N° Ordre...../DGM/FSI/UMBB/2019

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE M'HAMED BOUGARA - BOUMERDES

Faculté des Sciences de l'Ingénieur
Département Génie Mécanique

Laboratoire de Mécanique des Solides et Systèmes

Mémoire de Master

En vue de l'obtention du diplôme de **MASTER** en **:**

Filière : Electromécanique
Spécialité : Mécatronique

THEME

Optimisation de la fiabilité des systèmes en phase de conception

Présenté par :

Frik Abir

Boutiche Roumaissa

Promoteur : Dr. M. A. Mellal

Promotion : 2018-2019

Résumé

La concurrence féroce dans l'industrie force les industriels à s'acquitter des systèmes conçus à être le plus fiable possible. L'objectif de ce travail est d'optimiser la conception d'un système réseau en pont complexe et un système de protection de survitesse (turbine à gaz) via l'allocation de fiabilité et l'allocation de fiabilité-redondance, sous les contraintes de conception. Les problèmes sont résolus à l'aide de deux algorithmes bio-inspirés de l'intelligence artificielle : Gray Wolf Optimizer (GWO) et Shuffled Frog-Leaping Algorithm (SFLA). Une fonction de pénalité est implémentée afin de maîtriser les contraintes et les résultats obtenus comparent les performances des deux algorithmes.

Mots clés : Fiabilité des systèmes, Conception, Algorithmes bio-inspirés, Intelligence artificielle.

Abstract

Fierce competitiveness in the industry pressures manufacturers to acquire systems designed to be as reliable as possible. The aim of the present work is to optimize the design of a complex bridge network system and an overspeed protection system through the reliability allocation and reliability-redundancy allocation, under the design constraints. The problems are solved by resorting to two bio-inspired algorithms of artificial intelligence: Gray Wolf Optimizer (GWO) and Shuffled Frog-Leaping Algorithm (SFLA). A penalty function is implemented in order to handle the constraints and the results obtained compare the performances of both algorithms.

Keywords: System reliability, Design, Bio-inspired algorithms, Artificial intelligence.

ملخص

المنافسة الشرسة في الصناعة تجبر المصنعين على تصنيع أنظمة مصممة لتكون موثوقة قدر الإمكان. الهدف من هذا العمل هو تحسين تصميم نظام شبكة جسر معقد ونظام حماية من السرعة الزائدة (التوربينات الغازية) من خلال تخصيص الموثوقية وتخصيص اعتمادية التكرار، في ظل قيود التصميم .يتم حل المشكلات باستخدام خوارزميات الذكاء الاصطناعي المستوحاة من الطبيعة: Gray Wolf Optimizer (GWO) و Shuffled Frog-Leaping Algorithm (SFLA). يتم تنفيذ وظيفة جزاء للسيطرة على القيود و النتائج المتحصل عليها تقارن بين أداء الخوارزميات.

الكلمات المفتاحية: موثوقية النظام ، التصميم ، الخوارزميات المستوحاة من الطبيعة ، الذكاء الاصطناعي.

Remerciements

Nous remercions tout d'abord Dieu le tout puissant de nous avoir donné le courage, la santé et la patience afin d'arriver à la finalité de ce parcours.

Ce travail n'aurait jamais vu le jour sans le soutien et les encouragements de notre promoteur Dr. M. A. Mellal. Pour cela, nous tenons à le remercier pour sa grande disponibilité dans la direction de ce mémoire, ses précieux conseils et sa grande patience.

A la fin, nous remercions toutes les personnes de prêt ou de loin qui ont contribué à la réussite de ce mémoire.

A nos chères mères

A nos chers pères

A tous nos proches et nos amis

Sommaire

Chapitre I : Sûreté de fonctionnement des systèmes

Chapitre II : Etat de l'art sur l'optimisation

Chapitre III : Optimisation de la fiabilité des systèmes

Sommaire

Liste des figures

Liste des tableaux

Liste des abréviations

ABC : Artificial Bee Colony

ACO : Ant Colony Optimization.

ADAP-PSO : An adaptive particle swarm optimization.

AdD : Arbre de Défaillance.

ALO : Ant Lion Optimizer.

AMDEC : Analyses des Modes de Défaillance, leurs Effets et leurs Criticité.

APR : Analyse Préliminaire des Risques.

COA : Cuckoo Optimization Algorithm.

CPU : Temps d'exécution.

CS : Cuckoo Search.

DF : Diagramme de Fiabilité.

FGO : Fuzzy Global Optimization.

FPA : Flower Pollination Algorithm.

INGHS : Improved Novel Global Harmony Search.

IPSO : Improved Particle Swarm Optimization Algorithm.

GA : Algorithme Génétique.

GWO : Gray Wolf Optimizer.

NAFSA : Novel Artificial Fish Swarm Algorithm.

NMDE : Novel Modified Differential Evolution.

MRPS : Modified Random-to-Pattern Search

MTTF : Mean Time To Failure.

MTTR : Mean Time To Repair.

MUT : Mean Up Time.

MDT : Mean Down Time.

MTBF : Mean Time Between Failure.

NFE : Nombre de Fonction d'Evaluation.

NMDE : Novel Modified Differential Evolution.

PSFS : A Penalty guided Stochastic Fractal Search.

PSO : Particle Swarm Optimization.

SdF : Sûreté de Fonctionnement.

SFLA : Shuffled Frog-Leaping Algorithm.

WOA : Whale Optimization Algorithm.

Introduction générale

Introduction générale

Le monde assiste à un grand développement industriel où le besoin du bon fonctionnement des installations des entreprises les conduits à une grande concurrence économique, dans laquelle ils doivent consacrer leurs efforts pour fiabiliser et optimiser leurs équipements afin d'augmenter la fiabilité des systèmes mécatroniques (la plupart des systèmes dans les zones industriels sont des systèmes mécatronique; car cette dernière inclue quatre disciplines : mécanique, électronique, automatique et informatique) ainsi que la durée de vie des outils d'une part, et améliorer la productivité d'une autre part.

Pour garantir la fiabilité des systèmes, il est primordial d'ajouter des composants redondants, et augmenter la fiabilité des outils, en utilisant trois méthodes d'optimisations :

- Allocation de fiabilité.
- Allocation de redondance.
- Allocation de fiabilité-redondance.

L'objectif de ce mémoire est d'optimiser la fiabilité des systèmes à l'aide de deux algorithmes bio-inspirés (intelligence artificielle), dont le but est de minimiser le coût, et maximiser la fiabilité.

Ce travail englobe trois chapitres dépendants, où le premier présente des généralités sur la sûreté de fonctionnement (SdF) des systèmes ainsi que les méthodes et les variations de la fiabilité.

Nous citons les différents types d'optimisation et leurs méthodes de résolution dans le deuxième chapitre.

Ainsi, ce mémoire traite deux problèmes d'optimisations : un système réseau en pont complexe et un système de protection de survitesse (turbine à gaz).

Chapitre I

Sûreté de fonctionnement des systèmes

I.1 Introduction

Le haut niveau de performance des systèmes nécessite une meilleure application de la sûreté de fonctionnement. Dans ce chapitre, nous allons illustrer un aperçu sur la SdF et ses méthodes, en se basant sur la fiabilité. Cette dernière est indispensable dans tous les domaines industriels, tels que la conception, la maintenance, les systèmes complexes et les systèmes mécatroniques qui exigent un fonctionnement fiable.

I.2 Sûreté de fonctionnement

La sûreté de fonctionnement est définie comme la science des défaillances [1]. C'est une des préoccupations majeures en ingénierie pour répondre aux exigences de l'industrie opérationnelles et réglementaires. Suivant la norme NF EN 13306 la sûreté de fonctionnement est l'aptitude à fonctionner comme cela est requis et lorsque cela est requis. Elle précise que la sûreté de fonctionnement comprend la disponibilité, la sûreté, la sécurité, la durabilité et les facteurs qui les influencent (la fiabilité, la maintenabilité, les performances de logistique de maintenance, les conditions d'utilisation et l'influence du personnel d'exploitation). Aussi, suivant la norme CEI 60050 la sûreté de fonctionnement est l'aptitude à fonctionner quand et tel que requis. La sûreté de fonctionnement comprend la disponibilité, la fiabilité, la maintenabilité, l'efficacité de la logistique de maintenance et, dans certains cas, d'autres caractéristiques telles que la durabilité, la sûreté et la sécurité [2].

I.3 Eléments constitutifs de la sûreté de fonctionnement

I.3.1 Fiabilité

La norme européenne NF EN 13306 définit la fiabilité R(t) comme « l'aptitude d'un bien à accomplir une fonction requise, dans des conditions données, durant un intervalle de temps donné ». Selon la norme IEC 60050-191 :1990 : «capacité d'une entité à réaliser une fonction requise dans des conditions données pour un nombre donné de cycles ou un intervalle de temps».

Elle est définie par : $R(t) = P(E)$ non défaillante sur la durée $[0, t]$, en supposant qu'elle n'est pas défaillante à l'instant $t = 0$ [3], où P est la probabilité est E l'unité.

Ce qui peut s'exprimer par :

$$R(t) = P(T > t) \quad \textbf{(I.1)}$$

D'après Zwingelstein [3], on distingue plusieurs types de fiabilité (termes spécifiques) :

- **Fiabilité opérationnelle** (observée ou estimée) déduite de l'analyse d'entités identiques dans les mêmes conditions opérationnelles à partir de l'exploitation d'un retour d'expérience.
- **Fiabilité prévisionnelle** (prédite) correspondant à la fiabilité future d'un système et établie par son analyse, connaissant les fiabilités de ses composants.
- **Fiabilité extrapolée** déduite de la fiabilité opérationnelle par extrapolation ou interpolation pour des conditions ou des durées différentes.
- **Fiabilité intrinsèque** ou inhérente qui découle directement des paramètres de conception. Sans modification de conception des entités, il n'est pas possible d'obtenir un niveau de fiabilité supérieur à la fiabilité intrinsèque.

I.3.2 Disponibilité

La norme NF EN 13306 définit la disponibilité comme « l'aptitude d'un bien à être en état d'accomplir une fonction lorsqu'elle est requise dans des conditions données, en supposant que les ressources externes nécessaires sont mises à disposition ». Cette aptitude dépend de la combinaison de la fiabilité et de la maintenabilité du bien, de l'aptitude au soutien, ainsi que des actions de maintenance exécutées sur le bien [3].

La probabilité associée A(t) à l'instant t est aussi appelée « disponibilité » et s'exprime par :

$$A(t) = P(E) \text{ } \textit{non défaillante à l'instant } t. \quad \textbf{(I.2)}$$

Comme la fiabilité, plusieurs types de disponibilités peuvent être utilisés :

• **Disponibilité instantanée prévisionnelle :** c'est la probabilité qu'un bien soit en état d'accomplir la fonction requise à un instant donné, en supposant que les ressources externes nécessaires sont mises à disposition.

• **Disponibilité moyenne :** c'est la moyenne sur un intervalle de temps donné [t_1, t_2] de la disponibilité instantanée prévisionnelle, ou mesurée en phase opérationnelle par la durée.

• **Disponibilité asymptotique :** elle se déduit du MUT et du MTBF par la relation :

$$A = \frac{MUT}{MTBF} \tag{I.3}$$

I.3.3 Maintenabilité

La maintenabilité se distingue de la fiabilité sur la promptitude de reprise du service attendu après interruption [4]. Elle est définit comme l'aptitude d'une entité à être maintenue ou rétablie dans un état dans lequel elle peut accomplir une fonction requise, lorsque la maintenance est réalisée dans des conditions données avec des procédures et des moyens prescrits [5]. Suivant la norme NF EN 13306, elle se définit ainsi : « Dans des conditions données d'utilisation, comme l'aptitude d'un bien à être maintenu ou rétabli dans un état où il peut accomplir une fonction requise, lorsque la maintenance est accomplie dans des conditions données, en utilisant des procédures et des moyens prescrits ». La maintenabilité d'une entité réparable est caractérisée par une probabilité $M(t)$ que la maintenance d'une entité E accomplie dans des conditions données, avec des moyens et des procédures prescrits, soit achevée au temps t, sachant que l'entité était en panne à l'instant 0 [3].

$$M(t) = P\left[E \text{ est réparée sur } [0,t]\right] \tag{I.4}$$

I.3.4 Sécurité

C'est l'aptitude d'une entité à ne pas causer de dommages dans des conditions données ou à ne pas faire apparaître, dans des conditions données, des événements critiques ou catastrophiques, Ou bien l'absence de risque intolérable. C'est une méthodologie qui permet d'analyser et d'éliminer des risques à caractère inacceptable qui pourraient engendrer des blessures, porter atteinte directement ou indirectement à la santé des personnes, dégrader l'environnement et altérer la propriété [3].

$$S(t) = P\left[E \text{ évite des événements critiques ou catastrophiques sur} [0,t]\right] \tag{I.5}$$

I.3.5 Autres éléments de la sûreté de fonctionnement

- **Résilience :** la norme IEC 2382-14 définit la résilience comme l'aptitude d'une unité fonctionnelle à continuer d'accomplir une fonction exigée en présence d'anomalies ou d'erreurs
- **Intégrité :** l'intégrité qui assure que les informations sont complètes et non altérées, en particulier par des actions externes malveillantes. La norme IEC 61375-3-1 a définit l'intégrité comme : « capacité d'un système à reconnaître et à éliminer des données erronées en cas de dysfonctionnement d'un quelconque de ses sous-ensembles ».
- **Coût :** la fiabilité peut être augmentée dans une certaine mesure pour tout produit utilisant des composants de qualité et des procédures de fabrication, d'inspection et de test font l'objet d'une surveillance. Le coût initial augmente, mais le coût d'exploitation diminue avec la fiabilité.
- **Durabilité :** est l'aptitude d'une entité à accomplir une fonction requise dans des conditions données d'utilisation et de maintenance, jusqu'à ce qu'un état limite soit atteint [6].

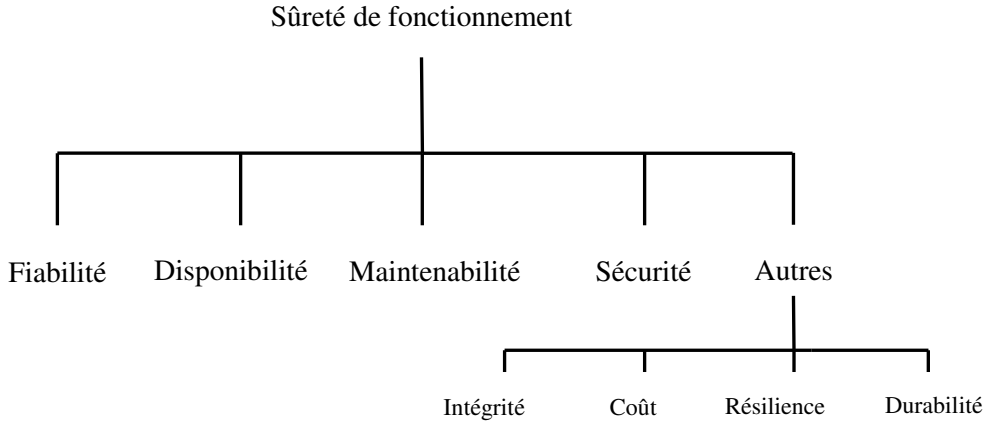

Figure I.1 : Eléments constitutifs de la sûreté de fonctionnement.

I.4 Méthodes d'analyse de la sureté de fonctionnement

La sûreté de fonctionnement est un regroupement des méthodes qui servent à identifier, analyser, gérer et réduire les risques liées aux systèmes. On peut classifier ces méthodes en deux classes [7] :

- **Approches qualitatives/quantitatives**
  - ✓ Qualitative : les résultats informent sur les caractéristiques du système.
  - ✓ Quantitative : les résultats de calcul de fiabilité, disponibilité...

- **Approches inductives/déductives**
  - ✓ Inductive : une démarche descendante, elle cherche à caractérisent sur le système et ses environnements en considérant un événement initiateur.
  - ✓ Déductive : une démarche ascendante, elle cherche à expliquer les causes d'un événement redouté.

Les principales méthodes d'analyse de la sûreté de fonctionnement sont les suivantes :

I.4.1 Analyse Préliminaire des Risques (APR)

Cette méthode inductive et qualitative apparue la première fois aux Etats-Unis au début des années soixante [6]. Elle consiste à résumer la maîtrise des risques par la SdF, où l'objectif est d'identifier les risques du système et leurs causes, ainsi que l'évaluation des accidents potentiels et la gravité des conséquences liées aux risques. Cette méthode est recommandée dès la première phase de conception [4].

L'APR est économique en termes de temps et ne demande pas trop de détails au niveau du système étudié par rapport à d'autres méthodes [8].

