où les solutions avec les plus grandes valeurs de *crowding distance* sont choisies ; par contre, le NSGA-III assure le maintien de la diversité par un ensemble de points de référence prédéterminés bien répartis qui sont mis à jour de manière adaptative. Chaque membre de la population doit être associé à un point de référence. Une ligne de référence

est définie pour chaque point de référence en reliant le point de référence à l'origine, puis, la distance entre chaque membre de la population et chaque ligne de référence est évaluée. Enfin, le point de référence avec la plus petite distance est associé au membre de la population ; plusieurs solutions peuvent être associées à un même point de référence, celle qui est la plus proche du point de référence est conservée (cette partie de l'algorithme est appelée opération de préservation de niche). Le temps d'exécution est généralement le même pour les deux algorithmes dans chaque scénario (voir les tableaux V-4 et V-5).

Une fois les algorithmes exécutés, de nombreuses solutions sont générées ; et le *k-means algorithme* est appliqué afin de regrouper les solutions originales contenues dans les fronts de Pareto en clusters (groupes). Les solutions d'un même cluster possèdent un degré maximal d'association, c'est-à-dire que les mêmes membres du cluster sont homogènes et très similaires les uns aux autres. Les solutions groupées pour l'algorithme NSGA-III sont illustrés dans la figure V-12.

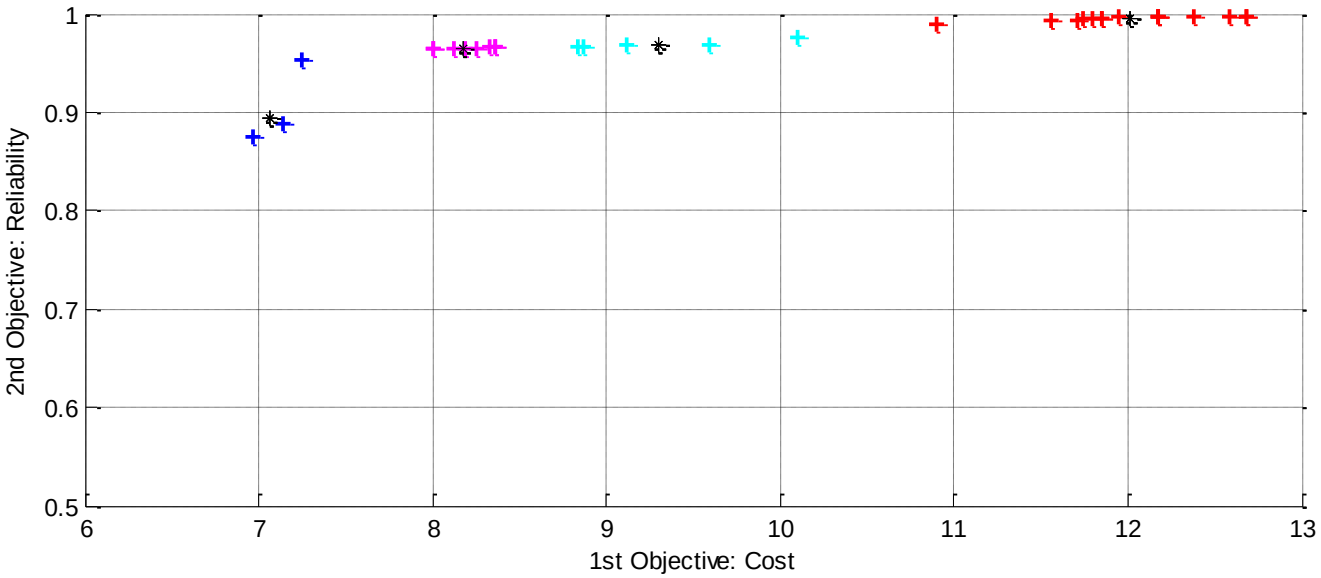

Figure V-12 Ensemble de Pareto en cluster.

Une fois le regroupement des solutions effectué par l'outil de clustering de données, le décideur dispose d'un plus petit nombre de solutions parmi lesquelles choisir. Le tableau V-6 résume les solutions obtenues par le *k-means algorithme*. Il contient le nombre de clusters, le nombre de solutions dans chaque cluster et sa solution représentative.

Tableau V- 6 Analyse de clustering.

Cluster	Nombre de solutions	Fiabilité	Coût
1	4	0.875180	6.968959
2	6	0.920390	7.198040
3	5	0.966732	8.581445
4	11	0.995803	12.013659

La solution la plus proche du centre de gravité du cluster est la solution représentative du cluster considéré. Afin de réduire encore plus le nombre de solutions, le décideur ne peut enquêter que sur le cluster qui contient les solutions les plus prometteuses. Ces solutions prometteuses sont situées dans la région du « genou » du Front de Pareto, où une petite amélioration de l'un des objectifs entraîne une baisse significative de l'autre.

V.6 Conclusions

Le but de ce chapitre était d'étudier le problème d'optimisation de la fiabilité des systèmes multi-objectif avec des paramètres flous. Une approche de solution basée sur quatre étapes a été proposée : appliquer la procédure de fonction de classement pour défuzzifier les paramètres flous en valeurs nettes, utiliser les NSGA-II et NSGA-III, identifier le meilleur front de Pareto par la méthode d'espacement, et enfin réduire le meilleur front de Pareto par l'analyse de clustering. Une étude de cas numérique traitée dans la littérature comme un problème d'allocation de redondance mono-objectif floue a été étudiée dans le présent chapitre en tant que problèmes d'allocation de redondance multi-objectif floue et de fiabilité-redondance. Il a été observé que l'allocation fiabilité-redondance donnait de meilleurs résultats et le NSGA-III a surpassé le NSGA-II dans la résolution de ce type de problème. L'analyse de clustering réduit la taille de l'ensemble de Pareto, ce qui aide le décideur à choisir la solution de compromis en fonction de ses attentes.

Les avantages de cette approche de solutions pour traiter l'optimisation de la fiabilité du système sont : elle prend en compte les données floues, gère les contraintes, compare rapidement

les Fronts de Pareto générés par les algorithmes d'optimisation et réduit la taille du meilleur front de Pareto.

Conclusion générale

Contributions et remarques finales

Plusieurs alternatives fonctionnelles et technologiques permettant de répondre au cahier des charges doivent être étudiées lors de la conception d'un système industriel. Chaque alternative est caractérisée par sa fiabilité, sa maintenabilité, sa productivité, etc. Une étude poussée doit donc être menée de manière à choisir les composants à utiliser pour respecter un ensemble de contraintes (fiabilité, cout, volume, poids, etc.) et ceci pour atteindre une conception optimale du système en question. De très nombreux travaux ont été menés dans cette direction. La diversité de structures et des options existantes pour l'amélioration de performances des systèmes, ont abouti à différentes classifications et modèles d'optimisation. Les systèmes sont généralement constitués de plusieurs sous-systèmes multi-composants.

