Algorithmes Evolutionnaires: Prise en Compte des Contraintes et Application Réelle

Sana Ben Hamida

19 février 2001

Table des matières

Chapitre 1

Introduction

Ce chapitre introduit les algorithmes évolutionnaires (*Evolutionary Algorithms:* **EA**) et présente les différentes notations et abréviations utilisées par la suite. La première section est une introduction générale aux EA, illustrant leurs origines, leur principe et leur historique. Les trois sections suivantes introduisent les trois représentations les plus utilisées dans les EA. La représentation binaire n'est pas utilisée dans la thèse, mais elle est présentée vue son importance dans l'historique des EA. Par contre, la plus grande partie de notre travail se base sur la représentation réelle. Elle est alors présentée en détails dans la section 1.3. La section 1.4 introduit la représentation fonctionnelle, utilisée dans la deuxième partie de la thèse. La suite du chapitre est consacrée au darwinisme artificiel et aux techniques d'amélioration.

1.1 Les Algorithmes Evolutionnaires

Les EA sont des algorithmes d'optimisation stochastique inspirés du paradigme de l'évolution darwinienne des populations. Introduits par Fogel en 1965 [36], Rechenberg en 1973 [97], Holland en 1975 [57], et popularisé par Goldberg en 1989 [42], ils font évoluer une population d'individus, où seulement les mieux adaptés à un environnement donné peuvent survivre et se reproduire. En termes mathématiques, l'environnement est la fonction de performance à optimiser, et elle constitue la seule information nécessaire aux EA, ce qui leur permet de traiter une grande variété de problèmes numériques, impossibles à traiter avec les méthodes d'optimisation classiques.

Depuis les années soixante, plusieurs tendances d'algorithmes évolutionnaires sont apparues, qui se classent en 4 catégories:

- les algorithmes génétiques (*Genetic Algorithms*: GA)[56]
- les stratégies d'évolution (*Evolution Strategies*: ES) [97]
- la programmation évolutionnaire (*Evolutionnary Programming*:EP) [36, 35]
- la programmation génétique (*Genetic Programming*: GP) [69]

Chacun de ces algorithmes manipule une population dont la taille, contrairement à une population naturelle, reste inchangée au long de l'évolution. Ils ont un principe de base commun, qui caricature le principe de l'évolution biologique établie par Darwin. En effet, seulement les plus aptes survivent à la sélection naturelle et peuvent se reproduire. L'itération de ce principe pendant plusieurs générations devrait faire apparaître dans la population les individus les plus adaptés à leur environnement, c'est à dire les optimaux pour la fonction de performance.

1.1.1 Principe Général

Le but principal des algorithmes évolutionnaires est de chercher, dans un espace d'état Ω, un élément de cet espace ayant la meilleure adaptation à l'environnement posé. Chaque élément de Ω est un **individu**, noté I. Une **population** est donc un ensemble de N éléments (individus) de Ω: $P = (I_1, I_2, \cdots, I_n)$. La mesure du degré d'adaptation de chaque individu constitue la **fonction de performance** F (en anglais **fitness**). L'objectif des EA est de faire évoluer une population P dans le but de trouver l'optimum I^*. Pour se faire, à chaque génération t, les individus de la population P(t) sont mutés et croisés avec des probabilités prédéfinies respectivement p_m et p_c, et ce sont les plus aptes qui survivent pour la génération suivante $t+1$, constituant la nouvelle population $P(t+1)$. Ce processus est répété pendant un certain nombre de **générations**, en espérant que les optima de F apparaissent dans la population.

Les étapes principales communes aux différents algorithmes d'évolution sont décrites dans la figure 1.1.

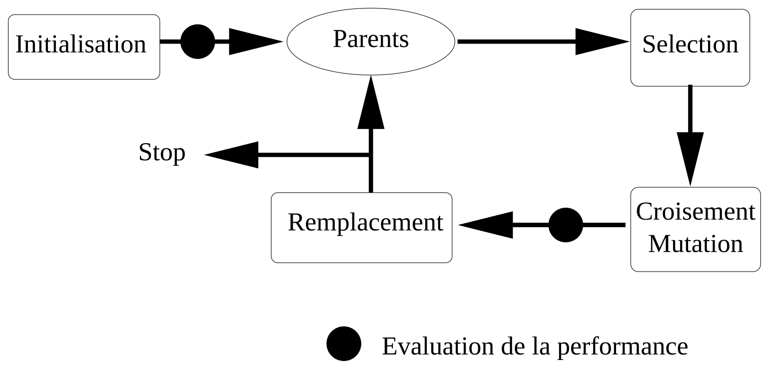


FIG. 1.1 – *Cycle d'un EA*

1. On commence par générer aléatoirement un ensemble d'individus définissant la population initiale $P(0)$ et calculer la performance de chacun.
2. Pour assurer l'évolution, des individus, sélectionnés selon leur performance, sont mutés et croisés entre eux avec des taux prédéfinis, donnant naissance à des enfants dont on évalue la performance.
3. La nouvelle population est constituée en sélectionnant les meilleurs parmi les enfants ou l'ensemble des parents et des enfants (selon le schéma d'évolution choisi).
4. On reboucle ensuite sur les étapes 2 et 3, jusqu'à ce qu'on arrive à la convergence ou à la satisfaction d'un des critères d'arrêt (e.g. nombre maximum de générations autorisé).

1.1.2 Définitions

L'espace de recherche des EA est l'espace Ω, appelé aussi, avec des termes génétiques, l'espace des **génotypes**, mais il peut être différent de l'espace des solutions, appelé l'espace des **phénotypes** sur lequel est défini la fitness. Par exemple, un génotype peut être un vecteur réel, et il représente une structure physique complexe dans l'espace des phénotypes.

La fonction de **codage** qui transforme un phénotype en un génotype doit être injective: à chaque phénotype correspond un seul génotype. La fitness du génotype est celle du phénotype correspondant. L'initialisation de la population $P(0)$ se fait, généralement, d'une façon aléatoire dans l'espace des génotypes. Le choix des opérateurs de croisement et de mutation dépend du codage des individus. Par contre, les opérateurs de sélection et de remplacement sont indépendants de la représentation génotypique, puisqu'ils utilisent uniquement la performance des individus.

Le croisement: C'est un brassage d'information entre les individus de la population. Il consiste à échanger des parties composantes (**gènes**) entre deux (ou plusieurs) solutions potentielles. La forme standard de l'opérateur de croisement est $c : (I_1, I_2) \to (I'_1, I'_2)$, qui croise, avec une certaine probabilité p_c $(0 \leqslant p_c \leqslant 1)$, deux parents I_1 et I_2 pour donner les enfants I'_1 et I'_2. D'autres formes de croisement sont possibles comme celle où un enfant est produit par plusieurs parents en même temps.

La mutation: L'opérateur de mutation modifie un ou plusieurs gènes du chromosome (individu) sélectionné, dans le but d'introduire une nouvelle variabilité dans la population. La forme générale de l'opérateur de mutation est $m : I \to I'$, qui mute un individu I pour donner I', avec une certaine probabilité p_m $(0 \leqslant p_m \leqslant 1)$.

La sélection: L'opérateur de sélection détermine les individus qui vont se reproduire par croisement et mutation. Généralement, ils sont choisis selon leur performance.

Le remplacement: Cette opération remplace les parents au moyen d'une sélection darwinienne parmi les enfants, avec participation éventuelle des parents.

A chacune des étapes de l'algorithme décrites ci-dessus, il faut effectuer le compromis entre **l'exploitation** et **l'exploration**. Ces deux notions, définies ci-dessous, sont des points clés dans les EA, et constituent un dilemme difficile à résoudre.

Exploitation: L'idée est d'exploiter efficacement l'information obtenue précédemment par les meilleures solutions pour spéculer sur la position de nouveaux points à explorer, avec l'espoir d'améliorer la performance. Son but est d'effectuer une recherche locale dans le voisinage des meilleurs individus de la population.
Toutefois, l'exploitation toute seule ne permet pas de préserver la **diversité génétique**, puisqu'elle guide la population vers le plus proche optimum local, qui risque de dominer la population rapidement. Cette dernière devient alors homogène et son évolution se réduit à l'évolution de l'individu dominant. Ce phénomène est appelé la **convergence prématurée**.

Exploration: Son but est d'explorer des nouvelles régions de l'espace de recherche pour introduire une information innovatrice dans la population. C'est l'exploration qui aide au maintient de la diversité de la population. Cependant, l'excès d'exploration ne permet pas à l'algorithme de converger (en cas extrême d'exploration, l'évolution de la population se résume à une **marche aléatoire**). Il faut donc maintenir un équilibre entre l'exploitation des bonnes solutions rencontrées et l'exploration des zones inconnues de Ω pour garantir l'efficacité de l'algorithme.

Typiquement, les opérations de sélection et de croisement sont des étapes d'exploitation, alors que l'initialisation et la mutation sont des étapes d'exploration. Toutefois, la mutation peut devenir un opérateur d'exploitation si les perturbations générées sont très petites. De plus, on peut perdre la diversité de la population au cours de la procédure de sélection des survivants pour la génération suivante. En effet, si on sélectionne plusieurs copies des meilleurs, le risque de la convergence prématurée se présente. Mais si la variance de la sélection est trop grande, c'est la

dérive génétique qu'on risque, qui permet à certains individus de survivre au détriment d'individus meilleurs, et éloigne à priori la population de l'optimum. Il faut donc que la sélection soit capable de maintenir la diversité tout en favorisant les meilleurs.

1.1.3 Historique

Les Algorithmes Génétiques: GA

Les GA sont probablement les algorithmes les plus connus et utilisés dans le calcul évolutionnaire. Il ont été développés dans les années soixante par Holland [56]. L'idée de base de son système était d'étudier, dans le cadre de la psychologie/biologie, le processus d'adaptation des populations, en se basant sur des données sensorielles introduites au système grâce à des détecteurs binaires [56, 57].

Les GA ont été appliqués à l'optimisation paramétrique pour la première fois par De Jong en 1975 [21], qui a posé les fondements de cette technique d'application. Cependant, le manque de puissance des ordinateurs à l'époque ne permettait pas leur application sur des problèmes réels de grande taille. Ce n'est que pendant les années quatre vingt dix, précisément avec l'apparition de l'ouvrage de référence écrit par Goldberg [42], que les GA se sont fait connaître dans la communauté scientifique.

Les Stratégies d'Evolution: ES

Les Stratégies d'Evolution sont dédiées à la résolution de problèmes d'optimisation numérique dans l'espace des vecteurs de réels. Les premiers efforts pour la mise en place des stratégies d'évolution ont eu lieu en 1964 à l'université de Berlin au cours de la résolution d'un problème aérodynamique [119]. En 1965, Rechenberg a introduit l'algorithme (1+1)-ES (section 1.5.3) qui fait évoluer un seul individu et utilise la mutation Gaussienne (section 1.3.2) pour assurer cette évolution. Il a proposé la règle 1/5 pour l'adaptation de la déviation standard de la mutation [97].

Cette stratégie a été la base pour la transition à $(\mu+\lambda)$-ES et (μ, λ)-ES (section 1.5.3) introduits par Schwefel en 1977 [120, 121]. De même, des nouvelles approches de mutation et de croisement ont été mises en place. Sont alors apparus les notions d'auto-adaptativité pour la mutation ainsi que des différents types de croisement entre vecteurs réels illustrés dans la section 1.3.

La Programmation Evolutionnaire: EP

La Programmation Evolutionnaire a été développée par L.J. Fogel [36] en Californie dans les années 60, dans l'espace des automates à états finis (figure 1.2), pour la résolution de problèmes de prédiction. La table de transition des automates est modifiée grâce à des mutations aléatoires uniformes dans l'alphabet discret correspondant. L'évaluation de la performance des individus correspond au nombre de symboles prédits correctement. Chaque automate de la population parente génère un enfant par mutation, et les meilleures solutions entre les parents et les enfants sont sélectionnées pour survivre, ce qui correspond à la stratégie $(\mu + \mu)$ dans la terminologie des ES.

Les EP ont été ensuite développés et leur domaine a été élargi par D.B. Fogel, afin qu'il puissent travailler dans l'espace réel [33, 34], où la sélection déterministe est remplacée par un tournoi stochastique.

La Programmation Génétique: GP

La première utilisation des structures arborescentes dans un algorithme génétique a été faite par Cramer en 1985 [16], dans le but de faire évoluer des sous-programmes séquentiels d'un langage

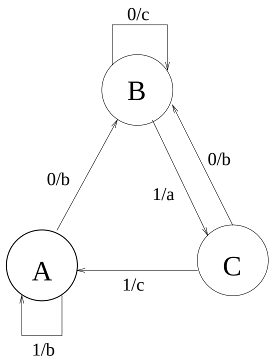


FIG. 1.2 – *Exemple d'un automate à états finis ayant trois états différents* $S = \{A, B, C\}$, *un alphabet d'entrée* $I = \{0, 1\}$, *et un alphabet de sortie* $O = \{a, b, c\}$. *Chaque arête entre deux états indique une transition possible, et la fonction de transition* $\delta : S \times I \rightarrow S \times O$ *est spécifiée par les labels au niveau des arêtes ayant la forme* i/o, *signifiant que* $\delta((s_i, i)) = (s_j, o)$.

algorithmique simple. L'algorithme d'évolution utilisé est le SSGA (c.f. section 1.5.4).

L'adoption de cette présentation pour définir la programmation génétique comme un nouvel algorithme évolutionnaire a été faite par John Koza en 1992 [69]. Son objectif initial était de faire évoluer des sous-programmes du langage LISP (figure 1.3). Grâce à l'ouvrage de Koza [69], l'utilisation de GP s'est étendue pour la résolution de nombreux types de problèmes dont les solutions peuvent être représentées par des structures arborescentes, comme les représentations fonctionnelles linéaires [89], les graphes [129, 107], les structures moléculaires, ...

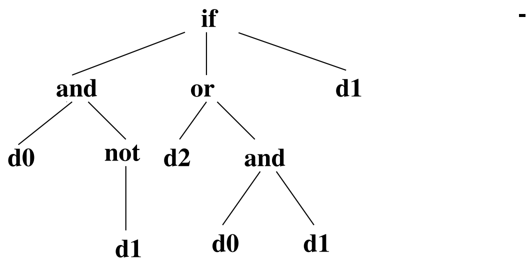


FIG. 1.3 – *Exemple d'une solution GP en LISP:* $\{d0, d1, d2\}$ *est un ensemble d'instructions constituant les terminaux, et* $\{if, and, or\}$ *sont des expressions LISP constituant les nœuds.*

1.2 La Représentation Binaire

Le codage binaire est le cadre général des GA traditionnels [42]. Chaque individu I est représenté par un vecteur binaire (ou chaîne de bits), où chaque élément prend la valeur 0,1:

$$I = (a_1, \cdots, a_l) \in \{0, 1\}^l,$$

où l est la taille du vecteur (nombre de bits). L'espace de recherche Ω (espace phénotypique) est alors $\{0,1\}^l$.

Cette représentation s'adapte bien à des problèmes où les solutions potentielles ont une représentation binaire canonique, comme les problèmes booléens. Elle s'applique aussi pour des problèmes d'optimisation paramétrique continue($f : \mathbb{R}^n \rightarrow \mathbb{R}$), mais il est nécessaire de définir une technique de codage adéquate de $\mathbb{R}^n$ vers $\{0,1\}^l$.

1.2.1 Le croisement binaire

Le croisement est vu comme l'opérateur d'exploration essentiel des GA. Son rôle consiste à combiner les génotypes de deux individus pour en obtenir deux nouveaux, en échangeant un ou plusieurs fragments des deux génotypes. On distingue plusieurs croisements possibles, dont les plus utilisés sont:

- Le croisement à 1 point [56]: un seul fragment est échangé selon un point de coupure choisi aléatoirement,
- Le croisement à 2 points [23, 24]: deux fragments sont échangés selon 2 points de coupure choisis aléatoirement,
- Le croisement Uniforme [128]: On échange les bits à chaque position indépendamment avec une probabilité de 0.5. Il peut être vu comme un croisement multi-points dont le nombre de coupures est déterminé aléatoirement au cours de l'opération.

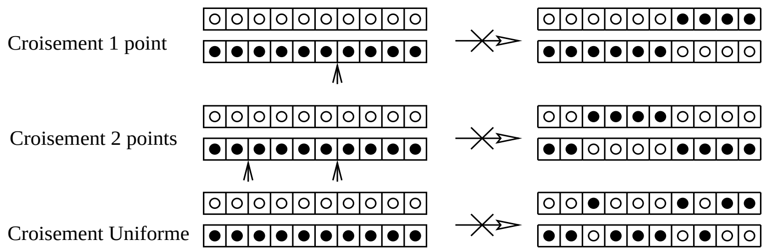


FIG. 1.4 – *Les divers croisements binaires: à 1 point, à 2 points et le croisement uniforme.*

1.2.2 La mutation binaire

C'est un opérateur qui inverse aléatoirement les bits du génotype avec une faible probabilité, typiquement de 0,01 à 0,001. Il est destiné à diversifier génétiquement les éléments de la population (en son absence, aucune caractéristique génétique nouvelle ne peut apparaître). Les mutations les plus utilisées sont:

- La mutation stochastique (*bit flip*) [64, 42]: inverse chaque bit indépendamment avec une probabilité $\frac{c}{l}$, avec $c > 0$ et l est la taille du vecteur. C'est la mutation la plus employée dans la représentation binaire.
- La mutation 1 bit: inverse le symbole d'un bit choisi au hasard avec une probabilité p_m [64].

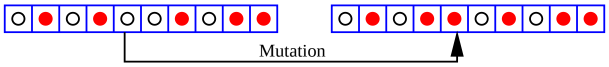


FIG. 1.5 – La mutation 1 bit: Le symbole du 5^{eme} bit a été inversé.

1.3 La Représentation Réelle

Avec la représentation réelle, l'espace de recherche est l'espace réel: $\Omega = \mathbb{R}^n$ (variables non bornées) ou une partie de l'espace réel: $\Omega = S$, avec $S \subset \mathbb{R}^n$ (variables bornées).

Cette représentation a été introduite initialement pour les stratégies d'évolution [120], mais son utilisation s'est étendue rapidement aux autres types d'algorithmes évolutionnaires. Pour des problèmes d'optimisation dans l'espace réel, l'espace des génotypes s'identifie à l'espace des phénotypes: $I = \overrightarrow{x} = (x_1, \cdots, x_n) \in \mathbb{R}^n$, où chaque $x_i \in \mathbb{R}^n$ est une variable objective.

1.3.1 Le croisement réel

Dans la représentation réelle, il y a deux manières de combiner deux allèles de deux parents: choisir l'un des deux allèles (croisement discret) ou combiner linéairement les deux (croisement intermédiaire ou arithmétique). Dans le deuxième cas, plusieurs variantes ont été proposées.

– Le croisement discret:
$$\forall i \in \{1, \cdots, n\},\ x'_i = x_{S,i} \text{ ou } x_{T,i},$$
où n est la taille du vecteur réel et S et T sont deux individus sélectionnés de la population parente pour l'ensemble des composantes de $\overrightarrow{x}$.

– Le croisement intermédiaire (arithmétique):
$$\forall i \in \{1, \cdots, n\},\ x'_i = \alpha x_{S,i} + (1-\alpha) x_{T,i},$$
où α est une variable aléatoire uniforme appartenant à l'intervalle $[0,1]$ ($\alpha = \mathcal{U}[0,1]$) et S et T sont deux individus sélectionnés pour l'ensemble des composantes de $\overrightarrow{x}$.
Une autre variante du croisement intermédiaire est le croisement BLX-ϕ ("*Blend Crossover*"), proposé par Eshelman et Schaffer [27], qui a la propriété de pouvoir élargir le domaine de définition de l'enfant, en utilisant un intervalle plus large pour α. En effet, α n'est plus compris entre 0 et 1, mais entre $(-\phi)$ et $(1+\phi)$, avec $0 < \phi < 1$ (figure 1.6). On note que le croisement BLX-0 correspond au croisement arithmétique traditionnel.

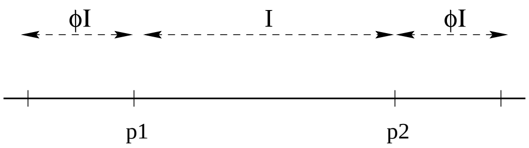


FIG. 1.6 – *BLX-ϕ.*

Une deuxième variante du croisement arithmétique a été proposée par Michalewiz [82], appelée le croisement uniforme. Cet opérateur a été conçu principalement pour le système

Genecop (GEnetic algorithm for Numerical Optimization of Constrained Problems) (chapitre 2, section 2.5.1). Il croise deux parents $\overrightarrow{x_1}$ et $\overrightarrow{x_2}$ à la position k, $(k \in [1,n])$, et donne les deux enfants $\overrightarrow{x}'_1$ et $\overrightarrow{x}'_2$ définis comme suit:

$$\begin{aligned}\overrightarrow{x}'_1 &= (x_1, \cdots, x_k, y_{k+1}.\alpha + x_{k+1}.(1-\alpha), \cdots, y_n.\alpha + x_n.(1-\alpha))\\ \overrightarrow{x}'_2 &= (y_1, \cdots, y_k, x_{k+1}.\alpha + y_{k+1}.(1-\alpha), \cdots, x_n.\alpha + y_n.(1-\alpha))\end{aligned}$$

avec $\alpha = \mathcal{U}[0,1]$.

Dans son ouvrage sur la théorie des stratégies d'évolution [121], Schwefel a imaginé un autre type de croisement où toute la population parente peut participer à la génération d'un enfant, appelé le croisement global. Chaque parent est sélectionné indépendamment. Comme pour le croisement simple, selon la stratégie de génération des allèles de l'enfant (discret ou intermédiaire), deux types de croisement global sont possibles: le croisement global discret et le croisement global intermédiaire.

On note que le croisement multi-parent existe aussi dans la représentation binaire [26]

- Le croisement global discret:

$$\forall i \in \{1, \cdots, n\},\ x'_i = x_{S_i,i},$$

  où S_i est un individu sélectionné de la population parente pour la composante x_i. On note qu'il y a autant de parents sélectionnés que de composantes dans l'individu.
- Le croisement global intermédiaire:

$$\forall i \in \{1, \cdots, n\},\ \ x'_i = \alpha_i x_{S_i,i} + (1-\alpha)x_{T_i,i},$$

  où $\alpha_i = \mathcal{U}[0,1]$ et S_i et T_i sont deux individus sélectionnés de la population parente pour la composante x_i.

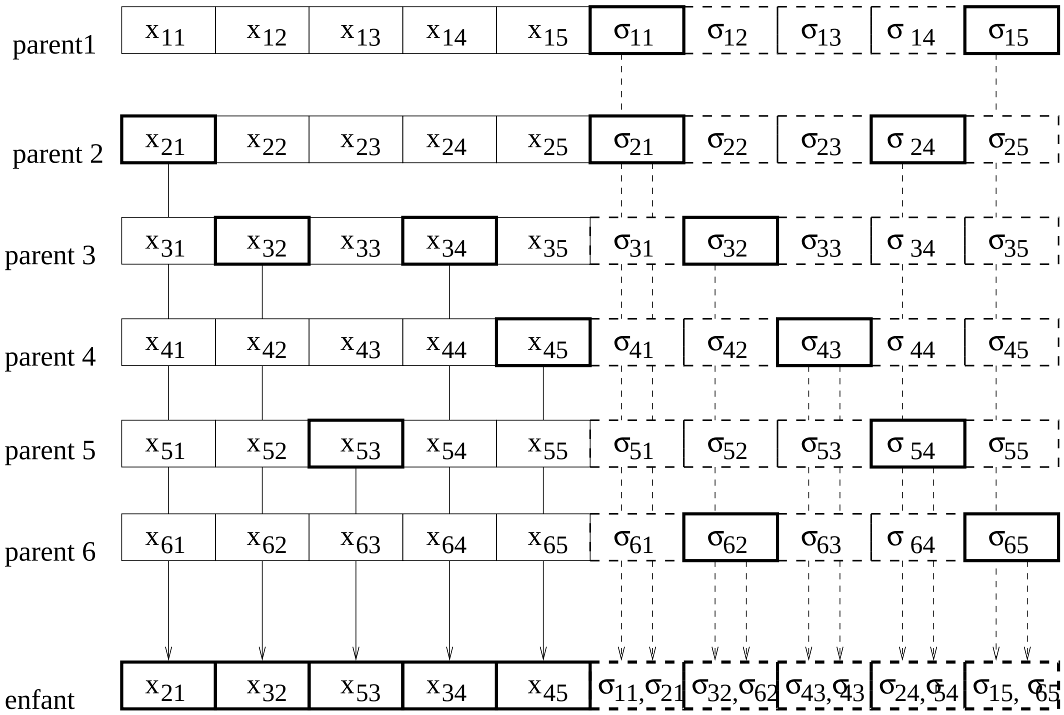
1.3.2 La mutation réelle

Une grande variété de stratégies de mutation ont été proposées pour la représentation réelle. Nous présentons ci-dessous les plus connues, classées en deux catégories: les mutations Gaussiennes, qui modifient un individu en lui ajoutant un bruit Gaussien [62, 97, 121, 122], et d'autres mutations basées sur des principes différents [82].

Les mutations Gaussiennes

La principale technique utilisée pour la mutation réelle est l'ajout d'un bruit Gaussien aux différentes composantes du vecteur $\overrightarrow{x}$. La difficulté de cette approche est l'ajustement de la déviation standard σ du bruit généré. En effet, au début de l'évolution d'une population, la déviation standard doit être assez élevée pour générer des fortes perturbations et ainsi explorer rapidement tout l'espace de recherche. Une fois les pics de la fonction étudiée localisés, l'algorithme doit être capable de déterminer avec précision les solutions optimales.

Le principal moyen d'assurer cette convergence est de générer des faibles perturbations pour la mutation des meilleurs individus. Pour se faire, la déviation standard de la mutation Gaussienne doit être de l'ordre de la précision demandée. Par contre, une petite déviation au début de l'évolution bride la recherche. Par conséquent, la déviation ne doit pas être constante au cours de



$$\overrightarrow{x'} = (x_{21}, x_{32}, x_{53}, x_{34}, x_{45}, (\alpha_1\sigma_{11} + (1-\alpha_1)\sigma_{21}), (\alpha_2\sigma_{32} + (1-\alpha_2)\sigma_{62}), \sigma_{43}, (\alpha_4\sigma_{24} + (1-\alpha_4)\sigma_{54}), (\alpha_5\sigma_{15} + (1-\alpha_5)\sigma_{65})),\ \text{avec } \alpha_i \in [0,1],\ i = 1, \cdots, 5$$

FIG. 1.7 – *Exemple d'un croisement global: les composantes principales (x_i) de l'enfant sont générés avec un croisement global discret, alors que ses déviations standards (σ_i) sont générées avec un croisement global intermédiaire.*

l'évolution. Pour résoudre ce problème, plusieurs stratégies adaptatives et auto-adaptatives ont été proposées pour ajuster la déviation au cours de l'évolution, comme la règle des $1/5$ proposé par Rechenberg en 73 [97] ou la mutation Log-Normale proposée par Schwefel en 81 [121]. Parmi les plus connues, on compte quatre approches, présentées ci-dessous.

- La mutation Gaussienne statique, qui modifie toutes les composantes d'un individu en ajoutant un bruit Gaussien:
$$\forall i \in \{1, \cdots, n\}, \;\; x'_i = x_i + N(0, \sigma),$$
où σ est unique pour toute la population. Il est défini par l'utilisateur et sa valeur reste inchangée au long de l'évolution.
- La mutation Gaussienne adaptative: la règle des $1/5$: La mutation statique, avec une grande valeur de σ, permet d'explorer l'espace de recherche, mais ne permet pas l'exploitation des meilleures solutions. Par contre, une petite valeur de σ permet une exploitation locale des solutions, mais ne permet pas de visiter des nouvelles régions de l'espace de recherche.
Pour résoudre ce problème, Rechenberg propose de mettre à jour la valeur de σ toutes les générations. Il a défini la règle des $1/5$ [97], qui mesure la probabilité p_s des mutations réussies (performance de I' est meilleure que celle de I), et ajuste σ à la génération t comme suit:
$$\sigma(t) = \begin{cases} \sigma(t-1) \times fact & si \;\; p_s < 0.2 \\ \sigma(t-1)/fact & si \;\; p_s > 0.2 \\ \sigma(t-1) & si \;\; p_s = 0.2 \end{cases}$$
où $0 < fact < 1$ est le taux d'adaptation de σ. Selon une étude faite par Schwefel en [121], une bonne valeur de $fact$ est 0.83. De plus, cette règle peut être appliquée toute les k générations au lieu de toutes les générations (k paramètre de la méthode).
Avec cette règle, le taux des mutations réussies doit être toujours proche de 0.2, ce qui n'est pas toujours un taux idéal pour le processus de recherche. En plus, l'utilisation d'un seul paramètre pour contrôler la mutation de toute la population limite la robustesse de cette stratégie. Les efforts se sont alors orientés vers l'auto-adaptativité des variances de la mutation, dans le but contrôler les perturbations générées pour chaque individu.
- La mutation Log-Normale auto-adaptative isotrope [121], où les variables x_i d'un individu $\overrightarrow{x}$ sont mutées en additionnant une quantité aléatoire normalement distribuée dans [0,1], avec une déviation standard σ unique pour toutes les variables. Chaque individu a son propre σ incorporé dans son code génotypique: $I = (\overrightarrow{x}, \sigma)$. σ est ajusté à chaque itération selon une loi Log-Normale en utilisant un taux d'adaptation τ, et ceci avant d'appliquer la mutation:
$$\sigma' = \sigma * \exp(\tau * N(0,1))$$
$$\forall i \in \{1, \cdots, n\}, \;\; x'_i = x_i + N_i(0, \sigma')$$
avec $0 < \tau \leqslant 1$. Selon Bäck et Schwefel [10], une valeur optimale pour τ serait: $\tau \simeq (\sqrt{n})^{-1}$
- La mutation Log-Normale auto-adaptative anisotrope [122], où une déviation standard σ_i est attachée à chaque variable x_i encapsulée dans le génotype ($I = (\overrightarrow{x}, \overrightarrow{\sigma})$). L'opérateur de mutation mute d'abord les déviations σ_i selon une loi Log-Normale, ensuite, il modifie la variable x_i en lui ajoutant un bruit Gaussien généré aléatoirement en utilisant la nouvelle déviation σ'_i:
$$\forall i \in \{1, \cdots, n\}, \;\; \sigma'_i = \sigma_i * \exp(\tau_L * N(0,1) + \tau * N_i(0,1))$$
$$\forall i \in \{1, \cdots, n\}, \;\; x'_i = x_i + N(0, \sigma'_i)$$

où $0 < \tau_L \leqslant 1$ est le taux d'adaptation local et $0 < \tau \leqslant 1$ est le taux d'adaptation global. Bäck et Schwefel proposent, pour définir τ et τ_L, les valeurs suivantes: $\tau \simeq (\sqrt{2(\sqrt{n})}^{-1}$ et $\tau_L \simeq (\sqrt{2n})^{-1}$.

- La mutation corrélée: C'est une autre approche de mutation auto-adaptative proposée par Schwefel, utilisant un écart-type matriciel pour la mutation des composantes d'une solution donnée, dans le but d'orienter la distribution normale vers la direction d'amélioration de la variable (figure 1.8). Un angle de rotation est alors incorporé dans le génotype de chaque individu qui devient $(\overrightarrow{x}, \overrightarrow{\sigma}, \overrightarrow{\alpha})$. La mutation d'une solution se passe en 3 étapes:
  1. D'abord, on mute les composantes du vecteur des déviations standards σ_i selon le principe décrit précédemment.
  2. On procède ensuite à la mutation du vecteur des angles d'orientation $\overrightarrow{\alpha}$ en additionnant une quantité normalement distribuée pour chaque composante α_j $(j = 1, \cdots, \frac{n(n-1)}{2})$.
  3. La troisième et dernière étape est la mutation de la composante x_i en ajoutant une quantité aléatoire selon une distribution normale de dimension n $(\overrightarrow{N}, \overrightarrow{C'})$.

$$\begin{aligned} \forall i \in \{1, \cdots, n\}, \ \ \sigma'_i &= \sigma_i * \exp(\tau_L * N(0,1) + \tau * N_i(0,1)) \\ \forall j \in \{1, \cdots, \ \tfrac{n(n-1)}{2}\}, \alpha'_j &= \alpha_j + \beta . N_j(0,1), \\ \overrightarrow{x'} &= \overrightarrow{x} + \overrightarrow{N}(\overrightarrow{0}, C'(\overrightarrow{\sigma'}, \overrightarrow{\alpha'})) \end{aligned}$$

  où β est un paramètre exogène, que Schwefel propose de le fixer à 0.0873.
  La matrice des covariances C est définie positive. Elle est calculée à partir du produit des $\frac{n(n-1)}{2}$ rotation (α_i) par la diagonale de (σ_i).

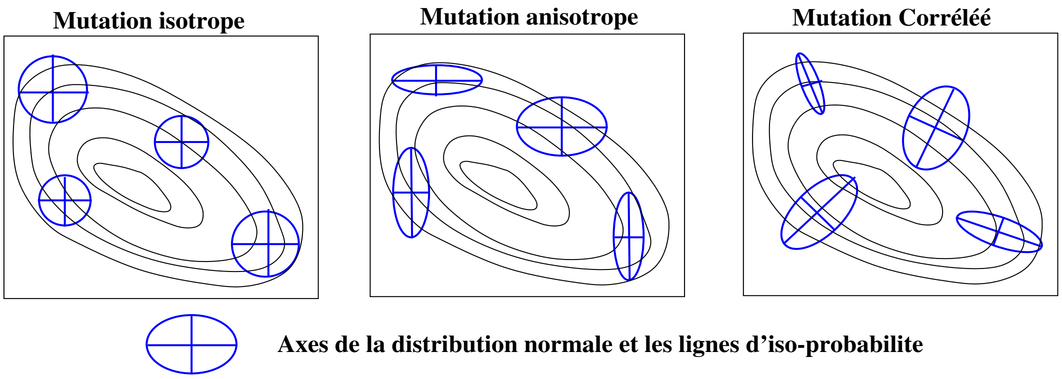


FIG. 1.8 – *Ellipsoïdes pour les directions de mutation pour les mutations auto-adaptatives isotrope, anisotrope et corrélée*

On note que les vecteurs $\overrightarrow{\sigma}$ des déviations standards et $\overrightarrow{\alpha}$ des angles de rotation sont aussi croisés et indépendamment du vecteur $\overrightarrow{x}$ (figure 1.7).

On note aussi que la complexité et la taille du code du génotype croîent en passant d'une méthode à une autre: de la mutation Gaussienne statique jusqu'à la mutation corrélée. Cette dernière n'est presque jamais utilisée, vue la complexité du calcul de la matrice des covariances C. De plus, les mutations auto-adaptatives nécessitent plus de paramètres à définir par l'utilisateur.

Cependant, des études comparatives réalisées par Saravanan et Fogel [109] et Saravanan et al [110], sur plusieurs cas test, montrent que la mutation Log-Normale avec auto-adaptativité a généralement une performance supérieure à celle adaptative. Une autre étude comparative entre la deuxième et la troisième stratégies présentées dans cette section, et une nouvelle stratégie de mutation proposée dans [94, 48] est présentée dans le chapitre 3, mais pour un cas test sous contraintes.

Autres opérateurs de mutation

D'autres opérateurs de mutation de vecteurs réels ont été proposés par Michalewicz dans [82], dédiés à des problèmes contraints (variables bornés et/ou soumises à des contraintes linéaires). Ces opérateurs ont été mis en place pour le système Genecop (chapitre 2, section 2.5.1). Ce sont des opérateurs unaires, qui à partir d'un parent $\overrightarrow{x} = (x_1, \cdots, x_k, \cdots, x_n)$, génèrent un enfant $\overrightarrow{x}' = (x_1, \cdots, x'_k, \cdots, x_n)$, où x'_k est l'élément muté et k est choisi aléatoirement dans l'intervalle $[1, n]$. Pour calculer x'_k, ils utilisent le domaine admissible des variables du rang k $[l(k), u(k)]$, défini à partir des bornes de l'espace de recherche et des contraintes du problème s'il y en a, à condition qu'elles soient linéaires. Trois types de mutation sont alors proposées.

- La mutation Uniforme:
  Avec la mutation uniforme, x'_k prend une valeur aléatoire (distribution de probabilité uniforme) dans l'intervalle $[l(k), u(k)]$.
- La mutation aux frontières:
  C'est une variation de la mutation uniforme. Avec cette mutation, x'_k prend une des deux valeurs des bornes de l'intervalle de validité correspondant: $l(k)$ ou $u(k)$, avec des probabilités égales. Cet opérateur a été construit afin de traiter les cas où l'optimum se trouve tout près ou sur les frontières du domaine admissible. Par contre, cet opérateur peut nuire au processus d'évolution si l'espace de recherche est très vaste.
- La mutation Non-Uniforme ou Triangulaire:
  C'est cet opérateur qui offre au système Genecop ses grandes capacités de convergence, en lui permettant un ajustement précis des solutions, spécialement pendant les dernières étapes de l'évolution. La définition de la nouvelle valeur de l'élément muté x'_k se fait comme suit:

  $$x'_k = \begin{cases} x_k + \Delta(t, u(k) - x_k) & \text{si S=0} \\ x_k - \Delta(t, x_k - l(k)) & \text{si S=1,} \end{cases}$$

  avec S est une variable aléatoire égale ± 1.
  La fonction $\Delta(t, y)$ retourne une valeur dans l'intervalle $[0, y]$ telle que la probabilité que $\Delta(t, y)$ soit proche de 0 augmente avec l'augmentation de t (t est le nombre de générations courant). Cette propriété permet à l'algorithme d'explorer l'espace de recherche uniformément pendant les premières générations, et très localement pendant des étapes plus avancées. La définition choisie par les auteurs pour cette fonction est:

  $$\Delta(t, y) = y.r.(1 - \frac{t}{T})^b,$$

  où $r = \mathcal{U}[0, 1]$, T est le nombre maximal de générations et b est un paramètre du système déterminant le degré de non-uniformité.

1.4 La Représentation Fonctionnelle

Dans la représentation fonctionnelle, une solution dans l'espace génotypique est représentée par une structure arborescente dynamique, souvent considérée comme un *programme génétique*.

En tant que structures d'arbres –graphes acycliques–, les *programmes génétiques* requièrent la donnée de l'ensemble des fonctions de base et des terminaux, décrivant alors le "langage de programmation" du problème à résoudre. En effet, avec les fonctions prenant place au niveau des nœuds et des terminaux s'assimilant à des feuilles de l'arbre, nous obtenons un jeu d'instructions constituant les briques élémentaires qui, assemblées judicieusement, doivent fournir une solution. Dans la représentation fonctionnelle, cette solution est une fonction à deux dimensions ($f(x, y)$), construit à partir de:

1. un ensemble de terminaux ($\mathcal{T}$): les variables, les constantes universelles, fonctions sans arguments (rnd(), time(), ...).
2. un ensemble de nœuds ($\mathcal{N}$): des opérateurs: $*, -, +$, des fonctions: *sin*, *cos*, ...

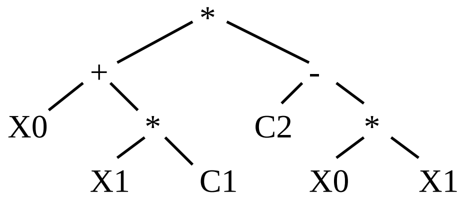


FIG. 1.9 – *Exemple d'une solution en GP:* $\mathcal{N} = \{*, +, -\}$ *et* $\mathcal{T} = \{X0, X1, \mathbb{R}\}$. *L'espace exploré est celui des polynômes réels à 2 variables.*

1.4.1 Le croisement en GP

Typiquement, la stratégie de la recombinaison consiste en un échange de deux sous-arbres des deux individus à croiser, sélectionnés *a priori* parmi les plus performants, donc contenant potentiellement des sous-structures intéressantes. Le choix des sous-arbres à échanger est généralement fait par tirage uniforme.

1.4.2 La mutation en GP

Le GP traditionnel proposé par Koza [69] n'utilise pas d'opérateurs de mutation. Pour assurer l'accès à toutes les primitives du langage de recherche (e.g. LISP) et assurer la diversité génétique, on utilise des populations de très grande taille, pour avoir le maximum d'information. C'est en 1996 que la mutation a été introduite par Angeline [6, 4] dans le but de réduire la taille des populations du GP standard.

On distingue de multiples sortes de mutations du fait de la complexité des structures arborescentes, dont les capacités exploratoires peuvent être locales ou, à l'inverse, de grande envergure. Parmi les différentes mutations, les plus courantes sont:

- Mutation par insertion (*grow mutation*): on ajoute un nœud et des feuilles complémentaires n'importe où dans l'arbre (figure 1.11).
- Mutation par promotion (*shrink mutation*) : un nœud interne est ôté et l'un de ses fils remonte prendre sa place (figure 1.12).

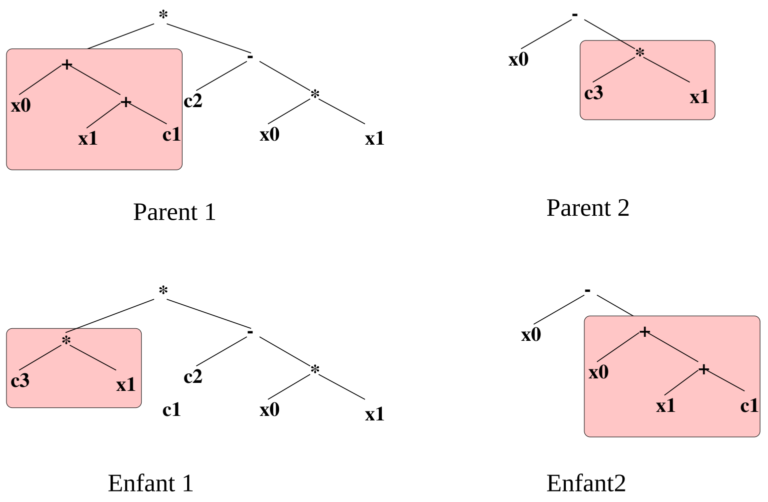


FIG. 1.10 – *Exemple de croisement de deux individus en GP*

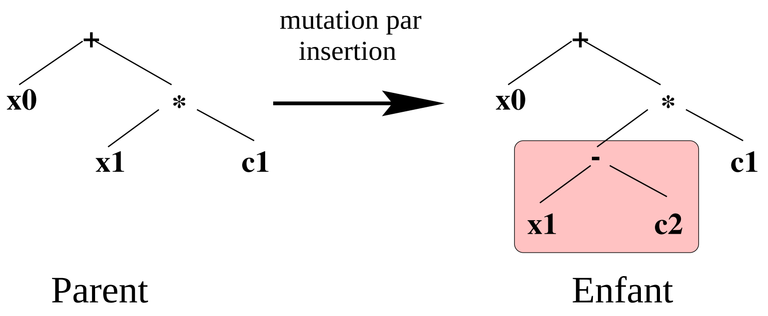


FIG. 1.11 – *Exemple de la mutation par insertion en GP.*

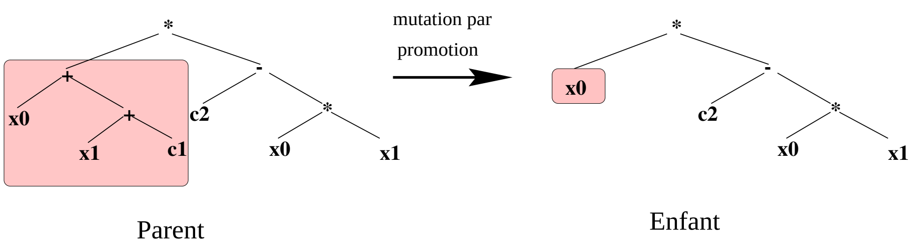


FIG. 1.12 – *Exemple de la mutation par promotion en GP.*

– Mutation d'un nœud (*cycle mutation*): un nœud de l'arbre est changé en un autre de même arité (figure 1.13).

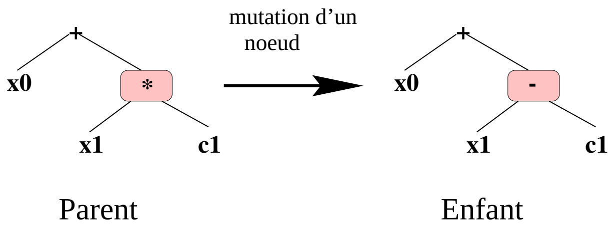


FIG. 1.13 – *Exemple de mutation d'un nœud en GP*

– Mutation d'une branche: on élague une branche de l'arbre et on la remplace par un sous-arbre généré aléatoirement (figure 1.14).

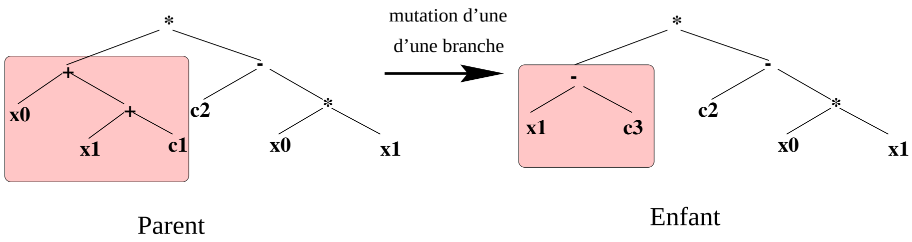


FIG. 1.14 – *Exemple de mutation d'une branche en GP*

Dans le cas où les terminaux peuvent être des constantes numériques, d'autres mutations ont été introduites, dont on cite:

– Mutation des constantes: quelques constantes sont modifiées selon une loi Gaussienne ou uniforme [4].
– Mutation optimisée des constantes: on applique quelques itérations d'un *hill climber* aléatoire en vue d'affiner les constantes [116].

1.5 Le Darwinisme Artificiel

Le Darwinisme dans un algorithme évolutionnaire est appelé dans deux étapes différentes au cours d'une génération donnée: l'étape de reproduction afin de sélectionner les parents qui vont se reproduire et l'étape de remplacement, qui remplace les individus non sélectionnés par ceux désignés à survivre, constituant ainsi la nouvelle population (figure 1.1).

La sélection est un opérateur principal utilisé dans la programmation évolutionnaire, dont le premier rôle est d'accentuer les meilleurs individus dans la population. Il sélectionne les individus relativement performants pour se reproduire. Il doit aider l'algorithme à converger, et en même temps, maintenir la diversité de la population, afin d'éviter la convergence prématurée. Plusieurs stratégies ont été mises en place, dont nous présentons les plus utilisées.

1.5.1 La sélection proportionnelle

La roulette

La première sélection mise en place pour les GA est le tirage à la roulette, ou *Roulette Wheel Selection* (*RWS*) introduite par Goldberg [42]. C'est une méthode stochastique qui exploite la métaphore d'une roulette de casino, qui compterait autant de cases que d'individus dans la population. La largeur de la case d'un individu $\overrightarrow{x_i}$ est proportionnelle à sa performance $f(\overrightarrow{x_i})$: $\frac{f(\overrightarrow{x_i})}{\sum_j^n f(\overrightarrow{x_j})}$. Le roue étant lancée, l'individu sélectionné est désigné par l'arrêt de la roue sur sa case (figure 1.15).

L'espérance n_i de nombre de copies d'un élément $\overrightarrow{x_i}$ de la population courante est donnée par l'expression:

$$n_i = \frac{N}{\sum_j^N f(\overrightarrow{x_j})} f(\overrightarrow{x_i}),$$

où N est le nombre d'individus. L'espérance maximale $Max(n_i)$ dans l'ensemble des éléments de la population est appelée la **pression sélective**.

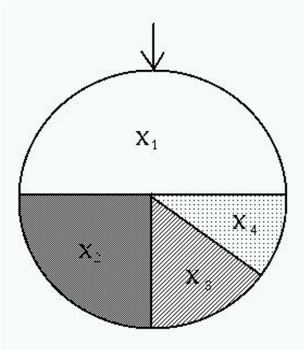


FIG. 1.15 – *Roue de loterie pour une population de 4 individus avec $f(x_i) = \{50, 25, 15, 10\}$. Pour tirer un élément, on lance la roue, et si elle s'arrête sur la case i, x_i est sélectionné.*

Cette méthode favorise les meilleurs individus, mais tous les individus ont toujours des chances d'être sélectionnés. Cependant, elle peut causer la perte de la diversité de la population si la pression sélective (ou n_i du meilleur) est élevée. De plus, sa variance et son coût sont élevés.