I.4.2 Analyse des Modes de Défaillance, leurs Effets et leurs Criticité (AMDEC)

L'AMDEC est une extension de l'AMDE (Analyse des modes de Défaillances et de leurs Effets). Méthode de type inductive et qualitative, et est la plus utilisée [4]. Elle sert à analyser les défaillances simples qui peuvent mener le système à une défaillance

globale. L'objectif est d'identifier les défaillances potentielles et les moyens de les prévenir ou de les atténuer. En revanche, dans le cas d'un système complexe l'AMDEC est difficile à maîtriser dû au grand nombre d'informations à traiter. Cette méthode n'est pas d'intérêt, sauf dans le cas où on peut avoir une conclusion [4], [5].

I.4.3 Arbre de Défaillance (AdD)

Cette méthode est une analyse déductive et quantitative qui représente les enchainements d'événement qui conduit le système à l'événement redouté en utilisant une structure arborescent, cette dernière se servie de tous les connecteurs logiques (ET, OU) inclusif et exclusif et leurs combinaisons. Comme l'AMDEC, l'AdD aussi dans le cas d'un système complexe il est difficile de tenir en compte les aspects temporels [4].

I.4.4 Diagramme de Fiabilité (DF)

Cette méthode de type inductive, appelée méthode de diagramme de succès, consiste à analyser et calculer la fiabilité et la disponibilité des systèmes. L'objectif est de représenter l'architecture du système sous des blocs série, parallèle, liés entre eux. Ce système fonctionne que dans le cas où cette chaîne ne soit pas rompue par certains blocs [4].

La DF peut être une analyse qualitative/quantitative, qualitative dans le cas d'une recherche sur les moyens de la réussite de la fonction récuse de système et les scénarios qui mènent à l'échec pour éviter les accidents. Quantitative dans le cas d'une identification la probabilité de bon fonctionnement du système (calcul de fiabilité, disponibilité,…) [5].

Le tableau I.1 représente la classification des méthodes de la sûreté de fonctionnement.

Tableau I.1 : Classification des méthodes de SdF.

Méthode de SdF	Inductive	Déductive	Qualitative	Quantitative
APR	✓		✓	
DF	✓		✓	✓
AMDEC	✓		✓	
AdD		✓		✓

I.5 Métriques de SdF

D'après [5], les grandeurs associées à la SdF peuvent être calculées à partir des mesures de probabilités. Les grandeurs suivantes caractérisent des durées moyennes (voir figure I.2) :

- MTTF : Durée moyenne de fonctionnement d'une entité avant la première défaillance (Mean time to failure).
- MTTR : Durée moyenne de réparation (anglais Mean Time To Repair).
- MUT : Durée moyenne de fonctionnement après réparation (Mean Up Time).
- MDT : Durée moyenne d'indisponibilité après défaillance (Mean Down Time).
- MTBF : Durée moyenne entre deux défaillances (Mean Time Between Failure).

$$MTBF = MDT + MUT \tag{I.6}$$

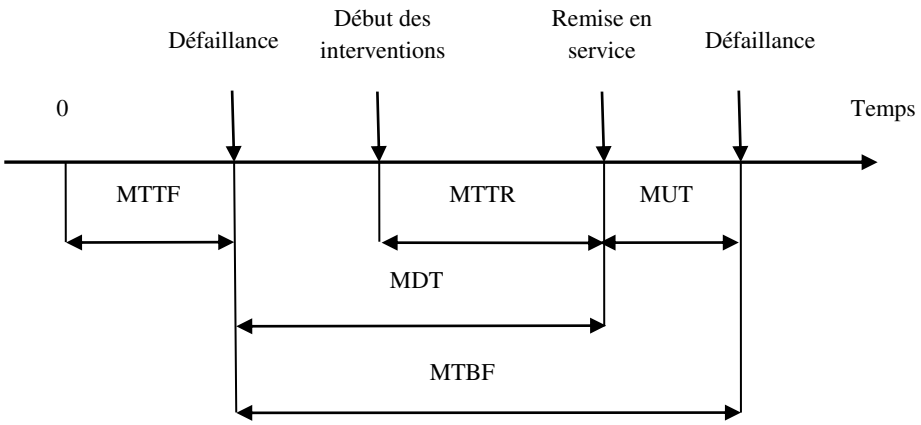

Figure I.2 : Grandeurs moyennes associées à la SdF.

I.6 Fiabilité des systèmes simples

I.6.1 Système série

La logique de fonctionnement d'un système série est que tous ses composants doivent fonctionner pour que le système fonctionne. Ce système est la seule configuration logique dans laquelle les composants avec des taux de défaillance constants induisent un taux de défaillance constant pour le système. La figure I.3 représente la forme d'un système série [9].

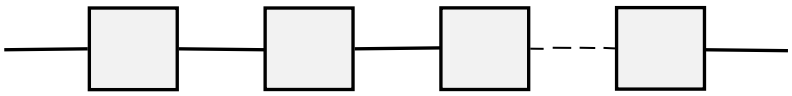

Figure I.3 : Système série.

En termes de probabilité pour que le système fonctionne, nous avons :

$$P = \prod_{i=1}^{N} p_i \tag{I.7}$$

Fiabilité du système :

$$R(t) = \prod_{i=1}^{N} R_i(t) \tag{I.8}$$

Pour les composants exponentiels, la fiabilité du système devient :

$$R(t) = e^{-\lambda t} < R_i(t) = e^{-\lambda_i t} \tag{I.9}$$

Avec :

$$\lambda = \sum_{i=1}^{N} \lambda_i = \textit{taux de défaillance du} \text{ système} \tag{I.10}$$

$$m = \frac{1}{\lambda} = \textit{temps moyen de défaillance} \text{ du système} \tag{I.11}$$

Le système tombe en panne à min (t_1, t_2, ..., t_N), où t_i est le temps de défaillance du composant *i*

I.6.2 Système parallèle

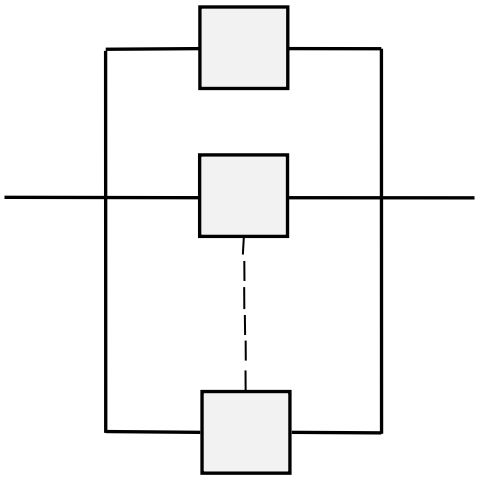

Figure I.4 : Système parallèle.

Pour que ce système fonctionne il suffit qu'un compassant fonctionne (voir figure I.4). Autrement dit, tous les composants font la même fonction afin que chacun puisse continuer l'opération avec succès [9].

En termes de probabilité de fonctionnement du système, nous avons :

$$p = 1 - \prod_{i=1}^{N} (1 - p_i) \tag{I.12}$$

Fiabilité du système :

$$R(t) = 1 - \prod_{i=1}^{N} \left[1 - R_i(t)\right] \tag{I.13}$$

Pour les composants exponentiels, la fiabilité du système devient :

$$R(t) = 1 - \prod_{i=1}^{N} \left[1 - e^{-\lambda_i t}\right] \tag{I.14}$$

Comme le système tombe en panne lorsque tous ses éléments tombent en panne, le temps de défaillance du système est max (t_1, t_2, ..., t_N).

I.7 Fiabilité et maintenance

I.7.1 Maintenance

D'après la norme ISO 9000, la maintenance est constituée d'un ensemble d'activités corrélées et coordonnées, qui utilisent des ressources et sont réalisées par des acteurs pour obtenir un résultat (actions techniques, administratives et de management) destinées à maintenir un bien ou à le rétablir dans un état dans lequel il peut accomplir la fonction requise.

I.7.2 Types de maintenance

I.7.2.1 Maintenance corrective

Selon la norme IEC 62278, une maintenance corrective est effectuée à la suite d'une détection de panne et destinée à remettre un produit dans un état lui permettant d'accomplir une fonction requise. On distinguer deux sortes de la maintenance corrective [10] :

- ✓ Maintenance palliative (dépannage).
- ✓ la Maintenance curative (réparation définitive).

I.7.2.2 Maintenance préventive

Selon la norme IEC 61918, la maintenance préventive est effectuée à intervalles prédéterminés ou selon des critères prescrits et destinée à réduire la probabilité de défaillance ou la dégradation du fonctionnement d'une entité. On peut distinguer 3 sortes de maintenances préventives [10] :

- ✓ Maintenance préventive systématique.
- ✓ Maintenance préventive conditionnelle.
- ✓ Maintenance préventive prévisionnelle.

I.7.3 Fiabilité en phase de la maintenance

L'état physique des matériels est influencé par la réparation corrective et la remise en état préventive et cela sert à améliorer le niveau de la fiabilité opérationnelle. On distingue quelques opérations qui améliorent la fiabilité en phase de maintenance, telles que [11] :

- **Remplacement systématique :** C'est un remplacement des composants ayant comme objectif de modéliser le matériel afin de réduire le taux de défaillance instantané, ces remplacements concernent que les matériels qui vieillissent et dont le taux de défaillance s'accroit au cours du temps.
- **Graissage :** c'est une tâche simple utilisée pour protéger le matériel contre la dégradation rapide. Il se ralentit l'accroissement de taux de défaillance au cours du temps par contre le graissage ne rajeunit pas le matériels.
- **Inspection externe :** cela s'exerce sans démonter les matériels et sans outillages spécial ce qui ne demande pas des frais (organiser des rondes).
- **Surveillance en fonctionnement :** ceci est comme l'analyse vibratoire et la thermographie, ce qui demande d'outillages particuliers de mesure et de l'analyse.
- **Contrôles :** ceci nécessite généralement le démontage du matériel, cela exige de le mettre hors service. Cette opération est coûteuse à cause de l'arrêt de production pondant les contrôles.
- **Réparations :** suite à défaillance qui appartiennent à la maintenance corrective.
- **Interventions préventives de remise en état :** c'est une action principale de la maintenance préventive conditionnelle. Elle permettre de réduire le taux de défaillance et remettre à neuf le matériels.
- **Essais :** sont des tests exercer sur les matériels en attente et sur tous, les appareils de protection et les matériels en redondance passive. Pour le but de garantir la disponibilité en détectant des pannes cachées.
- **Epreuves :** qui permettent de constater que l'on dispose de marges au-delà du régime de fonctionnement nominal.

I.8 Fiabilité en phase de conception

La fiabilité est tellement sensible à la conception, où toute une modification au niveau de la conception des composants peut causer de profonds changements de la fiabilité. C'est pourquoi il est important de spécifier la fiabilité du produit et les objectifs de maintenabilité avant tout travail de conception. La fiabilité en phase de conception est basée sur deux éléments principaux :

- ✓ Matériaux.
- ✓ Redondance.

I.8.1 Matériaux

Il est important en conception de bien choisir les matériaux pour optimiser la fiabilité. Ce choix est basé essentiellement sur l'application à laquelle ils sont destinés et le procédé grâce auquel ils seront mis en œuvre. On distingue plusieurs caractéristiques des matériaux comme [12]:

- ✓ Caractéristiques intrinsèques.
- ✓ Caractéristiques interactives.
- ✓ Caractéristiques attribuées.

Il existe des méthodes pour la sélection des matériaux. Cette procédure systématique doit être suffisamment générale pour pouvoir être appliquée à des conceptions très variées.

I.8.2 Redondance

La redondance joue un grand rôle dans l'augmentation de la fiabilité en phase de conception grâce à l'utilisation des composants supplémentaires en parallèle afin d'amélioré l'état de fonctionnement des systèmes. L'utilisation de la redondance dans la conception possède plusieurs avantages [13] :

- Tout degré de fiabilité souhaité peut être atteint.
- Une technique qui fournit une solution rapide.

I.9 Fiabilité du système mécatronique

Avant d'aborder la fiabilité, nous allons définir un système mécatronique. L'apparition de ce système au Japon en 1969 a fait une révolution dans le monde industriel qui est un domaine à la croisée de la mécanique, l'électronique, l'automatique et de l'informatique en temps réel. Le but de cette action est de synergie et améliorer les fonctionnalités d'un produit [14].

I.9.1 Mesures associées à la fiabilité

Les principes les mesures associées à la fiabilité sont comme suit :

- ✓ Densité de la probabilité.
- ✓ Taux de défaillance instantané.

I.9.2 Densité de probabilité

La densité de défaillance $f(t)$ de l'instant de la défaillance t s'obtient en dérivant la fonction de répartition $F(t)$ [3].

$$\int_0^{\infty} f(t)\, dt = 1 \tag{I.15}$$

$$F(t) = \int_0^{t} f(u)\, du \tag{I.16}$$

Ce terme est appelé « densité de probabilité de défaillance ». C'est une grandeur toujours positive pour $t > 0$.

Les principales lois de probabilité rencontrées dans les études de la fiabilité sont [3] :

- **Loi binomiale de paramètres *p* et *n* :** Cette loi concerne des unités d'usage discret. La variable aléatoire discrète prend des valeurs entières entre 0 et n avec une probabilité :

$$P(X = i) = C_n^i p^i (1-p)^{n-i} \tag{I.17}$$

Avec :

$$p + q = 1 \tag{I.18}$$

La valeur moyenne m et la variance var sont données par :

$$m = np \;\; et \;\; var(X) = npq \tag{I.19}$$

- **Loi de Poisson de paramètre m :** Cette loi concerne des unités d'usage discrètes. La variable aléatoire discrète prend des valeurs entières entre 0 et $+\infty$ avec une probabilité :

$$P(X = k) = \frac{m^k}{k!} exp(-m) \tag{I.20}$$

La valeur moyenne m et la variance sont données par :

$$var(X) = m \tag{I.21}$$

- **Loi exponentielle de paramètre constant λ :** Cette loi s'applique de façon privilégiée pour représenter les lois de défaillance de petits composants électroniques et électriques, La variable aléatoire est dans ce cas une variable continue t entre [0 et $+\infty$] dont la loi de densité de probabilité et donnée par :

$$f(t) = \lambda exp(-\lambda t) \tag{I.22}$$

- **Loi de Weibull :** Cette loi, développée par l'ingénieur suédois W. Weibull pour l'étude des défaillances métallurgiques de systèmes mécaniques, est très utilisée pour représenter le comportement des matériels (en particulier mécanique). La loi de densité de probabilité de défaillance est définie pour la loi de Weibull à trois paramètres :

$$f(t) = \frac{\beta(t-\gamma)^{\beta-1}}{\eta} exp-\left(\frac{(t-\gamma)}{\eta}\right)^{\beta} \tag{I.23}$$

Avec :

β, η, γ sont les paramètre de : forme, échelle et de position (sans unité).

La valeur moyenne et la variance sont données par :

$$m = \gamma + \eta\Gamma\left(\frac{1+\beta}{\beta}\right) \tag{I.24}$$

Et :

$$var(t) = \eta^2\left[\Gamma\left(\frac{2}{\beta}+1\right) - \Gamma^2\left(\frac{1+\beta}{\beta}\right)\right] \tag{I.25}$$

Avec :

$$\Gamma(b) = \int_0^{\infty} x^{b-1} exp(-x)\,dx \tag{I.26}$$

- **Loi log-normale ou loi de Galton** : Cette loi a été étudiée par *F*. Galton et est utilisée en sûreté de fonctionnement pour la modélisation de phénomène d'usure et pour la modélisation des problèmes de temps de réparation. La variable aléatoire est dans ce cas une variable continue *t* entre [0 et $+\infty$] dont la loi de densité de probabilité est donnée par :

$$f(t) = \frac{1}{\sigma\sqrt{2\pi}} \frac{1}{t} exp \frac{\left(ln(t) - \mu\right)^2}{2\sigma^2} \quad \text{(I.27)}$$

Avec :

μ est la moyenne des $ln(t)$ et σ est écart-type des $ln(t)$.

La valeur moyenne m et la variance sont données par :

$$m = exp\left(\mu + \frac{\sigma^2}{2}\right) \quad \text{(I.28)}$$

Et :

$$var(t) = \left(exp\left(\sigma^2\right) - 1\right) exp\left(2\mu + \sigma^2\right) \quad \text{(I.29)}$$

- **Loi normale** : La variable aléatoire est dans ce cas une variable continue t entre [-∞ et +∞] dont la loi de densité de probabilité est donnée par :

$$f(t) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{\left(t - \mu\right)^2}{2\sigma^2}\right) \quad \text{(I.30)}$$

La valeur moyenne et la variance sont données par :

$$m = \mu \; et \; var(t) = \sigma^2 \quad \text{(I.31)}$$

- **Autres lois utilisées dans la densité de la probabilité :**
  - Loi Gamma.
  - Loi Gamma généralisée.
  - Loi de Gompertz.
  - Loi Log-logistic.
  - Loi de Cauchy.
  - Loi inverse de Gauss.
  - Loi de Pareto.

