الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي

Université SAAD DAHLAB BLIDA

Faculté des Sciences

Département de Mathématiques

MEMOIRE DE MASTER

En mathématiques

Spécialité : Recherche opérationnelle

Thème :

Optimisation multi-objectifs avec l'algorithme NSGA II

Par :

Insaf Zekour

Devant le jury composé de :

Président de jury : Tami Omar MCB Blida - 1-

Examinateur : Boukhari Mohamed MAA Blida - 1-

Promoteur : Sebaa Karim Professeur Médéa

2021 / 2022

Remerciements

الحمد لله و الشكر له كما ينبغي لجلال وجهه و عظيم سلطانه ، عدد خلقه و رضا نفسه و زنة عرشه و مداد كلماته على أن من علي بإنجاز هذه الدراسة، و الصلاة و السلام على أفضل خلق الله نبيينا محمد عليه أفضل الصلاة و التسليم.

أتوجه بالشكر أولا لوالدي "شوقي زكور "و والدتي "أمال باشا" و أخواتي و أخي و عمي "مختار" .

أشكر أستاذ المكون "سبع كريم"الذي تفضل بالإشراف على هذا البحث و جميع أساتذة التخصص على الجهد المبذول لأجلنا، و إلى زملائي في التخصص : غفران، نسرين، سيليا، ياسمين...و الجميع شكرا

الشكر أيضا موصول إلى كل من دعمني من قريب أو بعيد أختص بالذكر : زكرياء، مراد،أيوب و نورالدين، هدى، جيهان، سيرين ....و عذرا لمن نسيت ذكرهم لكن شكرا.

الشكر الخاص للأستاذة: بوديسة و الأستاذ عبدلي على المساعدة العظيمة جزآهم الله ألف خير

أهدي تعبي لعائلة زكور لأعمام: ناجي، هشام و رياض و عمتي: آسيا و لكل من أزواجهم و أولادهم و لعائلة باشا لأخوالي: عبد النور،سمير و حليم و خالتي الحبيبة سامية و زهور و أولادها و شكر خاص لتيتا حبيبتي.

و جزيل الشكر و العرفان و أقدرني الله على رد معروفهم "محمد لوناس" الذي أهلكته معي في البرمجة و "بلال تمار " الذي ساعدني في التحرير جزيل الشكر و التقدير للجميع.

أرزاق الله كثيرة و رزقي كان في عبادها الصالحين رب أدمها علي نعمة و أوزعني أن أشكر نعمتك التي أنعمت علي وأن أعمل صالحا ترضاه.

تم بحمد الله

إنصاف زكور

Dédicace

بعد الحمد لله الذي تتم بنعمته الصالحات،

أهدي هذه المذكرة إلى كل من أمي التي غرست فينا حب العلم و طلبه،

إلى أبي الذي عمل جاهدا كي لا ينقصنا شيء وأن لا نطلب من غيره شيئا،

إلى أخواتي نجلاء،منار،إيناس و رميلة،و أخي الوحيد ياسر أشكرهم على حسن تعاملهم مع أيام التحرير،

إلى مريم بولغبار و مريم فضيل، صديقتي عمري و إلى أفضل ابنة خال لي صفاء باشا

إلى أطيب عمة قد تحصل عليها أية ابنة أخ في العالم "وهيبة زكور "و زوجها "سليمي محمد "رحمة الله عليهما

إلى جدتي الحبيبتين اللتان اتفقتا على الرحيل عنا في الفترة عينها "سمية جيكرف"و "كوشران فاطمة الزهراء"

أهدي هذا التعب إلى كل أساتذة التخصص و زملاء في الدراسة و إلى كل زملائي في النوادي العلمية

أشكركم جميعا و أعانني الله على رد جميلكم

إنصاف زكور

Résumé

L'optimisation multi-objectif est l'une des disciplines de la programmation mathématique traitant l'optimisation de multiples fonctions objectifs généralement contradictoires i.e. où la minimisation d'un objectif entraine la maximisation d'un autre.

Nous essayerons dans ce mémoire d'étudier l'un des outils les plus utilisé dans ce sens,
L'algorithme NSGA **II** (no dominated sorting Genetric Algorithms) reconnu pour son succès dans la résolution et l'analyse des compromis de plusieurs problèmes dans le monde de l'ingénierie.

Mot clefs : NSGA **II**, métaheuristique, Matlab(r).

ملخص

يعد التحسين الرياضيإتي متعدد الأهداف أحد تخصصات البرمجة الرياضي تيق التي تتعامل مع تحسين وظائف الأهداف المتعددة المتناقضة عمومًا ، أين يؤدي تقليل قيمة حل واحد إلى تعظيم قيمة حل آخر بالمقابل.

سنحاول في هذه الأطروحة دراسة أحد أكثر الأدوات استخدامًا في هذا الاتجاه ، والتعرف على خوارزمية NSGA **II** (خوارزميات جينية بدون فرز مهيمن) خوارزميق عرفت بنجاحها في حل وتحليل المفاضلات للعديد من المشاكل المتعددة الأهداف في عالم الهندسة.

الكلمات المفتاحية: NSGA **II**، métaheuristique، Matlab (r).

ABSTRACT

Multi-objective optimization is one of the disciplines of mathematical programming dealing with the optimization of multiple objective functions that are generally contradictory i.e. where the minimization of one objective leads to the maximization of another.

We will try in this thesis to study one of the most used tools in this direction,
The NSGA **II** (no dominated sorting Genetic Algorithms) algorithm is recognized for its success in solving and analyzing the trade-offs of several problems in the engineering world.

Keywords: NSGA **II**, métaheuristique, Matlab(r).

Tables des matières

Sommaire

Liste des figures

Liste des tableaux

Introduction générale

L’optimisation est une branche des mathématiques et de l’informatique cherchant à modéliser, à analyser et à résoudre analytiquement ou numériquement les problèmes qui consistent à déterminer quelle sont la ou les solution(s) satisfaisant un objectif quantitatif tout en respectant d’éventuelle contrainte.

Pour déterminer si une solution est meilleure qu'une autre, il est nécessaire que le problème introduise un critère de comparaison. Ainsi, la meilleure solution, appelée aussi solution optimale, est la solution ayant obtenu la meilleure évaluation au regard du critère défini.

Les problèmes d'optimisation sont utilisés pour modéliser de nombreux problèmes appliqués dans la conception des systèmes de l’industrie, le traitement d'images, les problèmes rencontrés dans les réseaux industriels, la conception d'emplois du temps . . . etc.

La majorité de ces problèmes sont qualifiés de difficiles, car leur résolution nécessite l'utilisation des algorithmes évolués. Il n'est en général pas possible de fournir dans tous les cas une solution optimale dans un temps raisonnable.

Lorsqu'un seul critère est donné par exemple, un critère de minimisation de coût, la solution optimale est clairement définie. C'est celle qui a le coût minimal.

La résolution des problèmes d’optimisation est assez délicate puisque le nombre fini et ou dénombrable et ou infini de solutions réalisables croît généralement avec la taille du problème, ainsi que sa complexité. Cela a poussé les chercheurs à développer de nombreuses méthodes de résolution en recherche opérationnelle (RO). Ces approches de résolution peuvent être classées en deux catégories : les méthodes exactes et les méthodes approchées. Les méthodes exactes gagnent en optimalité des solutions et perdent en temps d’exécution. à l’inverse les méthodes approchées qui ne garantissent pas de trouver une solution exacte, mais seulement une approximation en des temps raisonnables de calcul.

Mais dans de nombreuses situations, un seul critère peut être insuffisant. En effet, la plupart des applications traitées intègrent plusieurs critères simultanés, souvent contradictoires. Intégrer des critères contradictoires pose un réel problème. La solution idéale n'existe pas, et il faut donc trouver un compromis. En effet, en considérant deux critères contradictoires **a** et **b**, améliorer **a** détériore forcément **b** et inversement. Le concept de solution optimale devient alors plus difficile à définir. Dans ce cas, la solution optimale cherchée n'est plus un point unique, mais un ensemble de compromis. Résoudre un problème comprenant plusieurs

critères, appelé communément problème multi objectifs, consiste donc à calculer le meilleur ensemble de solutions de compromis appelé « les frontières de Pareto ».

Comme tout problème multi objectif, trouver une solution repose sur l'existence d'une solution qui optimise simultanément les différents objectifs du problème. Dans le domaine d'optimisation multi objectif, la relation de comparaison entre les solutions est appelée relation de « Dominance de Pareto ». Cette relation permet d'évaluer une solution et de conclure si elle est efficace ou non. Une solution est qualifiée comme non optimale, s'il existe une autre dans l'espace réalisable qui est meilleure que cette solution pour tous les objectifs.

L'exploitation de la relation de dominance de Pareto par les méthodes d'optimisation multi objectif a donnée de meilleurs résultats par rapport aux autres méthodes exactes n'utilisant pas ce concept. Une méthode exacte est une méthode utilisant une résolution exacte pour trouver une solution à un problème d'optimisation multi objectif. Par exemple, la méthode de la somme pondérée vise à transformer un problème multi objectif vers un problème mono-objectif en utilisant la somme des objectifs avec des poids d'importance associés à chaque objectif. Il est clair que cette méthode trouve une solution de type exact pour un problème multi objectif donné qui est une solution exacte mais ces méthodes soufrent de plusieurs problèmes. Si nous prenons un problème non-convexe, la méthode de la somme pondérée va perdre beaucoup de solutions qui peuvent être très utiles dans le processus d'aide à la décision. Cet inconvénient a été surpassé par l'utilisation des méthodes d'optimisation qui utilisent les métaheuristiques.

Les métaheuristiques sont généralement des algorithmes stochastiques itératifs, qui progressent vers un optimum global. Elles se comportent comme des algorithmes de recherche tentant d'apprendre les caractéristiques d'un problème de trouver une approximation de la meilleure solution.
Parmi les métaheuristiques les plus populaires dans le domaine d'optimisation multi objectif, on retrouve les algorithmes basés sur des populations de solutions (type algorithmes génétiques ou algorithmes évolutionnaires), en opposition avec les algorithmes à une seule solution (tels que le recuit simulé, la descente de gradient, la recherche tabou, etc.).

Le présent travail s'intéresse à l'étude de l'un des outils les plus utilisé dans ce sens, une métaheuristique puissante, l'algorithme NSGA-II (non dominated sorting Genetric

Algorithms) reconnu pour son succès dans la résolution et l'analyse des compromis de plusieurs problèmes dans le monde de l'ingénierie. L'objectif principal poursuivi dans le présent travail consiste à prouver l'efficacité d'extraction de l'ensemble Pareto par l'utilisation de l'algorithme NSGA-II pour l'appliquer aux fonctions tests qui sont donc choisis parmi plusieurs études antérieures importantes dans ce domaine.

Pour ce faire, notre projet se compose de trois chapitres.

Le chapitre-I porte sur les principales notions d'optimisation, méthodes classiques, méthodes méta heuristiques...

Au chapitre-II nous nous somme étalés sur les Algorithmes génétiques, les méthodes de codage, de sélection, de croisement et de mutation ainsi qu'aux tests d'arrêts. Ainsi, un exemple d'optimisation mono objective sera donné.

Il sera question au chapitre-III de l'optimisation multi-objectif, des notions de dominance, de la distance "crow ding distance, du fonctionnement du NSGA-II.

Ce chapitre se terminera par les résultats des fonctions tests de l'efficacité de Pareto.

Ce travail sera clôturé par une conclusion générale.

Chapitre I : les notions d'optimisation

L'optimisation c'est l'art de comprendre un problème réel, de pouvoir le transformer en un modèle mathématique que l'on peut étudier afin d'en extraire les propriétés structurelles et de caractériser les solutions du problème. Enfin, c'est l'art d'exploiter cette caractérisation afin de déterminer des algorithmes qui les calculent mais aussi de mettre en évidence les limites sur l'efficience et l'efficacité de ces algorithmes.

1. *Modélisation d'un problème d'optimisation :*

Quoi que soit la spécialité, l'étude et la résolution d'un problème commence toujours par la modélisation de celui-ci. Les problèmes d'optimisation n'échappent pas à cette étape et sont modélisés par (l'espace d'état, les variables, l'ensemble des contraintes et les fonctions objectifs)

Fonction objectif :

C'est le nom donné à la fonction f aussi appelée fonction de coût ou critère d'optimisation. f représente le but à atteindre pour le décideur, c'est cette fonction que l'algorithme d'optimisation va devoir « optimiser » (trouver un optimum : minimiser de coût, de durée, d'erreur ou maximiser profit …….etc.)

f (ou plusieurs) appelée aussi fonction de fitness qui pour toute solution $s \in S$ affecte une valeur fs appelée valeur de fitness. La valeur de la fitness représente la qualité de la solution et définit un espace de solutions potentielles au problème.

Variables :

Les variables du problème peuvent être de nature diverse (réelle, entière, booléenne…….etc.) Et exprimer des données qualitatives ou quantitatives.

L'espace d'état

L'espace d'état est défini par un ensemble des domaines de définition des variables du problème. Dans la plupart des problèmes, cet espace est fini car la méthode de résolution utilisée a besoin de travailler dans un espace restreint.

Ensemble des contraintes :

L'ensemble des contraintes définit sur l'espace d'état que les variables doivent satisfaire.
Ces contraintes sont linéaires ou non-linéaires entre eux, ils sont souvent des contraintes d'égalité ou d'inégalité et permettent en général de limiter l'espace de recherche.

2. *Les différents minima :*

Minimum global :

Soit $\mathbf{X^* \in D}$ tel que **D** est l'ensemble des domaines de définition des variables de la fonction objectif ***f,*** on dit que **X*** est un minimum global de ***f*** SSI :

$$\forall \mathbf{X \in D \text{ et } X \neq X^*} \quad , \quad f(\mathbf{X^*}) < f(\mathbf{X})$$

Minimum local fort :

Soit $\mathbf{X^* \in D}$ tel que **D** est l'ensemble des domaines de définition des variables de la fonction objectif ***f*** **,** et $\mathbf{X \in V(X^*)}$ tel que **V(X*)** est l'ensemble des voisinages de **X*** **,** on dit que **X*** est un minimum local fort de ***f*** SSI :

$$\forall \mathbf{X \in V(X^*) \text{ et } X \neq X^*} \quad , \quad f(\mathbf{X^*}) < f(\mathbf{X})$$

Minimum local faible :

Soit $\mathbf{X^* \in D}$ tel que **D** est l'ensemble des domaines de définition des variables de la fonction objectif ***f*** **,** et $\mathbf{X \in V(X^*)}$ tel que **V(X*)** est l'ensemble des voisinages de **X*** **,** on dit que **X*** est un minimum local fort de ***f*** SSI **:**

$$\forall \mathbf{X \in V(X^*) \text{ et } X \neq X^*} \quad , \quad f(\mathbf{X^*}) \leq f(\mathbf{X})$$

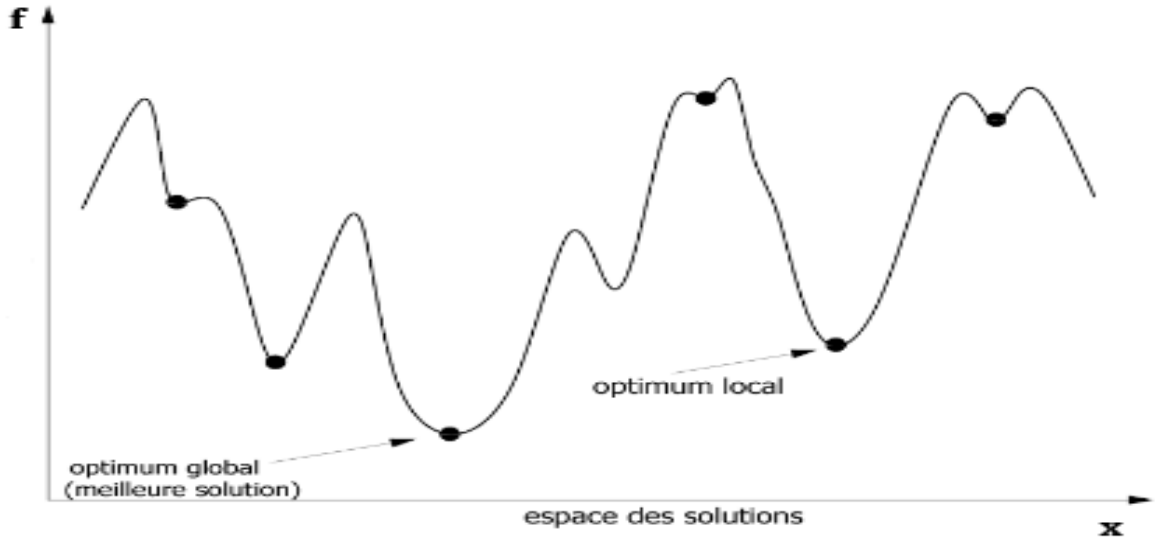


Figure 1: Les différents Minima.

3. *Résolution d'un problème d'optimisation :*

La résolution d'un problème d'optimisation consiste à maximiser (ou minimiser) une fonction objectif de ***n*** variables de décision soumises à un ensemble de contraintes exprimées sous forme d'équations ou d'inéquations.

La terminologie du problème est due à George B.Dantzig, inventeur de l'algorithme du simplexe (1947) pour la mise en forme mathématique du problème d'optimisation ***(Huguet, 2020-2021)*** :

- **Définir les variables de décision :**
  Ensemble des variables qui régissent la situation à modéliser.

- **Préciser la fonction objectif :**
  fonction mathématique composée des variables de décision qui représente le pr oblème modélisé.
- **Préciser les contraintes du problème :**
  Équations ou inéquations composées des variables de décision.

- **Préciser les paramètres du modèle :**

Ensemble des paramètres (constants) associés aux contraintes et à la fonction objectif qui limitent le modèle réalisable.

4. *Formulation mathématique* :

4.1. ***Fonction Objectif :*** à maximiser ou minimiser

$$f(x) = c_1x_1 + c_2x_2 + c_3x_3 + \dots + c_nx_n$$

4.2. ***Contraintes*****:**

$$a_{1\,1}x_1 + a_{1\,2}x_2 + a_{1\,3}x_3 + \dots + a_{1n}x_n \ (\leq, =, \geq)\ b_1$$
$$a_{2\,1}x_1 + a_{2\,2}x_2 + a_{2\,3}x_3 + \dots + a_{2n}x_n \ (\leq, =, \geq)\ b_2$$
$$a_{m\,1}x_1 + a_{m\,2}x_2 + a_{m\,3}x_3 + \dots + a_{m\,n}x_n \ (\leq, =, \geq)\ b_m$$

4.3. ***Contraintes de non négativité*** **:**

$x_j \geq 0$; $j = 1, 2, 3, \dots n$; **avec**

x_j **variables de décision (inconnues)**

a_{ij} , b_i , c_j **paramètres du programme linéaire**

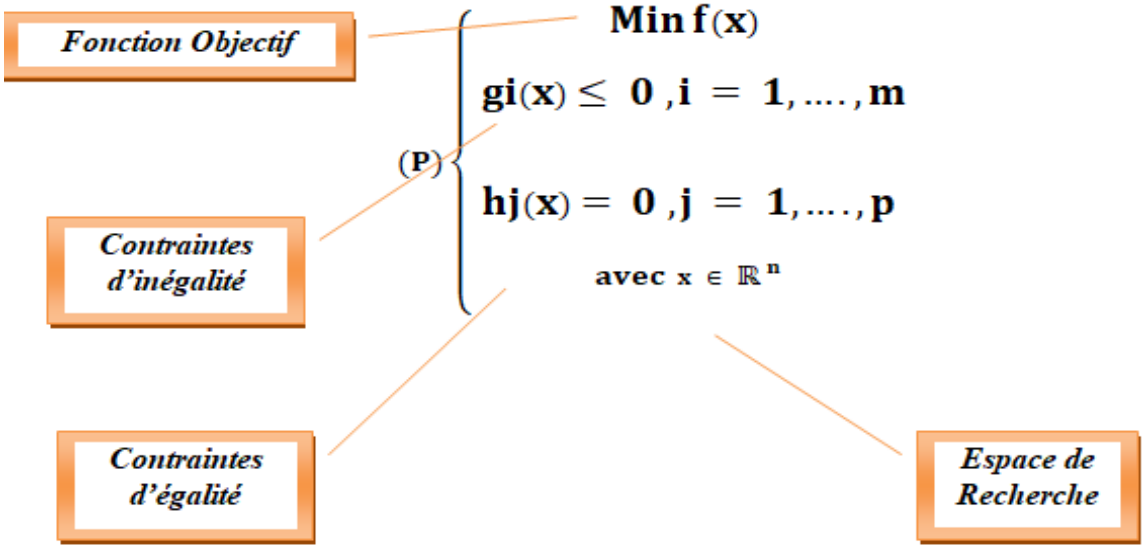


Figure 2 : La présentation mathématique d'un problème d'optimisation.

Après une formalisation (modélisation) efficace du problème d'optimisation et une bonne résolution, nous aurons les résultats suivants :

Solution réalisable : est dite si toutes les contraintes du modèle sont satisfaites.

Zone de solution : est dite s'il n'ya pas une solution réalisable unique mais un ensemble de solutions réalisables.

Solution optimale: est dite pour une solution réalisable où la fonction objectif atteint la meilleure valeur (maximum ou minimum) ***(Kleiner, 2013)***.

Remarque : Plusieurs solutions optimales possibles peuvent être obtenues.

5. *Classification de problèmes :*

Les problèmes d'optimisation sont classés selon la nature de la fonction et le domaine de définitions des variables de décision, le nombre de critères à optimiser, la taille du problème à résoudre,….. etc. Il est donc important de bien identifier à quelle catégorie le problème appartient, afin de choisir la méthode appropriée pour le résoudre, Il existe plusieurs classes d'optimisation, dont nous citons les plus connues ***(Samir, 2021)*** , ***(Zidani, 23 octobre 2013)***:

Problèmes d'optimisation « mono-objectif » ou « multi-objectif » :

C'est une classification basée sur le nombre de fonction objectif : lorsqu'un seul objectif (critère) est considéré, le problème d'optimisation est mono-objectif. Dans ce cas la solution optimale est clairement définie, c'est celle qui a le coût optimal (minimal, maximal).

De manière formelle, à chaque instance d'un tel problème est associé un ensemble δ des solutions potentielles respectant certaines contraintes et une fonction objectif $\boldsymbol{F}$: $\delta \rightarrow \mathbb{R}$ qui associe à chaque solution admissible $x \in \delta$ une valeur $F(x)$. Résoudre l'instance (δ, F) du problème d'optimisation consiste à trouver la solution optimale $x^* \in \delta$ qui optimise (minimise ou maximise) la valeur de la fonction objectif F. en définissant une relation d'ordre total permettant de comparer différentes solutions. Une solution x est meilleure qu'une solution y si $F(x) < F(y)$ dans le cas de minimisation ou $F(x) > F(y)$ dans le cas de maximisation.

Pour le cas de la minimisation : le but est de trouver $x^* \in \delta$ tel que $F(x^*) \leq F(x)$ pour tout élément $x \in \delta$ Un problème de maximisation peut être défini de manière similaire $(F(x^*) \geq F(x), \forall\, x \in \delta)$.

Notons que la maximisation d'une fonction F(x), peut facilement être transformée en un problème de minimisation : $\max F(x) = -\min(-F(x))$ est vise versa.
Un problème multi-objectif est défini par la recherche d'un compromis entre plusieurs fonctions objectifs contradictoires.
Les problèmes d'optimisation multi-objectif sont plus difficiles à résoudre, ceux comportant plus de trois objectifs sont appelés problèmes d'optimisation "Many-objective". Ce type d'optimisation apporte avec lui un certain nombre de défis qui doivent être abordés, ce qui souligne la nécessité de nouveaux algorithmes qui peuvent gérer efficacement le nombre croissant d'objectifs.
Alors, les problèmes d'optimisation mono-objectif regroupent les problèmes n'ayant qu'un objectif unique à optimiser et les problèmes d'optimisation multi-objectif possèdent plus d'un objectif à optimiser.

Problèmes d'optimisation « avec contraintes» ou «sans contraintes » :

C'est une classification basée sur l'existence de contraintes qui concerne la présence ou non de limitations sur les variables (de décision) qui peuvent être simplement des bornes géométriques ou des égalités ou inégalités non linéaires.
Les problèmes d'optimisation sans contraintes sont des problèmes pour lesquels toute solution appartenant à S (l'ensemble des solutions optimales de la (les) fonctions objectif) est considérée comme une solution faisable et acceptable.
Les problèmes d'optimisation avec contraintes sont des problèmes dont une solution $s \in S$ peut être considérée comme solution faisable que si elle satisfait l'ensemble des contraintes prédéfinies. Alors selon la présence ou non des contraintes sur les variables de décision, on détermine si notre problème est sans ou avec contrainte. Rappelons que les problèmes avec contraintes sont les plus compliqués à résoudre.