Le but de cette thèse est d'étudier le problème d'optimisation de la fiabilité d'un système multi objectif avec des paramètres flous. Une approche de solution basée sur quatre étapes a été proposée. Un cas numérique d'étude a été traité dans le présent travail, Il a été observé que l'allocation fiabilité-redondance offrait un meilleur résultat et le NSGA-III a surpassé le NSGA-II dans la résolution de ce type de problème. L'analyse de clustering réduit la taille de l'ensemble de Pareto, qui aide le décideur à choisir la solution de compromis en fonction de ses attentes

Perspectives

Les travaux futurs seront consacrés à traiter les limites de l'approche proposée en termes de fiabilité du système. Évaluer la présence de défaillances corrélées communes (CCF) en tenant compte des défaillances simultanées des composants. Le CCF représente le comportement réel de la fiabilité du système d'une manière plus réaliste. Un autre point à considérer est la mesure de l'importance de chaque composant pour obtenir une meilleure fiabilité du système.

Bibliographie :

[1] A. Villemeur. Sûreté de fonctionnement des systèmes industriels : Fiabilité, Facteurs humains, Informatisation. Paris : Eyrolles, 1988.

[2] A. Pages and M. Gondran. Fiabilité des systèmes. Editions Eyrolles, 1980.

[3] L. M. Klyatis. Glossary of terms and definitions. Dans Accelerated reliability and durability testing technology, pages 375–391. John Wiley & Sons, New Jersey, USA, 2011.

[4] A. Chowdhury et D. O. Koval. Reliability principles. Dans Power distribution system reliability: Practical methods and applications, pages 45–77. John Wiley & Sons, New Jersey, USA, 2009.

[5] M. A. Levin et T. T. Kalal. Reliability concepts. Dans Improving product reliability : Strategies and implementation, pages 47–63. John Wiley & Sons, Chichester, United Kingdom, 2003.

[6] A. hoyland and M. Rausand. *System Reliability Theory*. JohnWiley&Sons, 1994.

[7] C.E. Ebeling. *An introduction to reliability and maintainability engineering.* McGRAW-HILL companies, Inc.,1997.

[8] L. M. Klyatis. Glossary of terms and definitions. Dans Accelerated reliability and durability testing technology, pages 375–391. John Wiley & Sons, New Jersey, USA, 2011.

[9] Avizienis, J. C. Villemeur, et B. Randell. Dependability and its threats: A taxonomy. Dans Proceedings of WCC'2004: 18th World Computer Congress, Toulouse, France, 2004.

[10] Villemeur. Sûreté de fonctionnement des systèmes industriels. Eyrolles, France, 1997.

[11] H. Pham. Basic statistical concepts. Dans *Springer handbook of engineering statistics*, pages 3–48. Springer London, London, United Kingdom, 2006.

[12] STAPELBERG, Rudolph Frederick. Handbook of reliability, availability, maintainability and safety in engineering design. Springer Science & Business Media, 2009.

[13] A. Chowdhury et D. O. Koval. Reliability principles. Dans Power distribution system reliability: Practical methods and applications, pages 45–77. John Wiley & Sons, New Jersey, USA, 2009.

[14] J.C. Laprie. Guide de la sureté de fonctionnement. Cepadues-Editions,1996.

[15] W. KuoandV.R.Prasad.Anannotatedoverviewofsystem-reliabilityoptimization. *IEEE Transactionson Reliability*, 49(2):176{187,Jun 2000.

[16] Yalaoui. Allocation de fiabilité et de redondance dans les systèmes parallèles-série et série-parallèle. PhD thesis, Université de Technologies de Troyes, 2004.

[17] FA. Tillman, CL. Hwang, and W. Kuo. Optimization techniques for system reliability with redundancy-a review. *IEEE Transactions on Reliability*, R-26(3):148{155, 1977.

[18] Yalaoui, E. Chatelet, and C. Chu. A new dynamic programming method for reliability redundancy allocation in a parallel-series system. *IEEE Transactions on Reliability*, 54(2):254{261, June 2005.

[19] Z. Tian, M. J.Zuo, and H. Huang. Reliability-redundancy allocation for multi-state series-parallel systems. *IEEE Transactions on Reliability*, 57(2):303-310, June 2008.

[20] Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge Univeristy Press, Cambridge (2004)

[21] Yang, X.S.: Engineering Optimization: An Introduction with Metaheuristic Applications. Wiley, Hoboken, NJ (2010).

[22] Yang, X.S.: Nature-Inspired Optimization Algorithms. Elsevier Insight, London (2014)

[23] BENNIS, Fouad et BHATTACHARJYA, Rajib Kumar (ed.). Nature-Inspired Methods for Metaheuristics Optimization: Algorithms and Applications in Science and Engineering. Springer Nature, 2020.

[24] Singiresu S. Rao, Engineering Optimization, Theory and Practice, 4th edition, John Wiley & Sons, 2009.

[25] VASUKI, A. Nature-Inspired Optimization Algorithms. CRC Press, 2020.

[26] ANILE, Angelo Marcello, CUTELLO, Vincenzo, NICOSIA, Giuseppe, et al. Comparison among evolutionary algorithms and classical optimization methods for circuit design problems. In : 2005 IEEE Congress on Evolutionary Computation. IEEE, 2005. p. 765-772.

[27] Jon C. Nash, The (Dantzig) simplex method for linear programming, IEEE Computing in Science and Engineering, Vol. 2, No. 1, pp. 29–31, January/February 2000.

[28] Richard Bellman, The theory of dynamic programming, P-550, Presented to the American Mathematical Society, Wyoming, July 1954.

[29] J. E. Mitchell, Branch-and-cut algorithms for integer programming, In: Encyclopedia of Optimization, C. A. Floudas and P. M. Pardalos (eds). Dordrecht, The Netherlands: Kluwer, 2001.

[30] R. E. Gomory, Outline of an algorithm for integer solutions to linear programs, Bulletin of American Mathematical Society, Vol. 64, No. 5, pp. 275–278, 1958.

[31] Iztok Fister Jr., Xin-She Yang, Iztok Fister, Janez Brest, Dusan Fister, A brief review of nature-inspired algorithms for optimization, Elektrotehniški Vestnik, Vol. 80, No. 3, pp. 1–7, July 2013.

[32] HOLLAND, John H. Hidden order: How adaptation builds complexity. Addison Wesley Longman Publishing Co., Inc., 1996.

[33] Petr Bujok, Josef Tvrdik, Radka Polakova, Comparison of nature-inspired population-based algorithms on continuous optimization problems, Swarm and Evolutionary Computation (Elsevier), Vol. 50, Article ID 100490, November 2019.

[34] Xin-She Yang (Ed.), Nature inspired algorithms and applied optimization, Studies in Computational Intelligence (Springer), 2018.

[35] Xin-She Yang, Nature inspired metaheuristic algorithms: Success and new challenges, Journal of Computer Engineering and Information Technology, Vol. 1, No. 1, pp. 1–3, November 2012.

[36] Zhihua Cui, Rajan Alex, Rajendra Akerkar, Xin-She Yang, Recent advances on bioinspired computation, The Scientific World Journal, Hindawi Publishing Corporation, Vol. 2014, Article ID 934890, May 2014.

[37] Scott McCaulay, Biologically inspired computing algorithms: Relevance and implications for research technologies, Indiana University, Bloomington, IN. PTI Technical Report PTI-TR12-003, February 2012.

[38] Xin-She Yang, Nature-Inspired Metaheuristic Algorithms, 2nd edition, Luniver Press, 2010.

[39] Xin-She Yang, Nature Inspired Optimization Algorithms, 1st edition, Elsevier, London, 2014.