I.9.3 Taux de défaillance instantané

Le taux de défaillance représente la probabilité de défaillance à l'instant [t, $t+\Delta t$], sachant qu'il n'a pas eu de défaillance entre [0, t], en appliquant le théorème des probabilités conditionnelles [15] :

$$\lambda(t) = -\frac{1}{R(t)}\frac{dR(t)}{dt} \tag{I.32}$$

Le taux de défaillance passe par trois périodes :

- ✓ **Période de jeunesse :** dans cette période le taux est élevé puis il décroît, c'est la période de rodage.
- ✓ **Période de défaillance à temps constant :** le taux ce stabilise sur cette période.
- ✓ **Période de vieillissement ou d'usure :** le taux commence à croitre vite à cause du vieillissement des entités.

Tableau I.2 : Relations entre $R(t)$, $F(t)$, $f(t)$ et $\lambda(t)$.

Fonction	$R(t)$	$F(t)$	$f(t)$	$\lambda(t)$
$R(t)$	1	$1-F(t)$	$1-\int_0^{\infty} f(u)du$	$\exp\left[-\int_0^{t}\lambda(u)du\right]$
$F(t)$	$1-R(t)$	1	$\int_0^{t} f(u)du$	$1-\exp\left[-\int_0^{t}\lambda(u)du\right]$
$f(t)$	$-\frac{dR(t)}{dt}$	$-\frac{dF(t)}{dt}$	1	$\lambda(t)\exp\left[-\int_0^{t}\lambda(u)du\right]$
$\lambda(t)$	$-\frac{R'(t)}{R(t)}$	$\frac{\frac{dF(t)}{d(t)}}{1-F(t)}$	$\frac{f(t)}{1-\int_0^{t} f(u)du}$	1

Le tableau I.2 présente les relations entre la fiabilité, défiabilité, le taux et la densité de défaillance [3] .

I.10 Etat de l'art sur l'optimisation de la fiabilité des systèmes

I.10.1 Méthodes d'optimisation de la fiabilité des systèmes

L'optimisation de la fiabilité des systèmes ce fait par trois méthodes principales présentées comme suit [16] :

- ✓ Problème d'allocation de fiabilité.
- ✓ Problème d'allocation de redondance.
- ✓ Problème d'allocation de fiabilité-redondance.

I.10.1.1 Problème d'allocation de fiabilité

Cette méthode est basée sur des actions techniques, des règles de remplacement et des techniques de maintenance. Leurs objectif est l'amélioration de la fiabilité des composants en réduisant leur taux de défaillance et en augmentant leur taux de réparation [17].

La formulation mathématique générale du problème d'allocation de fiabilité est [16] :

$$R_s(r) = R_s(r_1, r_2, ..., r_m) \tag{I.33}$$

Sous contrainte :

$$g_j(r_1, r_1, ..., r_m) \le b \tag{I.34}$$

$$0 \le r_i \le 1 \;\; ; \;\; i = 1, 2, ..., m \tag{I.35}$$

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

où R_s est la fonction objectif du problème (fiabilité du système), g est l'ensemble des contraintes, r_i est la fiabilité de sous-système (i), m est le nombre de sous-systèmes et b est le vecteur de limitation des ressources.

I.10.1.2 Problème d'allocation de redondance

Afin d'améliorer la fiabilité du système, cette méthode utilise des composants supplémentaires sous forme parallèle pour augmenter l'état de fonctionnement de chaque sous-système. Il existe différents types de composants redondants pouvant être actifs ou en veille. Les composants redondants actifs sont généralement intégrés au

système au stade de la conception par contre les composants en veille (stand-by) ne sont pas activés lors de la phase initiale de fonctionnement du système. Il existe différents types de composants de veille, tels que les composants froids et chauds [17].

- **Actifs (hot)**

Plusieurs composants identiques fonctionnent simultanément, n'importe lequel d'entre eux serait capable de supporter la fonction normale du produit [18].

- **Stand-by**

Un système de secours est mis en réserve et ne fonctionne que lorsque le système principal tombe en panne [18].

- ✓ **Hot standby** Dans le cas d'une veille à chaud, l'unité en veille à une probabilité finie de tomber en panne également en veille [9].
- ✓ **Cold Standby** est une configuration dans laquelle le composant de veille n'est pas sujet à une panne jusqu'à ce qu'il soit allumé [9].

La formule mathématique générale du problème d'allocation de redondance est donnée comme suit [16] :

$$R_s(n) = R_s(n_1, n_2, ..., n_m) \tag{I.36}$$

Sous contraintes :

$$g_j(n_1, n_2, ..., n_m) \leq \mathrm{b} \tag{I.37}$$

$$1 \leq n_i \leq n_{i,\max,} \tag{I.38}$$

$$n_{\ i} \in \mathbb{Z}^+$$

où R_s est la fonction objectif du problème, g est l'ensemble des contraintes, n_i est le nombre de composants redondants, m est le nombre de sous-systèmes, b est le vecteur de limitation des ressources.

I.10.1.3 Problème d'allocation de fiabilité-redondance

Cette méthode inclue les deux précédentes méthodes, la fiabilité et la redondance [17].

La formule mathématique générale du problème d'allocation de Fiabilité-redondance est donnée comme suit [16] :

$$R_s(r,n) = R_s(r_1, r_2, ..., r_m; n_1, n_2, ..., n_m) \tag{I.39}$$

Sous contrainte :

$$g_j(r_1, r_2, ..., r_m; n_1, n_2, ..., n_m) \le b \tag{I.40}$$

$$0 \le r_i \le 1; \quad 1 \le n_i \le n_{i,\max,}\ i = 1, 2, ..., m \tag{I.41}$$

$$r_i \in [0,1] \subset \mathbb{R}^+ \ ; \ n_i \in \mathbb{Z}^+ \tag{I.42}$$

où R_s est la fonction objectif, g est l'ensemble des contraintes, r_i et n_i sont la fiabilité et le nombre de composants redondants dans le sous-système, respectivement, m est le nombre de sous-systèmes et b est le vecteur de limitation des ressources.

I.10.2 Travaux précédents

Plusieurs études ont été effectuées afin de résoudre les problèmes d'optimisation de la fiabilité des systèmes. En 2009, Frédérique Bicking [8], ont proposé une méthode basée sur les algorithmes génétique et des stratégies d'évolution. Elles combinent le principe du suivi des individus les plus aptes et les combinaisons génétiques pour un mécanisme de recherche élitiste. Afin d'illustrer la performance de la méthode proposée, deux systèmes complexes et deux systèmes mixtes multi-étages ont été étudiés et comparés avec le recuit simulé (SA) et une variante de l'algorithme de recuit simulé (I-NESA). Les résultats ont montré que les meilleures solutions sont obtenues par le (I-NESA). Autre méthode d'optimisation prometteuse a été proposée par M. Agrwal et al. [19] pour résoudre les problèmes d'allocation de redondance, nommée ASACO, testée sur divers systèmes complexes sous contraintes, linéaires et non linéaires. Les résultats étaient très encourageants et meilleurs que ceux obtenus par AGGA. Une autre proposition par Zou et al. [20] d'un nouvel algorithme modifié d'évolution différentielle

(NMDE) pour résoudre les problèmes d'optimisation sous contraintes. Ils ont testé l'algorithme sur 14 problèmes différents. L'algorithme NMDE est plus efficace que les autres méthodes de la littérature sur la recherche de meilleures solutions réalisables des problèmes les plus limités. Grag [21] a pris en compte un problème d'optimisation de type fiabilité-redondance avec une méthode de l'algorithme cuckoo search (CS) basée sur la pénalité avec des contraintes de ressources non linéaires. Cette méthode a été appliquée sur 25 analyses indépendantes. Après la comparaison avec d'autres méthodes, les solutions obtenues par (CS) sont meilleures. En 2016, Mellal et Zio [16] ont pris en considération trois types de problèmes d'optimisation de fiabilité mono-objectif afin d'optimiser la fiabilité des systèmes en utilisant l'algorithme PSFS (Penalty Guided Stochastic Fractal Search), appliqué sur dix études de cas numériques différentes, sous les contraintes telles que le coût, le poids et le volume avec une fonction de pénalité. La comparaison des résultats avec d'autres méthodes (telles que, GA, ES-PSO, ICS, etc.) a montré que le PSFS fournissait de meilleurs résultats. Trois méthodes ont été utilisées en 2017 par Mellal et al. [22] pour résoudre un problème d'optimisation d'allocation de fiabilité-redondance, tel que l'algorithme génétique (GA), l'algorithme d'optimisation du coucou (COA) et l'algorithme d'optimisation des essaims de particules (PSO), sous trois contraintes (volume, poids et coût). Le problème contient 20 sous-systèmes connectés en série avec 40 variables de décision (entières-réelles). Les résultats obtenus des trois méthodes ont montré clairement que le COA donne des résultats optimaux suivis par le PSO et le GA. Récemment en 2019, Mellal et Zio [23] ont proposé une nouvelle solution de méthode d'optimisation de la fiabilité des systèmes multi-objectif en développant un nouvel algorithme nommé ADAP-PSO. Il a été appliqué à 9 études de cas avec une comparaison avec le PSO classique. Les objectifs ont été normalisés dans un seul objectif en recourant à la méthode de la somme pondérée. Il a été prouvé que cet algorithme est plus performant que le PSO classique. Le tableau I.3 résume l'état de l'art.

Tableau I.3 : Résumé de l'état de l'art.

Référence	Fiabilité	Redondance	Fiabilité-redondance	Méthode
[8]	X	X		GA
[16]	X	X	X	PSFS
[22]			X	GA,COA,PSO
[23]			X	ADAP-PSO
[19]		X		ASACO
[20]	X			NMDE
[21]			X	CS

I.11 Conclusion

Dans ce chapitre, nous avons vu que la sûreté de fonctionnement des systèmes, dont laquelle la fiabilité est incluse, est un élément constructif essentiel dans la conception des systèmes est la maîtrise des risque. La fiabilité doit être améliorée par des méthodes de l'optimisation que nous allons aborder dans le chapitre II.

Chapitre II

Etat de l’art sur l’optimisation

II.1 Introduction

L'optimisation est comprise dans tous les domaines industriels, tel que la fiabilité des systèmes (déjà expliquée dans le premier chapitre).

Dans ce chapitre, nous présentons les différents types d'optimisation ainsi que leurs méthodes de résolution. On se focalisera beaucoup plus sur les algorithmes bio-inspirés.

II.2 Optimisation

L'optimisation est une branche mathématique qui sert à trouver le minimum ou le maximum d'une fonction (fonction de problème d'optimisation), appelée « fonction objectif ». Cette discipline est utilisée dans plusieurs domaines (recherche opérationnelle, ingénierie, biologie, etc.). La fonction objectif est écrite en fonction des variables de décision qui peuvent être réelles, entières ou binaires. La résolution du problème donne une meilleure solution, appelée « solution optimale ». Les méthodes de résolution peuvent être subdivisées en : méthodes classiques (basées sur les calculs mathématiques, telle que la continuité), méthodes non classiques (algorithmes bio-inspirés) adoptés comme une méthode pratique pour résoudre des problèmes d'optimisation complexes. Le problème d'optimisation est défini comme suit [24] :

$$\min_{x \in s} f(x) \quad \text{ou} \quad \max_{x \in s} f(x) \tag{II.1}$$

Les fonctions mathématiques d'un problème d'optimisation sont expliquées dans le paragraphe suivant [24] :

La modélisation d'un problème d'optimisation exige la définition des éléments qui composent les contraintes et la fonction objectif. Parmi ces éléments, certains sont connus (les paramètres du problème) et d'autres éléments sont inconnus (les variables). Si les variables représentent les décisions sur un ensemble continu de valeur, alors c'est une optimisation continue. Par contre si les variables prennent leurs valeurs dans un ensemble fini (ensemble des entiers), alors c'est une optimisation discrète.

Les contraintes et la fonction objectif s'expriment à l'aide des formules mathématiques qui sont généralement représentées comme suit :

$$f_i(X), (i = 1, 2, ..., I) \tag{II.2}$$

$$h_i(X),(j=1,2,...J) \tag{II.3}$$

$$g_l(X) \leq ou \geq 0,(l=1,2,...,L) \tag{II.4}$$

$$X=(x_1,x_2,...x_n)^T \tag{II.5}$$

où $f_i(X)$ est la fonction objectif de problème, $h_i(X)$ est la contrainte d'égalité, g_i (X) contrainte d'inégalité, et X est le vecteur de solution.

II.3 Contraintes

Ce sont des limitations qui doivent être satisfaites afin de considérer une solution acceptable. Cette limitation décrit les dépendances entre les variables de décision (combinatoire ou continue) et les paramètres du problème [25].

Il existe deux types de contraintes qui sont expliquées dans la section suivante.

II.3.1 Type de contraintes

La modélisation générale est définie comme suit :

$$g(X) \leq 0 \Leftrightarrow g(X)=g_i(X) \tag{II.6}$$

$$h(\mathrm{X})=0 \Leftrightarrow h(\mathrm{X})=h_k(\mathrm{X}) \tag{II.7}$$

$$Min \leq X \leq Max \tag{II.8}$$

Avec :

$$X=X_i \tag{II.9}$$

Ainsi que :

$$g(\mathrm{X}) \in R, h(\mathrm{X}) \in R_k \tag{II.10}$$

$g_i(X)$ est la contrainte d'inégalité, j est le nombre de fonctions, $h_i(X)$ est la contrainte d'égalité et k est le nombre de fonctions.

II.3.2 Maîtrise des contraintes

1. Méthode de pénalité

La méthode de pénalité est la plus utilisée dans la maîtrise des contraintes. Elle est basée principalement sur le choix de la valeur de pénalité pour chaque contrainte. Cette méthode convertie un problème d'optimisation sous contrainte à un problème sans contrainte avec une augmentation. Elle est appliquée à l'aide d'une fonction comme suit [25] :

$$\tilde{f}(X) = f(X) \pm p(X) \qquad \text{(II.11)}$$

2. Méthode basée sur la faisabilité des solutions

Cette méthode sert à réparer les solutions non faisables à l'aide d'une technique appelée « Homomorphous mapping », donc c'est faire le codage ou le décodage avec un cube unitaire dimensionnel afin d'assurer la réparation des solutions irréalisable [25].

3. Méthodes hybrides

Il existe plusieurs méthodes parmi elle la combinaison d'un algorithme génétique et d'autres techniques de l'intelligence artificielle en séparent la fonction objectif de la contrainte. Cette méthode est appliquée juste pour les fonctions (ou contraintes) dérivable [25].

II.4 Types d'optimisation

On peut catégoriser les problèmes d'optimisation selon plusieurs critères : les problèmes linéaires ou non linéaires, qui peuvent être avec contraintes ou sans contraintes, avec des valeurs combinatoires ou continues. Où l'espace de recherche peut être convexe ou non convexe, mais principalement on les catégorise en fonction du nombre d'objectifs (mono-objectif ou multi-objectif).

- **Linéaire :** C'est une classe de problème d'optimisation qui contient des fonctions objectif et de contraintes linéaires. Il existe diverses méthodes très efficaces pour la résolution de ce type, et est faite par la méthode du simplexe [26].
- **Non linéaire :** Cette classe de problèmes d'optimisation inclue des fonctions objectifs et/ou de contraintes non linéaire. Il existe plusieurs méthodes pour

résoudre ce type de problèmes, telle que la méthode approximative et la programmation non linéaire [26].

- **Convexe :** Les problèmes d'optimisation où l'espace de recherche convexe ont une forme générale comme suit [26], [27] :

$$Min\ ou\ Max\ f_0(x) \tag{II.12}$$

$$f_i(x) \le b_i, \tag{II.13}$$

$$i = 1,...,m$$

Telle que la fonction,

$$f_0,...,f_m\ R^n \to R \text{ est convexe} \tag{II.14}$$

C'est-à-dire,

$$f_i(\alpha x + \beta y) \le \alpha f_i(x) + \beta f_i(y) \tag{II.15}$$

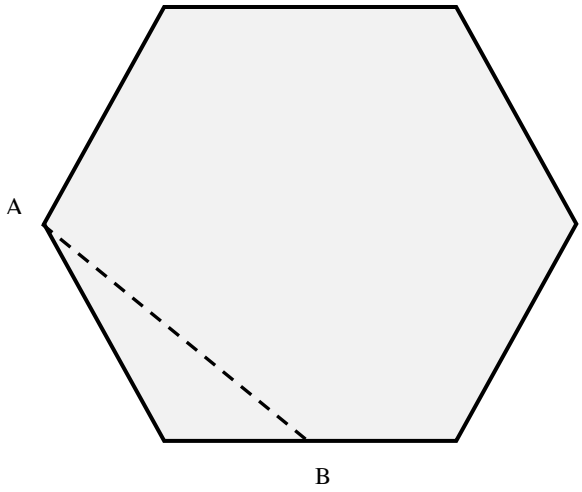

Figure II.1 : Espace de recherche convexe.