Problèmes d'optimisation « Mono-variable » ou « Multi-variable » :

Classification basée sur le nombre des variables de conception, les problèmes d'optimisation mono-variable désignent habituellement tous les problèmes dont la fonction objectif ne dépend que d'une seule variable. Les problèmes d'optimisation multi-variable rassemblent les problèmes dont la fonction objectif est définie par plus d'une variable.

Problèmes d'optimisation « continue » ou «discrète ou combinatoire » :

C'est une classification basée sur la nature des variables de conception. Dans certains problèmes d'optimisation, les variables n'ont de sens que si elles sont entières, on parle dans ce cas là d'optimisation discrète qui regroupe les problèmes dont la solution recherchée est une combinaison d'éléments parmi un ensemble fini d'objets.

A la différence de l'optimisation continue qui recherche une solution dans un ensemble infini d'objets (ex : l'ensemble des nombres réels).

Alors, si les variables de décision sont discrètes (entiers ou binaires), le problème est dit discret, le problème est dit continu si ses variables sont des réels, enfin il est dit combinatoire si ses variables sont une permutation sur un ensemble fini de nombres.

Les problèmes combinatoires sont généralement plus difficiles à résoudre car ils se heurtent à l'explosion du nombre de combinaisons à explorer. En plus la nature discrète des variables forme un espace non dérivable qui rend inutile toute technique basées sur le gradient.

Problèmes d'optimisation « linéaire » ou « non-linéaire» :

C'est une classification basée sur la nature des équations impliquées pour la fonction objectif et des contraintes.

Selon cette classification, les problèmes d'optimisation peuvent être classés comme des problèmes linéaire ou non-linéaire ou géométrique ou quadratique.

Problèmes d'optimisation « mono-model » ou « multi-model » :

C'est une classification basée sur le nombre des optimums, un problème dont F ne compte qu'un seul optimum est appelé problème uni-modale. Dans ce cas, un optimum local est aussi un optimum global. A l'inverse, Lorsqu'une fonction admet plusieurs optima locaux, elle est dite multimodale.

Problèmes d'optimisation « petite taille » ou « grande taille » :

La difficulté d'instance dépend de la complexité du problème à résoudre. Pour les problèmes NP-difficile, le nombre de solutions réalisables croît exponentiellement avec la taille de l'instance. Ce qui implique qu'il est possible de traiter efficacement des instances de petite

taille par une méthode exacte. Malheureusement, les méthodes exactes par natures énumératifs, souffrent de l'explosion combinatoire et ne peuvent s'appliquer à des problèmes de grandes tailles (dès que n dépasse des instances de taille modérée). Dans ce cas, il est nécessaire de faire appel à des méthodes approchées.

Problèmes d'optimisation « déterministe » ou « stochastique » :

Les problèmes d'optimisation déterministe considèrent que les données sont connues parfaitement, contrairement aux problèmes d'optimisation stochastique; par exemple une approche stochastique peut être pertinente dans le cas où les variables d'un problème sont les ventes futures d'un produit. Dans ce cas, l'incertitude peut être introduite dans le modèle.

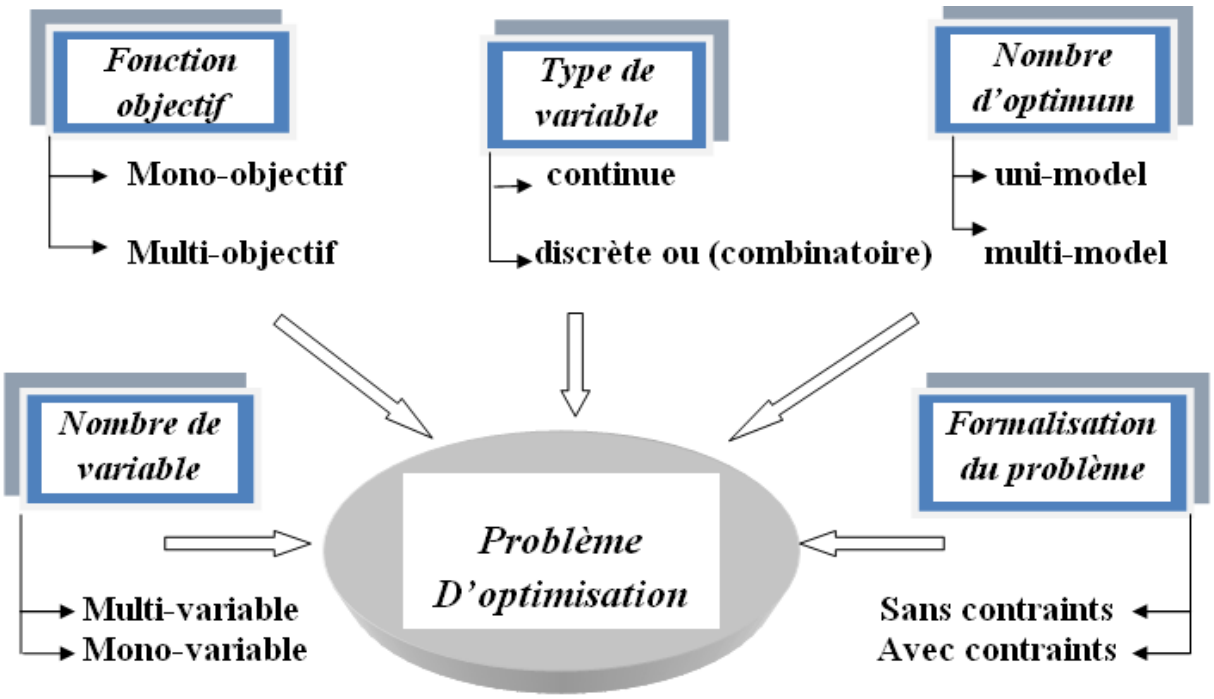


Figure 3: Classification des problèmes d'optimisation.

6. *Les méthodes d'optimisations :*

- **Définition :**

‘‘ Une méthode d'optimisation c'est de chercher un point ou un ensemble de points de l'espace d'état possible qui satisfait au mieux un ou plusieurs critères. Le résultat est appelé valeur optimale ou optimum ’’

Notez qu'il existe différents types de méthodes pour résoudre des problèmes d'optimisation, mais la question demeure de savoir comment choisir la méthode appropriée pour résoudre un problème proposé.

Il existe de nombreux algorithmes (méthodes) dans la littérature d'optimisation, et aucun algorithme unique ne convient à tous les problèmes. Le choix de la bonne méthode d'optimisation pour un problème donné est d'une importance cruciale et l'algorithme choisi pour une tâche d'optimisation dépendra en grande partie de :

Type de problème, qualité des solutions souhaitées, nature de l'algorithme, sources disponibles, …etc.

Alors, une bonne classification du problème conduit inévitablement à un bon choix de méthode.

Dans ce qui suit, nous expliquerons comment classer les méthodes d'optimisation selon le type de problème, les algorithmes peuvent être classés de plusieurs manières. Cette figure résume les principales classes de méthodes d'optimisation.

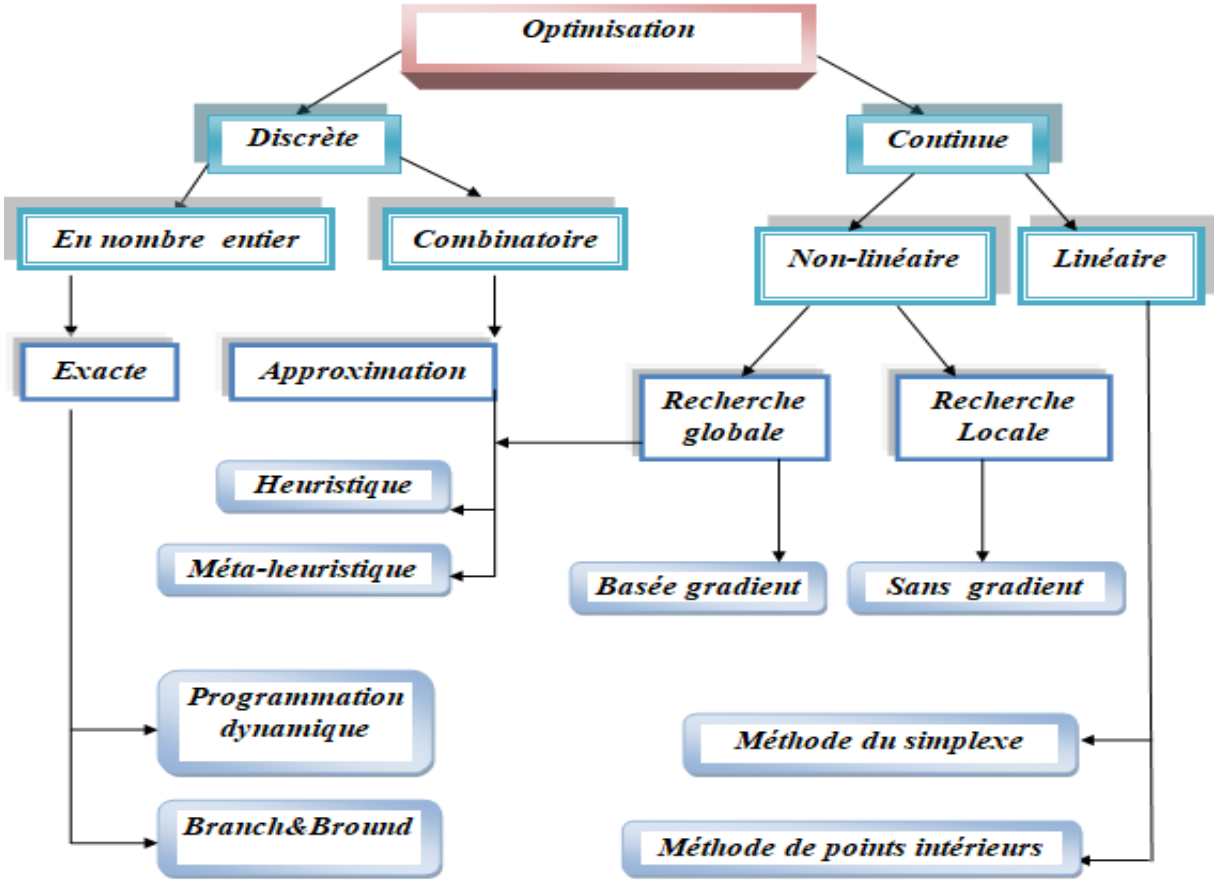


Figure 4: classification général des méthodes d'optimisations.

Les méthodes d'optimisation continue traitent les problèmes dont les variables sont autorisés à prendre n'importe quelle valeur réelle. Nous pouvons les classifier en deux types d'optimisation continue ***(Allaoua, 2017) , (Huguet, 2020-2021)*** **:**

Linéaire (programmation linéaire) :

Ce sont des techniques d'optimisation pour des problèmes caractérisés par une fonction objectif linéaire qui soumise à des contraintes d'égalité et /ou d'inégalités linéaire, exemples de ces méthodes :

6.1.1. Méthode de simplexe :

Un algorithme créé par George Dantzig en 1947, avec comme idée d'énumérer tous les sommets du polytope, d'évaluer la fonction objectif, d'ensuite prendre le minimum (nombre important de sommets) et à partir d'un sommet quelconque, chercher un sommet voisin qui améliore l'objectif (règle de pivotage).

Note : Ici, nous passons d'un problème continu à un problème discret !

6.1.2. Les méthodes de points intérieurs :

Forme une classe d'algorithmes qui ont l'avantage d'être polynomiaux lorsqu'ils sont appliqués à des problèmes d'optimisation linéaires, quadratiques convexes, positifs semi-définis ; et plus généralement aux problèmes d'optimisation convexe. L'idée de base de la méthode est d'utiliser des fonctions barrières pour décrire l'ensemble des solutions qui est convexe par la définition du problème. Contrairement à l'algorithme du simplexe, cette méthode permet d'atteindre l'optimum du problème en passant par l'intérieur de l'ensemble de solutions réalisables.

***Non-linéaire* :**

Il s'agit de méthodes d'optimisation traitant de problèmes caractérisés par :
Une fonction objectif non-linéaire, ou une fonction objectif linéaire soumise à des contraintes d'égalité et/ou d'inégalité non-linéaires, ces méthodes comportent de la :

6.2.1. Recherche globale :

Consiste à trouver la valeur optimale d'une fonction donnée par toutes les solutions possibles.

Quant aux méthodes de recherche globales, elles entrent dans le cadre de l'optimisation approximative, ce que nous expliquerons dans la deuxième partie de ce chapitre.

6.2.2. Recherche Locale :

À l'opposé de la recherche globale, la recherche locale consiste à trouver la valeur optimale dans l'ensemble voisin d'une solution candidate.

Les méthodes de recherche locale « **basée sur le gradient** » : sont des méthodes d'optimisation locales itératives qui utilisent largement l'information du gradient de la fonction objectif. $\boldsymbol{x^{(k+1)} = x^{(k)} + \propto g(\nabla f, x^{(k)})}$ Comme la méthode de Newton.

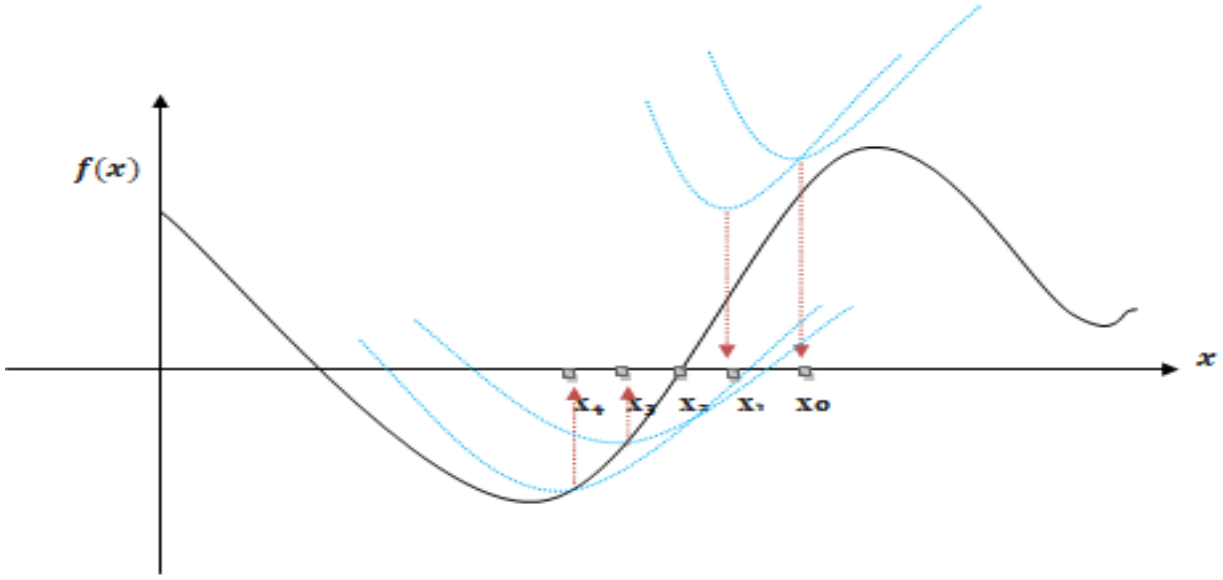


Figure 5: une convergence vers une solution optimale utilisant le gradient.

Les méthodes d'optimisation **discrète** qui doivent appartenir un ensemble discret. Dans la figure **– 4 -** nous constatons qu'il y a deux types d'optimisation discrète ***(MOHAMMED, octobre2015) (Huguet, 2020-2021)*** :

***Programmation en nombre entier* :** où l'ensemble discret est un sous-ensemble d'entier,
Les méthodes d'optimisation discrètes peuvent être exactes ou ses algorithmes sont capables de donner une solution optimale à un problème d'optimisation, par exemple :

6.1.1. *Branch & Bround* :

Pour plusieurs problèmes, notamment les problèmes d'optimisation combinatoire, l'ensemble de leurs solutions est fini (il est dénombrable dans tout les cas). Il est donc possible, en principe, de lister toutes ces solutions et de prendre celle qui nous convient. L'inconvénient majeur de cette approche est le nombre prohibitif de solutions : il n'est pas facile d'effectuer cette énumération. La méthode Branch and Bround (procédure par évaluation et séparation progressive) consiste à énumérer intelligemment ces solutions dans le sens où, en utilisant certaines propriétés du problème en question, cette technique parvient à éliminer les solutions partielles qui ne conduisent pas à la solution que l'on est en train de chercher.

De ce fait, nous parvenons souvent à obtenir la solution souhaitée dans un délai raisonnable. Bien sûr, dans le pire des cas, on se rabat toujours sur l'élimination explicite de toutes les solutions au problème. Pour ce faire, cette méthode est dotée d'une fonction permettant de limiter certaines solutions afin de soit les exclure, soit les maintenir comme solutions potentielles. Bien entendu, les performances du Branch and Bround dépendent, entre autres, de la qualité de cette fonction (sa capacité à exclure au plus vite des solutions partielles) ***(Allaoua, 2017)***.

6.3.3. Programmation dynamique :

La programmation dynamique est une technique algorithmique qui résout une classe spéciale de problèmes d'optimisation sous contraintes. Elle a été désignée pour cela premier mandat dans les années 1950 par le professeur Richard Bellman.
Elle s'applique aux problèmes d'optimisation dont la fonction objectif est décrite comme "la somme de fonctions monotones non décroissantes des ressources ».
Le principe de la programmation dynamique est de diviser le problème à résoudre en plusieurs sous-problèmes identiques, et de ne résoudre qu'un seul sous-problème à la fois en sauvegardant les résultats (à l'aide d'un tableau de résultats déjà calculés, rempli au fur et à mesure que nous résolvons les sous-problèmes)

Remarque : il s'agit d'une méthode ascendante : nous commençons généralement par les plus petits sous-problèmes et progressons vers des sous-problèmes de plus en plus difficiles.

Optimisation combinatoire :

Là où l'ensemble discret est un ensemble d'objets ou des structures combinatoires finis tels que : les affectations, les combinaisons, les horaires, les séquences,…etc.
Cependant, les méthodes d'optimisation utilisées dans ce type sont des méthodes approximatives i.e. Des algorithmes qui trouvent des solutions approximatives au problème d'optimisation, dans ce type de méthode on distingue plusieurs sous-classes ***(Allaoua, 2017)*** :

6.1.2. Les heuristiques :

Le mot heuristique a pour origine le grec ancien eurêka qui signifie "trouver" et qualifie tout ce qui utile à la découverte, à l'invention et à la recherche. Les heuristiques sont des méthodes d'optimisation relativement simples et rapides (avec une complexité de temps polynomial) pour résoudre des problèmes difficiles. Ces méthodes sont généralement spécifiques à un type de problème et sont donc dépendantes de celui-ci.

En (1963) Feignebaum et Feldman « Une méthode heuristique (ou simplement une heuristique) est une méthode qui aide à découvrir la solution d'un problème en faisant des conjectures plausibles mais faillible de ce qui est la meilleure chose à faire »

Les heuristiques sont des critères, des principes ou des méthodes permettant de déterminer parmi plusieurs lignes de conduite, celle qui promet d'être la plus efficace pour atteindre un but. Elle représente des compromis entre deux exigences : le besoin de rendre de tels critères simples et en même temps, le désir de les voir établir une distinction entre les bons et les mauvais choix ***(ghoumari, 2018)***.

6.1.3. Heuristiques d'insertion (amélioration) :

Les heuristiques d'insertion sont des heuristiques de construction très efficaces, l'heuristique d'insertion la plus connue dans la littérature est le plus proche voisin, son objectif est de corriger les tournées initialement créées comme le problème de routage de véhicules (VRP), voyageur de commerce (TSP) et d'autres types de problème de transport selon des besoins particuliers, son principe est de sélectionner à chaque itération le nœud le plus proche de l'itinéraire en construction, puis de l'insérer à l'endroit du cycle ce qui minimise le coût du tour ***(Samir, 2021)***.

6.1.4. Heuristique de construction :

L'idée de base d'une heuristique constructive est de réduire la taille du problème à chaque étape pour limiter progressivement l'ensemble des solutions réalisables. L'heuristique de construction construire progressivement le circuit en ajoutant une ville (nœud) à chaque étape. Ils s'arrêtent dès que la solution est trouvée et ne cherchent pas à l'améliorer. Dans cette catégorie, il y a l'heuristique du plus proche voisin, l'algorithme glouton. Les heuristiques

d'amélioration consistent à une fois qu'une route est générée par une heuristique de construction pour l'améliorer afin d'obtenir une tournée de qualité meilleure.
Les heuristiques de construction d'une solution initiale sont généralement simples et rapides, mais les résultats générés sont plus ou moins bons par rapport à la solution optimale ***(Samir, 2021)***.

Les métaheuristiques :

Le terme métaheuristique est composé de méta provenant du grec ancien qui signifie "au-delà" qui se traduit par " à un plus haut niveau" et heuristique mentionné ci-dessus.
Ce terme a été mentionné pour la première fois par Fred Glover lors de la conception d'une recherche taboue : "La recherche avec tabou peut être vue comme une 'métaheuristique', superposée à une autre heuristique. L'approche vise à éviter les optima locaux par une stratégie d'interdiction (ou, plus généralement, pénalisant) certains mouvements.
Nous voyons que les heuristiques sont des méthodes conçues pour résoudre spécifiquement un type de problème, tandis que les métaheuristiques visent à résoudre non pas un problème spécifique mais un plus large éventail de problèmes. Désormais, les métaheuristiques prospèrent davantage, ce qui peut être corrélé à l'augmentation de la puissance de calcul des ordinateurs ainsi qu'à leurs meilleurs résultats sur des problèmes d'optimisation toujours plus complexes.
Parmi les métaheuristiques les plus populaires, on retrouve les algorithmes basés sur des populations de solutions (comme les algorithmes génétiques ou les algorithmes évolutionnaires), par opposition aux algorithmes à solution unique (comme le recuit simulé, la descente de gradient, la recherche de tabous, etc.) ***(ghoumari, 2018)***.

Ces derniers seront l'un des sujets principaux de notre étude dans cette thèse, et nous en discuterons dans le reste de ce chapitre avec une explication détaillée. Mais d'abord, nous expliquerons pourquoi les métaheuristiques sont plus efficaces que les méthodes classiques.

7. *Capacité des métaheuristiques à s'extraire du minimum local :*

Pour surmonter l'obstacle des minima locaux, une idée s'est révélée très fructueuse, au point qu'elle est à la base de toutes les métaheuristiques : il s'agit d'autoriser, de temps en temps, des

mouvements vers le haut, c'est-à-dire d'accepter une dégradation momentanée de la situation, en changeant la configuration actuelle. C'est le cas si l'on passe de C1 à C2 (voir la figure). Un mécanisme de contrôle de la dégradation permet d'éviter la divergence du procédé. il devient alors possible de sortir du piège représenté par un minimum local, alors le but des métaheuristiques est d'explorer efficacement l'espace de recherche afin de déterminer des solutions (presque) optimales avec des mécanismes qui vous permettent d'éviter d'être bloqué dans les régions de l'espace de recherche ou elles peuvent faire appel à des heuristiques prenant en compte la spécificité du problème traité, mais ces heuristiques sont contrôlées par une stratégie de niveau supérieur, pour mieux orienter la suite du processus de recherche ***(siary, 2014)***.