[40] Xin-She Yang, Swarm intelligence based algorithms: A critical analysis, Evolutionary Intelligence (Springer), Vol. 7, No. 1, pp. 17–28, April 2014.

[41] Xin-She Yang, Suash Deb, Simon Fong, Xingshi He, Yu-Xin Zhao, From swarm intelligence to metaheuristics: Nature inspired optimization algorithms, IEEE Computer, Vol. 49, No. 9, pp. 52–59, September 2016.

[42] A. Hanif Halim, I. Ismail, Bio-inspired optimization method: A review, NNGT Journal: International Journal of Information Systems, Vol. 1, pp. 12–17, July 2014.

[43] Xin-She Yang, Su Fong Chien, Tiew On Ting, Computational intelligence and metaheuristic algorithms with applications, The Scientific World Journal, Hindawi Publishing Corporation, Vol. 2014, Article ID 425853, December 2014.

[44] Michael A. Lones, Metaheuristics in nature inspired algorithms, Proceedings of Genetic and Evolutionary Computation Conference (GECCO Comp '14), Vancouver, BC, Canada, pp. 1419–1422, July 2014.

[45] Xin-She Yang, Suash Deb, Simon Fong, Metaheuristic algorithms: Optimal balance of intensification and diversification, Applied Mathematics and Information Sciences, An International Journal, Vol. 8, No. 3, pp. 977–983, May 2014.

[46] Xin-She Yang, Suash Deb, Thomas Hanne, Xingshi He, Attraction and diffusion in nature-inspired optimization algorithms, Neural Computing and Applications, Vol. 31, No. 7, pp. 1987–1994, July 2019.

[47] D. H. Wolpert, W. G. Macready, No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1, pp. 67–82, April 1997.

[48] C. Schumacher, M. D. Vose, L. D. Whitley, The no free lunch and problem description length, Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation (GECCO '01), San Francisco, CA, United States, pp. 565–570, July 2001.

[49] Giorgos Karafotias, Mark Hoogendoorn, A. E. Eiben, Parameter control in evolutionary algorithms: Trends and challenges, IEEE Transactions on Evolutionary Computation, Vol. 19, No. 2, pp. 167–187, April 2015.

[50] T. Bäck. Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Oxford University Press, Oxford, UK (1996).

[51] C. A. C. Coello and G. B. Lamont D. A. V. Veldhuizen. Evolutionary algorithms for solving multi-objective problems, 2nd edition. Genetic and Evolutionary Computation. Springer US. (2007).

[52] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems; technical report 112. Technical report Computer Engineering and Networks

Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland. (2001).

[53] A. R. Pérez. Surrogate-assisted evolutionary multi-objective full model selection. Thèse de Doctorat, Instituto Nacional de Astrofísica, Óptica y Electrónica (2016).

[54] V. Pareto. Cours d'économie politique. Librairie Droz (1964).

[55] Z. Michalewicz and M. Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. Evolutionary Computation 4(1), 1–32 (1996).

[56] C. A. C. Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Computer Methods in Applied Mechanics and Engineering 191(11), 1245–1287 (2002).

[57] R. Courant. Variational methods for the solution of problems of equilibrium and vibrations. Bulletin of the American Mathematical Sociaty 49(1), 1–23 (1946).

[58] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. Nsga-ii: A fast and elitist multi-objective genetic algorithm. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)

[59] C-L. Hwang and A. S. M. Masud. , 62. Lecture Notes in Economics and Mathematical Systems, Springer Science & Business Media (2012).

[60] David A. Van Veldhuizen and Gary B. Lamont. Multiobjective evolutionary algorithm research: A history and analysis, (1998).

[61] J. J. Durillo and A. J. Nebro. jMetal: A Java framework for multi-objective optimization. Advances in Engineering Software 42(10), 760–771 (2011).

[62] J. R. Schott. Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization. Massachusetts Institute of Technology, Department of Aeronautics and Astronautics (1995).

[63] E. Zitzler. Evolutionary algorithms for multiobjective optimization: methods and applications. Doctoral dissertation ETH 13398, Swiss federal institue of technology (ETH). Zurich, Switzerland 1498 (1998).

[64] BEUME, Nicola. S-metric calculation by considering dominated hyper-volume as Klee's measure problem. Evolutionary Computation, 2009, vol. 17, no 4, p. 477-492.

[65] Soltani R. Reliability optimization of binary state non-repairable systems: a state of the artsurvey .Int J Ind Eng Comput2014; 5:339–64.

[66] CHEBOUBA, Billal Nazim, MELLAL, Mohamed Arezki, et ADJERID, Smail. Fuzzy multiobjective system reliability optimization by genetic algorithms and clustering analysis. Quality and Reliability Engineering International, 2021, vol. 37, no 4, p. 1484-1503.

[67] CHOPARD, Bastien et TOMASSINI, Marco. An introduction to metaheuristics for optimization. Springer International Publishing, 2018.

[68] Xu Z, Kuo W, Lin H H. Optimization limits in improving system reliability. IEEE TransReliab1990;39(1):51–60.

[69] Kuo W, Wan R. Recent advances in optimal reliability allocation. IEEE Trans Syst, Man Cybern A: SystHum2007;37(2):143–56.

[70] Krishna GJ, Ravi V .Modified harmony search applied to reliability optimization of complex systems. Harmony Search Algorithm. Germany: Springer Berlin Heidelberg; 2016.p.169–80.

[71] Hsieh YC, Chen TC, Bricker DL. Genetic algorithms for reliability design problems. Microelectron Reliab1998;38 (10):1599–605.

[72] Hsieh YC, You PS. An effective immune based two-phase approach for the optimal reliability–redundancy allocation problem. Appl Math Comput 2011;218 (4):1297–307.

[73] James Kennedy, Russell Eberhart, Particle swarm optimization, Proceedings of the IEEE International Conference on Neural Networks, Piscataway, NJ, Vol. IV, pp. 1942–1948. IEEE Press, 1995.

[74] Mahamed G. H. Omran, Particle swarm optimization methods for pattern recognition and image processing. Ph.D. thesis. University of Pretoria, Pretoria, 2005.

[75] Frans van den Bergh, An analysis of particle swarm optimizers. Ph.D. thesis. University of Pretoria, Pretoria, 2001.

[76] James Blondin, Particle swarm optimization: A tutorial, September 2009. http: //cs. armst rong. edu/s aad/c sci81 00/ps o_tut orial .pdf

[77] James Kennedy, Russell Eberhart, Yuhui Shi, Swarm Intelligence, Morgan Kaufmann Publishers, San Francisco, 2001.

[78] Reynolds, C. W., Flocks, herds, and schools: A distributed behavioral model, Computer Graphics, Vol. 21, No. 4, SIGGRAPH '87 Conference Proceedings, pp. 25–34, 1987.

[79] Millonas M. M, Swarms, Phase transitions and collective intelligence, In: Artificial Life III, C. G. Langton (ed). Reading, MA: Addison Wesley, pp. 417–443, 1993.

[80] M. Clerc, J. Kennedy, The particle swarm - Explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation, Vol. 6, No. 1, pp. 58–73, 2002.

[81] E.-G. Talbi, Metaheuristics: From Design to Implementation, vol. 74, John Wiley & Sons, 2009.