La figure II.1 représente un espace de recherche convexe, tel que tous les points appartiennent à la ligne (*AB*) sont des solutions convexe.

- **Non convexe :** Tant que les problèmes d'optimisation sont irréguliers, il existe différentes méthodes pour les résoudre [26], [27] :

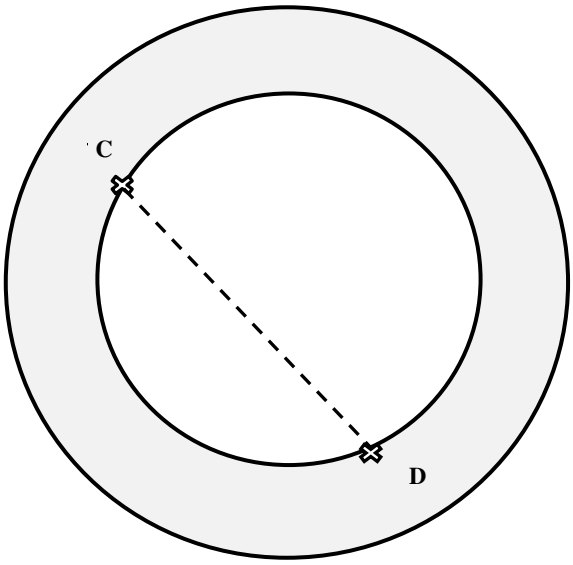

Figure II.2 : Espace de recherche non convexe.

La figure II.2 représente un espace de recherche non convexe, tel que tous les points appartiennent à la ligne (*CD*) sont des solutions non convexe.

II.4.1 Mono-objectif

L'optimisation mono-objectif contient un seul critère (une seule fonction objectif), dont le but est de minimiser ou maximiser *f*(*X*), sous un ensemble de contraintes ou sans contraintes. On formule ces contraintes par un ensemble d'inégalités ou d'égalités avec des méthodes de résolution, tel que les algorithmes bio-inspirés [28] :

$$Max\ ou\ Min\ f(X) \tag{II.16}$$

II.4.2 Multi-objectif

L'optimisation multi-objective est un problème avec plusieurs critères sous un ensemble de contraintes ou sans contraintes. Dans ce type d'optimisation la solution optimale n'est plus qu'une solution unique, mais un ensemble de solutions compromis entre les différents objectifs à optimiser [28].

Le problème est décrit comme suit :

$$F(X) = \{f_1(X), f_2(X), ..., f_i(X)\} \tag{II.17}$$

II.5 Méthodes de résolution

Les méthodes de résolution des problèmes d'optimisation peuvent être divisées en deux méthodes principales :

- Conventionnelles.
- Non conventionnelles.

II.5.1 Conventionnelles

Les méthodes conventionnelles sont des techniques mathématiques classiques. Elles sont classées en deux catégories [29] :

- Méthodes d'optimisation non linéaires basées sur la théorie de calcule différentiel ou le gradient.
- Méthodes de programmation linéaire qui sont basées sur les techniques du simplexe et du point intérieur.

II.5.2 Non conventionnelles

Principalement on trouve dans cette catégorie, les méthodes de l'intelligence artificielle (calculs évolutionnaires ou algorithmes bio-inspirés ou méta-heuristiques) [29].

Il existe plusieurs algorithmes, donc il est impossible de les citer tous. Dans cette section, on va citer quelques méthodes.

II.5.2.1 Gray Wolf Optimizer (GWO)

Le Gray Wolf Optimizer est un algorithme développé en 2014 par Mirjalili et al. [30] inspiré du mode de vie des loups gris, où la taille du groupe est de 5 à 12 loups.

Le chef Alpha (α) peut être mâle ou femelle. Le loup est principalement responsable de la prise de décisions concernant la chasse, le lieu de repos, etc.

Bêta (β) est le second niveau de la hiérarchie des loups gris. Les Bêtas sont des loups subordonnés qui aident le chef à prendre des décisions ou de faire d'autres activités dans le cadre du groupe, peut être mâle ou femelle tout comme le Alpha. Il est probablement le meilleur candidat qui peut remplacer le chef au cas où l'un des loups alpha décède ou devient très vieux.

Delta (δ), appelé subordonné, est une catégorie, qui joue des rôles très importants, tels que : avertir la meute en cas de danger, surveiller les limites du territoire, etc.

Oméga (ω) : Cet élément n'est pas important dans la meute. Il ne peut manger sans autorisation. Cet élément est utilisé pour diversifier les solutions (la population).

La chasse se déroule en trois étapes principales :

- ✓ Recherche de proie.
- ✓ Encerclement de la proie et l'attaque.
- ✓ Mise en œuvre pour effectuer une optimisation.

L'algorithme 1 présente le pseudo-code général du GWO.

Algorithme 1 : Pseudo-code général du GWO [30].

Initialiser la population des loups gris ;
Initialiser *a, A, et C ;*
Calculer l'aptitude de chaque agent de recherche ;
X_α= le meilleur agent de recherche ;
X_β=le deuxième meilleur agent de recherche ;
X_δ=le troisième meilleur agent de recherche ;
Tant que (t < nombre max d'itérations) **faire**
  Pour chaque agent de recherche
    Mettre à jour la position de l'agent de recherche actuel ;
  Fin pour
Mettre à jour *a, A, et C ;*
    Calculer l'aptitude de tous les agents de recherche ;
    Maitre à jour X_α, $X_{\beta,\ et} X_\delta$
    t=t+1
Fin Tant que
Retour X_α

II.5.2.2 Shuffled Frog-Leaping Algorithm (SFLA)

L'algorithme SFLA est une méta-heuristique proposer à l'origine par Eusuff et Lansey en 2003, il est inspiré de la mimétique naturelle du saute de grenouille et son comportement lors de la recherche de l'emplacement et la nourriture, tel que la recherche locale est accomplie par chaque grenouille. La population de grenouilles dans cet algorithme est divisé en plusieurs groupes, appelés « memeplexes ». Au sein de chaque memeplex, le comportement de chaque grenouille peut être affecté par d'autres grenouilles. Le SFLA a été développée pour résoudre les problèmes d'optimisation

combinatoire. Il combine les avantages de l'algorithme mimétique à base de gènes et l'algorithme d'optimisation basé sur le comportement social des essaims de particules [31]. L'algorithme 2 présente le pseudo-code général du SFLA.

Algorithme 2 : Pseudo-code général de SFLA [31].

---

1 : Initialiser les problèmes de l'algorithme ;
2 : Générer une population aléatoire ;
3 : Trouver la forme physique de la population ;
4 : Trier les membres (grenouille) par performance ;
5 : *Z*=1
6 : Séparer les grenouilles en groupes (memeplexes) ;
7 : Evolution mimétique pour chaque memeplex ;
Générer un sous-memeplex à partir du memeplex actuel ;
Trouver la pire position de grenouille ;
Mettre à jour la position de la pire grenouille ;
Trier le memeplex par performance ;
Si le nombre de sous-itérations est terminé, passez au memeplex suivant ;
Si l'évolution est faite pour tous les memeplexes, passez à l'étape suivante ;
8 : Mélangez les memeplexes et triez la population ;
9 : Mise à jour les meilleurs résultats ;
10 : *Z*++
11 : Si Z< Nombre max d'itération, routeur au l'étape 6 ;

II.5.2.3 Particle Swarm Optimization (PSO)

Le PSO est un algorithme développé en 1995 par Kennedy et Eberhart, inspiré par le comportement d'organismes sociaux appartenant à des groupes, tels que les oiseaux et les poissons. Par conséquent, le PSO commence par des solutions aléatoires, appelées particules, et l'ensemble des solutions est appelé essaim. Chaque particule a son vecteur de position et son vecteur de vitesse et est influencée par la meilleure solution (meilleure position atteinte) [32]. L'algorithme 3 présente le pseudo-code général du PSO.

Algorithme 3 : Pseudo-code général de PSO [32].

Initialisation aléatoire de tout l'essaim ;
Évaluer $f(x_i)$;
Pour chaque particule *i* **faire**
Mettre à jour les vitesses ;
Déplacez vers la nouvelle position ;
Si $f(x_i) < f$ (p meilleur i) **alors** p meilleur$_i$=x_i ;
Si $f(x_i) < f$ (g meilleur) **alors** g meilleur= x_i ;
Mise à jour (x_i, v_i);
Fin Pour
Mise à jour Jusqu'au critère d'arrêt ;

II.5.2.4 Ant Colony Optimization (ACO)

Cet algorithme a été développé par Dorigo en 1992, inspiré du quotidien de la vie des fourmis. Ces espèces laissent une odeur spécifique sur la trajectoire afin de connaître le chemin vers leurs clonies et les lieux de nourriture. Une concentration très élevée est fournie par les fourmis pour assurer cette opération. L'algorithme 4 présente le pseudo-code général de ACO [33].

Algorithme 4 : Pseudo-code général de ACO [33].

Initialiser les traces de phéromones ;
Evaluer les lieux ;
Pour chaque fourmi **faire**
Construction de la solution en utilisant la piste de phéromone ;
Mettez à jour les pistes de phéromone ;
Evaporation ;
Renforcement ;
Mise à jour jusqu'au critère d'arrêt ;
Résultat : Meilleure solution trouvée ou un ensemble de solutions ;

Le tableau II.1 résume quelques algorithmes bio-inspirés.

Tableau II.1 : Algorithmes de l'intelligence artificielle.

Méthode	Inspiration	Commentaire
COA [22]	Inspiré du comportement de l'oiseau coucou.	Gérer la recherche locale et globale, utiliser le vol Levy comme stratégie de recherche. Par contre il a un faible taux de convergence.
ACO [34]	Inspiré de la société des fourmis	Le calcul distribué dans ACO évite la convergence prématurée. Bien que la convergence soit garantie, mais le temps de convergence est non défini.
ALO [35]	Imite le mécanisme de chasse des lions dans la nature	Il est applicable à la résolution des problèmes réels avec des espaces de recherche inconnus.
WOA [36]	Simule le comportement de chasse des baleines à bosse.	WOA est simple et robuste et il permet de résoudre les problèmes de regroupement.
GA [37]	Basés sur l'évolution biologique.	Facile à mettre en œuvre, utilise des opérateurs simples et peut être utilisé pour résoudre des problèmes avec une complexité de calcul élevée. Par contre, il consomme le temps.
FPA [38]	Inspiré du processus de pollinisation des fleurs.	Flexible, simple et peu de paramètres. Utilisé pour gérer un objectif simple ou a multiple problèmes. Taux de convergence lent et une faible précision.
NMDE [20]	Nouvel algorithme, modifié de d'évolution différentielle	L'avantage principal est que chaque solution a son propre facteur d'échelle et son propre taux de recouvrement

II.6 Conclusion

Dans ce chapitre, nous avons présenté les différents types d'optimisation ainsi que leurs méthodes de résolution, en particulier les méthodes non conventionnelles basées sur les algorithmes de l'intelligence artificielle. Dans le chapitre suivant, nous allons implémenter deux algorithmes bio-inspirés pour résoudre deux problèmes de l'optimisation de la fiabilité des systèmes.

Chapitre III

Optimisation de la fiabilité des systèmes

III.1 Introduction

Ce chapitre est consacré à l'étude de l'optimisation de la fiabilité en phase de conception en utilisant deux algorithmes bio-inspirés (GWO et SFLA) présentés dans le chapitre II. Le but est les implémenter et comparer leurs performances. L'approche implémente une méthode de pénalité qui permet de guider la recherche vers les meilleures solutions dans l'espace réalisable, tel motionné dans le chapitre II.

III.2 Optimisation de la fiabilité des systèmes

Afin d'optimiser la fiabilité des systèmes, trois méthodes principales peuvent être suivies [16], [39] :

- ✓ Augmenter la fiabilité des composants (allocation de fiabilité).
- ✓ Utiliser des composants redondants en parallèle (allocation de redondance).
- ✓ Combinaison des deux méthodes précédant (allocation de fiabilité-redondance).

Le problème d'allocation de fiabilité implique des variables réelles, alors que le problème d'allocation de redondance implique des variables entières, et les deux (réelles-entières) pour l'allocation fiabilité-redondance [23]. Cette dernière est l'un des problèmes d'optimisation les plus complexes en ingénierie [16]. L'objectif de ces méthodes est de maximiser la fiabilité du système en prenant en considération les contraintes de conception des systèmes, telles que [16], [40] :

- Coût.
- Poids.
- Volume.

III.3 Etudes de cas

Dans cette partie, nous allons traiter deux problèmes différents où les deux sont des problèmes mono-objectif, non linéaires et avec contraintes. Il est à noter que les espaces de recherche de ces problèmes sont convexes.

III.3.1 Système réseau en pont complexe

Le problème présenté dans la figure III.1 (problème 1) est un problème d'allocation de fiabilité contenant 5 sous-systèmes connectés en réseau pont complexe [16], [41]–[44] :

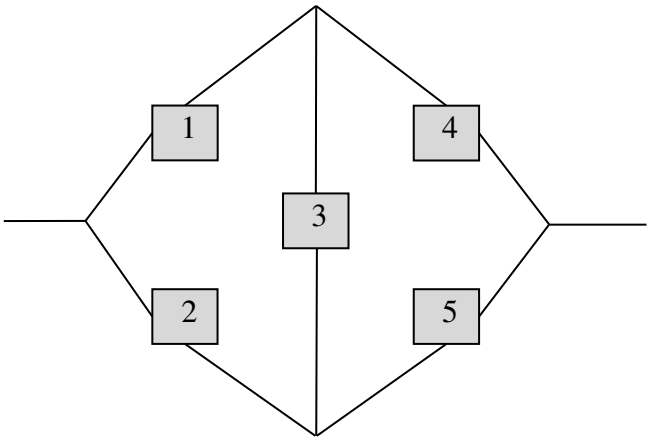

Figure III.1 : Système réseau en pont complexe.

L'objectif est de minimiser le coût de conception en prenant en considération la contrainte de fiabilité, où les variables sont des réelles.

Le problème mathématique du système est défini comme suit :

$$Minimiser\ C_s(r) = \sum_{i=1}^{5} a_i \exp\left(\frac{b_i}{1-r_i}\right) \tag{III.1}$$

Sous les contraintes :

$$0.5 \le r_i \le 1, \quad \text{, pour } \ i = 1,2,3,4,5 \tag{III.2}$$

$$0.99 \le R_s \le 1 \tag{III.3}$$

Avec :

$$\begin{aligned} R_s(r_1,r_2,r_3,r_4,r_5) &= r_1r_4 + r_1r_5 + r_2r_3r_4 + r_1r_3r_5 + 2r_1r_2r_3r_4r_5 \\ &- r_2r_3r_4r_5 - r_1r_3r_4r_5 - r_1r_2r_4r_5 - r_1r_2r_3r_5 - r_1r_2r_3r_4 \end{aligned} \tag{III.4}$$

$$a_i = 1 \ ; \ b_i = 0.0003 \quad \forall i$$

où C_s est la fonction objectif du problème 1, et R_s est la contrainte de fiabilité et r_i est la fiabilité de chaque composant dans chaque sous-système. Le tableau III.1 résume ce problème.

Tableau III.1 : Résumé du problème 1**.**

	Nombres	**Nature**
Variables	5	Réelles
Contraintes	2	Inégalité

III.3.1 Système de protection de survitesse (turbine à gaz)

Le problème présenté dans la figure III.2 (problème 2) est un problème d'allocation de fiabilité-redondance. L'objectif est de protéger la turbine à gaz en impliquant quatre vannes. Ces dernières doivent se fermer pour couper l'alimentation en carburant en cas de survitesse, où la survitesse est détectée par des systèmes électriques et mécaniques [16], [39], [43], [45].