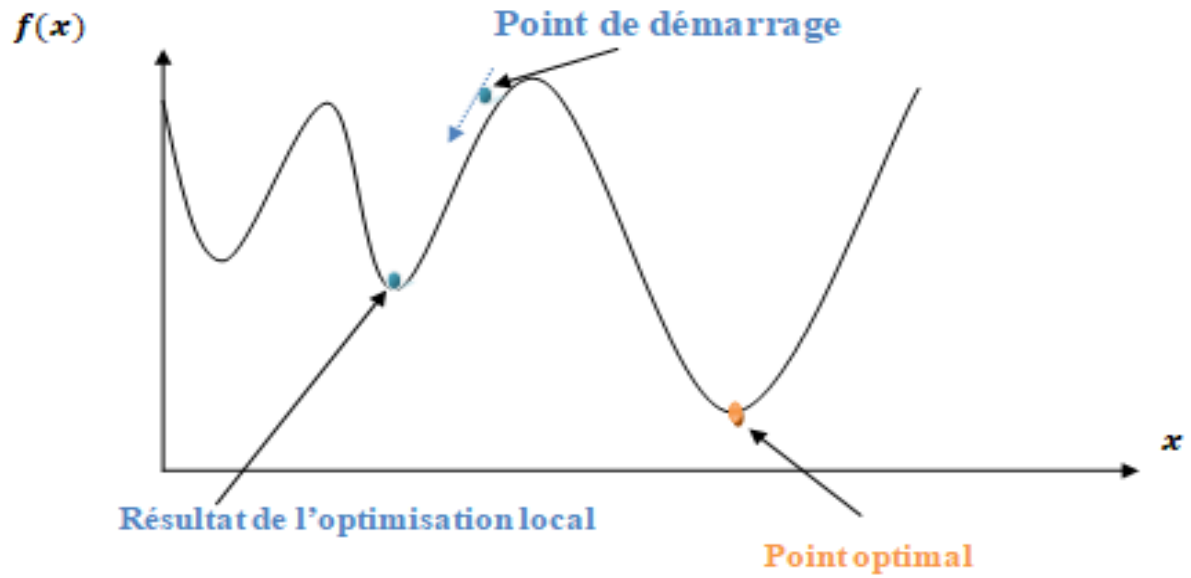


Figure 6 : sortir des minima locaux d'une fonction objectif avec les métaheuristiques.

8. *Principes des métaheuristiques :*

Étymologiquement, les métaheuristiques sont des abstractions des heuristiques. Il s'agit d'algorithmes basés sur des heuristiques et qui s'appuient sur 3 principes fondamentaux.

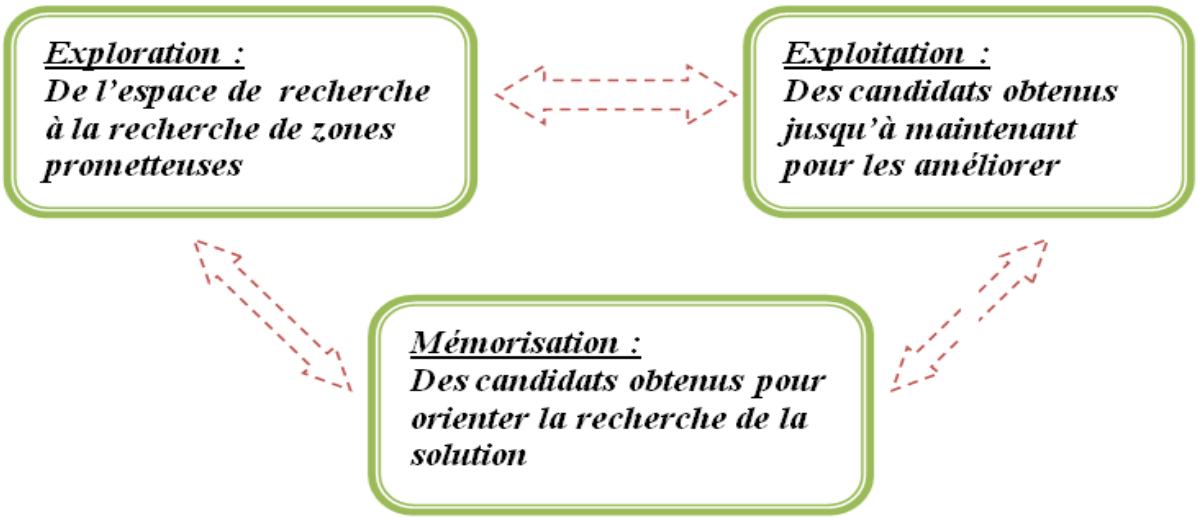


Figure 7: les trois principes des métaheuristiques.

Contrairement aux algorithmes de recherche locale, les métaheuristiques ont des mécanismes d'exploration qui cherchent, dans l'espace des solutions, des domaines prometteurs où la solution pourrait être trouvée. Cependant, l'algorithme doit non seulement explorer l'espace mais aussi exploiter les candidats obtenus pour tenter de les améliorer.

L'exploitation et l'exploration ne sont possibles qu'avec un mécanisme de mémorisation qui conserve certaines informations passées.

Il y a différents critères pour classifier et décrire ces algorithmes métaheuristiques, mais généralement, les métaheuristiques sont classées en deux groupes : les métaheuristiques de Trajectoire (à solution unique) et les métaheuristiques à base de population. Pour chaque groupe nous donnons un aperçu des métaheuristiques les plus répandues :

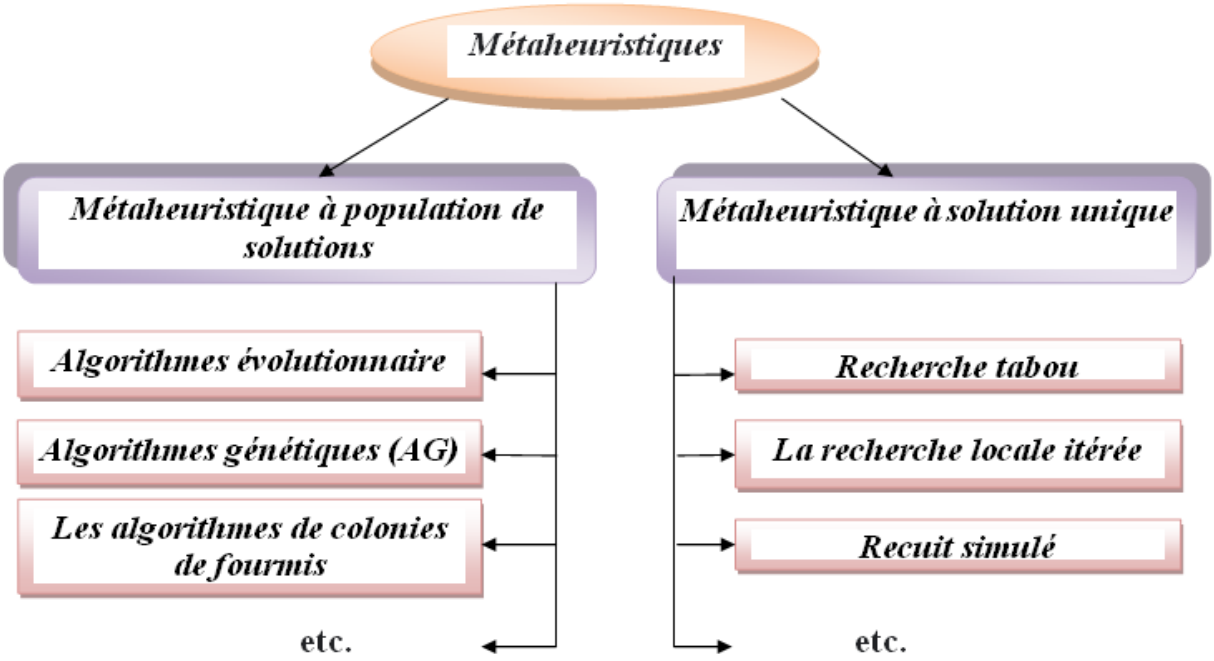


Figure 8 : classification des méthodes métaheuristiques.

Métaheuristiques trajectoire (à solution unique) :

Dans cette section, nous présentons des métaheuristiques basées sur une solution unique, également appelées méthodes de trajectoire. Contrairement aux métaheuristiques basées sur la population, ces algorithmes commencent la recherche par une seule solution initiale et s'en éloignent progressivement, et circule en construisant une trajectoire dans l'espace de recherche. De nombreuses méthodes reposent sur ce principe : on peut citer principalement la méthode du recuit simulé, la méthode GRASP, la méthode de descente, la recherche tabou, et la recherche locale itérée...etc ***(ghoumari, 2018)*** , ***(Zoubir, 2020/2021)***.

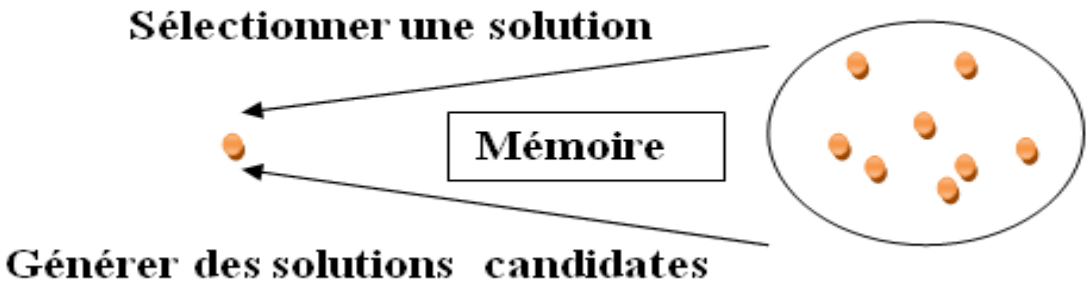


Figure 9: Illustration des principes généraux d'une métaheuristique à base de solution unique.

8.1.1. Recherche tabou :

La méthode de recherche Tabou est une technique de recherche proposée par Fred Glover en 1986. Elle est devenue un classique de l'optimisation combinatoire, et classée comme une méthode de recherche locale itérative avec mémoire adaptative.

En réalité, la méthode de recherche Tabou est similaire à la méthode de recherche simulée. Elle fonctionne avec une seule configuration courante, qui est actualisée au cours des itérations.

La recherche avec tabou, ou simplement "recherche tabou", est basée sur l'utilisation de structures de mémoire. Le principe de base est de choisir la meilleure solution du voisinage de la solution courante, parfois moins bonne que la solution courante.

Par conséquence, la recherche ne s'arrête pas au premier optimum local trouvé. Le danger est alors de revenir à des solutions déjà explorées. Pour éviter les cycles, nous stockons les dernières solutions visitées dans une liste tabou et nous interdisons tout mouvement qui conduit à une solution de cette liste. La liste tabou est donc une sorte de mémoire à court terme. Tout mouvement qui nous mène de la solution actuelle à une solution de liste tabou est appelé un mouvement tabou.

L'organigramme de l'algorithme de recherche Tabou est présenté à la figure-10- Alors, on peut voir que la méthode Tabou est basée sur deux principes. Le premier consiste à améliorer la valeur de la fonction objectif à chaque itération. En prenant à chaque fois la meilleure solution voisine. Si celle-ci n'existe pas, le choix se porte sur le moins mauvais voisin. Le second principe consiste à garder en mémoire les dernières solutions choisies et à ne pas les prendre en considération. L'inconvénient de cette méthode est que si la solution atteinte est très proche d'un optimum local, il y a une possibilité de ne pas pouvoir sortir de cet optimum local atteint et donc ne pas être certain que ce dernier est l'optimum global que nous recherchons ***(Cheriet, 2016)***, ***(siary, 2014)***.

La méthode de Tabou a été utilisée dans l'ordonnancement des lignes flow shop ou job shop**.**

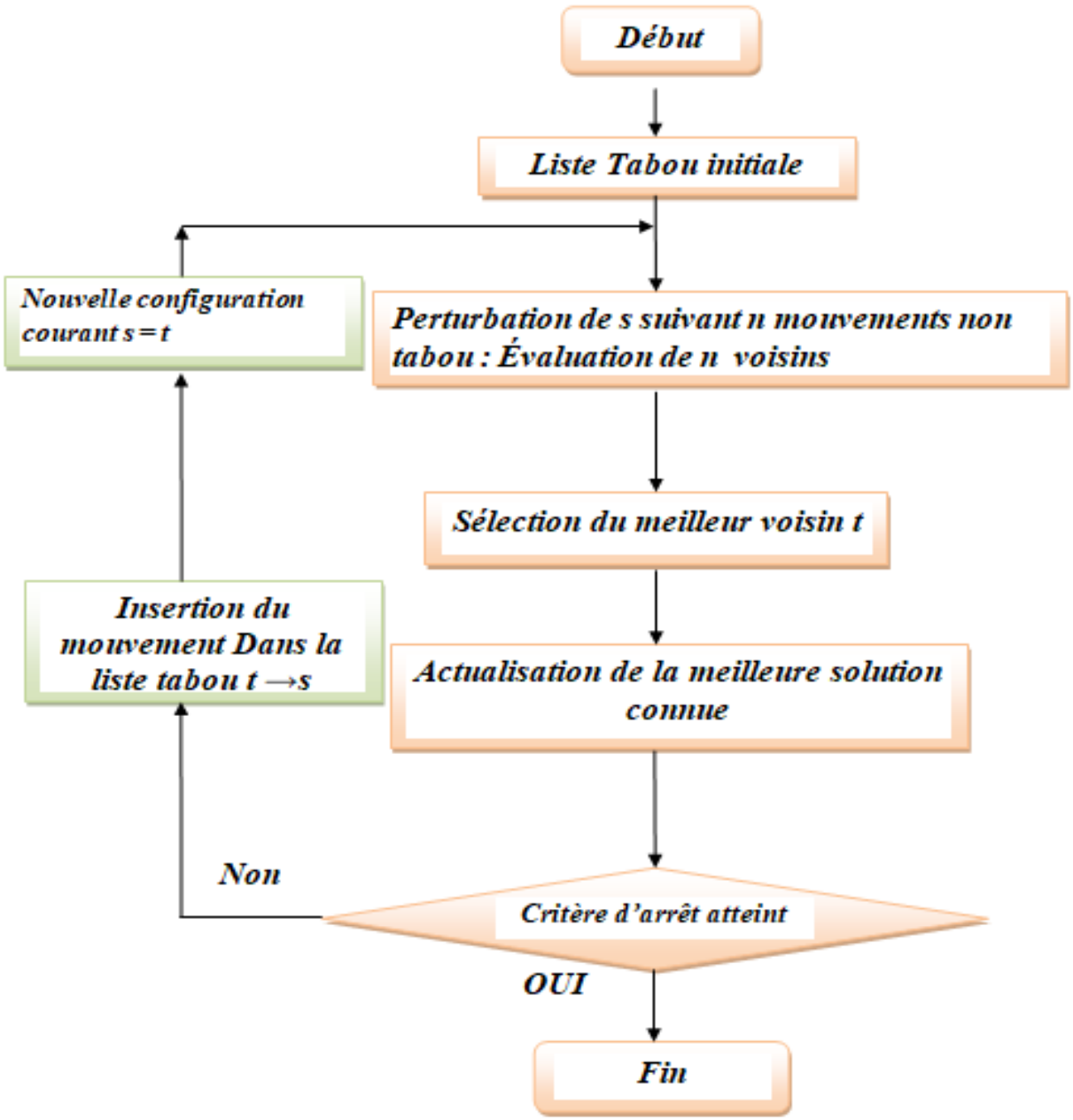


Figure 10:Schéma logique de l'algorithme de recherche Tabou.

8.1.2. Recuit simulé :

La méthode du recuit simulé trouve ses origines dans la mécanique statistique S. Kirkpatrick et ses collègues étaient des physiciens ; ils s'intéressaient à la configuration à basse énergie des matériaux magnétiques désordonnés, regroupés sous le terme de verres de spin. L'idée était de traiter ce problème en utilisant la technique expérimentale de recuit utilisée par les métallurgistes pour obtenir un état solide d'énergie minimale. Cette technique consiste à chauffer le matériau à une température élevée, puis à abaisser lentement sa température La méthode du recuit simulé transpose le processus de recuit à la résolution d'un problème d'optimisation ***(siary, 2014)***.

Problème d'optimisation	Système physique
Solution	**État du système**
Fonction objectif	**Énergie libre (E)**
Paramètres du problème	**Coordonnées des particules**
Trouver une bonne configuration	**Trouver les états à basse énergie**
Optimum global	**État stable ordonné**
Optimum local	**État métastable**
Recherche locale	**Trempe rapide**
Le paramètre T	**La température**

Tableau 1 : Analogie entre un problème d'optimisation et un système physique.

En pratique, la technique exploite l'algorithme de Metropolis, qui permet de décrire le comportement d'un système en « équilibre thermodynamique » à une certaine température T : partant d'une configuration donnée (par exemple, un placement initial de tous les composants), on fait subir au système une modification élémentaire (par exemple, on translate un composant ou on échange deux composants) ; si cette transformation a pour effet de diminuer la fonction objectif (ou énergie) du système, elle est acceptée ; si elle provoque au contraire une augmentation ΔE de la fonction objectif, elle peut être acceptée tout de même,

Avec la probabilité $e^{\frac{-\Delta E}{T}}$. On itère ensuite ce procédé, en gardant la température constante, jusqu'à ce que l'équilibre thermodynamique soit atteint, concrètement au bout d'un nombre « suffisant» de modifications. On abaisse alors la température, avant d'effectuer une nouvelle série de transformations : la loi de décroissance par paliers de la température est souvent empirique, tout comme le critère d'arrêt du programme.

L'organigramme de l'algorithme de recuit simulé est présenté à la figure 11. Lorsqu'il est appliqué au problème du placement des composants, le recuit simulé effectue une transformation du désordre en ordre, qui est représentée de manière imagée sur la figure 11.

On peut également visualiser certaines étapes de cette mise en ordre en appliquant la méthode au placement de composants sur les nœuds d'une grille.

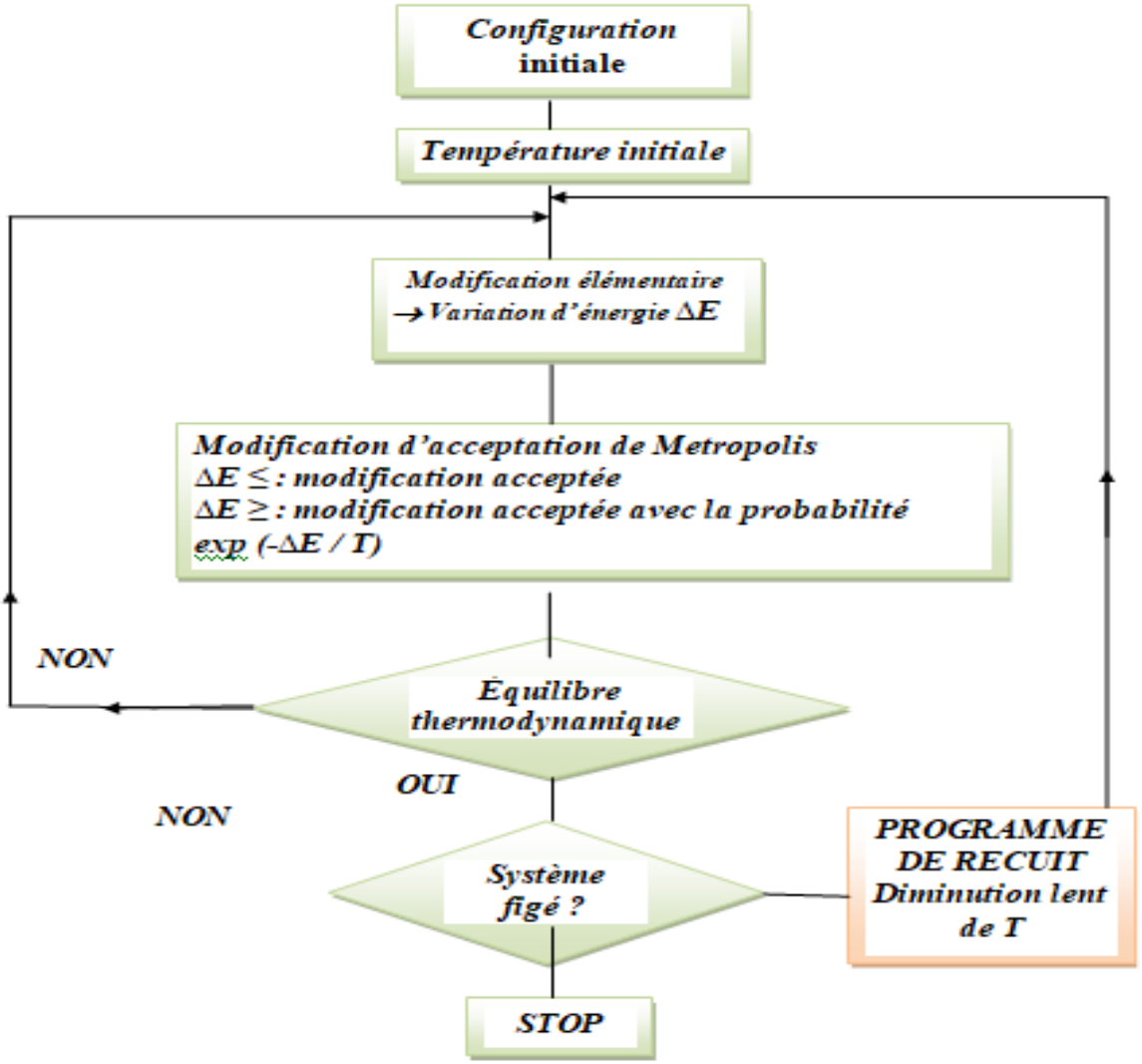


Figure 11: Schéma logique de l'algorithme de recuit simulé.

Métaheuristiques à population de solutions :

Contrairement aux méthodes précédentes, ces approches consistent à travailler avec un ensemble de solutions simultanément. Les métaheuristiques à base de population débutent la recherche avec une panoplie de solutions. Elles s'appliquent sur un ensemble de solutions afin d'en extraire la meilleure (l'optimum global) qui représentera la solution du problème traité. L'idée d'utiliser un ensemble de solutions au lieu d'une seule solution renforce la diversité de la recherche, améliorer l'exploration de l'espace de recherche et augmente la possibilité d'émergence de solutions efficace ***(siary, 2014) (Cheriet, 2016)***.

On distingue dans cette catégorie : les algorithmes évolutionnistes inspirés de la théorie de l'évolution de Charles Darwin et forment une classe importante des métaheuristiques à base de population, l'algorithme génétique, l'algorithme par colonies d'abeilles...etc.

Figure 12 : Illustration des principes généraux d'une métaheuristique à base de population.

8.1.3. Les algorithmes évolutionnaires & les algorithmes génétiques (AG) :

Les algorithmes évolutionnaires sont parmi les métaheuristiques à base de population, ils sont inspirés de la biologie toute en introduisant le théorème de Darwin en 1859 dans l'évolution des espèces.

Les algorithmes évolutionnaires représentent une famille d'algorithmes issus de la théorie de l'évolution par sélection naturelle, le principe fondamental étant que les individus les mieux adaptés à leur environnement survivent et peuvent se reproduire, laissant une descendance qui transmettra leurs gènes. La clef étant l'adaptation des individus face à la pression de l'environnement, l'analogie avec l'optimisation devient claire. Cette adaptation peut alors être assimilée à une optimisation des individus afin qu'ils soient de mieux en mieux adaptés à leur environnement, au fur et à mesure des générations (correspondant aux itérations de l'algorithme). Nous pourrons alors définir les algorithmes évolutionnaires comme des méthodes faisant évoluer un ensemble de solutions appelé "population". Les solutions, appelées "individus", sont représentées par leur génotype, qui s'exprime sous la forme d'un phénotype. Afin d'évaluer la performance d'un individu, on associe au phénotype la valeur de

la fonction objectif (ou fonction d'évaluation, fitness). Un algorithme évolutionnaire se décompose en plusieurs étapes, chacune d'elles étant associée à un opérateur décrivant la façon de manipuler les individus. Donc, en termes d'optimisation, l'évolution se traduit par un processus itératif de recherche de l'optimum dans l'espace de recherche.

Parmi les algorithmes évolutionnaires les plus connus et les plus répandus, on trouve les algorithmes génétiques qui se détachent en grande partie par la représentation des données du génotype, initialement sous forme d'un vecteur binaire et plus généralement sous forme d'une chaîne de caractères.

Pour les algorithmes génétiques (AGs) ; ce sont des algorithmes fondés sur les mécanismes de la sélection naturelle et de la génétique, utilisant les principes de la survie des structures les mieux adaptées. Ces algorithmes fabriquent des chromosomes qui codent chacun une solution potentielle à un problème donné à chaque étape (appelée génération). Ces chromosomes se combinent, mutent et sont sélectionnés en fonction de leurs qualités à répondre au problème afin de les explorer dans la génération suivante avec l'espoir d'améliorer la performance qui en résulterait.