[82] X.-S. Yang, S. Deb, Engineering optimisation by cuckoo search, Int. J. Math. Modell. Numer. Optim. 1 (4) (2010) 330–343.

[83] B.B. Mandelbrot, The Fractal Geometry of Nature, Times Books, 1983.

[84] T. Vicˇzek, Fractal Growth Phenomena, World Scientific Publishing Company Incorporated, 1992.

[85] M.F. Barnsley, S. Demko, Iterated function systems and the global construction of fractals, Proc. Roy. Soc. Lond. A. Math. Phys. Sci. 399 (1817) (1985) 243–275.

[86] P. Grassberger, I. Procaccia, Characterization of strange attractors, Phys. Rev. Lett. 50 (5) (1983) 346–349.

[87] P. Prusinkiewicz, Graphical applications of L-systems, in: Proceedings of Graphics Interface, 1986.

[88] J. Cannon, W. Floyd, W. Parry, Finite subdivision rules, Conformal Geom. Dynam. Am. Math. Soc. 5 (8) (2001) 153–196.

[89] K.J. Falconer, Random fractals, Math. Proc. Cambridge Philos. Soc 100 (3) (1986) 559–582.

[90] T.t. Witten, L. Sander, Diffusion-limited aggregation, Phys. Rev. B 27 (9) (1983) 5686.

[91] L. Niemeyer, L. Pietronero, H. Wiesmann, Fractal dimension of dielectric breakdown, Phys. Rev. Lett. 52 (12) (1984) 1033–1036.

[92] X.-S. Yang, S. Deb, Engineering optimisation by cuckoo search, International Journal of Mathematical Modelling and Numerical Optimisation, Vol. 1, No. 4, pp. 330–343, 2010.

[93] X.-S. Yang, S. Deb, Cuckoo search via Lévy flights, In: Proceedings of World Congress on Nature and Biologically Inspired Computing (NaBIC 2009)Figu, Coimbatore, India, pp. 210– 14, December 2009, published by IEEE, USA, ISBN: 978-1-4244-5053-4.

[94] Iztok Fister Jr., Dusan Fister, Iztok Fister, A comprehensive review of cuckoo search: Variants and hybrids, International Journal of Mathematical Modelling and Numerical Optimisation, Vol. 4, No. 4, pp. 387–409, 2013.

[95] MELLAL, Mohamed Arezki et WILLIAMS, Edward J. Cuckoo optimization algorithm with penalty function and binary approach for combined heat and power economic dispatch problem. *Energy Reports*, 2020, vol. 6, p. 2720-2723.

[96] Garg H, Sharma SP. Reliability–redundancy allocation problem of pharmaceutical plant. J Eng Sci Technol 2013;8 (2):190–8.

[97] B. N. Chebouba, M. A. Mellal, S. Adjerid, Three computational intelligence methods for system reliability, *The 2nd International Workshop on Signal Processing Applied to Rotating Machinery Diagnostics* SIGPROMD'2018 29-30 April 2018, Djelfa, Algeria

[98] METTAS, Adamantios. Reliability allocation and optimization for complex systems. In: *Annual Reliability and Maintainability Symposium. 2000 Proceedings. International Symposium on Product Quality and Integrity (Cat. No. 00CH37055)*. IEEE, 2000. p. 216-221.

[99] YEH, Wei-Chang et HSIEH, Tsung-Jung. Solving reliability redundancy allocation problems using an artificial bee colony algorithm. *Computers & Operations Research*, 2011, vol. 38, no 11, p. 1465-1473.

[100] GONZALEZ, Teofilo F. (ed.). *Handbook of Approximation Algorithms and Metaheuristics: Contemporary and Emerging Applications, Volume 2*. CRC Press, 2018.

[101] MALIK, Hasmat. Metaheuristic and Evolutionary Computation: Algorithms and Applications. Springer Nature, 2020.

[102] N. Srinivas and K. Deb, "Multiobjective function optimization using nondominated sorting genetic algorithms," Evol. Comput., vol. 2, no. 3, pp. 221–248, Fall 1995.

[103] K. Deb and D. E. Goldberg, "An investigation of niche and species formation in genetic function optimization," in Proceedings of the Third International Conference on Genetic Algorithms, J. D. Schaffer, Ed. San Mateo, CA: Morgan Kauffman, 1989, pp. 42–50.

[104] DEB, Kalyanmoy et JAIN, Himanshu. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. IEEE transactions on evolutionary computation, 2013, vol. 18, no 4, p. 577-601.

[105] DEB, Kalyanmoy, PRATAP, Amrit, AGARWAL, Sameer, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE transactions on evolutionary computation, 2002, vol. 6, no 2, p. 182-197.

[106] Garg H. An efficient biogeography based optimization algorithm for solving reliability optimization problems. Swarm Evolut Comput 2015; 24:1–10

[107] CHEBOUBA, Billal Nazim, MELLAL, Mohamed Arezki, et ADJERID, Smail. Multi-objective system reliability optimization in a power plant. In : 2018 International Conference on Electrical Sciences and Technologies in Maghreb (CISTEM). IEEE, 2018. p. 1-4.

[108] CHEBOUBA, Billal Nazim, MELLAL, Mohamed Arezki, ADJERID, Smaïl, et al. Multi-objective System Reliability-redundancy Allocation in a Power Plant by Considering Three Targets. In: 2020 7th International Conference on Control, Decision and Information Technologies (CoDIT). IEEE, 2020. p. 674-678.

[109] CHEBOUBA, Billal Nazim, MELLAL, Mohamed Arezki, ADJERID, Smail, et al. System reliability and cost optimization under various scenarios using NSGA-III. In : 2020 International Conference on Electrical Engineering (ICEE). IEEE, 2020. p. 1-6.

[110] CHEBOUBA, Billal Nazim, MELLAL, Mohamed Arezki, ADJERID, Smail, et al. System design optimization under constraint of reliability, International Conference on Advanced Mechanics and Renewable Energies CIMAER'2018 28 et 29 Novembre 2018, Boumerdes, Algeria.

[111] Garg H. An efficient biogeography based optimization algorithm for solving reliability optimization problems. *Swarm Evol Comput*.2015; 24:1-10.

[112] Soltani R. Reliability optimization of binary state non-repairable systems: a state of the art survey. *Int J Ind Eng Comput*. 2014; 5:339-364.

[113] KuoW, Prasad V, Tillman F, Hwang C. *Optimal Reliability Design: Fundamentals and Applications*. New York: University Press.

[114] Mellal MA, Zio E. A penalty guided stochastic fractal search approach for system reliability optimization. *Reliab Eng Syst Saf*. 2016; 152:213-227.

[115] Mellal MA, Zio E. System reliability-redundancy allocation by evolutionary computation. In *2017 2nd International Conference on System Reliability and Safety*. Milan, Italy: IEEE; 2017:15-19.

[116] Mellal MA, Williams EJ. Large scale reliability-redundancy allocation optimization problem using three soft computing methods. Ram M, *Modeling and Simulation Based Analysis in Reliability Engineering*. FL, USA: CRC Press Francis & Taylor; 2018:199-214.

[117] Zio E. Reliability engineering: old problems and new challenges. *Reliab Eng Syst Saf*. 2009; 94:125-141.