Afin de diminuer la variance, Baker a proposé en 1987 la sélection à la roulette avec reste stochastique, ou *stochastic universal sampling* (SUS) [12]. Avec cette approche, d'une façon similaire à *RWS*, on considère une roulette partitionnée en autant de cases que d'individus, où chaque case est de taille proportionnelle à la performance. Mais cette fois, les individus sélectionnés sont désignés par un ensemble de points équidistants (figure 1.16). Le nombre effectif de copies de l'individu $\overrightarrow{x_i}$ sera la partie entière inférieure ou supérieure de son espérance n_i:

$$E(\frac{n.f(\overrightarrow{x_i})}{\sum_{i=j}^n f(\overrightarrow{x_j})})$$

Le bruit de sélection étant plus faible, la dérive génétique se manifeste moins qu'avec la méthode *RWS*.

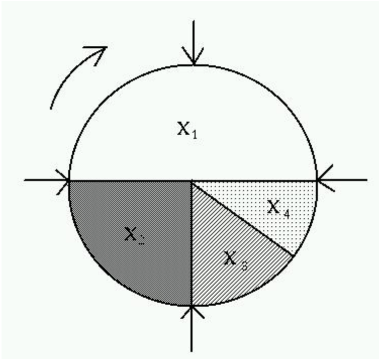


FIG. 1.16 – *Stochastic Universal Sampling Selection: pour sélectionner des individus, on lance la roue, et quand elle s'arrête, chaque individu dont la case est pointée par un marqueur est sélectionné.*

La mise à l'échelle (*scaling*)

Au cours de l'évolution d'une population, avec la sélection proportionnelle, les individus ayant les meilleures performances sont reproduits plus souvent que les autres et remplacent généralement les moins bons. Si la pression de sélection est élevée, le risque de convergence prématurée est important. Cette situation se produit lorsqu'un super-individu, nettement meilleur que les autres, devient majoritaire dans la population qu'il finit par envahir complètement (figure 1.17). L'exploration de l'espace des génotypes reste alors uniquement assurée par la mutation, puisque l'hybridation d'individus semblables ne crée plus de nouveautés. Lorsque ce point est atteint, l'exploration se ramène à une recherche locale centrée sur le super-individu.

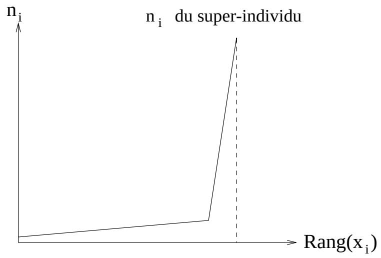
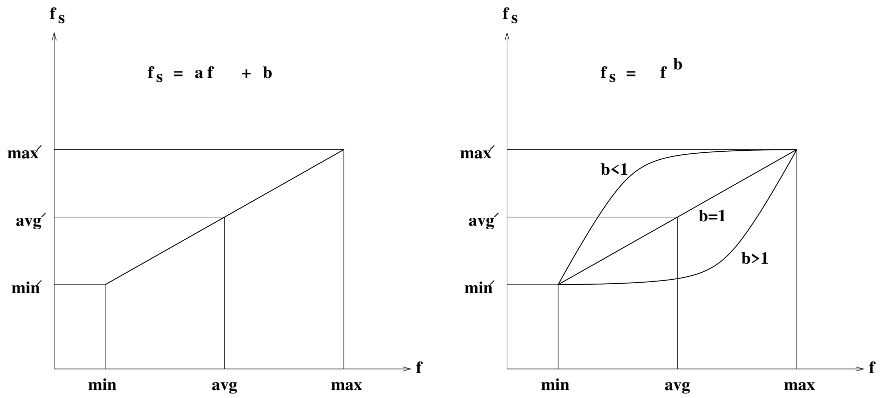


FIG. 1.17 – *Exemple d'un super-individu qui risque de dominer la population.*

Par contre, si la pression de sélection est trop faible, les meilleurs individus, qui ont un nombre espéré de descendants presque identique à celui des individus à faibles performances, ne peuvent plus guider l'évolution qui devient tributaire de la *dérive génétique*. Dans de telles situations, la sélection proportionnelle classique est insuffisante pour contourner la *dérive génétique*. Plusieurs techniques ont été alors mises en place afin de maintenir la diversité de la population, dont la mise à l'échelle et les techniques de nichage. Ces dernières, vu leur rôle primordial dans l'optimisation multimodale, ont connu pendant les dernières années une évolution importante, donnant naissance à plusieurs approches et heuristiques, dont nous présentons les plus connues dans la section 1.6.

Le principe de la mise à l'échelle consiste à effectuer une transformation affine de la fonction de performance brute f, afin de réduire les écarts entre les solutions au début du processus d'évolution (en cas de super-individu) et de les accentuer à la fin (lorsque la population commence à converger et les fitness des individus deviennent presque toutes comparables). C'est la performance modifiée f_s qui est utilisée lors de la sélection. Il existe deux types de mise à l'échelle, linéaire et exponentielle:

- Le *scaling* linéaire: $f_s = a.f + b$, où a et b sont calculés à chaque génération, de telle sorte que la pression sélective (max(n_i) ait une valeur donnée. L'allure de cette fonction est illustrée dans la (figure 1.18 (a)).
- Le *scaling* exponentiel: $f_s = f^{b(t)}$, où t désigne la génération courante et b est une fonction qui varie entre des valeurs proches de zéro au début de l'évolution (pour réduire les écarts de fitness) et des valeurs supérieures à 1 à la fin de l'évolution (pour accentuer les différences de fitness et favoriser les meilleurs) (figure 1.18 (b)).



(a) Fonction de *scaling* linéaire (b) Fonction de *scaling* exponentielle

FIG. 1.18 – *Exemple de fonctions de mise à l'échelle*

Le *Ranking*

Une autre variante de la sélection par roulette est la roulette par le rang, conçue également pour lutter contre les éventuels super-individus. La largeur de la case d'un individu donné est proportionnelle à son rang dans la population (triée par ordre décroissant de la performance). Les valeurs de la fitness sont alors sans importance pour la sélection, seul le classement des individus compte.

$$\text{largeur de la case i: } \frac{Rang(\overrightarrow{x_i})}{n \times (n-1)/2}.$$

1.5.2 Le tournoi

Comme pour le *Ranking*, la sélection par tournoi ne fait pas intervenir les valeurs des fitness, mais seulement des comparaisons entre les individus. Son principe est de choisir un groupe de q individus aléatoirement dans la population, de sélectionner d'une manière déterministe le meilleur dans ce groupe, et de recommencer l'opération jusqu'à l'obtention du nombre d'individus requis.

Au cours d'une génération, il y a autant de tournois que d'individus à sélectionner. La pression de sélection est ajustée par le nombre de participants à un tournoi (q). Un q élevé conduit à une forte pression de sélection.

L'avantage de cette technique de sélection est qu'elle est paramétrable par la valeur de q, peu sensible aux erreurs sur f et n'est pas chère à mettre en oeuvre et à exécuter. Par contre, sa variance est élevée [22].

Il existe plusieurs variantes de la sélection avec tournoi, dont on cite le tournoi de Boltzmann [130], qui assure que la distribution des valeurs d'adaptation d'une population soit proche d'une distribution de Boltzmann. Pour une compétition entre une solution courante $\overrightarrow{x_i}$ et une solution alternative $\overrightarrow{x_j}$, $\overrightarrow{x_i}$ gagne avec la probabilité $\frac{1}{1+e^{(f(\overrightarrow{x_i})-f(\overrightarrow{x_j}))/T}}$, où T est la température de sélection.

1.5.3 Le remplacement

Le remplacement déterministe

Le remplacement déterministe est utilisé dans les stratégies d'évolution. Son caractère purement déterministe lui donne un rôle clef dans l'évolution vu qu'il guide la recherche vers les zones des meilleurs individus. Il opère en sélectionnant les μ $(1 < \mu \leqslant \lambda)$ meilleures solutions parmi:

- l'union des μ parents et λ enfants: schéma appelé $(\mu + \lambda)$-ES,
- l'ensemble des λ enfants: schéma appelé (μ, λ)-ES.

Le remplacement $(\mu+\lambda)$ est élitiste et il garantit une amélioration monotone de la performance de la population, mais il s'adapte mal à un éventuel changement d'environnement. Par contre, avec le remplacement (μ, λ), la meilleure performance peut décroître, mais l'algorithme est plus flexible à des changements d'environnement. De plus, la régression des meilleures performances peut aider le processus de recherche à sortir des régions d'attraction des optima locaux et continuer l'exploration ailleurs. D'où la recommandation du schéma (μ, λ)-ES par Schwefel [121].

Le remplacement générationnel

Une autre technique de remplacement est le remplacement générationnel, utilisé principalement dans les GA standards. Avec cette approche, la population des enfants remplacent entièrement la population parente. La durée de vie d'un individu est alors d'une seule génération. Dans ses études sur la performance de cette technique, De Jong [21] a proposé de définir un paramètre G (*Generation Gap*), spécifiant le taux de remplacement à chaque génération, afin d'augmenter la durée de vie des meilleurs individus. $G = 1$ correspond au remplacement générationnel total. Une étude empirique faite par Grefenstette en 1986 [44] montre que des taux élevés de G donnent des meilleurs résultats que des taux faibles.

1.5.4 Les schémas d'évolution

On regroupe sous ce nom les ensembles sélection/remplacement, qui ne peuvent être dissociés lors des analyses du darwinisme au sein des algorithmes évolutionnaires. Un schéma d'évolution est donc la réunion d'une procédure de sélection et d'une procédure de remplacement. Toute combinaison des procédures présentées plus haut est licite. Toutefois, certaines combinaisons sont plus souvent utilisées, que ce soit pour des raisons historiques, théoriques ou expérimentales. Pour cette raison, les noms donnés sont souvent les noms des écoles historiques qui les ont popularisées. Ces écoles travaillaient sur des espaces de recherche bien précis, cependant, les schémas sont totalement indépendants de l'espace de recherche.

- **L'algorithme génétique standard: SGA** (*"Simple Genetic Algorithm"*)
  Ce schéma est basé sur le remplacement générationnel. A chaque génération, N parents sont sélectionnés stochastiquement par une des méthodes décrites dans les sections 1.5.1 et 1.5.2. Ces N parents donnent naissance ensuite à N enfants par application des opérateurs génétiques (avec des probabilités données). Enfin, ces N enfants remplacent purement et simplement les N parents pour la génération suivante.

- **Les stratégies d'évolution: ES+, ES,**
  Deux types de schémas sont possibles dans les stratégies d'évolution: le schéma "ES+" basé sur le remplacement déterministe $(\mu + \lambda)$, et le schéma "ES," basé sur le remplacement déterministe (μ, λ). A chaque génération, les λ enfants sont générés à partir des parents sélectionnés par tirage uniforme (on peut dire qu'il n'y a pas de sélection au sens darwinien). On note que ce schéma est souvent associé à la représentation réelle.

- **Le schéma stationnaire: SSGA** (*"Steady State Genetic Algorithm"*)
  Ce schéma consiste à produire, à chaque génération, seulement un ou deux enfants à partir d'un ou deux parents sélectionnés selon leurs performances. Ces enfants remplacent alors des individus de la population parente, sélectionnés soit d'une manière déterministe (les pires dans la population), soit par tournoi inversé, ou selon les âges (les plus vieux).
  Ce modèle s'applique aussi pour les stratégies d'évolution avec l'approche $((\mu + 1) - ES)$ où $\mu > 1$, introduite par Rechenberg en 73 [97], où tout les parents peuvent participer à la production de l'enfant. Cependant, il n'est pas très utilisé parce qu'il ne permet pas l'implantation de l'auto-adaptation pour la mutation.

- **La programmation évolutionnaire: EP**
  Le schéma utilisé ressemble à celui des stratégies d'évolution ES+, quoique développé complètement indépendamment. Cependant, avec EP, le nombre des enfants et celui des parents sont identiques: $(N + N) - ES$.

1.6 Le nichage

Les techniques de nichages ont pour objectif la formation et le maintien de sous-populations (niches) dans la même population (figure 1.19) dans le but de:

- éviter la convergence prématurée vers un optimum local,
- identifier plusieurs optima ou quasi-optima.

On distingue deux stratégies de nichage: les stratégies à voisinage implicite (pas de paramètre de voisinage à fixer) [59] et les stratégies à voisinage explicite. Les méthodes présentées dans cette section se basent sur le voisinage explicite.

Les méthodes utilisant la fonction de performance comme une source unique à partager entre les différents individus de la population bénéficient du plus grand succès par rapport aux autres méthodes de nichage, spécialement dans l'optimisation des fonctions multimodales. Parmi ces méthodes, on cite le *sharing* et l'éclaircissement élitiste, présentés dans les sections 1.6.2 et 1.6.4, respectivement.

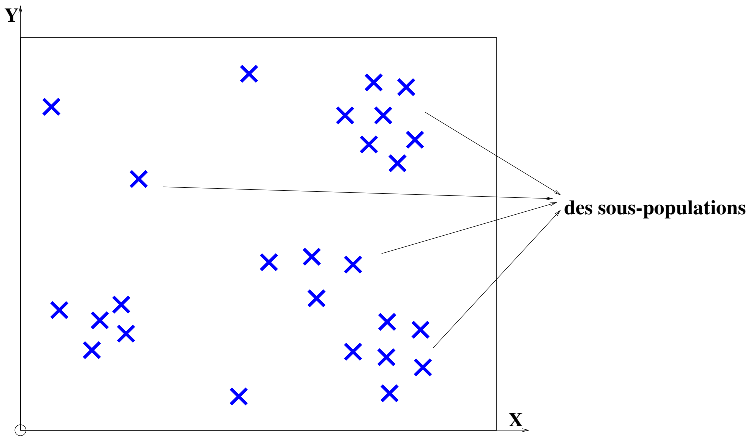


FIG. 1.19 – *Effet du nichage sur une population.*

1.6.1 Le surpeuplement (Crowding)

Initialement proposé par De Jong en 1975 [21], le *crowding* s'applique à un algorithme génétique simple et permet seulement à une fraction de la population de se reproduire à chaque génération. Chaque nouvel individu généré remplace un individu existant qui lui est très proche génotypiquement. Le déroulement de la procédure de *crowding* est le suivant.

Un certain pourcentage G des individus sont sélectionnés proportionnellement à leurs performances pour se reproduire par croisement et mutation. $G \times n$ individus de la population doivent être ensuite remplacés par les nouveaux enfants. Chaque élément inséré dans la population prend la place d'un autre choisi avec la procédure suivante:

1. Un échantillon de CF individus est sélectionné aléatoirement de la population, où CF est appelé le *"facteur de crowding"*.
2. Parmi les CF éléments, l'élément le plus proche de celui à insérer est sélectionné pour être remplacé. Le degré de *similarité* entre deux individus est défini en faisant une comparaison élément par élément de leurs génotypes. Le but de cette comparaison génotypique est d'assurer la descendance dans la zone d'espace du parent remplacé.
3. Ce procédé est ensuite itéré $G \times n$ fois afin de compléter la nouvelle population.

Le *crowding* est inspiré du phénomène écologique où les individus semblables dans une population naturelle, souvent de même espèce, sont en compétition entre eux pour l'acquisition des ressources limitées [79]. Par ailleurs, les individus dissemblables tendent à occuper des domaines distincts dans l'espace d'état, et ne sont donc pas en position de rivalité pour accéder aux ressources.

Le résultat général de cette technique est le maintien de la diversité et l'équilibre de la population, où les nouveaux membres d'une espèce particulière remplacent les anciens membres de cette même espèce, ce qui permet de stabiliser sa taille. Ainsi, le *crowding* permet d'éviter la convergence prématurée. Cependant, la seule diversité possible au cours de l'évolution est la diversité préexistante (de la population initiale), et la chance de création d'une nouvelle espèce est très faible.

1.6.2 Le partage (*sharing*)

La partage *explicite* de la performance a été introduit par Goldberg et Richardson en 1987 [43], où la performance d'un individu donné dépend du nombre d'individus qui lui sont proches dans la population. Son but principal est de pénaliser les individus qui sont trop proches les uns des autres. Ainsi, la performance *partagée* (*shared fitness*) f' d'un individu $\overrightarrow{x_i}$ est donnée par:

$$f'(\overrightarrow{x_i}) = \frac{f(\overrightarrow{x_i})}{\sum_{j=1}^{n} sh(d(\overrightarrow{x_i}, \overrightarrow{x_j}))}$$

où sh désigne la fonction de partage (*sharing function*) et elle a comme paramètre d'entrée la distance d entre deux solutions. Elle retourne '1' si les deux solutions sont identiques, '0' si la distance entre eux dépasse un certain seuil (σ_{share}) et une valeur intermédiaire pour des degrés de dissimilarité intermédiaires. La fonction de partage la plus utilisée est:

$$sh(d) = \begin{cases} 1 - (\frac{d}{\sigma_{share}})^{\alpha}, & \text{si } d < \sigma_{share} \\ 0, & \text{sinon,} \end{cases}$$

où α est une constante (typiquement égale à 1) utilisée pour contrôler l'allure de la fonction de partage. Le calcul de la distance entre deux individus peut se faire dans l'espace génotypique (e.g. distance de Hamming) ou phénotypique (e.g. distance Euclidienne), selon le problème traité. Cependant, des études expérimentales faites par Deb et Goldeberg en 1989 [20] sur les mêmes tests ont montré que le partage dans l'espace phénotypique donne des résultats légèrement meilleurs.

1.6.3 Le partage par classification

La procédure de partage classique a une complexité élevée ($O(N^2)$), vu qu'elle nécessite une comparaison deux à deux de tous les individus de la population. Afin de réduire cette complexité, Yin et Germay, en 1993, ont proposé de classifier la population en niches avant la procédure de partage [135]. Cette dernière est appliquée ensuite sur chaque niche, ce qui réduit la complexité à $O(Nlog(N))$.

Cette méthode a été ensuite rendue adaptative par Alliot [3] qui propose d'actualiser la distance minimale de nichage à chaque génération, de manière à maintenir un bon niveau de performance des individus appartenant aux différentes classes.

1.6.4 L'éclaircissement élitiste

L'éclaircissement est une autre version de nichage proposée par Pétrowski en 1996 [93], où toutes les ressources d'une sous-population sont réservées au meilleur du groupe, au lieu d'être partagées par tous les membres. Comme les méthodes précédentes, les différentes sous-populations sont déterminées grâce à un seuil de dissimilarité σ_{share}. La procédure d'éclaircissement préserve la performance de l'individu dominant du groupe, appelé "le gagnant", et affecte des valeurs nulles pour tout le reste des individus de la sous-population. En considérant le processus d'un point de vue écologique, ce sont les individus les plus forts de chaque groupe qui se partagent la totalité des ressources, et les plus faibles n'ont pas accès au ressources (performance nulle). Les niches des individus dominés ne sont pas connus (de fait qu'il peuvent être dominés par plusieurs gagnants en même temps), mais l'ensembles des gagnants est unique pour une population donnée.

Cette méthode peut être généralisée en acceptant plusieurs gagnants parmi les meilleurs individus d'une niche. Le nombre maximum de gagnants dans une niche défini sa capacité. L'opérateur de sélection est ensuite appliqué sur les survivants.

L'algorithme de la procédure d'éclaircissement se résume comme suit:

trier la population (par ordre décroissant de la performance)
Pour $i = 1$ **jusqu'à** $i = N - 1$ **faire**
Si $fitness(\overrightarrow{x_i})$ >0
nbgagants=1
pour $j = i + 1$ **jusau'à** $j = N - 1$ **faire**
Si $fitness(\overrightarrow{x_j})$>0 et $distance(\overrightarrow{x_i}, \overrightarrow{x_j}) < \sigma_{share}$
Si nbgagnants <Capacité_niche
nbgagants $\leftarrow$ nbgagnants + 1
sinon
$fitness(\overrightarrow{x_j})$=0
Fin

avec :
- $fitness(\overrightarrow{x_i})$: est la performance de l'individu $\overrightarrow{x_i}$,
- $distance(\overrightarrow{x_i}, \overrightarrow{x_j})$: retourne la distance entre les points $\overrightarrow{x_i}$ et $\overrightarrow{x_j}$,
- σ_{share}: le rayon d'éclaircissement,
- Capacité_niche: capacité de chaque niche (nombre maximum de gagnants par niche).

Cette méthode a l'avantage de contrôler la densité des points dans chaque région de l'espace de recherche. En plus, sa complexité est inférieure à celle du partage classique ($O(C.N)$, où C est le nombre des sous-populations).

1.7 L'optimisation multi-critère

L'objet de l'optimisation multi-critère est la maximisation ou la minimisation de plusieurs fonctions objectifs simultanément. Par exemple, dans un procédé de fabrication, on cherche à maximiser un gain tout en minimisant un coût.

$$Minimiser f(\overrightarrow{x}) = (f_1(\overrightarrow{x}), \cdots, f_m(\overrightarrow{x}))$$

où f_i est une fonction objectif et m et le nombre d'objectifs.

Le *front de Pareto* optimal:
C'est l'ensemble des solutions non-dominées dans l'espace de recherche. La notion de domination pour un problème de minimisation est définie comme suit:

Si $\overrightarrow{x_1}$ et $\overrightarrow{x_2}$ sont deux solutions, alors $\overrightarrow{x_1}$ domine $\overrightarrow{x_2}$ ($\overrightarrow{x_2} \prec \overrightarrow{x_1}$) si et seulement si:

1. $\forall j \in \{1, \cdots, m\}: f_j(\overrightarrow{x_2}) \leqslant f_j(\overrightarrow{x_1})$
2. $\exists i \in \{1, \cdots, m\}: f_i(\overrightarrow{x_2}) < f_i(\overrightarrow{x_1})$

Deux buts majeurs sont visés au cours de l'optimisation multi-critère:

1. guider la recherche vers la surface de Pareto globale,
2. maintenir la diversité de la population au niveau du front de Pareto.

Les premières approches proposées tiennent compte principalement du premier objectif. D'autres approches ont été introduites ensuite qui sont plus adaptées à l'optimisation de plusieurs objectifs.

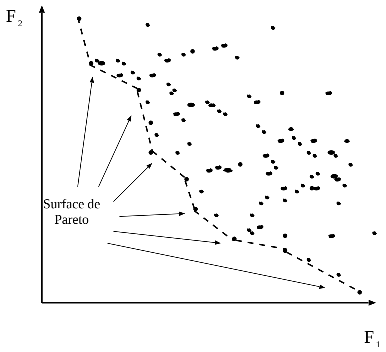


FIG. 1.20 – *Le front de Pareto pour un problème de minimisation à deux objectifs (F_1, F_2). Le graphe est présenté dans l'espace phénotypique.*

1.7.1 Les méthodes classiques

Les méthodes classiques ne sont pas spécifiques aux algorithmes évolutionnaires. La première approche qui a été proposée est l'approche séquentielle qui combine les fonctions objectifs pour se ramener à un problème mono-objectif: $F(\overrightarrow{x}) = \sum_{j=1}^{m} w_j f_j(\overrightarrow{x})$, où w_j est un poids utilisé pour la j^{eme} fonction. Cette fonction est optimisée avec des différents vecteurs de poids afin de trouver plusieurs solutions proches du front de Pareto [42]. Cependant, cette méthode est très coûteuse et ne permet pas de trouver des points des régions non-convexes du front de Pareto.

Une autre approche consiste à optimiser une seule fonction parmi les objectifs et à considérer les autres comme des contraintes: $f_j(\overrightarrow{x}) \leqslant \epsilon_j$. On résout le problème avec différents vecteurs de ϵ_j pour obtenir différentes solutions de Pareto. Toutefois, cette méthode reste coûteuse et dépendante du choix des ϵ_j.

1.7.2 Les méthodes évolutionnaires

Dans le domaine évolutionnaire, le but de l'optimisation multi-objectif est d'échantillonner le *front de Pareto* optimal (figure 1.20). Plusieurs techniques ont été proposées que nous présentons selon l'ordre chronologique de leur apparition.

Vector-Evaluated GA: VEGA
Proposée par Schaffer en 1984 [111], cette méthode repose sur une stratégie de sélection qui prend en compte l'ensemble des objectifs. Pendant chaque itération, la population est divisée en m sous-ensembles, et de chaque sous-population, on sélectionne des individus selon leurs performances pour un seul des objectifs. Cependant, avec une telle sélection, les points se répartissent sur les extrêma du front du Pareto.

Multiobjective Optimization GA: MOGA
Proposée par Fonseca et Fleming en 1993 [37], cette méthode est basée sur le **classement Non dominé**, dont le but est de classer les individus de la population par niveaux de non-dominance. L'algorithme se déroule comme suit:

On parcourt toute la population afin de chercher les solutions non-dominées. Ces dernières déterminent le premier front de Pareto, elles sont alors retirées de la population. Le même procédé est répété jusqu'à ce que tous les individus soient classés (figure 1.21).

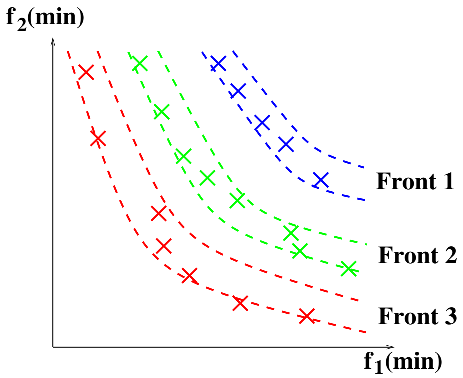


FIG. 1.21 – *Les fronts du classement Non Dominé*

Fonseca et Fleming affectent à chaque individu un rang qui n'est autre que le numéro du front auquel il appartient. La procédure de sélection utilise ces rangs pour sélectionner ou éliminer des blocs de points. Le risque avec cette méthode est la convergence prématurée à cause de la grande pression de sélection.

Niched Pareto GA: NPGA
Proposé par Horn et Nafpliotis en 1993 [61, 60], NPGA utilise le tournoi suivant au cours du processus de sélection: Au début du processus, un nombre spécifique ($tdom$) d'individus sont tirés au hasard de la population, ainsi que deux individus I_1 et I_2 parmi lesquels on va sélectionner un gagnant. I_1 et I_2 sont comparés à tous les éléments de la sous-population. Si l'un est dominé et l'autre est non-dominé, alors le non-dominé est sélectionné. Sinon (les deux sont dominés ou non-dominés), l'individu le plus isolé sur le front de Pareto (ayant l'indice de nichage le plus petit) sera le gagnant.

Nondominated Sorting GA: NSGA
Comme MOGA, l'approche NSGA est basée sur le classement non dominé. Elle affecte à chaque individu une performance selon le front de Pareto auquel il appartient [125]. Ensuite, pour la sélection, elle applique une méthode de nichage par front (qui respecte la hiérarchie entre les fronts) pour maintenir la diversité.

Strength Pareto GA: SPGA
La sélection avec SPGA est basée simultanément sur le concept de non-domination et l'élitisme [137, 138]. A partir de la population initiale, l'algorithme construit une population externe avec les solutions non-dominées. Cette population est mise à jour chaque génération. L'algorithme lui fait appel pendant les opérations de reproduction (figure 1.22). Pour définir les performances des individus, la population externe et la population courante sont combinées et les solutions non-dominées auront une performance qui dépend du nombre de points qu'elles dominent.

Ziztzler et *al.*, dans une étude comparative faite dans [136] qui teste les différentes techniques d'optimisation multi-objectif sur six problèmes de référence, ont établi un classement de

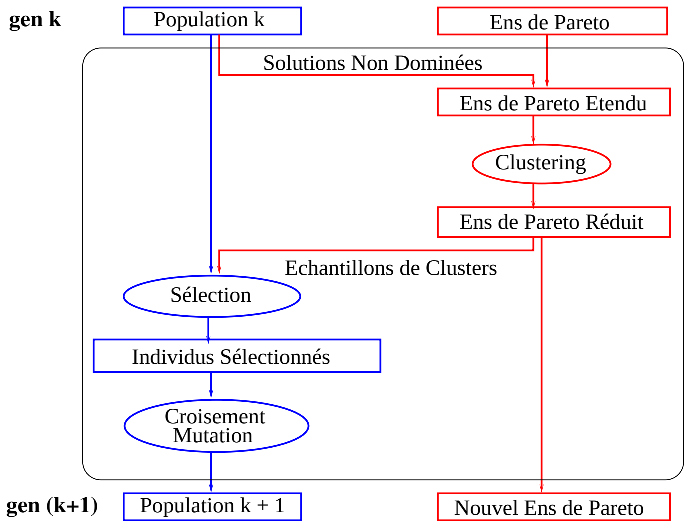


FIG. 1.22 – *L'algorithme de SPGA*

ces méthodes selon un ordre de mérite décroissant:

1. SPGA
2. NSGA
3. VEGA
4. MOGA

1.8 Applicabilité des EA

Pour résoudre un problème d'optimisation réel, on a le choix entre deux approches: l'approche déterministe et l'approche stochastique. Si l'approche déterministe (e.g. descente de gradient) est applicable (fonctions assez régulières), elle robuste quand on est en présence d'un seul optimum. Mais si le problème a plusieurs optima locaux et on cherche à localiser l'optimum global, cette approche est très souvent incapable d'explorer efficacement l'espace de recherche et converge rapidement vers l'optimum local le plus proche. Dans ce cas, il faut essayer une méthode stochastique comme le recuit simulé ou les algorithmes évolutionnaires.

L'utilisation des méthodes évolutionnaires dans la résolution de certains problèmes réels est récente, mais elle a été appliquée avec succès dans de nombreux domaines, comme l'optimisation de formes [46, 105, 65], la planification [2, 118, 45], la classification [50, 72, 80], le traitement d'images [17, 40, 77], ...

Un des cas où l'utilisation des EA est envisageable est le cas des **problèmes inverses** mal posés, souvent rencontrés dans le domaine de la physique, chimie, mécanique, ... Dans ces domaines, la résolution d'un **problème direct** consiste à mettre en place un processus de simulation des résultats expérimentaux du phénomène étudié, à partir d'un ensemble de données. Cette simulation repose sur une fonction ou un algorithme qui modélise le comportement du phénomène selon les lois physiques, chimiques ...

Le problème inverse consiste, partant de conditions expérimentales et du comportement du phénomène physique observé dans ces conditions, à déterminer une loi modélisant le phénomène physique de façon satisfaisante. La performance d'une loi est donnée par la différence entre le comportement simulé, obtenu par la résolution du problème direct pour cette loi avec les conditions expérimentales données, et le comportement réellement observé avec les mêmes conditions expérimentales. La figure 1.23 indique les étapes du calcul de la performance.

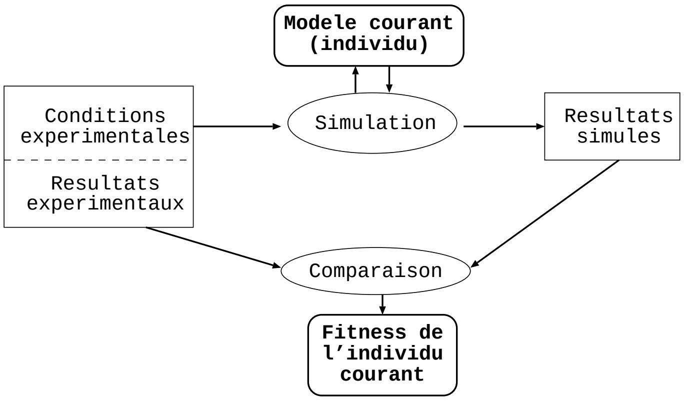


FIG. 1.23 – *Le problème inverse*

Fréquemment, les problèmes inverses ne peuvent pas être traités par les approches classiques faute de disposer d'une fonction performance assez régulière et dont les dérivées soient connues. Étant donné que l'optimisation évolutionnaire requiert uniquement que la fonction performance soit calculable, elle est applicable à tout problème inverse correspondant à un problème direct numériquement résolu.

Dans la deuxième partie de ce travail, nous présentons une application réelle dans le domaine de la physique, modélisée par un problème inverse. Le but de cette application est d'optimiser la forme d'une lame de phases d'un système laser dédiée à la fusion nucléaire, en vue de minimiser la perte d'énergie.

Première partie

Algorithmes Evolutionnaires: Prise en compte des contraintes

Chapitre 2

État de l'art

Dans ce chapitre, nous présentons l'optimisation sous contraintes ainsi que les méthodes les plus connues, classiques et évolutionnaires, qui traitent ce problème. La présentation des méthodes évolutionnaires est accompagnée par des exemples de résultats commentés et discutés.

2.1 Introduction

Les techniques de calcul évolutionnaire deviennent de plus en plus populaires, vu leur grand potentiel dans la résolution des problèmes d'optimisation complexes, souvent impossibles à traiter avec les méthodes d'optimisation classiques. Les capacités, parfois étonnantes, des algorithmes génétiques à résoudre certains problèmes numériques difficiles, ont incité les ingénieurs dans différents domaines scientifiques (physique, mécanique, chimie, biologie, ...) à adopter les techniques évolutionnaires dans la résolution de leurs problèmes d'optimisation. Or, les problèmes posés dans le monde réel sont souvent soumis à des contraintes techniques et/ou des contraintes de l'environnement (exemple: minimisation du coût de fabrication d'un objet tout en respectant les contraintes techniques et écologiques). De ce fait, pendant la dernière décennie, plusieurs stratégies ont été proposées pour prendre en compte les contraintes avec les algorithmes évolutionnaires. Ce chapitre présente et analyse les méthodes les plus connues.

Les contraintes sont habituellement exprimées sous formes d'équations et d'inéquations. Le problème général est alors formulé comme suit:

$$\text{optimiser } f(\overrightarrow{x}),\ \overrightarrow{x} = (x_1, \ldots, x_n) \in \mathcal{F} \subseteq \mathcal{S} \subseteq \mathbb{R}^n,$$

tels que:

$$\begin{cases} g_i(\overrightarrow{x}) \leqslant 0, & \text{pour } i = 1, \ldots, q \text{ (contraintes d'inégalité)} \\ h_j(\overrightarrow{x}) = 0, & \text{pour } j = q+1, \ldots, m \text{ (contraintes d'égalité)} \end{cases}$$

où f, g_i et h_j sont des fonctions réelles dans $\mathbb{R}^n$.

Dans tout ce qui suit, nous ne considérons que des problèmes de minimisation. Pour la maximisation, la correspondance se fait en inversant le critère $f(\overrightarrow{x})$.

L'ensemble $\mathcal{S} \subseteq \mathbb{R}^n$ est l'espace de recherche déterminé par les bornes supérieures et inférieures des domaines des variables:

$$l(i) \leqslant x_i \leqslant u(i) \quad \text{pour } 1 \leqslant i \leqslant n$$

La satisfaction de l'ensemble des contraintes (g_j, $(j = 1 \cdots q)$, h_j, $(j = q+1 \cdots m)$) définie l'espace des solutions admissibles, dit le domaine faisable $\mathcal{F}$. Toute solution appartenant à $\mathcal{F}$ est une solution faisable, sinon, elle est infaisable. La recherche de l'optimum faisable est d'autant plus difficile que la taille de l'espace faisable est petite et sa forme est complexe (e.g. des petites régions dispersées). Le rapport entre la taille de l'espace faisable et celle de l'espace de recherche $|\mathcal{F}|/|\mathcal{S}|$ peut être utilisé comme un indice de difficulté du problème [82].

Entre autres, la recherche de l'optimum est souvent plus facile quand il est à l'intérieur du domaine faisable que quand il est sur sa frontière. Ce dernier cas se présente si une (ou plusieurs) contrainte du problème est active au niveau de la solution globale. C'est souvent le cas pour les problèmes réels.

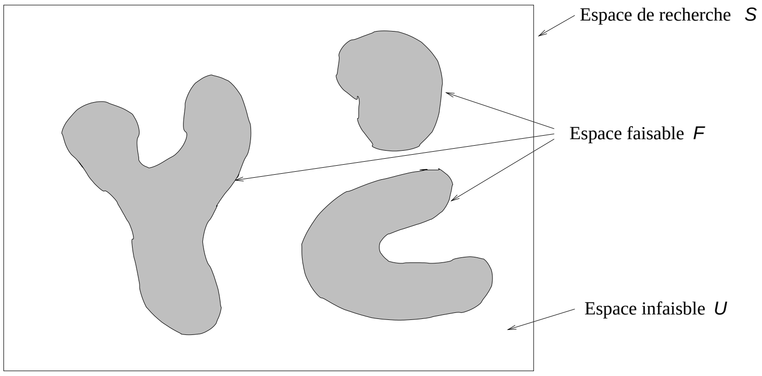


FIG. 2.1 – *L'espace de recherche et ses parties faisables et infaisables*

Depuis les années 50, plusieurs techniques ont été mises en place pour résoudre le problème d'optimisation sous contraintes. Ces méthodes se classent en deux catégories: les méthodes déterministes et les méthodes stochastiques. Les méthodes déterministes se caractérisent par une exploration déterministe de l'espace d'état. Elle permettent de situer la direction de l'optimum par rapport à chacun des points de cet espace, et de diminuer, de proche en proche, la distance entre le point courant et l'optimum. Cependant, elles sont toutes applicables sous des conditions strictes se rapportant à la convexité, la dérivabilité et la linéarité de la fonction objectif et contraintes.

Lorsqu'aucune de ces méthodes ne peut être envisagée, reste le recours aux méthodes stochastiques, parmi les quelles les algorithmes évolutionnaires.

Il n'y a pas une méthode évolutionnaire standard pour déterminer l'optimum global d'un problème contraint. La question qui se pose est "comment traiter les individus infaisables?". Deux catégories se sont formées en répondant à cette question: une qui ne considère que les individus faisables, et donc la fonction d'adaptation n'est évaluée que dans le domaine faisable, et la deuxième considèrent tous les individus dans l'espace de recherche, mais définie une fonction d'évaluation spéciale pour les individus infaisables. Le choix de la bonne catégorie dépend de la nature du problème (e.g. si la foncion objectif est définie dans le domaine infaisable ou non). Plusieurs stratégies ont été proposées dans les deux catégories.

Ce chapitre présente les principales méthodes de prise en compte des contraintes dans la littérature. La première section donne un bref rappel des méthodes déterministes les plus connues. Le reste du chapitre est consacré au méthodes évolutionnaires. Ces dernières sont classées en quatre catégories:

- méthodes basées sur les fonctions de pénalité,
- méthodes basées sur la recherche des solutions faisables,
- méthodes basées sur la préservation de la faisabilité des solutions,
- méthodes hybrides.

Les plus utilisées sont celles basées sur les fonctions de pénalité, vue la facilité de leur implantation. Ces méthodes donnent souvent des bons résultats, en particulier si l'optimum n'est pas sur la frontière du domaine faisable. Si elles échouent, le recours à des méthodes plus sophistiquées s'impose.

2.2 Méthodes déterministes

Considérons le problème d'optimisation sans contraintes:

$$P_x : \text{ minimiser } f(\overrightarrow{x}), \quad \overrightarrow{x} \in \mathbb{R}^n$$

où $f : \mathbb{R}^n \to \mathbb{R}$ est supposée régulière.
Le principe des méthodes déterministes est fondé sur la notion de direction de descente. On dit que d est une **direction de descente** de f en $\overrightarrow{x} \in \mathbb{R}^n$ si $f'(\overrightarrow{x}).d < 0$. Par définition de la dérivée, cette condition revient à:

$$f(\overrightarrow{x} + u.d) < f(\overrightarrow{x}), \ \ \forall u > 0 \text{ suffisamment petit}$$

Les méthodes à directions de descente utilisent cette idée pour minimiser une fonction. Elles construisent la suite des itérés $\{\overrightarrow{x}_k\}_{k \geqslant 1}$ approchant une solution $\overrightarrow{x}^*$ par la récurrence

$$\overrightarrow{x}_{k+1} = \overrightarrow{x}_k + u_k d_k, \text{ pour } k \geqslant 1,$$

où u_k est appelé le *pas* et d_k est une direction de descente de f en $\overrightarrow{x}_k$. L'algorithme général des méthodes à direction de descente est comme suit:

1. Choix d'un itéré initial $x_1 \in \mathbb{R}^n$;
   Initialisation: $k = 1$;
2. Test d'arrêt: Si $\nabla f(x) \simeq 0$, arrêt de l'algorithme;
3. Choix d'une direction de descente d_k;
4. *Recherche linéaire:* déterminer un pas $\alpha_k > 0$ le long de d_k de manière à "faire décroître f suffisamment";
5. $x_{k+1} = x_k + \alpha_k d_k$;
   Accroître k et aller en 1.

Dans les problèmes sans contraintes, le test d'arrêt de l'algorithme porte sur la petitesse du gradient: $\nabla f(\overrightarrow{x}_k) \simeq 0$. En présence des contraintes, cette condition devient insuffisante. En effet, l'optimum retourné doit être admissible (appartenant à l'espace faisable). Il est donc nécessaire d'introduire des conditions sur la satisfaction des contraintes du problème. Dans ce cas, ce sont les conditions d'optimalité de Karush, Kuhn et Tucker (KKT) basées sur les **multiplicateurs de Lagrange** qui sont utilisées [51, 74].

Soit c l'ensemble des contraintes du problèmes: $c = \{g_i \ (i = 0, \cdots, q), h_i(i = q+1, \cdots, m)\}$. Si f et g_i sont convexes et h_i sont affines, alors il existe $\lambda^* \in \mathbb{R}^m$ tel que l'on ait:

$$(KKT) \begin{cases} \nabla \mathcal{L}(\overrightarrow{x}^*, \lambda^*) = 0, \\ h_j(\overrightarrow{x}^*) = 0 \\ g_j(\overrightarrow{x}^*) \leqslant 0 \\ (\lambda^*)_j \geqslant 0 \\ (\lambda^*)_j . c_j(\overrightarrow{x}^*) = 0 \end{cases}$$

où $\mathcal{L}(\overrightarrow{x}, \lambda)$ est le **lagrangien** associé au problème défini par:

$$\mathcal{L}(\overrightarrow{x}, \lambda) = f(\overrightarrow{x}) + \lambda^T c(\overrightarrow{x}).$$

Le vecteur λ a autant de composantes qu'il y a de contraintes, chaque composante de λ multipliant la contrainte à laquelle elle est associée dans le *lagrangien*. Si la contrainte est non active au niveau de $\overrightarrow{x}^*$, alors le multiplicateur correspondant $(\lambda^*)_j$ est nul.

Outre la condition d'optimalité, les algorithmes de descente s'affrontent à une autre difficulté au cours de processus itératif. A chaque itération, la direction de descente doit être admissible. Plusieurs solutions ont été proposées pour prendre en compte les contraintes avec les algorithmes à direction de descente, dont nous présentons les plus connues.

2.2.1 Méthodes basées sur le calcul du gradient

C'est un algorithme itératif qui permet de se déplacer successivement dans les directions de plus grandes pentes. Partant d'un point quelconque de l'espace de recherche, on commence par calculer la valeur du gradient dans son voisinage immédiat, et on effectue un déplacement dans la direction où le gradient est maximum. Le but de cet algorithme est d'atteindre un point où le gradient s'annule.

Le passage de $\overrightarrow{x}_k$ à $\overrightarrow{x}_{k+1}$ s'effectue de la façon suivante: $\overrightarrow{x}_{k+1} = \overrightarrow{x}_k - u.f'(\overrightarrow{x}_k)$ et $u > 0$ paramètre de l'algorithme, avec: $f'(\overrightarrow{x}_k)$ gradient de la fonction f en $\overrightarrow{x}_k$. Ainsi, $f(\overrightarrow{x}_{k+1}) = f(\overrightarrow{x}_k) - u.||f(\overrightarrow{x}_k)||^2 + o(u||f'(\overrightarrow{x}_k)||)$ qui implique la décroissance de f pour u suffisamment petit. La méthode est souvent modifiée afin d'autoriser u à varier à chaque itération.

Plusieurs variantes ont été mises en place afin de prendre en compte les contraintes. Par exemple, dans le cas où la fonction objectif f est non linéaire soumise à des contraintes linéaires, la méthode du gradient réduit de Wolfe [28] peut s'appliquer. Par contre, si f est convexe avec des contraintes linéaires, on applique plutôt la méthode du gradient projeté de Rosen [103]. Cette méthode part d'un point de la frontière et passe, à chaque étape, par la projection du gradient sur un sous-domaine de $\mathcal{S}$ défini par les contraintes actives au niveau du point courant.

Ces deux méthodes ont été généralisées afin de traiter le cas de f convexe dans un domaine convexe fermé, qui ont donné, respectivement, la méthode du gradient projeté généralisée [104] et la méthode de gradient réduit généralisée [1].

2.2.2 La méthode de Newton

Lorsque les dérivées secondes de la fonction objectif sont calculables, la méthode de Newton peut être envisagée. Son principe s'approche de celui des méthodes de gradient, à part qu'elle utilise, en plus du gradient, le Hessien de f assorti de la condition d'admissibilité des contraintes pour calculer la direction de recherche. Sa convergence est plus rapide que celle des méthodes de gradient. Ainsi, le minimum non contraint d'une forme quadratique définie positive est obtenu en une seule itération, ce qui laisse préjuger de son efficacité lorsqu'elle est appliquée à une fonction tout à fait générale.

La structure de cet algorithme est la même que celle de la méthode des gradients, à l'exception de la direction de recherche qui est calculée de la façon suivante: $d_k = -(f''(\overrightarrow{x}_N))^{-1}.f'(\overrightarrow{x}_k)$, avec $f''(\overrightarrow{x}_k)$ est le Hessien de f en $\overrightarrow{x}_k$. Le point obtenu à chaque itération est le minimum de l'approximation quadratique de f. Cependant, celui-ci ne correspond pas nécessairement au minimum de f dans la direction de Newton. Le pas de progression le long de celle-ci doit être déterminé par une procédure de recherche linéaire (méthode de Newton généralisée). En plus, certaines difficultés peuvent apparaître lorsque f n'est pas strictement convexe ou que le Hessien n'est pas inversible en certains points. Pour éviter ces inconvénients, différents remèdes ont été proposés, comme par exemple le procédé de Levenberg-Marquardt destiné à forcer le caractère défini positif du Hessien en ajoutant des termes positifs à sa diagonale [7].

Comme pour la méthode du gradient, l'adaptation de la méthode de Newton aux problèmes à contraintes linéaires demande l'établissement d'une matrice de projection appropriée P. La direction de recherche devient alors: $d_k = -P.H^{-1}.f'(\overrightarrow{x}_k)$, où H est la matrice Hessienne. En général, P projette non orthogonalement tout vecteur dans le plan tangent aux contraintes actives.

2.2.3 Les méthodes à métriques variables

La nécessité d'inverser la matrice Hessienne à chaque itération est le principal inconvénient de la méthode de Newton. Cette objection est la base d'algorithmes de minimisation dans lesquels le Hessien, ou son inverse, est approximé sur la base d'informations du premier ordre accumulées au cours du processus itératif (méthodes de quasi-Newton [7]). L'approximation est faite par une suite de matrices définies positives M_k, dont la limite à l'optimum est le Hessien en ce point. La descente d_k est alors définie par: $d_k = -M^{-1}{}_k.f'(\overrightarrow{x}_k)$.

Pour une forme quadratique définie positive de dimension N, la méthode converge en N+1 itérations. Les formules d'approximation de la Hessienne tiennent compte des contraintes linéaires du problème. Le principe du calcul de la direction de recherche est le même que celui de la méthode de Newton, excepté que la matrice Hessienne est remplacée par son approximation, et la matrice de projection est établie sur la base de cette approximation. Une autre stratégie pour prendre en compte les contraintes linéaires consiste à mélanger les formules d'approximation et la méthode du gradient projeté. Cela conduit à adopter les directions de recherche: $d_k = -S_k.f'(x_k)$, tant que la base des contraintes actives n'a pas changé. S_k est une approximation de $P.H^{-1}$ utilisée dans la méthode de Newton.

Les algorithmes à métriques variables les plus utilisés sont BFGS (Broyden Fletcher Goldfarb Shanno) [31] ou sa variante LM-BFGS pour les problèmes de grande taille. Ils calculent l'approximation de la matrice Hessienne à partir des directions de recherche conjuguées satisfaisant la règle de Wolfe [133].

2.2.4 La méthode des directions admissibles

Appelée aussi méthode SQP (*Programmation Quadratique Successive*)[123], la méthode des directions admissibles est une variante de la BFGS. Elle utilise le cône de directions admissibles en un point pour rediriger les directions de recherche produites par BFGS. Elle procède en deux étapes:

– détermination d'une direction de recherche admissible;

- calcul d'un pas de progression minimisant la fonction objectif le long de la direction sélectionnée.