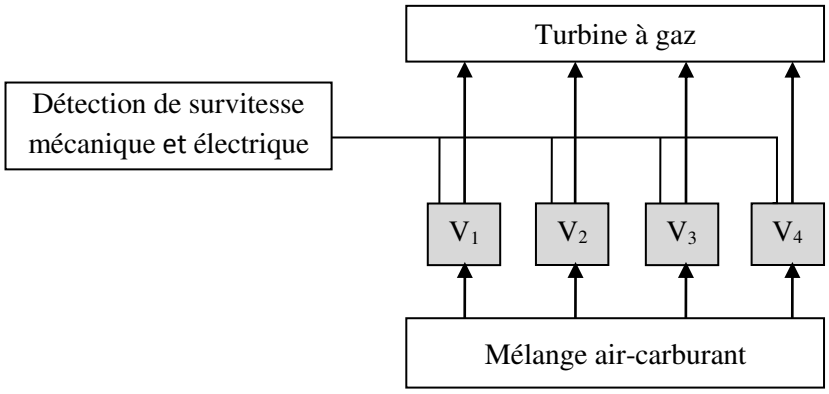

Figure III.2 : Système de protection de survitesse.

Le modèle mathématique est donné comme suit :

$$Maximiser\ \ R_S(r,n) = \prod_{i=1}^{4}\left[1-(1-r_i)^{n_i}\right] \qquad \text{(III.5)}$$

Sous contraintes :

$$g_1(r,n) = \sum_{i=1}^{4} v_i n_i^2 \le V \qquad \text{(III.6)}$$

$$g_2(r,n)=\sum_{i=1}^{4}\alpha_i(-T/\ln r_i)^{\beta_i}\left[n_i+\exp\left(n_i/4\right)\right]\le C \qquad \text{(III.7)}$$

$$g_3(r,n)=\sum_{i=1}^{4}w_i n_i\exp(n_i/4)\le W \qquad \text{(III.8)}$$

$$0.5\le r_i\le 1\ ,\quad r_i\in[0,1]\subset\mathbb{R}^+ \qquad \text{(III.9)}$$

$$1\le n_i\le 10\ ,\quad n_i\in\mathbb{Z}^+\ ;i=1,2,....,4 \qquad \text{(III.10)}$$

où R_s est la fonction objectif du problème 2 et R_{c1}, R_{c2}, R_{c3} sont les ressources non consommées de V, C, W respectivement, r_i est la fiabilité de chaque composant dans chaque sous-système, n_i est le nombre de composants dans chaque sous-système. Le tableau III.2 résume ce problème. T est le temps de mission.

Tableau III.2 : Résumé du problème 2.

	Nombres	Nature
Variables	8	4 entières, 4 réelles
Contraintes	5	Inégalité

Tableau III.3 : Données du problème 2.

Sous-système *i*	**$10^{-5}\alpha$**	**β_i**	**v_i**	**ω_i**	***W***	***C***	***V***	***T* (*h*)**
1	1,0	1,5	1	6	250	400	450	1000
2	2,3	1,5	2	6				
3	0,3	1,5	3	8				
4	2,3	1,5	2	7				

Il est à noter que les données des deux problèmes sont en unités arbitraires.

III.4 Solutions proposées

Nous allons résoudre les deux problèmes décrits précédemment dans la section III.2 à l'aide de deux algorithmes de l'intelligence artificielle (GWO et SFLA).

III.4.1 Grey Wolf Optimizer (GWO)

L'algorithme 1 représente le pseudo-code de GWO implémenté pour résoudre ces problèmes.

Algorithme 1 : Pseudo-code du GWO implémenté.

Introduction des paramètres : N_{pop}, N_{itr} ;
Générer les agents de recherche initiaux ;
Tant que $z \leq N_{itr}$ ***faire***
    Evaluer la fonction d'objectif globale ;
    Maîtrise des contraintes à l'aide de la fonction de pénalité ;
    Pour j=1 : taille (position, 1)
        Renvoie les agents qui dépassent les limites de l'espace de recherche ;
        Calculer la fonction objectif pour chaque agent de recherche ;
        Mise à jour : Alpha, Beta, et Delta ;
    Pour j=1:taille (Positions, 1)
    Pour k=1:taille (Positions, 2)
        Mise à jour de la position des agents de recherche, y compris les omégas ;
        Positions $(j,k)=(X_1+X_2+X_3)/3$;
Fin Tant que
Affichage des résultats.

La figure III.3 représente l'organigramme général du GWO implémenté.

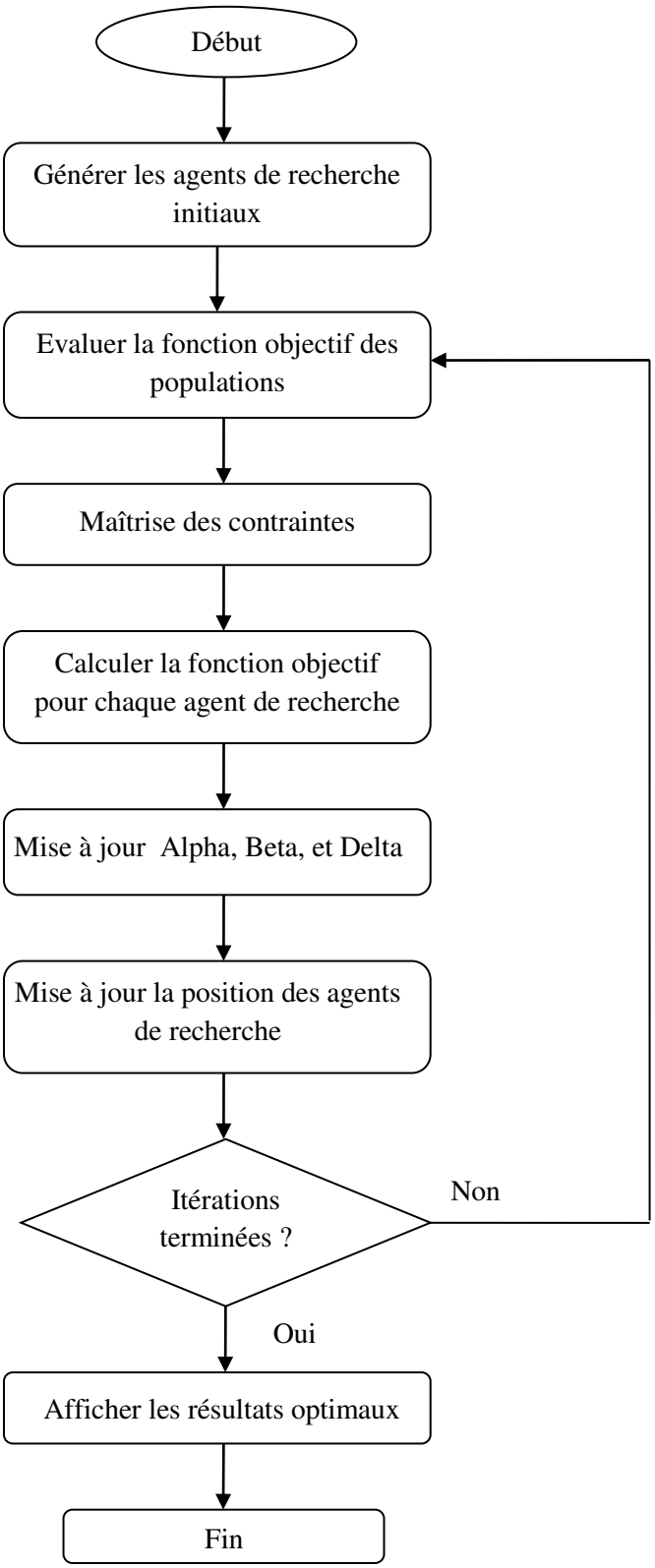

Figure III.3 : Organigramme du GWO implémenté.

III.4.2 Shuffled Frog-Leaping Algorithm (SFLA)

L'algorithme 1 représente le pseudo-code du SFLA implémenté pour résoudre ces problèmes, où les paramètres utilisés sont représenté dans le tableau III.4.

Tableau III.4 : Paramètre de SFLA implémenté**.**

Paramètres	Valeurs
Nombre de memeplexes	5
Membre de chaque memeplex	10
Membre de chaque sous-memeplex	5

Algorithme 2 : Pseudo-code du SFLA implémenté.

Introduction des paramètres : N_{pop}, N_{itr} ;
Générer population des grenouilles aléatoirement ;
Calculer la valeur de remise en forme de chaque grenouille ;
Trier les membres en fonction de performance ;
Tant que $z \leq N_{itr}$ **faire**
    Séparer les membres en memeplex ;
    Evaluer la fonction d'objectif ;
    Maitrise des contraintes à l'aide de la fonction de pénalité ;
    Pour i=1 :N_{pop}
        pop(i).Position = unifrnd (VarMin, VarMax, VarSize) ;
        pop(i).Cost = CostFunction (pop(i).Position) ;
    Pour it = 1:MaxIt
        Trouver la meilleure position ;
        Mise à jour de la mauvaise position ;
        Mélanger les memeplexes et trier la population ;
        Mise à jour de la solution optimale ;
Fin Tant que
Affichage des résultats.

La figure III.3 représente l'organigramme général du SFLA implémenté.

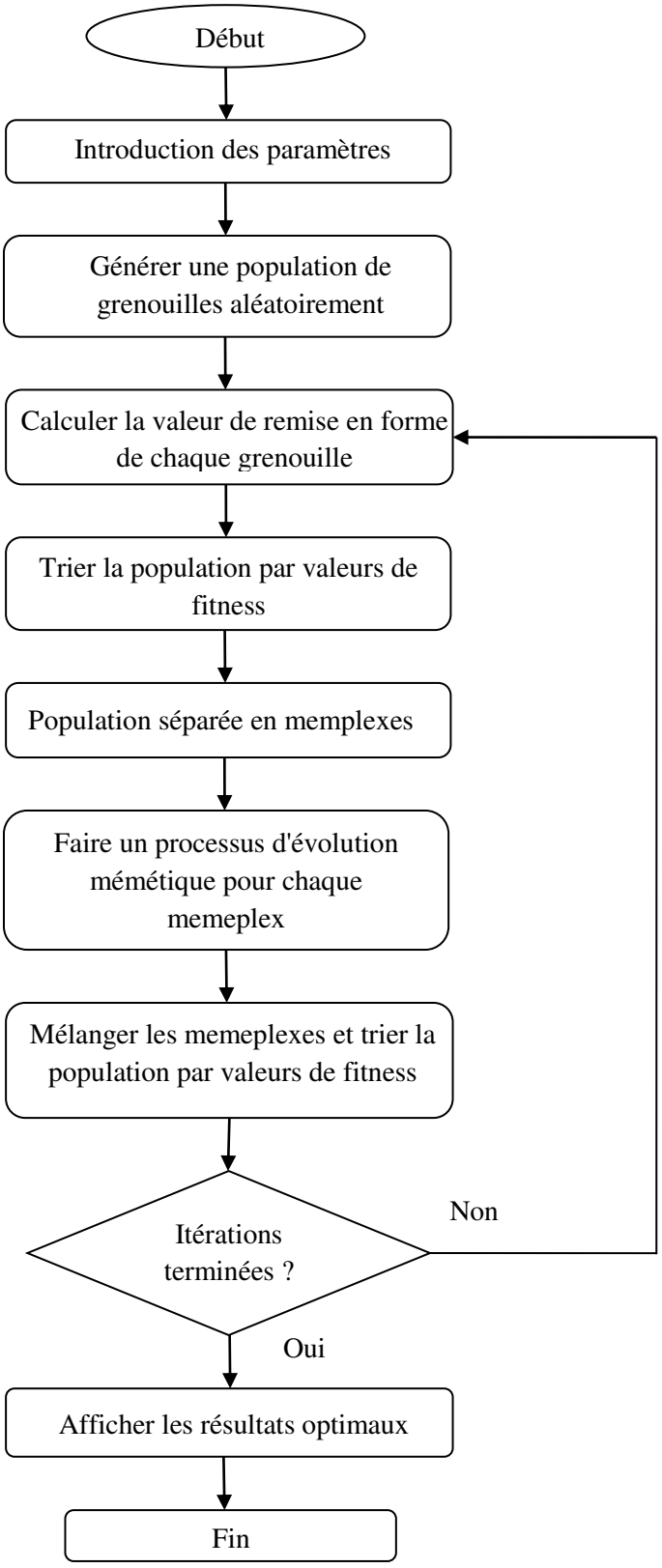

Figure III.4 : Organigramme du SFLA implémenté.

III.5 Résultats et commentaires

Les deux approches proposées dans ce chapitre ont été programmées à l'aide de MATLAB 2017a et exécutées sur un PC avec les spécificités suivantes : processeur Intel (R) Core (TM) i3 M380 2.53GHz, RAM de 4GB, Windows 7 (64bits).

Les tableaux III.6-III.33 illustrent les résultats obtenus par les deux algorithmes de l'intelligence artificielle (GWO et SFLA). Ces programmes ont été lancés en 10 exécutions indépendantes, où les meilleures performances sont représentées en gras. Les Paramètres d'optimisation des algorithmes employés sont présentés dans le tableau III.5.

Les critères de performances sont comme suit :

- Meilleure, moyenne et mauvaise solution.
- Nombre de fonctions d'évaluation (NFE).
- Temps d'exécution en s (CPU).
- Ecart type (σ).

Tableau III.5 : Paramètres des algorithmes.

	Problème 1		Problème 2	
	N_{pop}	N_{itr}	N_{pop}	N_{itr}
Paramètres 1	5	50	50	500
Paramètres 2	50	500	100	1000

Problème 1 : Réseau en pont complexe.

Problème 2 : Système de protection de survitesse.

Tableau III.6 : Résultats donnés par GWO (problème 1, paramètres 1).

No	r_1	r_2	r_3	r_4	r_5	C_s	R_s
1	0,859639	0,971339	0,767836	0,974184	0,921231	5,02945904	0,9914
2	0,830009	0,964586	0,735278	0,928457	0,976041	5,02820937	0,9908
3	0,942235	0,946583	0,620571	0,948915	0,922648	**5,02140562**	0,9906
4	0,957334	0,875464	0,912711	0,891308	0,972540	5,02665966	0,9905
5	0,990539	0,831620	0,500000	0,968660	0,845855	5,04616837	0,9908
6	0,971808	0,858845	0,896119	0,989464	0,685495	5,04555575	0,9918
7	0,842484	0,980608	0,655592	0,559357	0,995045	5,08147048	0,9922
8	0,966620	0,828409	0,543501	0,974445	0,946486	5,02886554	0,9904
9	0,751908	0,969472	0,905481	0,949004	0,969438	5,03002884	0,9901
10	0,885781	0,965101	0,523962	0,964212	0,936055	5,02501413	0,9900

Tableau III.7 : *Suite du tableau III.6.*

No	CPU (s)	NFE	σ
1	0,226350	250	1,694824E-02
2	0,026535	250	
3	0,031055	230	
4	0,023544	250	
5	0,025232	215	
6	0,010062	250	
7	0,007857	250	
8	0,007569	250	
9	0,006891	250	
10	0,006558	250	

Tableau III.8 : Valeurs de C_s données par GWO (problème 1, paramètres 1).

C_s	Meilleure	Moyenne	Mauvaise
	5,02140562	5,03628368	5,08147048

Les tableaux III.6-III.7 représentent les résultats obtenus par l'algorithme GWO avec une taille de population égale à 5 et un nombre d'itérations maximal de 50, où la meilleure valeur de (C_s) donnée est 5,02140562 avec un NFE égale à 230 et un σ qui égale à 1,694824E-02 avec une consommation de temps (CPU) de 0.031055s.

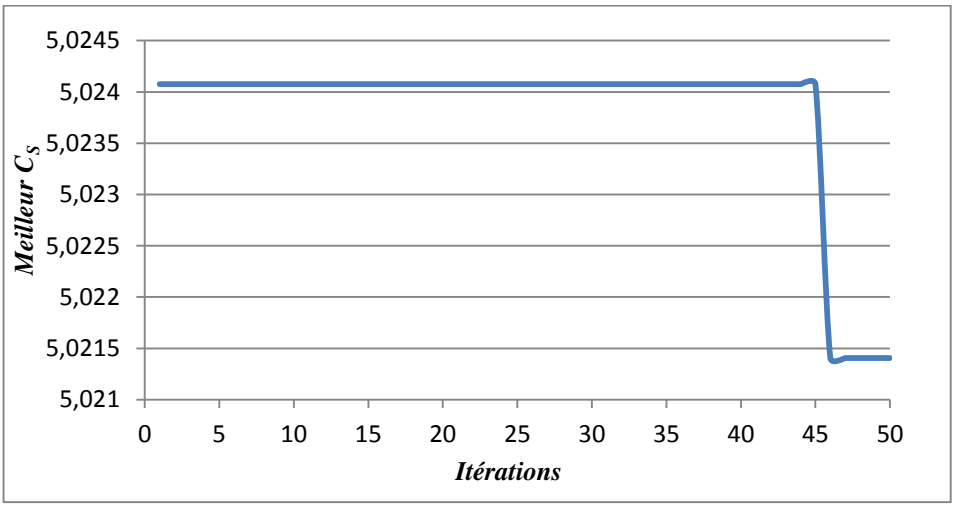

Figure III.5 : Meilleur coût donné par GWO (problème 1, paramètres 1).