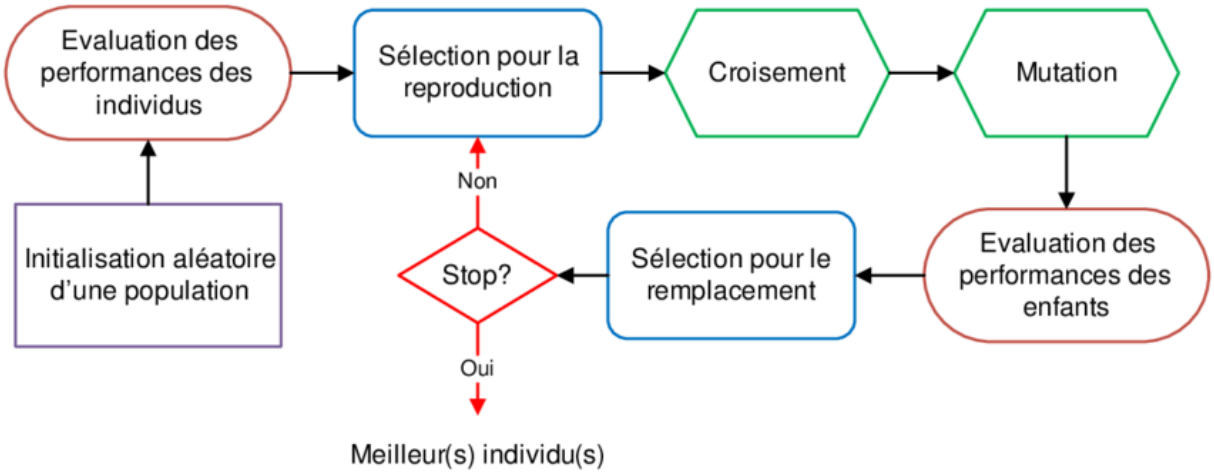


Figure 13 : Principe d'un algorithme évolutionnaires.

8.1.4. Les algorithmes de colonies de fourmis :

Cette approche, due à Colorni, Dorigo et Maniezzo, s'efforce de simuler la capacité collective de résolution de certains problèmes, observée chez une colonie de fourmis, dont les

membres sont pourtant individuellement dotés de facultés très limitées. Apparues sur terre il y a quelque 100 millions d'années, les fourmis sont en effet l'une des espèces les plus prospères : 10 millions de milliards d'individus, répartis partout sur la planète : leur poids total est du même ordre de grandeur que celui des humains ! Le succès de cette espèce soulève de nombreuses questions. En particulier, les entomologistes ont analysé la collaboration qui s'établit entre les fourmis pour aller chercher de la nourriture à l'extérieur de la fourmilière ; il est remarquable que les fourmis suivent toujours le même chemin et que ce chemin soit le plus court possible. Cette conduite est le résultat d'un mode de communication indirecte, via l'environnement : la « stigmergie». Chaque fourmi dépose, le long de son chemin, une substance chimique, dénommée « phéromone» ; tous les membres de la colonie perçoivent cette substance et orientent préférentiellement leur marche vers les régions les plus « odorantes».

Il en résulte notamment la faculté collective de retrouver rapidement le plus court chemin, si celui-ci se trouve obstrué fortuitement par un obstacle (figure -14-). En s'inspirant de la modélisation de ce comportement, Dorigo et al. Ont proposé un nouvel algorithme pour la résolution du problème du voyageur de commerce. Depuis ces travaux, la démarche a été étendue à beaucoup d'autres problèmes d'optimisation, combinatoires ou même continus.

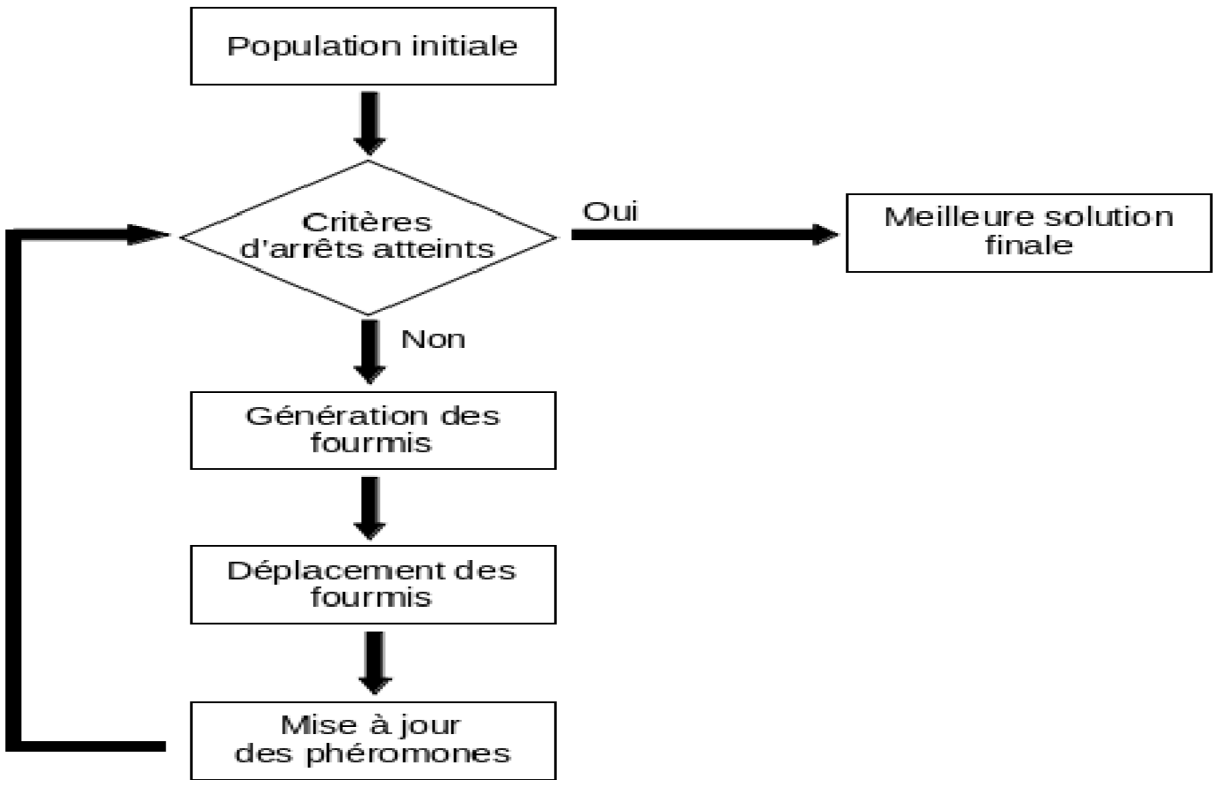


Figure 14 : Structure de l'algorithme de colonies de fourmis.

***Conclusion* :**

Dans ce chapitre, nous avons essayé de rassembler un état de l'art sur les problèmes d'optimisation en général, tout en montrant la manière de définir un problème pareil, ses contraintes, ses objectifs ainsi que les méthodes utilisées pour le résoudre tout en respectant le domaine de définition de ses variables à fin de trouver l'optimum de la fonction objectif.
Nous avons parlé des méthodes utilisées dans le domaine de l'optimisation mono ou multi-objectif soit les problèmes mathématiques ou les métaheuristiques.
Parmi les métaheuristiques existant, nous trouvons les algorithmes évolutionnaires qui exercent le principe de la génétique pour résoudre ces problèmes et ils ont démontrés leur efficacité.
Pour cette raison et puisque notre travail se base sur l'un des algorithmes génétiques, nous parlerons dans le chapitre suivant en détails de cet algorithme.

Chapitre II : les algorithmes génétiques

Introduction :

Dans le chapitre précédent, nous avons présenté un état de l'art sur les notions et les concepts liés à notre domaine d'étude. Historiquement, les premières méthodes d'optimisation sont issues des travaux de Fermat, Lagrange, Hamilton, Newton et Gauss. Les premières méthodes de la programmation linéaire (dédiées à la planification de programmes militaires) ont été introduites par Kantorovitch en 1960, puis améliorées par Dantzig en 1963. Au fil du temps, une énorme croissance de la taille et de la complexité des problèmes d'optimisation s'est produite ; ce qui a motivé le développement de nouvelles méthodes performantes, rapides et de qualité (les métaheuristiques).

Dans le cas de problèmes NP-difficiles deux possibilités sont offertes. Si le problème est de petite taille, alors un algorithme exact permettant de trouver la solution optimale peut être utilisé (Branch & Bround, programmation dynamique...). Malheureusement, ces algorithmes souffrent de l'explosion combinatoire et ne peuvent s'appliquer à des problèmes de grandes tailles. Dans ce cas, il est nécessaire de faire appel à des heuristiques ou métaheuristiques permettant de trouver de bonnes solutions approchées.

Dans ce chapitre, nous allons décrire en détail la méthode évolutive la plus connue dans ce domaine qui était inspirée de la théorie de l'évolution et des processus biologiques et permettent à des organismes de s'adapter à leur environnement. Il s'agit de l'algorithme génétique qui a été proposé dans le milieu des années 60.

1. Les Algorithmes génétiques :

Les théories de l'évolution présentent une bonne inspiration en utilisant comme exemple les êtres vivants évoluant sous l'effet du milieu : les mieux adaptés ont plus de chances de survivre et donc de se reproduire. La génétique, dont l'objet est d'étudier les mécanismes de l'hérédité, propose quant à elle un modèle permettant d'expliquer la transmission de ces caractéristiques d'une génération à l'autre.

La sélection naturelle repose alors sur trois principes : le principe de variation, le principe d'adaptation, et le principe d'hérédité.

En 1975, John Holland a développé le premier algorithme génétique pour résoudre des problèmes énormes et complexes.

Par conséquent, de nombreuses métaheuristiques ont été développées, la plupart ont été inspirées par la nature ou des processus artificiels, tels que le recuit simulé, optimisation par des colonies de fourmis ***(Cheriet, 2016)***...etc.

- **Définitions :**

Théoriquement, la méthode des algorithmes génétiques s'inspire de l'évolution des espèces dans leur cadre naturel, et consiste à faire évoluer des populations dont les individus, solutions de notre problème, tendent à s'améliorer en vue de nos objectifs au fur et à mesure que les générations se succèdent. Ces algorithmes présentent un héritage génétique qui est transmis de génération en génération et conduit les individus les plus adaptés à survivre. Le processus est répété jusqu'à un certain critère d'arrêt.

Les termes utilisés pour expliquer son fonctionnement sont empreintes à la biologie,

A savoir :

- une population est un ensemble d'individus (ici, un ensemble de combinaisons d'épaisseur qui répondent au problème)

La qualité du codage des données conditionne le succès des algorithmes génétiques.

- un individu est une solution au problème (ici, une combinaison d'épaisseur qui vérifie le problème)

Et nous devons disposer des cinq éléments suivants :

- un principe de codage de l'élément de population. Cette étape associe à chacun des points de l'espace d'état une structure de données. Elle se place généralement après une phase de modélisation mathématique du problème traité.
- un mécanisme de génération de la population initiale. Ce mécanisme doit être capable de produire une population d'individus non homogène qui servira de base pour les générations futures ***(Jean BÉRARD, 26 juin 2000)***.

Remarque : le choix de la population initiale est important car il peut rendre plus ou moins rapide la convergence vers l'optimum global.

- Une fonction à optimiser. Celle-ci retourne une valeur de $\boldsymbol{R}^{+}$ appelée $\boldsymbol{fitness}$ ou fonction d'évaluation de l'individu.

- des opérateurs permettant de diversifier la population au cours des générations et d'explorer l'espace d'état. L'operateur de croisement recompose les gènes d'individus existent dans la population, l'opérateur de mutation a pour but de garantir l'exploration de l'espace d'états.
- des paramètres de dimensionnement : taille de la population, nombre totale de générations ou critère d'arrêt, probabilités d'application des opérateurs de croisement et de mutation.

2. Les principes de l'algorithme génétique :

Avant de présenter l'application de l'approche génétique sur notre problème, nous présentons dans cette section une description générale du fonctionnement des algorithmes génétiques.
Les algorithmes génétiques utilisent la théorie de Darwin sur l'évolution des espèces.
Elle repose sur trois principes : le principe de variation, le principe d'adaptation et le principe d'hérédité.

- **Le principe de variation :**

Chaque individu au sein d'une population est unique. Ces différences, plus ou moins importantes, vont être décisives dans le processus de sélection.

- **Le principe d'adaptation :**

Les individus les plus adaptés à leur environnement atteignent plus facilement l'âge adulte. Ceux ayant une meilleure capacité de survie pourront donc se reproduire davantage.

- **Le principe d'hérédité :**

Les caractéristiques des individus doivent être héréditaires pour pouvoir être transmises à leur descendance. Ce mécanisme permettra de faire évoluer l'espèce pour partager les caractéristiques avantageuses à sa survie ***(Jean BÉRARD, 26 juin 2000)***.

3. Objectifs :

Les algorithmes génétiques appartiennent à la famille des algorithmes évolutionnaires. Leur but est d'obtenir une solution approchée à un problème d'optimisation, lorsqu'il n'existe pas de méthode exacte (ou que la solution est inconnue) pour le résoudre en un temps raisonnable.
Ceci consiste à déterminer les extrêmes d'une fonction : $\boldsymbol{f : X \rightarrow R}$, où X est un ensemble quelconque appelé espace de recherche et f est appelée fonction d'adaptation ou fonction d'évaluation ou encore fonction fitness.

La fonction agit comme une «boite noire » pour l'AG. Aussi, des problèmes très complexes peuvent être approchés par programmation génétique sans avoir de compréhension particulière du problème.

- Les algorithmes génétiques représentent une méthode approchée polynomiale et vinrent pallier le handicap exponentiel des méthodes exactes.
- Par leur caractère générique et formel, ils peuvent être non seulement une méthode approchée pour un certain type de problème mais une métaheuristique qui pourrait être projetée à tout type de problème
- Inspirés d'un phénomène naturel, ils vinrent concurrencer les métaheuristiques existantes en termes de qualité de solution et temps de calcul.

4. Terminologie :

Avant d'aborder comment fonctionnent les AGs, nous devons également définir la terminologie employée, nous allons présenter quelques mots de vocabulaire relatifs à la génétique. Ces mots sont souvent utilisés pour décrire un algorithme génétique ***(Bontemps, 2000)*** :

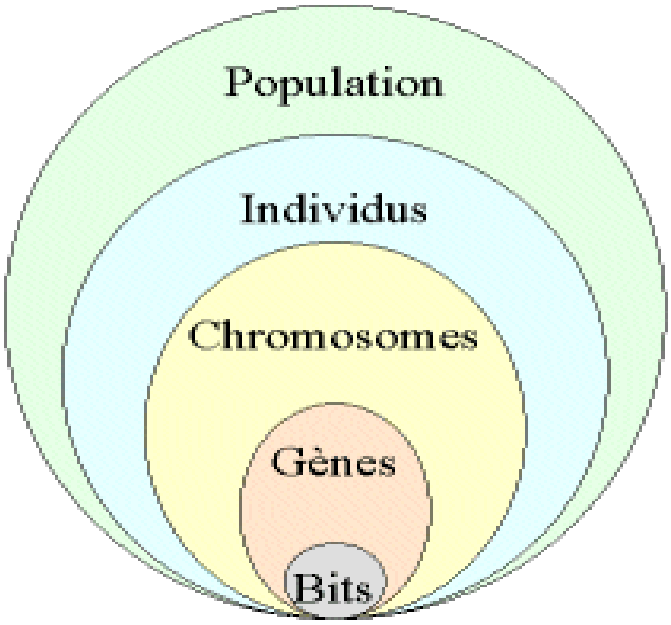


Figure 15: représentation schématique de vocabulaire du l'algorithme génétique.

4.1.Chromosome :

En biologie, il est défini comme le porteur de l'information génétique nécessaire à la construction et au fonctionnement d'un organisme. C'est une chaîne représentant les caractéristiques de l'individu.

Dans le cadre des AG, un chromosome (ou individu) est une représentation d'une solution possible de l'espace des solutions, constitué d'un ensemble de gènes (Holland décrit un chromosome comme une chaîne de bits).

4.2.Gène (Bits) :

En biologie, il représente une partie du chromosome, chaque chromosome est constitué d'un certain nombre de gènes.

Pour un AG, c'est une caractéristique, une particularité où chaque chromosome est divisé en un ensemble d'unités le constituant appelé gènes.

Exemple : Dans le codage binaire, un gène vaut soit 0 soit 1.

4.3.Allèle:

Dans les systèmes naturels, l'allèle est une composante du gène. Les allèles sont les différentes valeurs que peuvent prendre les gènes. Dans les AG, l'allèle est également appelé valeur caractéristique.

***4.4.Locus* :** Le locus est la position d'un gène dans le chromosome.

4.5.Génotype :

Dans les systèmes naturels, l'ensemble du "matériel" génétique est appelé le génotype. Dans les AG, Le génotype est constitué de gènes situés sur des chromosomes stockés dans le noyau des cellules sous la forme d'une longue chaîne d'acide désoxyribonucléique (ADN).

L'ensemble des chaînes est appelé structure.

La représentation des données du génotype, initialement sous forme d'un vecteur binaire et plus généralement sous forme d'une chaîne de caractères.

4.6.Phénotype :

Dans les systèmes naturels, l'organisme formé par l'interaction de l'ensemble du "matériel" génétique avec son environnement est appelé phénotype qui est l'ensemble des protéines et des enzymes qui peuvent être fabriqués à partir de l'ADN.
Dans les AG, les structures décodées forment un ensemble de paramètres donnés, ou solutions ou bien points de l'espace des solutions i.e. Chaque génotype (chromosome) représente une solution potentielle à un problème d'optimisation. La valeur de cette solution potentielle est appelée le phénotype.

4.7.L'individu :

En biologie, un individu est une forme qui est le produit de l'activité des gènes. Pour un AG, il se réduit à un chromosome ; on parle donc de chromosome ou d'individu pour désigner le même objet, qui est une solution.

Les individus correspondent aux "solutions" du problème à optimiser. Ces solutions doivent être "codées" pour que le traitement puisse être effectué par l'algorithme génétique. Cette représentation codée d'une solution s'appelle un chromosome et est composée de gènes. Chaque gène peut représenter une variable, un élément de la solution, ou une partie plus abstraite.

La méthode de codage la plus couramment utilisée par l'algorithme génétique est le codage vectoriel. Chaque solution est représentée par un vecteur. Ce vecteur peut être binaire ou de tout type discret dénombrable (nombre entier, caractères, etc.). On peut aussi utiliser un type continu (par exemple les nombres réels) mais dans ce cas, il faut aussi revoir les opérations qui modifient le contenu des chromosomes (la fonction qui crée aléatoirement les chromosomes et les opérateurs génétiques).

4.8.Population :

Dans un système naturel, une population est simplement un groupe d'individus.

Par analogie, il est défini comme l'ensemble des chromosomes (individus). On l'appelle aussi une génération ou un ensemble de solutions possibles.
Le but des algorithmes génétiques est d'explorer une partie de l'ensemble des solutions possibles (population) afin de déterminer la solution la plus appropriée (c'est-à-dire une solution qui obtient le meilleur résultat pour une fonction objectif donnée).
Pour ce faire, ils travaillent à partir d'un ensemble initial de chromosomes, constitué de gènes.

4.9.Genèse:

C'est le point de départ de notre algorithme, il s'agit d'une population initiale de taille N.

4.10. Population initiale :

Habituellement, au départ d'un algorithme génétique, il faut créer une population d'individus. Ces individus sont générés par une fonction simple. Cette fonction affecte à chaque individu qu'elle génère une valeur aléatoire pour chacun de ses gènes. L'algorithme génétique peut également utiliser comme population de départ une population déjà créée à priori qui peut être le résultat d'une autre stratégie, la solution serait dans ce cas certes meilleure puisqu'on part d'une solution approchée qui substitue une solution aléatoire.
Le choix de la population initiale a une importance majeure sur les résultats obtenus par la(es) fonction(s) objectif(s) et la diversité de la population doit être entretenue aux cours des générations afin d'explorer le plus largement possible l'espace de recherche. C'est le rôle des opérateurs de croisement et de mutation.

4.11. Une fonction objectif :

f (ou plusieurs) appelé aussi fonction de coût, fonction fitness ou fonction d'évaluation, qui pour toute solution $s \in S$ affecte une valeur fs appelée valeur de fitness.
Cette fonction donne, en valeur numérique (habituellement réelle), la qualité d'un individu. Le résultat fournit par la fonction d'évaluation va permettre de sélectionner ou de refuser un individu pour ne garder que les individus ayant le meilleur coût en fonction de la population courante. Cette méthode permet de s'assurer que les individus performants seront conservés, alors que les individus peu adaptés seront progressivement éliminés de la population.

Exemple : pour le PVC (problème de voyageurs de commerce), la fonction d'évaluation utilisée calcule la distance parcourue par le commis voyageur pour un chemin donné.

4.12. Évaluation :

C'est la phase de calcul de la fonction de fitness, où on calcule la fitness de chaque individu, par rapport à son phénotype.

L'évaluation d'un individu ne dépendant pas de celle des autres individus, le résultat fournit par la fonction d'évaluation va permettre de sélectionner ou de refuser un individu pour ne garder que les individus performants seront conservé, alors que les individus peu adaptés seront progressivement éliminés de la population.

4.13. Les parents :

Dans un système naturel, les individus peuvent se reproduire en créant de nouveaux individus formant une nouvelle génération pour assurer la continuité de la vie.

Dans le contexte d'une AG, les parents correspondent aux individus qui peuvent s'imposer pour donner naissance à de nouveaux individus descendants afin de former une nouvelle génération.

Le nombre d'individus qui coopèrent pour créer un nouvel individu est souvent égal à 2.

On parle alors de parents qui génèrent des enfants. Cependant, il est également possible de combiner plus de 2 solutions pour créer des enfants. Certaines méthodes évolutionnistes utilisent par exemple l'information contenue dans l'ensemble de la population pour créer un nouvel individu.

4.14. La progéniture :

C'est l'ensemble des nouveaux chromosomes obtenus par les processus de croisements et mutations.

***4.15.Une génération* :** est une itération de l'algorithme et correspond à une population,

Exemple : une génération n est l'ensemble des chromosomes obtenus lors de la n ième itération.

5. Le fonctionnement d'un algorithme génétique :

Les algorithmes génétiques sont, à l'heure actuelle, l'approche la plus utilisée parmi les méthodes évolutives. Ils sont des algorithmes d'optimisation s'appuyant sur des techniques dérivées de l'évolution naturelle : croisements, mutations et sélections basés sur la survie du meilleur. Ils travaillent sur une population de solutions (individus), plusieurs solutions en parallèle, et non pas sur une solution unique. Ce qui, couplé à des opérateurs aléatoires réduits la possibilité de tomber sur un optimum local.

Le fonctionnement de tout AG peut être décrit par le principe illustré par la figure-12-.En règle générale, pour pouvoir exploiter efficacement le potentiel d'un AG, indépendamment du problème traité, il est demandé de prendre en compte les 5 étapes suivantes :

– création d'une population initiale
– calcul des objectifs pour chaque individu de la population (étape "d'évaluation")
– création de nouveaux individus par mutations et/ou croisements
– formation d'une nouvelle population avec d'éventuels nouveaux individus
– réitération du processus sur plusieurs générations.

On commence par générer une population d'individus de façon aléatoire. Pour passer d'une génération k à la génération k+1, les trois opérations suivantes sont répétées pour tous les éléments de la population k. Des couples de parents P1 et P2 sont sélectionnés en fonction de leurs adaptations. L'opérateur de croisement leur est appliqué avec une probabilité Pc (généralement autour de 0.6) et génère des couples d'enfants C1 et C2. D'autres éléments P sont sélectionnés en fonction de leur adaptation. L'opérateur de mutation leur est appliqué avec la probabilité Pm (Pm est généralement très inférieur à Pc) et génère des individus mutés P'. Le niveau d'adaptation des enfants (C1, C2) et des individus mutés P' sont ensuite évalués (Sélectionnés) avant insertion dans la nouvelle population.

Différents critères d'arrêt de l'algorithme peuvent être choisis :

Le nombre de générations que l'on souhaite exécuter peut être fixé à priori. C'est ce que l'on est tenté de faire lorsque l'on doit trouver une solution dans un temps limité.

L'algorithme peut être arrêté lorsque la population n'évolue plus ou plus suffisamment rapidement.

Si les conditions d'arrêt sont vérifiées STOP, S = {les meilleurs individus}.