La détermination de la direction de recherche se base sur le concept des directions admissibles introduit par Zoutendijk [139]. Selon ce principe, chaque étape de la minimisation consiste en une recherche linéaire le long d'une direction ne quittant pas immédiatement le domaine admissible. Le vecteur $\overrightarrow{s}$ vérifiant cette condition constitue une direction admissible.

Ayant déterminer une direction de descente, il reste à fixer le pas de progression à effectuer le long de cette direction. Deux éventualités sont à envisager:

- si aucune contrainte n'est violée, il faudra déterminer le pas de progression maximum qui minimise la foncion objectif ou pour lequel la direction de recherche rencontre une nouvelle contrainte ou une contrainte déjà active.
- si une ou plusieurs contraintes sont violées, on cherchera le pas qui minimise la contrainte la plus violée ou qui minimise la foncion objectif si la conception correspondante est admissible.

2.2.5 Méthodes de pénalité

Elles s'appliquent quand f est continue de $\Omega_0 \to \mathbb{R}$, où $\Omega \in \mathbb{R}$ est un domaine ouvert. Elles remplacent la recherche directe de l'extremum avec contraintes par celle d'une succession d'extrema sans contraintes. Le problème P_x est alors transformé en:

$$P_r : \text{ minimiser } A(\overrightarrow{x}, r) = f(\overrightarrow{x}) + r.\pi(x)$$

On cite la méthode de pénalité à points intérieurs de Fiacco et Cormick [30] et celle de Courant [15] pour les problèmes avec contraintes d'inégalité, et la méthode de pénalité à points extérieurs qui traite les contraintes d'égalité et d'inégalité. Ces méthodes ont été combinées afin de bénéficier de leurs avantages respectifs, et ont donné naissance à la méthode mixte [29].

La question qui se pose avec les méthodes de pénalité est de savoir si en résolvant P_r, on résout P_x. Cette question conduit à la notion de pénalisation *exacte*, où toute solution de P_r est une solution de P_x. La méthode des multiplicateurs de Lagrange [101] est basée sur cette notion, où la résolution du problème P_x passe par la résolution de son problème dual.

La méthode de pénalité à points intérieurs (la fonction de barrière)

L'approche par fonction de barrière consiste à minimiser une fonction auxiliaire dont le minimum non contraint est situé dans le domaine admissible. Celle-ci est construite pour former une barrière à la frontière du domaine afin de se prémunir d'une violation des contraintes. La fonction auxiliaire est écrite sous la forme:

$$A(x, r) = f(\overrightarrow{x}) + rB(\overrightarrow{x}), \text{ avec } r > 0,$$

où la fonction de barrière $B(\overrightarrow{x})$ est positive à l'intérieur du domaine admissible et tend vers l'infini lorsque la conception s'approche de sa frontière. Différents types de fonctions $B(\overrightarrow{x})$ peuvent être envisagées, comme la fonction logarithmique ou la fonction inverse.

En réduisant progressivement l'action des contraintes par la modification du paramètre de contrôle r, une séquence de problèmes sans contraintes est générée. Leur solutions forment une suite convergeant vers la solution du problème réel.

La méthode de pénalité à points extérieurs

Contrairement à la méthode à points intérieurs, la méthode de pénalité à points extérieurs pénalise fortement la violation des contraintes:

$$A(\overrightarrow{x}, r) = f(\overrightarrow{x}) + 1/rP(\overrightarrow{x}) \text{ avec } r > 0$$

où $P(\overrightarrow{x})$ est positif à l'extérieur du domaine admissible et nul à l'intérieur. De la sorte, la fonction auxiliaire tend vers l'infini lorsque la violation des contraintes augmente.

Cette méthode diffère de la précédente en 3 points:

- Le point de départ ne doit plus être nécessairement admissible.
- Les contraintes d'égalité peuvent être prises en compte.
- L'algorithme génère une séquence de conceptions non admissibles, approchant l'optimum par l'extérieur du domaine admissible.

Les deux méthodes ont été combinées pour donner la méthode à points intérieurs étendue dans laquelle la fonction auxiliaire est définie partout dans l'espace de recherche et la séquence de points générée est admissible.

Pénalisation exacte: Le *Lagrangien*

Si le problème est strictement convexe et les conditions du (KKT) sont satisfaites en $\overrightarrow{x}^*$, alors la fonction $\mathcal{L}(\overrightarrow{x}, \lambda)$ admet un minimum global $\overrightarrow{x}$ quelque soit λ fixé. Il existe donc une correspondance unique entre $\overrightarrow{x}$ et λ, donnée par le problème non contraint. Pour cette raison, le problème dual (minimiser $\mathcal{L}(\overrightarrow{x}, \lambda^*)$ pour λ^* donné avec $c_j(\overrightarrow{x}) = 0$ $j = 1, \cdots, m$) est équivalent au problème initial.

$$\mathcal{L}(\overrightarrow{x}, \lambda) = f(\overrightarrow{x}) - \sum_{j=1}^{q} \lambda_j c_j(\overrightarrow{x})$$

Ainsi, $\mathcal{L}(\overrightarrow{x}, \lambda)$ est une fonction de pénalisation exacte pour les problèmes convexes [100].

Pénalisation exacte: Le *Lagrangien augmenté*

Le *Lagrangien* n'est pas une fonction de pénalisation exacte si le problème n'est pas convexe. Pour remédier à cet inconvénient, on peut considère la résolution du problème *lagrangien* par une méthode de pénalité. Le terme de pénalité est ajouté, non pas à la fonction objectif $f(\overrightarrow{x})$, mais à la fonction $\mathcal{L}(\overrightarrow{x}, \lambda)$. On obtient ainsi la fonction de **Lagrangien augmentée** [101] du problème à contrainte:

$$A(\overrightarrow{x}, \lambda, r) = f(\overrightarrow{x}) - \sum_{j=1}^{q} \lambda_j h_j(\overrightarrow{x}) + r/2 \sum_{j=1}^{q} ||h_j(\overrightarrow{x})||^2 + \sum_{j=1}^{m} [\lambda_j max(\frac{-\lambda_j}{r}, g_j(\overrightarrow{x})) + r/2(max(\frac{-\lambda_j}{r}, g_j(\overrightarrow{x})))^2]$$

Pour des valeurs données de λ et r, la méthode des multiplicateurs consiste à appliquer un algorithme de minimisation sans contraintes à la fonction $A(\overrightarrow{x}, \lambda, r)$. Un point $\overrightarrow{x}(\lambda, r)$ est ainsi obtenu. Les valeurs de λ et r sont alors actualisées et une nouvelle itération est effectuée.

Si λ est nul, la méthode se ramène aux méthodes classiques à fonctions de pénalité. Par contre, si le vecteur λ est le vecteur des multiplicateurs optimaux λ^*, la solution du problème initial est obtenue quelle que soit la valeur positive de r.

2.2.6 La programmation linéaire

Un programme linéaire est une technique d'optimisation d'une fonction linéaire, composée de n variables de décision, tout en respectant un ensemble de contraintes linéaires prenant la forme d'équations ou d'inéquations. La programmation linéaire est la branche des mathématiques consacrée à l'étude des problèmes linéaires et tout particulièrement, aux conditions d'existence de solutions optimales. Elle a été généralisée pour prendre en compte des problèmes non linéaires, en les remplaçant par des suites de sous-problèmes obtenus en linéarisant la fonction objectif et les contraintes. Cependant, cette linéarisation est souvent très difficile à réaliser pour les problèmes réels.

En résumé, trois dates importantes jalonnent l'histoire de la programmation linéaire. La première, 1947, correspond à la découverte par G. Dantzig [18] de l'algorithme du simplexe, actuellement la méthode la plus utilisée pour résoudre des programmes linéaires. Il a fallut ensuite attendre plus de vingt ans pour qu'en 1979 Khachian [68] présente le premier algorithme polynômial de programmation linéaire. Sa méthode, basée sur le principe des ellipsoïdes offre principalement un intérêt théorique sans réelle mise en oeuvre efficace.

Cinq ans plus tard, en 1984, Karmarkar proposa également un algorithme polynômial de programmation linéaire [67] basé sur des techniques de points intérieurs qui avaient été initialement développées pour des problèmes non linéaires. Les algorithmes de points intérieurs disponibles aujourd'hui offrent des alternatives concurrentielles, parfois même supérieures aux algorithmes basés sur la méthode du simplexe. Une bonne présentation de ces méthodes est fournie dans [131].

2.2.7 Méthodes de Séparation - *Branch and bound*

Elles consistent à découper l'espace initial de recherche en domaines de plus en plus restreints afin d'isoler l'optimum global. Une borne inf (ou sup) du critère est associée à chacun des domaines afin de guider la recherche sans exploration exhaustive de ces derniers. L'algorithme crée un arbre de recherche dont chacun des noeuds représente un sous domaine. Celui-ci est alors évalué par son minorant. Si ce minorant est inférieur à l'évaluation d'une solution déjà atteinte, la recherche à partir du noeud est arrêtée. Cet arbre évolue au gré de la recherche et l'algorithme consiste à parcourir dynamiquement ce dernier jusqu'à l'obtention de l'optimum global.

L'efficacité de la procédure dépend essentiellement du choix du principe de séparation et d'évaluation. Plus ce choix sera judicieux, moins on aura à construire de branches dans l'arborescence (la construction de l'arborescence complète équivaudrait à examiner toutes les solutions possibles). Le Branch and Bound s'applique essentiellement à des problèmes discrets. Elle reste limité à des espaces de faibles dimensions pour lesquels le temps d'exploration, même partiel, reste raisonnable.

2.2.8 Discussion

Les méthodes déterministes présentées dans cette section ne constituent qu'une partie de l'ensemble des méthodes de la littérature. Nous ne présentons pas les autres méthodes, afin de ne pas nous éloigner du coeur de ce travail. L'objectif de cette section est de montrer que la prise en compte des contraintes constitue la difficulté majeure pour les méthodes d'optimisation. Des solutions efficaces ont été mises en place, mais sous des conditions fermes sur les caractéristiques de la fonction objectif et les fonctions contraintes. Si elles ne sont pas vérifiées, elles ne peuvent pas être appliquées.

Entre autres, la majorité des méthodes déterministes sont adaptées pour résoudre des problèmes de taille réduite dans lesquels il n'y a qu'un seul optimum. Pour des problèmes multi-modaux de grande taille, ce sont les techniques d'optimisation stochastiques comme les algorithmes évolutionnaires qu'il faut envisager.

2.3 Méthodes évolutionnaires: Pénalisation

La majorité des méthodes de prise en compte des contraintes sont basées sur le concept de fonctions de pénalité, qui pénalisent les individus violant les contraintes dans la population. Le problème initial avec contraintes est alors transformé en un problème sans contraintes:

$$\begin{array}{c} \text{minimiser } f(\overrightarrow{x}) + penal(\overrightarrow{x}) \\ penal(\overrightarrow{x}) = 0 \text{ ssi } \overrightarrow{x} \in \mathcal{F} \end{array} \tag{2.1}$$

La difficulté majeure dans cette catégorie de méthodes et la définition de la fonction de pénalité $penal(\overrightarrow{x})$. Plusieurs techniques avec différentes approches et différents heuristiques ont été mises en place. Les plus populaires utilisent la mesure de violation de contraintes pour définir la pénalité:

$$penal(\overrightarrow{x}) = F(\sum_{j=1}^{m} \alpha_j v_j^{\beta}(\overrightarrow{x}))\,, \tag{2.2}$$

où α_j sont les cœfficients de pénalisation, β est un paramètre (souvent égale à 2), et $v_j(\overrightarrow{x})$ est la mesure du degré de violation de la $j^{\text{ème}}$ contrainte du problème posé, calculée comme suit:

$$v_j(\overrightarrow{x}) = \begin{cases} max(0, g_j(\overrightarrow{x})), & \textit{pour } 1 \leqslant j \leqslant q \;\; \textit{(les contraintes d'inégalité)} \\ |h_j(\overrightarrow{x})|, & \textit{pour } q+1 \leqslant j \leqslant m \;\; \textit{(les contraintes d'égalité)} \end{cases} \tag{2.3}$$

Pour les problèmes de type CSP (*"Constrained Satisfactory Problem"*), les contraintes sont des fonctions booléennes, et la satisfaction d'une contrainte est vérifiée si sa fonction est égale à 1. Dans ce cas, le taux de violation, est calculé avec le nombre de fonctions à 0 (contraintes non satisfaites). Nous ne nous intéresserons pas à ce type de problème dans notre travail, car nous ne traiterons que les problèmes numériques.

L'utilisation de la mesure de violation des contraintes dans la pénalisation aide à différencier les individus infaisables. En effet, ceux qui ont un taux de violation très élevé sont pénalisés plus que ceux qui ont des faibles violations, ce qui permet de les forcer vers l'espace réalisable. Cependant, cette mesure est généralement insuffisante pour définir la fonction de violation, surtout si l'écart entre les valeurs de la fonction objectif et les taux de violation est très élevé (e.g. fonction objectif de l'ordre de 10^5 et les taux de violation de l'ordre de 10^{-1}). Des cœfficients de pénalisation sont alors utilisés pour régler la quantité à ajouter à la fonction objectif. La définition de ces cœfficients pour un problème donné est une mission très délicate. Avec un taux faible, l'algorithme risque de retourner une solution violant les contraintes, si la performance des infaisables est meilleure que celle des faisables, même avec la pénalisation. Par ailleurs, une pénalisation trop forte interdit tout passage par l'espace infaisable et augmente les risques de la convergence prématurée, spécialement si $\mathcal{F}$ est non convexe ou s'il est disjoint.

Prenons un exemple simple, où on cherche à minimiser $f(\overrightarrow{x}) = 0.5sin(x) + 5$, avec $0 \leqslant x \leqslant 10$. Le problème est soumis à une seule contrainte très simple: $x \leqslant 3.5$. L'optimum de ce problème est $\overrightarrow{x^*} = 3.5$. La fonction de pénalisation est définie comme suit:

$$penal(\overrightarrow{x}) = p * |x - 3.5| \text{ si } x > 3.5$$

Malgré la simplicité du problème, un algorithme évolutionnaire risque d'échouer dans sa résolution si la valeur de p n'est pas adéquate. La figure 2.2 illustre la courbe de la fonction objectif $f(\overrightarrow{x})$ et celle de la fonction de performance $f(\overrightarrow{x}) + penal(\overrightarrow{x})$ dans le domaine infaisable, avec $p = 0.1$. On y voit clairement que le minimum pour la fonction de performance est $\overrightarrow{x^*} = 4.5$, qui est une solution infaisable. C'est la solution retournée par l'algorithme après l'évolution. Un cœfficient de pénalité de 0.1 est donc trop faible pour garantir la faisabilité des solutions trouvées.

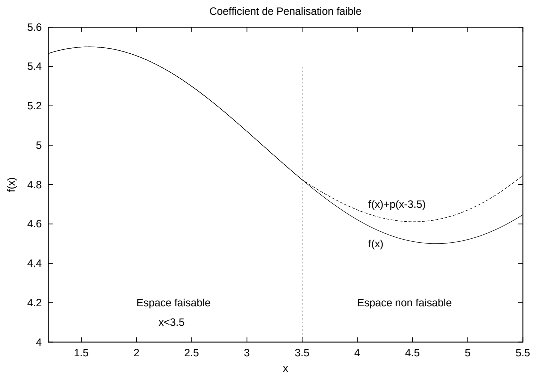


FIG. 2.2 – *Courbes de la fonction objectif* $f(x) = 0.5sin(x) + 5$ *et de la fonction de performance* $f(x) + p * |x - 3.5|$, *avec p=0.1.*

Pour remédier à ce problème, il faut utiliser un cœfficient de pénalisation plus élevé. Par contre, une valeur très grande résulte en un rejet brusque de tous les infaisables de la population dès le début de l'évolution. Dans ce cas, l'exploration est pénalisée, puisqu'on interdit tout passage par l'espace infaisable. La figure 2.3 illustre la courbe de $f(\overrightarrow{x})$ et celle de $f(\overrightarrow{x}) + penal(\overrightarrow{x})$ dans le domaine infaisable, avec $p = 1.7$. La pente de la courbe de la fonction de performance pour $x \geqslant 3.5$ est très élevée, ce qui induit un rejet immédiat des solutions localisées dans cette partie de l'espace.

Étant donné que l'optimum est sur la frontière, l'algorithme aura des difficultés à s'approcher des bords. Il y a toujours plus de chance de localiser l'optimum quand on explore les alentours de la frontière des deux côtés que d'un seul côté.

Entre autres, si l'espace faisable est non convexe ou disjoint, la présence des infaisables permet à l'algorithme de passer d'une partie à une autre de $\mathcal{F}$, et assurer ainsi l'exploration de tout le domaine réalisable.

Le choix de la pénalisation doit prendre en compte les propriétés topologiques de l'espace faisable, le nombre de contraintes actives au niveau de l'optimum, le rapport entre la taille de $\mathcal{F}$ et celle de $\mathcal{S}$, ainsi que le type de la fonction objectif et des fonctions des contraintes.

On distingue trois approches pour définir la fonction de pénalité, l'approche statique, l'approche dynamique et l'approche adaptative. Avec l'approche statique, les cœfficients de pénalité sont fixés

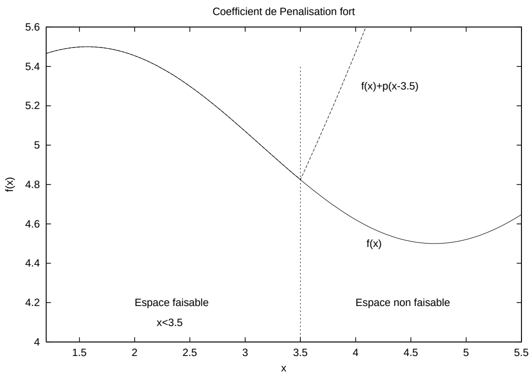


FIG. 2.3 – *Courbes de la fonction objectif* $f(x) = 0.5sin(x)$ *et de la fonction de performance* $f(x) + p * |x - 3.5|$*, avec p=1.7.*

une fois pour toutes, et gardent les mêmes valeurs pour tout le reste de l'évolution. Par contre, avec les approches dynamique et adaptative, leurs valeurs sont modifiées tout au long de l'évolution, selon un schéma défini par l'utilisateur pour la première, et selon l'état historique et/ou courant de la population pour la deuxième. La fonction de pénalité dynamique est généralement croissante afin d'assurer la faisabilité des solutions à la fin de l'évolution, mais le schéma de modification n'est pas simple à déterminer et dépend du problème posé. En ce qui concerne la méthode adaptative, elle modifie les cœfficients de pénalité en se basant sur des informations extraites de la population, essentiellement la faisabilité du meilleur ou d'une certaine proportion de la population.

Remarque: Les méthodes statiques et dynamiques ne sont pas spécifiques aux algorithmes évolutionnaires. L'approche statique peut être appliquée à toute méthode d'optimisation (section 2.2.5), et celle dynamique peut être adaptée à des algorithmes d'optimisation itératives. Seules les méthodes adaptatives sont spécifiques au calcul évolutionnaire.

2.3.1 La pénalité mortelle ou “death penalty”

C'est une méthode très simple qui ne fait que rejeter les solutions infaisables de la population [8]. En fait, elle ne ressemble pas exactement à une méthode de pénalisation puisque le rejet se fait au cours de la sélection, mais elle peut être vue comme une méthode de pénalisation avec une pénalité infinie:

$$penal(\overrightarrow{x}) = +\infty$$

Exemple de Résultat

Pour évaluer la méthode, Michalewicz [81] l'a testé sur le problème suivant, qui consiste à minimiser la fonction:

$$G7(\overrightarrow{x}) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45,$$

sous les contraintes suivantes:

$105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 \geqslant 0,$
$-3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2x_3^2 + 7x_4 + 120 \geqslant 0,$
$-10x_1 + 8x_2 + 17x_7 - 2x_8 \geqslant 0,$
$-x_1^2 - 2(x_2 - 2)^2 + 2x_1x_2 - 14x_5 + 6x_6 \geqslant 0,$
$8x_1 - 2x_2 - 5x_9 + 2x_{10} + 12 \geqslant 0,$
$-5x_1^2 - 8x_2 - (x_3 - 6)^2 + 2x_4 + 40 \geqslant 0,$
$3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} \geqslant 0,$
$-0.5(x_1 - 8)^2 - 2(x_2 - 4)^2 - 3x_5^2 + x_6 + 30 \geqslant 0,$

et les bornes: $-10.0 \leqslant x_i \leqslant 10.0, \quad i = 1, \ldots, 10.$

Ce problème a 3 contraintes linéaires et 5 autres non linéaires. La fonction G7 est quadratique et a comme optimum global:
$\overrightarrow{x}^* = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927)$
où $G7(\overrightarrow{x}^*) = 24.3062091$. Six (sur huit) contraintes sont actives aux alentours de l'optimum global (toutes sauf les deux dernières).

La méthode de pénalité mortelle a donné des solutions d'une qualité moyenne dont la meilleure a comme valeur 25.653. Pour évaluer cette méthode, il est nécessaire d'initialiser la population avec des points faisables. En effet, des tests réalisés par Michalewicz en 1995 ont prouvé qu'en partant d'une population aléatoire, cette méthode donne des résultats peu performants.

Discussion

La qualité des résultats donnés par cette méthode dépend fortement du degré de faisabilité de la population initiale. En effet, cette technique ne peut fonctionner que s'il y a assez d'individus faisables dans la population initiale. Entre autres, si l'espace faisable est dispersé et la population initiale ne couvre qu'une partie de $\mathcal{F}$, l'algorithme aura des grandes difficultés à explorer toutes ses régions.

Sa dépendance vis à vis du schéma d'initialisation cause la non stabilité de la méthode et augmente la dispersion des solutions retournées. C'est pour ces raisons que cette méthode a généralement une performance très faible.

2.3.2 Pénalités statiques

Cette méthode a été proposée par Homaïfar, Lai et Qi en 1994 [58]. Ils proposent de définir pour chaque contrainte une famille d'intervalles de violation déterminant chacun un cœfficient de pénalité approprié. L'algorithme de la méthode se résume dans les étapes suivantes:

- pour chaque contrainte, créer un certain nombre (l) de taux de violation,
- pour chaque taux de violation et pour chaque contrainte, créer un cœfficient de pénalité $R_{ij}(i = 1, 2, \cdots, l, j = 1, 2, \cdots, m)$; les cœfficients ayant les valeurs les plus larges sont affectés aux taux de violation les plus élevés,
- la population initiale est initialisée aléatoirement sans tenir compte de la faisabilité des individus,
- faire évoluer la population; chaque individu est évalué avec la formule suivante:

$$eval(\overrightarrow{x}) = f(\overrightarrow{x}) + \sum_{j=1}^{m} R_{ij} v_j^2(\overrightarrow{x})$$

Exemple de résultats

Quelques expériences rapportées par Michalewicz dans [81] indiquent que la méthode peut donner des résultats satisfaisants si les intervalles des violations et les cœfficients de pénalisation sont bien ajustés au problème posé. Par exemple, Homaïfar et al. [58] ont fait des tests pour le problème suivant [52]:

Minimiser $G4(\overrightarrow{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$,

sous les contraintes suivantes:

$0 \leqslant 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 \leqslant 92$
$90 \leqslant 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 \leqslant 110$
$20 \leqslant 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 \leqslant 25$,

et les bornes:

$78 \leqslant x_1 \leqslant 102$, $33 \leqslant x_2 \leqslant 45$, $27 \leqslant x_i \leqslant 45$ pour $i = 3, 4, 5$.

La solution optimale est $\overrightarrow{x}^* = (78, 33, 29.995256, 45, 36.775812)$, où $G4(\overrightarrow{x}^*) = -30665.5$. Deux contraintes sont actives au niveau de l'optimum, la première (borne supérieure) et la troisième (borne inférieure).

La meilleure solution trouvée par Homaïfar et al. sur 10 essais est : $\overrightarrow{x} = (80.49, 35.07, 32.05, 40.33, 33.34)$ avec $G4(\overrightarrow{x}) = -30005.7$, ce qui n'est pas proche de l'optimum. Ces résultats peuvent être améliorés par l'optimisation des valeurs des cœfficients de pénalité, mais cette recherche n'est pas simple à faire.

Discussion

Le grand inconvénient de cette méthode est le nombre de paramètres à définir avant l'évolution. Pour m contraintes, la méthode nécessite au total $m(2l+1)$ paramètres: m paramètres pour établir le nombre d'intervalles pour chaque contrainte, l paramètres pour définir les bornes des intervalles (taux de violation) pour chaque contrainte, et l paramètres représentant les cœfficients de pénalité R_{ij} pour chaque contrainte. Ainsi, pour $m = 5$ contraintes et $l = 4$ taux de violation, on a besoin de définir 45 paramètres! Ce grand nombre de paramètres rend la qualité des résultats de la méthode très dépendante du choix de leurs valeurs. Il est possible, que pour un problème donné, il existe un ensemble de paramètres optimal pour lequel le système retourne des solutions très proches de l'optimum. Cependant, trouver cet ensemble est en lui même un problème d'optimisation difficile.

2.3.3 Pénalités dynamiques

Contrairement à la méthode précédente, avec la stratégie dynamique, les cœfficients de pénalité sont modifiés au long de l'évolution. Leurs valeurs sont alors augmentées progressivement, afin d'obliger l'algorithme à retourner des solutions faisables. Deux méthodes ont été proposées en 1994, la première par *Joines et Houk* [63], et la deuxième par *Michalewicz et Attia* [83].

Méthode de Joines et Houk, 94 (63)

L'idée de base de cette méthode est d'augmenter la pénalité au fur et à mesure de l'évolution afin d'augmenter la pression sur l'algorithme pour trouver des individus faisables. Ainsi, à une génération donnée, la pénalité est calculée comme suit:

$$penal(\overrightarrow{x}) = (C \times t)^{\alpha} \sum_{j=1}^{m} v_j^{\beta}(\overrightarrow{x}),$$

Où C, α, $et\ \beta$ sont des constantes. Un bon choix de ces paramètres rapporté par Joines et Houck [63] est $C = 0.5$, $\alpha = \beta = 2$.

Exemple de Résultats

Joines et Houck ont testé leur méthode sur un des problèmes proposés par Hock et Schittkowski dans [55] et qui consiste à minimiser la fonction:

$G5(\overrightarrow{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + 0.000002/3x_2^3$,

sous les contraintes suivantes:

$x_4 - x_3 + 0.55 \geqslant 0$, $x_3 - x_4 + 0.55 \geqslant 0$,

$1000sin(-x_3 - 0.25) + 1000sin(-x_4 - 0.25) + 894.8 - x_1 = 0$,

$1000sin(x_3 - 0.25) + 1000sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$

$1000sin(x_4 - 0.25) + 1000sin(x_4 - x_3 - 0.25) + 1294.8 = 0$,

et les bornes:

$0 \leqslant x_i \leqslant 1200 \quad \text{pour} \quad i = 1,2 \quad \text{et} \quad -0.55 \leqslant x_i \leqslant 0.55 \quad \text{pour} \quad i = 3,4$

La solution optimale connue pour ce problème est $\overrightarrow{x}^* = (679.9453, 1026.067, 0.1188764, -0.3962336)$, avec $G5(\overrightarrow{x}^*) = 5126.4981$.

Le meilleur résultat des expériences reportées par Joines et Houck en 1994 (sur plusieurs essais et différentes valeurs de α et β) a comme valeur de la fonction objectif 5126.6653. Il n'y avait pas des solutions strictement réalisables à cause des contraintes d'égalité, mais la somme des violations était très petite (de l'ordre de 10^{-4}).

Discussion

Cette méthode nécessite beaucoup moins de paramètres que la méthode des pénalités statiques et donne des meilleurs résultats grâce à la pression croissante sur les individus infaisables due au terme $(C \times t)^\alpha$ de la fonction de pénalité. Cependant, cette pression peut parfois être trop forte et gêner le processus d'exploration. Souvent le facteur $(C \times t)^\alpha$ augmente très rapidement et son utilité décroît en parallèle. Ainsi, le système a des faibles chances d'échapper des optima locaux: dans la majorité des expériences réalisés par Michalewicz [81], la meilleure solution est trouvée dans les premières générations. Par contre, la méthode a donné des bons résultats pour des problèmes où la fonction objectif était quadratique.

Méthode de Michalewicz et Attia, 94 (Pénalités analytiques)

Cette méthode, proposée par *Michalewicz* et *Attia* en 1994 [83] est utilisée dans le système **Genocop II**. Genocop II est une deuxième version du système Genocop ("GEnetic algorithm for Numerical Optimization of COnstrained Problems"). Ce dernier avait le handicap de ne pouvoir traiter que les contraintes linéaires (c.f. section 2.5.1). D'autres options lui ont alors été rajoutées lui permettant de prendre en compte tout type de contraintes, donnant ainsi naissance au système Genocop II.

L'algorithme commence d'abord par différentier les contraintes linéaires LC de celles non linéaires NC. Il génère ensuite une population initiale qui doit satisfaire l'ensemble LC. Elle peut être constituée par des copies d'un seul point $\overrightarrow{x}_s$ satisfaisant toutes les contraintes linéaires. La faisabilité de la population par rapport à LC est maintenue au cours de l'évolution grâce à l'ensemble des opérateurs spéciaux du système Genocop (c.f. section 2.5.1), qui transforment une solution faisable en une autre faisable.

Pour prendre en compte les contraintes non linéaires, *Michalewicz* et *Attia* se sont inspirés de la stratégie de la température de refroidissement du Recuit Simulé pour définir une fonction de pénalisation, utilisée pour l'évaluation des individus infaisables:

$$penal(\overrightarrow{x}, \tau) = \frac{1}{2\tau} \sum_{j \in A} {v_j}^2(\overrightarrow{x}),$$

où τ est la température du système, initialisé par l'utilisateur. τ est diminué à chaque génération, selon un schéma de "refroidissement" choisi aussi par l'utilisateur. L'objectif de l'utilisation d'un schéma de refroidissement est d'augmenter la pression sur les individus irréalisables au fil de l'évolution. L'algorithme s'arrête quand τ atteint la température minimale τ_f, choisi préalablement.

Exemple de résultats

Un des problèmes testés par *Michalewicz* et *Attia* [32] est :

Minimiser $G6(\overrightarrow{x}) = (x_1 - 10)^3 + (x_2 - 20)^3$,

sous les contraintes non linéaires suivantes:

$(x_1 - 5)^2 + (x_2 - 5)^2 - 100 \geqslant 0$,

$-(x_1 - 6)^2 - (x_2 - 5)^2 + 82.81 \geqslant 0$,

et les bornes:

$13 \leqslant x_1 \leqslant 100$ et $0 \leqslant x_2 \leqslant 100$.

La solution globale connue est $\overrightarrow{x}^* = (14.095, 0.84296)$, avec $G6(\overrightarrow{x}^*) = -6981.81388$.
Les deux contraintes du problème sont actives aux alentours de l'optimum.

Genocop II a réussi à résoudre facilement ce problème. Il s'approche de très prés de l'optimum dès la $12^{\text{ème}}$ génération.

Discussion

Cette méthode nécessite le choix de deux paramètres importants: la température initiale τ_0 et la température finale τ_f. Il lui faut aussi un schéma de "refroidissement" (*freezing temperature*) pour diminuer la température τ. Des expérimentations faites par les auteurs [83] montrent que leur algorithme peut converger en quelques itérations avec un bon choix du schéma de "refroidissement", comme il peut donner des résultats peu satisfaisants avec d'autres schémas. La question qui se pose est comment choisir ces paramètres pour un problème d'optimisation donné. Si τ_0 est très grande (pénalisation initiale faible), l'algorithme risque de converger vers les zones des meilleurs infaisables. Et si cette zone est loin de la localisation exacte de l'optimum, la croissance de la pénalisation ne l'aidera pas à sortir de ces zones pour explorer d'autres environs de l'espace. De même, si τ_0 est très petite (pénalisation initiale forte), l'algorithme convergera rapidement vers les zones des meilleurs faisables, et l'exploration des autres parties l'espace de recherche n'est plus assurée.

2.3.4 Pénalités adaptatives

L'idée de base des méthodes basées sur des pénalités adaptatives est d'introduire dans la fonction de pénalité un composant dépendant de l'état du processus de recherche à une génération donnée. Ainsi, le poids de la pénalité est adapté à chaque itération et il peut être augmenté ou diminué selon la qualité des solutions dans la population. Deux méthodes ont été proposées, la première en 1992 par *Hadj-Alouane* et *Bean* [47], la deuxième en 1993 par *Smith* et *Tate* [124].

méthode de Hadj-Alouane et Bean, 1992

Avec cette méthode, le poids de la pénalité dépend de la qualité de la meilleure solution trouvée à la génération t. La fonction de pénalité est définie comme suit:

$$penal(\overrightarrow{x}) = \lambda(t)\sum_{j=1}^{m} v_j^2(\overrightarrow{x})$$

où $\lambda(t)$ est mise à jour chaque génération t comme suit:

$$\lambda(t+1) = \begin{cases} (1/\beta_1).\lambda(t), & si\ x_b \in \mathcal{F} \text{ durant les k dernières générations} \\ \beta_2\lambda(t) & si\ x_b \in (\mathcal{S} - \mathcal{F}) \text{ durant les k dernières générations} \\ \lambda(t) & sinon, \end{cases}$$

où x_b est le meilleur individu et $\beta_1, \beta_2 > 1$ (avec $\beta_1 \neq \beta_2$ pour éviter les cycles). Autrement dit, cette méthode diminue la valeur de la composante $\lambda(t+1)$ à la génération $t+1$ si tous les meilleurs individus pendant les k dernières générations étaient faisables, et augmente le poids de la pénalité si tous les meilleurs individus pendant les k dernières générations étaient infaisables. S'il y a eu pendant ces générations à la fois des meilleures solutions faisables et d'autres infaisables, $\lambda(t+1)$ garde la même valeur que $\lambda(t)$.

Le but de *Hadj-Alouane* et *Bean* en utilisant les pénalités adaptatives était d'augmenter l'efficacité du processus de recherche. Ainsi, si les contraintes ne posent pas de problèmes pour l'algorithme, les pénalités diminuent, sinon, elles augmentent.

Exemple de résultat

Hadj-Alouane et *Bean* ont implanté cette méthode pour résoudre le problème MCIP ("*Multiple-Choice Integer Program*") défini mathématiquement comme suit:

minimiser $C.\overrightarrow{x}$ sous les conditions:
$A\overrightarrow{x} - b \geqslant 0$,
$\sum_{j=1}^{n_i} x_{ij} = 1$, pour $i = 1, 2, \cdots, m$ (MCIP),
$x_{ij} \in \{0, 1\}$.

Ce problème est NP-complet et sa complexité augmente exponentiellement avec n. Trois types d'instances du problème ont été essayées:

- de petite taille: $(m, n_i, k) = (10, 5, 1)$,
- de taille moyenne: $(m, n_i, k) = (40, 20, 10)$,
- de grande taille: $(m, n_i, k) = (100, 50, 20)$.

Pour chaque taille, 10 instances, générées aléatoirement, ont été testées, avec 10 essais pour chacune. Les autres paramètres de la méthode β_1 et β_2 ont été fixés à 4 et 2.8 respectivement.

Pour toutes les configurations, l'algorithme a réussi à trouver l'optimum. Le temps médian, maximal et minimal (sur les 10 instances) pour les trois tailles est donné dans le tableau suivant:

	temps médian minimal	temps médian maximal
petite taille	11	64
taille moyenne	78	105
taille maximale	141	197

Discussion

C'est la première méthode qui a introduit l'adaptativité des cœfficients de pénalisation. Cependant, la stratégie d'adaptation ne repose que sur des information sur l'état du meilleur individu pendant les k dernières générations. Elle ne tient pas compte de l'état général de la population. Si le meilleur est faisable pendant un long moment de l'évolution mais tous les autres individus ne le sont pas, le taux de pénalisation est diminué, alors qu'il devrait être augmenté pour forcer l'algorithme à générer d'autre solutions réalisables.

méthode de Smith et Tate, 1993

La fonction de pénalisation adaptative proposée par *Smith* et *Tate* incorpore, comme la méthode précédente, une composante indiquant l'état d'évolution du processus de recherche, ainsi qu'une composante indiquant le degré de violation des contraintes. La première composante dépend de la qualité de la meilleure solution trouvée pendant l'évolution (jusqu'à l'itération courante t). La deuxième composante est déterminée par la distance des meilleures solutions infaisables au domaine faisable. Le but de cette fonction est d'élargir l'espace de recherche en introduisant les solutions infaisables intéressantes (proches du domaine faisable), ce qui peut faciliter le processus de recherche quand l'optimum se trouve sur les bords du la région réalisable.

La fonction de pénalité est définie comme suit:

$$penal(\overrightarrow{x}) = (F_{feas}(t) - F_{all}(t)) \sum_{j=1}^{m} (v_j(\overrightarrow{x})/q_j(t))^k,$$

où $F_{all}(t)$ est la valeur de la fonction objectif (sans pénalisation) de la meilleure solution trouvée pendant l'évolution (jusqu'à la génération courante t), $F_{feas}(t)$ est l'évaluation de la meilleure solution faisable trouvée pendant l'évolution, $q_j(t)$ est l'estimation du seuil d'extension de faisabilité pour chaque contrainte j ($1 \leqslant j \leqslant m$), et k est une constante qui permet d'ajuster la "sévérité" de la fonction de pénalisation. Il faut noter que les seuils $q_j(t)$ sont dynamiques; ils sont ajustés durant le processus de recherche.

Exemple de résultat

Smith et *Tate* on testé leur méthode sur un problème de disposition de départements sur une surface donnée: *"the unequal area facility layout problem"*. Ils considèrent un rectangle de dimension $H \times W$, et une collection de n départements ayant des surfaces bien spécifiées, dont la somme est égale à $H.W$. A chaque paire de départements (j, k) est associée une valeur $F(j, k)$ présentant le flot du trafic entre eux. L'objectif est de partitionner la région en un ensemble de secteurs, un par département, tout en minimisant la somme:

$$\sum_{j=k}^{n} F(j, k).\delta(j, k, \Pi),$$

où $\delta(j, k, \Pi)$ est la distance entre le centroïde du département j et celui du département k dans la partition Π. A chaque département on associe une contrainte sur la longueur minimale du bord.

Pour 20 départements, la meilleure solution sur 10 essais a été trouvée avec k=1 et elle est de 5140.3. Cependant, des valeurs plus petites ou plus grandes de k affectent la qualité des résultats.

Par exemple, pour k=0.5, la meilleure performance est égale à 5305, et elle est de 5278 pour k=3.

Discussion

Comme pour la méthode de *Hadj-Alouane* et *Bean*, la fonction de pénalisation ne tient pas compte de l'état général de la population. Seules les performances des meilleures solutions faisables et infaisables sont prises en compte.

Entre autres, les expérimentations faites par les auteurs montrent que la qualité des résultats est très sensible au paramètre k, utilisé pour l'ajustement de la "sévérité" de la pénalisation.

2.3.5 Supériorité des individus faisables

L'approche de la supériorités des individus faisables repose sur le principe suivant: toute solution faisable est meilleure que toute solution infaisable. Deux méthodes utilisant ce principe ont été proposées: la première a été mise en place par *Powell* et *Skolnick* en 1993 [95], et la deuxième est récente et a été proposée par *Deb* en 2000 [19].

méthode de Powell et Skolnick, 1993

Pour un problème de minimisation, l'approche de *Powell* et *Skolnick* se résume à projeter toutes les évaluations des solutions faisables dans l'intervalle $(-\infty, 1)$ et celles des solutions infaisables dans l'intervalle $(1, +\infty)$. Pour se faire, elle utilise aussi les fonctions de pénalité, mais avec un principe modifié. En effet, en plus des taux de violation des contraintes, elle utilise l'évaluation des solutions faisables.

$$penal(\overrightarrow{x}) = r\sum_{j=1}^{m} v_j(\overrightarrow{x}) + \theta(t, \overrightarrow{x})$$

où r est une constante définie à par l'utilisateur.

Le composante $\theta(t, \overrightarrow{x})$ est une fonction dépendante de la génération en cours, et ayant une grande influence sur l'évaluation des individus infaisables:

$$\theta(t, \overrightarrow{x}) = \begin{cases} 0, & \text{si } \overrightarrow{x} \in \mathcal{F} \\ max\left\{0, max_{\overrightarrow{y}\in\mathcal{F}}\left\{f(\overrightarrow{y})\right\} - min_{\overrightarrow{y}\in(\mathcal{S}-\mathcal{F})}\left\{f(\overrightarrow{y}) + r\sum_{j=1}^{m} v_j(\overrightarrow{y})\right\}\right\}, & sinon. \end{cases}$$

Avec cette heuristique supplémentaire, les individus infaisables dépendent des faisables: leurs performances ne peuvent pas être meilleures que la performance de la pire solution faisable ($max_{\overrightarrow{x}\in\mathcal{F}}\left\{f(\overrightarrow{x})\right\}$).

Cette méthode nécessite le choix d'un seul paramètre r. L'utilisation d'une petite valeur permet à l'algorithme d'explorer le domaine irréalisable en parallèle du domaine réalisable, mais si r est grand, peu de points infaisables survivent dans la population (les individus faisables dominent la population dès les premières générations de l'évolution).

Exemples de résultats

Cette méthode a également été testée par *Michalewicz* [81] sur le cas test $G9$ suivant:

Minimiser $G9(\overrightarrow{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$, sous les contraintes suivantes:

$127 - 2x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 \geqslant 0,$
$282 - 7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 \geqslant 0,$
$196 - 23x_1 - x_2^2 - 6x_6^2 + 8x_7 \geqslant 0,$
$-4x_1^2 - x_2^2 + 3x_1x_2 - 2x_3^2 - 5x_6 + 11x_7 \geqslant 0$

et les bornes: $-10.0 \leqslant x_i \leqslant 10.0$, $i = 1, \ldots, 7$.

Ce problème a 4 contraintes non linéaires. La fonction G9 est aussi non linéaire et son minimum global est:
$\overrightarrow{x}^* = (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227)$, où $G9(\overrightarrow{x}^*) = 680.6300573$.

L'algorithme a donné pour ce problème des résultats raisonnables; la meilleure solution trouvée était 680.934.

Discussion

Cette technique a été testée par ses auteurs sur 10 problèmes, et elle a réussi à trouver l'optimum exacte pour quelques uns, mais elle échoue à trouver une solution faisable si le domaine de faisabilité est trop petit ou si le rapport $|F|/|S|$ est petit.

Méthode de Deb, 2000

La méthode proposée par *Deb* [19] évite le calcul de la fonction objectif dans le domaine irréalisable. L'approche proposée se base sur la sélection à tournoi de dimension 2, où deux solutions sont comparées selon les critères suivants:

1. toute solution faisable est meilleure que toute solution infaisable,
2. parmi deux solutions faisables, celle ayant la meilleure performance est sélectionnée,
3. parmi deux solutions infaisables, celle ayant le degré de violation le plus petit est sélectionnée.

La fonction de performance n'utilise pas de cœfficients de pénalisation pour l'évaluation des solutions non faisables, mais se base sur leur taux de violation des contraintes:

$$F(\overrightarrow{x}) = \begin{cases} f(\overrightarrow{x}), & \text{si } \overrightarrow{x} \in \mathcal{F}, \\ f_{max} + \sum_{j=1}^{m} g_j(\overrightarrow{x}), & \text{sinon,} \end{cases}$$

où f_{max} est la valeur de la fonction objectif de la pire solution faisable dans la population. Ainsi, la performance d'un individu infaisable ne dépend pas seulement des degrés de violation des contraintes, mais aussi de la population courante.

Pour maintenir la diversité dans le domaine réalisable, la méthode utilise une technique de nichage (chapitre 1, section 1.6), appliquée pendant l'étape de sélection: un individu faisable $\overrightarrow{x}$ est sélectionné s'il est considéré comme gagnant. Un ensemble de n individus sont alors sélectionnés aléatoirement parmi les solutions réalisables. S'ils sont tous distants de $\overrightarrow{x}$ d'une distance supérieure à $\overline{d}$ (paramètre défini par l'utilisateur), $\overrightarrow{x}$ est considéré comme gagnant.

Exemples de résultats

Deb a testé sa méthode sur un ensemble de problèmes de référence, comme la fonction G9 citée ci-dessus. Pour ce test, la méthode a donné comme meilleure solution 680.634 et comme solution médiane 680.642. La pire solution était de 680.651. Ces résultats sont considérés comme bons, puisqu'ils sont très proches de la valeur de l'optimum.

D'autres expériences ont aussi été réalisées, pour la résolution de la fonction G7, défini dans le paragraphe 2.3.1. La technique a donné comme meilleur résultat, résultat médian et pire résultat, respectivement les valeurs 24.372, 24.409 et 25.075. Ces résultats montre une bonne performance de cette méthode, puisque la meilleure solution est à 0.27% près de l'optimum, et l'écart reporté entre la meilleure et la pire solution est assez petit.

Discussion

Cette méthode a l'avantage qu'elle ne nécessite pas de l'utilisateur le choix de paramètres supplémentaires (exepté le paramètre $\overline{d}$ de la procédure de nichage) et qu'elle ne calcule pas la fonction objectif dans le domaine infaisable.

Par contre, comme pour la méthode précédente, cette technique échoue si le rapport $|F|/|S|$ est très petit et l'algorithme ne réussit pas à localiser des faisables.

2.3.6 "Segregated GA" SGGA (Leriche et Al, 95)

Le SGGA a été proposé par *Le Riche et al.* en 1995 [75] comme une nouvelle approche de prise en compte des contraintes par pénalisation, qui utilise deux pénalités différentes en même temps. Sa stratégie d'ajustement des cœfficients de pénalisation repose sur le dilemme suivant:

- un cœfficient de pénalisation très petit donne des solutions irréalisables(..),
- un cœfficient de pénalisation élevé restreint la recherche à l'intérieur du domaine réalisable et interdit tout passage par le domaine irréalisable, ce qui peut éventuellement nuire au processus de recherche et empêcher la convergence vers l'optimum global.

L'idée générale de la méthode est d'ajuster les cœfficients de pénalité pour chaque problème afin d'assurer une convergence dans l'espace faisable tout en permettant une exploration efficace .

Le problème qui se pose est qu'il n'y a pas une solution générale pour un ajustement optimal des cœfficients de pénalité, ni dans le calcul numérique, ni dans le calcul évolutionnaire. Pour résoudre ce problème, l'algorithme génétique ségrégationnel SGGA propose de *désensibiliser* la méthode aux choix des cœfficients de pénalité, en utilisant à la fois une grande et une petite valeur, et ainsi assurer la diversité de la population.

SGGA établie à partir de la population deux groupes d'individus coexistant et coopérant entre eux, et qui diffèrent l'un de l'autre par la méthode de calcul de la performance de leurs membres: chaque groupe utilise un cœfficient de pénalité différent. Les deux groupes sont séparés (*segregated*) pendant l'étape de la sélection, où chaque individu est classé dans une liste utilisant une des deux pénalisation. La nouvelle population est constituée par la sélection des meilleurs dans les deux listes.

Deux avantages résultent de cette stratégie:

1. Étant donné que les paramètres de pénalisation sont différents, les deux groupes vont avoir des trajectoires dans l'espace de recherche différentes: celle du groupe avec une grande pénalité sera à l'intérieur du domaine faisable, alors que celle du groupe ayant une faible pénalité sera à l'extérieur de $\mathcal{F}$. Entre autres, grâce à l'hybridation des deux groupes, chacun aide l'autre à éviter la convergence prématurée (s'éloigner des optima locaux). Ainsi, le SGGA a un mécanisme d'exploration plus robuste qu'un EA standard utilisant une seule pénalisation.
2. Dans les problèmes d'optimisation sous contraintes, souvent l'optimum global est sur la frontière entres les domaines réalisable et irréalisable. L'hybridation entre les faisables (groupe

avec une grande pénalité) et les non faisables (groupe avec une faible pénalité) favorise l'exploration de la frontière et l'optimum global est ainsi localisé assez facilement.

Exemple de résultat

Cet algorithme a été testé dans [84] sur un problème d'optimisation de forme soumis à un ensemble de contraintes, et les résultats ont été comparés avec ceux donné par un algorithme génétique standard (GA) et un algorithme génétique élitiste (EGA) sans ségrégation et utilisant un cœfficient de pénalisation constant. Trois valeurs de pénalisation (p) ont été essayés pour GA et EGA: $p = 0.4, 0.5, et 5$, et 3 combinaisons pour SGGA: ($p1 = 0.5, p2 = 0.4$), ($p1 = 5, p2 = 0.4$) et ($p1 = 5, p2 = 0.5$). Le GA a montré des faibles performances avec les pénalités 0.4 et 5, où il a retourné des solutions infaisables pour une grande partie des essais. De même, EGA a montré une grande sensibilité aux valeurs de pénalisation, où il a échoué avec $p = 0.4$ et a montré une faible performance avec $p = 5$. Par contre, SGGA a été un peu moins sensible à la combinaison des cœfficients de pénalisation choisie. Il a résussi à trouver des solutions faisables pour tous les tests et a atteint sa meilleure performance avec ($p1 = 5, p2 = 0.5$).