La figure III.5 représente le meilleur coût donné par l'algorithme GWO (paramètres 1). La courbe est stable pondant les 45 premières itérations, puis elle converge jusqu'à la meilleure solution avec un coût optimal de 5,02140562.

Tableau III.9 : Résultats donnés par GWO (problème 1, paramètres 2).

No	r_1	r_2	r_3	r_4	r_5	C_s	R_s
1	0,938791	0,934198	0,722792	0,934414	0,938663	5,02005323	0,9900
2	0,933304	0,937396	0,757149	0,937737	0,934437	5,01996388	0,9900
3	0,940061	0,934199	0,679259	0,940310	0,935208	5,02020258	0,9900
4	0,931857	0,935663	0,791978	0,935660	0,936528	**5,01994055**	0,9900
5	0,939431	0,935113	0,677014	0,938345	0,937304	5,02020286	0,9900
6	0,936628	0,934740	0,810890	0,933413	0,932971	5,01994173	0,9900
7	0,936058	0,930833	0,824614	0,933726	0,935725	5,01997683	0,9900
8	0,936834	0,934946	0,734126	0,936572	0,936948	5,02002224	0,9900
9	0,939914	0,935738	0,643311	0,939513	0,937976	5,02034663	0,9900
10	0,929987	0,937777	0,796376	0,934605	0,936674	5,01994772	0,9900

Tableau III.10 : *Suite du tableau III.9.*

No	CPU (s)	NFE	σ
1	0,847499	24250	1,346793E-04
2	0,431736	24900	
3	0,411610	25000	
4	0,480275	24900	
5	0,355004	24950	
6	0,339813	25000	
7	0,389689	25000	
8	0,367078	25000	
9	0,366178	23600	
10	0,409083	24900	

Tableau III.11 : Valeurs de C_s données par GWO (problème 1, paramètres 2).

C_s	Meilleure	Moyenne	Mauvaise
	5,01994055	5,02005982	5,02034663

Les tableaux III.9-III.10 représentent les résultats obtenus par l'algorithme GWO avec une taille de population égale à 50 et un nombre d'itérations maximal de 500. La meilleure valeur de (C_s) est égale à 5.01994055 avec un NFE de 24900, une consommation de temps (CPU) de 0.480275 s et un σ égale à 1.346793E-04.

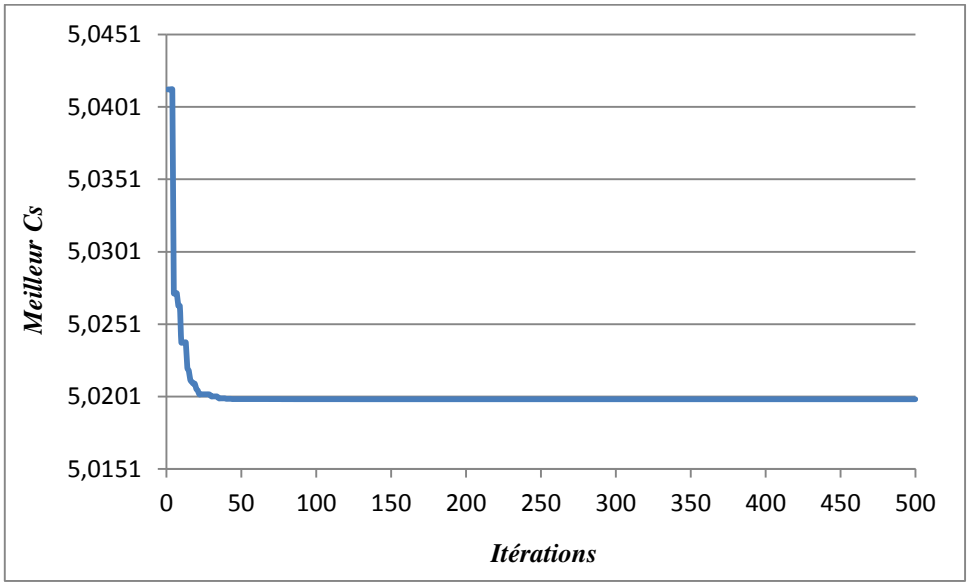

Figure III.6 : Meilleur coût donné par GWO (problème 1, paramètres 2).

La figure III.6 représente le meilleur coût donné par l'algorithme GWO (paramètres 2). Cette figure illustre une convergence rapide jusqu'à la meilleure solution au niveau de l'itération 40 avec une valeur optimale de 5,01994055.

Tableau III.12 : Résultats donnés par SFLA (problème 1, paramètres 1).

No	r_1	r_2	r_3	r_4	r_5	C_s	R_s
1	0,934766	0,935961	0,786965	0,934563	0,934787	5,01992014	0,9900
2	0,934666	0,934893	0,781433	0,935427	0,935631	5,01992248	0,9900
3	0,934361	0,935395	0,789161	0,936127	0,933967	5,01992052	0,9900
4	0,936948	0,931511	0,789034	0,937962	0,933266	5,01993528	0,9900
5	0,935560	0,934440	0,787227	0,935213	0,934834	**5,01991929**	0,9900
6	0,934996	0,935154	0,795515	0,935907	0,933164	5,01992135	0,9900
7	0,936039	0,933523	0,788674	0,935763	0,934558	5,01992091	0,9900
8	0,929352	0,940418	0,787928	0,934645	0,935192	5,01995948	0,9900
9	0,937432	0,933187	0,790492	0,935762	0,933274	5,01992665	0,9900
10	0,928963	0,940032	0,777618	0,934813	0,936757	5,01996481	0,9900

Tableau III.13 : *Suite du tableau III.12.*

No	CPU(s)	NFE	σ
1	1,930929	250	1,619814E-05
2	1,453093	245	
3	1,266169	245	
4	1,360883	245	
5	1,360631	240	
6	1,454644	230	
7	1,396704	235	
8	1,365605	250	
9	1,391349	250	
10	1,335716	250	

Tableau III.14 : Valeurs de C_s données par SFLA (problème 1, paramètres 1).

C_s	Meilleure	Moyenne	Mauvaise
	5,01991929	5,01993109	5,01996481

Les tableaux III.12-III.13 représentent les résultats obtenus par l'algorithme SFLA avec une taille de population égalé à 5 et un nombre d'itération maximal de 50. La meilleure valeur de (C_s) est 5,01991929 avec un NFE égale à 240, un CPU de 1.360631s et un σ qui égale à 1.619814E-05.

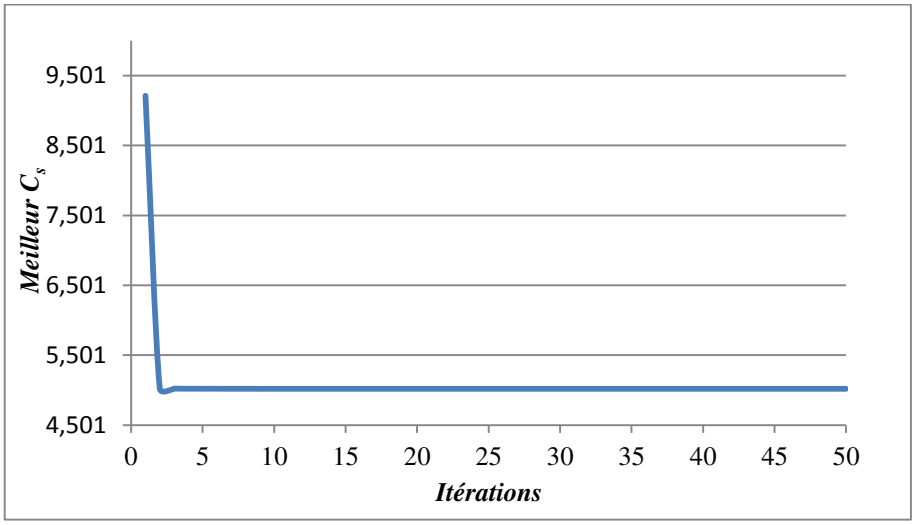

Figure III.7 : Meilleur coût donné par SFLA (paramètres 1).

La figure III.7 représente le coût optimal donné par l'algorithme SFLA (paramètres 1). La courbe illustre une convergence rapide jusqu'à la stabilisation à l'itération 5 avec une meilleure solution de 5,01991929.

Tableau III.15 : Résultats donnés par SFLA (problème 1, paramètres 2).

No	r_1	r_2	r_3	r_4	r_5	C_s	R_s
1	0,934990	0,935180	0,789896	0,934474	0,935146	5,01991853	0,9900
2	0,934774	0,934938	0,792922	0,934686	0,935098	5,01991821	0,9900
3	0,935086	0,934836	0,791855	0,935016	0,934662	**5,01991820**	0,9900
4	0,934727	0,935125	0,793173	0,934659	0,934961	5,01991826	0,9900
5	0,935186	0,934807	0,792924	0,934371	0,935130	5,01991850	0,9900
6	0,934520	0,935000	0,794312	0,934985	0,934855	5,01991826	0,9900
7	0,935229	0,934829	0,789454	0,934844	0,934935	5,01991849	0,9900
8	0,934340	0,935633	0,795084	0,934388	0,934918	5,01991916	0,9900
9	0,934732	0,935124	0,791518	0,935041	0,934736	5,01991823	0,9900
10	0,934961	0,934939	0,789732	0,934720	0,935188	5,01991837	0,9900

Tableau III.16 : *Suite du tableau III.15.*

No	CPU (s)	NFE	σ
1	11,950912	21050	2,744631E-07
2	14,031096	23750	
3	11,947548	21900	
4	12,365663	21800	
5	12,503154	23650	
6	12,220336	21800	
7	12,107528	24500	
8	12,765308	24050	
9	12,922485	24050	
10	12,019701	24100	

Tableau III.17 : Valeurs de C_s données par SFLA (problème 1, paramètres 2).

C_s	Meilleure	Moyenne	Mauvaise
	5,01991820	5.01991842	5,01991916

Les tableaux III.15-III.16 représentent les résultats obtenus par l'algorithme SFLA avec une taille de population égale à 50 et un nombre d'itération maximal de 500. La meilleure valeur de (C_s) est 5.01991820 avec un NFE qui est égale 21900, un CPU de 11,947548s et un σ (2,744631E-07).

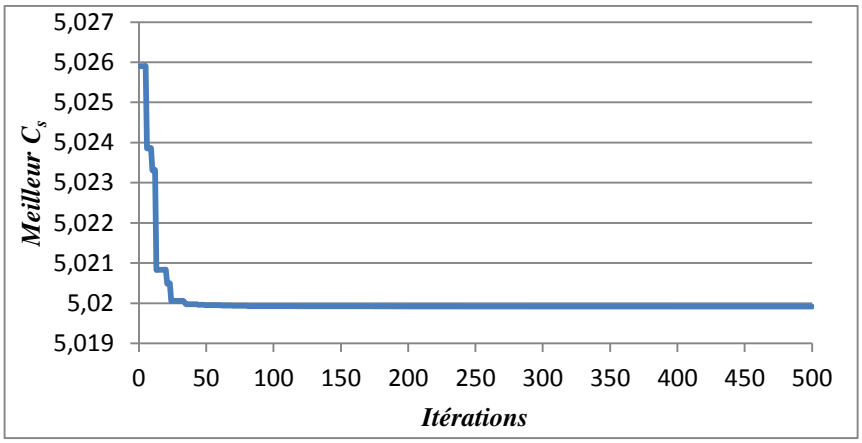

Figure III.8 : Meilleur coût donné par SFLA (problème 1, paramètres 2).

La figure III.8 représente la solution optimale donnée par l'algorithme SFLA (paramètres 2), obtenu au niveau de l'itération 50 après une convergence rapide, avec un meilleur coût de 5,01991820**.**

Tableau III.18 : Résultats donnés par GWO (problème 2, paramètres 1).

No	(n_1, n_2, n_3, n_4)	r_1	r_2	r_3	r_4
1	(5, 5, 5 ,5)	0,900905	0,883902	0,916991	0,885807
2	(5, 6, 4, 5)	0,902711	0,849465	0,947266	0,888343
3	(5, 6, 4, 5)	0,899485	0,850485	0,950000	0,887714
4	(6, 6, 4, 4)	0,855933	0,838400	0,943317	0,918590
5	(4, 5, 5, 6)	0,933511	0,884355	0,914941	0,843409
6	(5, 5, 5, 5)	0,898119	0,887203	0,917829	0,883946
7	(4, 6, 5, 5)	0,933319	0,847107	0,912530	0,883033
8	(5, 6, 4, 5)	0,903969	0,846674	0,947327	0,889139
9	(5, 5, 4, 6)	0,901361	0,888998	0,948468	0,847923
10	(5, 5, 5, 5)	0,899054	0,885028	0,917428	0,885628

Tableau III.19 : *Suite du tableau III.18.*

No	R_{c1}	R_{c2}	R_{c3}	R_s
1	50	0,091035	28.803701	0,99994599363
2	55	0,196068	24.801883	**0,99995455993**
3	55	0,059876	24,801883	0,99995446512
4	62	0,268523	14,221477	0,99991900562
5	37	0,084428	2,206288	0,99994057766
6	50	0,021554	28,803701	0,99994596685
7	37	0,174440	11,644708	0,99994044364
8	55	0,081303	24,801883	0,99995439834
9	55	0,613474	15,363463	0,99995438861
10	50	0,258644	28,803701	0,99994602086

Tableau III.20 : *Suite du tableau III.18.*

No	CPU (s)	NFE	σ
1	0,925297	25000	1,035884E-05
2	0,706927	25000	
3	0,892623	24950	
4	0,847446	24950	
5	0,894149	25000	
6	0,970676	24950	
7	0,773557	25000	
8	0,842636	25000	
9	0,791334	24900	
10	0,785345	25000	

Tableau III.21 : Valeurs de R_s données par GWO (problème 2, paramètres 1).

R_s	Meilleure	Moyenne	Mauvaise
	0,99995455993	0,99994558202	0,99991900562

Les tableaux III.18-III.20 représentent les résultats obtenus par GWO avec une taille de population égale à 50 et un nombre d'itération maximal de 500, où la meilleure solution (R_s) donnée est 0,99995455993, avec un NFE 25000, une consommation de temps (CPU) de 0,706927s et un σ égale à 1,035884E-05.

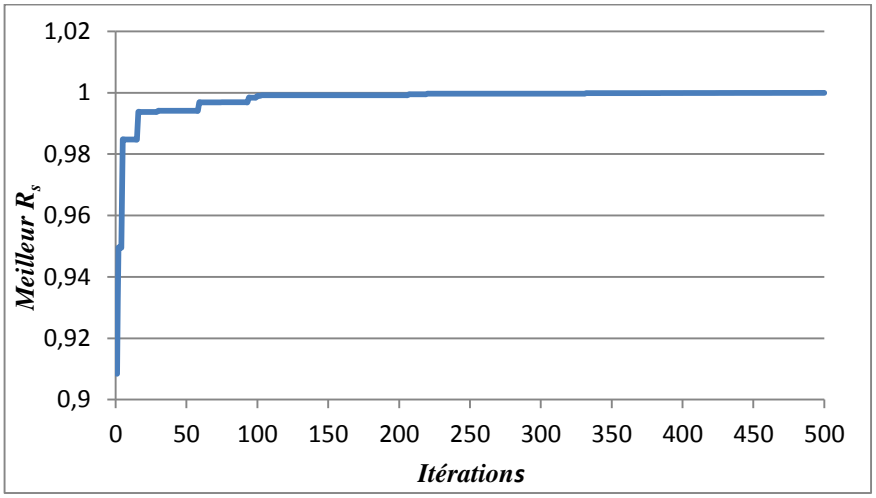

Figure III.9 : Meilleure R_s donnée par GWO (problème 2, paramètres 1).

La figure III.9 représente la solution optimale donnée par l'algorithme GWO (paramètres 1), obtenu au niveau de l'itération 100 après une convergence rapide, avec une meilleure solution de 0,99995455993.

Tableau III.22 : Résultats donnés par GWO (problème 2, paramètres 2).