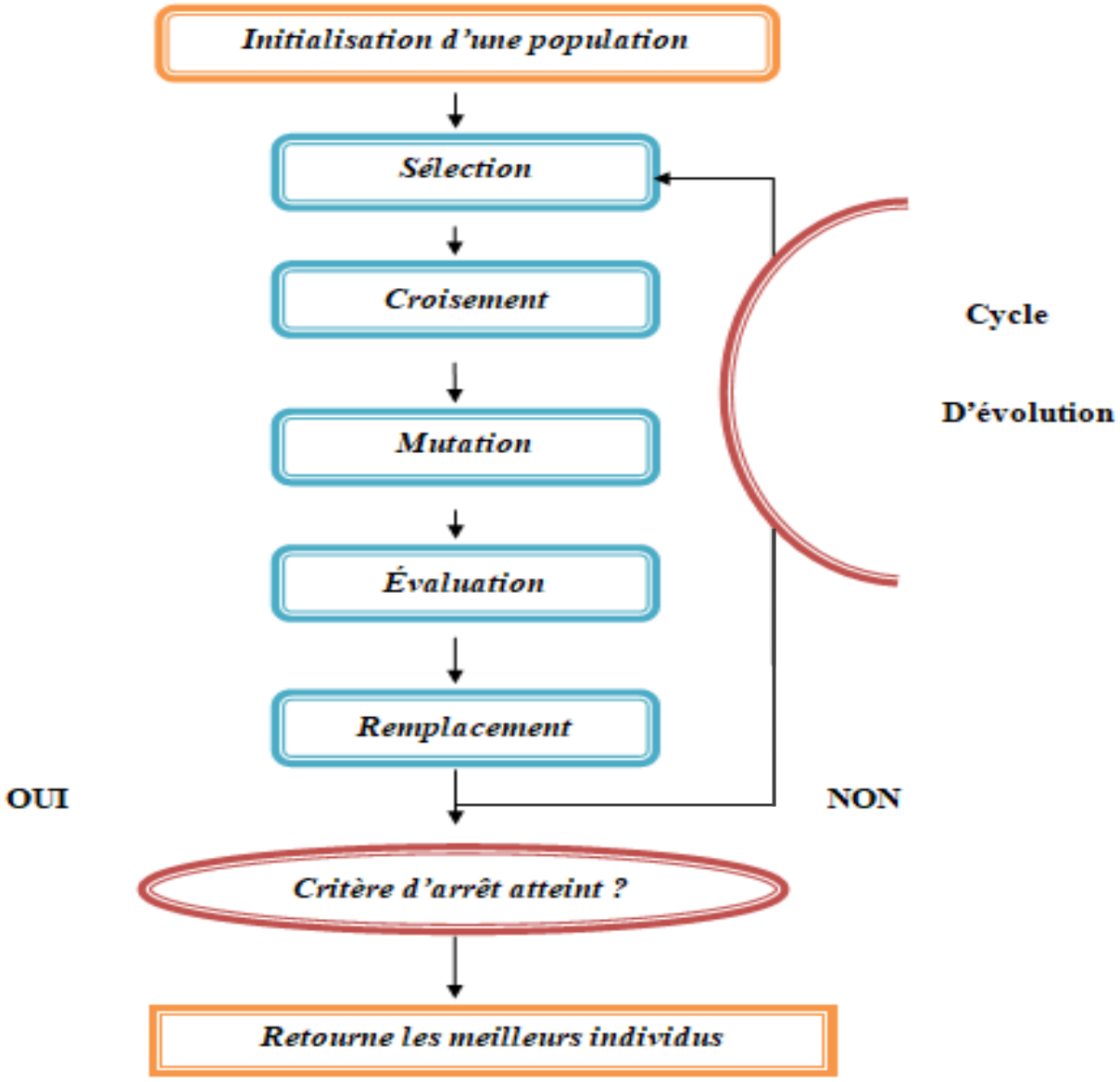


Figure 16 : Organigramme de fonctionnement d'un algorithme génétique.

6. Les méthodes de codage :

Premièrement, il faut représenter les différents états possibles de la variable dont on cherche la valeur optimale sous une forme utilisable par un AG : c'est le codage des solutions (individus), un des facteurs les plus importants, si ce n'est le plus important.

Le choix du codage est souvent délicat. L'objectif est bien sûr d'abord de pouvoir coder n'importe quelle solution. Mais il est souhaitable, au-delà de cette exigence, d'imaginer soit un codage tel que toute chaîne de caractères représente bien une solution réalisable du problème, soit un codage qui facilite ensuite la conception du croisement de telle sorte que les « enfants » obtenus à partir de la recombinaison de leurs « parents » puissent être associés à des solutions réalisables, au moins pour un grand nombre d'entre eux ***(Jean BÉRARD, 26 juin 2000)***.

Parmi les techniques les plus fréquemment utilisées pour coder les individus, on distingue :

6.1. Codage binaire :

Dans ce type de codage, pour un individu on code ses variables et on les concatène. Le gène est codé par un caractère binaire, 0 ou 1. C'est le plus courant et celui qui a été employé lors de la première application des algorithmes génétiques.par exemple, la chaîne binaire

Ch :

1000	**0110**	**1101**

Correspond à un individu défini par 3 variables (8, 6, 13) en codage binaire naturel sur 4 bits pour chacune. C'est le codage le plus utilisé pour plusieurs raisons : pour des raisons historiques, ce codage a été utilisé par J. Holland et ses étudiants ; plusieurs résultats théoriques sont basés sur ce codage, et il est facile de mettre en place les opérateurs génétiques (sélection, croisement et mutation) avec ce codage .Ce pendant si la longueur de la chaîne augmente la performance de l'algorithme diminuera.

6.2. La fonction de décodage d :

Elle doit permettre le passage de la représentation binaire vers la représentation en termes d'états de la variable initiale. Si cette variable prend des valeurs entières, nous devons avoir : $\boldsymbol{d}: \{\mathbf{0}, \mathbf{1}\}^{\boldsymbol{l}} \rightarrow \mathbb{N}$ où l est la longueur de la chaîne). Le décodage le plus souvent retenu est un simple changement de base. Par exemple la chaîne A = {0, 0, 0, 1, 1} peut être décodée de manière à donner

$$\mathbf{0} \times \mathbf{2^4} + \mathbf{0} \times \mathbf{2^3} + \mathbf{0} \times \mathbf{2^2} + \mathbf{1} \times \mathbf{2^1} + \mathbf{1} \times \mathbf{2^0} = \mathbf{3}$$

De manière plus générale, toute chaîne binaire A peut donc être décodée en une valeur entière x selon la règle suivante :

$$x = d(A) = \sum_{i=1}^{l} a_i 2^{l-i}$$

D'une façon générale il existe deux types de difficultés dans le choix d'un codage :

- La capacité de pouvoir s'adapter au problème de façon à limiter la taille de l'espace de recherche et engendrer les nouveaux chromosomes le plus possible (c'est à dire : Coder des solutions qui soient réalisables).

- Facilité de manipulation dans les opérations de génération de nouveaux gènes.

6.3. Codage réel :

Une autre manière de coder les chromosomes d'un algorithme génétique est le codage à l'aide de caractères multiples (par opposition aux bits). Il s'agit de concaténation des variables xi d'un individu x, par exemple un individu x (25, 31, 8) est codé par :

Ch :

25	**31**	**8**

Il a le mérite d'être simple. Chaque chromosome est en fait un vecteur dont les composantes sont les paramètres du processus d'optimisation. Par exemple, si on recherche l'optimum d'une fonction de n variables $f(x_1, x_2, ..., x_n)$, on peut utiliser tout simplement un chromosome ch contenant les n variables :

Ch :

X1	**X2**	**...**	**Xn-1**	**Xn**

Avec ce type de codage, la procédure d'évaluation des chromosomes est plus rapide vu l'absence de l'étape de transcodage (du binaire vers le réel). Ce dernier est plus précis que le codage binaire, l'espace de recherche est similaire à celui du problème, l'évaluation de la fonction coût est plus rapide ; mais son alphabet est infini et il a besoin d'opérateurs

appropriés. Néanmoins ce codage présent l'inconvénient majeur de la non-réalisabilité des solutions obtenues (dans une bonne partie des cas) après application de différents types d'opérateurs.

7. Les opérateurs de reproduction d'un algorithme génétique :

Les algorithmes génétiques sont basés sur un phénomène naturel : l'évolution. Plus précisément, ils supposent, qu'a priori, deux individus adaptés à leur milieu donnent, par recombinaison de leurs gènes, des individus mieux adaptés. Pour ce faire, trois opérateurs sont à disposition : la sélection, le croisement et la mutation, plus un opérateur optionnel, l'élitisme.

Plus précisément, à l'aide des individus anciens on prélève dans certains individus de la population courante, une partie de leurs caractéristiques en choisissant certaines parties des chromosomes qui les représentent ; puis on recombine ces différentes parties pour constituer les individus de la nouvelle population ***(Ahmad Hassanat, 2019)***.

7.1. Les méthodes de sélection :

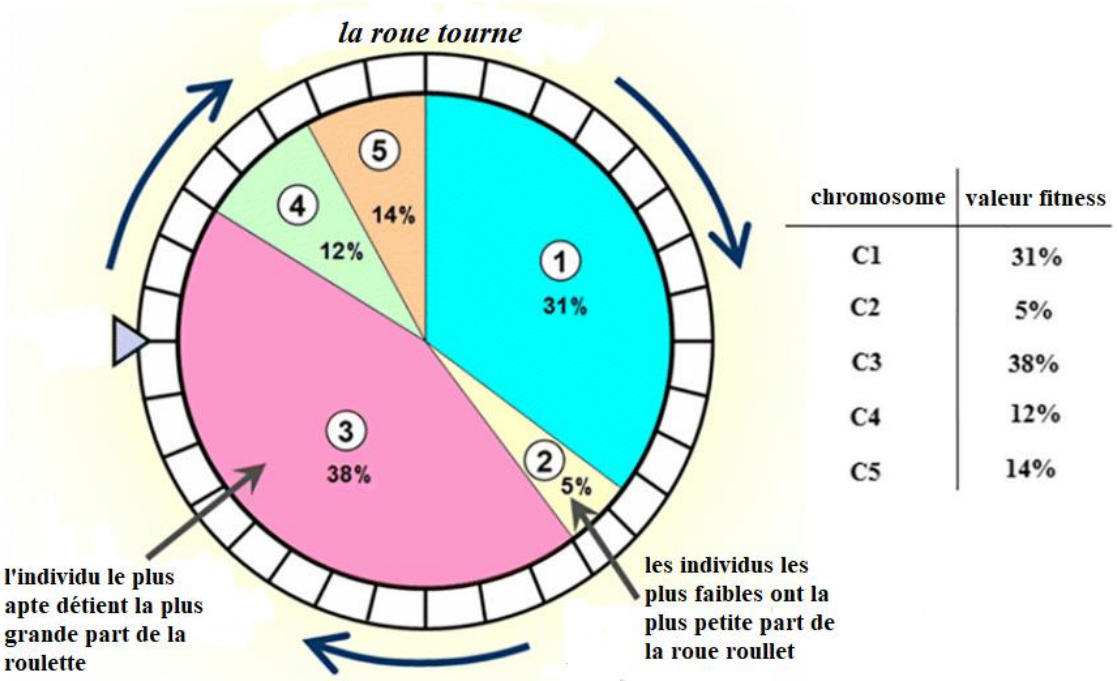
La sélection est un opérateur essentiel pour améliorer la qualité d'une population. Son objectif est de retenir et multiplier les meilleurs individus qui contribueront à l'opération de croisement, et d'éliminer les plus mauvais.

Cet opérateur permet de définir quels seront les individus de $\boldsymbol{P_i}$ qui vont être dupliqués dans la nouvelle population $\boldsymbol{P_{i+1}}$ et vont servir de parents (application de l'opérateur de croisement). Soit $\boldsymbol{N}$ le nombre d'individus de P, on doit en sélectionner $\boldsymbol{N/2}$ (l'opérateur de croisement nous permet de repasser à N individus). Les deux principes de sélection suivants sont les plus couramment utilisés ***(Allaoua, 2017)*** , ***(Cheriet, 2016)***:

7.1.1. Sélection par roulette (Wheel):

Les parents sont sélectionnés en fonction de leurs performances. Meilleur est le résultat codé par un chromosome, plus grandes sont ses chances d'être sélectionné. Il faut imaginer une sorte de roulette de casino sur laquelle sont placés tous les chromosomes de la population, la place accordée à chacun des chromosomes étant en relation avec sa valeur

d'adaptation. Ensuite, la bille est lancée et s'arrête sur un chromosome. Les meilleurs chromosomes peuvent ainsi être tirés plusieurs fois et les plus mauvais ne sera jamais sélectionnés.



chromosome	valeur fitness
C1	31%
C2	5%
C3	38%
C4	12%
C5	14%

Figure 17: Roue servant à sélectionner un individu pour le croisement.

7.1.2. Sélection par tournoi :

Cette méthode de sélection augmente les chances des individus de "mauvaise qualité" par rapport à leur fitness, de participer à l'amélioration de la population. En effet, c'est une compétition entre les individus d'une sous-population de taille M (M ≤ N) prise au hasard dans la population. Le paramètre M est fixé à priori par l'utilisateur. L'individu de meilleure qualité par rapport à la sous-population sera considéré comme vainqueur et sera sélectionné pour l'application de l'opérateur de croisement. Le paramètre M joue un rôle important dans la méthode du tournoi. Par conséquent le choix de M permet de faire varier la pression sélective. De cette manière, on contrôle les chances de sélection des individus les plus performants par rapport aux plus faibles.

Dans le cas où M = N avec N est la taille de la population. Le résultat par la sélection de la méthode du tournoi donne à chaque fois un seul individu qui réduit l'algorithme génétique à un algorithme de recherche local travaillant sur une seule solution à la fois. Ce type d'algorithmes a pour inconvénient de converger parfois rapidement vers un optimum local. Dans le cas M = 1, la méthode de sélection du tournoi correspond à la sélection aléatoire.

Des études comparatives entre les différentes méthodes de sélection ont montré que la sélection par rang et la sélection par tournoi donnent de meilleurs résultats. Ces méthodes ont été comparées sur des critères de qualité des résultats obtenus et de vitesse de convergence

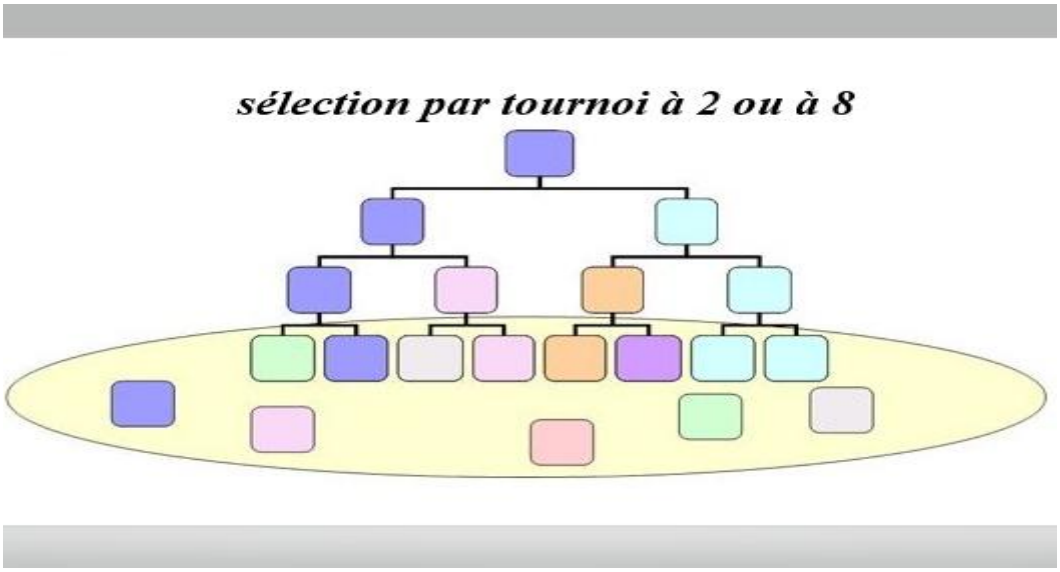


Figure 18 : sélection des individus par tournoi pour le croisement.

***7.1.3. Sélection par rang* :**

La sélection par rang trie d'abord la population par fitness. Chaque chromosome se voit associé un rang en fonction de sa position. Le plus mauvais chromosome aura le rang 1, le suivant 2, et ainsi de suite jusqu'au meilleur chromosome qui aura le rang N (pour une population de N chromosomes). La sélection par rang d'un chromosome est la même que par roulette, mais les proportions sont en relation avec le rang plutôt qu'avec la valeur de l'évaluation. Avec cette méthode de sélection, tous les chromosomes ont une chance d'être sélectionnés. Cependant, elle conduit à une convergence plus lente vers la bonne solution. Ceci est dû au fait que les meilleurs chromosomes ne diffèrent pas énormément des plus mauvais.

Il existe encore de nombreuses méthodes de sélection telles que : Sélection uniforme, Probabilité de sélection proportionnelle à l'adaptation, Sélection Steady-State,...etc. Mais par manque de temps et afin d'être le plus concis possible, nous ne pouvons pas tous les expliquer à part ce qui a été mentionné plus haut.

7.2. Les méthodes de croisement :

Le croisement est l'opérateur principal des algorithmes génétiques (AG). Son rôle est de choisir au hasard deux individus parents parmi ceux sélectionnés avec une probabilité Pc appelée probabilité de croisement pour les combiner et créer deux nouveaux individus enfants, ceci afin d'enrichir la diversité de la population. Donc cet opérateur joue un rôle important dans la convergence de l'AG, en lui permettant de concentrer une partie de la population autour des meilleurs individus. Sans oublier que les méthodes de croisement sont liées au codage, et donc le croisement de chromosomes codés en binaire ne sera pas le même que celui d'un chromosome codé en entier, mais le principe de base reste identique ***(Jean BÉRARD, 26 juin 2000)***.

Pour conclure sur les opérateurs de croisement, nous en présentons quelques types :

7.2.1. Le croisement à un point :

Pour chaque couple, on choisit au hasard un point de croisement (figure-19-). Le croisement s'effectue directement au niveau binaire, et non au niveau des gènes. Un croisement peut être coupé au milieu d'un gène.

Il consiste à diviser chacun de deux parents en deux parties à la même position, choisie au hasard. Le premier enfant est composé de deux parties des deux parents. La première partie est celle du premier parent et la deuxième partie est celle du deuxième parent. Le deuxième enfant est composé de deux parties, une première partie du deuxième parent et la deuxième partie du premier parent.

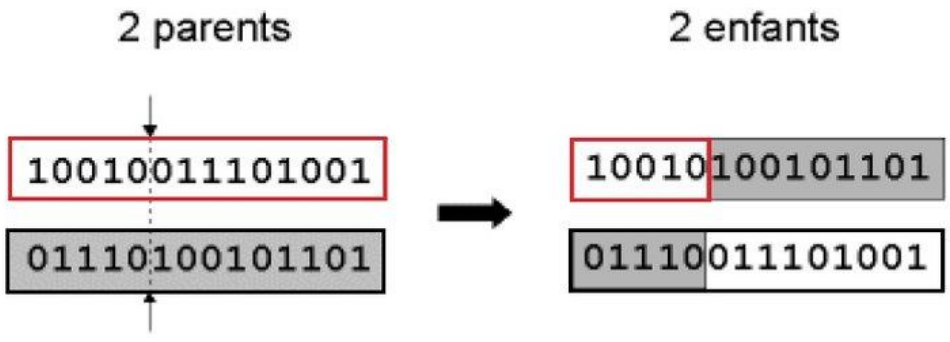


Figure 19 : Représentation schématique de Croisement à un point.

7.2.2. Croisement à deux points :

On choisit au hasard deux points de croisement. Cette méthode consiste à fixer deux positions et dépend de ces deux derniers, le premier enfant sera la copie du premier parent en remplaçant sa partie entre les deux positions par celle du deuxième parent. La même opération sera appliquée pour déterminer le deuxième enfant en inversant les rôles du premier parent et du deuxième parent.

Les mathématiciens ont utilisé cet opérateur car il est généralement considéré comme plus efficace que le précédent. Néanmoins ils n'ont pas constaté de différence notable dans la convergence de l'algorithme (figure-20-).

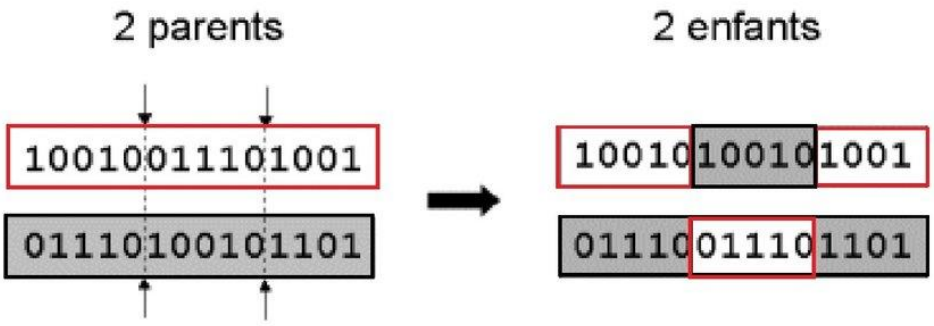


Figure 20 : Représentation schématique de Croisement à deux points.

7.3. Les méthodes de mutation :

L'objectif de la mutation dans les GA est d'introduire de la diversité dans la population échantillonnée. Les opérateurs de mutation sont utilisés pour tenter d'éviter les minima locaux en empêchant la population de chromosomes de devenir trop similaire les uns aux autres, ce qui ralentit ou même arrête la convergence vers l'optimum global ***(Bontemps, 2000)***.

La mutation est un changement aléatoire selon une certaine règle probabiliste qui a lieu sur les génotypes, avec une faible probabilité Pm (fixée par l'utilisateur) de la valeur d'un ou plusieurs allèles d'un chromosome.

Classiquement, la mutation est une inversion d'un bit au hasard ou remplacement au hasard d'un caractère par un autre, un petit nombre de gènes, avec un faible taux de probabilité (n'est pas élevé à 5%). Bien entendu, comme pour le croisement, la mutation dépend du problème.

Les AGs utilisent cet opérateur pour conserver la diversité de la population et évite en particulier à l'AG de converger vers un optimum local, et dans le but d'explorer, dans le domaine de conception, de nouvelles zones de solution afin de localiser l'optimum global.

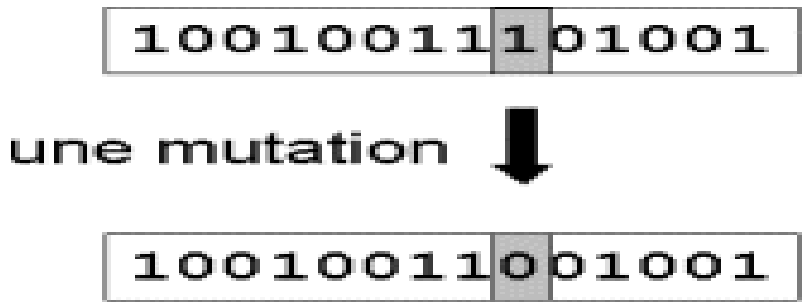


Figure 21: Représentation schématique de Mutation.

***7.3.1. Mutation en codage binaire* :**

Un ou plusieurs gènes, selon la taille du chromosome, et un taux gènes mutés gènes totales très faible sont choisis aléatoirement. Ils sont alors inversés (1→0 et 0→1) (figure.).

Cependant, on trouve plusieurs opérateurs de mutation :

• Transposition de deux allèles consécutifs : Cette mutation consiste à choisir deux allèles consécutifs au hasard et d'échanger leurs valeurs respectives.

• Transposition de deux allèles quelconques : Cette mutation consiste à choisir deux allèles au hasard et d'échanger leurs valeurs respectives

• Inversion d'allèles : Cette mutation consiste à choisir deux allèles au hasard et d'inverser l'ordre des allèles contenus dans la zone sélectionnée.

Il existe encore de nombreuses méthodes de mutations telles que : Mutation locale, Mutation uniforme...etc. Mais par manque de temps et afin d'être le plus concis possible, nous ne pouvons pas tous les expliquer à part ce qui a été mentionné plus haut.