[118] Zhang E, Chen Q. Multi-objective reliability redundancy allocation in an interval environment using particle swarm optimization. *Reliab Eng Syst Saf*. 2016; 145:83-92.

[119] 119- Xiao Jian Y, Shi J, Cheng J. Reliability technology using GO methodology: a review. *Qual Reliab Eng Int*. 2019; 35:2513-2539.

[120] Mellal MA, Zio E, Williams EJ. Cost minimization of repairable systems subject to availability constraints using efficient cuckoo optimization algorithm. *Qual Reliab Eng Int*. 2020;36 (3):1098- 110.

[121] Chebouba BN, Mellal MA , Adjerid S , Benazzouz D . System reliability and cost optimization under various scenarios using NSGA-III. *2020 International Conference on Electrical Engineering*. Istanbul, Turkey: 2020.

[122] Chern MS. On the computational complexity of reliability redundancy allocation in a series system. *Oper Res Lett*. 1992; 11:309-315.

[123] Kim H, Kim P. Reliability–redundancy allocation problem considering optimal redundancy strategy using parallel genetic algorithm. *Reliab Eng Syst Saf*. 2017; 159:153-160.

[124] Zhang E,Wu Y, Chen Q. A practical approach for solving multi-objective reliability redundancy allocation problems using extended barebones particle swarm optimization. *Reliab Eng Syst Saf*. 2014; 127:65-76.

[125] Chiun Ming L. Fuzzy programming and data envelopment analysis for improving redundancy-reliability allocation problems in series parallel systems. *Int J Phys Sci*. 2013; 8:635-646.

[126] Fonseca CM, Fleming PJ. Genetic algorithms formultiobjective optimization: formulation, discussion and generalization. In *Genetic Algorithms - Fifth International Conference*. SanMateo, CA, USA; 1993.

[127] Horn J, Nafpliotis N, Goldberg DE. A niched Pareto genetic algorithm formultiobjective optimization. In *IEEE Conference on Evolutionary Computation*. Orlando, FL, USA; 1994.

[128] Jain H, Deb K. An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, Part II: handling constraints and extending to an adaptive approach. *IEEE Trans Evol Comput*. 2014; 18:602-622.

[129] Srinivas N, Deb K. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evol Comput*. 1994; 2:221-248.

[130] 130- Zitzler E, Thiele L. An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach. TIK-Report, Zurich, Switzerland; 1998.

[131] Corne DW, Knowles J, Oates M. The Pareto envelope-based selection algorithm for multiobjective optimization. In *Sixth International Conference on Parallel Problem Solving from Nature*. Paris, France: Springer; 2000:839-848.

[132] Van Veldhuizen DA, Lamont GB. Multiobjective optimization with messy genetic algorithms. In *Proceedings of the ACM Symposium on Applied Computing*. Como, Italy; 2000.

[133] Zitzler E, Laumanns M, Thiele L. SPEA2: improving the strength Pareto evolutionary algorithm. In *EUROGEN'2001: Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*. Athens, Greece; 2001.

[134] Corne DW, Jerram N, Knowles J, Oates M, Martin J. PESA-II: region-based selection in evolutionary multiobjective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*. San Francisco, CA. 2001:283-290.

[135] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans Evol Comput*. 2002; 6(2):182-197.

[136] Zhang Q, Lizhang H. MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Trans Evol Comput*. 2007; 11(6):712- 731.

[137] Wang Z, Tianshi C, Tang K, Yao X. A multi-objective approach to redundancy allocation problem in parallel-series systems. In *2009 IEEE Congress on Evolutionary Computation*. CEC; 2009.

[138] Chebouba BN, Mellal MA, Adjerid S.Multi-objective system reliability optimization in a power plant. In *2018 International Conference on Electrical Sciences and Technologies in Maghreb*. Algiers, Algeria; 2018.

[139] Sharif M, Guilani PP, Shahriari M. Using NSGA II algorithm for a three objectives redundancy allocation problem with *k*-out-of-*n* subsystems. *Optim Ind Eng*. 2016; 9(19):87-96.

[140] Mellal MA, Zio E. An adaptive particle swarm optimization method for multi-objective system reliability optimization. *J Risk Reliab*. 2019; 233(6):990-1001.

[141] Lü H, Yang K, Huang X, Yin H, Shangguan WB, Yu D. An efficient approach for the design optimization of dual uncertain structures involving fuzzy random variables. *Comput Methods Appl Mech Eng*. 2020; 371:113331.

[142] Lü H, Yang K, Huang X, Yin H. Design optimization of hybrid uncertain structures with fuzzy-boundary interval variables. *Int J Mech Mater Des*. 2020. https://doi.org/10.1007/s10999-020-09523-9

[143] Muhuri PK, Ashraf Z, Lohani QMD. Multi-objective reliability-redundancy allocation problem with interval type-2 fuzzy uncertainty. *IEEE Trans Fuzzy Syst*. 2017; 26:1339-1355.

[144] Gupta N, Haseen S, Bari A. Reliability optimization problems with multiple constraints under fuzziness. *J Ind Eng Int*. 2016; 12(4):459-467.

[145] Taboada HA, Baheranwala F, Coit DW, Wattanapongsakorn N. Practical solutions for multi-objective optimization: an application to system reliability design problems. *Reliab Eng Syst Saf*. 2007; 92(3):314-322.

[146] Li Z, Liao H, Coit DW. A two-stage approach for multi-objective decision making with applications to system reliability optimization. *Reliab Eng Syst Saf*. 2009; 94(10):1585-1592.

[147] Taboada HA, Espiritu JF, Coit DW. MOMS-GA: a multi-objective multi-state genetic algorithm for system reliability optimization design problems. *IEEE Trans Reliab*. 2008; 57:182-191.

[148] Pakhira MK. A linear time-complexity *k*-means algorithm using cluster shifting. In *Proceedings — 2014 6th International Conference on Computational Intelligence and Communication Networks, CICN 2014*. Institute of Electrical and Electronics Engineers Inc.; 2014:1047-1051.

[149] Kumar A, Kaur J. A new method for solving fuzzy linear programs with trapezoidal fuzzy numbers. *J Fuzzy Set Valued Anal*. 2011;2011:1-12.

[150] Mellal MA, Williams EJ. Optimal replacement strategy of obsolete industrial components under fuzzy data. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*. 2020;234(3):349–357.

[151] Rao RV, Rai DP. Optimization of fused deposition modeling process using teaching-learning-based optimization algorithm. *Eng Sci Technol*. 2016;19:587-603.

[152] Liu J, Liu J. Applying multi-objective ant colony optimization algorithm for solving the unequal area facility layout problems. *Appl Soft Comput*. 2019;74:167-189.

[153] Bosse S, Splieth M, Turowski K. Multi-objective optimization of IT service availability and costs. *Reliab Eng Syst Saf*. 2016;147:142-155.

[154] Mellal MA, Pecht M. A multi-objective design optimization framework for wind turbines under altitude consideration. *Energy Convers Manag*. 2020; 222:113212.

[155] Kaufman L, Rousseeuw PJ. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley; 2005.

[156] Rahman MA, Islam MZ. A hybrid clustering technique combining a novel genetic algorithm with K-Means. *Knowledge Based Syst*. 2014; 71:345-365.