No	(n_1, n_2, n_3, n_4)	r_1	r_2	r_3	r_4
1	(5, 5, 4, 6)	0,900538	0,887970	0,948363	0,851154
2	(5, 5, 4, 6)	0,901412	0,887837	0,948545	0,850277
3	(5, 6, 4, 5)	0,901917	0,849050	0,948102	0,888570
4	(5, 5, 4, 6)	0,902527	0,888426	0,947300	0,849612
5	(5, 5, 4, 6)	0,901134	0,888492	0,948826	0,849068
6	(4, 6, 5, 5)	0,932964	0,845554	0,914741	0,883942
7	(6, 5, 4, 5)	0,863253	0,884723	0,947629	0,885710
8	(5, 6, 4, 5)	0,901086	0,850792	0,948462	0,887788
9	(5, 5, 4, 6)	0,902528	0,888413	0,947145	0,849811
10	(5, 5, 4, 6)	0,901827	0,887925	0,949187	0,848889

Tableau III.23 : *Suite du tableau III.22.*

No	R_{c1}	R_{c2}	R_{c3}	R_s
1	55	0,002604	15,363463	0,99995463573
2	55	0,011544	15,363463	0,99995465523
3	55	0,007862	24,801883	**0,99995465911**
4	55	0,026254	15,363463	0,99995462923
5	55	0,032416	15,363463	0,99995463515
6	37	0,025117	11,644708	0,99994067403
7	66	0,016788	24,801883	0,99994608247
8	55	0,007660	24,801883	0,99995465135
9	55	0,031391	15,3634633	0,99995462005
10	55	0,002771	15,3634633	0,99995462622

Tableau III.24 : *Suite du tableau III.22.*

No	CPU (s)	NFE	σ
1	2,615925	100000	4,663845E-06
2	2,486809	100000	
3	2,610122	99900	
4	2,335636	100000	
5	2,938650	99900	
6	2,231196	100000	
7	2,273113	99900	
8	2,373993	100000	
9	2,169412	100000	
10	2,222050	100000	

Tableau III.25 : Valeurs de R_s données par GWO (problème 2, paramètres 2).

R_s	Meilleure	Moyenne	Mauvaise
	0,99995465911	0,99995238685	0,99994067403

Les tableaux III.22-III.24 représentent les résultats donnés par GWO avec une taille de population égale 100 et un nombre d'itération maximal de 1000, où la meilleure solution R_s est égale à 0,99995466, avec un NFE (99900), un CPU de 2.610122s et l'écart type qui égale à 4,663845E-06.

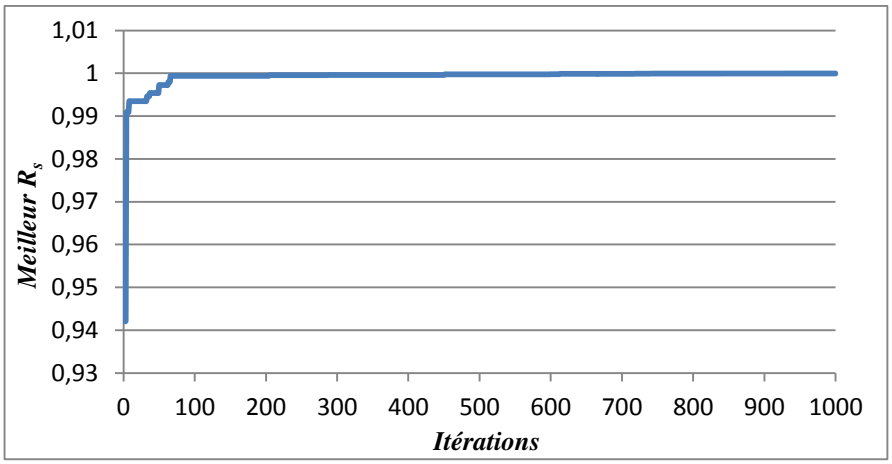

Figure III.10 : Meilleur R_s donnée par GWO (problème 2, paramètres 2).

La figure III.10 représente la meilleure solution de R_s donnée par l'algorithme GWO (paramètres 2). Cette figure illustre une convergence rapide jusqu'à la meilleure solution au niveau de l'itération 100 avec une valeur optimale de 0,99995465911.

Tableau III.26 : Résultats donnés par SFLA (problème 2, paramètres 1).

No	(n_1, n_2, n_3, n_4)	r_1	r_2	r_3	r_4
1	(5, 5, 4, 6)	0,901669	0,888148	0,948171	0,849954
2	(5, 5, 4, 6)	0,901395	0,888207	0,948255	0,850032
3	(5, 5, 4, 6)	0,901688	0,888190	0,948175	0,849858
4	(5, 5, 4, 6)	0,901675	0,888109	0,948235	0,849937
5	(5, 5, 4, 6)	0,901800	0,888165	0,948108	0,849867
6	(5, 5, 5, 5)	0,899236	0,885485	0,916215	0,885545
7	(6, 5, 4, 5)	0,862952	0,885634	0,946693	0,885593
8	(5, 5, 5, 5)	0,899232	0,885593	0,916281	0,885420
9	(5, 5, 4, 6)	0,901669	0,888212	0,948165	0,849853
10	(5, 6, 5, 4)	0,894028	0,839850	0,911917	0,917948

Tableau III.27 : *Suite du tableau III.26.*

No	R_{c1}	R_{c2}	R_{c3}	R_s
1	55	4,629053e-04	15,363463	0,99995467441
2	55	9,589199e-04	15,363463	0,99995467339
3	55	3,260146e-04	15,363463	0,99995467448
4	55	4,349488e-04	15,363463	0,99995467405
5	55	6,897289e-04	15,363463	0,99995467399
6	50	7,191281e-04	28,803701	0,99994615118
7	66	0,001014	24,801883	0,99994613670
8	50	0,003938	28,803701	0,99994615082
9	55	0,002728	15,363463	**0,99995467455**
10	46	0,005278	18,223295	0,99991913949

Tableau III.28 : *Suite du tableau III.26.*

No	CPU (s)	NFE	σ
1	13,885823	24900	1,052303E-05
2	12,794721	24500	
3	15,887117	23600	
4	12,782097	24400	
5	12,289326	24800	
6	12,946034	24800	
7	13,522960	22900	
8	12,449543	25000	
9	16,935067	24850	
10	12,427251	24750	

Tableau III.29 : Valeurs de R_s données par SFLA (problème 2, paramètres 1).

R_s	Meilleure	Moyenne	Mauvaise
	0,99995467455	0,99994856230	0,99991913949

Les tableaux III.26-III.28 représentent les résultats de fiabilité des systèmes donnés par SFLA avec une taille de population égale à 50 et un nombre d'itération maximal de 500, la meilleure solution égale à 0,99995467455. Avec un NFE, CPU et σ (24850 ; 16,935067s ; 1,052303E-05).

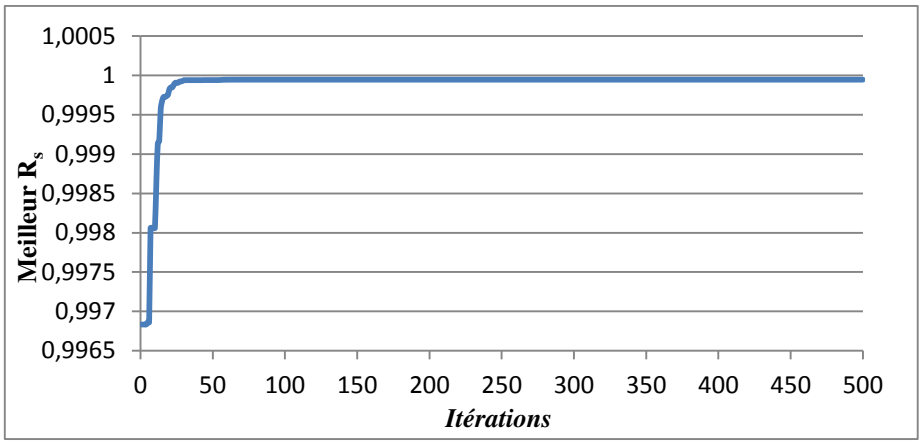

Figure III.11 : Meilleur R_s donnée par SFLA (problème 2, paramètres 1).

La figure III.11 représente R_s optimale donnée par l'algorithme SFLA (paramètres 1). La courbe illustre une convergence rapide jusqu'à la stabilisation à l'itération 50 avec une meilleure solution de 0,99995467455.

Tableau III.30 : Résultats donnés par SFLA (problème 2, paramètres 2).

No	(n_1, n_2, n_3, n_4)	r_1	r_2	r_3	r_4
1	(4, 5, 5, 6)	0,932745	0,884220	0,915347	0,845319
2	(6, 5, 4, 5)	0,862917	0,885617	0,946748	0,885585
3	(5, 5, 4, 6)	0,901623	0,888282	0,948078	0,849888
4	(6, 5, 4, 5)	0,862830	0,885663	0,946752	0,885565
5	(5, 5, 5, 5)	0,899162	0,885452	0,916123	0,885650
6	(5, 5, 4, 6)	0,901627	0,888215	0,948158	0,849901
7	(5, 6, 4, 5)	0,901604	0,850038	0,948029	0,888239
8	(5, 6, 4, 5)	0,901625	0,849936	0,9481749	0,888183
9	(5, 5, 4, 6)	0,901592	0,888234	0,948144	0,849919
10	(5, 5, 5, 5)	0,899246	0,885492	0,916240	0,885525

Tableau III.31 : *Suite du tableau III.30.*

No	R_{c1}	R_{c2}	R_{c3}	R_s
1	37	3,339325 e-04	2,206288	0,99994069260
2	66	3,144835 e-04	24,801883	0,99994613666
3	55	0,001106	15,363463	0,99995467445
4	66	0,001590	24,801883	0,99994613650
5	50	3,177882 e-04	28,803701	0,99994615053
6	55	9,316615 e-04	15,363463	0,99995467464
7	55	0,004250	24,801882	0,99995467403
8	55	0,002238	24,801882	0,99995467460
9	55	0,0041257	15,363463	**0,99995467466**
10	50	0,002991	28,803700	0,99994615443

Tableau III.32 : *Suite du tableau III.30.*

No	CPU (s)	NFE	σ
1	45,511150	92800	5,051297E-06
2	61,208035	93400	
3	65,724051	86300	
4	65,042952	93600	
5	63,912607	74900	
6	72,701637	92100	
7	75,406627	98900	
8	80,002501	84800	
9	74,046595	98300	
10	74,625632	93400	

Tableau III.33 : Valeurs de R_s données par SFLA (problème 2, paramètres 2).

R_s	Meilleure	Moyenne	Mauvaise
	0,99995467466	0,99994986431	0,99994069260

Les tableaux III.30-III.32 représente les résultats donnés par SFLA avec une taille de population égale à 100 et un nombre d'itérations maximal de 1000. La meilleure solution R_S est donnée 0.99995467466, avec un NFE de 98300, un temps d'exécution (CPU) de 74,046595s et σ qui égale à 5,051297E-06.

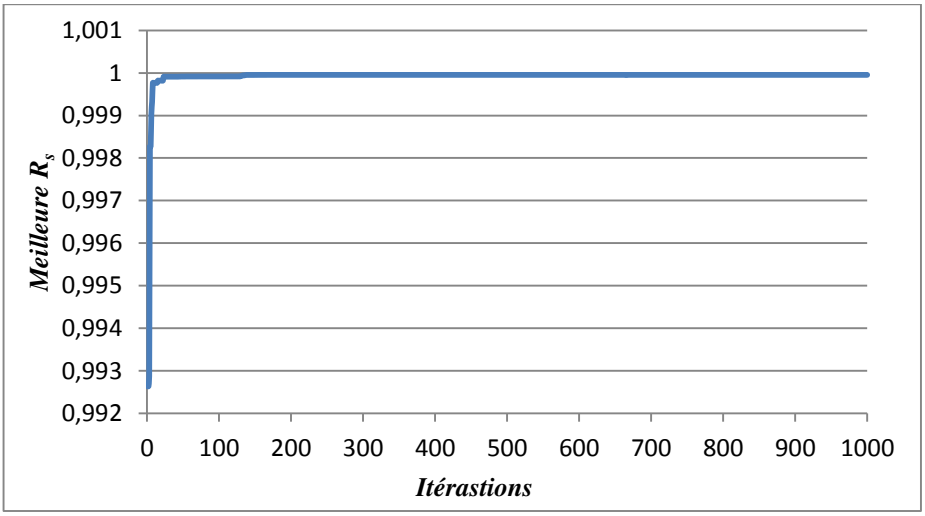

Figure III.12 : Meilleur R_s donnée par SFLA (problème 2, paramètres 2).

La figure III.12 représente R_s optimale donnée par l'algorithme SFLA (paramètres 2). La courbe illustre une convergence rapide jusqu'à la stabilisation au niveau de l'itération 20 avec une meilleure solution de 0,99995467466.

III.6 Comparaison entre les deux algorithmes

Les tableaux III.34-III.36 représentent la comparaison entre les performances des deux algorithmes (GWO et SFLA) dans le cas des deux problèmes.

Tableau III.34 : Comparaison dans le problème 1 (GWO et SFLA).

	C_s	NFE	CPU(s)	σ
GWO	5,01994055	24900	**0,480275**	1,346793E-04
SFLA	**5,01991820**	**21900**	12,019701	**2,744631E-07**

A partir du tableau III.34, on peut constater que SFLA a fourni le meilleur résultat (C_s=5,01991820) en consommant encore le moins de NFE (21900) et le moins de σ (2,744631E-07) par rapport à GWO (C_s=5,01994055 ; 24900 ; 2,744631E-07). En revanche, le GWO a consommé moins de CPU (0,031055s) et SFLA (1,360631s).

Les figures III.13-III.14 représentent la consommation entre SFLA et GWO en fonction de CPU et NFE dans le cas du problème 1.

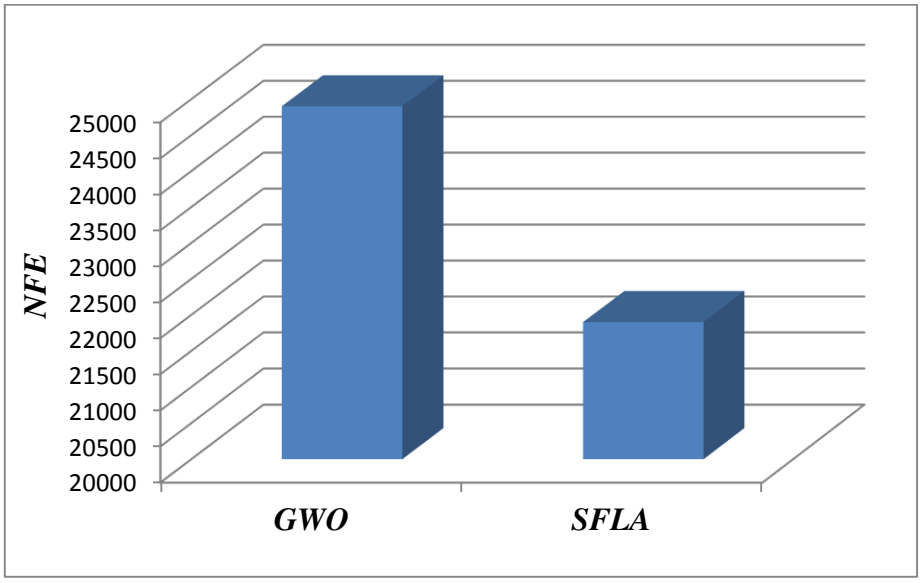

Figure III.13 : NFE consommé par GWO et SFLA (problème 1).

Figure III.14 : CPU consommé par GWO et SFLA (problème 1).

Tableau III.35 : Comparaison dans le problème 2 (GWO et SFLA).

	R_s	NFE	CPU	σ
GWO	0,99995465911	99900	**2,610122**	**4,66384E-06**
SFLA	**0,99995467466**	**98300**	74,046595	5,05129E-06

D'après le tableau III.11, on peut observer qu'encore une fois le SFLA a fourni le meilleur résultat (R_s=0,99995467466) en consommant le moins de NFE (98300) par rapport à GWO (R_s=0,99995465911, 9900). En revanche, GWO a consommé le moins de CPU (2.610122s) et le moins de σ 4,663845E-06 par rapport SFLA (74,046595s, 5,051297E-06).

Les figures III.15-III.16 représentent la consommation entre SFLA et GWO en fonction de CPU et NFE dans le cas du problème 2.

Figure III.15 : NFE consommé par GWO et SFLA (problème 2).