Discussion

Les résultats des expérimentations faites dans [84] ont montré que la performance de SGGA est sensible au choix des cœfficients de pénalisation, même si cette sensibilité est inférieure à celle d'un GA standard utilisant une pénalisation statique.

Entre autres, cette méthode n'a été testée que sur un problème d'optimisation de forme. D'autres tests sur des fonctions de référence sont nécessaires pour pouvoir juger sa robustesse.

2.3.7 Stochastic Ranking (Runarsson et Yao, 2000)

Proposée par *Runarsson* et *Yao* en 2000 [106], cette méthode propose une nouvelle approche pour créer une balance entre la fonction objectif et la fonction de pénalisation, basée sur un classement stochastique des individus.

Si on considère que les individus sont évalués avec l'équation 2.1, alors la fonction de pénalisation est définie comme suit:

$$penal(\overrightarrow{x}) = r_t.\theta(\overrightarrow{x}),$$

avec r_t est le cœfficient de pénalité et $\theta(\overrightarrow{x})$ est la somme des violations:

$$\theta(\overrightarrow{x}) = \sum_{j=1}^{m} v_j(\overrightarrow{x})$$

où les v_j sont les mesures de violations données par l'équation 2.3

Pour comparer deux individus adjacents $\overrightarrow{x_i}$ et $\overrightarrow{x}_{i+1}$, *Runarsson* et *Yao* ont introduit la notion de cœfficient de pénalisation critique:

$$\tilde{r}_i = \frac{f(\overrightarrow{x}_{i+1}) - f(\overrightarrow{x}_i)}{\theta(\overrightarrow{x}_i) - \theta(\overrightarrow{x}_{i+1})}, \text{ pour } \theta(\overrightarrow{x}_i) \neq \theta(\overrightarrow{x}_{i+1})$$

Pour un choix donné de $r_g > 0$, trois types de comparaisons sont possibles:

1. Comparaison dominée par la fonction objectif: $f_i \leqslant f_{i+1}$, $\theta_i \geqslant \theta_{i+1}$ et $0 < r_t < \tilde{r}_i$;
2. Comparaison dominée par la fonction de pénalité: $f_i \geqslant f_{i+1}$, $\theta_i < \theta_{i+1}$ et $0 < \tilde{r}_i < r_t$;

3. Comparaison non dominée: $f_i < f_{i+1}$, $\theta_i < \theta_{i+1}$ et $\tilde{r}_i < 0$.

Si $\overline{r}_t$ et $\underline{r}_t$ sont, respectivement, le plus grand et le plus petit cœfficient de pénalité critique calculés à partir des individus adjacents classés selon la fonction objectif, alors il faut que $\underline{r}_t < r_t < \overline{r}_t$ pour que la pénalisation soit significative. Si $r_t < \underline{r}_t$, alors toutes les comparaisons sont basées sur la fonction objectif: cas de *sous-pénalisation*. Par contre, si $r_t > \overline{r}_t$, alors toutes les comparaisons sont basées sur la fonction de pénalisation seulement: cas de *sur-pénalisation*.

Chercher une bonne stratégie pour ajuster r_t à chaque génération, tout en évitant la *sous-pénalisation* et la *sur-pénalisation*, est en lui même un problème d'optimisation. Pour éviter le passage par cette étape, *Runarsson* et *Yao* proposent le classement stochastique. Ils définissent une probabilité P_f qui sert à décider d'utiliser la fonction objectif ou la fonction de pénalisation pendant la comparaison. Ainsi, deux individus adjacents $\vec{x}_i$ et $\vec{x}_{i+1}$, où au moins un est infaisable, ont une probabilité P_f d'être comparés selon leurs valeurs de la fonction objectif, et une probabilité $(1 - P_f)$ d'être comparés selon leurs taux de violation des contraintes. Si les deux individus sont faisables, $P_f = 1$.

Exemple de Résultats

Cette méthode a été testée sur un ensemble de fonctions de référence (voir Annexe A). Les meilleurs résultats ont été enregistrés avec $P_f = 0.45$, et ils sont illustrés dans le tableau 2.1.

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11
b	-15	0.803515	1	-30665.5	5126.49	-6961.81	24.3	0.095825	680.63	7054.31	0.75
a	-15	0.781975	1	-30665.5	5128.88	-6875.94	24.37	0.095825	680.65	7559.19	0.75
w	-15	0.726288	1	-30665.5	5142.47	-6350.26	24.64	0.095825	680.76	8835.65	0.75

TAB. 2.1 – *Résultat de l'application de la méthode de "Stochastic Ranking" sur 11 cas test, donnant la meilleure (b), la moyenne (a) et la pire (w) solution sur 30 essais effectués pour chaque fonction.*

Discussion

L'avantage de la méthode de *"Stochastic Ranking"* est qu'elle est simple à mettre en oeuvre et ne demande pas de l'utilisateur la définition d'un grand ensemble de paramètres. Elle a réussi à donner des très bons résultats pour la plupart des tests, mais elle présente une grande sensibilité à la valeur de la probabilité P_f. En effet, l'étude expérimentale faite par les auteurs montre qu'il suffit de modifier légèrement cette valeur (e.g. 0.47 au lieu de 0.45) pour que la qualité des résultats se détériore.

2.4 Méthodes évolutionnaires: Recherche des solutions faisables

On compte essentiellement deux méthodes dans cette catégorie: la réparation des individus infaisables et l'échantillionnage de l'espace faisable. L'objectif de ces deux méthodes est de ramener les individus infaisables dans le domaine réalisable. Les deux sous sections suivantes donnent une description détaillée de ces deux techniques.

2.4.1 Réparation des individus infaisables: Genocop III

C'est la troisième version du système Genocop proposée par *Michalewicz* et *Nazhiyath* en 1995 [86]. Elle est basée sur l'idée de réparation des individus infaisables et introduit aussi quelques concepts de co-évolution. Cette méthode incorpore le système Genocop d'origine (décrit dans la section 2.5.1), et elle l'étend en faisant évoluer deux populations séparées, où le développement d'une population influence l'évaluation des individus dans l'autre. La première population P_s comporte des points qui satisfont les contraintes linéaires du problème, et ils sont appelés les points de recherche. La faisabilité de ces points (par rapport aux contraintes linéaires) est maintenue par les opérateurs spéciaux du système Genocop (c.f. section 2.5.1). La deuxième population P_r comporte des points qui satisfont toutes les contraintes du problème (linéaires et non linéaires), et ils sont appelés les points de référence. Les points de référence $\overrightarrow{r}_i$ de P_r, étant faisables, sont évalués directement avec la fonction objectif ($eval(\overrightarrow{r}) = f(\overrightarrow{r})$). Par contre, les points de recherche de P_s qui ne sont pas faisables sont réparés avant d'être évalués. Le processus de réparation se fait comme suit:

> Soit un point de recherche $\overrightarrow{s} \in P_s$.
> **Si** $\overrightarrow{s} \in \mathcal{F}$, alors $eval(\overrightarrow{s}) = f(\overrightarrow{s})$,
> **sinon** ($\overrightarrow{s} \notin \mathcal{F}$), **alors**
>> **répéter**
>>> sélectionner un point de référence $\overrightarrow{r}$ de P_r
>>> générer un nombre aléatoire a dans l'intervalle $[0, 1]$
>>> $\overrightarrow{z} = a\overrightarrow{s} + (1-a)\overrightarrow{r}$
>>
>> **tant que** ($\overrightarrow{z}$ est infaisable)
>
> $eval(\overrightarrow{s}) = eval(\overrightarrow{z}) = f(\overrightarrow{z})$
> remplacer $\overrightarrow{s}$ par $\overrightarrow{z}$ dans P_s avec une probabilité p_r
> **si** $f(\overrightarrow{z}) < f(\overrightarrow{r})$,
> **alors** remplacer $\overrightarrow{r}$ par $\overrightarrow{z}$ dans P_r

$\overrightarrow{s}$ est remplacé par $\overrightarrow{z}$ dans la population P_s avec une certaine probabilité de remplacement p_r. On note aussi qu'il y a une certaine asymétrie entre l'évolution des deux populations P_s et P_r. En effet, l'application des opérateurs de reproduction et de la procédure de sélection à la population P_s se fait à chaque génération, alors qu'elle ne se fait que toutes les k générations pour la population P_r, où k est un paramètre de la méthode.

La stratégie de co-évolution des deux populations est donnée par la procédure générale du système Genocop III présentée ci-dessous.

> **Procédure Genocop III**
> **début**
>> $t \leftarrow 0$
>> initialiser $P_s(t)$, $P_r(t)$
>> évaluer $P_s(t)$, $P_r(t)$
>> **Tant que** (**non** Condition fin) **faire**
>> **debut**
>>> $t \leftarrow t+1$
>>> sélectionner $P_s(t)$ de $P_s(t-1)$
>>> Reproduction de $P_s(t)$

```
        évaluer Ps(t)
        si  t mod k = 0 alors
            reproduction de Pr(t)
            sélectionner Pr(t) de Pr(t − 1)
            évaluer Pr(t)
        fin si
    fin tant que
fin procédure
```

On remarque que l'étape de reproduction précède la sélection dans la procédure d'évolution de P_r, due à la faible probabilité de générer un enfant faisable. Ainsi, d'abord les enfants sont générés, ensuite, les meilleurs individus faisables parmi les parents et les enfants sont sélectionnés pour former la nouvelle population (ce qui est similaire à la sélection $(\mu + \lambda)$ dans la terminologie des ES (chapitre 1, section 1.5.3)).

Exemple de résultats

Les résultats des expérimentations de Genocop III sur quelques problèmes sont d'une qualité assez satisfaisante. Par exemple, des tests ont été réalisés pour le problème suivant [55], prouvé très difficile pour la majorité des méthodes de prise en compte des contraintes:

Minimiser $G10(\overrightarrow{x}) = x_1 + x_2 + x_3$,
sous les contraintes suivantes:

$1 - 0.0025(x_4 + x_6) \geqslant 0$,
$1 - 0.0025(x_5 + x_7 - x_4) \geqslant 0$,
$1 - 0.01(x_8 - x_5) \geqslant 0$,
$x_1x_6 - 833.33252x_4 - 100x_1 + 83333.333 \geqslant 0$,
$x_2x_7 - 1250x_5 - x_2x_4 + 1250x_4 \geqslant 0$,
$x_3x_8 - 1250000 - x_3x_5 + 2500x_5 \geqslant 0$,

et les bornes:

$100 \leqslant x_1 \leqslant 10000,\ 1000 \leqslant x_i \leqslant 10000,\ i = 2, 3,\ 10 \leqslant x_i \leqslant 1000,\ i = 4, \ldots, 8.$

La problème a trois contraintes linéaires et trois autres non linéaires. La fonction G10 est linéaire et son minimum global est: $\overrightarrow{x}^* = (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.9799, 286.4162, 395.5979)$, où $G10(\overrightarrow{x}^*) = 7049.330923$. Toutes les contraintes sont actives au niveau de l'optimum global.

Genocop III a réussi à donner 7286.650 comme meilleure solution, ce qui était considérée, à l'époque, une bonne performance, vue la difficulté du problème.

Discussion

L'avantage avec Genocop III est qu'il n'évalue pas la fonction objectif dans l'espace non réalisable et qu'il retourne toujours une solution faisable. Par contre, il a beaucoup de difficulté à créer la population des points de référence si le rapport $|\mathcal{F}|/|\mathcal{S}|$ est très petit. Dans ce cas, il est possible que l'ensemble initial des points de référence soit une copie multiple d'un seul point faisable.

Entre autres, si l'espace faisable est disjoint et la population P_r a été initialisée dans une seule composante de $\mathcal{F}$, alors le système aura des difficultés de générer des nouveaux individus faisables dans les autres composantes de $\mathcal{F}$.

2.4.2 "Behavioral memory"

Cette méthode a été proposée par *Schoenauer* et *Xanthakis* en 1993 [117]. Ce nom est inspiré de l'article de DeGaris [41], qui a introduit la notion de mémoire comportementale de la population: la population ne contient pas seulement des informations sur le strict optimum, mais aussi des informations sur son comportement dans le passé.

Le but principal de cette méthode est d'échantillionner l'espace faisable en traitant les différentes contraintes du problème une par une et dans un ordre particulier. L'algorithme part d'une population aléatoire, ensuite, pour chaque contrainte, il fait évoluer la population jusqu'à ce qu'un certain pourcentage de la population devienne faisable pour la contrainte en cours, tout en continuant à respecter les contraintes précédentes. La population générée à la fin de chaque évolution sert de point de départ à l'évolution pour la contrainte suivante. Elle peut être vue comme une mémoire comportant des informations essentielles sur le contexte de l'évolution et nécessaire au bon déroulement des étapes suivantes.

La fonction objectif est optimisée dans la dernière étape en utilisant le principe de la pénalité mortelle ("death penalty") pour les individus infaisables.

Échantillonnage de l'espace faisable

Les différentes étapes de l'algorithme d'échantillionnage de l'espace faisable sont:

1. Initialiser la population aléatoirement.
2. Faire évoluer la population avec $eval(\overrightarrow{x}) = M_j(\overrightarrow{x})$, jusqu'à ce qu'un certain pourcentage donné ϕ de la population soit faisable pour la contrainte j.
3. La population courante sera le point de départ pour l'évolution de la contrainte suivante. Les points qui ne satisfont pas les (j-1) contraintes précédentes sont éliminés de la population.
4. Si $j < m$ (encore des contraintes à traiter), alors aller à 2.

M_j est une fonction mesurant le taux de violation de la $j^{\text{ème}}$ contrainte en cours de traitement. Le paramètre ϕ $(0 < \phi < 1)$ est la proportion minimale d'individus dans la population courante respectant la contrainte en cours de traitement, permettant au système de passer à la contrainte suivante.

Cette technique nécessite en plus une procédure de nichage pour maintenir la diversité de la population. En effet, le processus de traitement de contraintes linéairement peut générer à la dernière phase une population située dans un domaine très réduit de l'espace de recherche. Ce problème peut être résolu par la procédure de nichage, mais il faut bien ajuster le rayon de nichage σ.

Les fonctions de performance pour l'évolution des contraintes:

Pendant la minimisation d'une contrainte C_i du problème, la fonction de performance (M_i) est définie avec $M - C_i$, où M est un nombre positif suffisamment grand. Le choix de M ne peut pas être absolu pour les différentes contraintes pour les deux raisons suivantes:

1. il n'est pas souvent facile d'approximer la violation maximale d'une certaine contrainte,

2. choisir une constante très grande peut causer des ambiguïtés de distinction entre les points réellement faisables et les points approximativement faisables pour la contrainte C_i.

Pour les causes citées ci-dessus, le paramètre M est ajusté dynamiquement au cours de l'évolution.

Exemple de résultats

La méthode de la mémoire comportementale a été testée par *Schoenauer* et *Xanthakis* [117] sur le problème suivant:

Maximiser la fonction $G8(\overrightarrow{x}) = \frac{\sin^3(2\pi x_1)\cdot\sin(2\pi x_2)}{x^3\cdot(x_1+x_2)}$,
$0 \leqslant x_1 \leqslant 10$ et $0 \leqslant x_2 \leqslant 10$.
avec les contraintes

c1: $(\overrightarrow{x}) = x_1^2 - x_2 + 1 \leqslant 0$,
c2: $(\overrightarrow{x}) = 1 - x_1 + (x_2 - 4)^2 \leqslant 0$

et les bornes:

$0 \leqslant x_i \leqslant 10$ pour $1 \leqslant i \leqslant n$.

La fonction G8 a plusieurs optima locaux, les plus hauts pics se trouvent au long de l'axe des x. L'optimum connu se trouve à l'intérieur du domaine faisable et a comme coordonnées:
$X^* = (-1.737459, -0.167763)$, avec $G8(X^*) = 0.095825$.

L'algorithme ne trouve pas de difficulté à amener 80% de la population dans l'espace réalisable de la contrainte $c1$ (première étape), ensuite, dans l'espace réalisable des deux contraintes (deuxième étape), comme le montre la figure 2.4.

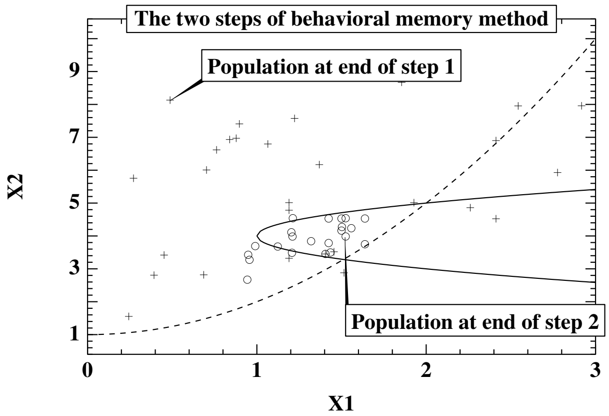


FIG. 2.4 – *Population de G8 après l'étape de l'échantillionnage. Les points "+" indiquent la population après le traitement de la première contrainte, alors que les cercles indiquent la population après le traitement de la deuxième contrainte.*

L'optimisation de la fonction principale G8 s'est fait en dernière étape avec succès. Cependant, pour garantir ce succès, la diversité de la population doit être préservée: étant donné que la population résultante est utilisée comme point de départ pour une autre évolution, elle doit échantillonner le plus uniformément possible l'espace faisable dans les deux premières étapes. Une procédure de nichage a été utilisée pour accomplir cet objectif.

Discussion

Cette méthode évite l'évaluation de la fonction objectif dans le domaine non faisable, ce qui peut être un grand avantage pour les problèmes où la fonction à optimiser n'est pas définie hors l'espace faisable. Mais elle peut échouer si le domaine faisable est très petit par rapport à l'espace de recherche, ou s'il a plusieurs composantes dispersées.

Par ailleurs, la procédure d'échantillonage nécessite le choix d'un ordre linéaire pour le traitement des différentes contraintes du problème. Ce choix a une grande influence sur la qualité des résultats donnés par l'algorithme. En effet, selon les expérimentations faites par les auteurs, différents choix ont donné des résultats différents, d'où la dépendance de l'approche proposée à l'ordre de l'échantillionnage.

En plus de l'ordre des traitement des contraintes, la méthode nécessite deux autres paramètres: un seuil de faisabilité ϕ et un facteur de nichage σ. Pour un problème donné, un choix non précis de leurs valeurs affecte la performance de la méthode.

2.5 Méthodes évolutionnaires: Préservation de la faisabilité des solutions

Toutes les méthodes de cette catégories ont un but commun qui est de conserver la faisabilité de la population. Elles utilisent des opérateurs de reproduction spécifiques (opérateurs fermés) qui permettent de générer à partir d'individus faisables d'autres qui sont aussi faisables.

2.5.1 Le système GENOCOP

La première version de **Genocop** (**GE**netic algorithm for **N**umerical **O**ptimization of **CO**nstrained **P**roblems) a été développée en 1991 par *Michalewicz* et *Janikow* [85]. L'idée de base de ce système est de conserver la faisabilité des individus, c'est à dire transformer des individus faisables en individus faisables.

Le système Genocop ne traite que les problèmes avec des contraintes linéaires. Il commence d'abord par diviser les contraintes en deux sous-ensembles d'égalités et d'inégalités. Les contraintes d'égalité sont éliminées par élimination d'un certain nombre de variables du problème, qui sont remplacées par des combinaisons linéaires des variables restantes. Les contraintes d'inégalité sont alors mises à jour en remplaçant les variables éliminées par leurs combinaisons linéaires. Les contraintes restantes, étant linéaires, forment un espace de faisabilité convexe. Ainsi, il est assez facile de définir des opérateurs fermés qui maintiennent la faisabilité des solutions.

Par exemple, le croisement arithmétique de deux individus faisables $\overrightarrow{x}$ et $\overrightarrow{y}$ $(a\overrightarrow{x} + (1 - a)\overrightarrow{y},\ a = \mathcal{U}[0,1])$ génère toujours, dans un domaine convexe, un individu faisable. Pour la mutation, Genocop procède en deux étapes. Il détermine d'abord le domaine faisable pour chaque composante de l'individu à muter. Les nouvelles valeurs sont alors prises à l'intérieur de ces domaines.

Les opérateurs de croisement

Pour le croisement, Genocop possède trois types d'opérateurs, et chacun d'entre eux est appliqué avec une certaine probabilité. Les deux premiers opérateurs sont le croisement arithmétique et le croisement uniforme, présentés dans la section 1.3.1 du chapitre 1. Pour obtenir le plus grand échange d'information possible entre les parents pendant le croisement uniforme, les auteurs utilisent $\alpha = 1$, et si un des enfants n'appartient pas à $\mathcal{F}$, alors α et diminué d'une petite constante $\frac{1}{\pi}$. Cette opération est répétée jusqu'à ce que les deux enfants soient faisables, ou $\alpha = 0$. Dans ce cas, les enfants sont nécessairement faisables puisqu'ils sont identiques au parents.

Un autre croisement a été ajouté, appelé croisement heuristique. Cette opération est particulière pour les trois raisons suivantes:

1. elle utilise les valeurs de la fonction objectif pour déterminer la direction de recherche,
2. elle produit un seul enfant,
3. elle peut ne pas produire d'enfant.

Cet opérateur génère un enfant $\overrightarrow{x}_3$ à partir des parents $\overrightarrow{x}_1$ et $\overrightarrow{x}_2$ en appliquant la règle suivante:

$$\overrightarrow{x}_3 = r.(\overrightarrow{x}_2 - \overrightarrow{x}_1) + \overrightarrow{x}_2,$$

où $r = \mathcal{U}[0, 1]$, et le parent $\overrightarrow{x}_2$ a une performance meilleure ou égale à celle de $\overrightarrow{x}_1$.

Si l'enfant généré n'est pas faisable, cette opération est répétée avec une autre valeur aléatoire de r. Si après w tentatives aucune solution valide n'est trouvée, l'opérateur ne retourne pas d'enfant.

Les opérateurs de mutation

Trois types de mutations sont adoptées dans le système Genocop, où chacune est appliquée avec une probabilité définie préalablement: la mutation uniforme, la mutation aux frontières et la mutation non-uniforme. La définition exacte de ces opérateurs est donnée au chapitre 1, section 1.3.2. Chacune de ses opérations est ici restreinte aux domaines de validité des variables (voir chapitre 1, section 1.3.2). Ces domaines sont calculés à partir des bornes de l'espace de recherche et des contraintes du problème.

La mutation aux frontières est utilisée afin de traiter les cas où l'optimum se trouve tout près ou sur la frontière du domaine faisable, alors que la mutation non-unifome offre au système des grandes capacités de convergence en permettant un ajustement précis des solutions, spécialement pendant les dernières étapes de l'évolution.

Exemple de résultats

Le système Genocop a été testé sur la fonction G1 ayant 13 variables et soumise à 9 contraintes linéaires, définie comme suit:

Minimiser $G1(\overrightarrow{x}) = 5x_1 + 5x_2 + 5x_3 + 5x_4 - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=1}^{1} 3x_i$,

sous les contraintes suivantes:

$2x_1 + 2x_2 + x_{10} + x_{11} \leqslant 10$,
$2x_1 + 2x_3 + x_{10} + x_{12} \leqslant 10$,
$2x_2 + 2x_3 + x_{11} + x_{12} \leqslant 10$,
$-8x_1 + x_{10} \leqslant 0$,
$-8x_2 + x_{11} \leqslant 0$,
$-8x_3 + x_{12} \leqslant 0$,
$-2x_4 - x_5 + x_{10} \leqslant 0$,
$-2x_6 - x_7 + x_{11} \leqslant 0$,
$-2x_8 - x_9 + x_{12} \leqslant 0$,

et les bornes:

$0 \leqslant x_i \leqslant 1$ pour $1 \leqslant i \leqslant 9$, et $0 \leqslant x_i \leqslant 100$ pour $10 \leqslant i \leqslant 12$ et $0 \leqslant x_{13} \leqslant 1$.

Son optimum global est $\overrightarrow{x}* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$ et $G1(\overrightarrow{x}*) = -15$. Six contraintes sont actives au niveau de la solution globale.

Genocop arrive à trouver l'optimum à chaque essai en moins de 1000 générations.

Discussion

Cette méthode a donné des bons résultats pour des problèmes dont l'espace de faisabilité est convexe. Elle a l'avantage de ne pas demander des paramètres supplémentaires. Par contre, elle peut être un peu coûteuse, vue qu'une partie de ses opérateurs de croisement nécessitent plusieurs itérations avant de générer un enfant faisable, comme le croismenet uniforme et le croisement heuristique.

Son principal inconvénient reste son incapacité de traiter les problèmes ayant des contraintes non linéaires. Une tentative pour résoudre ce problème a été faite en créant une deuxième version du système: Genocop II (voir section 2.3.3).

2.5.2 Recherche sur la frontière de la région faisable

Souvent, pour l'optimisation sous contraintes, une proportion des contraintes est active au niveau de l'optimum. Ainsi, l'optimum se situe sur la frontière du domaine réalisable $\mathcal{F}$. *Michalewicz* et *Schoenauer* ont proposé une approche originale qui permet une exploration efficace de la frontière de la région faisable [113, 114, 115]. Ils ont introduit un algorithme évolutionnaire qui part d'une population initiale constituée de points sélectionnés aléatoirement sur la frontière de $\mathcal{F}$, et les fait évoluer tout en conservant leur faisabilité grâce à des opérateurs génétiques fermés.

La frontière recherchée est supposée être une surface régulière $\mathbf{S}$ de dimension $n-1$ dans l'espace $\mathbb{R}^n$.

Les opérateurs utilisés doivent être capables de générer des points de la surface $\mathbf{S}$ (figure 2.5) et doivent respecter les conditions suivantes:

1. le croisement doit être capable de générer les points dans le voisinage des deux parents,
2. la mutation doit être ergodique et respecte le principe la forte causalité (une petite mutation ne doit générer qu'une petite modification au niveau de la performance).

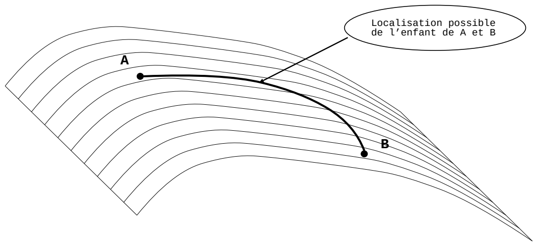


FIG. 2.5 – Opérateur de croisement de surface

L'ensemble des opérateurs proposés par *Schoenauer* et *Michalewicz* est résumé ci-dessous. Ils ont traité en premier deux cas particuliers, où la frontière de $\mathcal{F}$ forme une sphère pour le premier cas et une hyperboloïde pour le deuxième. Ils ont alors proposé des opérateurs spéciaux à ce type de surface. Ils ont ensuite proposé des opérateurs généraux, applicables sur tout type de surface, mais dont la définition dépend fortement de la surface, et leur application n'est pas toujours possible ni aisée.

Cas d'une sphère

L'opérateur de croisement sphérique produit un enfant ($\overrightarrow{z}$) à partir de deux parents ($\overrightarrow{x}$) et ($\overrightarrow{y}$), où chaque composante est calculée comme suit:

$$z_i = \sqrt{\alpha {x_i}^2 + (1-\alpha){y_i}^2},\ i \in [1, n],$$

avec $\alpha = \mathcal{U}[0,1]$.
De même, la mutation sphérique transforme (x_i) en sélectionnant 2 indices $i \neq j$ et en appliquant:

$$x_i \rightarrow p.x_i \text{ et } x_j \rightarrow q.x_j,\ \text{ avec } q = \sqrt{(\frac{x_i}{x_j})^2(1-p^2)+1}.$$

où $p = \mathcal{U}[0,1]$.

Cas d'une hyperboloïde

En présence d'une surface d'hyperboloïde, *Michalewicz* propose d'utiliser le croisement géométrique qui génère un enfant ($\overrightarrow{z}$) à partir des deux parents ($\overrightarrow{x}$) et ($\overrightarrow{y}$) comme suit:

$$(x_i)(y_i) \rightarrow ({x_i}^{\alpha}{y_i}^{1-\alpha}),\ \text{ avec } \alpha = \mathcal{U}[0,1].$$

Afin de préserver la faisabilité des solutions, la mutation choisie a un principe très simple qui mute deux composantes d'une solution, sélectionnées aléatoirement, en multipliant l'une par q et l'autre par $1/q$, où q est un facteur aléatoire respectant les bornes des variables.

Opérateurs généraux

Les opérateurs présentés ci-dessus ne sont applicables que dans des cas particuliers. Pour le cas général, les auteurs ont proposé quelques approches possibles pour concevoir des opérateurs de reproduction pour une surface quelconque de l'espace $\mathbb{R}^n$ dont la forme analytique est donnée.

Les opérateurs basés sur les courbes

Une approche pour concevoir des opérateurs de surface fermés est de se baser sur des courbes tracées sur la surface. A partir d'une courbe joignant deux points différents, on peut définir un opérateur de croisement en choisissant comme enfant un point sur cette courbe.

De même, à partir d'un faisceau de courbes partant d'un point, on peut définir un opérateur de mutation en choisissant aléatoirement une courbe du faisceau et en sélectionnant ensuite un point sur cette courbe.

Parmi les formes de courbes possibles, on cite les courbes géodésiques.

Les opérateurs basés sur les plans

Une autre façon de définir les courbes et ainsi d'autres opérateurs de surface est d'utiliser l'intersection de la surface S avec des plans bidimensionnels.

Considérons deux points A et B appartenant à S et un vecteur $\overrightarrow{v}$ non collinéaire à $\overrightarrow{AB}$ et non orthogonal au vecteur gradient au point A. Le plan défini par $(A, \overrightarrow{AB}, \overrightarrow{v})$ coupe la surface S auprès du point A, définissant ainsi une courbe de S. Si cette courbe connecte les points A et

B, un opérateur de croisement approprié peut être spécifié. Toutefois, cette procédure échoue si la connection n'est pas assurée.

D'une façon similaire, l'opérateur de mutation peut être défini en choisissant le gradient au point A au lieu du vecteur $\overrightarrow{AB}$, la distance au parent étant choisie préalablement suivant une loi de probabilité donnée (e.g. Gaussienne).

Les opérateurs paramétriques

Avec une représentation paramétrique de la surface, il est facile de spécifier des opérateurs fermés. Soit la surface S (de dimension $k < n$) définie par $x_i = s_i(t_1, \cdots, t_k)$, $t_i \in [a_i, b_i]$, pour $i = 1, \cdots, k$, où les fonctions s_i sont des fonctions régulières de $\mathbb{R}^k$ vers $\mathbb{R}$. Le choix d'un point aléatoire de S revient à choisir uniformément dans $\prod [a_i, b_i]$ les k valeurs des paramètres $t_1, \cdots, t_k$. Pour ce qui suit, on donne la relation suivante: $(x_i) = S[(t_i)]$.

L'opérateur de croisement peut être défini par:

$$S[(t_i)], S[(u_i)] \rightarrow S[(\alpha t_i + (1 - \alpha) u_i)],$$

avec $\alpha = \mathcal{U}[0, 1]$. De même, la mutation peut être donnée par:

$$S[(t_i)] \rightarrow S[(t_i + N(0, \sigma_i))]$$

où $N(0, \sigma_i)$ est une distribution normale de centre 0 et de variance σ_i, éventuellement adaptative.

Généralisation par projection sur une sphère

Tous les opérateurs généraux présentés ci-dessus sont facilement applicables dans le cas d'une frontière formant une surface sphérique. Pour profiter de la simplicité d'un telle surface, les auteurs proposent de généraliser ce cas, en utilisant le principe de codage/décodage.

Pour ramener une surface quelconque à une sphère S, *Schoenauer* et *Michalewicz* proposent d'appliquer une projection des points de la frontière vers la sphère unité du centre c, où c est un point de la région réalisable (figure 2.6), et de faire évoluer les solutions codées sur la sphère. Pour retrouver les solutions réelles, il suffit de faire la projection dans le sens inverse. La performance de la solution codée est celle de la solution réelle correspondante.

Les différentes opérations effectuées pendant une générations sont:

$$b \in \mathcal{F} \xrightarrow{\text{codage}} s \in \mathcal{S} \xrightarrow{\text{évolution}} s' \in \mathcal{S} \xrightarrow{\text{décodage}} b' \in \mathcal{F}$$

Procédure de codage

Supposons que l'espace $\mathcal{F}$ est étoilé. A partir d'un point faisable x_f et un autre non faisable x_u, on cherche le point de la surface **S** défini par l'intersection du segment passant par les point x_f et x_u et la surface de frontière grâce à une méthode dichotomique. Le degré de précision des recherches est choisi à l'avance.

Exemple de résultats

Michalewicz a testé les opérateurs "hyperboloïdaux" sur le problème suivant:

$$G2(\overrightarrow{x}) = \left| \frac{\sum_{i=1}^{n} cos^4(x_i) - 2 \prod_{i=1}^{n} cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right|,$$

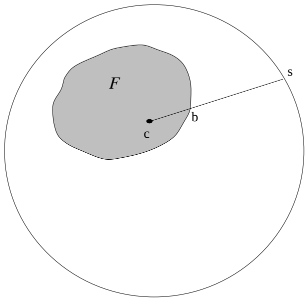


FIG. 2.6 – *Technique de généralisation: projection des points de la frontière sur la sphère unité de centre c.*

soumis aux contraintes suivantes:

$\prod_{i=1}^{n} x_i \geqslant 0.75$, $\sum_{i=1}^{n} x_i \leqslant 7.5n$, et les bornes $0 \leqslant x_i \leqslant 10$ for $1 \leqslant i \leqslant n$.

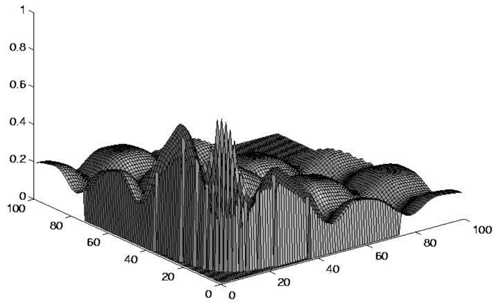


FIG. 2.7 – *Courbe de la fonction G2 testée. Les solutions non réalisables sont mises à zéro.*

La fonction G2 est non linéaire et son maximum global est inconnu. Ce problème montre une certaine difficulté dans la localisation de la frontière, illustrée par la figure 2.7, où les points irréalisables sont mis à zéro. Seule la première contrainte est active au niveau de l'optimum ($\prod_{i=1}^{n} x_i = 0.75$). la frontière définie par cette contrainte forme un hyperboloïde.

Ce sont alors les opérateurs spécifiques à ce genre de surface qui ont été utilisés pour résoudre ce problème. Outre ces opérateurs spéciaux, les auteurs ont testé en parallèle la méthode généralisée.

La figure 2.8 montre que les opérateurs géométriques (méthode spéciale) ont illustré en moyenne une meilleure performance que les opérateurs sphériques avec projection (méthode générale), mais les deux méthodes ont réussi à obtenir de très bons résultats, où elle a donné comme meilleure et pire solution, avec la méthode spéciale, 0.803553 et 0.802964 respectivement.

Discussion

La stratégie d'utiliser des opérateurs spéciaux pour explorer la frontière est robuste quand elle est applicable. Cependant, elle ne traite que les problèmes dont l'optimum est sur la frontière. De

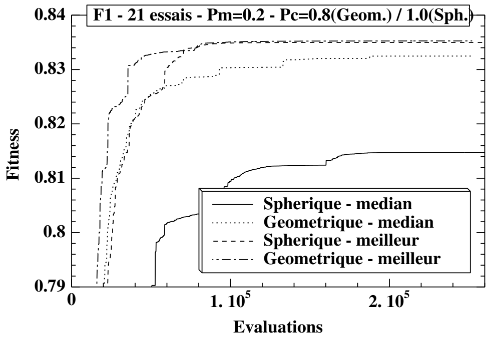


FIG. 2.8 – *Les meilleures solutions et les solutions médianes données par les opérateurs géométriques et les opérateurs sphériques (méthode générale) pendant la résolution du problème G2.*

plus, les opérateurs généraux sont compliqués à mettre en oeuvre, spécialement ceux basés sur les plans.

Entres autre, la méthode généralisée est coûteuse en temps de calcul, vue qu'elle nécessite le passage par une méthode dichotomique pour la recherche de la frontière (complexité$\simeq$ Log(précision)).

2.5.3 "Homomorphous mapping"

Proposée en 1999 par *Koziel* et *Michalewicz* [73], cette méthode utilise des décodeurs afin de transformer un problème sous contraintes à un autre sans contraintes. Elle fait évoluer une population d'individus codés, où chacun correspond à une solution dans l'espace de recherche réel. Pour gérer les contraintes avec les décodeurs, la technique doit satisfaire les conditions suivantes:

1. pour chaque solution $s \in \mathcal{F}$, il y a une solution codée d,
2. chaque solution codée d correspond à une solution faisable s,
3. toutes les solutions dans $\mathcal{F}$ doivent être représentées par le même nombre de codes,
4. la procédure de codage/décodage T ne doit pas être trop complexe et doit être rapide en temps de calcul,
5. tout petit changement dans la solution codée ne doit générer qu'un tout petit changement dans la solution réelle correspondante.

Le "homomorphous mapping" proposée par *Koziel* et *Michalewicz* est une technique de codage/décodage entre un espace de recherche faisable $\mathcal{F}$ quelconque et le cube unitaire de dimension n: $[-1, 1]^n$. Elle est capable de prendre en compte les espaces convexes et non convexes, avec une étape de codage/décodage supplémentaire pour le deuxième cas.

Espace convexe

Il est relativement facile d'établir une bijection entre un espace de recherche faisable convexe quelconque et un cube de dimension n: $[-1, 1]^n$. Il suffit d'établir une procédure de projection dans les deux sens (codage/décodage) comme s'est présentée dans la figure 2.9.

Description:

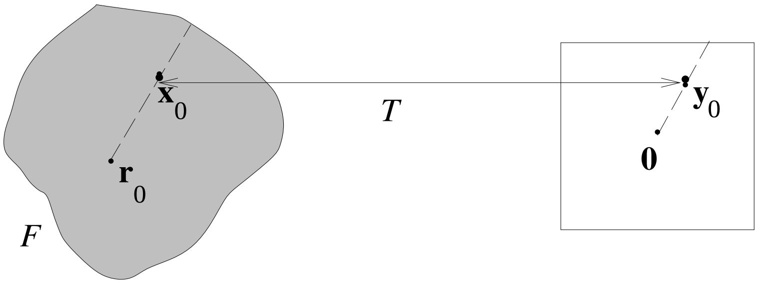


FIG. 2.9 – *Exemple de projection de points entre le cube* $[-1,1]^n$ *et un espace convexe* $\mathcal{F}$ *(cas bidimensionnel).*

Soit $\overrightarrow{y}_0 \in [-1,1]^n$ la solution codée du point $\overrightarrow{x}_0 \in \mathcal{F}$. Le point $\overrightarrow{y}_0$ définit un demi-segment ayant comme origine le centre du cube $\overrightarrow{O}$ et coupe la frontière du cube au point $\overrightarrow{y}_M$ (c.f. figure 2.9). Si on considère un point de référence $\overrightarrow{r}_0 \in \mathcal{F}$, le point codé $\overrightarrow{y}_0 \in \mathcal{F}$ correspondant à $\overrightarrow{x}_0$ est défini par:

$\overrightarrow{y}_0 = (\overrightarrow{x}_0 - \overrightarrow{r}_0).\tau$, où $\tau = \frac{||\overrightarrow{y}_M||}{||\overrightarrow{x}_M - \overrightarrow{r}_0||}$. $\overrightarrow{y}_M$ est déterminé avec une procédure de recherche dichotomique comme celle décrite dans le paragraphe 2.5.2.

Cette technique satisfait toutes les conditions d'un bon décodeur citées ci-dessus, et peut être améliorée avec un choix optimal de $\overrightarrow{r}_0$.

Espace non convexe

La technique proposée pour traiter les espaces faisables non convexes est une généralisation de celle décrite ci-dessus. Elle est capable de transformer le cube $[-1,1]^n$ en l'espace faisable $\mathcal{F}$ du problème indépendamment de sa forme. Dans ce cas, tout segment d'origine un point de référence $\overrightarrow{r}_0 \in \mathcal{F}$ et de direction quelconque peut couper la frontière de l'espace faisable en plus d'un point (figure 2.10).

Description:
soit φ la généralisation de la méthode de décodage T précédemment décrite.
Soit L le segment entre un point de référence quelconque $\overrightarrow{r}_0 \in \mathcal{F}$ est un point $\overrightarrow{s}$ de la frontière de l'espace de recherche S défini par: $L(\overrightarrow{r}_0, \overrightarrow{s}) = \overrightarrow{r}_0 + t.(\overrightarrow{s} - \overrightarrow{r}_0)$, pour $0 \leqslant t \leqslant 1$. Si l'espace réalisable $\mathcal{F}$ est convexe, l'intersection entre L et la frontière de $\mathcal{F}$ est exactement un point, pour certain $t_0 \in [0,1]$. Par conséquent, $\varphi : [-1,1]^n \to \mathcal{F}$ est définie comme suit:

$$\varphi(\overrightarrow{y}) = \begin{cases} \overrightarrow{r}_0 + y_{max}.t_0.(g(\overrightarrow{y}/y_{max} - \overrightarrow{r}_0)) & if \overrightarrow{y} \neq \overrightarrow{O} \\ \overrightarrow{r}_0 & if \overrightarrow{y} = \overrightarrow{O} \end{cases}$$

où $y_{max} = max_{i=1}^{n}|y_i|$. La figure 2.9 illustre cette transformation.

Si l'espace $\mathcal{F}$ est non convexe, L peut avoir plusieurs points d'intersection avec $\mathcal{F}$, et dans ce cas, au lieu d'avoir un seul intervalle de faisabilité $[0, t_0]$, on peut avoir plusieurs intervalles: $[t_1, t_2], \cdots, [t_{2k-1}, t_{2k}]$, où k est le nombre de sous-intervalles de faisabilité. Ainsi, il est nécessaire d'introduire une autre fonction de codage δ qui transforme l'intervalle $[0,1]$ en un ensemble d'intervalles $[t_{2i-1}, t_{2i}]$, et une autre fonction de décodage γ qui fait la transformation inverse:

$$\delta : (0,1] \to \cup_{i=1}^{k}(t_{2i-1}, t_{2i}].$$
$$\gamma : \cup_{i=1}^{k}(t_{2i-1}, t_{2i}] \to (0,1]$$

La définition générale de la technique de transformation est identique à la précédente (cas d'un espace convexe), à part qu'on applique un codage/décodage supplémentaire (δ et γ), illustré dans la figure 2.10

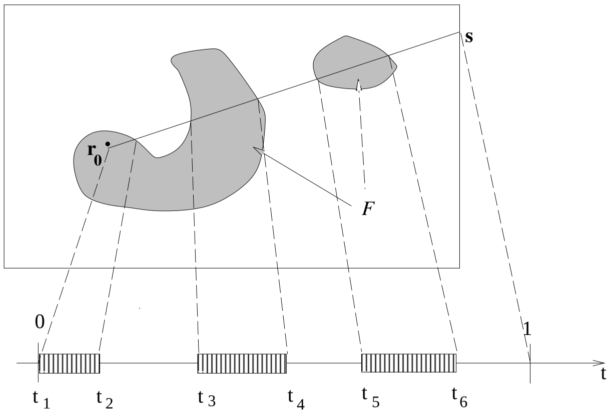


FIG. 2.10 – *Un segment de ligne dans un espace $\mathcal{F}$ non-convexe et ses sous intervalles correspondant, donnés par le codage δ (cas bidimensionnel).*

Exemple de résultats

Cette technique a été testée sur les 11 cas test (de G1 à G11) proposées dans [87]. La définition complète des ces fonctions est donnée dans l'annexe A. Deux séries d'expérimentations ont été réalisées pour tester cette méthode. Pour la première série, le point de référence $\overrightarrow{r_0}$ est choisi aléatoirement. Pour la deuxième, les points de référence sont les meilleures solutions données par les premières expériences. Le tableau ci-dessous récapitule l'ensemble des résultats après la deuxième série de tests.

	G1	G2	G3	G4	G6	G7	G8	G9	G10	G11
b	-14.7184	0.79486	0.9978	-30661.5	-6944.4	25.090	0.095825	681.72	7321.2	0.75
a	-14.6478	0.78722	0.997	-30653.1	-6720.4	25.545	0.0871551	682.56	7498.6	0.75
w	-14.5732	0.78279	0.996	-30645.6	-6390.6	26.182	0.029143	683.8	7685.8	0.75

TAB. 2.2 – *Résultat de l'application de la méthode de* "Homomorphous mapping" *sur 10 cas test, donnant la meilleure (b), la moyenne (a) et la pire (w) solution sur 20 essais effectués pour chaque fonction. Les résultats de la fonction G5 ne sont pas présentés parce qu'ils ne sont pas significatifs.*

Les résultats donnés par cette méthode sont moyens mais leur qualité peut être améliorée considérablement avec un choix optimal du point de référence. Toutefois, la méthode a échoué dans la résolution du cas test G5 qui est soumis à des contraintes d'égalité et d'inégalité simultanément.

Discussion

C'est une méthode originale mais difficile à mettre en oeuvre et qui nécessite beaucoup de temps de calcul, spécialement pendant la procédure de recherche des points de la frontière (voir la discussion de la méthode de recherche surs la frontière, section 2.5.2). Entre autres, la technique de codage généralisée pour les espaces non-convexes peut violer le cinquième point des conditions nécessaires de validité d'un décodeur portant sur la forte causalité. En effet, une petite extension dans l'un des sous intervalles codés génère une grande modification au niveau des solutions décodées. De plus, numériquement, il est très difficile de trouver les limites des sous intervalles dans $[0, 1]$ correspondant aux différentes régions dans l'espace faisable, due à la difficulté de localiser toutes les composantes connexes de $\mathcal{F}$. Ainsi, l'application de cette stratégie sur des problèmes réels s'avère presque impossible.

2.6 Méthode évolutionnaires: méthodes Hybrides

L'objectif général des méthodes hybrides est de séparer la fonction objectif des contraintes du problème. On trouve dans la littérature deux approches possibles pour réaliser cette séparation. La première traite les contraintes avec des procédures d'optimisation numériques déterministes, alors que la fonction objectif est toujours optimisée par un algorithme évolutionnaire. Avec la deuxième approche, c'est l'algorithme génétique lui même qui fait cette séparation, soit en introduisant les techniques du multi-objectif, soit en adoptant le modèle de la co-évolution. Ces différentes stratégies sont présentées successivement dans les sections suivantes.

2.6.1 Utilisation des procédures d'optimisation déterministes

Il est relativement facile de développer des méthodes qui combinent les techniques de calcul évolutionnaire avec les procédures déterministes pour la résolution des problèmes d'optimisation numériques. En 1992, *Waagen et al.* [132] ont combiné une méthode de calcul évolutionnaire basée sur une représentation réelle des solutions [35] avec la méthode de *"direction set"* de Hook-Jeeves. La méthode hybride a été testée sur 3 fonctions, mais elles n'étaient pas soumises à des contraintes.

En 1995, *Myung et al.*[88] ont considéré une approche similaire et ils l'ont testé sur des problèmes avec contraintes. De même, ils ont combiné une technique de calcul évolutionnaire basée sur la représentation réelle des solutions avec une autre méthode déterministe développée par Maa et Shanblatt en 1992 [78]. Cependant, alors que la méthode de *Waagen et al.* [132] a incorporé l'algorithme de *"direction set"* comme un opérateur génétique spécifique au problème, *Myung et al.* [88] ont divisé tout le processus d'optimisation en deux phases. Durant la première phase, l'algorithme évolutionnaire optimise la fonction:

$$eval(\overrightarrow{x}) = f(\overrightarrow{x}) + \frac{s}{2}\left(\sum_{j=1}^{m} {f_j}^2(\overrightarrow{x})\right),$$

où s est une constante et f_j est la fonction de la $j^{\text{ème}}$ contrainte du problème ($f_j = g_j$ si $1 \leqslant j \leqslant q$ et $f_j = h_j$ si $q < j \leqslant m$). Après la fin de cette étape, une deuxième phase d'optimisation est appliquée sur les meilleurs solutions trouvées durant la première phase [78], afin de les améliorer.