[157] Zio E, Bazzo R. A comparison of methods for selecting preferred solutions in multiobjective decision making. In: Kahraman C, *Computational Intelligence Systems in Industrial Engineering. Atlantis Computational Intelligence Systems*. Springer; 2012.

[158] Zio E, Bazzo R. A clustering procedure for reducing the number of representative solutions in the Pareto front of multiobjective optimization problems. *Eur J Oper Res*. 2011; 210:624-634.

[159] Bector CR, Chandra S. *Fuzzy Mathematical Programming and Fuzzy Matrix Games*. Springer-V. Germany; 2005.

[160] Sharma U, Aggarwal S. Solving fully fuzzy multi-objective linear programming problem using nearest interval approximation of fuzzy number and interval programming. *Int J Fuzzy Syst*. 2018; 20:488-499.

Annexe 1 :

Complexité de calcul de la solution proposé

Soit *N* la taille de la population (taille des données d'entrée) et *M* le nombre d'objectifs :

- **Procédure de défuzzification** : la procédure de défuzzification est appliquée pour convertir les données de système floues en données nettes. Les l'algorithme parcourt la longueur de la taille des données d'entrée ; par conséquent, le temps requis pour accomplir cette tâche est proportionnel à la taille des données d'entrée. En conséquence, cette partie du cadre a une complexité de calcul de *O(N)*.
- **Génération des fronts de Pareto** : le tri rapide non dominé est la partie la plus complexe en termes de calcul des deux algorithmes NSGAII et NSGA-III. Pour trouver si une solution est dominée, cette dernière est comparée à toutes les autres solutions de la population qui nécessite O(NM). Afin de trouver tous les membres du premier niveau non dominé dans la population, $O(N^2M)$ calculs sont nécessaires. À ce stade, tous les individus du premier front sont trouvés. Pour trouver les individus dans le prochain front non dominé, les étapes précédentes sont répétées, ce qui nécessite également des calculs $O(N^2M)$ pour chaque front. En conséquence, la complexité de calcul de cette partie de solution est $O(N^2M)$ dans les pires scénarios.
- **Meilleur front de Pareto** : La sélection du meilleur front de Pareto se fait par calcul des mesures des métriques de performance de chaque algorithme et les comparer ; par conséquent, la complexité de calcul de cette partie de la solution est proportionnelle à la taille des solutions non dominées générées. En conséquence, la complexité de cette partie est O(N).
- **Analyse de clustering** : la complexité de calcul de cette partie de la solution proposé est également proportionnelle à la taille des solutions générées. En conséquence, il a une complexité O(N) (ordre linéaire).

Annexe 2 : les valeurs du *Spacing metric*

Scénario 1 :

NSGA-II

0	20.847753	5.173196	15.316249	5.440659	11.364607	8.772169	9.319437	6.635043	12.875402	13.966661	14.940454
20.847753	0	15.674556	5.531503	15.407093	9.483145	12.075583	11.528315	14.212709	7.972350	6.881091	5.907298
5.173196	15.674556	0	10.143053	0.267462	6.191411	3.598972	4.146240	1.461846	7.702205	8.793464	9.767258
15.316249	5.531503	10.143053	0	9.875590	3.951642	6.544080	5.996812	8.681206	2.440847	1.349588	0.375794
5.440659	15.407093	0.267462	9.875590	0	5.923948	3.331510	3.878777	1.194384	7.434743	8.526002	9.499795
11.364607	9.483145	6.191411	3.951642	5.923948	0	2.592438	2.045170	4.729564	1.510794	2.602053	3.575847
8.772169	12.075583	3.598972	6.544080	3.331510	2.592438	0	0.547267	2.137125	4.103233	5.194492	6.168285
9.319437	11.528315	4.146240	5.996812	3.878777	2.045170	0.547267	0	2.684393	3.555965	4.647224	5.621017
6.635043	14.212709	1.461846	8.681206	1.194384	4.729564	2.137125	2.684393	0	6.240359	7.331618	8.305411
12.875402	7.972350	7.702205	2.440847	7.434743	1.510794	4.103233	3.555965	6.240359	0	1.091258	2.065052
13.966661	6.881091	8.793464	1.349588	8.526002	2.602053	5.194492	4.647224	7.331618	1.091258	0	0.973793
14.940454	5.907298	9.767258	0.375794	9.499795	3.575847	6.168285	5.621017	8.305411	2.065052	0.973793	0

NSGA-III

0	1.241901	7.714554	7.689358	1.238848	4.041322	6.346164	4.476448	5.263194	0.248486	3.433324	2.220058	12.912083	0.758326	2.270782	7.009969	2.381235	0.931416
1.241901	0	6.472653	8.931259	2.480749	5.283223	7.588065	3.234547	4.021293	0.993414	4.675225	3.461959	14.153984	2.000227	1.028880	8.251870	3.623136	0.310484
7.714554	6.472653	0	15.403912	8.953402	11.755876	14.060719	3.238105	2.451360	7.466067	11.147878	9.934612	20.626637	8.472880	5.443772	14.724523	10.095790	6.783137
7.689358	8.931259	15.403912	0	6.450509	3.648035	1.343193	12.165807	12.952552	7.937844	4.256033	5.469299	5.222725	6.931031	9.96014	0.679388	5.308122	8.620774
1.238848	2.480749	8.953402	6.450509	0	2.802474	5.107316	5.715297	6.502042	1.487334	2.194475	0.981209	11.673235	0.480521	3.509630	5.771120	1.142387	2.170264
4.041322	5.283223	11.755876	3.648035	2.802474	0	2.304842	8.517771	9.304516	4.289808	0.607998	1.821264	8.870760	3.282995	6.312104	2.968646	1.660086	4.972738
6.346164	7.588065	14.060719	1.343193	5.107316	2.304842	0	10.822613	11.609359	6.594651	2.912840	4.126106	6.565918	5.587838	8.616946	0.663804	3.964929	7.277581
4.476448	3.234547	3.238105	12.165807	5.715297	8.517771	10.822613	0	0.786745	4.227962	7.909773	6.696507	17.388532	5.234775	2.205666	11.486418	6.857684	3.545032
5.263194	4.021293	2.451360	12.952552	6.502042	9.304516	11.609359	0.786745	0	5.014707	8.696518	7.483252	18.175277	6.021520	2.992412	12.273163	7.644430	4.331777
0.248486	0.993414	7.466067	7.937844	1.487334	4.289808	6.594651	4.227962	5.014707	0	3.681810	2.468544	13.160569	1.006812	2.022295	7.258455	2.629722	0.682929
3.433324	4.675225	11.147878	4.256033	2.194475	0.607998	2.912840	7.909773	8.696518	3.681810	0	1.213265	9.478759	2.674997	5.704106	3.576644	1.052088	4.364740
2.220058	3.461959	9.934612	5.469299	0.981209	1.821264	4.126106	6.696507	7.483252	2.468544	1.213265	0	10.692025	1.461731	4.490840	4.789910	0.161177	3.151474
12.912083	14.153984	20.626637	5.222725	11.673235	8.870760	6.565918	17.388532	18.175277	13.160569	9.478759	10.692025	0	12.153756	15.182865	5.902114	10.530847	13.843499
0.758326	2.000227	8.472880	6.931031	0.480521	3.282995	5.587838	5.234775	6.021520	1.006812	2.674997	1.461731	12.153756	0	3.029108	6.251642	1.622909	1.689742
2.270782	1.028880	5.443772	9.960140	3.509630	6.312104	8.616946	2.205666	2.992412	2.022295	5.704106	4.490840	15.182865	3.029108	0	9.280751	4.652017	1.339365
7.009969	8.251870	14.724523	0.679388	5.771120	2.968646	0.663804	11.486418	12.273163	7.258455	3.576644	4.789910	5.902114	6.251642	9.280751	0	4.628733	7.941385
2.381235	3.623136	10.095790	5.308122	1.142387	1.660086	3.964929	6.857684	7.644430	2.629722	1.052088	0.161177	10.530847	1.622909	4.652017	4.628733	0	3.312652
0.931416	0.310484	6.783137	8.620774	2.170264	4.972738	7.277581	3.545032	4.331777	0.682929	4.364740	3.151474	13.843499	1.689742	1.339365	7.941385	3.312652	0