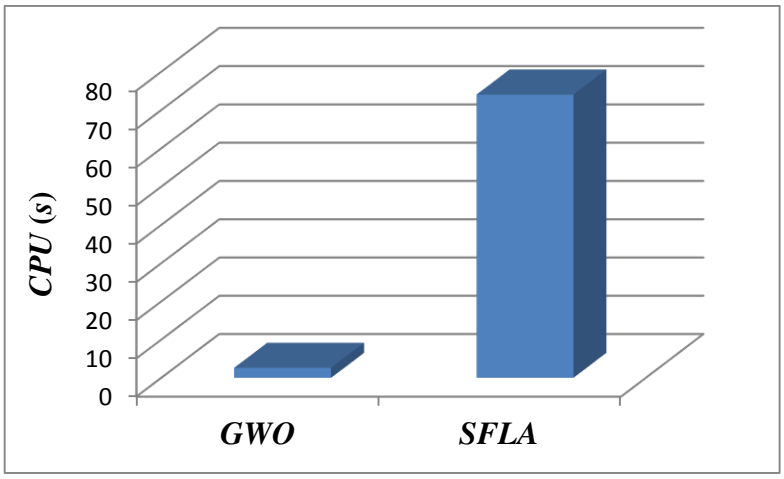

Figure III.16 : CPU consommé par GWO et SFLA (problème 2).

Tableau III.36 : Meilleurs résultats du problème 1.

Méthode	C_s	NFE	CPU	σ
ABC[3] [43]	5,01991875	20000	-	7,5105E-06
MRPS[4] [46]	5,01992	-	-	-
I-NESA[5][42]	5,01993	100000	-	-
FGO[7] [41]	5,02042	-	-	-
GWO[6]	5,01994055	24900	0,480275	1,346793E-04
SFLA[2]	5,01991820	21900	12,019701	2,744631E-07
PSFS[1] [16]	**5,019918127360**	**5000**	-	**1,6E-11**

Le tableau III.36 représente une récapitulation par rapport aux résultats trouvés dans ce travail (GWO et SFLA) ainsi que d'autres travaux de la littérature, tels que l'ABC, MRPS, I-NESA, FGO, et PSFS. On peut constater que la solution obtenue par SFLA est prometteuse.

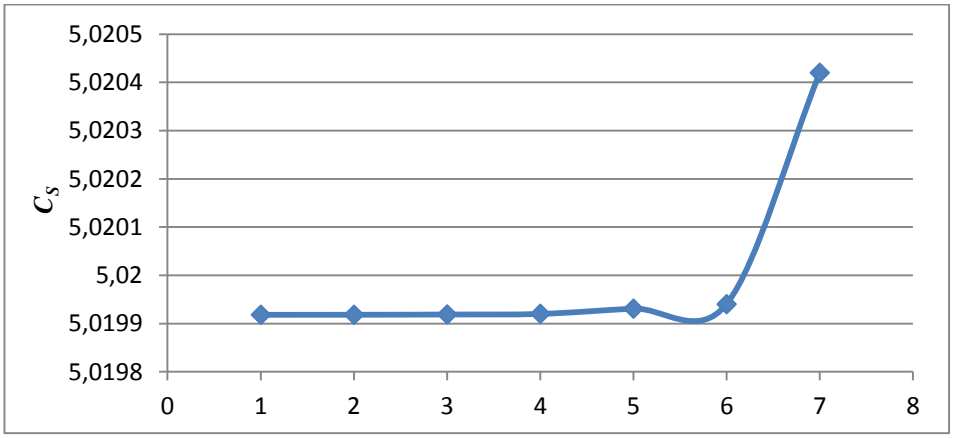

Figure III.17 : Courbe des meilleurs résultats (problème 1).

Tableau III.37 : Meilleur résultat du problème 2.

Réf	R_s	NFE	CPU	σ
PSO[8][47]	0,99990474	100.000	-	-
INGHS[5] [48]	0,9999546745	75000	-	1,5221E-05
CS[4] [21]	0,999954674585	-	-	6,9761E-09
GWO[7]	0,99995465911	99900	2,610122	4,6638E-06
SFLA[3]	0,99995467466	98300	74,04659	5,0512E-06
NAFSA[2] [2]	0,99995467467	50000	-	4,43E-06
PSFS[1] [16]	**0,99995467467678**	**5000**	-	**2,8E-20**
IPSO[6] [49]	0,999954674264402	13500	-	1,3895E-05

Le tableau III.37 représente une comparaison des résultats trouvés dans ce travail (GWO et SFLA) par rapport aux autres travaux de la littérature, tels que PSO, INGHS, CS, NFSA, IPSO et PSFS.

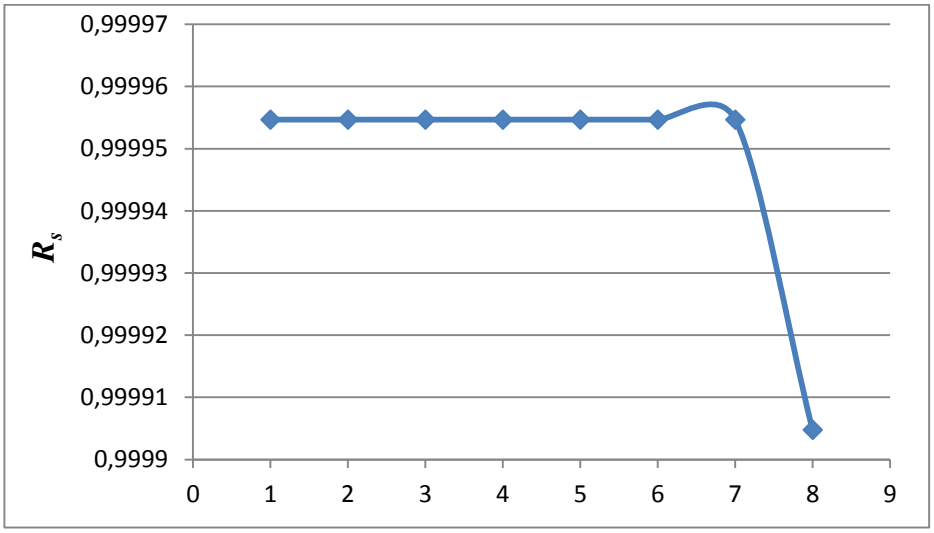

Figure III.18 : Courbe des meilleurs résultats (problème 2).

III.7 Conclusion

Dans ce chapitre, deux algorithmes bio-inspirés ont été proposé pour résoudre deux problèmes de l'optimisation de la fiabilité. Les résultats ont montré que SFLA présente de meilleures solutions par rapport au GWO dans les deux problèmes. En revanche, le GWO a consommé le moins de temps. En outre, les résultats obtenus dans ce chapitre sont prometteurs par rapport à la littérature.

Conclusion générale

Conclusion générale

Dans ce mémoire, nous avons optimisé deux systèmes de fiabilité : un système réseau en pont complexe et un système de protection de survitesse (turbine à gaz). On a débuté ce travail par une explication des principaux éléments de la sûreté de fonctionnement (SdF) qui jouent un rôle très efficace dans l'industrie afin d'éviter les défaillances, et la maintenance corrective coûteuse qui décrois le rendement dans l'usine industrielle. La fiabilité occupe une tâche tellement essentielle dans la sûreté de fonctionnement des systèmes quelle que soit sa phase de conception, maintenance ou les systèmes mécatroniques ainsi que les systèmes complexes. Ce chapitre a fourni aussi un état de l'art sur l'optimisation de la fiabilité des systèmes.

Dans le deuxième chapitre, nous avons présenté les différents types d'optimisation et leurs méthodes de résolution telles que les méthodes conventionnelles (méthode mathématiques classiques) et non conventionnelles (méthodes de l'intelligence artificielle).

Le troisième chapitre a traité deux applications des problèmes de fiabilité à l'aide de deux algorithmes bio-inspirés (SFLA) et (GWO), dont le but est de les implémenter et de faire une comparaison entre les résultats. Ces derniers ont montré que le SFLA a fourni les meilleures performances. En revanche, le GWO a consommé moins de temps. Les résultats obtenus ont concurrencé ceux de la littérature. La combinaison des deux algorithmes est un projet intéressant pour l'appliquer au future afin d'obtenir de meilleures performances.

Bibliographie

[1] S. Khalfaoui, “Méthode de recherche des scénarios redoutés pour l’évaluation de la sûreté de fonctionnement des systèmes mécatroniques du monde automobile,” Thése de doctorat, Institut National Polytechnique de Toulouse, France, 2003.

[2] Q. He, X. Hu, H. Ren, and H. Zhang, “A novel artificial fish swarm algorithm for solving large-scale reliability-redundancy application problem,” ISA Trans., vol. 59, pp. 105–113, 2015.

[3] G. Zwingelstein, “Sûreté de fonctionnement principaux concepts,” Techniques de l’Ingénieur, Référence S 8 250v3, pp. 1–30, 2019.

[4] Y. Mortureux, “La sûreté de fonctionnement: démarches pour maîtriser les risques,” Techniques de l’Ingénieur, Référence BM 5008, pp. 1–9, 2002.

[5] A. G. Mihalache, “Modélisation et évaluation de la fiabilité des systèmes mécatroniques : application sur systéme embarqué” Thèse de doctorat, Université d’Angers, France, 2007.

[6] R.Laronde, “Fiabilité et durabilité d’un système complexe dédié aux énergies renouvelables - Application à un système photovoltaïque,” Thèse de doctorat, Université d’Angers, France, 2011.

[7] D. Noyes et F. Pérès, “Analyse des systèmes sûreté de fonctionnement,” Techniques de l’Ingénieur, Référence AG 3520, pp. 1–14, 2013.

[8] F. Bicking, “Allocation de la fiabilité par algorithme génétique : application à la conception d'un système instrumenté de sécurité,” Thése de doctorat, Université de Nancy, Françe, 2009.

[9] E. Zio, “Series on quality, reliability and engineering statistics,” in An introduction to the basics of rellablllty rho risk analysis, World Scientific, Italy, 2006.

[10] I. Gueye, “Création de bases de connaissances interconnectées-institut de formation / entreprise - par la capitalisation des connaissances en maintenance industrielle,” Thèse de doctorat, Université de Bordeaux, France, 2018.

[11] A. Despujols, “Maintenance, sûreté de fonctionnement et management des actifs de production,” Techniques de l’Ingénieur, Référnce MT9202 v1, pp. 1–16,

2009.

[12] Y. Brechet, M. F. Ashby, M.Dupeux, F.Louchet, “Choix et usinage des matériaux,” Techniques de l’Ingénieur, Référence T5100 v1, 1996.

[13] Y. Santhosh Kumar Reddy, “Reliability engineering basics and optimization techniques,” Lecture Notes, Indian Electronic Theses & Dissertations, 2014.

[14] P. Devalan, “Introduction à la mécatronique,” Techniques de l’Ingénieur, Référence BM8000 v1, 2010.

[15] G. Zwingelstein, “Sûreté de fonctionnement des systèmes industriels complexes,” Techniques de l’Ingénieur, Référence S 8 250v2, pp. 1–23, 2013.

[16] M. A. Mellal and E. Zio, “A penalty guided stochastic fractal search approach for system reliability optimization,” Reliab. Eng. Syst. Saf., vol. 152, pp. 213–227, 2016.

[17] H. A. Khorshidi, I. Gunawan, et M. Y. Ibrahim, “Investigation on system reliability optimization based on classification of criteria,” Proc. IEEE Int. Conf. Ind. Technol., Cape Town, South Africa, pp. 1706–1711, 2013.

[18] D. Crowe and A. Feinberg, “Design for reliability,” Des. Reliab., CRC Press, USA, pp. 1–558, 2017.

[19] M. Agarwal and V. K. Sharma, “Ant colony approach to constrained redundancy optimization in binary systems,” Appl. Math. Model., vol. 34, no. 4, pp. 992–1003, 2010.

[20] D. Zou, H. Liu, L. Gao, and S. Li, “A novel modified differential evolution algorithm for constrained optimization problems,” Comput. Math. with Appl., vol. 61, no. 6, pp. 1608–1623, 2011.

[21] H. Garg, “An approach for solving constrained reliability-redundancy allocation problems using cuckoo search algorithm,” Beni-Suef Univ. J. Basic Appl. Sci., vol. 4, no. 1, pp. 14–25, 2015.

[22] M. A. Mellal and E. J. Williams, The cuckoo optimization algorithm and its applications, 1st ed, Elsevier, USA, pp. 269–277, 2017.

[23] M. A. Mellal and E. Zio, “An adaptive particle swarm optimization method for multi-objective system reliability optimization,” Proc. Inst. Mech. Eng. Part O J.

Risk Reliab., 2019. DOI: 10.1177/1748006X198552814

[24] Cours sur l'optimisation, Ecole catholique, 2011.

[25] P. Chootinan et A. Chen, "Constraint handling in genetic algorithms using a gradient-based repair method," Comput. Oper. Res., vol. 33, no. 8, pp. 2263–2281, 2006.

[26] S. Boyd et L. Vandenberghe, "Convex optimization," Lecture Notes, Stanford University, USA, 2009.

[27] G. Pita Gil, "Application de techniques de commande avancées dans le domaine automobile," Thèse de doctorat, Supélec, France, 2011.

[28] S. Jacquin, "Hybridation des métaheuristiques et de la programmation dynamique pour les problèmes d'optimisation mono et multi-objectif : Application à la production d'énergie," Thèse de doctorat, Université de Lille1, France, 2015.

[29] M. Hamed, "Dispatching economique dynamique par utilisation de méthodes d'optimisation Globales," Mémoire de Master, Université Mohamed Khider, Biskra, 2013.

[30] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," Adv. Eng. Softw., vol. 69, pp. 46–61, 2014.

[31] M. Karakoyun et A. Babalik, "Data clustering with shuffled leaping frog algorithm (SFLA) for classification," *Int'l Conference on Intelligent Computing, Electronics Systems and Information Technology*, Kuala Lumpur, Malaysia, 2015.

[32] D. Ilea, "Conception optimale des moteurs à réluctance variable à commutation électronique pour la traction des véhicules électriques légers," Thèse de doctorat, Ecole centrale de Lille, France, 2011.

[33] M. A. Mellal and E. J. Williams, "A survey on ant colony optimization, particle swarm optimization, and cuckoo algorithms," IGI Global, USA, pp. 37–51, 2017.

[34] N. Monmarché, "Algorithmes de fourmis artificielles : applications à la classification et à l'optimisation," Thèse de doctorat, Université François Rabelais Tours, France, 2000.

[35] S. Mirjalili, "The ant lion optimizer," Adv. Eng. Softw., vol. 83, pp. 80–98, 2015.

[36] J. Nasiri and F. M. Khiyabani, “A whale optimization algorithm (WOA) approach for clustering,” Cogent Math. Stat., vol. 5, no. 1, pp. 1–13, 2018.

[37] H. Garg, “A hybrid GSA-GA algorithm for constrained optimization problems,” Inf. Sci. (Ny)., vol. 478, pp. 499–523, 2019.

[38] X. S. Yang, “Flower pollination algorithm for global optimization,” Springer, Germany, pp. 240–249, 2012.

[39] H. Garg, “An efficient biogeography based optimization algorithm for solving reliability optimization problems,” Swarm Evol. Comput., vol. 24, pp. 1–10, 2015.

[40] E. Zio, “Reliability engineering : Old problems and new challenges,” Reliab. Eng. Syst. Saf., vol. 94, pp. 125–141, 2009.

[41] V. Ravi and P. J. Reddy, “Fuzzy global optimization of complex system reliability,” vol. 8, no. 3, pp. 241–248, 2000.

[42] V. Ravi, “Nonequilibrium simulated annealing-algorithm applied to reliability optimization of complex systems,” vol. 46, no. 2, 1997.

[43] H. Garg, M. Rani, et S. P. Sharma, “Computers & operations research an ef fi cient two phase approach for solving reliability – redundancy allocation problem using arti fi cial bee colony technique,” Comput. Oper. Res., vol. 40, no. 12, pp. 2961–2969, 2013.

[44] Y. C. Hsieh and P. S. You, “An effective immune based two-phase approach for the optimal reliability-redundancy allocation problem,” Appl. Math. Comput., vol. 218, no. 4, pp. 1297–1307, 2011.

[45] L. dos S. Coelho, “An efficient particle swarm approach for mixed-integer programming in reliability-redundancy optimization applications,” Reliab. Eng. Syst. Saf., vol. 94, no. 4, pp. 830–837, 2009.

[46] C. Centre, “Short communication reliability optimization of complex systems using modified random-to-pattern search (MRPS),” vol. 243, no. November 1998, pp. 239–243, 1999.

[47] C. L. Huang, “A particle-based simplified swarm optimization algorithm for reliability redundancy allocation problems,” Reliab. Eng. Syst. Saf., vol. 142, pp.

221–230, 2015.

[48] H. Bin Ouyang, L. Q. Gao, S. Li, and X. Y. Kong, “Improved novel global harmony search with a new relaxation method for reliability optimization problems,” Inf. Sci. (Ny)., vol. 305, pp. 14–55, 2015.

[49] P. Wu, L. Gao, D. Zou, and S. Li, “An improved particle swarm optimization algorithm for reliability problems,” ISA Trans., vol. 50, no. 1, pp. 71–81, 2011.