Dans la figure ci-dessous on peut voir le rôle des opérateurs génétiques dans la reproduction de population. (String représente le chromosome ou l'individu)

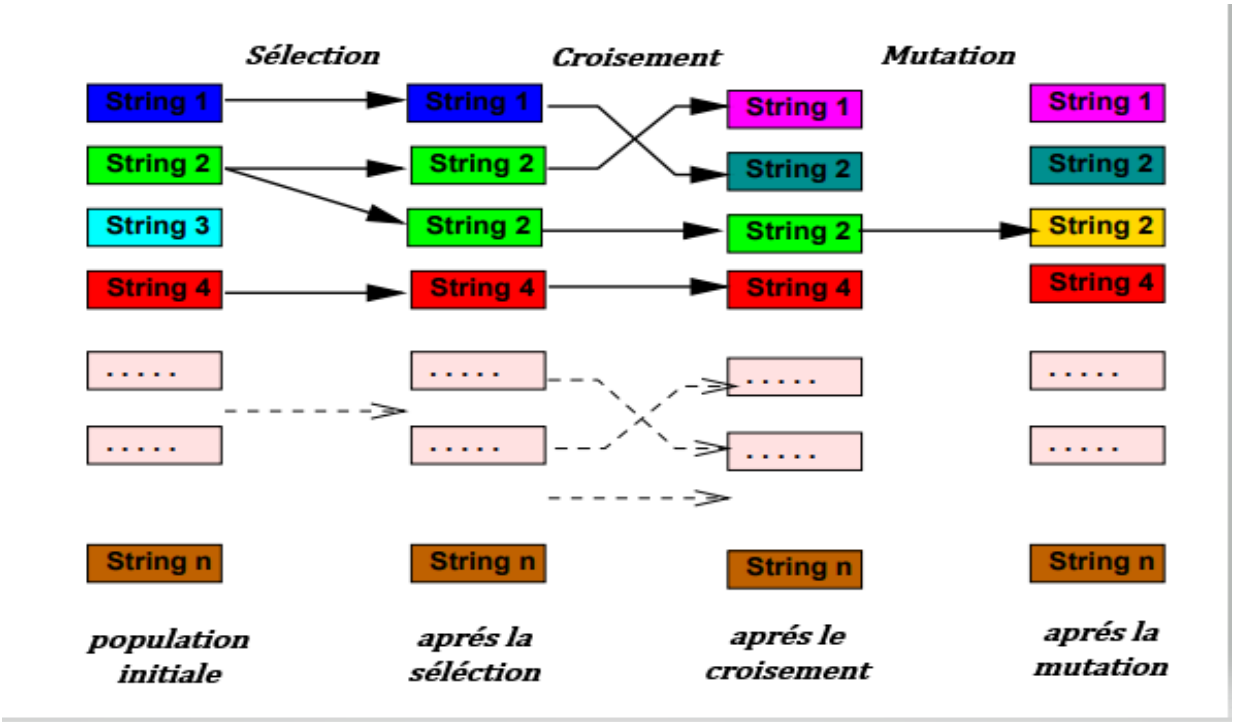


Figure 22 : Une vision générale sur la reproduction génétique.

8. Remplacement :

Le remplacement consiste à déterminer quels sont les individus de la génération courante, constituée de parents et d'enfants, qui seront les parents de la génération suivante.
Comment réinsérer le fils dans le groupe ? Les solutions sont :

- Éliminer le moins bon du groupe.
- Éliminer l'individu qui ressemble le plus au nouvel individu.
- Remplacer un des deux parents.
- La nouvelle génération remplace l'ancienne génération sauf le meilleur individu de l'ancienne génération qu'il ne faut pas surtout pas supprimer sous peine de voir l'adaptation globale diminuer !

9. Les tests d'arrêts :

C'est la condition qui détermine la fin de l'évolution de la population. Plusieurs modes de ce critère peuvent être adoptés selon les résultats empiriques et les contraintes du problème utilisé, parmi lesquels nous trouvons :

- Arrêt après exécuter l'algorithme pendant un nombre d'itérations proportionnel à la taille du problème.
- Arrêt après consommation d'un temps estimé proportionnel à la taille du problème.
- Arrêt s'il n'y a pas d'amélioration de la qualité pendant un certain nombre d'itérations qui doit encore être dépendant de la taille du problème.
- Arrêt si une certaine valeur seuil prédéfinie de la solution recherché est atteinte.
- Arrêt si la différence entre les deux solutions les plus récentes dépasserait un certain seuil prédéfini.

10. Les paramètres d'un algorithme génétique :

Les opérateurs de l'AG sont guidés par un certain nombre de paramètres généralement fixés à l'avance et dont dépend très fortement la « bonne » convergence de l'algorithme. Présentons brièvement les principaux paramètres de l'AG ***(Bontemps, 2000)***.

10.1. La taille de la population et la longueur du codage de chaque individu :

Il est conseillé de prendre comme taille de la population la valeur correspondant à la longueur du codage des individus. En effet, si cette taille est très grande, l'évaluation de tous les individus de la population peut s'avérer trop longue. Par contre, si elle est très petite, l'algorithme peut converger trop rapidement.

10.2. La probabilité de croisement Pc :

Elle dépend de la forme de la fonction d'évaluation. Son choix est généralement expérimental et sa valeur est très souvent prise entre 0.5 et 0.9. Plus elle est élevée, plus la population subit des changements importants. Ainsi, la convergence est très rapide si le taux de croisement est proche de 1.

10.3. La probabilité de mutation Pm :

Le taux de mutation est généralement faible, le risque d'un taux élevé étant de modifier les meilleurs individus et ainsi, de s'éloigner de l'optimalité.

***Exemple d'optimisation mono-objectif* :**

Le pseudo-code : algorithme génétique.

1. Définition du problème à résoudre (Fonction Fitness)
2. Initialisation des paramètres de l'algorithme

(N-gen : nombre de génération, N-pop : taille de pop, Xmin, Xmax, Pc, Pm, n-bit)

3. **Pop=** Initialisation de la population
4. Évaluation de la population **Pop-1**
5. **Tant que** (i ≤ N-gen) **alors**

// Appliqué les opérateurs de reproductions « les enfants » //

6. Codage en binaire des individus de **Pop-i**
7. **Enfants-i =** croisement avec probabilité Pc dans la population **Pop-i** (Parents)
8. **Enfants-i** = mutation avec probabilité **Pm** des **Enfants-i**
9. **Enfants-P =** Décoder du génotype vers le phénotype **Enfants-i**.
10. Évaluation de la population **Enfants-P**

// Sélection par roulette //

11. **Pop-(i+1) =** sélection les plus aptes individus {**Enfants-P, Parents**}
12. Fin tant que.

Nous avons proposé l'application de le Fonction ***F6*** de « Schaffer's fonction » dans l'algorithme génétique.

Le nombre de génération : 50

La taille de population : 50

La probabilité de croisement : 0.8

La probabilité de mutation : 0.03

Xmin = -100 , Xmax = 100

Présentation du problème du F6-schaffer :

« Fonction F6 de Schaffer est une fonction de test qui comprend de nombreuses oscillations/pics vers lesquels il est difficile pour les techniques d'escalade de converger (les pics sont appelés optima locaux). La fonction F6 est conçue pour avoir son pic à l'origine avec une valeur de un. »

Formulation Mathématique :

$$f(x,y) = 0.5 + \frac{sin^2\left(\sqrt{x^2+y^2}\right) - 0.5}{[1 + 0.001 \cdot (x^2+y^2)]^2}$$

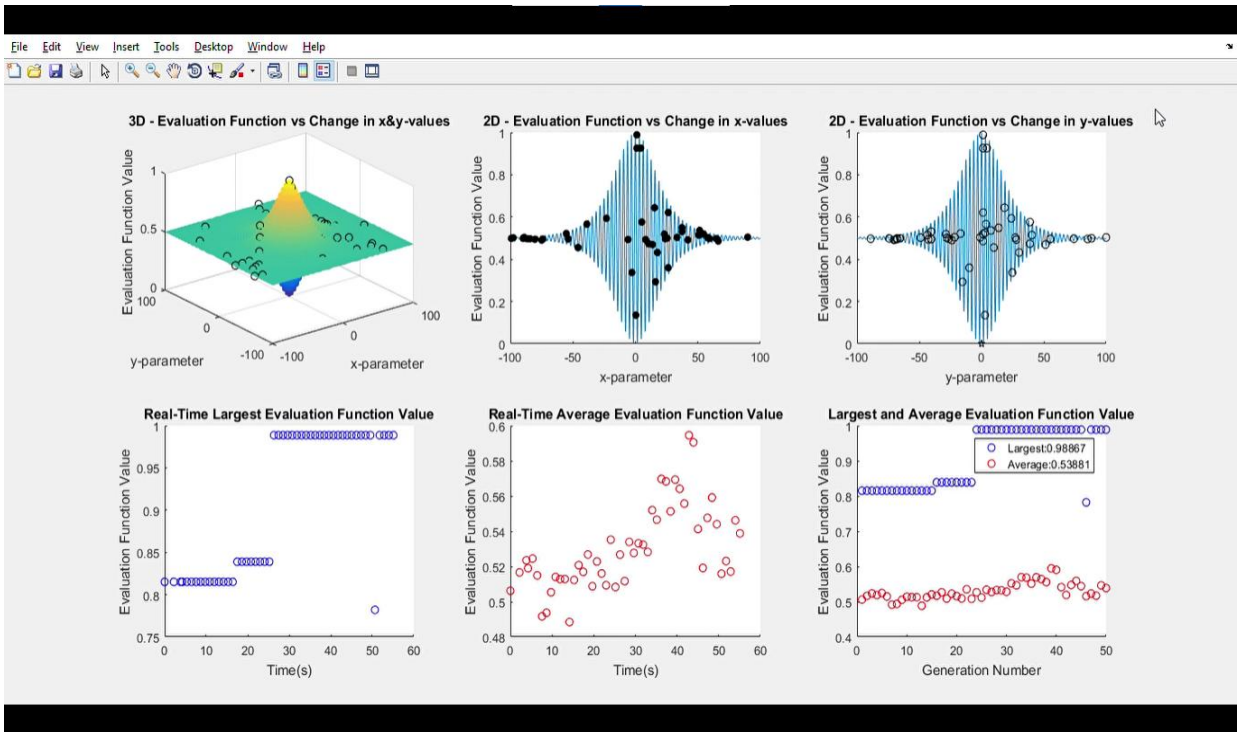


Figure 23: résolution de F6-SCHAFFER par le GA.

En effet, la figure -23- représente, le fonctionnement de l'algorithme génétique de la recherche vers l'optimum global de la fonction F6 Schaffer. La figure se compose de six graphes. Le premier en 3D, représentent les valeurs de la fonction d'évaluation en fonction de x et y permettant d'avoir un aperçu panoramique sur les optimums des générations. Les deuxièmes et troisièmes graphes (2D) représentent, respectivement, les valeurs d'évaluation de la fonction F6 Schaffer en fonction de x et y.

Le quatrième graphe représente la meilleure valeur prise à chaque génération et le cinquième graphe montre la moyenne des valeurs trouvées à chaque génération au cours du processus.

On voit bien, d'après le sixième graphe, que nous obtenons une meilleure fitness de notre dernière génération (voir fig23).

$$f(x*) = \mathbf{0.988669621141524}$$
$$(x*, y*) = (\mathbf{-1.17647058823529}, \mathbf{-1.17647058823529})$$

***Conclusion* :**

Dans ce chapitre, nous avons présenté un état de l'art sur l'un des algorithmes évolutionnaire appelés algorithmes génétiques. Nous avons expliqué comment les algorithmes génétiques exploitent le principe de la génétique darwinienne pour résoudre les différents types de problèmes d'optimisation d'une manière générale.

Le chapitre suivant présentera les concepts de base d'optimisation multi-objectif et l'un de ses algorithmes génétiques, l'algorithmes développés : NSGA-II (Algorithmes Génétiques sans Tri Dominé), nous parlerons de ce dernier en détail puisque nous allons l'utiliser dans nos recherches pour résoudre notamment des problèmes d'optimisation multi-objectif.

Chapitre III : l'optimisation multi-objectifs

***Introduction* :**

Contrairement à l'optimisation classique mono-objectif, où il s'agit de chercher une solution réalisable dite optimale, dans l'optimisation multiobjectifs la solution optimale n'est pas unique, il s'agit d'un ensemble dit "ensemble de solutions efficaces" où "Solutions de Pareto". En effet, c'est après avoir trouvé les solutions de problèmes multi-objectifs que d'autres difficultés surviennent : il faut sélectionner une solution de cet ensemble. La solution qui sera choisie par l'utilisateur va refléter les compromis opérés par le décideur (DM) en anglais « Decision Maker » vis-à-vis de fonctions objectives considérées.

Dans ce chapitre, un ensemble des définitions et des concepts de base de l'optimisation multi-objectif sont définis formellement, suivi par une explication concernant l'une des méthodes évolutionnaires de résolution des problèmes d'optimisation multi-objectifs
Le NSGA-II.

1. L'optimisation multi-objectif :

Définition : Un problème multi-objectifs ou multicritères peut être défini comme un problème dont on recherche l'action qui satisfait un ensemble de contraintes et optimise un vecteur de fonctions objectifs. Les problèmes d'optimisation ont en général plusieurs solutions car la définition d'un optimum ne peut pas être établie dans les problèmes multiobjectifs.

L'optimisation multi-objective peut être définie comme la recherche de la meilleure ou des meilleures solutions possibles d'un problème donné. Souvent, les problèmes d'optimisation sont des problèmes multi-objectifs. Si les objectifs sont antagonistes, alors on n'a pas une seule solution optimale mais un ensemble de solutions de compromis.

Donc elle permet de modéliser des problèmes réels faisant intervenir de nombreux critères (souvent conflictuels) et contraintes. Dans ce contexte, la solution optimale recherchée n'est plus un simple point, mais un ensemble de bons compromis satisfaisant toutes les contraintes.

2. Formulation mathématique du problème d'optimisation multi-objectif :

Un problème d'optimisation peut s'écrire sous la forme générale suivante :

$$\boldsymbol{Minimiser}\ \boldsymbol{f_m(X), m = 1, \dots, M};$$

$$\textbf{Satisfaisant} \begin{cases} \boldsymbol{g_j(x) \geq 0, m = 1, \dots, J}; \\ \boldsymbol{h_k(x) = 0,\ k = 1, \dots, K}; \\ \boldsymbol{x_i^l \leq x_i \leq x_i^S, i = 1, \dots, n}; \end{cases}$$

Avec :

- $\boldsymbol{X = [x_1\ x_2\ \dots\ x_n]^T}$ **:** C'est le vecteur de décisions avec $\boldsymbol{xi}$ les variables du problème et $\boldsymbol{n}$ le nombre de variables de décisions.
- $\boldsymbol{(f_m(X), m = 1, \dots, M)}$**:** Le vecteur de fonction objectif avec $\boldsymbol{f_m}$ les objectifs ou critères de décision et **M** le nombre d'objectifs.
- $\boldsymbol{\left(g_j(X) \geq 0, j = 1, \dots, J\right)}$ **:** ce sont les contraintes d'inégalité à satisfaire avec **J** le nombre de contraintes.
- $\boldsymbol{(h_k(X)}$**=0, k=1,...,K) :** ce sont les contraintes d'égalité à satisfaire avec **K** le nombre de ces contraintes**.**
- $\boldsymbol{(\ x_i^l \leq x_i \leq x_i^s\ , i = 1, \dots, n)}$ **:** ce sont les bornes inférieures et supérieures d'i-ème variable de décision.

On se limite aux problèmes de minimisation, puisque la maximisation d'une fonction $\boldsymbol{f(X)}$ peut facilement être transformée en un problème de minimisation :

$$\textbf{max}\ \left(\boldsymbol{f(X)}\right) = -\textbf{min}\ (-\boldsymbol{f(X)}).$$

L'union des domaines de définition de chaque variable et les contraintes forment un ensemble Ω que nous appelons l'ensemble des actions réalisables ou l'espace de recherche.
Nous appellerons $\boldsymbol{\Lambda}$ l'ensemble des objectifs réalisables ou espace des objectifs ou espace des critères comme illustre la figure-24-

Cette situation est représentée à la figure **24** est pour le cas $\boldsymbol{m = 2}$ et $\boldsymbol{k = 3}$.

Après avoir trouvé les solutions du problème multi-objectif que d'autres difficultés surviennent : il faut sélectionner une solution dans cet ensemble. La solution qui sera choisie par l'utilisateur va refléter les compromis opérés par le décideur vis-à-vis des différentes fonctions objectif ***(Wafa, 2021).***

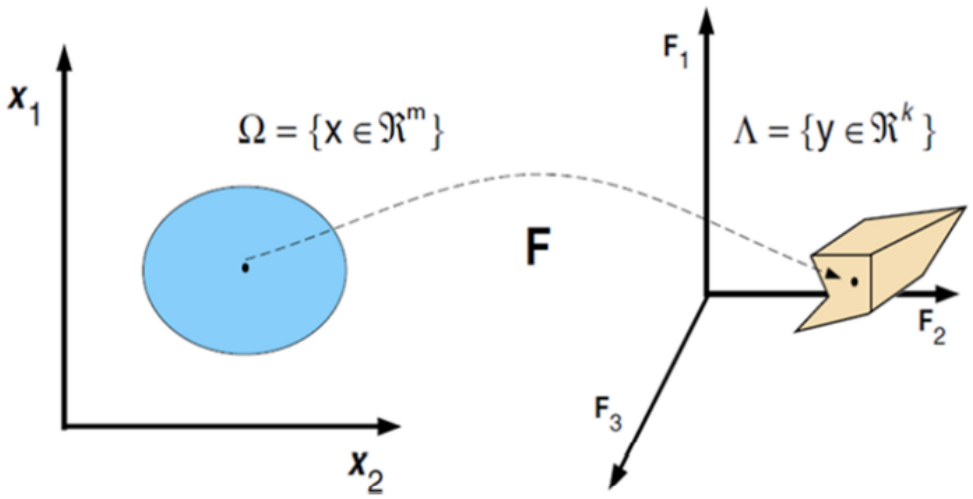


Figure 24: Représentation de l'espace de recherche et de son image par la fonction multi-objectif.

3. Exemple d'un problème multi-objectif :

Artisan du cuir fabriquant des sacs et des manteaux sur commande spéciale. L'artisan et ses entraîneurs travaillent 8 heures par jour. Sachant que tous les sacs et manteaux utilisent la même quantité de cuir pour être fabriqués.

Un sac prend une heure et un manteau prend 90 minutes. L'artisan doit faire au moins 2 articles par jour pour être rentable et peut faire au plus 5 sacs et 4 manteaux par jour. L'intérêt net est de 3000 dinars pour le sac et de 2000 dinars pour le manteau. La fabrication du sac occasionne une perte de 60 cm et 50 cm de cuir du manteau. L'artisan veut maximiser ses profits et minimiser ses déchets, ce qui nous donne un problème au format MOP (problème multi-objectifs).

Notons par $\boldsymbol{x_1}$ le nombre de sacs et $\boldsymbol{x_2}$ le nombre de manteaux à fabriquer par jour.

- **Les contraintes sont :**

1. $\boldsymbol{x1 + \frac{3}{2}x2 \leq 8}$: temps de travail.
2. $\boldsymbol{x1 + x2 \geq 2}$: la commande minimale.
3. $\boldsymbol{x1 \leq 5\ et\ x2 \leq 4}$ commande maximale.

- **Les fonctions objectifs :**

1. max $\boldsymbol{f1(x) = 3000x1 + 2000x2}$ (maximiser le profit total en dinars).
2. min $\boldsymbol{f2(x) = 0.6x1 + 0.5x2}$ (minimiser le déchet total en mètres).

Le problème multiobjectifs (MOLP) qui modélise ce problème peut s'écrire comme suit : (MOLP)

$$\begin{cases} \boldsymbol{max\ f1(x) = (3x1 + 2x2) \times 1000.} \\ \boldsymbol{min\ f2(x) = (6x1 + 5x2) \times 0.1.} \\ \boldsymbol{x1 + \frac{3}{2}x2 \leq 8.} \\ \boldsymbol{x1 + x2 \geq 2.} \\ \boldsymbol{x1 \leq 5.} \\ \boldsymbol{x2 \leq 4.} \\ \boldsymbol{x1, x2 \geq 0} \end{cases}$$

4. Problématique d'une optimisation multi-objectif :

La difficulté principale d'un problème multi-objectif est qu'il n'existe pas une définition de la solution optimale. Le décideur peut simplement exprimer le fait qu'une solution est préférable à une autre mais il n'existe pas une solution meilleure que toutes les autres.

Dés lors, résoudre un problème multi-objectif ne consiste pas à rechercher la solution optimale mais l'ensemble des solutions satisfaisantes pour lesquelles on ne pourra pas effectuer une opération de classement. Les méthodes de résolution de problèmes multi-objectifs sont donc des méthodes d'aide à la décision car le choix final sera laissé au décideur.

5. Notion d'optimalité et de dominance :

Question : Comment peut-on choisir la solution optimale de ce problème multiobjectifs ? Dans l'optimisation mono-objectif, la solution optimale peut être unique ou multiple mais la valeur de la fonction objectif est unique. Dés que l'optimisation devient multi-objectif, le choix de solution selon l'optimalité n'a plus de sens mais il devient lié à la dominance de cette solution avec les autres solutions dans l'espace objectif. On parle de « solution efficace » ou « solution optimale au sens de Pareto » ou « solution non dominée".

Définitions :

$\boldsymbol{Wicksell\ et\ al.}(\mathbf{1913})\boldsymbol{Soit\ x\ et\ x'\ deux\ solutions\ de\ X.\ et} f\boldsymbol{i}(\boldsymbol{x})\ \boldsymbol{tq}\ \boldsymbol{i} \in [[\mathbf{1};\ \boldsymbol{p}]]$

- $\boldsymbol{x\ domine\ faiblement\ x'}, \boldsymbol{not}\acute{\boldsymbol{e}}\ \boldsymbol{x} \leqq \boldsymbol{x'} \Leftrightarrow \boldsymbol{fi}(\boldsymbol{x}) \leq \boldsymbol{fi}(\boldsymbol{x'}) \forall \boldsymbol{i} \in [[\mathbf{1};\ \boldsymbol{p}]]$
- $\boldsymbol{x\ domine\ x'}, \boldsymbol{not}\acute{\boldsymbol{e}}\ \boldsymbol{x} \leq \boldsymbol{x'} \Leftrightarrow \begin{cases} \boldsymbol{fi}(\boldsymbol{x}) \leq \boldsymbol{fi}(\boldsymbol{x'}) \forall \boldsymbol{i} \in [[\mathbf{1};\ \boldsymbol{p}]] \\ \boldsymbol{fi}(\boldsymbol{x}) < fi(\boldsymbol{x'}) \exists \boldsymbol{i} \in [[\mathbf{1};\ \boldsymbol{p}]] \end{cases}$
- $\boldsymbol{x\ domine\ strictement\ x'}, \boldsymbol{not}\acute{\boldsymbol{e}}\ \boldsymbol{x} < x' \Leftrightarrow \boldsymbol{fi}(\boldsymbol{x}) < fi(x') \forall \boldsymbol{i} \in [[\mathbf{1};\ \boldsymbol{p}]]$

Exemple : Par exemple, pour le cas de maximisation, considérons les trois vecteurs suivants :

$$\boldsymbol{Z_1} = \begin{pmatrix} \mathbf{12} \\ \mathbf{-2} \\ \mathbf{4} \end{pmatrix} \quad ; \ \boldsymbol{Z_2} = \begin{pmatrix} \mathbf{10} \\ \mathbf{-4} \\ \mathbf{0} \end{pmatrix} \quad ; \boldsymbol{Z_3} = \begin{pmatrix} \mathbf{10} \\ \mathbf{-2} \\ \mathbf{4} \end{pmatrix}$$

– Nous constatons que $\boldsymbol{Z_1}$ domine strictement $\boldsymbol{Z_2}$ et $\boldsymbol{Z_1}$ domine faiblement $\boldsymbol{Z_3}$.

– Par contre, $\boldsymbol{Z_3}$ domine faiblement $\boldsymbol{Z_2}$ ***(Wafa, 2021)***.

6. Optimalité de Pareto :

Pour un ensemble de solutions fini, toutes les solutions peuvent être comparées deux par deux selon le principe de dominance, et nous pouvons déduire quelle solution domine l'autre. A la fin, nous obtenons un ensemble où aucune des solutions ne domine l'autre, cet ensemble est appelé ensemble des solutions non dominées.