Cette phase est répétée jusqu'à ce que le système suivant atteint l'équilibre (voir section 2.2.5):

$$\overrightarrow{x}' = -\nabla f(\overrightarrow{x}) - \left[\sum_{j=1}^{m} \nabla f_j(\overrightarrow{x})(sf_j(\overrightarrow{x}) + \lambda_j)\right], \tag{2.4}$$

où les multiplicateurs de Lagrange λ_j sont mis à jour tel que: ${\lambda_j}' = \epsilon.sf_j$ avec ϵ est une constante ayant une petite valeur positive.

Exemple de résultats

La méthode hybride de *Myung et al.* [88] a été testée avec succès sur quelques problèmes sous contraintes. Par exemple, un des cas test était de minimiser:

$$G11(\overrightarrow{x}) = x_1^2 + (x_2 - 1)^2,$$

soumise à la contraintes non linéaire suivante:

$$x_2 - x_1^2 = 0,$$

et les bornes: $-1 \leqslant x_i \leqslant 1$, $i = 1, 2$.

Deux solutions globales sont connues pour ce problème qui sont: $\overrightarrow{x}^* = (\pm 0.70711, 0.5)$, avec $G11(\overrightarrow{x}^*) = 0.75000455$.

Dès la première phase du processus d'évolution, l'algorithme converge rapidement (100 générations) vers un des deux optima $(\pm 0.70711, 0.5)$. La seconde phase ne fait que conduire le système à l'équilibre. Dans le cas de ce problème, elle n'a pas servi à la localisation de l'optimum.

Discussion

On ne peut pas établir un jugement plus précis sur la performance de la méthode, étant donné qu'elle n'a pas été testée sur des problèmes de dimensions plus élevées. Par contre, elle ne peut pas être appliquée sur des problèmes où la fonction objectif ou une des fonctions des contraintes n'est pas dérivable, puisqu'elle nécessite le calcul du gradient de toute les fonctions du problème dans le système 2.4.

2.6.2 Approche multi-objectif

L'idée avec l'approche multi-objectif est de minimiser simultanément la fonction objectif et les violations des différentes contraintes du problème. Quelques méthodes considèrent la valeur de la fonction objectif ainsi que les violations v_j $(j = 1, \cdots, m)$ comme les éléments d'un vecteur de taille $(m+1)$, et l'algorithme évolutionnaire aura alors pour mission la minimisation des différentes composantes de ce vecteur $\overrightarrow{v} = (f, v_1, \cdots, v_m)$.

Méthode de *Parmee* et *Purchase*

L'approche multi-objectif a été introduite par *Parmee* et *Purchase* en 1994 [92] dans le domaine de développement des techniques de l'optimisation de formes sous contraintes. Ils utilisent la méthode multi-objectif proposée par *Schaffer* [111], nommée "Vector Evaluated Genetic Algorithm" (VEGA) (section 1.7.2 du chapitre 1). Le principe de leur méthode est de créer un ensemble de régions de recherche locales dans l'espace faisable, générés à partir de points réalisables. Pour obtenir un point faisable, ils appliquent VEGA en considérant les violations comme des objectifs. Dès qu'un point faisable est repéré, une autre procédure est lancée pour créer un hypercube autour de ce point, constituant une zone de recherche locale. L'objectif est que la grande proportion de l'hypercube soit dans l'espace faisable.

Une fois l'ensemble des hypercubes créé, un autre GA est lancé pour l'optimisation de la fonction objectif du problème posé. La performance de chaque hypercube est une combinaison de la fonction objectif et les degrés de violation des contraintes par des points fixes de sa surface.

Méthode de *Surry et al.*

Une autre technique a été proposée par *Surry et al.* en 1995 [126], appelée COMOGA ("Constrained Optimization by Multi-Objective Genetic Algorithms"), qui traite les contraintes comme des critères d'un problème multi-objectif, et optimise en parallèle la fonction objectif d'une manière classique. Pour se faire, elle associe à chaque solution un rang de Pareto R et une fitness f:

$$I_R(\overrightarrow{x}) = (R_{(c_1,\cdots,c_m)}(\overrightarrow{x}), f(\overrightarrow{x})).$$

Le rang d'une solution n'est calculé qu'avec les degrés de violation des contraintes, et il est égale au nombre de points que la solution correspondante domine (principe du Classement Non Dominé proposé par *Fonesca* et *Fleming* en 1993 [37], voir section 1.7.2 du chapitre 1). La sélection des survivants pour la génération suivante se fait en deux étapes: d'abord, $p_{cost} \times N$ individus sont sélectionnés proportionnellement à leurs performances f en utilisant un tournoi, ensuite, le reste des individus ($(1 - p_{cost}) \times N$) sont sélectionnés linéairement selon leurs rangs R.

Pour éviter la convergence vers une solution irréalisable, la valeur de p_{cost} est adaptée à chaque génération selon le taux d'individus faisables dans la population courante par rapport à un taux de référence τ. Le schéma de cette méthode est résumé dans les étapes suivantes:

1. Calculer les degrés de violation des contraintes pour toutes les solutions.
2. Calculer les rangs de Pareto R pour toutes les solutions en se basant sur les degrés de violation.
3. Évaluer la fonction de performance f.
4. Sélectionner une proportion p_{cost} des parents en se basant sur f, et le reste en se basant sur R.
5. Appliquer les opérateurs de croisement et de mutation.
6. Ajuster p_{cost}: si la proportion des individus faisables est inférieure au taux de référence τ, diminuer p_{cost}: $p_{cost} \leftarrow (1 - \epsilon)p_{cost}$, sinon, augmenter p_{cost}: $p_{cost} \leftarrow 1 - (1 - p_{cost})(1 - \epsilon)$, où $0 < \epsilon << 1$.

Exemple de résultat

La méthode a été testée sur un problème de conception d'un réseau de gaz (disposition et type des tuyaux) [126], et elle a donné des résultats de bonne qualité.

Cependant, elle n'a pas enregistré le même degré de précision pour les quelques fonctions de référence testées par *Surry* et *Radcliffe* dans [127]. Elle a été appliquée sur les cas test G1, G7, G9 et G10, définies respectivement, dans les paragraphes 2.5.1, 2.3.1, 2.3.5, 2.4.1. Avec $\tau = 0.1$, $\epsilon = 0.1$ et $P_{cost} = 0.5$, COMOGA a donné comme valeur médiane, sur 10 essais, -14.996 pour G1, 24.509 pour G7, 680.69 pour G9 et 7556.85 pour G10. Ces résultats sont satisfaisants, mais COMOGA n'a réussi à localiser l'optimum pour aucun des 4 problèmes. Par contre, toutes les solutions retournées sont faisables.

Discussion

La prise en compte des contraintes par l'approche multi-objectif est récente. Les résultats préliminaires donnés par COMOGA prouvent que cette technique est prometteuse. L'inconvénient de COMOGA est le manque de précision pour les problèmes réels.

D'après *Surry* et *Radcliffe*, il est possible d'améliorer la performance de leur algorithme en introduisant une technique de nichage pour maintenir la diversité.

2.6.3 La co-évolution (90)

L'introduction du modèle de co-évolution dans les problèmes d'optimisation sous contraintes a été publié pour la première fois par *Paredis* en 1994 [90], qui introduit la méthode **CCS**: "Co-evolutionary Constraint Satisfaction". Avec ce modèle, une population de solutions potentielles co-évolue avec une population de contraintes: les solutions les plus adaptées satisfont plus de contraintes, alors que les contraintes les plus adaptées sont violées par plus de solutions. Ceci signifie que les individus de la population des solutions sont considérés dans tout l'espace de recherche S, sans distinction entre faisables et infaisables. L'évaluation des individus est déterminée sur la base des mesures des violations des contraintes: les meilleures contraintes (contraintes actives) contribuent plus dans l'évaluation des solutions.

Le cœur de **CCS** ressemble à un algorithme génétique standard, avec deux principales nouveautés apportées à la sélection et l'évaluation des solutions.

Paredis s'est inspiré du modèle de la "proie/prédateur" pour définir la stratégie de sélection des individus pour la génération suivante: les survivants parmi les membres d'une des deux populations (pression de sélection) dépendent de la performance des membres de l'autre population. Autrement dit, il y a une interaction inverse des performances entre les deux populations.

Pour le calcul de la performance des solutions, une notion de *rencontre* solution-contrainte est introduite, pendant laquelle une solution reçoit une récompense (+1) si elle satisfait la contrainte rencontrée, et une pénalisation (-1) sinon. Par contre, chaque contrainte appelée reçoit une récompense si la solution appelante la viole, et une pénalisation sinon. Une mémoire est alors associée à chaque individu, regroupant l'ensemble des récompenses et pénalisations reçues pendant les m dernières générations (m: paramètre de la méthode, choisi préalablement), et dont la somme constitue sa performance.

Dans des travaux plus récents, Paredis propose d'améliorer le processus de co-évolution en essayant d'équilibrer le taux d'évolution des deux populations. En effet, la population des solutions a tendance à évoluer beaucoup plus rapidement que celle des contraintes. Pour régler ce problème, il est possible de définir une probabilité d'évolution pour chaque population, mais pour assurer équilibre, il faut choisir les probabilités adéquates. Une autre technique proposée par *Paredis* récemment [91], appelée méthode X, adapte les taux d'évolution de chaque population au cours des générations selon le nombre de solutions valides (satisfaisant les solutions de l'autre population).

Exemple de résultats

Cette méthode s'avère efficace si le problème est soumis à un grand nombre de contraintes. Par exemple, *Paredis* a testé sa méthode sur le problème de "n-reines", qui consiste à placer n reines sur un échiquier de $n \times n$ de telle façon que deux reines ne peuvent pas s'attaquer entre elles (leurs numéros de lignes et colonnes et leurs diagonales sont différents):

$$\begin{array}{lll} x_i \neq x_j & i \neq j & \text{; contraintes lignes-colonnes} \\ |x_i - x_j| \neq |i - j| & i \neq j & \text{; contraintes diagonales} \end{array}$$

Pour n=50, le problème est soumis à 100 contraintes.

CCS est arrivé à trouver des solutions qui satisfont toutes les contraintes pour ce problème difficile.

Discussion

Cette méthode a montré une grande robustesse dans la résolution d'un problème combinatoire difficile. Mais elle devient peu significative si le nombre de contraintes du problème posé est très petit. C'est pourquoi elle n'a pas été testée sur des fonctions de référence. On ne peut pas donc comparer sa performance avec les autres méthodes évolutionnaires de prise en compte des contraintes.

2.6.4 Autres techniques

D'autres approches basées sur des nouvelles générations d'algorithmes évolutionnaires ont été proposées, mais elles sont encore en phase de croissance. Par exemple, il y a eu quelques travaux réalisés par*Bilchev* et *Parmee* [13] pour la généralisation du concept des colonies de fourmis [14] au problèmes numériques. Les quelques expériences réalisées dans ce domaine sont très prometteuse. Une autre orientation de recherche se dirige vers les algorithmes culturels [99].

2.7 Résumé des différents résultats présentés

Afin de pouvoir effectuer des études comparatives, nous avons regroupé l'ensemble des tests d'évaluation des méthodes cités dans ce chapitre, précisément ceux concernant les fonctions de référence présentées dans l'annexe A. Tous les résultats proviennent de la littérature. Ce sont des tests effectués par les auteurs de la méthode ou publiés par des auteurs qui ont implanté la méthode concernée afin de tester son efficacité.

Les caractéristiques des fonctions test sont données dans la tableau 2.7. Pour chaque cas, on donne le nombre de variables, le type de la fonction f, un rapport relatif entre la taille de l'espace faisable et celle de l'espace de recherche, le nombre de contraintes de chaque catégories (inégalité linéaires IL, inégalité non linéaires IN et égalités non linéaires EN), ainsi que le nombre de contraintes actives à l'optimum.

La désignation des méthodes est donnée dans la liste suivante:

- M1(f): la méthode de "death penalty", présentée dans la section 2.3.1. La mention (f) indique que l'algorithme part d'une population initiale faisable.
- M2: la méthode des pénalités statiques, présentée dans la section 2.3.2.
- M3: la méthode des pénalités dynamiques, présentée dans la section 2.3.3.
- M4: la méthode des pénalités analytiques ou Genocop II, présentée dans la section 2.3.3.
- M5: la méthode de supériorité des individus faisables donnée par Powell et Skolnick, présentée dans la section 2.3.5.
- M6: la méthode de supériorité des individus faisables donnée par Deb, présentée dans la section 2.3.5.

- M7: la méthode de *Stochastic Ranking* de Runarsson et Yao, présentée dans la section 2.3.7.
- M8: la méthode de réparation des individus faisables ou Genocop III, présentée dans la section 2.4.1.
- M9: la méthode de “behavioral memory”, présentée dans la section 2.4.2.
- M10: la méthode de recherche sur la frontière de l'espace faisable (*Boundary Operators*) présentée dans la section 2.5.2.
- M11: la méthode de “homomorphous mapping”, présentée dans la section 2.5.3.
- M12: la méthode multi-objectif de Surry et al. (COMOGA) présentée dans la section 2.6.2.
- M13: la méthode hybride utilisant des méthodes d'optimisation déterministes, présentée dans la section 2.6.1.

Chacune de ces méthodes a été testée sur au moins une fonction de référence. Les résultats correspondants sont regroupés dans le tableau 2.4. Les méthodes M10 et M13 ont été testées sur un seul cas test de référence. Leurs résultats ne sont pas présentés dans le tableau 2.4, faute de place.

M10 a été appliquée sur le cas test G2, et a donné comme meilleure et pire solution 0.803553 et 0.802964, respectivement. M13 n'a été appliquée que sur G11, et a réussi à trouver l'optimum 0.75 pour tous les essais.

Fonction	nb variables	Type de f	ρ	IL	EN	IN	a
G1	13	quadratique	0.0111%	9	0	0	6
G2	k	non linéaire	99.8474%	0	0	2	1
G3	k	polynomiale	0.0000%	0	1	0	1
G4	5	quadratique	52.1230%	0	0	6	2
G5	4	cubique	0.0000%	2	3	0	3
G6	2	cubique	0.0066%	0	0	2	2
G7	10	quadratique	0.0003%	3	0	5	6
G8	2	non linéaire	0.8560%	0	0	2	2
G9	7	polynomiale	0.5121%	0	0	4	2
G10	8	linéaire	0.0010%	3	0	3	6
G11	2	quadratique	0.0000%	0	1	0	1

TAB. 2.3 – *Résumé des 11 cas test. Le ration $\rho = |F|/|S|$ est déterminé manuellement en générant 1 000 000 points aléatoires dans l'espace de recherche S et en testant à chaque fois si le point appartient à F ou non. Pour G2 et G3, le nombre de variables k a été fixé à 50. IL, EN et IN représentent le nombre des inégalités linéaires, des équations et des inégalités non linéaires, respectivement.*

Discussion

Il est difficile établir une comparaison adéquate entre les différentes méthodes de prise en compte des contraintes pour les deux raisons suivantes:

- Certaines de ces méthodes n'ont pas été testées sur des fonctions de référence permettant d'évaluer leurs résultats. En plus, quelques unes n'ont été appliquées que sur un ou deux problèmes, ce qui n'est pas suffisant pour juger la performance de la technique.

– Ces approches ont été implantées sur des algorithmes évolutionnaires différents. Un jugement justifié sur leur robustesse dans la résolution de l'ensemble des cas test de référence nécessite leur implantation sur la même plate-forme. C'est l'objectif de nos futurs travaux dans ce domaine. Une partie de ces méthodes (spécialement celles basées sur les fonctions de pénalité) seront sélectionnées et implantées sur une même base, dans le but d'effectuer une étude comparative.

Si on ne tient pas compte de ces points, le tableau 2.4 montre que les meilleurs résultats sont ceux donnés par la méthode de classement stochastique (*stochastic ranking*) présentée dans la section 2.3.7.

Entre autres, le tableau 2.4 montre la majorité des résultats correspondent aux méthodes de la première catégorie (basées sur les fonctions de pénalisation). Ces méthodes sont généralement les plus utilisées, vue la simplicité de leur implantation. Cependant, la méthode de "death penalty" et celle des pénalités statiques ne sont pas très utilisées, à cause de la dépendance de la première au schéma d'initialisation et le grand nombre de paramètres à définir pour la deuxième. Ce sont alors les pénalités dynamiques et adaptatives qui sont souvent choisies par les utilisateurs.

Au cours de la présentation de ces méthodes, nous avons en parallèle analysé les inconvénients de chacune. Pour les pénalités dynamiques, c'est surtout le caractère croissant des cœfficients de pénalisation qui peut nuire au processus d'exploration. Il peut causer la perte rapide de la diversité de la population et pousser l'algorithme vers la convergence prématurée. Cet inconvénient disparaît avec les méthodes de pénalités adaptatives, mais ces dernières ne tiennent pas compte de l'état courant et le degré de faisabilité de la population. Dans le chapitre 4, nous proposons une nouvelle technique de pénalisation adaptative qui permet de résoudre ces deux problèmes.

Cas Test	opt. Exact		Catégorie (1)							Catégorie (2)		Catégorie (3)	
			M1(f)	M2	M3	M4	M5	M6	M7	M8	M9	M11	M12
$G1$	-15.000	b	**-15**	-15.002	**-15**	**-15**	**-15**	–	**-15**	**-15**	**-15**	-14.72	-14.997
		m	-14.999	-15.002	**-15**	**-15**	**-15**	–	**-15**	**-15**	**-15**	-14.46	-14.996
		w	-13.616	-15.001	-14.999	**-15**	-14.999	–	**-15**	**-15**	-14.998	-14.05	-14.994
		c	0, 0, 0	0, 0, 4	0, 0, 0	0, 0, 0	0, 0, 0	–	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
$G2$	0.803603	b	–	–	–	–	–	–	0.803515	–	–	0.79506	–
		m	–	–	–	–	–	–	0.781975	–	–	0.79176	–
		w	–	–	–	–	–	–	0.726288	–	–	0.78427	–
$G3$	1	b	–	–	–	–	–	–	**1**	–	–	0.9978	–
		m	–	–	–	–	–	–	**1**	–	–	0.997	–
		w	–	–	–	–	–	–	**1**	–	–	0.996	–
$G4$	-30665.5	b	–	–	–	–	–	–	**-30665.5**	–	–	-30662.5	–
		m	–	–	–	–	–	–	**-30665.5**	–	–	-30643.8	–
		w	–	–	–	–	–	–	**-30665.5**	–	–	-30617.0	–
$G5$	5126.49	b	–	–	–	–	–	–	**5126.49**	–	–	–	–
		m	–	–	–	–	–	–	5128.88	–	–	–	–
		w	–	–	–	–	–	–	5142.47	–	–	–	–
$G6$	-6961.81	b	–	–	–	-6955.015	–	–	**-6961.81**	–	–	-6901.5	–
		m	–	–	–	–	–	–	-6875.94	–	–	-6191.2	–
		w	–	–	–	–	–	–	-6350.26	–	–	-4235.7	–
$G7$	24.306	b	25.653	24.690	25.48	18.917	17.388	24.372	**24.3**	25.883	–	25.132	24.340
		m	27.116	29.258	26.905	24.41	22.932	24.409	24.37		–	26.619	24.509
		w	32.477	36.060	42.358	44.302	48.866	25.075	24.64		–	38.682	24.710
		c	0, 0, 0	0, 1, 1	0, 0, 0	0, 1, 0	1, 0, 0		0, 0, 0		–	0, 0, 0	0, 0, 0
$G8$	0.95825	b	–	–	–	–	–	–	**0.095825**	–	–	0.095825	–
		m	–	–	–	–	–	–	**0.095825**	–	–	0.087155	–
		w	–	–	–	–	–	–	**0.095825**	–	–	0.029143	–
$G9$	680.630	b	680.847	680.771	680.787	680.642	680.805	680.634	**680.63**	680.640	680.836	681.43	680.663
		m	681.826	681.262	681.111	680.718	682.682	680.642	680.65	–	680.175	682.18	680.690
		w	689.417	689.660	682.798	680.955	685.738	680.51	680.76	680.889	685.640	682.88	680.755
		c	0, 0, 0	0, 0, 1	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0		0, 0, 0	0, 0, 0	0, 0, 0
$G10$	7049.331	b	7872.948	2282.723	3117.22	7377.976	2101.367	7060.221	**7054.31**	7286.65	7485.667	7215.8	7081.43
		m	8559.23	2449.798	4213.497	8206.151	2101.411	–	7559.19	–	8271.292	9141.7	7556.85
		w	8668.648	2756.679	6056.211	9652.901	2101.551	–	8835.65	–	8752.421	11894.5	8322.51
		c	0, 0, 0	0, 3, 0	0, 3, 0	0, 0, 0	1, 2, 0	–	0, 0, 0	–	0, 0, 0	0, 0, 0	0, 0, 0
$G11$	0.75	b	–	–	–	–	–	–	**0.75**	–	–	**0.75**	–
		m	–	–	–	–	–	–	**0.75**	–	–	**0.75**	–
		w	–	–	–	–	–	–	**0.75**	–	–	**0.75**	–

TAB. 2.4 – *Récapitulatif des meilleurs résultats publiés pour 11 cas test. La séquence des 3 chiffres dans la colonne c indique le nombre de contraintes violées par la solution médiane par plus de 1.0, plus de 0.1 et plus de 0.001, respectivement. L'abscence de cette ligne signifie que les solutions retournées sont faisables.*

Chapitre 3

La mutation logarithmique

Résumé

Dans le chapitre précédent, nous avons présenté les principales stratégies de prise en compte des contraintes dans les problèmes d'optimisation. Ces méthodes ont deux grands objectifs: ramener les individus de la population dans l'espace faisable et aider l'algorithme à explorer et exploiter efficacement cet espace. La majorité des méthodes proposent des solutions pour accomplir le premier objectif, mais trouvent des difficultés pour accomplir le deuxième. Souvent, pour assurer l'évolution de la population, elles utilisent des opérateurs de reproduction standards. Dans ce chapitre, nous montrons que, même si le domaine faisable n'est pas trop petit, la capacité d'exploration de ces opérateurs n'est pas optimale pour la résolution des problèmes sous contraintes. Nous proposons aussi un opérateur de mutation logarithmique, conçu pour explorer, à la fois localement et globalement, l'espace de recherche pour un domaine borné. Cet opérateur montre un comportement très efficace dans la résolution du problème de la *sphère* tronquée (une version avec contraintes de la fonction *sphère*), par comparaison avec les opérateurs standards, grâce à sa capacité de maintenir la diversité de la population tout au long de l'évolution, tout en assurant une bonne exploitation des meilleures solutions.

Associé à l'opérateur de croisement BXL-0.5 et à un opérateur de sélection spécifique capable de prendre en compte les contraintes, l'opérateur de mutation logarithmique permet aux algorithmes évolutionnaires d'atteindre souvent une meilleure performance, sur plusieurs cas test de référence, que celle atteinte en utilisant d'autres opérateurs standards.

3.1 Introduction

Nous commençons par rappeler la définition d'un problème d'optimisation sous contraintes:

$$\text{optimiser } f(\overrightarrow{x}), \overrightarrow{x} = (x_1, \dots, x_n) \in \mathcal{F} \subseteq \mathcal{S} \subseteq \mathbb{R}^n,$$

tels que:

$$\begin{cases} g_i(\overrightarrow{x}) \leqslant 0, & \text{for } i = 1, \dots, q \\ h_j(\overrightarrow{x}) = 0, & \text{for } j = q+1, \dots, m \end{cases}$$

où f, g_i et h_j sont des fonctions réelles dans l'espace de recherche $\mathcal{S}$. La satisfaction de l'ensemble des contraintes (g_i, h_j) définie le domaine faisable $\mathcal{F}$.

Les deux principaux objectifs des techniques évolutionnaires d'optimisation sous contraintes sont:

1. ramener les individus de la population dans le domaine faisable
2. explorer efficacement le domaine faisable afin de trouver l'optimum global.

La plupart des techniques ont trouvé des solutions efficaces pour atteindre le premier objectif, comme les méthodes de pénalisation (chapitre 2, section 2.3), la méthode de réparation des individus faisables (chapitre 2, section 2.4.1), ou la méthode d'échantillonage de l'espace faisable (chapitre 2, section 2.4.2) ... Par contre, la difficulté réside dans la bonne exploration du domaine faisable, surtout si celui-ci est très petit ou si l'optimum est sur la frontière. Pour résoudre ce problème, certains auteurs ont choisi de considérer la fonction objectif sur tout l'espace de recherche, faisable et non faisable, en vue d'aider l'algorithme dans sa recherche de l'optimum global.

Cependant, considérer la fonction objectif dans le domaine non faisable n'est pas toujours possible. En effet, il est possible que la fonction ne soit pas définie dans le domaine non faisable. Ainsi, si le problème ou la stratégie d'optimisation choisie ne permet pas l'évaluation de la fonction objectif hors de l'espace faisable, l'algorithme a besoin d'opérateurs de reproduction plus aptes que les opérateurs standards afin d'assurer une exploitation et une exploration efficace du domaine faisable.

Le but du travail décrit dans ce chapitre est de faire une comparaison entre les performances atteintes par un algorithme évolutionnaire sur certains problèmes d'optimisation sous contraintes, utilisant des combinaisons de différents opérateurs de croisement et de mutation. Nous commençons d'abord par introduire l'opérateur de mutation logarithmique, qui est destiné spécialement pour explorer localement et globalement l'espace faisable. Nous étudions, ensuite, la capacité d'exploitation et d'exploration de cet opérateur, en comparaison avec d'autres opérateurs de reproduction standards, sur le problème de la *sphère* tronquée. Finalement, quelques résultats obtenus par l'application de la mutation logarithmique et le croisement BLX-0.5 sur 8 cas test sont présentés et comparés avec des résultats obtenus avec d'autres configurations d'opérateurs.

Ce travail a été réalisé en collaboration avec **Alain Pétrowski**, maître de Conférence à l'Insitut National des Télécommunications à Evry. Cette collaboration fait l'objet de deux publications [94] [48].

3.2 L'opérateur de mutation logarithmique

Parmi les opérateurs de mutation standards des EA avec représentation réelle, on doit citer la mutation Gaussienne pour les domaines non bornés, issue des Stratégies d'Evolution [97, 98]. Cet opérateur se base sur la notion d'adaptativité pour ajuster la déviation au cours de l'évolution. Plusieurs stratégies adaptatives et auto-adaptatives on été proposées, comme la règle des 1/5 proposé par Rechenberg en 73 [97] ou la mutation Log-Normale proposée par Schwefel en 81 [121] (voir chapitre 1, section 1.3.2). L'objectif d'une telle stratégie est de générer des grandes perturbations au début de l'évolution pour explorer rapidement tout l'espace de recherche, et des faibles perturbations dans une phase plus avancée, pour assurer la convergence. Cependant, cette méthode nécessite de l'utilisateur le choix crucial des déviations initiales et celui, moins sensible, de leurs taux d'adaptations. De plus, si l'algorithme converge vers un optimum local, elle ne l'aide pas à quitter sa région d'attraction, puisque l'exploration devient très locale.

Une autre approche consiste à utiliser un opérateur de mutation adéquat capable de générer à la fois des petites et grandes perturbations tout au long de l'évolution. Yao et Liu [134, 134] proposent de générer des perturbations selon la distribution de Cauchy et d'adapter sa déviation selon le même principe que la mutation Gaussienne auto-adaptative. Cependant, la variance d'une telle mutation est infinie, et elle s'applique aussi dans un domaine non borné.

Pour résoudre ces problèmes, nous proposons la mutation logarithmique, qui est capable de générer à la fois des petites et grandes perturbations dans un intervalle donné, et ceci avec des probabilités bien équilibrées, tout au long de l'évolution.

3.2.1 Définition

Soit $\overrightarrow{X}$ un vecteur dans l'espace de recherche. L'idée de la mutation logarithmique est que la probabilité de générer une perturbation dans l'intervalle $[10^{-n+1}, 10^{-n}]$ pour une composante du vecteur $\overrightarrow{X}$ soit indépendante de n. Un opérateur qui vérifie cette condition est un opérateur capable d'explorer l'espace de recherche à la fois localement et globalement pendant l'évolution.

Soit x_i une composante du vecteur $\overrightarrow{X}$ avant mutation et x'_i sa valeur après mutation:

$$x'_i = x_i + S \cdot \delta_i, \tag{3.1}$$

où S, le signe de la perturbation, est une variable aléatoire égale à ± 1, et δ_i est une variable positive aléatoire dans l'intervalle $[m_i, M_i]$, où m_i désignant le degré de précision des recherches demandé (e.g. 10^{-9}) et M_i étant déterminée à partir des bornes du domaine (sa définition exacte est donnée dans la section 3.2.2).

Notre objectif devient alors que la probabilité de générer la perturbation δ_i dans l'intervalle $[r, a_i r]$ ne dépende que de a_i, où a_i est une variable aléatoire supérieure à 1. On peut formuler cet objectif comme suit:

$$\forall r \in \left[m_i, \frac{M_i}{a_i}\right], \; P(r < \delta_i < a_i r) = \phi_i(a_i) \tag{3.2}$$

Soit $F(\delta_i)$ la fonction de distribution de la loi de la perturbation. La propriété 3.2 est vérifiée si $F(\delta_i)$ est une fonction logarithmique:

$$F(\delta_i) = A_i \ln \delta_i + B_i \tag{3.3}$$

où A_i et B_i dépendent de M_i de m_i. Ainsi, b_i, fonction de a_i, peut être exprimée comme suit:

$$\phi_i(a_i) = A_i \ln a_i.$$

Si on pose $X = A_i \ln \delta_i + B_i$, alors $F(X)$ est uniforme. En effet, si $P[X \leqslant x] = F(x)$, alors $P[F(x) \leqslant x] = P[X \leqslant F^{-1}(x)] = F(F^{-1}(x)) = x$. Ceci implique que la distribution de la variable aléatoire $X = A_i \ln \delta_i + B_i$ est uniforme dans l'intervalle $[0, 1]$. Ainsi, la variable aléatoire δ_i peut être obtenue à partir d'une distribution uniforme $U(0, 1)$ par:

$$\delta_i = e^{\frac{U(0,1) - B_i}{A_i}}$$

où A_i et B_i sont déterminés à partir de m_i et M_i par:

$$A_i = \frac{1}{\ln M_i - \ln m_i}$$

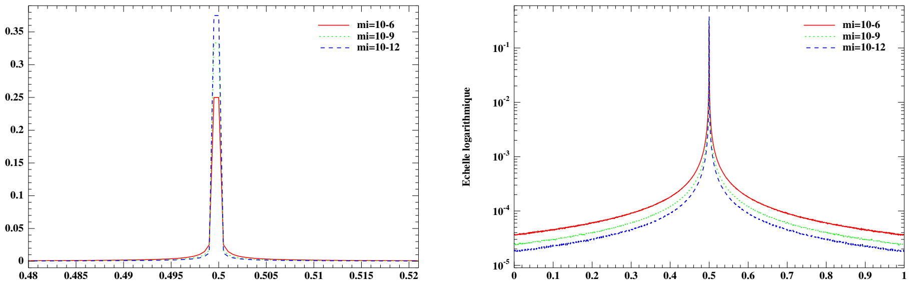
$$B_i = -\frac{\ln m_i}{\ln M_i - \ln m_i}$$

Ceci nous mène à l'expression finale de δ_i:

$$\delta_i = m_i \left(\frac{M_i}{m_i}\right)^{U(0,1)} \tag{3.4}$$

M_i dépend de la façon de prise en compte des bornes de l'espace de recherche. Une procédure pour déterminer sa valeur est proposée dans la section 3.2.2.

La figure 3.1 illustre la fonction de densité de la mutation logarithmique avec des différents degrés de précision: $m_i = 10^{-12}$, $m_i = 10^{-9}$ et $m_i = 10^{-6}$, dans l'intervalle [0,1]. La courbe (a) est un zoom de la courbe réelle pour $x \in [0.48, 0.52]$, afin de pouvoir visualiser la différence entre les trois courbes. La courbe (b) représente les fonctions de densité à l'échelle logarithmique.



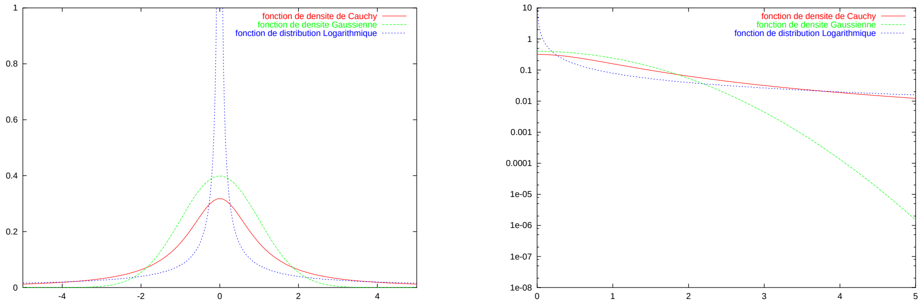
(a): Fonction de densité (zoom) (b): (a) à l'échelle logarithmique

FIG. 3.1 – *Fonction de densité de la mutation logarithmique avec des différents degrés de précision.*

La caractéristique principale de la fonction de densité logarithmique est qu'elle est définie dans un domaine borné ($[-l, l]$, $l \in \mathbb{R}$), alors que celui des fonctions de densité de Cauchy ($f(x) = \frac{1}{\pi} \cdot \frac{t}{t+x^2}$) et de Gauss ($f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$) est non borné ($[-\infty, +\infty]$). Entre autres, les fonctions de Cauchy et de Gauss tendent vers 0 quand x tend vers l'infini. Au contraire, la fonction logarithmique est uniforme quelques soit la taille de son intervalle de définition ($\forall l \in \mathbb{R}$).

La figure 3.2 illustre les 3 types de fonctions dans l'intervalle $[-5, 5]$, avec $m_i = 10^{-9}$ pour la fonction de densité logarithmique et $t = 1$ pour celle de Cauchy. La courbe (a) montre que la distribution logarithmique ne ressemble pas à celles Gaussienne et de Cauchy, du fait qu'elle est plus uniforme et dépendante du degré de précision choisie. Par contre, elle est presque similaire à la fonction de Cauchy quand elles sont présentées à l'échelle logarithmique (courbe (b)). La grande pente de la courbe Gaussienne montre son rapprochement rapide de 0 avec l'augmentation de x, contrairement à celle de Cauchy, dont la pente à l'échelle logarithmique est à peine visible, reflétant un rapprochement de 0 beaucoup plus lent. Par exemple, la fonction de distribution Gaussienne présentée par la figure 3.2 s'annule numériquement pour $x \simeq 30$, alors que celle de Cauchy ne s'annule numériquement que pour $x \simeq 10^{154}$.

On note que l'opérateur de mutation logaritmique ressemble à celui utilisé par Schlierkamp-Voosen and Mühlenbein pour les "Breeder Genetic Algorithm" [112], où il proposent une mutation capable de générer des perturbations dans un domaine borné, mais avec des probabilités non équilibrées.



(a): Fonctions de densité $x \in [-5:5]$ (b): (a) à l'échelle logarithmique $x \in [0:5]$

FIG. 3.2 – *Fonctions de densité logarithmique, Gaussienne tronquée et de Cauchy tronquée dans l'intervalle* $[-5, 5]$*, avec* $m_i = 10^{-9}$ *et* $t = 1$*. Le pic de la courbe de la fonction logarithmique est tronqué afin de pouvoir visualiser les autres courbes. La courbe (b) représente ces fonctions à l'échelle logarithmique dans l'intervalle* $[0:5]$*.*

3.2.2 Respecter les bornes de l'espace de recherche

Plusieurs méthodes peuvent être utilisées pour forcer la valeur de l'allèle mutée à rester dans l'espace de recherche. La plus simple est de tronquer les valeurs qui débordent aux valeurs des bornes $x_{i\,\max}$ ou $x_{i\,\min}$. Cette technique peut causer une accumulation des solutions sur la frontière de l'espace de recherche, au détriment des parties internes du domaine.

Une autre méthode, utilisée souvent avec les mutations Gaussiennes, est de répartir la distribution dans l'intervalle, e.g. “rebondir” sur la frontière de l'espace de recherche. Cependant, si des optima se trouvent sur la frontière, leur localisation avec précision va être ardue, à moins que la déviation standard soit adaptative.

Une meilleure technique pour résoudre ce problème est de choisir la valeur de la borne supérieure M_i dans l'équation 3.4 tel que l'intervalle de la perturbation correspond à la borne inférieure $x_{i\,\min}$ ou la borne supérieure $x_{i\,\max}$ de l'espace de recherche. Ceci implique:

$$M_i = \begin{cases} x_i - x_{i\,\min} & \text{si } S = -1 \\ x_{i\,\max} - x_i & \text{sinon} \end{cases}$$

où S est le signe de la perturbation définie dans l'équation 3.1 ($S = \pm 1$).

3.3 Comparaison de différents opérateurs de reproduction

3.3.1 Cas test : La fonction *Sphère* inversée

En vue d'examiner la capacité d'exploration de quelques opérateurs génétiques standards et celle de l'opérateur logarithmique présenté ci-dessus, une étude expérimentale sur un cas test simple avec des dimensions différentes a été réalisée. Cette étude a pour but de comparer l'habileté des différents algorithmes testés à amener le meilleur individu de la population le plus proche possible de l'optimum. Deux cas tests faciles dérivés de la fonction *Sphère* inversée ont été choisis pour cet objectif. La définition de la fonction *Sphère* inversée est:

$$F_n(\overrightarrow{x}) = \frac{1}{\left(1 + \sqrt{\sum_{i=1}^{n} x_i{}^2}\right)}$$

où les variables sont soumises aux contraintes suivantes:

$$x_i \geqslant 1 \; (i = 1, \cdots, n).$$

Toutes les contraintes sont actives au niveau de l'optimum global qui est localisé au point $X^* = (1, 1, \cdots, 1)$.

La stratégie de sélection adoptée pour cette première étude expérimentale est la méthode de "death penalty" qui rejette les individus infaisables de la population (c.f. chapitre 2, section 2.3.1).

Quand on applique la méthode de "death penalty ", la fonction de performance devient:

$$f_n(\overrightarrow{x}) = \begin{cases} 1/(1 + \sqrt{\sum_{i=1}^{n} x_i{}^2}) & \text{si } x_i \geqslant 1 \, (i = 1, \cdots, n) \\ 0 & \text{sinon} \end{cases}$$

Pour ce cas test, nous avons choisi de limiter l'espace de recherche à l'intervalle $[-10, 10]^n$:

$$\forall i \in \{1, \cdots, n\}, \; -10 \leqslant x_i \leqslant 10$$

La figure 3.3 présente la fonction inverse de la Sphère tronquée pour $n = 1$ et $n = 2$.

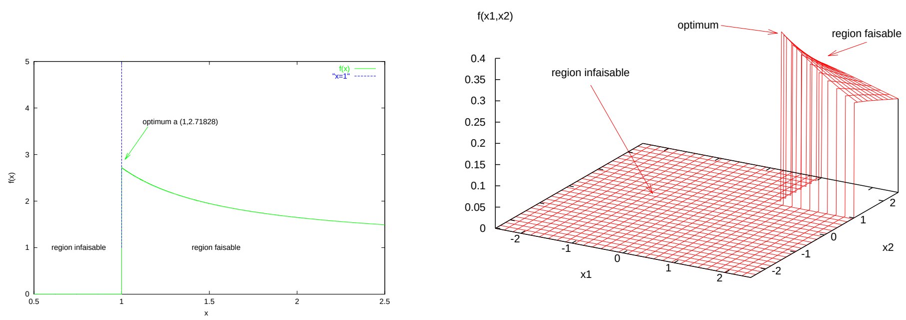


Fig. 3.3 – *La fonction inverse de la* Sphère *tronquée en dimension 1 (gauche) et dimension 2 (droite).*

Deux cas sont considérés dans cette étude: la cas monodimensionnel où $n = 1$: f_1, et un cas multidimensionnel avec $n = 16$: f_{16}.

3.3.2 les opérateurs de reproduction

Différentes combinaisons de différents opérateurs génétiques ont été utilisées pour résoudre le problème posé. Nous avons sélectionné deux opérateurs de croisement et trois opérateurs de mutation. Les opérateurs de croisement sont:

– le BLX-0.5 proposé par Eshelman and Schaffer [27] (noté *BLX* dans la suite),

– le croisement arithmétique proposé par Michalewicz (noté *Arithm* dans la suite).

La définition exacte de ces deux opérateurs est donnée dans la section 1.3.1 du chapitre 1.

Le croisement BLX-α a la propriété d'étendre des deux côtés l'intervalle I entre les deux parents à recombiner, de telle façon que leur enfant ait un domaine de définition plus large. Par conséquent, cette propriété donne au croisement BLX-0.5 une capacité d'exploration plus grande que celle du croisement arithmétique.

Pour la mutation, les trois stratégies sélectionnées pour ce test sont:

– la mutation logarithmique (notée *Log*) présentée dans la section 3.2,
– la mutation Gaussienne (notée *Gauss*), avec adaptation des déviations standards selon la règle de 1/5 donné par Rechenberg (1973) [97] (voir chapitre 1, section 1.3.2).
– la mutation Log-Normale auto-adaptative isotrope de Schwefel (1981) [121] (notée *LN*) (voir chapitre 1, section 1.3.2).

La mutation Gaussienne et la mutation Log-Normale ont été choisies pour cette étude parce qu'elles ont fourni des résultats excellents pour l'optimisation de la fonction *sphère* sans les contraintes.

3.3.3 Configuration de l'algorithme

L'algorithme évolutionnaire utilisé pour cette étude est un GA à remplacement générationnel, basé sur un codage réel et une sélection avec un classement linéaire des individus ayant une performance non nulle. La probabilité de sélection des individus à performance nulle est mise à zéro pour mettre en place la stratégie de la "death penalty". Pour chaque expérience, un ou deux opérateurs de reproduction sont sélectionnés de la liste des opérateurs proposés dans le paragraphe précédent. Le taux de croisement est fixé à 0.9. Le taux de mutation varie entre 0.2 et 0.9. Les paramètres spécifiques à l'opérateur de mutation choisi sont résumés dans le tableau 3.1. On note que pendant la mutation d'un chromosome avec l'opérateur logarithmique dans le cas multidimensionnel, seuls quelques allèles (n_m) sont modifiés. Par contre, avec la mutation Log-Normale et la mutation Gaussienne, tous les allèles sont mutés, afin de garder le principe de base de ces opérateurs (c.f.chapitre 1, section 1.3.2).

mutation logarithmique	mutation Log-Normale	mutation Gaussienne
- précision des recherches demandée: $m_i = 10^{-25}$ - nombre d'allèles permis à muter pour le cas multidimensionnel: $n_m = 2$	- déviation standard initiale: $\sigma_0 = 0.03$ - taux d'apprentissage: $\tau = 1$	- déviation standard initiale: $\sigma_0 = 0.03$ - facteur d'adaptation: $fact = 0.9$

TAB. 3.1 – *Paramètres des différentes mutations testées*

Le nombre d'individus dans la population est fixé à 70. Le nombre maximal de générations permis dépend du cas test.

3.3.4 Résultats avec la méthode de "death penalty" et discussion

Des séries d'expériences de 30 tests chacune ont été effectuées pour les deux fonctions étudiées: monodimensionnelle ou multidimensionnelle. L'ensemble des résultats est résumé dans les figures de 3.4 à 3.11 . Quand la probabilité de mutation est supérieure à 0, les résultats d'une expérience sont présentés par une surface 3D. Elle représente la distance Euclidienne $d(g, p_m)$ entre l'optimum et la meilleure solution trouvée en fonction du nombre de générations g et le taux de mutation

p_m. $d(g, p_m)$ est la valeur médiane des distance minimales sur les 30 tests. La figure 3.4 montre les performances des deux opérateurs de croisement (appliqués avec une probabilité de 0.9) dans la résolution du cas monodimensionnel, quand aucun opérateur de mutation n'est appliqué. Chacune des figures de 3.6 à 3.11 illustre les résultats de l'application d'un des trois opérateurs de mutation avec des différentes probabilités, associés à un des deux opérateurs de croisement, ainsi que le cas où aucun croisement n'est appliqué.

Cas monodimensionnel: fonction f_1

Le Nombre de générations pour les tests de f_1 a été fixé à 500. Les figures 3.4, 3.6, 3.7 et 3.8 illustrent les résultats obtenus avec les différentes configurations du GA.

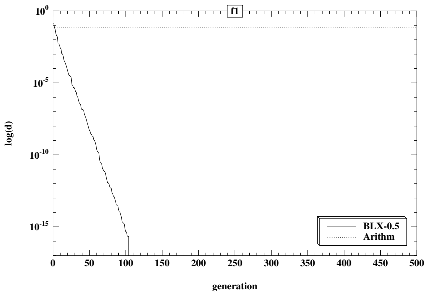


FIG. 3.4 – f_1*: $d(g)$ obtenu avec les croisements BLX et Arithm sans mutation.*

La figure 3.4 montre clairement comment les recherches basées sur l'opérateur *BLX* arrivent très rapidement à converger et à trouver une solution toute proche de l'optimum ($\log(d) \simeq 10^{-16}$ en seulement 100 générations). Cependant, le GA utilisant l'opérateur *Arithm* n'a pas réussi à converger et sa performance est aussi faible à la fin qu'au début de l'évolution. Ces résultats ne sont pas surprenants à cause de l'effet contractant du croisement arithmétique (figure 3.5), qui pénalise l'exploration et mène l'algorithme à une convergence prématurée loin de l'optimum.

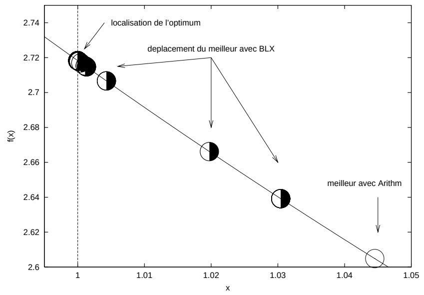


FIG. 3.5 – f_1*: Déplacement des meilleurs individus au cours de l'évolution avec les croisement BLX et Arithm. On remarque que le meilleur individu correspondant au croisement Arithm s'est stabilisé sur le même point, alors que celui obtenu avec le croisement BLX converge rapidement vers la localisation de l'optimum.*

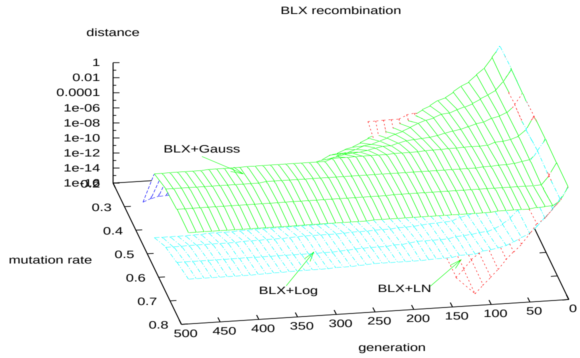


FIG. 3.6 – f_1: $d(g, p_m)$ *obtenue avec le croisement* BLX *et les différentes mutations*

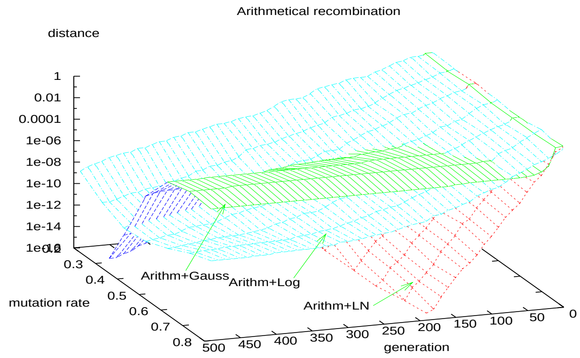


FIG. 3.7 – f_1: $d(g, p_m)$ *obtenue avec le croisement* Arithm *et les différentes mutations*

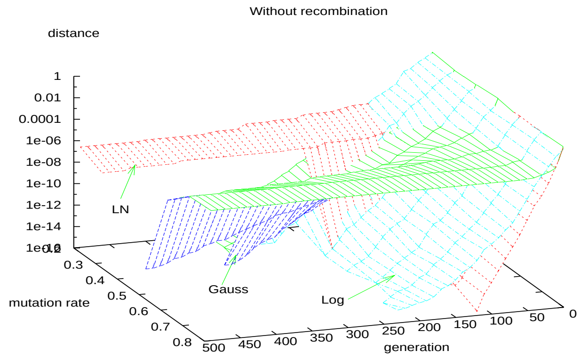


FIG. 3.8 – f_1: $d(g, p_m)$ *obtenue avec les différentes mutations sans croisement*

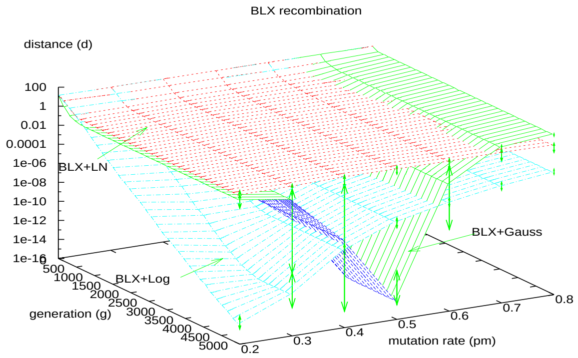


FIG. 3.9 – f_{16}: $d(g, p_m)$ *obtenue avec le croisement* BLX *et les différentes mutations*

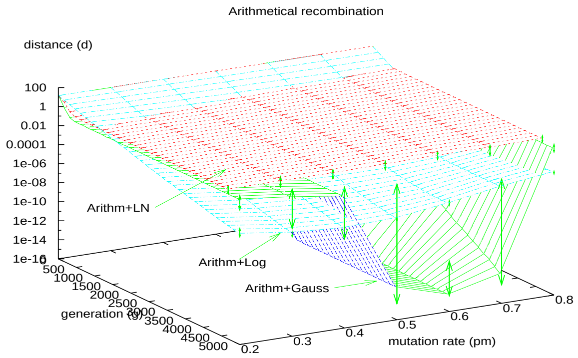


FIG. 3.10 – f_{16}: $d(g, p_m)$ *obtenue avec le croisement* Arithm *et les différentes mutations*

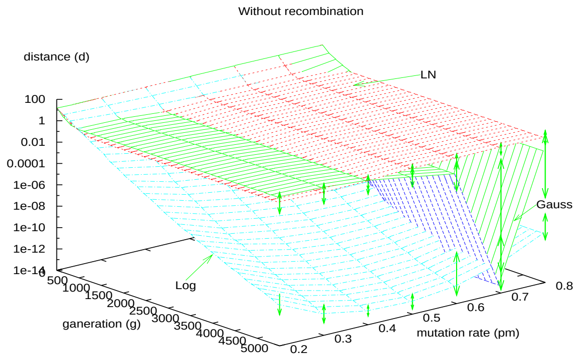


FIG. 3.11 – f_{16}: $d(g, p_m)$ *obtenue avec les différentes mutations sans croisement*

Les figures 3.6 et 3.7 illustrent les résultats donnés par les deux types de croisement combinés aux différentes stratégies de mutation, alors que la figure 3.8 résume la performance de l'AG en utilisant les 3 opérateurs de mutation sans croisement. Ces figures montrent que la mutation Log-Normale donne les meilleurs résultats pour toutes les configurations de croisement. Toutefois, la qualité des résultats demeure inférieure à celle donnée par le *BLX* tout seul.