Scénario 2:

NSGA-II

0	14.533579	7.870454	1.265699	3.971685	5.703516	1.874199	6.668438	3.256367	7.517448	5.480277	4.476393	7.098598	2.486642	4.932391	6.904385	3.040788	2.622905	2.777316	4.601721	2.806747
14.533579	0	6.663125	13.267880	10.561893	8.830062	12.659380	7.865141	11.277211	7.016131	9.053301	10.057186	7.434981	12.046937	9.601188	7.629194	11.492792	11.910674	11.756263	9.931858	11.726832
7.870454	6.663125	0	6.604755	3.898769	2.166938	5.996255	1.202015	4.614087	0.353005	2.390177	3.394060	0.771856	5.383812	2.938063	0.966069	4.829666	5.247549	5.093138	3.268733	5.063707
1.265699	13.267880	6.604755	0	2.705985	4.437817	0.608500	5.402739	1.990668	6.251749	4.214578	3.210694	5.832899	1.220942	3.666692	5.638686	1.775088	1.357205	1.511617	3.336021	1.541047
3.971685	10.561893	3.898769	2.705985	0	1.731831	2.097485	2.696753	0.715317	3.545763	1.508592	0.504708	3.126913	1.485043	0.960706	2.932700	0.930897	1.348780	1.194368	0.630035	1.164938
5.703516	8.830062	2.166938	4.437817	1.731831	0	3.829317	0.964922	2.447149	1.813932	0.223239	1.227122	1.395082	3.216874	0.771125	1.200869	2.662728	3.080611	2.926200	1.101796	2.896769
1.874199	12.659380	5.996255	0.608500	2.097485	3.829317	0	4.794239	1.382168	5.643249	3.606077	2.602194	5.224399	0.612442	3.058192	5.030186	1.166588	0.748705	0.903117	2.727521	0.932547
6.668438	7.865141	1.202015	5.402739	2.696753	0.964922	4.794239	0	3.412071	0.849010	1.188161	2.192045	0.430159	4.181797	1.736047	0.235946	3.627650	4.045533	3.891122	2.066718	3.861691
3.256367	11.277211	4.614087	1.990668	0.715317	2.447149	1.382168	3.412071	0	4.261081	2.223909	1.220026	3.842231	0.769725	1.676024	3.648017	0.215579	0.633462	0.479051	1.345353	0.449620
7.517448	7.016131	0.353005	6.251749	3.545763	1.813932	5.643249	0.849010	4.261081	0	2.037171	3.041055	0.418850	5.030806	2.585057	0.613063	4.476660	4.894543	4.740132	2.915728	4.710702
5.480277	9.053301	2.390177	4.214578	1.508592	0.223239	3.606077	1.188161	2.223909	2.037171	0	1.003883	1.618321	2.993635	0.547886	1.424108	2.439489	2.857372	2.702961	0.878556	2.673530
4.476393	10.057186	3.394060	3.210694	0.504708	1.227122	2.602194	2.192045	1.220026	3.041055	1.003883	0	2.622204	1.989751	0.455997	2.427991	1.435605	1.853488	1.699077	0.125327	1.669646
7.098598	7.434981	0.771856	5.832899	3.126913	1.395082	5.224399	0.430159	3.842231	0.418850	1.618321	2.622204	0	4.611956	2.166207	0.194213	4.057810	4.475693	4.321282	2.496877	4.291851
2.486642	12.046937	5.383812	1.220942	1.485043	3.216874	0.612442	4.181797	0.769725	5.030806	2.993635	1.989751	4.611956	0	2.445749	4.417743	0.554146	0.136263	0.290674	2.115078	0.320105
4.932391	9.601188	2.938063	3.666692	0.960706	0.771125	3.058192	1.736047	1.676024	2.585057	0.547886	0.455997	2.166207	2.445749	0	1.971994	1.891603	2.309486	2.155075	0.330670	2.125644
6.904385	7.629194	0.966069	5.638686	2.932700	1.20086	5.030186	0.235946	3.648017	0.613063	1.424108	2.427991	0.194213	4.417743	1.971994	0	3.863597	4.281480	4.127069	2.302664	4.097638
3.040788	11.492792	4.829666	1.775088	0.930897	2.662728	1.166588	3.627650	0.215579	4.476660	2.439489	1.435605	4.057810	0.554146	1.891603	3.863597	0	0.417882	0.263471	1.560932	0.234041
2.622905	11.910674	5.247549	1.357205	1.348780	3.080611	0.748705	4.045533	0.633462	4.894543	2.857372	1.853488	4.475693	0.136263	2.309486	4.281480	0.417882	0	0.154411	1.978815	0.183841
2.777316	11.756263	5.093138	1.511617	1.194368	2.926200	0.903117	3.891122	0.479051	4.740132	2.702961	1.699077	4.321282	0.290674	2.155075	4.127069	0.263471	0.154411	0	1.824404	0.029430
4.601721	9.931858	3.268733	3.336021	0.630035	1.101796	2.727521	2.066718	1.345353	2.915728	0.878556	0.125327	2.496877	2.115078	0.330670	2.302664	1.560932	1.978815	1.824404	0	1.794973
2.806747	11.726832	5.063707	1.541047	1.164938	2.896769	0.932547	3.861691	0.449620	4.710702	2.673530	1.669646	4.291851	0.320105	2.125644	4.097638	0.234041	0.183841	0.029430	1.794973	0