Définition *(Ensemble des solutions non-dominées) :*

Soit P un ensemble de solutions, l'ensemble des solutions non dominées P′ est l'ensemble des solutions non-dominées par aucun autre membre de l'ensemble ***P***.
Si l'ensemble ***P*** représente la totalité de l'espace de recherche ***S***, l'ensemble des solutions non-dominées ***P'*** est appelé ensemble « Pareto-Optimal » dans l'espace de décision ou « front de Pareto » dans l'espace des objectifs. Il s'agit de l'ensemble des solutions que l'utilisateur cherche à obtenir à travers l'optimisation. Alors l'ensemble est ***Pareto-optimal*** P est l'ensemble des solutions non-dominées de l'espace de recherche faisable ***S (Junhui Liu)***.

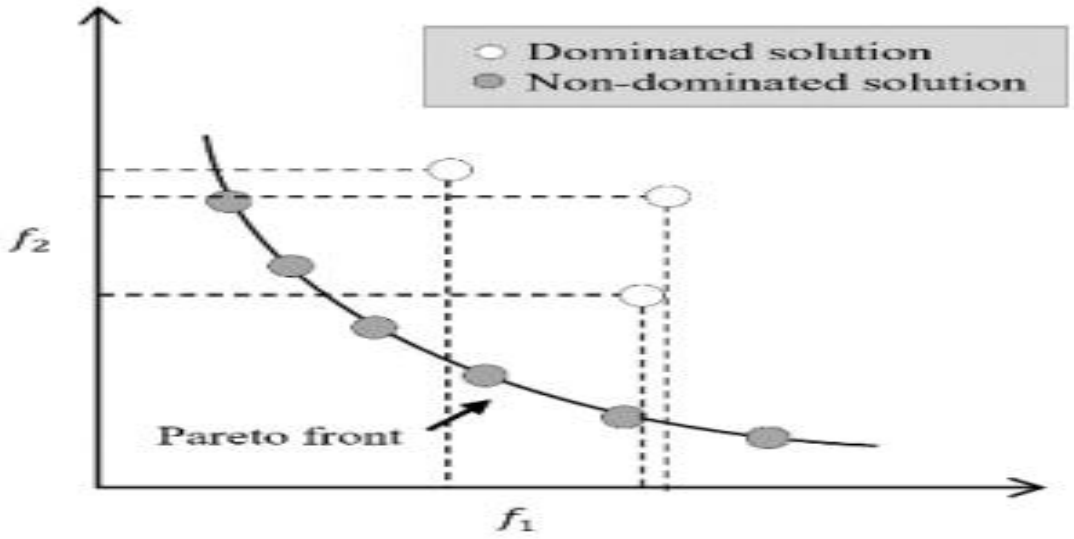


Figure 25 : Représentation des solutions selon la dominance.

7. L'optimisation multi-objectif et les algorithmes évolutionnaires :

De nombreux algorithmes évolutionnaires multi-objectifs ***(Multi-Objective Evolutionary Algorithms, MOEAs)*** utilisent la notion de dominance de Pareto pour classer les solutions et définir les stratégies de sélection (reproduction ou survie). En général, il fait partie des algorithmes élitistes. Ils conservent les meilleures solutions dans la population durant les générations ou les sauvegardent dans une archive.

Dans le premier cas, ces solutions participent au cycle de reproduction et peuvent donc influencer le cheminement de l'espace de recherche. En revanche, le nombre de solutions non

dominées peut très vite croître pour les problèmes multi-objectifs et en contre partie réduire ainsi le nombre de places possibles (dans la population). Pour surmonter ce problème, un opérateur de préservation de la diversité est souvent utilisé. ***NSGA-II*** (Elitiste Non-dominated Sorting Genetic Algorithm) est l'un des algorithmes les plus connus de cette catégorie.

Dans le cas de l'approche utilisant une archive de solutions non dominées, ces dernières ne participent pas nécessairement à la reproduction. La conception de la dominance est généralement, dans ce cas, combinée à d'autres indicateurs pour définir des valeurs de fitness. ***(Kalyanmoy Deb, VOL. 6, NO. 2, APRIL 2002)***

8. Le NSGA-II :

Historique :

L'Algorithme génétique de tri non dominé (NSGA-II) est un performant mécanisme d'exploration de l'espace de décision basé sur Algorithme génétique (AG) pour résoudre les problèmes d'optimisation multi-objectifs (MOOPs). Il a été initialement proposé par [Deb et al] en l'an 2000 dans "International Conference" sur la résolution de problèmes parallèles à partir de la nature.

En 2002, il a été publié en tant qu'article de recherche complet dans le journal IEEE Transactions on "Evolutionary Computation".

Et depuis lors, il a été cité plus de 20630 fois selon IEEE Xplore. Sans doute, NSGA-II est le 4ème article de revue le plus cité dans la base de données de l'IEEE Xplore. En outre, selon Google scholar, il a été cité plus de 35240 fois, dont plus de 19600 citations provenant du web.

Ces informations suffisent à démontrer la popularité de la NSGA-II pour le traitement des données.
Au cours de ses 20 ans d'existence, NSGA-II a été mis en œuvre sur un large éventail de ***(MOOPs)*** ayant variables continues et discrètes. ***(Kalyanmoy Deb, APRIL 2002)***

9. La structure basique de NSGA-II :

La philosophie de NSGA-II repose sur quatre grands principes, qui sont : Le tri non-dominé, l'opérateur de préservation de l'élite, la distance de surpeuplement et l'opérateur de sélection. Ces principes sont décrits dans les sous-sections suivantes.

9.1. Tri non-dominé :

Dans cette procédure, les membres de la population sont triés en utilisant le concept de dominance de Pareto. Le processus de tri non-dominant commence par la détermination du premier ensemble des solutions non dominées. Ces individus de premier rang sont ensuite placés en première ligne et retirés de la population initiale.

Après cela, la procédure de tri non-dominant est exécutée sur le reste des solutions de la population. ***(N. Siinivas, 1995)***

En poursuivant, les solutions non- dominés de la population restante se voient attribuer le deuxième rang et placés sur le deuxième front. Ce processus se poursuit jusqu'à ce que tous les solutions de la population soient placés sur différents fronts en fonction de leur classement, comme le montre la figure -26- .

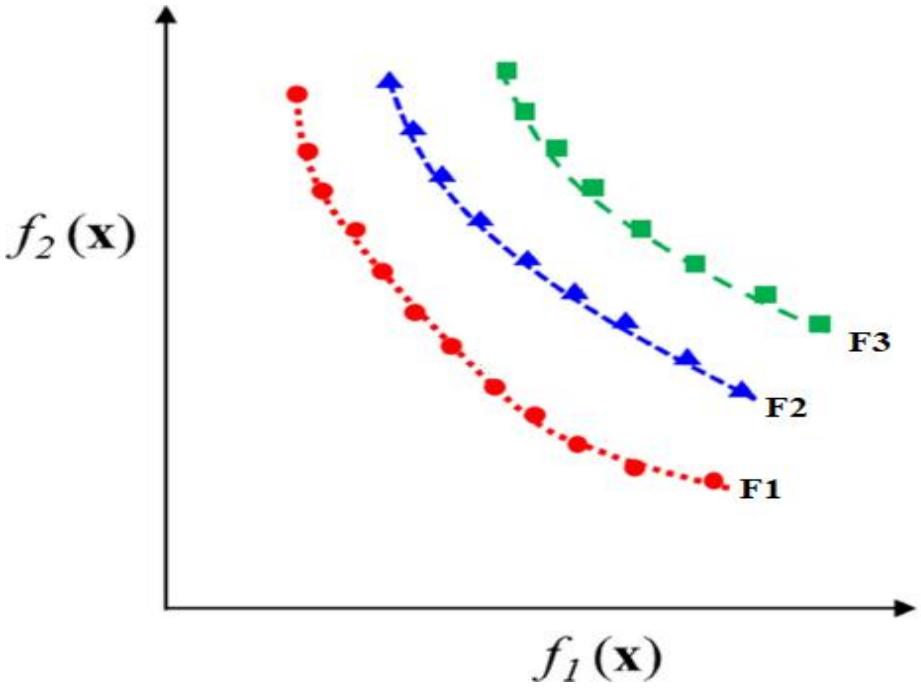


Figure 26 : Classification des individus selon le rang de Pareto.

Le pseudo-code : le tri non-dominées.

Pour $\mathbf{p} \in \mathbf{Pop}$ // pour chaque individu dans la population actuelle //

$\boldsymbol{S}_{\boldsymbol{p}} = \varnothing$ // L'ensemble des solutions dominées par **p** //

$\boldsymbol{n_p} = \mathbf{0}$ // L'indices des solutions dominées par **p** //

// Création de 1er Pareto //

Pour $\mathbf{q} \in \mathbf{Pop}$ // tester la dominance de p par apport les autres individus //

Si (**p** < **q**) donc // si **p** domine **q** //

$\boldsymbol{S}_{\boldsymbol{p}} = \boldsymbol{S}_{\boldsymbol{p}} \boldsymbol{U} \{\boldsymbol{q}\}$ //ajouter q dans L'ensemble des solutions dominées par **p** //

Sinon

$\boldsymbol{n_p} = \boldsymbol{n_p} + \mathbf{1}$ // incrémenté conteur de l'index des solutions dominées par **p** //

Si $\boldsymbol{n_p} = \mathbf{0}$ alors

Classe **p** = 1

$\boldsymbol{F}_{\mathbf{1}} = \boldsymbol{F}_{\mathbf{1}} \boldsymbol{U} \{\boldsymbol{p}\}$ // p appartient au premier Front //

$\boldsymbol{i} = \mathbf{1}$ // Initialiser conteur des Front //

Tant que $\boldsymbol{F}_{\boldsymbol{i}} \neq \varnothing$

Q = ∅ //stock des individus du prochain Front //

Pour $\mathbf{p} \in \boldsymbol{F}_{\boldsymbol{i}}$

Pour $\mathbf{q} \in \boldsymbol{S}_{\boldsymbol{p}}$

$\boldsymbol{n_q} = \boldsymbol{n_q} - \mathbf{1}$

Si $\boldsymbol{n_q} = \mathbf{0}$ alors

Classe **q = i +1** // q appartient au prochain Front //

$\boldsymbol{Q} = \boldsymbol{Q}\ \boldsymbol{U} \{\boldsymbol{p}\}$

i = i + 1

$\boldsymbol{F}_{\boldsymbol{i}} = \boldsymbol{Q}$

9.2. Opérateur de présentation de l'élite :

NSGA-II utilise, en plus d'une stratégie élitiste, un mécanisme de préservation de la diversité. A chaque génération, la population des parents et la population des enfants sont fusionnées et classées en plusieurs fronts de Pareto.

La population de la génération suivante est constituée en choisissant des solutions parmi ces fronts de Pareto en commençant par le premier. Lorsque la taille du front à utiliser est supérieure au nombre de places restantes à pourvoir dans la population future, les solutions sont choisies en fonction de leur valeur de distance de surpeuplement **"crowding distance "**. ***(SHANU VERMA, 2021)***

9.3. La distance de surpeuplement "crowding distance " :

Une fois que le tri des non-dominés est terminé, la distance crowding est attribuée. NSGA-II utilise outre une stratégie élitiste, un mécanisme de préservation de la diversité. A chaque génération, les populations des parents et celle des enfants sont fusionnés et classées en plusieurs fronts de Pareto.

La population de la génération suivante est constituée en choisissant des solutions dans ces fronts de Pareto en commençant par le premier. Lorsque la taille du front à utiliser est supérieure au nombre de places restantes à pourvoir dans la future population, les solutions sont choisies selon leur valeur de crowding distance. Il s'agit d'un indicateur qui calcule la distance moyenne, sur l'ensemble des objectifs, entre une solution donnée et ses voisins directs dans l'espace des résultats (l'espace des objectifs). Alors, cette technique s'applique sur le dernier front pour compléter la taille de la population parente pour la génération suivante. Au lieu d'exclure arbitrairement certains membres du dernier front, les points qui réaliseront la diversité la plus élevée des points sélectionnés seront choisis.
(SESHADRI, 2005)

La distance d'encombrement de la i solution est la longueur moyenne des côtés du cuboïde, comme le montre la figure -28-. Si $\boldsymbol{fij}$ est la jème valeur d'une fonction objective pour le ième individu et, $\boldsymbol{fmax}$ **j** et $\boldsymbol{fminj}$ sont respectivement les valeurs maximale et minimale de la jème fonction objective parmi tous les individus de ce front. Ensuite, la distance d'encombrement de l'ième individu est définie comme la distance moyenne de deux solutions les plus proches de part et d'autre, La distance de crowding est calculée comme suit:

$$\boldsymbol{cd(i)} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{k}} \frac{\boldsymbol{f_j^{i+1} - f_j^{i-1}}}{\boldsymbol{f_j^{max} - f_j^{min}}}$$

Où k est le nombre de fonctions objectifs.

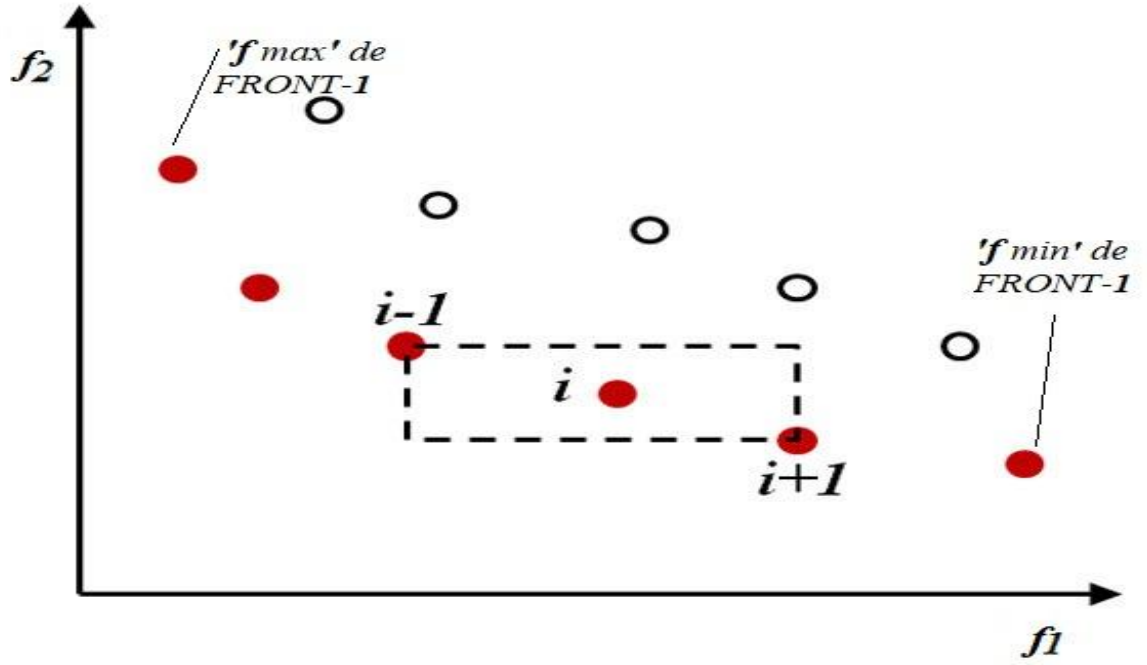


Figure 27 : le calcul de 'crowding-distance'.

Le pseudo-code : crowding distance.

$\boldsymbol{l} = |\boldsymbol{F}|$ *// nombre des solutions dans le dernier Front //*
Pour tout ***i,*** l'ensemble $\boldsymbol{F}[\boldsymbol{i}]_{\boldsymbol{distance}} = \boldsymbol{0}$ *// initialiser la distance //*
Pour chaque objectif ***m***
F = sort (***F, m)*** //sort est utilisé pour chaque valeur d'une objectif //
$\boldsymbol{F}[\boldsymbol{1}]_{\boldsymbol{distance}} = \boldsymbol{F}[\,\boldsymbol{l}\,]_{\boldsymbol{distance}} = \infty$ // Pour que les points limites sont toujours sélectionnés //
Pour i = 2 à ($\boldsymbol{l} - \boldsymbol{1}$)
$\boldsymbol{F}[\boldsymbol{i}]_{\boldsymbol{distance}} = \boldsymbol{F}[\,\boldsymbol{i}]_{\boldsymbol{distance}} + (\,\boldsymbol{F}[\boldsymbol{i}+\boldsymbol{1}]_{\boldsymbol{m}} = \boldsymbol{F}[\,\boldsymbol{i}-\boldsymbol{1}\,]_{\boldsymbol{m}}) / (\,\boldsymbol{F}[\boldsymbol{1}]_{\boldsymbol{m}} - \boldsymbol{F}[\,\boldsymbol{l}\,]_{\boldsymbol{m}})$

Fin pour
Fin pour

9.4. OPÉRATEUR DE SÉLECTION :

La population de la génération suivante est sélectionnée à l'aide d'un opérateur de sélection de type "crowded tournament", qui utilise le rang de Pareto des membres de la population et leurs distances de surpeuplement (crowding distance) pour la sélection.

Les règles pour sélectionner quel membre choisir pour la prochaine génération sont les suivantes :

(i) Si les deux membres de la population sont de rangs différents, alors celui qui a le meilleur rang est sélectionné pour la prochaine génération.

(ii) Si les deux membres de la population ont le même rang, automatiquement, celui qui a la distance de surpeuplement la plus élevée est sélectionné pour la génération suivante.

(iii) Si les deux membres de la population ont le même rang et la même distance crowding, ce qui est un cas très rare, nous choisissons un membre au hasard parmi eux pour être sélectionné pour la génération suivante.

(SHANU VERMA, 2021)

10. Les opérateurs génétiques de NSGA-II :

NSGA-II met en œuvre les opérateurs génétiques en binaire (ses concepts sont déjà expliqué) pour établir le pool de reproduction (c'est-à-dire croisement et mutation). Et pour les opérateurs génétiques codés en réel, le NSGA-II utilise le croisement binaire simulé (SBX) opérateur pour croisement et mutation polynomiale (PM) ***(Qi Wang, May 2019)***:

$$x_i^{(1,t+1)} = \frac{1}{2}\left[\left(x_i^{(1,t)} + x_i^{(2,t)}\right) - \beta_i\left(x_i^{(2,t)} - x_i^{(1,t)}\right)\right]$$
$$x_i^{(2,t+1)} = \frac{1}{2}\left[\left(x_i^{(1,t)} + x_i^{(2,t)}\right) + \beta_i\left(x_i^{(2,t)} - x_i^{(1,t)}\right)\right]$$

Où $x_i^{(i,t+1)}$ est l'enfant i avec t composants, $x_i^{(i,t)}$ est le parent sélectionné et β_i (≥ 0) est un échantillon d'un nombre aléatoire généré ayant la densité :

$$p(\beta) = \frac{1}{2}(\eta_c + 1)\beta^{\eta_c}, \quad if\ 0 \leq \beta \leq 1$$
$$p(\beta) = \frac{1}{2}(\eta_c + 1)\frac{1}{\beta^{\eta_{c+2}}}, \quad if\ \beta \geq 1$$

Calculer β_i en assimilant l'aire sous la courbe de probabilité égale à u (un nombre aléatoire) Cette distribution peut être obtenue à partir d'un nombre aléatoire u uniformément échantillonné entre $u \in [0, 1]$. η_c est l'indice de distribution pour le croisement.

$$\begin{cases} \beta(u) = (2u_i)^{\frac{1}{(\eta_c+1)}} & Si \quad u_i \leq 0.5 \\ \beta(u) = \dfrac{1}{[2(1-u_i)]^{\frac{1}{(\eta_c+1)}}} & Sinon \end{cases}$$

Et la mutation polynomiale (PM) :

$$y_i^{(1,t+1)} = x_i^{(1,t+1)} + \delta_i \left(x_i^{(max)} - x_i^{(min)} \right)$$

Où $y_i^{(1,t+1)}$ est l'enfant après le croisement, $x_i^{(1,t+1)}$ est le parent avec $x_i^{(max)}$ est la limite maximale sur le composante parent et $x_i^{(min)}$ est la limite minimale. δ_i est une petite variation qui est calculée à partir d'une distribution polynomiale en utilisant :

$$\begin{cases} \delta_i = (2r_i)^{\frac{1}{\eta_m+1}} - 1 & , if \ r_i < 0.5 \\ \delta_i = 1 - [2(1-r_i)]^{\frac{1}{\eta_m+1}} & , if \ r_i \geq 0.5 \end{cases}$$

r_i est un nombre aléatoire uniformément échantillonné entre $[0, 1]$ et η_m est l'indice de distribution de mutation.

11. La stratégie du NSGA-II :

La stratégie d'élitisme implicite garantit que les meilleures solutions jamais trouvées dans l'historique de recherche sont toujours conservées dans la population.
Cela permet à une nouvelle population d'être dérivée de la combinaison des parents et de leur progéniture via l'approche de tri rapide non dominé.

NSGA-II propose également une technique de gestion des contraintes pour traiter efficacement les problèmes contraints et prend en charge les représentations de codage binaire et réel.

12. Le fonctionnement de NSGA-II :

NSGA-II est conçu pour trouver un ensemble de solutions optimales, appelées solutions non dominées, ou ensemble de Pareto. Une solution non dominée est une solution qui fournit un compromis approprié entre tous les objectifs sans qu'aucun ne soit dégradé, comme décrit dans l'algorithme si dessous.

1) La procédure commence par générer une population initiale et aléatoire $\boldsymbol{P_t}$ de taille $\boldsymbol{N}$.

2) Ensuite, une nouvelle population $\boldsymbol{Q_t}$ est créée après avoir effectué des opérations de croisement et de mutation sur la population $\boldsymbol{P_t}$.

3) Après cela, les populations $\boldsymbol{P_t}$ et $\boldsymbol{Q_t}$ sont combinées pour former une nouvelle population $\boldsymbol{R_t}$ de taille 2*N*.

4) la procédure de tri non dominé est effectuée sur Rt. Ensuite, les membres de la population membres de la population $\boldsymbol{R_t}$ sont classés en différents fronts selon leur niveau de non-domination.

Le processus suivant consiste à sélectionner $\boldsymbol{N}$ membres de $\boldsymbol{R_t}$ pour créer la population suivante $\boldsymbol{P_{t+1}}$.

5) la nouvelle population $\boldsymbol{P_{t+1}}$ est remplie par des solutions de différents fronts non-dominées, une à la fois. Le remplissage commence par le premier front non-dominé (de classe 1) et se poursuit avec les points de second front non-dominé, et ainsi de suite.

Dans le cas où l'on approche du remplissage du $\boldsymbol{P_{t+1}}$ avec les solutions de fronts, et que la taille du dernier front dépasse le nombre d'emplacements disponibles.
Ici, nous recourons à la distance d'encombrement (***crowding distance***) pour ne choisir que les solutions avec la distance d'encombrement la plus élevée, à partir de ce front pour les transférer à $\boldsymbol{P_{t+1}}$.

6) Si la taille de $\boldsymbol{P_{t+1}}$ est toujours inférieure à $\boldsymbol{N}$, alors la même procédure est suivie pour les fronts consécutifs suivants, jusqu'à ce que la taille de $\boldsymbol{P_{t+1}}$ devienne égale à *N*.

7) Une fois la taille de $\boldsymbol{P_{t+1}}$ est rempli, tous les fronts qui ne peuvent pas être inclus sont alors supprimés.

Les populations $\boldsymbol{P_{t+2}}$, $\boldsymbol{P_{t+3}}$, $\boldsymbol{P_{t+4}}$, . . . pour les générations suivantes sont construites en utilisant la même procédure jusqu'à ce que les critères d'arrêt ne soient pas satisfaits.

Le fonctionnement de NSGA-II est illustré à la Figure -28-.