On note que le comportement de l'AG basé sur le BLX-0.5 associé à la mutation logarithmique est très satisfaisant, surtout pour les taux de mutation assez faibles.

On remarque aussi, à partir de la figure 3.8, que le croisement n'est pas nécessaire pour arriver à des bons résultats, mais le taux de mutation optimal dépend de la stratégie adoptée (élevé pour *LN*, faible pour *Gauss*).

L'effet du taux d'application de chaque type de mutation sur la qualité des résultats est illustré dans la figure 3.12. Pour s'approcher de l'optimum à une distance inférieure à 10^{-12}, *Gauss* nécessite d'être appliqué avec un taux inférieur à 0.5, alors que *LN* est plus performant avec des probabilités d'application supérieures à 0.4. Cependant, la dépendance de *Log* au taux de mutation est faible et même négligeable, puisqu'il arrive toujours à localiser l'optimum, mais un peu plus rapidement avec des probabilités supérieures à 0.4.

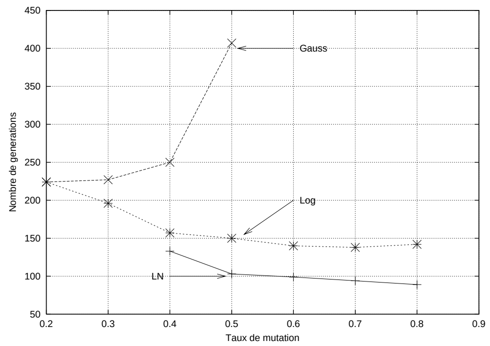


FIG. 3.12 – f_1*: Nombre de générations médian (sur 30 essai) mis par l'algorithme en utilisant les différentes mutations (sans croisement) pour s'approcher de l'optimum à une distance inférieure à* 10^{-12} *(*$d < 10^{-12}$*). Si le point n'est pas présenté, c'est que l'algorithme n'a pas atteint ce degré de précision avec la probabilité de mutation correspondante.*

En conclusion, on peut dire que pour le cas de f_1, toutes les configurations arrivent rapidement à des résultats satisfaisants ($10^{-6} \leqslant log(d) \leqslant 10^{-16}$). Les meilleurs performances sont données par *BLX* tout seul et ensuite *LN* avec des taux de mutations assez élevées. Par contre, ces observations ne sont plus valables pour le cas multidimensionnel f_{16}, où la recherche devient plus difficile.

Cas multidimensionnel: fonction f_{16}

Le nombre maximum de générations permis pour les tests de f_{16} est 5000. Les résultats obtenus avec les différentes configurations de l'AG sont illustrés dans les figures 3.9, 3.10 et 3.11.

Les résultats obtenus avec les deux opérateurs de croisement *BLX* et *Arithm* sans mutation ne sont pas présentés parce qu'ils sont très médiocres: les deux algorithmes convergent prématurément loin de l'optimum. Ainsi, le bon comportement de *BLX* observé dans le cas de f_1 n'est pas préservé pour f_{16}. Dans ce cas, on constate que la mutation devient nécessaire pour le processus de recherche.

De manière similaire, la mutation Log-Normale, contrairement au cas monodimensionnel, n'est plus capable de permettre la convergence de l'algorithme vers l'optimum, et ceci pour toutes les configurations.

En ce qui concerne la mutation Gaussienne, sa performance est irrégulière et très dépendante du taux de son application dans toutes les configurations testées. Cet opérateur nécessite un ajustement très délicat de ses paramètres pour être efficace. Associé au croisement arithmétique et appliqué avec une probabilité de 0.6, il a donné les meilleurs résultats (Log(d)$\simeq 10^{-16}$), mais cette performance disparaît dès qu'on augmente ou on diminue le taux de mutation.

Le comportement de la mutation logarithmique demeure robuste et aussi efficace que pour f_1 ($10^{-6} \leqslant log(d) \leqslant 10^{-16}$). Elle ne nécessite pas un ajustement précis du taux de mutation pour atteindre une bonne performance, spécialement si elle est associée au *BLX* ou si elle est appliquée sans croisement. En effet, son comportement est assez régulier et présente une capacité d'exploration très satisfaisante, quelques soit la configuration de l'algorithme (figure 3.13).

On peut remarquer aussi que les configurations sans croisement sont parmi les plus performantes quand le taux de mutation est suffisamment grand. La figure 3.13 confirme cette remarque. La convergence est plus rapide avec une probabilité de mutation entre 0.6 et 0.8. Cependant, au niveau de degré de précision, le meilleur résultat a été enregistré avec un taux de mutation de 0.5.

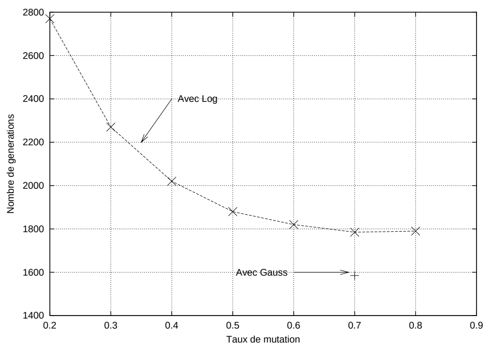
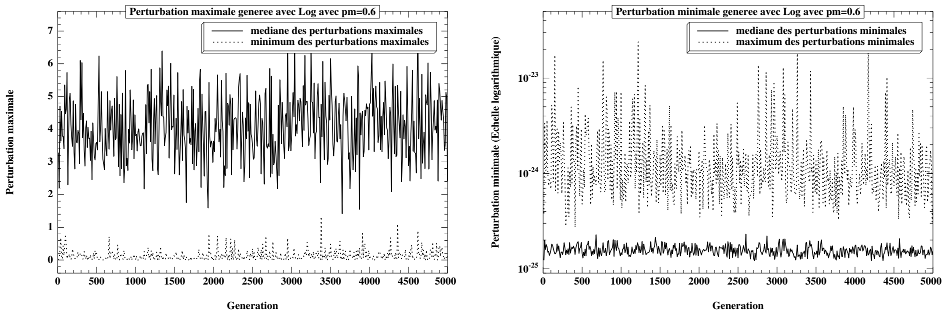


FIG. 3.13 – f_{16}*: Nombre de générations médian (sur 30 essai) mis par l'algorithme en utilisant les différentes mutations (sans croisement) pour s'approcher de l'optimum à une distance inférieure à* 10^{-6} *(*$d < 10^{-6}$*). Si le point n'est pas présenté, c'est que l'algorithme n'a pas atteint ce degré de précision. La mutation* LN *n'est pas présentée parce qu'elle ne donne pas de résultats performants.*

Quand l'opérateur de mutation logarithmique (*Log*) est appliqué sans croisement, les distances médianes à l'optimum sont comprises entre $6.2.10^{-16}$ et 9.10^{-14}. Contrairement à la mutation Gaussienne, l'influence de la probabilité de mutation sur la qualité des résultats est faible. Ceci s'explique par la capacité de la mutation logarithmique de générer des fortes et faibles perturbations, indépendamment de la probabilité de mutation, et à tout moment de l'évolution. La courbe 3.14(a) montre que la valeur médiane des plus grandes perturbations générées au cours de l'évolution est comprise entre 1 et 7. Des perturbations aussi grandes permettent de produire des enfants très différents des parents, appartenant à des nouvelles régions de l'espace de recherche. Elles aident ainsi l'algorithme à éviter la convergence prématurée. Cependant, elles ne lui permettent pas d'exploiter localement les meilleures solutions. Pour aider l'algorithme à converger, l'opérateur de mutation *Log* génère en parallèle des faibles perturbations, permettant de visiter les alentours des meilleurs individus. La figure 3.14(b) illustre la courbe médiane et minimale des plus petites perturbations générées pour le cas multidimensionnel. Elles sont dépendantes du degré de précision demandé. Pour cette étude, il a été fixé à 10^{-25}. Les plus petites perturbations sont alors comprises entre 10^{-25} et 10^{-23}. Cet intervalle peut être ajusté par l'utilisateur selon les nécessités du problème. Une valeur plus élevé (e.g. 10^{-9}) pour le paramètre de l'ordre de précision m_i (voir section 3.3.3) permet d'avoir des perturbations plus grandes.



(a): *Log*: Perturbations maximales

(b): *Log*: Perturbations minimales (échelle logarithmique)

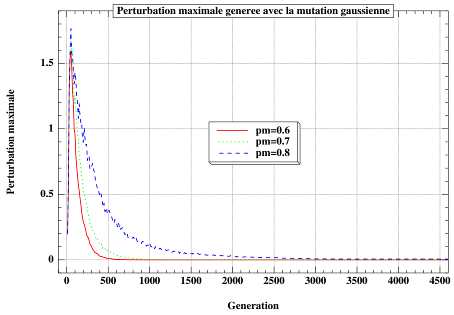
FIG. 3.14 – *Courbes des perturbations maximales et minimales générées avec la mutation Logarithmique pour toutes les composantes mutées de tous les individus de la population.*

Cette grande variation des taux de mutation n'est plus observée pour le cas de la mutation Gaussienne et Log-Normale. La figure 3.15 montre l'évolution des valeurs médianes (sur 30 essais) des plus grandes perturbations générées par l'opérateur *Gauss* (courbes (a) et (b)) et l'opérateur *LN* (courbes (d) et (c)). Comme pour le cas de *Log*, ces perturbations sont mesurées sur l'ensemble des composantes mutées de tous les individus de la population.

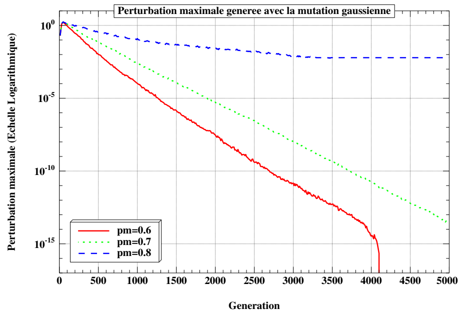
Les formes des courbes des deux opérateurs sont presque similaires. Au début de l'évolution, les valeurs médianes des perturbations maximales sont croissantes. Ceci est dû à l'augmentation des déviations standards, tant que la population n'est pas stable. Une fois que la population commence à se stabiliser (l'algorithme commence à converger), les valeurs des déviations décroîent rapidement, ce qui implique, en parallèle, la réduction de l'intervalle des perturbations. Si cette convergence est prématuré, la mutation ne peut pas l'aider à l'éviter avec des déviations aussi petites. Entre autres, l'habileté de l'algorithme à localiser avec précision l'optimum global dépend fortement des déviations standards à la phase finale de l'évolution.

Par exemple, pour le cas de la mutation Gaussienne, appliquée avec une probabilité de 0.7, la valeur médiane de la déviation standard $\sigma(t)$ descend jusqu'à $3.9.10^{-20}$, ce qui a permis à la meilleure solution de s'approcher de l'optimum à une distance qui descend jusqu'à $1.95.10^{-14}$. Cependant, avec des probabilités de mutation plus élevées ou plus petites, vue la dépendance du schéma d'adaptation de la déviation au taux de mutation, ce degré de précision n'est pas atteint. En effet, avec $p_m = 0.8$, $\sigma(t)$ ne descend pas au dessous de $2.3.10^{-5}$ (où sa valeur médiane est de $1.9.10^{-3}$, figure 3.15(b)) et la meilleure solution sur les 30 essais est à une distance de $2.8.10^{-4}$ de l'optimum. Avec la mutation logarithmique, ce problème ne se pose pas étant donné que la précision des recherches est indépendante de la probabilité de mutation, et elle est réglable par l'utilisateur.

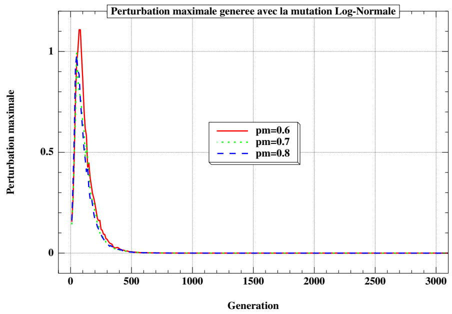
Par contre, avec une probabilité de mutation de 0.6, la valeur de $\sigma(t)$ décroît très rapidement, où elle arrive à la fin de l'évolution à 10^{-25}. Des telles déviations permettent une exploitation locale des meilleures solutions, mais la capacité d'exploration de l'algorithme est perdue à une phase avancée de l'évolution, ce qui explique les convergences prématurées observées avec $p_m = 0.6$.



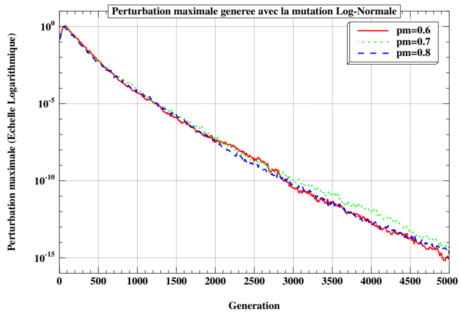
(a): *Gauss*: Perturbations maximales



(b): *Gauss*: Courbe (a) à l'échelle logarithmique



(c): *LN*: Perturbations maximales



(d): *LN*: Courbe (c) à l'échelle logarithmique

FIG. 3.15 – *Courbes médianes des perturbations maximales générées avec la mutation Gaussienne (courbes (a) et (b)) et la mutation Log-Normale (courbes (c) et (d)) pour l'ensemble des composantes de tous les individus mutés, avec des probabilités de mutation de* 0.6, 0.7 *et* 0.8.

Pour conclure cette première étude expérimentale, on résume que les meilleurs résultats ont été enregistrés avec 5 configurations parmi l'ensemble des configurations testées. Elles sont illustrées par le tableau 3.2, qui donne aussi la distance médiane à l'optimum donnée par chacune.

	Config. 1	Config. 2	Config. 3	Config. 4	Config. 5
Config.	*Log* ($p_m = 0.5$)	*Gauss* ($p_m = 0.7$)	*Log* ($p_m = 0.2$) + *BLX* ($p_c = 0.9$)	*Gauss* ($p_m = 0.5$) + *BLX* ($p_c = 0.9$)	*Gauss* ($p_m = 0.6$) + *Arithm* ($p_c = 0.9$)
$d(g, p_m)$	$9.7.10^{-14}$	$4.5.10^{-14}$	9.10^{-15}	$7.4.10^{-15}$	$5.5.10^{-15}$

TAB. 3.2 – *Résumé des configurations qui ont donné les meilleures distance médianes* $d(g, p_m)$ *pour le cas multidimensionnel* f_{16}.

3.4 Résultats comparatifs

L'opérateur de mutation logarithmique a donné les meilleurs résultats en terme de performance et robustesse pour le problème de la *Sphère* tronquée analysé dans la section 3.3. Pour confirmer cette robustesse, nous étendons l'étude expérimentale à la résolution d'autres fonctions références avec un GA basé sur l'opérateur de mutation logarithmique associé à l'opérateur de croisement BLX. D'autres expérimentations sont effectuées en parallèle avec les configurations ayant enregistré les meilleurs résultats pour la *Sphère* tronquée, sélectionnées du tableau 3.2. Une autre configuration avec la mutation Log-Normale et BLX est aussi testée, afin de pouvoir comparer les différentes mutations sur d'autres cas test que la *Sphère* tronquée. Nous comparons ensuite les différents résultats obtenus avec toutes les configurations résumées ci-dessous:

1. *Log* $(p_m = 0.4)$ + *BLX* $(p_c = 0.9)$ (une deuxième série d'expériences est effectuée avec une meilleure approximation des taux de mutation et de croisement pour chaque cas test),
2. *Log* $(p_m = 0.5)$,
3. *Gauss* $(p_m = 0.7)$,
4. *Gauss* $(p_m = 0.5)$ + *BLX* $(p_c = 0.9)$,
5. *Gauss* $(p_m = 0.6)$ + *Arithm* $(p_c = 0.9)$,
6. *LN* $(p_m = 0.6)$ + *BLX* $(p_c = 0.9)$.

Les expériences ont été effectuées sur 8 fonctions G1, G2, G4, G6, G7, G8, G9 et G10, sélectionnées de l'ensemble des cas test proposés dans [87] (la définition complète de ces fonctions est donnée dans l'annexe A).

Le but de cette étude expérimentale est de montrer la nécessité d'améliorer le choix des opérateurs d'exploration pour les problèmes contraints quand on ne dispose pas d'une technique adéquate de prise en compte des contraintes. Les opérateurs standards peuvent être robustes si l'algorithme dispose d'une stratégie appropriée pour traiter les contraintes. Dans le chapitre 4 nous proposons un nouvel algorithme pour l'optimisation sous contraintes, qui peut être appliqué avec plusieurs types d'opérateurs.

Entre autres, la méthode de "death penalty" n'est pas appliquée dans cette deuxième série d'expériences, parce qu'elle nécessite une population initialisée dans l'espace faisable. Elle a été remplacée, pour les tests de cette section, par une méthode de sélection rudimentaire, basée sur un tri spécial des individus de la population. L'objectif principal de cette sélection est de ramener les individus dans le domaine faisable, sans prendre en compte la localisation de l'optimum global faisable.

3.4.1 Définition d'un opérateur de sélection pour les problèmes d'optimisation sous contraintes

Dans cette approche, les contraintes du problème ne sont prises en compte que pendant la sélection. L'opérateur de sélection qu'on propose pour accomplir cette tâche est simple et rudimentaire, basé sur un tri approprié des individus, présenté ci-dessous. Une fois les individus sont classés, les survivants sont déterminés par une sélection proportionnelle sur le rang (chapitre 1, section 1.5.1).

classement des individus

Le tri des individus est basé sur la faisabilité des individus et leurs performances. Il est inspiré de la sélection par le rang proposée par Baker [11]. Les différentes étapes de cet arrangement sont comme suit:

1. On ordonne les individus faisables selon les valeurs de la fonction objectif. Une liste de n_F faisables est alors obtenue:
$$\mathcal{L}_{\mathcal{F}} = (\overrightarrow{F}_1, \overrightarrow{F}_2, ..., \overrightarrow{F}_{n_{\mathcal{F}}})$$
où $\overrightarrow{F}_1$ représente le meilleur individu de la population
2. On ordonne ensuite les individus infaisables selon les valeurs d'une fonction de mesure de contraintes $C(\overrightarrow{x})$, qui a la propriété de croître quand les violations d'une ou de plusieurs contraintes augmentent. La définition de la fonction $C(\overrightarrow{x})$ est décrite à la suite dans le paragraphe 3.4.1. Une liste de $n_{\mathcal{U}}$ individus infaisables est alors obtenue:
$$\mathcal{L}_{\mathcal{U}} = (\overrightarrow{U}_1, \overrightarrow{U}_2, ..., \overrightarrow{U}_{n_{\mathcal{U}}})$$
où $\overrightarrow{U}_1$ représente le meilleur individu infaisable en terme de violations, c'est à dire l'individu dont la valeur de $C(\overrightarrow{U}_1)$ est minimale, i.e. qui viole le moins de contraintes possible.
3. L'arrangement final de la population est obtenu par la concaténation les deux listes $\mathcal{L}_{\mathcal{F}}$ et $\mathcal{L}_{\mathcal{U}}$, de telle façon que l'individu $\overrightarrow{F}_1$ soit le le premier et l'individu $\overrightarrow{U}_{n_{\mathcal{U}}}$ soit le dernier dans la liste résultante $\mathcal{L}_{\mathcal{P}}$:
$$\mathcal{L}_{\mathcal{P}} = (\overrightarrow{F}_1, \overrightarrow{F}_2, ..., \overrightarrow{F}_{n_F}, \overrightarrow{U}_1, \overrightarrow{U}_2, ..., \overrightarrow{U}_{n_{\mathcal{U}}})$$

La fonction de performance de chaque individu est déduite de cet arrangement global en utilisant la méthode habituelle:
$$f(\overrightarrow{F}_i) = \left(1 - \frac{i}{n_{\mathcal{F}} + n_{\mathcal{U}}}\right)^p \tag{4}$$
pour un individu faisable $\overrightarrow{F}_i$, et
$$f(\overrightarrow{U}_i) = \left(1 - \frac{i + n_{\mathcal{F}}}{n_{\mathcal{F}} + n_{\mathcal{U}}}\right)^p \tag{5}$$
pour un individu infaisable $\overrightarrow{U}_i$.

Le choix de la valeur du paramètre p dépend de la pression de sélection désirée. Pour toutes les expériences décrites dans cette section, nous avons choisi une sélection linéaire: $p = 1$.

L'inconvénient de cet opérateur de sélection rudimentaire est qu'il n'est pas capable de prendre en compte le cas d'un espace faisable très petit ou ayant une mesure nulle (c'est le cas des contraintes d'égalité). Pour cette raison, les expériences décrites ci-dessous excluent ce type de problèmes. Si des contraintes d'égalité sont présentes, des algorithmes de sélection plus sophistiqués sont nécessaires. Dans les chapitres 4, 5 et 6, nous proposons un nouvel algorithme de prise en compte des contraintes, capable de traiter les contraintes d'égalité.

Mesure des violations de contraintes

Soit $g_i(\overrightarrow{x}) \leqslant 0$, $i = 1, \ldots, q$, l'ensemble des contraintes.

La méthode la plus simple de construire une mesure des violations des contraintes $C(\overrightarrow{x})$ est de faire une somme pondérée des valeurs $g_i(\overrightarrow{x})$ quand $g_i(\overrightarrow{x})$ est positive. Mais tout choix d'expression de mesure des violations des contraintes est robuste, à condition qu'elle soit monotone par rapport aux taux de violation. Ainsi, outre la fonction somme, plusieurs autres expressions peuvent être utilisées avec la même efficacité. Par exemple, nous avons utilisé l'expression suivante pour les résultats présents dans cette section:

$$C(\overrightarrow{x}) = 1 - \prod_{i=1}^{q} c_i(\overrightarrow{x}) \tag{6}$$

avec:

$$c_i(\overrightarrow{x}) = \begin{cases} \frac{1}{1+g_i(\overrightarrow{x})} & \text{si } g_i(\overrightarrow{x}) > 0 \\ 1 & \text{sinon} \end{cases} \tag{7}$$

3.4.2 Conditions expérimentales

Le GA utilisé dans cette deuxième série de tests a les même conditions expérimentales que celui utilisé pour le problème de la *Sphère* tronquée (section 3.3), exception faite pour la sélection qui incorpore le tri décrit ci-dessus.

Pour les premières séries de tests avec *Log* et *BLX*, les taux de mutation p_m et de croisement p_c ainsi que le nombre maximum d'allèles perturbés dans un génotype muté n_m dépendent du problème. Une autre série d'expériences a été effectuée avec des paramètres constants fixés comme suit: $p_m = 0.4$, $n_m = 2$ et $p_c = 0.9$. Ces deux séries ont été faites afin de montrer le degré de dépendance de la performance de l'algorithme aux paramètres génétiques. Les taux de mutation et de croisement pour les autres configurations sont ceux qui ont donné les meilleurs résultats pour le cas de la *Sphère* tronquée. Le paramètre de l'ordre de précision de l'opérateur *Log* a été fixé à 10^{-9}.

L'optimum global du problème G2 était difficile à trouver avec un algorithme génétique standard, même en utilisant la mutation logarithmique. Ceci est dû aux fréquentes convergences prématurées vers un des optima locaux tout autour de l'optimum global. Pour résoudre ce problème, la diversité de la population dans l'espace de recherche est maintenue grâce à une méthode de nichage, à savoir la procédure d'éclaircissement élitiste proposée par Pétrowski[93].

3.4.3 Résultats

Les résultats des deux séries d'expériences pour les 8 fonctions effectuées avec *Log* et *BLX* sont résumés dans le tableau 3.3. Le nombre de générations maximum G est de 5000. Le nombre réel d'itérations est parfois inférieur à cette limite. Les valeurs exactes sont indiquées dans le tableau. Deux séries d'expériences ont été réalisées, la première avec des paramètres communs fixes, et la deuxième avec les meilleures approximations des paramètres correspondant au problème testé.

Toutes les solutions trouvées pour les différents tests sont faisables. En considérant la précision des calculs, l'optimum exacte a été trouvé pour 4 problèmes sur 8. Pour tous les problèmes, la plus grande erreur entre la valeur médiane obtenue est la valeur de l'optimum est inférieure à 3%.

La comparaison entre les résultats obtenus avec des paramètres fixes et ceux de la deuxième série montre qu'ils sont presque similaires, exceptés ceux de G2. L'influence très légère de la variation des valeurs des paramètres sur la qualité des résultats ne fait que confirmer la robustesse de l'opérateur logarithmique. Les mauvais résultats pour le cas de G2 dans la première série des tests n'est en fait que la conséquence du manque de nichage, qui est nécessaire pour ce problème vue la multitude d'optima locaux aux alentours de l'optimum global. Les résultats sont largement meilleurs dans la deuxième série de test avec le nichage. La technique appliquée est l'éclaircissement élitiste, dont la définition est donnée dans la section 1.6.4 du chapitre 1.

		G1	G2	G4	G6	G7	G8	G9	G10
Opt. Ext		-15.000	0.803619	-30665.5	-6961.81	24.3062	0.095825	680.630	7049.3309
G		5000	5000	2000	1000	5000	100	5000	5000
Paramètres fixes $p_m = 0.4$, $n_m = 2$, $p_c = 0.9$	b	**-15.000**	0.70861	**-30665.5**	-6961.11	**24.338**	**0.095825**	**680.630**	**7066.36**
	m	**-15.000**	0.59767	**-30665.5**	-6959.10	24.829	**0.095825**	**680.637**	7262.19
	w	**-15.000**	0.51791	**-30665.5**	-6956.59	26.582	**0.095825**	**680.657**	8499.93
Résultats avec une meilleure approximation des paramètres		$p_m = 0.4$ $n_m = 2$ $p_c = 0.6$	$p_m = 1.0$ $n_m = 10$ $p_c = 0.8$ nichage	$p_m = 0.4$ $n_m = 2$ $p_c = 1$	$p_m = 0.1$ $n_m = 2$ $p_c = 1.0$	$p_m = 0.4$ $n_m = 2$ $p_c = 1.0$	$p_m = 0.4$ $n_m = 2$ $p_c = 1.0$	$p_m = 0.4$ $n_m = 2$ $p_c = 0.9$	$p_m = 0.4$ $n_m = 2$ $p_c = 1.0$
	b	**-15.000**	0.80248	**-30665.5**	**-6961.81**	24.384	**0.095825**	**680.630**	7070.33
	m	**-15.000**	0.79430	**-30665.5**	**-6961.81**	**24.509**	**0.095825**	**680.637**	7259.64
	w	**-15.000**	0.75832	**-30665.5**	**-6961.81**	**24.974**	**0.095825**	**680.657**	8429.79

TAB. 3.3 – *Résultats expérimentaux donnant, pour chaque fonction, la meilleure solution (b), la solution médiane (m) et la pire solution (w) obtenues sur les 30 essais réalisés avec les opérateurs* Log *et* BLX.

Les résultats pour toutes les autres configurations utilisant les opérateurs standards sont résumés dans le tableau 3.4. De même qu'avec la configuration *BLX + Log*, le nombre de générations est limité à 5000, et le cas test G2 est traité avec l'éclaircissement élitiste. On remarque que la qualité de la majorité des résultats est médiocre. On peut en déduire que la robustesse de l'algorithme avec la première configuration (*Log + BLX*) est due essentiellement à la capacité exploratoire des opérateurs de reproduction. L'opérateur de sélection utilisé ne fait que forcer les individus vers l'espace faisable. Cependant, un tel opérateur ne garantit pas la faisabilité des solutions retournées. En effet, quelques configurations n'arrivent pas à localiser des individus faisables, comme celle où l'opérateur de mutation *Gauss* est appliquée sans croisement.

Quand l'opérateur *Log* est appliqué sans croisement, les résultats sont assez satisfaisants et se rapprochent de ceux trouvés quand il est associé à l'opérateur *BLX*. Ceci confirme sa grande capacité d'exploration, même pour des problèmes plus difficile que celui de la *Sphère* tronquée. Ceci est nullement le cas de l'opérateur *Gauss* qui a donné les pire résultats quand il est appliqué sans croisement. Toutefois, il atteint sa meilleure performance quand il est associé à l'opérateur *Arithm*, où il a réussi à localiser l'optimum pour quelques cas test (G6 et G8) et a donné un très bon résultat pour le problème difficile G10.

Contrairement à ce qui est attendu, associé à l'opérateur de croisement *BLX*, la mutation Log-Normale n'a pas montré la même incapacité que celle observée pour le cas multidimensionnel de la *Sphère* tronquée. Elle a réussi à localiser l'optimum pour les cas G6 et G8, et a atteint une performance assez satisfaisante pour les cas de G9 et G10 . Cependant, sa performance reste faible pour la résolution des cas test G1, G2, G4 et G7.

Les résultats de cette deuxième étude expérimentale confirment l'observation établie dans la première étude, portant sur la détérioration de la performance des opérateurs de reproduction standards en présence des contraintes. Ils confirment aussi la robustesse de l'opérateur de mutation logarithmique proposé dans la section 3.2. Ce dernier diffère des autres opérateurs standards par sa capacité de maintenir la diversité de la population tout au long de l'évolution, permettant ainsi une meilleure exploration de l'espace de recherche.

		G1	G2	G4	G6	G7	G8	G9	G10
Opt. Ext		-15.000	0.803619	-30665.5	-6961.81	24.3062	0.095825	680.630	7049.3309
Log($p_m = 0.5$)	b	**-15.000**	0.79671	**-30665.5**	**-6961.81**	24.9679	**0.095825**	680.831	7109.56
	m	**-15.000**	0.783501	**-30665.5**	**-6961.81**	27.7496	**0.095825**	681.286	9398.4
	w	-13.000	0.767257	-30558	**-6961.81**	48.1981	**0.095825**	686.465	12218.1
	f	1	1	1	1	1	1	1	1
Gauss($p_m = 0.7$)	b	-7.375	0.363116	-30531.7	-6701.08	182.346	0.095812	747.91	12788.47
	m	-3.347	0.281915	-30408.5	-5864.04	638.218	0.095439	847.19	19759.8
	w	-0.436	0.249097	-30279.7	-4379.55	3445.55	0.093078	1044.55	28930.5
	f	0	1		0	0	1		0
Gauss($p_m = 0.6$) + *Arithm*($p_c = 0.9$)	b	-8.484	0.400506	-30574.3	**-6961.81**	24.3514	**0.095825**	680.871	**7065.93**
	m	-2.857	0.287416	-30480.5	**-6961.81**	24.8831	**0.095825**	681.316	7592.17
	w	-0.756	0.242547	-30399.7	**-6961.81**	26.2143	**0.095825**	681.899	12716.1
	f	0	1	1	1	1	1	1	1
Gauss($p_m = 0.5$) + *BLX*($p_c = 0.9$)	b	-7.336	0.304416	-30532.9	-6935.8	29.5514	**0.095825**	685.223	12005.2
	m	-2.937	0.272794	-30383.2	-6892.48	31.6901	0.095818	689.93	19963.3
	w	0.597	0.242578	-30247.5	-6806.56	34.2349	0.095795	697.322	24792
	f	0	1	1	0	1	1	1	0
LN($p_m = 0.6$) + *BLX*($p_c = 0.9$)	b	-9.818	0.415224	-30657.2	**-6961.81**	28.8184	**0.095825**	680.857	7258.68
	m	-6.430	0.344053	-30525.4	**-6961.81**	40.3441	**0.095825**	686.513	8307.52
	w	-4.580	0.297337	-30338.2	**-6961.81**	546.653	**0.095825**	717.17	13845.7
	f	0	1	1	1	1	1	1	1

TAB. 3.4 – *Résultats expérimentaux donnant, pour chaque fonction, la meilleure solution (b), la solution médiane (m) et la pire solution (w) obtenues sur les 30 essais réalisés avec chacune des 5 configurations testées. Le paramètre f indique la faisabilité des meilleures solutions retournées: 1: faisable, 0: infaisable.*

3.5 Conclusion

Cette étude a permis de montrer que certains opérateurs de reproduction, utilisés habituellement dans la recherche évolutionnaire, ne sont pas optimaux pour la résolution des problèmes sous contraintes. En effet, leur capacité d'exploration diminue avec la restriction de l'espace de recherche. Une étude expérimentale montre que certains opérateurs génétiques standards, alors qu'ils sont capables d'accomplir des excellentes performances dans la résolution du problème d'optimisation de la fonction *Sphère*, perdent leur robustesse en présence de contraintes, et dans certaines conditions, échouent complètement à s'approcher de l'optimum.

Pour améliorer la performance des opérateurs d'exploration en présence des contraintes, une solution alternative intéressante est l'opérateur de mutation logarithmique proposé dans ce chapitre. Il a été conçu pour explorer à la fois localement et globalement l'espace de recherche, même lorsqu'il est difficile d'accès et/ou de mesure réduite, sans avoir à ajuster des paramètres pour optimiser sa performance. Des tests sur le problème de la sphère sous contraintes avec différentes probabilités de mutations et dans différentes conditions ont donné des résultats très satisfaisants, s'ils ne sont pas parfaits, reflétant le comportement robuste et efficace du l'opérateur de mutation logarithmique.

Associé à l'opérateur de croisement BLX-0.5 et un simple schéma de sélection pour prendre en compte les contraintes, la mutation logarithmique permet au GA d'atteindre des très bonnes performances sur 8 problèmes de référence, en arrivant à s'approcher avec précision de l'optimum global dans la majorité des essais. Sa performance est souvent meilleure que d'autres opérateurs de reproduction testés avec le même schéma de sélection sur les mêmes cas test.

Cette étude montre la nécessité d'optimiser l'exploration de l'espace de recherche en présence des contraintes. Une solution est d'améliorer les opérateurs de reproduction afin qu'ils soient capable de maintenir la diversité de la population au cours de l'évolution, tout en permettant une bonne exploitation des meilleures solutions. Dans les chapitres suivants, nous proposons une autre stratégie pour maintenir la diversité, basée sur un schéma de pénalisation adaptatif.

Chapitre 4

Un Algorithme adaptatif pour l'optimisation sous contraintes: ASCHEA

Résumé

La notion d'adaptativité est devenue un sujet clé pour les Algorithmes Evolutionnaires, surtout après les apports des derniers travaux faits en adaptativité dans les Stratégies d'Evolutions. Le principe de l'adaptativité est de permettre à l'algorithme d'ajuster automatiquement ses paramètres, en se basant sur des informations sur l'état d'évolution de la population.

On propose, dans ce chapitre, une nouvelle méthode de prise en compte des contraintes dans les problèmes d'optimisation évolutionnaire, basée sur des mécanismes adaptatifs dépendant de l'état courant de la population. Ces mécanismes se présentent au sein de trois composantes de l'algorithme. Premièrement, une fonction de pénalité adaptative qui modifie les cœfficients de pénalisation selon la proportion d'individus faisables dans la population: son but est de maintenir la diversité de la population aussi long temps que possible; Deuxièmement, une stratégie spéciale de sélection pour le processus de croisement, appelée stratégie de séduction/sélection, qui, à un certain moment de l'évolution, croise les individus faisables avec les non faisables, afin d'explorer les régions aux alentours des frontières du domaine réalisable $\mathcal{F}$; La troisième et dernière composante concerne la sélection qui est ajustée afin de favoriser un certain nombre d'individus faisables. L'union de ces trois ingrédients a donné naissance à un nouvel algorithme adaptatif pour l'optimisation sous contraintes, appelé ASCHEA, *Adaptive Segregational Constraint Handling Evolutionnary Algorithm*.

ASCHEA a montré une grande robustesse dans la résolution d'un ensemble de fonctions de référence. Il a enregistré des performances souvent meilleures que celles jamais atteintes pour certains de ces cas test.

4.1 Introduction

Dans le chapitre 2, nous avons présenté les méthodes les plus connues pour la prise en compte des contraintes dans la programmation évolutionnaire. Parmi ces méthodes, les plus utilisées sont celles basées sur les fonctions de pénalité (catégorie I), vue la facilité et la rapidité de leur implantation.

Nous rappelons qu'un problème d'optimisation sous contraintes est défini comme suit:

$$\text{minimiser } f(\overrightarrow{x}),\ \overrightarrow{x} = (x_1, \ldots, x_n) \in \mathcal{F} \subseteq \mathcal{S} \subseteq \mathbb{R}^n,$$

tels que

$$\begin{cases} g_j(\overrightarrow{x}) \leqslant 0, & \text{for } j = 1, \ldots, q \\ h_j(\overrightarrow{x}) = 0, & \text{for } j = q+1, \ldots, m \end{cases}$$

où f, g_i et h_j sont des fonctions réelles dans l'espace de recherche $\mathcal{S}$. La satisfaction de l'ensemble des contraintes $(g_i,\, h_j)$ définie le domaine faisable $\mathcal{F}$.

Le principe des méthodes par pénalisation est d'additionner à la valeur de la fonction objectif f des solutions infaisables une quantité positive, afin de défavoriser tel individus dans la population:

$$eval(\overrightarrow{x}) = f(\overrightarrow{x}) + penal(\overrightarrow{x}), \tag{4.1}$$

où

$$penal(\overrightarrow{x}) = 0 \text{ ssi } \overrightarrow{x} \in \mathcal{F}$$

Malgré leur simplicité apparente, ces méthodes ont beaucoup de difficulté à trouver la définition adéquate pour la fonction de pénalité. Les approches les plus populaires utilisent la mesure des violations des contraintes dans la définition de $penal(\overrightarrow{x})$:

$$penal(\overrightarrow{x}) = \sum_{j=1}^{q} \alpha_j g_j^+(\overrightarrow{x}) + \sum_{j=q+1}^{m} \alpha_j |h_j|(\overrightarrow{x}), \tag{4.2}$$

où x^+ est la partie positive de x.
Les nombres réels $\alpha_j, j = 1 \ldots m$ sont appelés *les coefficients de pénalité*, dont la détermination des valeurs appropriées constitue la difficulté majeure pour les approches concernées. Toutes les techniques qui ont été proposées peuvent être affectées à une des classes suivantes (voir chapitre 2, section 2.3):

- les méthodes de pénalisation statique: les *cœfficients de pénalité* sont fixés au départ et restent inchangés pour le reste de l'évolution [8] [58],
- les méthodes de pénalisation dynamique: les *coefficients de pénalité* sont modifiés tout au long de l'évolution selon une certaine stratégie, généralement croissante, afin d'assurer la réalisabilité de la population à la fin de l'évolution [63] [83],
- les méthodes de pénalisation adaptative: les *coefficients de pénalité* sont adaptés au fur et mesure de l'évolution selon l'état courant et/ou l'historique de la population [47] [124] [95].

Les approches les plus performantes sont celles de la classe adaptative, vu qu'elles tiennent compte de l'information courante et historique sur l'état de la population au cours de l'évolution. Avec ce genre d'approche, la valeur de pénalité peut croître ou décroître selon l'importance des violations des contraintes et l'état de la recherche. Deux approches ont été déjà proposées [47, 124], présentées dans la section 2.3.4 du chapitre 2. Cependant, ces deux techniques ne tiennent compte que de l'état des meilleures solutions, sans se soucier de l'état général de la population. Par exemple, le degré de faisabilité de la population n'intervient pas dans la stratégie d'adaptation des cœfficients de pénalisation.

Pour ces différentes raisons, nous proposons un nouvel algorithme adaptatif de prise en compte des contraintes, appelé **ASCHEA** ("Adaptive Segregational Constraint Handling Evolutionnary Algorithm"). Il se base sur une méthode adaptative d'ajustement *des coefficients de pénalité* et utilis des techniques de sélection et de croisement appropriées. L'objectif d'ASCHEA est d'assurer une exploration efficace de tout l'espace de recherche, afin de maximiser les chances d'une localisation exacte de l'optimum global, qu'il soit à l'intérieur du domaine faisable ou sur sa frontière. Le principe de base ainsi que les différentes composantes de la méthode sont décrits dans la section suivante.

4.2 L'Algorithme Segrégationnel Adaptatif ASCHEA

Dans le chapitre précédent, nous avons montré la nécessité du maintient de la diversité de la population au long de l'évolution, spécialement pour les problèmes sous contraintes. Nous avons alors proposé la mutation logarithmique, qui a l'avantage de pouvoir générer des petites et des fortes perturbations avec des probabilités équilibrées, ce qui permet d'explorer localement et globalement l'espace de recherche.

Dans ce chapitre, nous proposons une autre technique pour favoriser la diversité de la population, basée sur une fonction de pénalité adaptative. C'est la composante principale du nouvel algorithme **ASCHEA**, dont l'objectif est de maintenir à la fois des individus faisables et infaisables dans la population, selon les nécessités du problème. Pour se faire, ASCHEA possède ses propres outils, qui sont:

- **Une fonction de pénalisation adaptative** qui utilise des informations globales sur la population [54] pour ajuster les cœfficients de pénalité. Avec cette fonction, les individus infaisables sont pénalisés ou favorisés, selon le taux de leur présence dans la population.
- un croisement basé sur **une stratégie de séduction/sélection**, où dans certains cas, les individus faisables ne peuvent se reproduire qu'avec les infaisables, en vue d'explorer les régions aux alentours de la frontière du domaine $\mathcal{F}$,
- **une sélection ségrégationnelle** qui fait une distinction entre les faisables et les non faisables, et dont le but est de maintenir un certain taux de faisabilité de la population.

Une explication détaillée de ces trois ingrédients est donnée dans les trois sous-paragraphes suivants.

Dans un premier temps, nous allons considérer un cœfficient de pénalité $\alpha(t)$ unique pour toute les contraintes. Dans le chapitre suivant, une autre définition sera donnée à la fonction de pénalisation, où chaque contrainte du problème aura son propre cœfficient de pénalité.

Pour tout le reste du chapitre, nous définissons les deux notations suivantes:

- τ_t: la proportion des individus faisables dans la population à la génération t,
- τ_{target}: proportion définie par l'utilisateur; l'idée d'ASCHEA est de maintenir τ_t le plus proche possible du paramètre τ_{target} au long de l'évolution.

4.2.1 Penalité adaptative au niveau population

Nous proposons dans ce paragraphe une nouvelle fonction de pénalisation adaptative basée sur le taux de faisabilité de la population. En vue d'accomplir le but général de la méthode qui est le maintient de la diversité de la population, cette fonction ajuste les cœfficients de pénalité selon la proportion d'individus faisables à la génération courante t. La définition de la fonction de pénalité est donnée par l'équation (4.2) et celle de l'ajustement de $\alpha(t)$ est donnée par l'équation (4.3).

Dans un premier temps, nous avons commencé par considérer un cœfficient de pénalisation unique $\alpha(t)$ pour toutes les contraintes. Ainsi, la faisabilité d'un individu est définie par toutes les contraintes du problème simultanément. Il suffit qu'il y ait une seule contrainte violée pour que l'individu devient infaisable. Autrement dit, l'ensemble des contraintes est considéré comme une seule contrainte, et $\alpha(t)$ est le cœfficient de pénalité pour toute la population à la génération t. La valeur de $\alpha(t)$ croît ou décroît d'un certain taux automatiquement, selon le degré de réalisabilité général de la population.

L'augmentation de la valeur du cœfficient de pénalité $\alpha(t)$ permet de favoriser les individus faisables pendant les procédures de sélection ultérieures, alors que sa diminution favorise les non faisables.

$$\begin{array}{ll} \text{si } (\tau_t > \tau_{target}) & \alpha(t+1)(\overrightarrow{x}) = \alpha(t)/fact \\ \text{sinon} & \alpha(t+1)(\overrightarrow{x}) = \alpha(t) * fact \end{array} \tag{4.3}$$

où $fact > 1$ est un paramètre de l'algorithme défini préalablement par l'utilisateur. Ainsi, la fonction d'évaluation de $\overrightarrow{x}$ devient:

$$eval(\overrightarrow{x}) = \left\{ \begin{array}{ll} f(\overrightarrow{x})\,, & si\,\overrightarrow{x} \in \mathcal{F} \\ f(\overrightarrow{x}) + \alpha(t)(\sum_{j=1}^{q} g_j^+(\overrightarrow{x}) + \sum_{j=q+1}^{m} |h_j|(\overrightarrow{x}))\,, & sinon. \end{array} \right.$$

Deux cas se présentent:

- $\alpha(t)$ prend des grandes valeurs: dans ce cas les individus infaisables sont pénalisés et ce sont les faisables qui occupent les meilleures positions dans le classement total de la population, et ont ainsi plus de chance de survivre;
- $\alpha(t)$ prend des petites valeurs: dans ce cas, si $\overrightarrow{x_1}$ est faisable et $\overrightarrow{x_2}$ est infaisable et $f(\overrightarrow{x_2})$ est équivalente ou meilleure de $f(\overrightarrow{x_1})$, $\overrightarrow{x_2}$ devient plus performant que $\overrightarrow{x_1}$ et a plus de chance d'être sélectionné pour la génération suivante. Ainsi, ce sont plutôt les individus faisables qui sont pénalisés.