NSGA-III

0	0.505732	0.881994	4.460063	10.053923	4.386400	4.704208	1.783770	5.832573	1.130681	0.972550	4.345072	3.116406	0.096768	3.845916	0.826802	0.731820	0.935326	2.605877	5.472035	3.871791	4.593652	3.589684	5.644530	4.526936	0.305049
0.505732	0	0.376262	3.954331	9.548191	3.880668	4.198476	1.278038	5.326841	0.624949	0.466818	3.839340	2.610674	0.408963	3.340184	0.321070	0.226088	0.429593	2.100144	4.966303	3.366059	4.087920	3.083951	5.138798	4.021204	0.200683
0.881994	0.376262	0	3.578069	9.171928	3.504405	3.822213	0.901775	4.950579	0.248687	0.090556	3.463077	2.234411	0.785225	2.963921	0.055192	0.150174	0.053331	1.723882	4.590041	2.989797	3.711657	2.707689	4.762535	3.644942	0.576945
4.460063	3.954331	3.578069	0	5.593859	0.073663	0.244144	2.676293	1.372510	3.329382	3.487513	0.114991	1.343657	4.363294	0.614147	3.633261	3.728243	3.524737	1.854186	1.011972	0.588272	0.133588	0.870379	1.184466	0.066873	4.155014
10.053923	9.548191	9.171928	5.593859	0	5.667523	5.349715	8.270153	4.221349	8.923241	9.081372	5.708851	6.937517	9.957154	6.208006	9.227121	9.322103	9.118597	7.448046	4.581887	6.182131	5.460271	6.464239	4.409393	5.526986	9.748874
4.386400	3.880668	3.504405	0.073663	5.667523	0	0.317808	2.602630	1.446174	3.255718	3.413849	0.041327	1.269994	4.289631	0.540483	3.559598	3.654580	3.451074	1.780523	1.085636	0.514608	0.207252	0.796716	1.258130	0.140536	4.081351
4.704208	4.198476	3.822213	0.244144	5.349715	0.317808	0	2.920438	1.128365	3.573526	3.731657	0.359136	1.587802	4.607439	0.858291	3.877406	3.972388	3.768882	2.098331	0.767827	0.832416	0.110556	1.114524	0.940322	0.177271	4.399159
1.783770	1.278038	0.901775	2.676293	8.270153	2.602630	2.920438	0	4.048804	0.653088	0.811219	2.561302	1.332636	1.687001	2.062146	0.956968	1.051950	0.848444	0.822106	3.688265	2.088021	2.809882	1.805914	3.860760	2.743166	1.478721
5.832573	5.326841	4.950579	1.372510	4.221349	1.446174	1.128365	4.048804	0	4.701892	4.860023	1.487501	2.716168	5.735805	1.986657	5.005771	5.100754	4.897248	3.226697	0.360538	1.960782	1.238922	2.242890	0.188043	1.305637	5.527524
1.130681	0.624949	0.248687	3.329382	8.923241	3.255718	3.573526	0.653088	4.701892	0	0.158131	3.214390	1.985724	1.033912	2.715234	0.303879	0.398861	0.195355	1.475195	4.341354	2.741110	3.462970	2.459002	4.513848	3.396255	0.825632
0.972550	0.466818	0.090556	3.487513	9.081372	3.413849	3.731657	0.811219	4.860023	0.158131	0	3.372521	2.143855	0.875782	2.873365	0.145748	0.240730	0.037224	1.633326	4.499485	2.899241	3.621101	2.617133	4.671979	3.554386	0.667501
4.345072	3.839340	3.463077	0.114991	5.708851	0.041327	0.359136	2.561302	1.487501	3.214390	3.372521	0	1.228666	4.248303	0.499155	3.518270	3.613252	3.409746	1.739195	1.126963	0.473280	0.248579	0.755388	1.299458	0.181864	4.040023
3.116406	2.610674	2.234411	1.343657	6.937517	1.269994	1.587802	1.332636	2.716168	1.985724	2.143855	1.228666	0	3.019637	0.729510	2.289604	2.384586	2.181080	0.510529	2.355629	0.755385	1.477245	0.473277	2.528124	1.410530	2.811357
0.096768	0.408963	0.785225	4.363294	9.957154	4.289631	4.607439	1.687001	5.735805	1.033912	0.875782	4.248303	3.019637	0	3.749147	0.730033	0.635051	0.838557	2.509108	5.375267	3.775023	4.496883	3.492915	5.547761	4.430168	0.208280
3.845916	3.340184	2.963921	0.614147	6.208006	0.540483	0.858291	2.062146	1.986657	2.715234	2.873365	0.499155	0.729510	3.749147	0	3.019114	3.114096	2.910590	1.240039	1.626119	0.025875	0.747735	0.256232	1.798613	0.681020	3.540867
0.826802	0.321070	0.055192	3.633261	9.227121	3.559598	3.877406	0.956968	5.005771	0.303879	0.145748	3.518270	2.289604	0.730033	3.019114	0	0.094982	0.108524	1.779075	4.645233	3.044989	3.766850	2.762882	4.817728	3.700134	0.521752
0.731820	0.226088	0.150174	3.728243	9.322103	3.654580	3.972388	1.051950	5.100754	0.398861	0.240730	3.613252	2.384586	0.635051	3.114096	0.094982	0	0.203506	1.874057	4.740215	3.139971	3.861832	2.857864	4.912710	3.795116	0.426770
0.935326	0.429593	0.053331	3.524737	9.118597	3.451074	3.768882	0.848444	4.897248	0.195355	0.037224	3.409746	2.181080	0.838557	2.910590	0.108524	0.203506	0	1.670551	4.536709	2.936465	3.658326	2.654358	4.709204	3.591610	0.630276
2.605877	2.100144	1.723882	1.854186	7.448046	1.780523	2.098331	0.822106	3.226697	1.475195	1.633326	1.739195	0.510529	2.509108	1.240039	1.779075	1.874057	1.670551	0	2.866159	1.2659150	1.987775	0.983807	3.038653	1.921060	2.300827
5.472035	4.966303	4.590041	1.011972	4.581887	1.085636	0.767827	3.688265	0.360538	4.341354	4.499485	1.126963	2.355629	5.375267	1.626119	4.645233	4.740215	4.536709	2.866159	0	1.600244	0.878384	1.882352	0.1724945	0.945099	5.166986
3.871791	3.366059	2.989797	0.588272	6.182131	0.514608	0.832416	2.088021	1.960782	2.741110	2.899241	0.473280	0.755385	3.775023	0.025875	3.044989	3.139971	2.936465	1.265915	1.600244	0	0.721860	0.282107	1.772738	0.655145	3.566742
4.593652	4.087920	3.711657	0.133588	5.460271	0.207252	0.110556	2.809882	1.238922	3.462970	3.621101	0.248579	1.477245	4.496883	0.747735	3.766850	3.861832	3.658326	1.987775	0.878384	0.721860	0	1.003968	1.050878	0.066715	4.288602
3.589684	3.083951	2.707689	0.870379	6.464239	0.796716	1.114524	1.805914	2.242890	2.459002	2.617133	0.755388	0.473277	3.492915	0.256232	2.762882	2.857864	2.654358	0.983807	1.882352	0.282107	1.003968	0	2.054846	0.937252	3.284634
5.644530	5.138798	4.762535	1.184466	4.409393	1.258130	0.940322	3.860760	0.188043	4.513848	4.671979	1.299458	2.528124	5.547761	1.798613	4.817728	4.912710	4.709204	3.038653	0.172494	1.772738	1.050878	2.054846	0	1.117593	5.339481
4.526936	4.021204	3.644942	0.066873	5.526986	0.140536	0.177271	2.743166	1.305637	3.396255	3.554386	0.181864	1.410530	4.430168	0.681020	3.700134	3.795116	3.591610	1.921060	0.945099	0.655145	0.066715	0.937252	1.117593	0	4.221887
0.305049	0.200683	0.576945	4.155014	9.748874	4.081351	4.399159	1.478721	5.527524	0.825632	0.667501	4.040023	2.811357	0.208280	3.540867	0.521752	0.426779	0.630276	2.300827	5.166986	3.566742	4.288602	3.284634	5.339481	4.221887	0