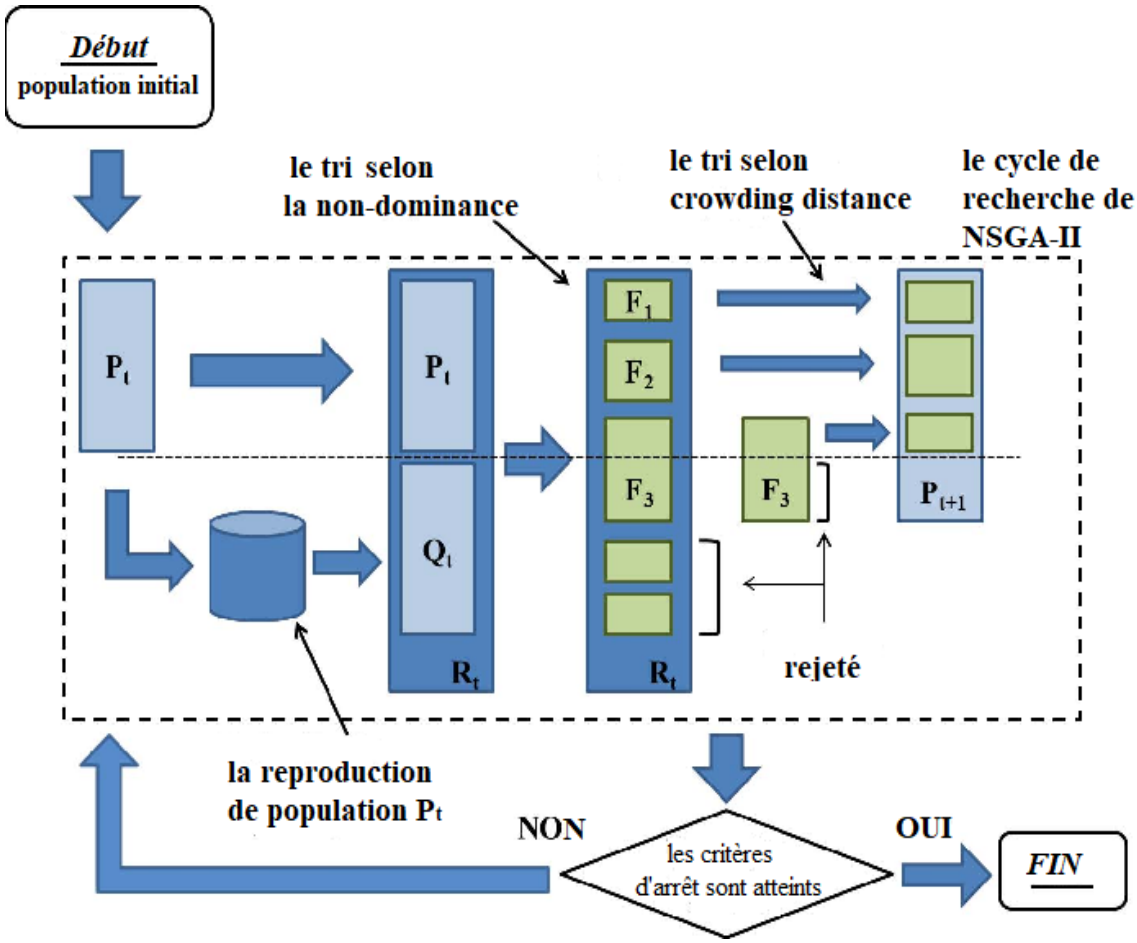


Figure 28: Le fonctionnement de l'algorithme NSGA-II.

Le pseudo-code : NSGA-II.
1. Initialise = N-gen, N-pop, Pc, Pm ;
2. initialiser $\boldsymbol{P_0}$ de taille N-pop ;
3. Tant que **i** < N-gen faire
4. Reproduction de $\boldsymbol{P_0}$ = Q ; // les mêmes opérateurs de reproduction que l'algorithme génétique //
5. R = P U Q //combiner les deux populations //
6. Évaluation de tous les individus par fitness ;
7. Trier les solutions par « ***non domination*** » et « ***crowding distance*** » ;
8. sélectionner le meilleur **N-gen-ième** par rapport « ***non domination*** » et « ***crowding distance*** »
9. fin Tant que ;
10. Extraire le meilleur Pareto front ;

Partie : résultats et discussion

1. *Méthodologie de recherche :*

A fin de constater l'efficacité de NSGA2 dans l'extraire de l'ensemble des Paretos, on a utilisé des fonctions tests, pour analyser la capacité de NSGA2 à atteindre l'ensemble de Pareto.

Les problèmes de test sont choisis parmi plusieurs études antérieures importantes dans ce domaine. Veldhuizen a cité plusieurs problèmes de test qui ont été utilisés dans le passé.

Parmi eux, nous choisissons quatre problèmes : une étude de Schaffer (SCH), et l'étude de Kursawe (KUR). En 1999, le premier auteur a proposé une manière systématique de développer des problèmes de test pour l'optimisation multi-objectif. Zitzler et al ont suivi ces directives et ont proposé six problèmes de test. Nous choisissons une de ces six problèmes et l'appelons ZDT1, nous choisissons aussi le problème shaffer(SCH), problème Kursawe(KUR) et le problème polonie (POL).

2. *Formulation mathématique :*

Problème	*n*	*L'intervalle des variables*	*L'optimum*	*Formulation Mathématique*
SCH	*1*	$[-\mathbf{10^3}, \mathbf{10^3}]$	*x ∈[0, 2]*	$\begin{cases} f_1(x) = x^2 \\ f_2(x) = (x-2)^2 \end{cases}$
KUR	*3*	$[-\mathbf{5}, \mathbf{5}]$	*Se référer à [1]*	$\begin{cases} f_1(x) = \sum_{i=1}^{n-1} (-10\, exp(-0.2\sqrt{x_i^2 + x_{i+1}^2})) \\ f_2(x) = \sum_{i=1}^{n} (\lvert x_i \rvert^{0.8} + 5 \sin x_i^3) \end{cases}$
ZDT1	*30*	$[\mathbf{0}, \mathbf{1}]$	x_1 *∈[0, 1] et* x_i *= 0, i=2,..,n*	$\begin{cases} f_1(x) = x_1 \\ f_2(x) = g(x)[1 - \sqrt{x_1 / g(x)}] \\ g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1) \end{cases}$
POL	*2*	$[-\pi, \pi]$	*Se référer à [1]*	$\begin{cases} f_1(x) = [1 + (A_1 - B_1)^2 + (A_2 - B_2)^2] \\ f_2(x) = [(x_1+3)^2 + (x_2+1)^2] \\ A_1 = 0.5\sin 1 - 2\cos 1 + \sin 2 - 1.5\cos 2 \\ A_2 = 1.5\sin 1 - \cos 1 + 2\sin 2 - 0.5\cos 2 \\ B_1 = 0.5\sin x_1 - 2\cos x_1 + \sin x_2 - 1.5\cos x_2 \\ B_2 = 1.5\sin x_1 - \cos x_1 + 2\sin x_2 - 0.5\cos x_2 \end{cases}$

Tableau 2-Formulation mathématique des fonctions teste.

Toutes les fonctions ci-dessus sont à minimiser.

On a implémenté les fonctions testées dans l'algorithme NSGA-II ; avec les paramètres suivants :

Le nombre de génération : 100.

La taille de population : 100.

La probabilité de croisement : 0.9

La probabilité de mutation : 0.5

Xmin, Xmax, les nombres des fonctions objectives et leurs formulations mathématiques, ils sont mentionnés dans le tableau -2-

3. *Résultats :*

Comparaison illustrative entre le vrai ensemble de Pareto de fonction test et celle de le NSGA-II.

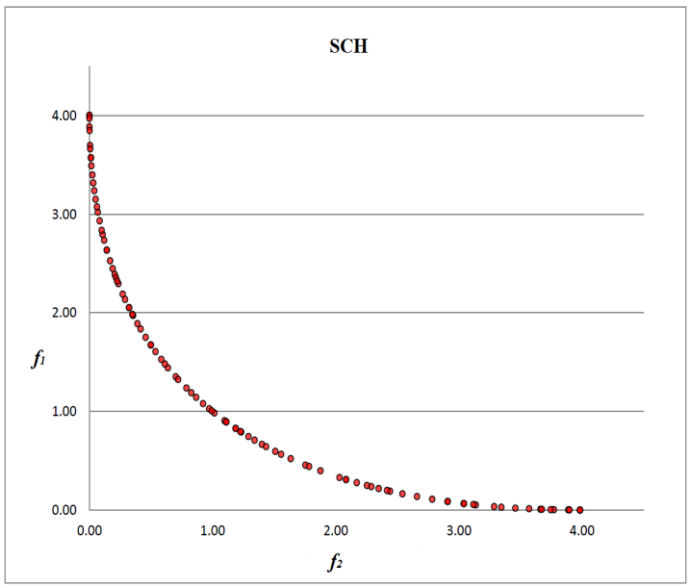


Figure 29:le vrai front-Pareto de fonction shaffer

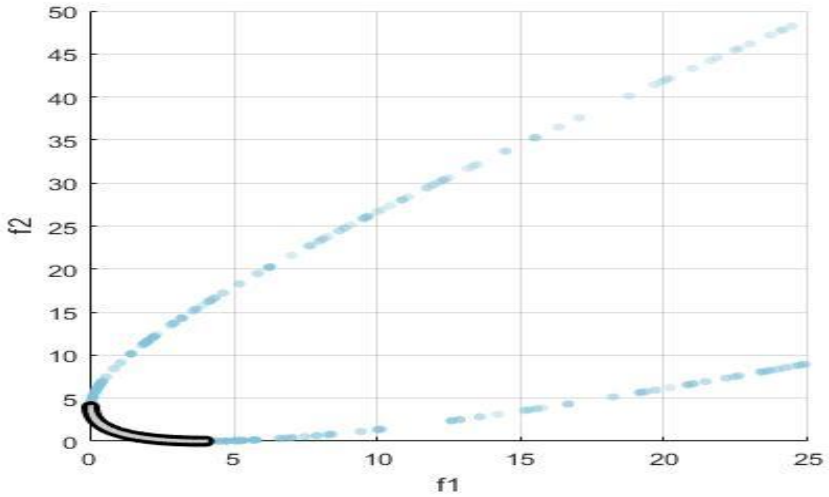


Figure 30: le front de Pareto SCH par NSGA2

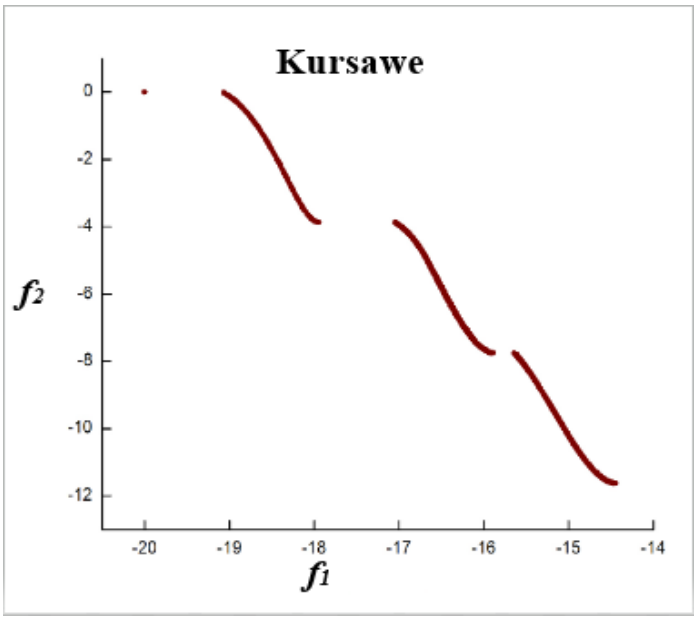


Figure 31: le vrai front-Pareto de fonction Kursawe

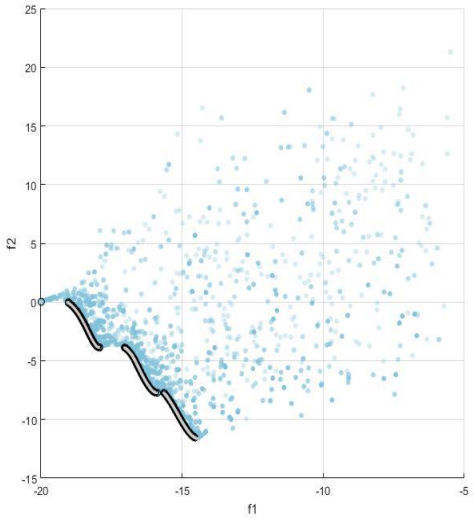


Figure 32: le front de Pareto Kursawe par NSGA2

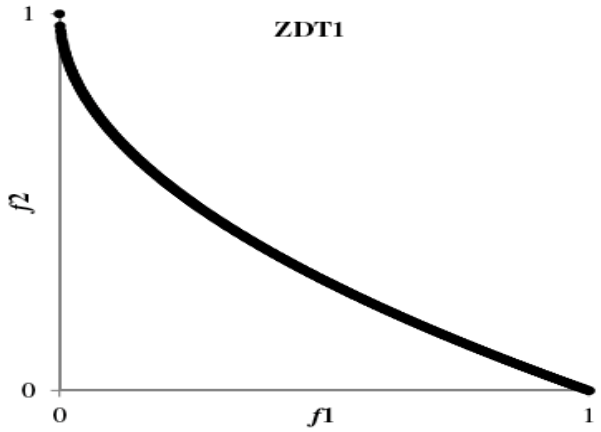


Figure 33: le vrai front-Pareto de fonction ZDT1

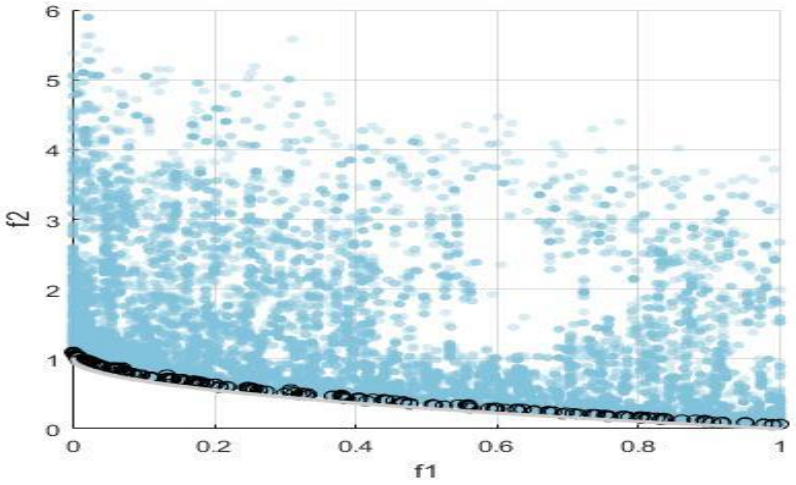


Figure 34: le front- Pareto de ZDT1 par le NSGA2

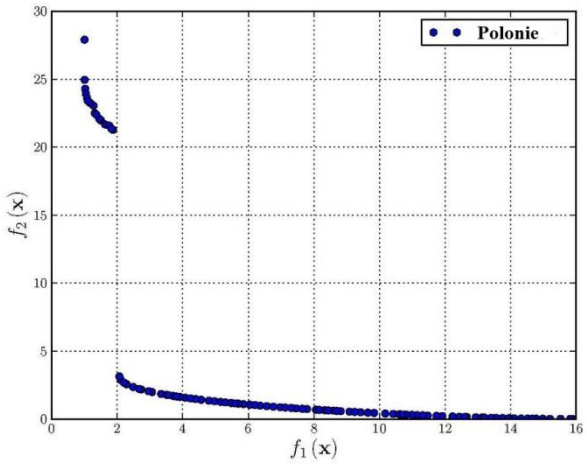


Figure 35: le vrai front-Pareto de fonction Polonie

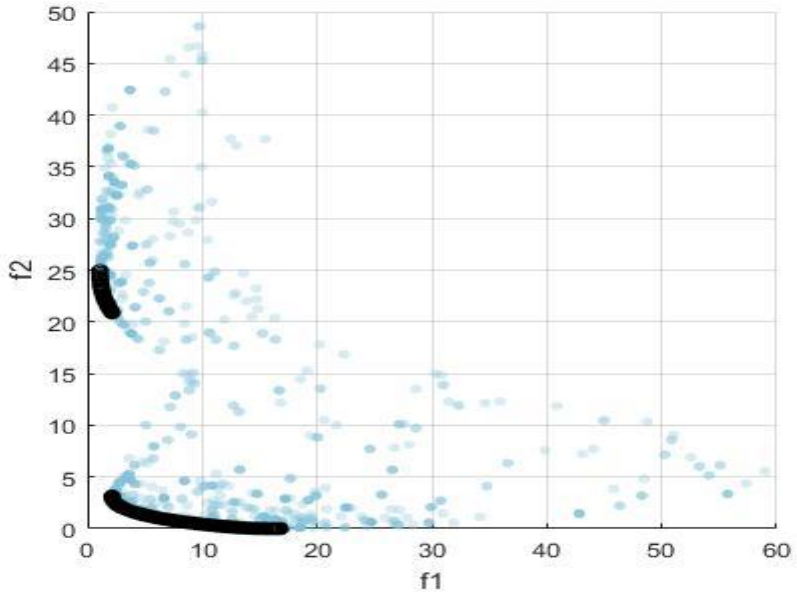


Figure 36: : le front- Pareto de Polonie par le NSGA2

Observation :

Nous avons commencé par une population de taille 100 individus et 100 générations et le processus s'est terminé par l'obtention des figures ci-dessus, on voit que comparant les deux ensembles de Pareto selon "son intervalle et front de Pareto " (voir la courbe avec les point noirs) que le NSGA-II arrive à avoir l'ensemble de Pareto pour chaque une de ces fonctions tests ce qui implique son efficacité à résoudre les problèmes multi-objectifs.

Conclusion générale

Notre travail s'est intéressé à l'étude de l'un des outils les plus utilisé dans ce sens, une métaheuristique puissante, l'algorithme NSGA-II (non dominated sorting Genetric Algorithms). Cet algorithme est d'ailleurs très reconnu pour son succès dans la résolution et l'analyse des compromis de plusieurs problèmes multi-objectifs dans le monde de l'ingénierie. L'objectif principal poursuivi dans le présent travail consiste à prouver l'efficacité d'extraction de l'ensemble Pareto par l'utilisation de l'algorithme NSGA-II.

En raison de sa garantie d'élitisme, nous avons choisi d'étudier NSGA2 parmi tous les algorithmes MOP. Nous avons utilisé les fonctions test de l'efficacité de Pareto, parce que son front de Pareto est connu (le résultat est déjà démontré), qui nous aide par conséquence à voir l'efficacité de NSGA-II.

Le présent travail est loin d'être achevé. Ainsi nous proposons aux futures étudiantes en mastère de RO de poursuivre ce travail en enrichissant les axes suivants :

- Faire une étude comparative profonde entre les algorithmes multi objectifs.
-Trouver une relation entre le problème multi-objectif posé et les critères d'arrêt de l'algorithme utilisé
- Après avoir trouvé une population de solutions « Pareto», nous proposons au décideur d'ajouter leur propre critère pour le choix d'une seule solution.

Bibliographies

[1] A comprehensive review on NSGA-II for multi-objective combinatorial optimization problems / auth. shanu verma vacla snasel. - [s.l.] : IEEE access, 2021.

[2] A fast and elitist multiobjective genetic algorithm : NSGA-II / auth. kalyanmoy deb amrit pratap, sameer agarwal, and t. meyarivan. - [s.l.] : IEEE transacion on evolutionarycomputation,vol.6, april 2002.

[3] A fast elitist multiobjective genetic algorithm / auth. seshadri aravind. - 2005. - p. 4.

[4] A simulated annealing based multi-objective optimization algorithm: amosa / auth. sanghamitra bandyopadhyay sriparna saha, ujjwal maulik,nd kalyanmoy deb.

[5] Algorithme génétique multi-objectifs adaptatif / auth. wahabou abdou christelle bloch, damien charlet, fran¸cois spies. - [s.l.] : hal open science, 2015.

[6] An improved NSGA-II algorithm based on crowding distance elimination strategy / auth. junhui liu xindu chen.

[7] Benchmark et automatisation du tuning des algorithmes / auth. rifaieh nabila benharkat rami. - 2015.

[8] Contribution à la synthèse et l'optimisation multi-objectif par essaims particulaires de lois de commande robuste rst de systèmes dynamiques / auth. madiouni riadh. - université paris-est : hal open science.

[9] Cours d'optimisation sans contrainres / auth. mohammed dr belloufi. - souk-ahras : universitè mohamed chèrif messaadia, octobre2015.

[10] Cours sur optimisation multiobjectifs / auth. zoubir ramdani. - bordj-bouararidj : université mohamed al-bachir al-ibrahimi, 2020/2021.

[11] Effects of removing overlapping solutions on the performance of the NSGA-II algorithm / auth. yusuke nojima kaname narukawa, shiori kaige, and hisao ishibuchi. - [s.l.] : department of industrial engineering, osaka prefecture university., 2005.

[12] Elitist non-dominated sorting GA-II (NSGA-II) as a parameter-less multi-objective genetic algorithm / auth. tran khoa duc. - [s.l.] : IEEE articals., 2005.

[13] État de l'art des méthodes d'optimisation globale [article] / auth. siarry gérard berthia et patrick. - france : rairo operations research, 2002. - université de nantes.

[14] Genetic algorithm / auth. mathew tom v. - [s.l.] : indian institute of technology bombay., 2006.

[15] Introduction à l'analyse prescriptive / auth. huguet m.-j.. - toulouse : [s.n.], 2020-2021.

[16] Mathématiques pour l'optimisation [book] / auth. molle i. sau et c.. - [s.l.] : lp sil, 2002.

[17] Métaheuristique [book] / auth. siary patrick. - [s.l.] : eyrolles, 06/03/2014.

[18] Métaheuristique hybride pour l'optimisation multi-objectif / auth. cheriet abdelhakim. - [s.l.] : université mohamed khider biskra, 2016.

[19] Métaheuristiques adaptatives d'optimisation continue basées sur des méthodes d'apprentissage / auth. ghoumari asmaa.

[20] Méthode de recuit simulé pour l'optimisation de l'affectation d'opérateurs sur une ligne de production / auth. sana bouajaja najoua dridi. - [s.l.] : university of tunis elmanar, 2017.

[21] Méthodes et outils d'optimisation / auth. kleiner mathias. - paris : art et métier.paris tech, mai 2013.

[22] Mise en oeuvre de l'algorithme génétique de type NSGAII pour l'optimisation multi-objectif d'une opération de fraisage en bout / auth. idir belaidi kamal mohammedi, belaid brachemi. - [s.l.] : hal open science, 2021.

[23] Multi-objective optimization using non dominated sorting in genetic algorithms / auth. n. siinivas kalyanmoy deb. - [s.l.] : the massachusetts institute of technology, 1995.

[24] Multi-objective 3-dimensional dv-hop localization algorithm with NSGA-II [report] / auth. xingjuan cai penghong wang, lei du, zhihua cui *, wensheng zhang and jinjun chen, senior. - 2019.

[25] Optimisation combinatoire multi-objectif : apport des méthodes coopératives et contribution à l'éxtraction de connaissances. [book section] / auth. dhaenens-flipo clarisse // diriger des recherches de l'u.s.t.l.. - [s.l.] : universit´e des sciences et technologies de lille, 2005.

[26] Optimisation évolutionnaire multi-objectif parallèle : application à la combustion diesel / auth. yagoubi mouadh // hal open science. - [s.l.] : université paris sud .

[27] Optimisation multi-objectif : études de cas / auth. wafa aouadj. - 2021.

[28] Optimisation multiobjectif et son application aux problèmes bioinformatique / auth. samir mahdi // thèse de doctorat en science. - 2021.

[29] Optimisation multi-objectif pour la sélection de modèles / auth. clément chatelain yannick oufella, sébastien adam, yves lecourtier,laurent heutte.. - france : hal open science.

[30] Optimisation multi-objectifs à base de métamodèle pour les procédés de mise en forme / auth. ejday mohsen. - [s.l.] : paris tech, mars 2011.

[31] Optimisation multi-objectifs de structure sous critère decontrainte à l'aide de modèles éléments finis mixtes / auth. pierre garambois sébastien besset, louis jézéquel. - [s.l.] : hal open science, mars 2017.

[32] Optimisation multi-objective des problèmes combinatoires:application à la génération des horaires d'examens finaux / auth. côté pascal. - montréal : école de technologie supérieure université du québec.

[33] Parameterization of NSGA-II for the optimal design of water distribution systems / auth. qi wang libing wang, wen huang, zhihong wang, shuming liu, and dragan a. savic.. - [s.l.] : school of civil and transportation engineering, guangdong university of technology., may 2019.

[34] Représentation de solution en optimisation continue, multi-objectif et applications / auth. zidani m.hafid. - [s.l.] : l'école mohammadia d'ingenieurs, 23 octobre 2013.

[35] Support de cours d'optimisation combinatoire focus sur les méthodes de résolution approchée / auth. allaoua hemmak. - [s.l.] : université mohamed boudiaf de m'sila, 2017.

[36] Un modele base sur les algorithmes genetiques et le front de pareto pour l'optimisation multi-objectif / auth. idir belaidi brahim merdjaoui , kamal mohammedi. - [s.l.] : 10ème colloque national aip primeca , 2007.

[37] Un principe d'invariance pour un algorithme génétique en population finie / auth. jean bérard alexis bienvenüe. - [s.l.] : université claude-bernard, 26 juin 2000.