L'oscillation de la pénalisation entre des petites et des grandes valeurs engendre en parallèle une oscillation du taux de faisabilité de la population entre 0 et 1. Afin de ne pas perdre tous les faisables quand les infaisables sont favorisés et garder ainsi un certain degré de faisabilité de la population (τ_t strictement supérieur à 0), une sélection ségrégationnelle a été introduite, permettant de favoriser les meilleures solutions faisables pendant le processus de sélection. Cette approche est présentée dans le paragraphe 4.2.3

La pénalisation initiale $\alpha(0)$ est calculée en utilisant la population initiale (initialisée par tirage uniforme sur $\mathcal{S}$):

$$\alpha(0) = \frac{\bar{F}}{\bar{V}} * 1000,$$

avec $\bar{F}$ est la moyenne des valeurs absolues de la fonction objectif f de tous les individus la population $P(0)$:

$$\bar{F} = \frac{\sum_{i=1}^{n} |f(\overrightarrow{x_i})|}{\mu}$$

et $\bar{V}$ est la moyenne des valeurs absolues des taux de violations de toute la population:

$$\bar{V} = \frac{\sum_{i=1}^{n} |v(\overrightarrow{x_i})|}{\mu},$$

où $v(\overrightarrow{x_i})$ est la somme des violations des différentes contraintes de l'individu $\overrightarrow{x}_i$, calculée comme suit:

$$v(\overrightarrow{x_i}) = \sum_{j=1}^{m} v_j(\overrightarrow{x_i}),$$

avec:

$$v_j(\overrightarrow{x_i}) = \begin{cases} max(0, g_j(\overrightarrow{x_i})), & pour\ 1 \leqslant j \leqslant q \text{ (contraintes d'inégalité)} \\ |h_j(\overrightarrow{x_i})|, & pour\ q+1 \leqslant j \leqslant m \text{ (contraintes d'égalité)} \end{cases}$$

Le choix de la définition de $\alpha(0)$ a été fait de telle façon que la pénalisation initiale domine la valeur de la fonction objectif pour garantir que tous les individus infaisables soient défavorisés. Ainsi, on force l'algorithme à retrouver rapidement des faisables. Cette définition permet d'éviter de chercher manuellement la bonne valeur initiale de $\alpha(0)$ adaptée à chaque type de problème.

4.2.2 Sélection/séduction basée sur les contraintes

Dans la majorité des problèmes d'optimisation réels, il est difficile de deviner préalablement la position de l'optimum dans l'espace de recherche, mais très souvent, on sait qu'une ou plusieurs contraintes sont actives à proximité de l'optimum global. (e.g. minimisation du coût de fabrication d'un objet tout en respectant les contraintes techniques). Dans ce cas, on peut avoir la certitude que l'optimum se trouve sur la frontière du domaine faisable. D'un autre côté, restreindre la recherche aux alentours des frontières, ce qui peut être très puissant quand c'est applicable [114], est très dépendant du problème. Par ailleurs, concentrer la recherche à l'intérieur du domaine $\mathcal{F}$ [117] [86] ne garantit pas une bonne exploration des frontières, mais peut être très bénéfique si l'optimum est à l'intérieur de $\mathcal{F}$.

En vue de permettre à l'algorithme une meilleure exploration des frontières et d'attirer les individus infaisables plus rapidement vers l'espace faisable, nous proposons d'utiliser, pendant l'hybridation, une stratégie de sélection appropriée: la stratégie de séduction/sélection [102, 53], où on croise les individus faisables avec les infaisables. Toutefois, pour que l'exploration de tout l'espace faisable continue, ce mécanisme n'est appliqué qu'aux faisables et à des moments précis de l'évolution. La décision de l'application de ce mécanisme dépend du taux de faisabilité de la population: si $\tau_t > \tau_{target}$, le processus d'exploration se déroule normalement et les couples à croiser sont choisis parmi les individus de toute la population selon leurs performances. Sinon, on oblige les individus faisables à se reproduire avec les infaisables, afin de générer des enfants proches des frontières du domaine faisable (figure 4.1). Plus précisément, pour un premier parent x_1 déjà sélectionné, la sélection de son partenaire (x_2) se fait comme suit:

si (0 <$\tau_t < \tau_{target}$) **et** (x_1) est faisable
    sélectionner (x_2) parmi les individus infaisables (selon sa fonction objectif)
sinon sélectionner (x_2) selon sa performance

Cette stratégie peut être appliquée avec tout type de croisement. Dans notre cas, nous avons choisi le croisement arithmétique afin d'accomplir l'objectif d'ASCHEA de produire des enfants proches de la frontière de l'espace réalisable pour une meilleure exploration de ses alentours.

Dans le chapitre 3, nous avons montré que le caractère contractant du croisement arithmétique limite sa capacité d'exploration. Avec les objectifs d'ASCHEA, ce caractère devient un avantage.

4.2.3 La Sélection Ségrégationnelle

Pour accroître la chance de survie des individus faisables (quand le cœfficient de pénalité est faible), un opérateur de sélection spécifique est utilisé dans cet algorithme. Cette sélection, appelée sélection ségrégationnelle, peut être vue comme une stratégie intermédiaire entre la méthode basée

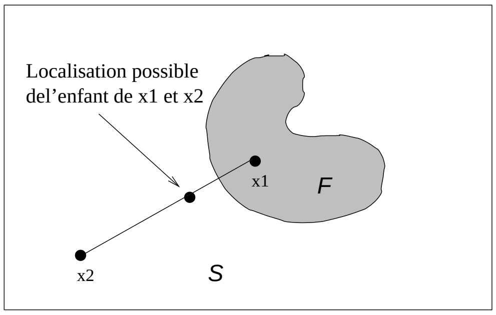


FIG. 4.1 – *Effet de la strégie de séduction/sélection utilisée dans le croisement.*

sur la supériorité des points faisables [95] (chapitre 2, section 2.3.5) et la méthode de remplacement utilisée dans le GA Ségrégationnel ("Segregated GA" [75] (chapitre 2, section 2.3.6)).

La sélection ségrégationnelle est un mécanisme de remplacement déterministe utilisé avec les Stratégies d'Evolution. Avec le schéma des ES, à partir de μ parents, λ enfants sont générés (chaque parent donne naissance à $\frac{\lambda}{\mu}$ enfants en moyenne). Parmi ces λ enfants (pour la stratégie ",") ou parmi les μ parents et les λ enfants (pour la stratégie "+"), les μ meilleurs individus sont sélectionnés pour devenir les nouveaux parents (chapitre 1, section 1.5.3). Avec le schéma de la sélection ségrégationnelle, les nouveaux μ parents sont sélectionnés de la manière suivante.

Soit $0 < \tau_{select} < 1$ une proportion définie préalablement, avec ($\tau_{select} < \tau_{target}$). L'opérateur de sélection ségrégationnelle commence par sélectionner, sans remplacement, parmi les individus faisables, en se basant sur leurs performances, jusqu'à ce que $\tau_{select} * \mu$ individus soient sélectionnés, ou jusqu'à qu'il n'y ait plus d'individus faisables disponibles. La sélection du reste de la population se fait en appliquant la sélection standard déterministe sur le reste des individus triés selon leurs performances (avec pénalisation), et sans tenir compte de leur faisabilité. Le remplacement est appliqué une fois les μ individus survivants déterminés.

Ainsi, seulement une proportion τ_{select} des individus faisables est considéré supérieure à tous les points infaisables. La sélection ségrégationnelle d'ASCHEA peut être vue comme celle du "GA Ségrégationnel" [75], quand le cœfficient de pénalité supèrieur s'approche de l'infini.

La procédure de la sélection ségrégationnelle se passe en deux étapes: d'abord la sélection de $\tau_{select} * \mu$ faisables (s'ils existent), ensuite on remplie la population avec une sélection déterministe standard. Son algorithme est comme suit:

Procédure `Sélection Ségrégationnelle`

`trier la population (par ordre décroissant des valeurs des performances)`
$N_s \leftarrow 0$ `(nombre d'individus sélectionnés)`
$\tau_f \leftarrow 0$ `(taux d'individus faisables sélectionnés)`
$i \leftarrow 0$
Tant que $(i < (\mu + \lambda)$ `et` $\tau_f < \tau_t$ `et` $\tau_f < \tau_{select})$
  Si $\overrightarrow{x_i}$ `est faisable`
    `sélectionner` $\overrightarrow{x_i}$
    $N_s \leftarrow N_s + 1$
    $\tau_f \leftarrow N_s / \mu$

```
        fin si
        i ← i + 1
    Fin tant que
    i ← 0
    Tant que (N_s < μ)
        Si x_i est non déjà sélectionné
            sélectionner x_i
            N_s ← N_s + 1
        fin si
        i ← i + 1
    Fin tant que
    Fin Procédure
```

4.2.4 Discussion

La première conséquence de l'utilisation de la sélection ségrégationnelle est une sorte d'élitisme des faisables: dès qu'un individu faisable apparaît, il ne peut disparaître de la population que s'il est remplacé par un autre faisable meilleur, même si le cœfficient de pénalité parvient à des très petites valeurs et les individus infaisables deviennent largement favorisés.

Cette sorte d'élitisme permet de conserver la trajectoire de recherche donnée par les meilleurs points faisables. Par ailleurs, l'apparition des points non faisables dominant la population aide l'algorithme à trouver d'autres trajectoires d'exploration de l'espace. En plus, la présence des deux types de solutions (faisables et infaisables), permet au croisement basé sur la séduction/sélection d'assurer son rôle, et créer ainsi de nouvelles trajectoires de recherche aux alentours des frontières. Ainsi, 3 types de trajectoires se créent au cours du processus d'exploration:

1. trajectoires données par les meilleurs points faisables,
2. trajectoires données par les meilleurs points infaisables,
3. trajectoires données par le croisement des faisables avec les infaisables basé sur le mécanisme de séduction/sélection.

Cette diversification assure une exploration rapide et efficace de l'espace de recherche. Ceci permet à ASCHEA de s'adapter à des différents types de problèmes et spécialement d'être capable de s'adapter aux deux localisations possibles de l'optimum global: sur la frontière du domaine faisable ou à l'intérieur de la région faisable.

Pour le premier cas, c'est la stratégie de séduction/sélection qui aide à explorer les deux côtés de la frontière. Dans le deuxième cas, ASCHEA, grâce à son adaptativité, permet à la population de rejoindre le domaine faisable. A ce moment, c'est la sélection standard qui est appliquée pour le croisement, et celle déterministe pour le remplacement. Ainsi, c'est l'exploration des ES standards qui prend place. Toutefois, on note que, pour ce deuxième cas, au début de l'évolution, la convergence de la population n'est pas définitive. En effet, tant qu'il y a des points proches de la frontière, la réapparition des solutions infaisables est toujours possible. Ceci ne peut être que bénéfique pour le processus de recherche, puisque leur présence permet leur hybridation avec les faisables, ce qui donne naissance à des points éparpillés dans l'espace faisable, donnant à l'algorithme plusieurs possibilités de trajectoires à l'intérieur de $\mathcal{F}$ avant la convergence. Cette diversification ne fait que stimuler le processus d'exploration, spécialement si la forme de l'espace faisable est complexe.

4.3 Expérimentation

Pour bien comprendre le comportement de notre algorithme au cours de l'évolution, nous présentons dans cette section une étude expérimentale détaillée de l'application d'ASCHEA sur deux cas test, sélectionnés dans [73]. La première fonction (G6) a deux contraintes d'inégalité non linéaires, et son optimum se trouve sur la frontière du domaine faisable (Fig. 4.2,(a)). La deuxième fonction (G8) a aussi deux contraintes d'inégalité non linéaires et son optimum se trouve au milieu de la région faisable (Fig. 4.2, (b)).

- **Minimiser** $G6(\overrightarrow{x}) = (x_1 - 10)^3 + (x_2 - 20)^3$,

sous les contraintes non linéaires suivantes:

$(x_1 - 5)^2 + (x_2 - 5)^2 - 100 \geqslant 0$,
$-(x_1 - 6)^2 - (x_2 - 5)^2 + 82.81 \geqslant 0$,

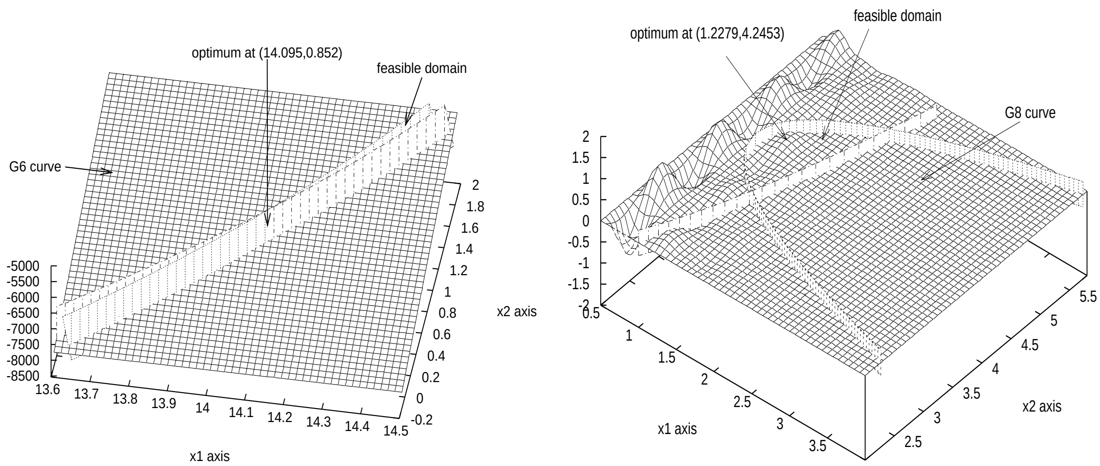
avec : $13 \leqslant x_1 \leqslant 100$ et $0 \leqslant x_2 \leqslant 100$.

- **Maximiser** $G8(\overrightarrow{x}) = \frac{\sin^3(2\pi x_1)\cdot\sin(2\pi x_2)}{x^3\cdot(x_1+x_2)}$,

sous les contraintes suivantes:

$x_1^2 - x_2 + 1 \leqslant 0$,
$1 - x_1 + (x_2 - 4)^2 \leqslant 0$

avec : $0 \leqslant x_1 \leqslant 10$ et $0 \leqslant x_2 \leqslant 10$.



(a): Fonction G6 (b): Fonction G8

FIG. 4.2 – *Surface des fonctions test. Les frontières sont rendues visibles artificiellement.*

L'optimum global de G6 est $\overrightarrow{x^*} = (14.095, 0.84296)$, avec $G6(\overrightarrow{x^*}) = -6981.81388$. Les deux contraintes du problème sont actives. La fonction G8 a beaucoup d'optima locaux, les plus hauts pics se trouvent au long de l'axe des x. L'optimum connu se trouve à l'intérieur du domaine faisable et a comme coordonnées:
$\overrightarrow{x^*} = (-1.737459, -0.167763)$, avec $G8(\overrightarrow{x^*}) = 0.095825$.

4.3.1 Conditions expérimentales

ASCHEA utilise un ES-(100+300) avec la stratégie de sélection ségrégationnelle décrite dans le paragraphe 4.2.3. La mutation utilisée est la mutation Gaussienne standard avec auto-adaptativité anisotrope des déviations standards [121] (chapitre 1, section 1.3.2), ayant comme valeur initiale 0.03. Le taux d'apprentissage global τ du processus auto-adaptatif des déviations est fixé à 0.1, alors que la taux local τ' a été fixé à 1. Le croisement adopté est le croisement arithmétique standard accompagné de la stratégie de séduction/sélection. Les deux opérateurs de reproduction (croisement/mutation) sont appliqués avec une probabilité de 0.9.

Pour toutes les expériences, les paramètres spécifiques à ASCHEA ont été choisis comme suit:

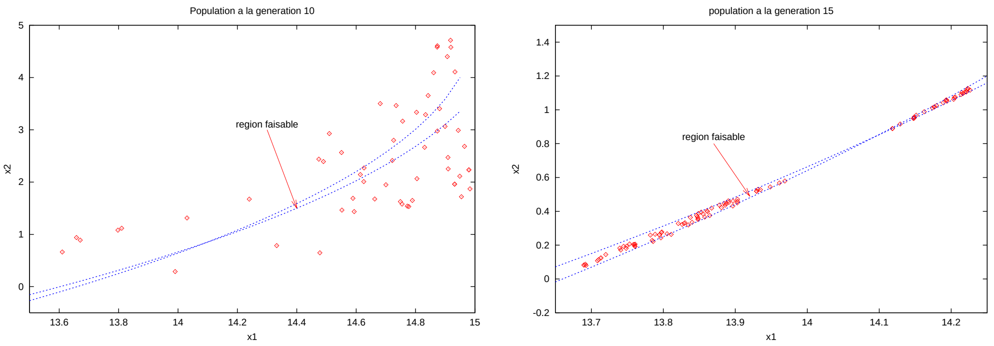
- $\tau_{select} = 0.3$
- $\tau_{target} = 0.6$
- $fact = 1.1$

Le choix de la valeur de τ_{select} permet de maintenir la réalisabilité d'au moins 30% de la population, alors que celui de la valeur de τ_{target} permet de garder un certain équilibre entre les faisables et les non faisables.

Une série d'au moins 31 expériences de 5000 générations a été effectuée pour chaque test.

4.3.2 Cas 1: Solution sur la frontière du domaine faisable (G6)

Pour ce cas test, ASCHEA a réussi à trouver l'optimum très rapidement, et ceci pour chaque essai (Fig. 4.3 et 4.5). Pendant les premières générations, dès que des individus faisables apparaissent, la sélection ségrégationnelle permet de les garder au sein de la population tant qu'il n'y a pas d'autres individus faisables meilleurs qui peuvent les remplacer. La stratégie de séduction/sélection pour le croisement et ainsi appliquée, ce qui augmente la proportion des points réalisables dans la population, et *attire* quelques uns vers la frontière du domaine $\mathcal{F}$. La figure 4.3 montre comment l'algorithme aide les individus à converger rapidement (5 générations) vers la région faisable.

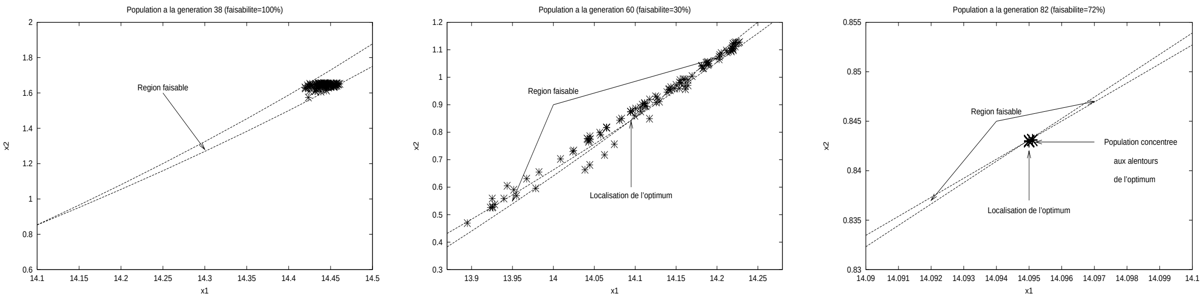


(a): Génération 10 (b): Génération 15

FIG. 4.3 – *Deux vues de la population de G6. Attention, l'échelle change pour la deuxième vue.*

La sélection déterministe des stratégies d'évolution permet de renforcer la capacité d'exploitation d'un algorithme, mais pas sa capacité d'exploration. En effet, avec ce genre de sélection, la population converge assez rapidement vers la zone des meilleurs trouvés (qui peut être la zone de l'optimum global comme elle peut être celle d'un optimum local). Ensuite, l'algorithme se concentre sur l'exploitation de cette zone afin de s'approcher le maximum possible de la localisation exacte de l'optimum (c'est surtout le rôle de la mutation auto-adaptative). Ainsi, l'exploration du reste de l'espace de recherche n'est plus assurée pendant les phases avancées de l'évolution.

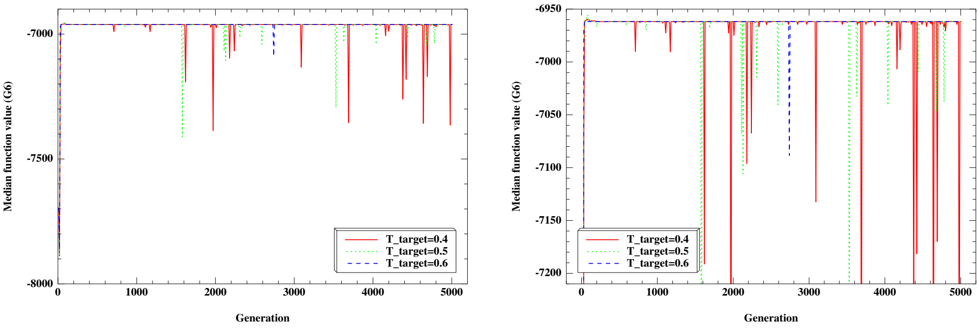
Avec la fonction de pénalisation adaptative d'ASCHEA, l'exploration de tout l'espace de recherche continue, puisque la diversité est maintenue au long de l'évolution. En effet, la réintégration des infaisables dans la population est fréquente, dès qu'ils deviennent meilleurs que les faisables, grâce à la fonction de pénalisation. La figure 4.4 montre bien comment la population converge vers la zone de l'optimum ($\alpha(t)$ croissante), puis diverge rapidement ($\alpha(t)$ décroissante) pour s'étendre sur un espace beaucoup plus large qu'il soit réalisable ou non.



(a): Taux de faisabilité 100% (b): Taux de faisabilité 30% (c) Taux de faisabilité 72%

FIG. 4.4 – *Vue de la population de G6 à la génération 38 (a), 60 (b) et 82 (c). Attention, l'échelle change d'une vue à une autre.*

Ce processus continue tout au long de l'évolution, et fait que le meilleur dans la population peut devenir irréalisable: les pics dans la figure 4.5 (les valeurs basses correspondent aux meilleures solutions non faisables).



(a) (b): zoom

FIG. 4.5 – *Valeur médiane des meilleures solutions trouvées sur 31 essais (pour le cas test G6) avec des différentes valeurs de τ_{target} (0.4, 0.5, 0.6). On remarque que le nombre de pics correspondant aux meilleures solutions infaisables augmente avec la baisse de la valeur de τ_{target}.*

Cependant, ASCHEA ne perd jamais son meilleur faisable, grâce à la sélection ségrégationnelle. En effet, même si les individus non faisables arrivent à occuper les premières places dans le classement général de la population, un certain nombre (τ_{select}) d'individus faisables (y compris le meilleur) sont sélectionnés en premier lieu. Ainsi, on assure un certain degré de faisabilité de la population tout au long de l'évolution (figure 4.6), et on assure que les trajectoires d'exploration données par les meilleurs points dans l'espace $\mathcal{F}$ sont retenues par l'algorithme et bien exploitées.

La fréquence d'apparition des meilleurs infaisables dans la population dépend de la valeur de τ_{target}. La figure 4.5 montre que le nombre de pics correspondants aux meilleurs infaisables augmente avec la baisse de la valeur de τ_{target}. En effet, des petites valeurs de τ_{target} favorisent plus les infaisables, alors que des grandes valeurs favorisent plus les faisables.

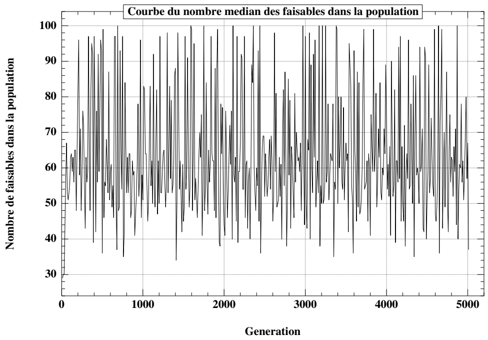


FIG. 4.6 – *Nombre médian de faisables dans la population au cours de la résolution de la fonction G6. Le degré de faisabilité de la population est très variable au cours de l'évolution, mais il ne descend jamais au dessous d'un certain seuil ($\tau_{select} = 30\%$).*

La figure 4.6 illustre le phénomène de disparition/réapparition des individus non faisables pendant les 5000 générations d'évolution. En fait, ce phénomène est présent tant que l'algorithme tourne. La réapparition facile des infaisables est due à la localisation de la population qui se concentre aux alentours de la frontière à côté de l'optimum (figure 4.4 (c)). La proximité des individus des bords de $\mathcal{F}$ induit qu'une petite mutation peut projeter un faisable de l'autre côté de la frontière, il devient ainsi infaisable. Par conséquent, dans le cas où l'optimum se trouve sur la frontière de $\mathcal{F}$, il n'y a pas de convergence définitive de la population. En effet, pour le cas de G6, l'optimum est trouvé en moyenne à la génération 84, mais l'exploration continue pendant les 5000 générations de chaque essai.

La figure 4.7 illustre la distance médiane (sur 31 essais) entre l'individu ayant la meilleure performance et l'optimum réel du problème. Les pics dans la zone inférieure de la courbe ($distance \simeq 10^{-4}$) indiquent que le meilleur est très proche de l'optimum, alors que ceux de la zone supérieure indiquent que le meilleur courant s'est éloigné de l'optimum. ASCHEA réussi à approcher de très prêt la fitness de l'optimum dès les premières générations. Il arrive à le localiser avec précision dans l'espace après la génération 1000, mais cette convergence n'est pas définitive. En effet, le meilleur point courant change de localisation selon le degré de faisabilité de la population et les taux de pénalisation. Sa trajectoire au long de l'évolution oscille entre la position de l'optimum à l'intérieur de $\mathcal{F}$ (qui, une fois trouvé, ne quitte plus la population), et les zones des optima locaux à l'extèrieur de $\mathcal{F}$.

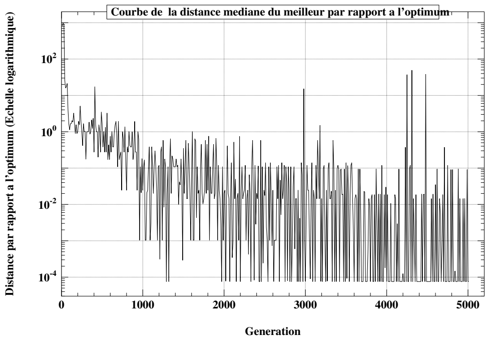
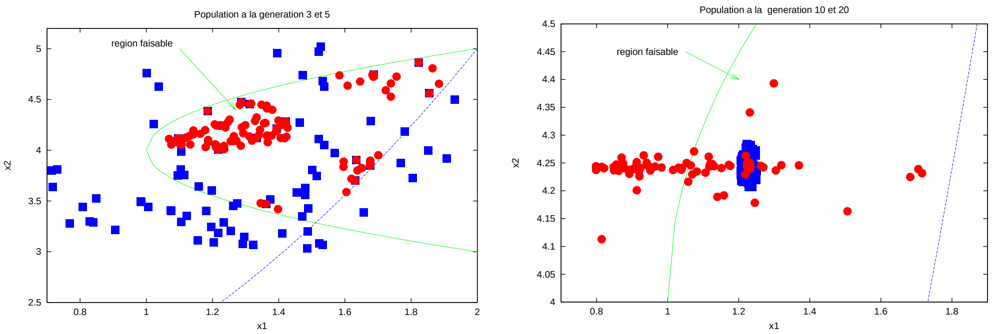


FIG. 4.7 – *Distance euclidienne médiane entre le meilleur courant (indépendamment de la faisabilité) et l'optimum exact du problème G6. Les oscillations montrent que le meilleur n'est pas stable et change selon l'état général de la population courante.*

4.3.3 Cas 2: La solution à l'intérieur du domaine faisable (G8)

La deuxième localisation possible de l'optimum est à l'intérieur du domaine faisable, loin de la frontière. C'est le cas pour la fonction G8. Comme pour le premier cas, 31 tests indépendants ont été effectués, qui ont tous donné la valeur exacte de l'optimum. La figure 4.8 montre le comportement de la population pendant les 20 premières générations. Pendant les 3 premières générations, les individus étaient bien répartis sur tout l'espace de recherche, réalisable et non réalisable (individus cercles bleus dans la figure 4.8 (a)). Mais l'algorithme les ramène en totalité et rapidement au domaine faisable, et à la cinquième génération, toute la population devient faisable (individus carrés rouges dans la figure 4.8 (a)). Cette convergence continue jusqu'à la génération 10, où toute la population se concentre aux alentours de l'optimum (figure 4.8 (b), individus carrés bleus) . Pendant ces générations, le processus d'évolution continue normalement avec une sélection déterministe standard, alors que la valeur du cœfficient de pénalisation décroît rapidement. Ainsi, pendant les quelques générations suivantes, l'algorithme attire quelques individus vers l'extérieur du domaine faisable (individus cercles rouges dans la figure 4.8 (b)). Cependant, grâce à la sélection ségrégationnelle, une fraction $\tau_{select} = 0.3$ de la population reste faisable, et les meilleurs faisables ne sont jamais perdus.

Une fois l'optimum est localisé, la population a tendance à converger vers sa zone à l'intérieur de l'espace réalisable et loin de la frontière. Ainsi, la réapparition des infaisables devient un peu plus difficile que pour le cas de G6, puisque les individus ne sont plus aux proximités des frontières. Ce phénomène donc se manifeste pendant la première phase de l'évolution, mais une fois toute la population a convergé vers la zone de l'optimum, la recherche commence à se stabiliser. En effet, quand la population devient 100% faisable (en moyenne, le taux de faisabilité de 100% est atteint à la génération 78, figure 4.9) et s'approche de la zone de l'optimum, le cœfficient de pénalité commence à se décrémenter. Mais pour atteindre des petites valeurs permettant de favoriser les infaisables, il met un certain temps, pendant lequel la population continue de converger vers l'optimum et la diversité de la population diminue. Si la population devient uniforme (degré de similarité à l'optimum élevé), la réapparition des individus de l'autre côtés de la frontière devient très difficile, même avec des taux de pénalisation largement petits, puisqu'elle devient dépendante de la mutation uniquement. Cette dernière, étant auto-adaptative, ne pourra pas générer des points



(a): Génération 3 et 5 (b): Génération 10 et 20

FIG. 4.8 – *Des vues de la population de G8 au cours d'un essai.*

très différents des parents. Ainsi, si la population est loin de la frontière (concentrée au près de l'optimum à l'intérieur de $\mathcal{F}$), l'algorithme trouve de difficulté à assurer sa diversité tout au long de l'évolution, au contraire de ce qui a été observé pour le cas de G6.

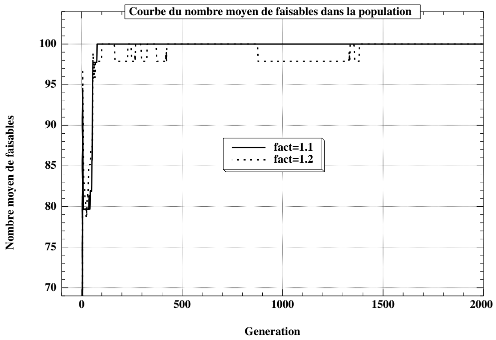


FIG. 4.9 – *Nombre moyen de faisables dans la population au cours de la résolution de G8, pour deux valeurs de $fact$: 1.1 et 1.2.*

Cette convergence plus ou moins rapide n'est pas de tout gênante pour le processus de recherche dans le cas de G8, puisque l'optimum est trouvé dès les premières générations (en moyenne à la 32ème génération). ASCHEA arrive à maintenir la diversité de la population pendant encore quelques générations (pour quelques dizaines de générations). Cependant , si on est en présence d'un problème plus difficile nécessitant un temps de recherche plus lent, ASCHEA reste capable de stimuler la diversification de la population pendant un long moment de l'évolution. Pour se faire, il suffit de diminuer le temps nécessaire pour donner aux cœfficients de pénalisation des petites valeurs permettant de favoriser les infaisables. Ceci est possible si on augmente légèrement la valeur de $fact$, utilisé pour l'augmentation et la diminution des cœfficients de pénalité. Pour les expériences décrites ci-dessus, $fact$ a comme valeur 1.1. Une valeur plus élevée (e.g. $fact = 1.2$) permet de décrémenter plus rapidement $\alpha(t)$, avant que la population ne converge totalement vers la zone de l'optimum (quelques faisables sont encore dispersés dans $\mathcal{F}$), ce qui facilite la réapparition des points du côté irréalisable de l'espace de recherche, mais leur disparition est aussi plus rapide.

La figure 4.9 illustre ce phénomène. On voit bien que le nombre moyen de faisables dans la population avec $fact = 1.1$ se stabilise à 100 dès la génération 78, alors que celui donné par $fact = 1.2$ continue à varier jusqu'à la génération 1383. Cependant, même si la diversité continue plus longtemps avec $fact = 1.2$ qu'avec $fact = 1.1$, elle fini par disparaître. La nature auto-adaptative de la mutation utilisée ne permet pas de générer des fortes perturbations à une phase avancée de l'évolution, ce qui rend la chance de réapparition des infaisables très faible. Ceci est dû à la diminution progrossive des valeurs des déviations standards au cours de leur modification pendant la mutation. La courbe 4.10 illustre les valeurs médianes sur 31 essais des déviations standars maximales enregistrées sur toutes les déviations locales de tous les individus de la population.

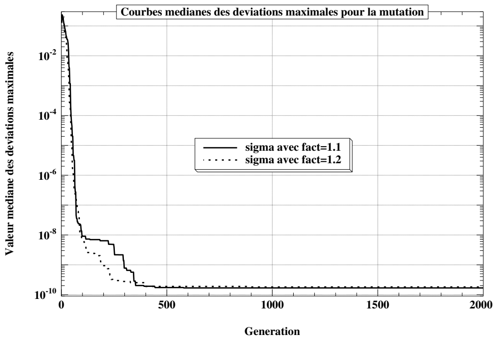


FIG. 4.10 – *Courbes médianes des déviations standards maximales enregistrées pour la mutation auto-adaptative au cours de la résolution de G8, pour $fact = 1.1$ et $fact = 1.2$, représentée à l'échelle logarithmique.*

Quand l'optimum est sur la frontière, la génération d'individus infaisables par la mutation auto-adaptative est facile, étant donné qu'elle n'a pas besoin d'appliquer des grandes modifications sur les individus, puisqu'ils sont proches du bord de $\mathcal{F}$. Si l'optimum est à l'intèrieur de $\mathcal{F}$, pour maintenir la diversité aussi long temps que l'algorithme tourne comme pour le cas de G6, il est nécessaire d'utiliser une mutation capable de générer des faibles et fortes perturbations tout au long de l'évolution. Dans le chapitre précédent, nous avons proposé la mutation logarithmique qui vérifie cette condition. L'utilisation de cette mutation permet de retrouver le phénomène d'oscillation de taux de faisabilité observé pour le cas de G6. Il est illustré par la courbe 4.11.

4.4 Etude des composantes d'ASCHEA

Après avoir montré le comportement général d'ASCHEA sur deux cas test, une deuxième étude expérimentale sur les mêmes cas test est présentée dans cette section, dont le but est d'expliciter le rôle de chaque composante d'ASCHEA. Un moyen de comprendre le rôle d'une procédure composante est de voir le comportement de l'algorithme en son absence. Nous avons alors réalisé des expériences en remplaçant à chaque fois une des composantes par une autre procédure standard ayant le même objectif de base.

La première étude se porte sur la fonction de pénalisation adaptative. Cette dernière a été remplacée par une pénalisation statique, dont les cœfficients sont choisis par l'utilisateur et ne sont pas adaptés au cours de l'évolution. Les autres composantes concernant le croisement et la sélection sont maintenues.

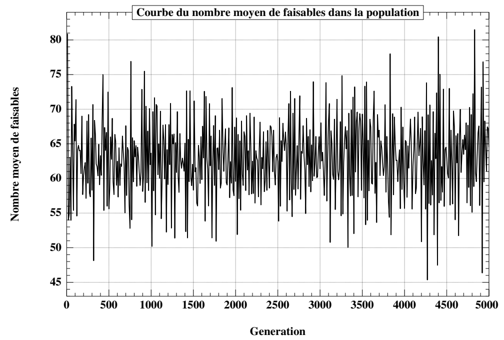


FIG. 4.11 – *Nombre moyen de faisables dans la population au cours de la résolution de G8 en utilisant la mutation logarithmique.*

Pour le deuxième cas, c'est la sélection ségrégationnelle qui a été remplacée par une sélection déterministe standard, et la pénalisation et le croisement restent inchangés.

Enfin, des tests ont été effectués avec un croisement basé sur une sélection standard des parents, indépendamment de leur faisabilité.

4.4.1 Premier cas: Pénalité statique

Pour étudier l'importance de la fonction de pénalisation d'ASCHEA, cette dernière a été remplacée par une pénalisation statique. Les individus infaisables sont alors évalués avec la fonction 4.1, mais le cœfficient de pénalisation $\alpha(t)$ n'est plus adapté au cours de l'évolution.

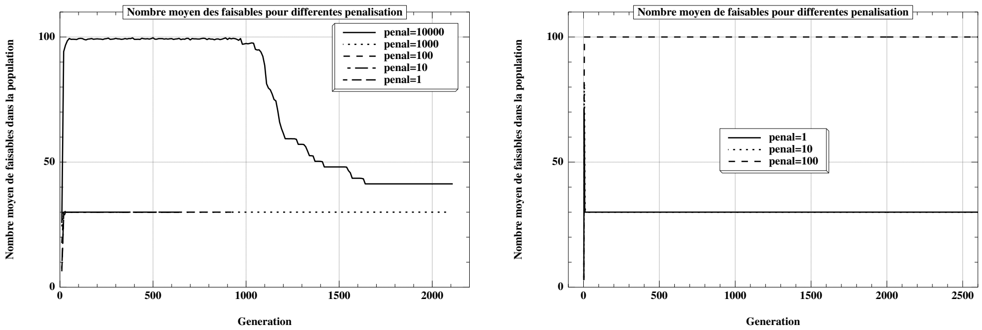
Des séries de 31 tests avec des différentes valeurs de α ont été effectuées pour les cas test G6 et G8. Les valeurs de α sont comprises entre 1 et 1000, afin de visualiser l'effet du choix des petites et des grandes valeurs de pénalisation.

A part la pénalisation, ces tests ont été réalisés dans les mêmes conditions expérimentales que celle de la section 4.3.

Pour trouver la pénalisation adéquate pour un problème donné, on est obligé de tester plusieurs valeurs jusqu'à ce qu'on trouve celle qui permet de donner des résultats satisfaisants. Si la pénalisation est faible, la population tend à converger vers les optima locaux à l'extérieur de $\mathcal{F}$. Cependant, grâce à la sélection ségrégationnelle, une portion τ_{select} des faisables réussit à survivre. Ce phénomène se présente pour chacun des deux cas test G6 et G8. Les courbes de la figure 4.12 montrent que, quand la pénalisation est faible ($\alpha = 1, 10, 100, 1000$ pour G6 et $\alpha = 1, 10$ pour G8), le degré de faisabilité de la population est limité à τ_{select}, assurée par la sélection ségrégationnelle.

En fait, la population se divise en deux groupes de taille fixe. Le premier groupe est celui des faisables, de taille $\tau_{select} * \mu$ (μ est la taille de la population). Le deuxième est celui des infaisables, de taille $(1 - \tau_{select}) * \mu$, et il converge vers les zones des optima locaux à l'extérieur de $\mathcal{F}$. La figure 4.13 montre le phénomène de partition de la population pour les cas tests G6 et G8. On y voit clairement la convergence rapide de chaque groupe vers les points dominants.

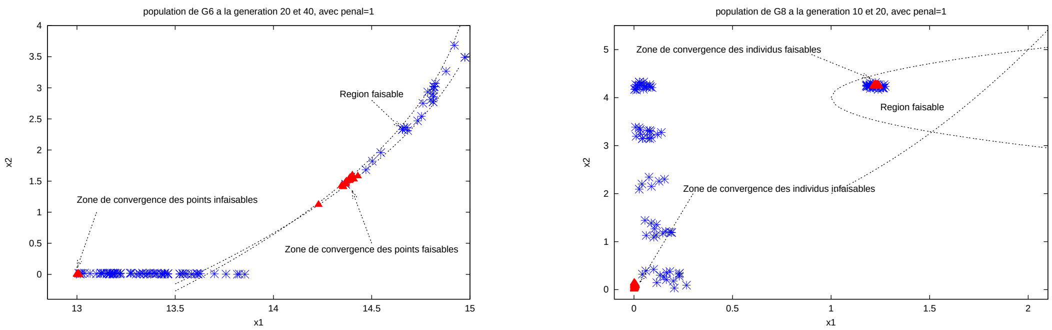
Après cette convergence, l'état de la population se stabilise et aucun changement n'est remarqué. Il est possible, par chance, que le meilleur du groupe des faisables soit l'optimum recherché. Pour le cas de G8, l'optimum a été localisé. Mais si le problème est plus difficile, la probabilité de le trouver devient faible. C'est le cas quand la région faisable est très petite. Dans ce cas, la chance de



(a): Cas test G6 (b): Cas test G8

FIG. 4.12 – *Courbes des nombres moyens des faisables dans la population sur 31 essais, avec différentes valeurs de pénalisation.*

trouver des faisables dès les premières générations est faible, et la population va converger vers les zones des optima locaux infaisables. C'est le cas aussi quand l'espace faisable est dispersé. Dans ce cas, le groupe des faisables se dirige vers la zone des premiers points faisables trouvés. Si cette zone n'est pas celle de l'optimum global, on risque la convergence prématurée, vue que la population perd sa diversité rapidement.



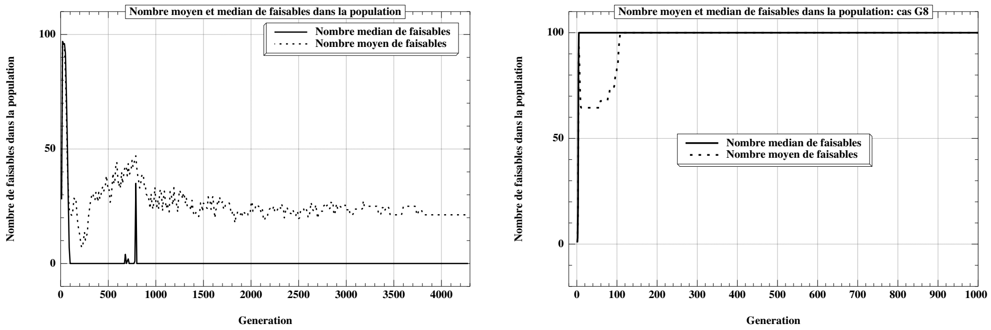
(a)G6: Génération 20 (étoiles) et 40 (triangles) $\alpha = 10$ (b)G8: Génération 10 (étoiles) et 20 (triangles) $\alpha = 1$

FIG. 4.13 – *Population de G6 (a) à la génération 20 et 40 et celle de G8 (b) à la génération 10 et 20 pour un essai de résolution avec une pénalité statique.*

Par contre, si la pénalisation est très élevée ($\alpha = 10000$ pour G6 et $\alpha = 100$ pour G8), dès que des individus faisables apparaissent, les infaisables sont éliminés de la population, et le taux de faisabilité atteint 100% rapidement (courbe 4.12). La population suit à une grande vitesse le trajet des points dominants et son état se stabilise à une phase prématurée de l'évolution. Cependant, pour le cas de G6, vu que l'optimum est sur la frontière de $\mathcal{F}$, quand toute la population vient se concentrer à ses alentours, il est possible que la mutation génère des individus infaisables, mais toujours tout près de la frontière. A ce moment, contrairement à ce qui est observé à la première phase de l'évolution, la pénalisation n'est plus assez élevée pour défavoriser tel individus au moment de la sélection, puisque le taux de violation est très faible. Ils s'imposent donc de nouveau dans la population (figure 4.12, courbe (a)). Heureusement que la sélection ségrégationnelle ne permet pas la perte des meilleurs faisables.

4.4.2 Deuxième cas: Remplacement de la sélection ségrégationnelle

La deuxième étude dans cette section porte sur l'apport de la sélection ségrégationnelle à l'efficacité d'ASCHEA. Nous l'avons alors remplacé par une sélection déterministe standard, tout en appliquant la pénalisation adaptative et la stratégie de séduction/sélection pour le croisement. Des séries de 31 essais ont été réalisées pour les cas test G6 et G8, toujours dans les mêmes conditions expérimentales.



(a): Cas test G6 (b): Cas test G8

FIG. 4.14 – *Courbes des nombres moyens et médians de faisables dans la population sur 31 essais pour le cas tes G6 (a) et le cas test G8 (b).*

Les résultats expérimentaux montrent que la sélection ségrégationnelle est primordiale pour ASCHEA afin qu'il puisse retourner une solution faisable à la fin de l'évolution, spécialement dans les cas où l'optimum se trouve sur la frontière du domaine faisable. En effet, avec la pénalisation adaptative, les infaisables peuvent devenir meilleurs que les faisables. Ils sont alors favorisés au cours de la sélection déterministe, tant que le cœfficient de pénalisation est trop petit. ASCHEA risque alors de perdre la trace des individus faisables et retourner une solution infaisable à la fin de l'évolution. C'est ce qui est passé pour plus de la moitié des essais du cas G6. Pendant quelques tests, avec l'augmentation de $\alpha(t)$, les individus ont retrouvé le chemin vers l'espace réalisable, et ASCHEA a pu retourner l'optimum recherché. Pour tous les autres essais, bien que l'optimum a été localisé à un certain moment de l'évolution, la diminution du cœfficient de pénalisation a causé la convergence de toute la population vers les optima locaux dans l'espace irréalisable, et elle n'a pas réussi ensuite à retrouver le chemin vers $\mathcal{F}$ (courbe (a) de la figure 4.14). Même la stratégie de séduction/sélection, qui normalement aide l'algorithme à attirer les individus vers l'espace faisable, ne joue plus aucun rôle puisqu'il n'y a que des infaisables dans la population. Le seul moyen d'assurer la faisabilité d'une proportion de la population est de favoriser quelques individus réalisables pendant la sélection, précisément les meilleurs, ce qui est l'idée de la sélection ségrégationnelle.

Pour le cas où l'optimum se trouve à l'intérieur de $\mathcal{F}$, au moment où le cœfficient de pénalisation prend des grandes valeurs, toute la population converge vers les zones des meilleurs individus faisables. En quelques générations, la population devient 100% faisable (courbe (b) de la figure 4.14) et son état commence à se stabiliser. Si le problème n'est pas difficile (comme le cas de G8), l'exploration standard des ES est suffisante pour localiser l'optimum. La sélection déterministe peut, dans ce cas, s'adapter avec les autres composantes d'ASCHEA.

Toutefois, même si l'optimum est à l'intérieur de $\mathcal{F}$, il y a toujours le risque que la population converge de l'autre côtés de la frontière. Supposons qu'au début de l'évolution, les individus infaisables, même pénalisés, soient meilleurs que les quelques individus faisables trouvés. Si ces derniers ne sont pas gardés afin qu'ils attirent les autres individus, l'algorithme aura des difficultés à trouver d'autres points dans $\mathcal{F}$ une fois que la population a convergé vers les zones des meilleurs infaisables. Si tous les individus sont concentrés dans la même zone infaisable, même un énorme cœfficient de pénalisation ne fait aucune différence entre eux, puisqu'ils ont des taux de violations trés proches. A ce moment, la pénalisation devient inutile.

On peut en conclure que la sélection ségrgationnelle est indispensable à ASCHEA pour garantir la réalisabilité des solutions finales, indépendamment de la localisation de l'optimum global.

4.4.3 Troisième cas: Remplacement de la stratégie de séduction/sélection

Contrairement aux deux autres composantes, l'absence de la stratégie de séduction/sélection pour le croisement ne semble pas affecter sensiblement la performance d'ASCHEA. Les tests réalisés pour les fonctions G6 et G8 avec une sélection standard pour le croisement ont tous rendu l'optimum facilement et rapidement. Pour le cas où l'optimum se trouve sur la frontière du domaine faisable, ASCHEA réussit à explorer ces zones grâce à la diversité de la population maintenue par la pénalisation adaptative. Avec la présence des individus faisables et infaisables dans la population, il y a toujours une certaine probabilité que les parents sélectionnés pour le croisement soient d'espèces différentes. Ainsi, l'exploration des frontières est toujours assurée.

Pour le deuxième cas, l'exploration de la frontière n'est pas nécessaire, vue que l'optimum est à l'intérieur de $\mathcal{F}$.

La question qui se pose alors est "quel rôle joue la séduction/sélection dans la procédure de recherche d'ASCHEA?" Pour répondre à cette question, nous avons effectué des séries d'expériences de 31 essais pour les cas tests G6 et G8 avec une sélection standard pour le croisement, basée sur la performance des individus. Les paramètres d'ASCHEA restent inchangés.

La comparaison des résultats des ces expériences avec les résultats précédents montre que le temps moyen mis par l'algorithme pour localiser l'optimum a changé. Le tableau 4.1 illustre ce changement.

	Avec *séduction/sélection*		Sans *séduction/sélection*	
	Meilleur Nb Gen	Nb Gen Moyen	Meilleur Nb Gen	Nb Moyen Gen
G6	42	85	51	99
G8	12	32	11	23

TAB. 4.1 – *Nombre de générations minimal et moyen mis par ASCHEA pour localiser l'optimum de G6 et G8 avec et sans la sélection basée sur la séduction/sélection pour le croisement.*

La conclusion qu'on peut déduire à partir des résultats de G6 est que la stratégie de séduction/sélection joue le rôle de stimulateur du processus d'exploration de la frontière. Elle permet à l'algorithme de visiter plus rapidement les zones de la frontière, en générant plus d'enfants dans ses alentours. Les résultats du tableau 4.1 pour le cas test G6 confirment cette conclusion. Le temps moyen mis par ASCHEA pour localiser l'optimum passe de 85 générations avec la stratégie de séduction/sélection à 99 générations avec une sélection standard. Etant donné que pour la majorité des problèmes sous contraintes l'optimum est sur la frontière, cette stratégie ne peut être que bénéfique pour la procédure d'exploration, spécialement si l'espace faisable a une forme complexe (contraintes non linéaires) et l'exploration de ses bords est difficile.

Par contre, pour le cas où l'optimum est á l'intérieur de l'espace faisable, cette stratégie n'aide pas beaucoup l'algorithme (tableau 4.1, cas G8), mais elle n'est pas gênante, vue que l'exploration des zones internes de $\mathcal{F}$ est toujours assurée.

4.5 Plus de résultats expérimentaux

En vue de montrer la robustesse de notre approche, nous avons étendu l'étude expérimentale sur les 11 cas test proposés dans [87] et [73] (de G1 à G11 – voir Annexe A). Pour chaque cas, 31 essais ont été effectués dans les mêmes conditions expérimentales que celles décrites dans la section 4.3.1, exception faite pour la fonction G10, où les résultats sont plus significatifs avec une probabilité de mutation égale à 0.3. En effet, l'approche proposée ne peut être robuste que si l'algorithme trouve des individus faisables. Dans le cas de G10, un taux de mutation élevé diminue la chance de trouver des solutions valides. Ainsi, pour cette fonction, la population est mutée avec des faibles probabilités.

Les résultats des différentes expériences sont résumés dans le tableau 4.2. Toutes les solutions trouvées pour chaque cas sont faisables. L'optimum exact a été trouvé pour 7 problèmes parmi les 11 étudiés.

Pour les problèmes avec contraintes d'égalité (G3, G5 et G11), les équations ont été remplacées par des inéquations, où chaque fonction doit être inférieure à un nombre positif très petit ϵ ($h_j(\overrightarrow{x}) \leqslant \epsilon$). Pour la partie expérimentale, ϵ a été fixé à 10^{-16}. On note que cette stratégie de prendre en compte des contraintes n'est pas standard. En effet, la pratique générale est de transformer l'égalité par une couple d'inégalités ($-\epsilon \leqslant h_j(\overrightarrow{x}) \leqslant \epsilon$), permettant de limiter le taux d'erreur pour un problème donné. La majorité des méthodes dans la littérature utilisent $\epsilon = 10^{-3}$ ou $\epsilon = 10^{-4}$. Cependant, cette stratégie limite la performance d'ASCHEA. Nous avons donc choisi d'utiliser une seule inégalité. D'autres solutions sont proposées dans le chapitre 6 pour généraliser la méthode.

La première conclusion qu'on peut faire à partir du tableau 4.2 est que ASCHEA a donné des résultats très satisfaisants pour toutes les fonctions testées, même pour celles soumises à des contraintes d'égalité, où le remplacement des équations par des inégalités s'est avéré efficace pour les 3 problèmes avec contraintes d'égalité (G3, G5 et G11).

Par ailleurs, la qualité moyenne des résultats de G2 est due spécialement au manque de nichage, permettant d'éviter la convergence prématurée vers un des plusieurs optima locaux entourant la solution globale. Une stratégie de nichage sera utilisée pour une version plus optimisée d'ASCHEA, introduite dans le chapitre suivant.

Entre autres, il est possible d'augmenter la performance de l'algorithme et ainsi améliorer encore la qualité des résultats en ajustant avec précision ses paramètres. Par exemple, pour le cas de G3*, le nombre d'essais réussis (optimum trouvé) croit avec le temps de calcul permis à l'algorithme. En effet, plus on augmente le nombre d'itérations, plus la qualité des résultats s'améliore. Ceci est dû à la grande capacité d'ASCHEA de poursuivre l'exploration de l'espace de recherche, même à une phase très avancée de l'évolution. Cette capacité est encore plus grande si l'optimum est sur la frontière (c'est le cas de G3 après avoir transformé la contrainte d'égalité en inégalité). Le tableau 4.3 confirme cette observation.

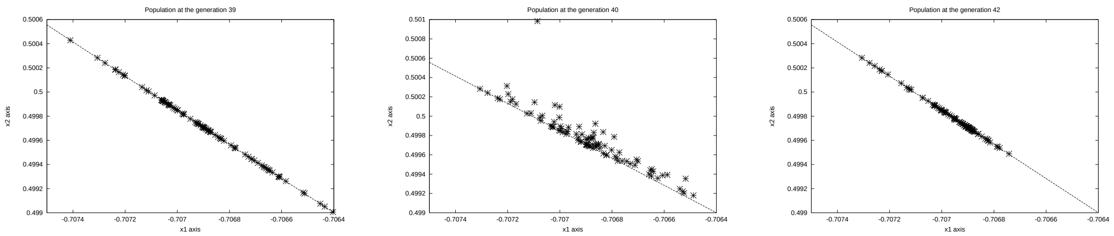
Fonction	valeur de l'Opt.	Opt. trouvé			Après 5000 gen		
		Nb. succès	Meilleur (nb gen)	Moy. (nb gen)	Meilleur	médian	Moy.
G1	-15	24	249	261	**-15**	-15	-14.6819
G2	0.803553	–	–	–	0.800781	0.506	0.5462
G3*	1.0	1	4205	4205	**1**	0.999979	0.999844
G4	-30665.5	7	109	159	**-30665.5**	-30658.9	-30650.745
G5*	5126.4981	1	53	53	**5126.49**	5162.98	5202.75
G6	-6961.81	31	42	85	**-6961.81**	-6961.81	-6961.81
G7	24.3062	–	–	–	24.36	24.505	24.6109
G8	0.095825	31	12	32	**0.095825**	0.095825	0.095825
G9	680.630	1	4267	4267	**680.630**	680.637	680.648
G10	7049.33	–	–	–	7095.15	7840.67	8858.64
G11*	0.75	31	27	61	**0.75**	0.75	0.75

TAB. 4.2 – *Résumé des résultats pour les 11 cas test, donnant pour chaque cas lameilleure performance, la performance , lmédiane et moyenne des meilleures solutions faisables trouvées sur les 31 essais, après 5000 générations (les 3 dernières colonnes). Quand l'optimum est trouvé au moins une fois, les 3 premières colonnes donnent le nombre de succès (optimum trouvé), le meilleur temps et le temps moyen (en nombre de générations) mis par l'algorithme pour trouver l'optimum (que pour les essais munis avec succès). Les fonctions G2 et G3 ont été testées avec, respectivement, 20 et 10 variables.*

Après 5000 gen		Après 10000 gen		Après 20000 gen	
Moy.	Nb succès	Moy.	Nb succès	Moy.	Nb succès
0.99984	1	0.99990	3	0.99999	7

TAB. 4.3 – *Résultats donnés par l'algorithme pour le cas test G3* avec des différents nombres de générations.*

Le remplacement des contraintes d'égalité par des inégalités s'est avéré efficace pour les cas de G3, G5 et G11. Pour ce dernier, la convergence est encore plus rapide, vue que l'exploration des frontières est plus facile. En effet, le temps moyen de convergence est de 61 générations. La figure 4.15 montre l'évolution de la population entre les générations 39 et 42, illustrant le processus d'exploration des frontières.



(a): $\tau_{39} = 0.95$, $\alpha(40) = 0.5998$ (b): $\tau_{40} = 0.3$, $\alpha(41) = 0.6578$ (c): $\tau_{42} = 0.64$, $\alpha(43) = 0.7257$

FIG. 4.15 – *Population de G11 aux générations 39 (a), 40 (b), 42 (c).*

N.B. Nous signalons que les résultats du cas test G5 ne correspondent pas à ceux que nous avons publié dans [49]. En effet, les résultats de [49] sont ceux d'une fonction très ressemblante à la fonction référence G5 étudiée dans la littérature sa définition exacte est donnée dans l'annexe A). Cette différence est due essentiellement à un manque de précision de notre première source, où nous avons extrait cette fonction.

Une autre étude paramétrique a été effectuée pour le paramètre qui peut avoir une grande influence sur la performance d'**ASCHEA** et qui n'est autre que le taux limite d'augmentation et de diminution du cœfficient de pénalisation: τ_{target}. Par exemple, pour le cas de la fonction G2, une valeur plus élevée de τ_{target} et $fact$ a des résultats meilleurs que ceux du tableau 4.2: avec $\tau_{target} = 0.7$ et $fact = 1.3$, l'algorithme a donné comme meilleure performance de G2 0.80603 avec une moyenne de 0.606.

Par contre, pour le cas test G4, **ASCHEA** a donné des meilleurs résultats avec des valeurs de τ_{target} plus petites (tableau 4.4). Ceci peut être dû au fait que l'optimum de ce problème se trouve sur la frontière du domaine faisable. En effet, des valeurs moins élevées de τ_{target} accentue l'exploration des frontières du domaine faisable, puisque les non faisables réapparaissent plus fréquemment dans la population, ce qui favorise le croisement d'individus d'espèces différentes (faisables avec infaisables). Ainsi, les chances de naissance des enfants sur la frontière augmente. Ce phénomène est expliqué avec plus de détails dans le paragraphe 4.3.2.

$\tau_{target} = 0.6$	$\tau_{target} = 0.55$	$\tau_{target} = 0.5$	$\tau_{target} = 0.45$	$\tau_{target} = 0.4$	$\tau_{target} = 0.35$
-30650.745 (7)	-30656.56 (10)	-30653.15 (15)	-30658.93 (20)	-30664.53 (26)	-30664.6 (25)

TAB. 4.4 – *Résultats donnés par l'algorithme sur le cas test G4 pour des différentes valeurs de τ_{target}, illustrant la performance médiane et le nombre d'essais munis avec succès (nombre entre parenthèses) pour chaque cas.*

Entre autres, pour les cas test G7, même si l'algorithme n'a pas trouvé l'optimum exact, les meilleures solutions sont très proches de la valeur de l'optimum: toutes les solutions sont entre 24.36 et 25.75. De même pour G9, l'optimum exact a été localisé une seule fois, mais toutes les solutions sont de très bonne qualité: entre 680.63 et 680.7. Ces résultats peuvent probablement être améliorés par un ajustement très fin des paramètres.

De même pour le cas de G10, l'optimum n'a pas été localisé, mais on peut considérer les résultats comme performants, vue la difficulté du problème. En effet, G10 est soumise à 6 contraintes, toutes actives au niveau de l'optimum global. Ce problème a été jugé très difficile par toutes les méthodes de prise en compte des contraintes qui ont essayé de le résoudre. Les meilleures solutions données dans la littérature sont 7054.31 enregistrée par la méthode de classement stochastique (chapitre 2, section 2.3.7) et 7060.22 enregistrée par la méthode de supériorité des individus faisables proposée par Deb [19] (chapitre 2, section 2.3.5). ASCHEA n'a pas réussi à dépasser ces performances, où il a donné 7095.15 comme meilleure solution, mais ce résultat peu être considéré comme satisfaisant en comparaison avec les autres méthodes qui échoué à résoudre ce problème (e.g. la méthode des pénalités dynamiques de Joines et Houk (chapitre 2, section 2.3.3 ou la méthode de supériorité des individus faisables proposée par Powell et Skolnick (chapitre 2, section 2.3.5)).

4.6 Conclusion

Dans ce chapitre, nous avons introduit **ASCHEA**, un algorithme évolutionnaire original pour l'optimisation sous contraintes. Le but d'ASCHEA est de maintenir la diversité de la population, en permettant aux solutions infaisables de coexister avec celles faisables, tout en favorisant légèrement ces dernières. Pour se faire, **ASCHEA** utilise un mécanisme de pénalisation adaptative, associé à une sélection ségrégationnelle qui favorise une proportion des individus faisables, afin d'assurer un degré de réalisabilité minimal de la population. Il utilise aussi, pour la reproduction des individus, une sélection spécifique, basée sur une stratégie de *séduction/sélection*, qui oblige, dans certains cas, les individus faisables de se reproduire avec ceux infaisables, pour permettre une meilleure exploration des alentours de la frontière du domaine réalisable. Par ailleurs, ASCHEA est aussi capable d'explorer l'intérieur du domaine faisable, grâce à sa grande capacité d'adaptation aux différents types de problèmes.

ASCHEA a donné de très bons résultats sur les 11 cas test que nous avons adopté pour nos expérimentations (voir Annexe A). Cependant, cette première étude est limitée à un cœfficient de pénalisation unique pour l'ensemble des contraintes du problème. Considérer l'ensemble des contraintes comme une seule n'est pas toujours idéal pour la procédure de recherche, spécialement si la difficulté de statisfaction varie d'une contrainte à une autre. Dans le prochain chapitre, une autre définition sera donnée à la fonction de pénalisation, utilisant un cœfficient de pénalité propre à chaque contrainte.

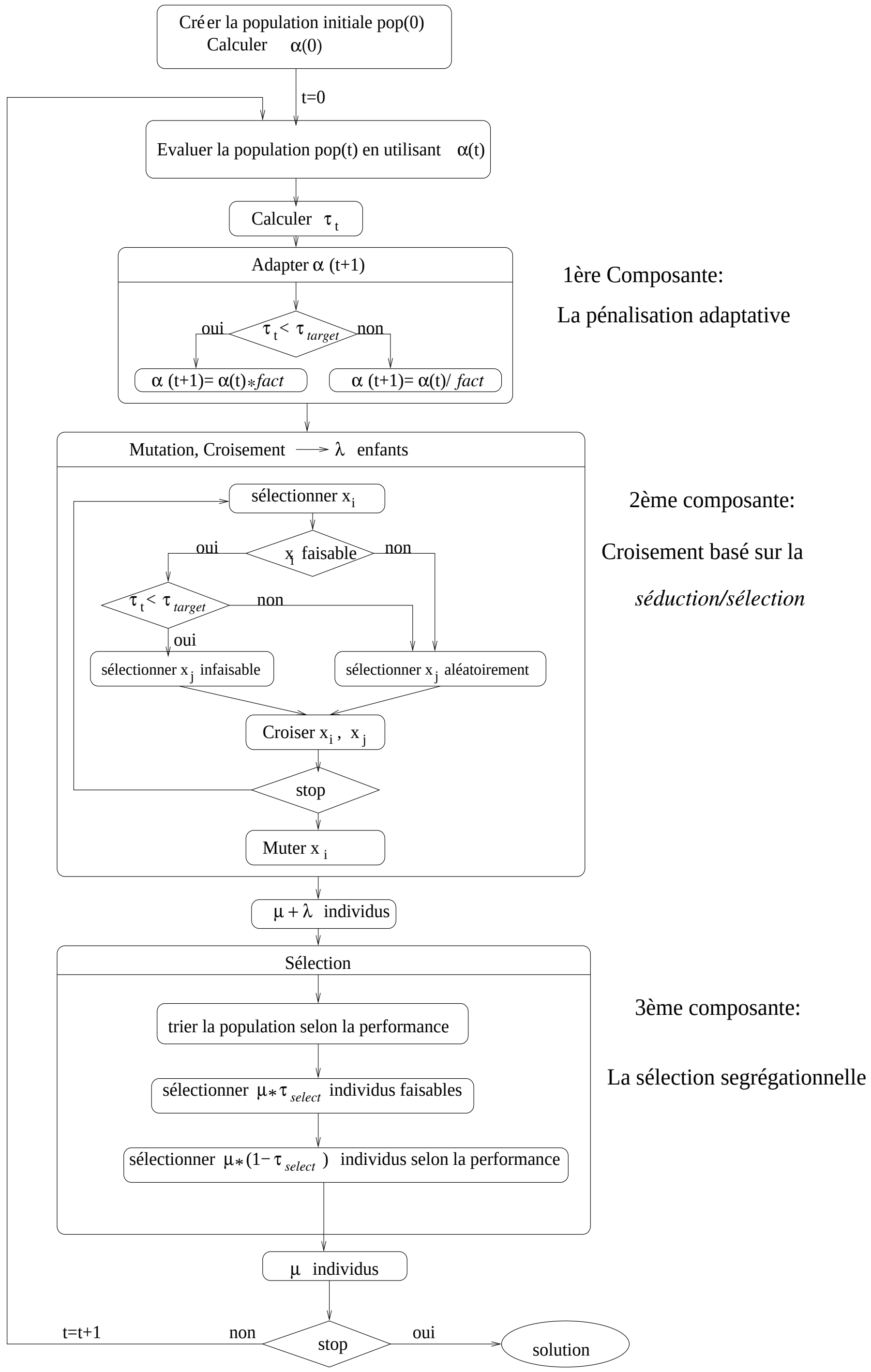


FIG. 4.16 – *Organigramme global d'ASCHEA*

Chapitre 5

ASCHEA: Optimisation de la fonction de Pénalisation

Résumé

Nous avons introduit dans le chapitre précédent un nouvel algorithme adaptatif pour l'optimisation sous contraintes: ASCHEA. Le but général d'ASCHEA est de maintenir la diversité de la population en offrant aux solutions infaisables une chance de survivre au long de l'évolution, mais une proportion de celles faisables est toujours favorisée. Pour ce faire, ASCHEA utilise un mécanisme de pénalisation adaptative, dont le cœfficient est ajusté selon le degré de faisabilité de la population. Cependant, l'utilisation d'un cœfficient de pénalisation commun à toutes les contraintes peut nuire au processus de recherche, surtout si le degré de difficulté varie d'une contrainte à une autre.

Dans ce chapitre, nous proposons une nouvelle version d'ASCHEA, utilisant une fonction de pénalisation optimisée. Cette dernière utilise un cœfficient de pénalité adaptatif spécifique à chaque contrainte, permettant d'une part d'orienter l'exploration selon la faisabilité de la population par rapport à chacune d'elles, et d'autre part de se concentrer sur les contraintes les plus difficile à satisfaire. Entre autres, une technique de nichage a été introduite afin de prendre en compte les fonctions multimodales. Une série d'expérimentations a été réalisée afin de décider de la meilleure implantation de la technique.

Une étude expérimentale sur un ensemble de cas test montre une amélioration de la qualité des résultats par rapport à ceux donnés par la première version d'ASCHEA, et prouve l'apport bénéfique de la nouvelle stratégie de pénalisation et le nichage.

5.1 Introduction

Dans le chapitre précédent, nous avons présenté un nouvel algorithme adaptatif pour l'optimisation sous contraintes: **ASCHEA**. Cet algorithme opte pour une meilleure exploration de l'espace de recherche grâce au maintient de la diversité de la population. Cette diversité est assurée grâce à une fonction de pénalisation adaptative, qui se base sur le degré de faisabilité de la population pour ajuster le cœfficient de pénalité. Ce dernier est augmenté ou diminué afin de favoriser, respectivement, les solutions faisables ou celles infaisables, en vue de garder les deux types d'individus au long de l'évolution. Un taux limite de faisabilité de la population est toujours respecté grâce à la sélection ségrégationnelle.

ASCHEA a montré des bonnes performances dans les expérimentations réalisées dans le chapitre précédent, au cours de la résolution de 11 problèmes de référence (cf. annexe A). Cependant, l'étude expérimentale a montré que ASCHEA a des légères difficultés à prendre en compte un grand nombre de contraintes simultanément, en particulier si le degré de difficulté diffère d'une contrainte à une autre.

Ce cas se présente, par exemple, avec la résolution de la fonction test G10, prouvée très difficile pour la majorité des méthodes de prise en compte des contraintes. Cette fonction est soumise à 6 contraintes, toutes actives au niveau de l'optimum, et ASCHEA, malgré la bonne qualité des résultats qu'il a donnés en comparaison avec ceux déjà présentés pour ce problème, n'a pas réussi à localiser avec précision la solution globale. Ceci est dû à la difficulté de s'approcher des lignes de saturation de certaines contraintes (frontière du domaine faisable), vu que la population se concentre plus sur une partie des bords de $\mathcal{F}$ que sur une autre.

Pour résoudre ce problème, et ainsi améliorer la performance d'ASCHEA, nous proposons dans ce chapitre une nouvelle définition de la fonction de pénalisation. Cette dernière utilise un cœfficient de pénalité propre à chaque contrainte, au lieu d'un seul pour l'ensemble des contraintes. Chaque cœfficient est ajusté selon le degré de faisabilité de la population par rapport à la contrainte correspondante, sans tenir compte des autres contraintes. Ceci permettra d'orienter le processus d'exploration de l'espace selon la disposition courante des individus, afin de forcer quelques uns vers les zones réalisables des contraintes les plus difficile à satisfaire. Autrement dit, si une contrainte n'est pas active au niveau de l'optimum, elle ne participera pas activement au processus de recherche, en particulier pendant les phases avancées de l'évolution. Par contre, si on est en présence d'une contrainte active à l'optimum, l'algorithme doit essayer de forcer une partie des individus vers les zones de la frontière correspondante (aussi bien du côté faisable que celui infaisable), afin d'assurer une exploration efficace à ses alentours.

Dans la deuxième partie de ce chapitre, nous introduisons une technique de nichage, afin de prendre en compte les fonctions multimodales, comme la fonction test G2. La méthode proposée est inspirée de la technique d'éclaircissement élitiste de Pétrowski [93]. Cependant, son principe a été légèrement modifié en vue de l'adapter aux spécificités d'ASCHEA.

L'efficacité de la procédure de nichage est très dépendante de la mesure de similarité entre les individus. La mesure la plus utilisée dans l'espace réel est la distance euclidienne. Dans ce cas, il faut définir le rayon qui détermine la taille de chaque niche. Pour éviter le passage par une recherche manuelle, qui peut être très lourde, de la meilleure valeur du rayon pour chaque problème, nous proposons, dans la dernière partie de ce travail, une stratégie pour l'adapter automatiquement au long de l'évolution.

Des études expérimentales avec et sans adaptation du rayon de nichage sont présentées, dans le but de comparer les deux stratégies et de montrer comment la dépendance de la qualité des résultats à la valeur du rayon de nichage diminue nettement quand ce dernier est adapté au cours de l'évolution.

5.2 La fonction de pénalisation

Nous rappelons que la fonction de pénalisation utilisée dans la version originale d'ASCHEA est définie comme suit:

$$penal(\overrightarrow{x}) = \alpha(t)(\sum_{j=1}^{q} g_j^+(\overrightarrow{x}) + \sum_{j=q+1}^{m} |h_j|(\overrightarrow{x})), \tag{5.1}$$

où x^+ est la partie positive de x.

Le choix de considérer un cœfficient de pénalisation général $\alpha(t)$ pour toutes les contraintes a été pris par mesure de simplicité. L'ajustement de $\alpha(t)$ au cours de l'évolution se fait selon le degré de faisabilité de la population τ_t par rapport à l'ensemble des contraintes:

$$\begin{array}{ll} \text{si } (\tau_t > \tau_{target}) & \alpha(t+1)(\overrightarrow{x}) = \alpha(t)/fact \\ \text{sinon} & \alpha(t+1)(\overrightarrow{x}) = \alpha(t) * fact \end{array} \tag{5.2}$$

où $fact > 1$ est un paramètre de l'algorithme défini préalablement.

En vue d'optimiser la performance d'ASCHEA, chaque contrainte j aura son propre cœfficient de pénalisation $\alpha_j(t)$, qui décidera du taux de participation de cette contrainte à la pénalisation des solutions irréalisables. La fonction de pénalisation 5.1 devient alors:

$$penal(\overrightarrow{x}) = \sum_{j=1}^{q} \alpha_j(t) g_j^+(\overrightarrow{x}) + \sum_{j=q+1}^{m} \alpha_j(t)|h_j|(\overrightarrow{x}), \tag{5.3}$$

Chaque cœfficient est ajusté selon la proportion $\tau_t(j)$ des individus respectant la contrainte j à la génération courante t. Le taux de faisabilité limite τ_{target} permettant de décider l'augmentation ou la diminution du cœfficient de pénalisation ainsi que le facteur de d'adaptation $fact$ sont communs à toutes les contraintes:

$$\begin{array}{ll} \text{si } (\tau_t(j) > \tau_{target}) & \alpha_j(t+1)(\overrightarrow{x}) = \alpha_j(t)/fact \\ \text{sinon} & \alpha_j(t+1)(\overrightarrow{x}) = \alpha_j(t) * fact \end{array} \tag{5.4}$$

Comme pour le cœfficient de pénalisation général, les pénalisations initiales $\alpha_j(0)$ sont calculées à partir de la performance de la population à la première génération, ainsi que les taux de violation des contraintes:

$$\begin{array}{ll} \text{si } \sum_{i=1}^{n} v_j(\overrightarrow{x_i}) = 0 & \alpha_j(0) = 1, \\ \text{sinon} & \alpha_j(0) = \left(\frac{\sum_{i=1}^{n} |f(\overrightarrow{x_i})|}{\sum_{i=1}^{n} |v_j(\overrightarrow{x_i})|}\right) * 100^{1}, \end{array} \tag{5.5}$$

où $v_j(\overrightarrow{x_i})$ est le taux de violation de la j^{eme} contrainte correspondant à l'individu $\overrightarrow{x_i}$:

$$v_j(\overrightarrow{x_i}) = \begin{cases} max(0, g_j(\overrightarrow{x_i})), & si\, 1 \leqslant j \leqslant q \\ |h_j(\overrightarrow{x_i})|, & si\, q+1 \leqslant j \leqslant m \end{cases}$$

1. Dans le chapitre précédent, le quotient a été multiplié par 1000. Ici, il est multiplié par 100, parce que le dénominateur a une valeur plus petite, puisqu'il n'est plus la somme des violations de l'ensemble des contraintes, mais la somme des violations de la contrainte j. Ceci devrait éviter d'initialiser les cœfficients de pénalité avec de très grandes valeurs.

Le fait de définir un cœfficient de pénalisation pour chaque contrainte permet d'ajuster l'influence relative de chacune. En effet, si une contrainte n'est pas active, son cœfficient prendra des petites valeurs et elle sera presque négligée au cours du processus de recherche. Par contre, si une des contraintes actives est difficile à satisfaire, le taux de pénalisation correspondant prendra des grandes valeurs, afin que sa satisfaction devient une priorité pour le processus de recherche.

Entre autres, la diminution de la valeur du cœfficient de pénalisation d'une contrainte active permet de faire apparaître des individus infaisables à ses alentours. Le croisement de ces points avec d'autres points de l'autre côté de la frontière stimule l'exploration dans cette région. Ainsi, toutes les contraintes actives du problème bénéficient d'une exploration appropriée au cours de l'évolution.

5.3 Étude Expérimentale

Dans le chapitre précédent, nous avons étudié en détails le comportement de l'algorithme dans les deux cas de localisation possible de l'optimum par rapport à l'espace faisable: sur la frontière de l'espace réalisable et à l'intérieur du domaine réalisable. Une étude similaire sera présentée dans cette section afin de comprendre les apports des nouvelles modifications apportées à l'algorithme. Pour assurer la visibilité et pouvoir comparer les nouveaux résultats avec ceux donnés par ASCHEA dans sa première version, on gardera les mêmes fonctions que celles étudiées au chapitre précédent, c'est à dire les fonctions G6 et G8. On rappelle que G6 a deux contraintes inégalité et son optimum se trouve sur la frontière du domaine faisable, alors que l'optimum de G8 se trouve à l'intérieur du domaine réalisable et elle est soumise aussi à deux contraintes inégalité. Les définitions de ces fonctions sont donnés dans le paragraphe 4.3 du chapitre 4, et leurs surfaces sont illustrées dans la figure 4.2.

Un autre cas test est ajouté à cette étude expérimentale, qui lui aussi a une solution globale localisée sur la frontière du domaine faisable, mais diffère du premier cas par la proportion de contraintes actives au niveau de l'optimum. Contrairement au cas de G6, où toutes les contraintes sont actives, la fonction G4 du troisième cas d'étude est soumise à 6 contraintes dont seulement 2 sont actives. Le problème correspondant est défini comme suit:

$$\textbf{Minimiser } G4(\overrightarrow{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141,$$

sous les contraintes suivantes:

C1: $85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 \geqslant 0$
C2: $0 \leqslant 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 \leqslant 92$
C3: $80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 \geqslant 90$
C4: $80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 \leqslant 110$
C5: $9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 \geqslant 20$
C6: $9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 \leqslant 25,$

et les bornes:

$78 \leqslant x_1 \leqslant 102,\ 33 \leqslant x_2 \leqslant 45,\ 27 \leqslant x_i \leqslant 45$ pour $i = 3, 4, 5.$

La solution optimale est $\overrightarrow{x}^* = (78, 33, 29.995256, 45, 36.775812)$, où $G4(\overrightarrow{x}^*) = -30665.5$. Les deux contraintes saturées au niveau de l'optimum sont C1 et C6.

Ce problème est introduit dans la partie des expérimentations détaillées en vue d'étudier l'apport de l'utilisation des cœfficients de pénalisation individuels dans la présence de plusieurs contraintes dont une proportion n'est pas active.

5.3.1 Conditions expérimentales

Pour permettre les comparaisons entre les résultats donnés par les deux versions d'ASCHEA, nous avons gardé les mêmes conditions expérimentales que celles posées pour la première série d'expériences. Ainsi, ASCHEA utilise toujours un ES(100+300) avec la sélection ségrégationnelle décrite dans le paragraphe 4.2.3 du chapitre 4. Les paramètres spécifiques à ASCHEA gardent toujours les mêmes valeurs, à part celle de τ_{target} qui a été réduite légèrement, étant donné que la faisabilité est calculée par rapport à chaque contrainte et non pas l'ensemble des contraintes. Ainsi le taux de faisables pour une contrainte j augmente plus rapidement que le taux de faisables satisfaisant toutes les contraintes. Une valeur moins élevée de τ_{target} permet de favoriser un peu plus d'infaisables pour la contrainte j, permettant une meilleure exploration de la frontière correspondante.

- $\tau_{select} = 0.3$
- $\tau_{target} = 0.5$
- $fact = 1.1$

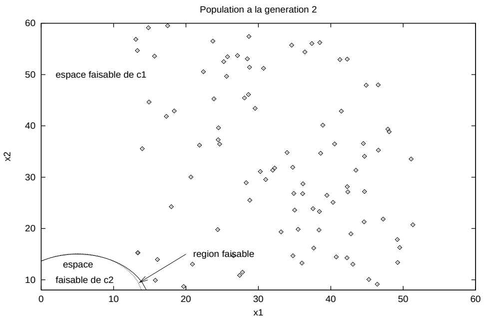
5.3.2 Cas 1: Solution sur la frontière du domaine faisable

Dans le cas de la fonction G6, comme avec sa première version, ASCHEA réussit à trouver facilement et rapidement (en moyenne après 150 générations) l'optimum global. Cependant, l'évolution de la population est légèrement différente de celle décrite dans le chapitre précédent. Les paragraphes suivants présentent les différentes phases de l'évolution de la population, ainsi que les changements remarqués par rapport à l'étude précédente et leurs origines.

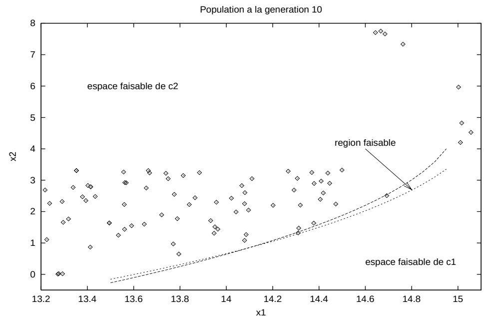
D'une façon similaire qu'avec la version précédente d'ASCHEA, au début de l'évolution, l'augmentation de la valeur de la pénalisation accompagnée de la stratégie de séduction/sélection du croisement exerce une pression sur les individus afin de les amener progressivement vers l'espace réalisable. L'étude expérimentale du cas test G6 réalisée dans le chapitre précédent a montré comment la population se dirige rapidement vers l'espace $\mathcal{F}$ et s'approche de la localisation de l'optimum. C'est le même phénomène qui se répète avec cette nouvelle version, mais moins rapidement puisque le processus d'évolution a été changé. La modification du mécanisme de pénalisation a modifié le schéma de progression de la population dans l'espace.

En fait, chaque cœfficient de pénalisation exerce de son côté une pression sur les individus afin de les pousser vers l'espace réalisable de la contrainte correspondante. De ce fait, si pour un individu non faisable donné la pression du premier cœfficient de pénalisation est supérieure à celle du deuxième cœfficient, il tentera d'abord de satisfaire la première contrainte et ensuite la deuxième. Afin d'exposer clairement les différentes étapes d'évolution de la population, un exemple d'essai de résolution de la fonction G6 est présentée ci-dessous en détail.

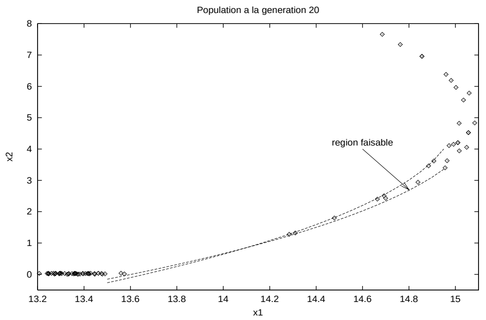
Au cours de l'initialisation de la population, vues les bornes des variables, la probabilité de générer des individus dans l'espace faisable de C1 est beaucoup plus élevée que celle de C2. En fait, cette dernière est presque nulle. La courbe 5.1(a) montre qu'à la deuxième génération, toute la population est dans l'espace faisable de C1 ($\tau_2(1) = 1$), mais aucun individu n'est dans celui de C2 ($\tau_2(2) = 0$). Le cœfficient α_2 est alors initialisé avec une très grande valeur ($\simeq 40914.2$), vu le grand taux de violation de C2 (équation 5.5). Avec une telle pénalisation, la satisfaction de C2 devient une priorité pour l'algorithme. Pendant les générations suivantes, alors que α_2 augmente pour forcer les individus vers la zone faisable de C2, α_1 diminue, tant que $\tau_t(1)$ est supérieur à



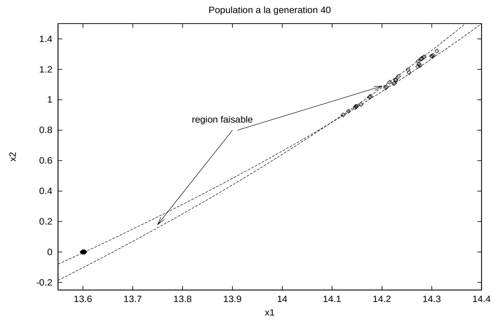
(a) $\tau_2(1) = 1$, $\tau_2(2) = 0$



(b) $\tau_{10}(1) = 0.13$, $\tau_{10}(2) = 1$



(c) $\tau_{20}(1) = 0.3$, $\tau_{20}(2) = 1$



(d) $\tau_{40}(1) = 0.3$, $\tau_{40}(2) = 1$

FIG. 5.1 – *L'évolution de la population entre les générations 10 et 40, pendant un essai de résolution du cas test G6.*

τ_{target}. Ceci va entraîner la population, à la génération 10, de l'autre côté de la frontière, dans la région faisable de C2 (figure 5.1(b)). L'évolution des valeurs de α_1 et α_2 et les taux de faisabilité correspondant pendant les 10 premières générations est donnée dans le tableau 5.1. Ce n'est qu'à partir de la génération 20 qu'un certain équilibre entre les deux pénalisations commence à s'établir, et la pression sur les infaisables devient partagée (figure 5.1(c)). Les individus sont alors attirés vers l'espace faisable des deux contraintes (figure 5.1(d)). La figure 5.3 montre la grande différence des poids des deux pénalisations pendant les premières générations, qui a ensuite disparu une fois qu'une proportion supérieure ou égale à τ_{select} de la population est faisable.

Le phénomène qui se manifeste ensuite et à plusieurs reprises pendant l'évolution est le regroupement/dispersion des individus aux alentours des frontières (des côtés réalisables et irréalisables de chaque contrainte), tout en s'approchant de l'optimum. Les individus sont attirés en même temps par le meilleur faisable qui les guide vers la localisation du l'optimum, et les meilleurs non faisables qui les attirent à l'extérieur de l'espace réalisable. Les points infaisables réapparaissent dans la population du côte de de la région infaisable de C1 et C2, et ceci à la suite ou simultanément que la diminution des cœfficients de pénalisation correspondants, et disparaissent aussi tôt que ces derniers prennent des grandes valeurs.

Dans le cas de G6, ces différentes étapes sont résumées par la figure 5.2. La première courbe (courbe (a)) présente la population à la génération 100. On y voit un point isolé à gauche, c'est la

Gen (t)	$\tau_t(1)$	$\alpha_1(t+1)$	$\tau_t(2)$	$\alpha_2(t+1)$	τ_t
1	1	1	0	40914.2	0
2	1	0.909	0	45005.6	0
3	0.99	0.826	0.01	49506.2	0
4	0.97	0.751	0.03	54456.8	0
5	0.92	0.683	0.08	59902.5	0
6	0.64	0.62	0.37	65892.7	0
7	0.06	0.683	1	59902.5	0.01
8	0.08	0.751	1	54456.8	0.06
9	0.1	0.826	1	49506.2	0.08
10	0.13	0.909	1	45005.6	0.1

TAB. 5.1 – *Évolution des cœfficients de pénalisation α_1 et α_2 ainsi que les taux de faisabilité de C1 ($\tau_t(1)$), de C2 ($\tau_t(2)$) et général (de l'ensemble des contraintes) (τ_t) pendant les 10 premières générations d'un test de résolution de G6.*

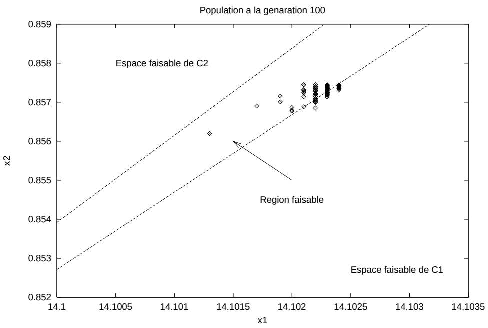
meilleure solution faisable courante, qui va attirer les autres vers sa zone. La figure 5.2(b) montre le déplacement effectué en 10 générations de toute la population vers la gauche, en direction de l'optimum global. A ce moment de l'évolution, le cœfficient de pénalité de la deuxième contrainte α_2 commence à décroître, ce qui correspond à l'apparition des points en dehors de la borne inférieure de $\mathcal{F}$. Cette baisse continue avec les deux cœfficients de pénalité α_1 et α_2, pour donner, à la génération 120, une population répartie sur les régions faisables et infaisables des deux côtés (figure 5.2, courbe (c)). Le croisement de ces solutions entre eux donne naissance à des points encore plus proches de l'optimum global. Entre temps, les cœfficients de pénalité augmentent et ASCHEA exerce à nouveau une pression sur les individus pour les ramener dans l'espace réalisable, ce qui donne, à la génération 140, une population plus regroupée (figure 5.2(c)). Pour cet essai, la solution globale est trouvée à la génération 201.

Étant donné que les deux contraintes sont actives au niveau de l'optimum, les individus restent fragiles par rapport au taux de pénalisation de chaque contrainte. Ainsi, l'augmentation/diminution des cœfficients est accompagnée par une apparition/disparition des solutions irréalisables tout au long de l'évolution. Cette sensibilité au α_j est due à l'emplacement de la population qui se regroupe aux alentours de la frontière, proche de la localisation de l'optimum global. La figure 5.3 montre bien l'allure de l'évolution des cœfficients de pénalisation α_1 et α_2. Pendant les premières générations, les deux cœfficients prennent des grandes valeurs (vus taux élevés de violations), afin de forcer la population vers l'espace faisable. Ensuite, leurs valeurs commencent à osciller dans un intervalle de taille très variable tant que l'optimum n'est pas trouvé, et dans un intervalle ayant une taille plus ou moins stable dans des phases plus avancées de l'évolution (figure 5.3). Cet intervalle est d'autant plus petit que le degré de précision des solutions est élevé (distance réelle à la frontière plus petite).

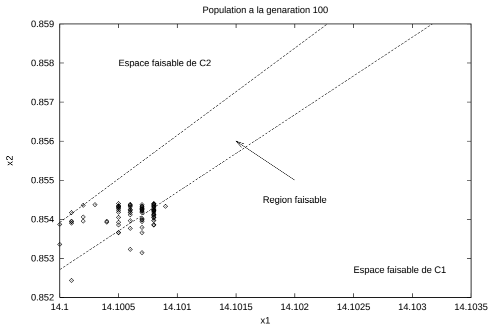
5.3.3 Cas 2: Solution à l'intérieur du domaine faisable

Dans le cas de G8, l'optimum se trouve à l'intérieur du domaine faisable, donc les contraintes ne sont pas actives à ses alentours. Le rôle principale de la pénalisation est donc de ramener une proportion des individus dans l'espace réalisable, et d'assurer la diversité de la population aussi long temps que possible pour une meilleure exploration de la région $\mathcal{F}$.

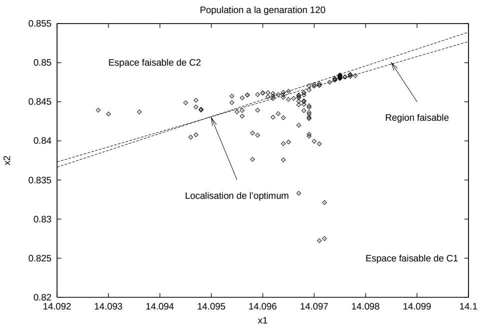
ASCHEA n'a eu aucune difficulté à localiser rapidement (en moyenne après 39 générations)



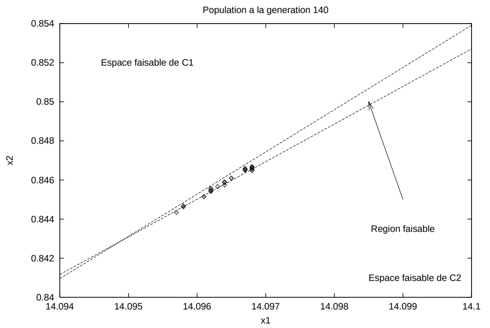
(a): Population à la génération 100
$\tau_{100}(1) = 1,\ \tau_{100}(2) = 0.78$
$\alpha_1(100) = 304.48,\ \alpha_2(100) = 421.72$



(b): Population à la génération 110
$\tau_{110}(1) = 0.97,\ \tau_{110}(2) = 0.57$
$\alpha_1(110) = 117.39,\ \alpha_2(110) = 162.59$



(c) Population à la génération 120
$\tau_{120}(1) = 0.85,\ \tau_{120}(2) = 0.45$
$\alpha_1(120) = 45.25,\ \alpha_2(120) = 11.05$



(d) Population à la génération 140
$\tau_{140}(1) = 0.39,\ \tau_{140}(2) = 0.79$
$\alpha_1(140) = 9.84,\ \alpha_2(140) = 91.77$

FIG. 5.2 – *L'évolution de la population entre les générations 100 et 140, pendant un essai de résolution du cas test G6. Pour les deux premières courbes, l'intervalle contenant la variable x1 est de* $[14.1, 14.1035]$*, alors qu'il est de* $[14.094, 14.1]$ *pour les deux autres, où la population s'est déplacée vers la gauche pour s'approcher de la zone de l'optimum.(N.B. L'alignement des points dans (a) et (b) est l'effet du zoom dans un intervalle très petit.)*

l'optimum global, pour les 31 essais. Au début de l'évolution, il tente d'abord de forcer le maximum d'individus vers l'espace faisable. Comme pour le cas de G6, on remarque un certain déséquilibre entre les pressions que subissent les individus de côté du cœfficient de C1 ($x_1^2 - x_2 + 1 \leqslant 0$) et du cœfficient de C2 ($1 - x_1 + (x_2 - 4)^2 \leqslant 0$). La courbe 5.4 montre bien ce déséquilibre, qui se manifeste dès la première génération, où la différence entre $\alpha_1(0)$ et $\alpha_2(0)$ est assez grande. En effet, $\alpha_1(0)$ est initialisé avec des petites valeurs ($\simeq 1$), car le taux de violation de C1 est très faible (équation 5.5), alors que $\alpha_2(0)$ prend des valeurs plus grandes ($\simeq 10$), car le taux de violation de C2 est plus important.

Contrairement au cas de G6, ce déséquilibre n'est pas dû en premier lieu aux taux de faisabilité de la population initiale par rapport à chaque contrainte, mais aux taux de violation des individus infaisables, dont les membres sont dispersés dans l'espace de recherche, parfois très loin de la frontière de la région faisable. Un exemple est illustrée dans la figure 5.5, où la courbe (a) (points

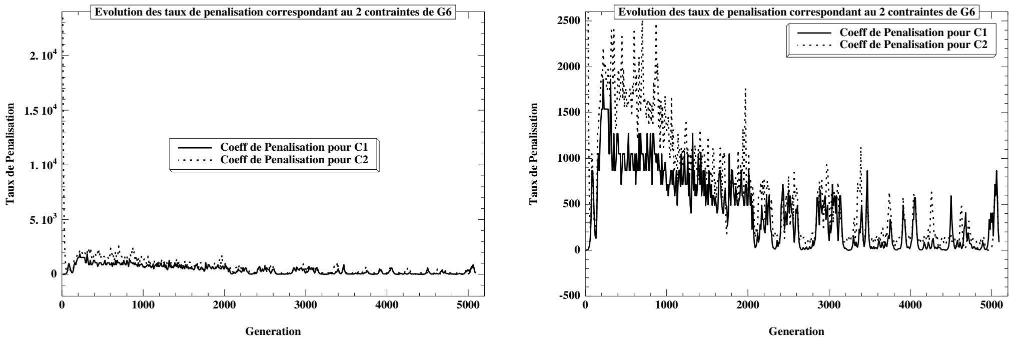


FIG. 5.3 – *Courbes médianes d'évolution des taux de pénalisation $\alpha_1(t)$ et $\alpha_2(t)$ correspondant aux deux contraintes de G6. La courbe de droite est un zoom de la première courbe montrant d'une façon plus visible le phénomène d'oscillation des cœfficients de pénalité.*

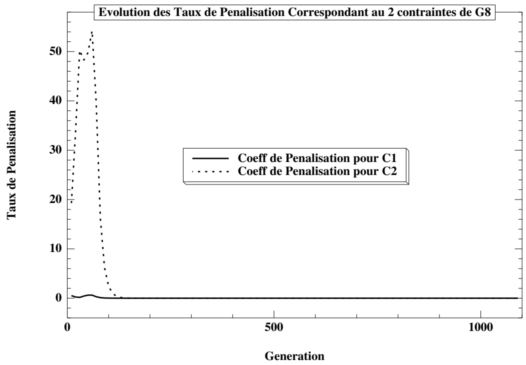
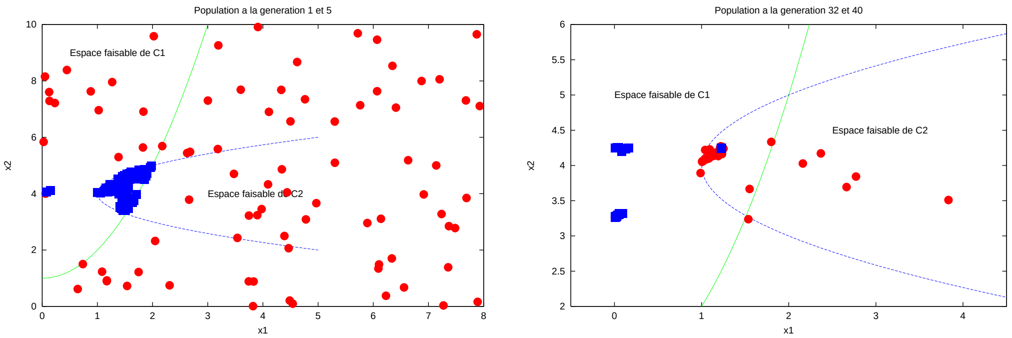


FIG. 5.4 – *Courbes médianes d'évolution des taux de pénalisation $\alpha_1(t)$ et $\alpha_2(t)$ correspondant aux deux contraintes de G8.*

cercles) montre qu'une proportion des individus générés sont dans les espaces faisables de chaque contrainte, mais contrairement à C1, la majorité de ceux qui violent C2 sont très loin de la frontière correspondante.

Une fois une grande proportion de la population est faisable (figure 5.5(a), points carrés), α_2 commence à décroître. Quelques points peuvent encore basculer de l'autre côte de la frontière, parce qu'ils y sont proches (phénomène de disparition/réapparition des non faisables). Les infaisables apparaissent soit dans la région de C1 (figure 5.5(b), points carrés), soit dans la région de C2 (figure 5.5(b)), points cercles) selon les valeurs courantes des cœfficients de pénalité α_1 et α_2. Mais, une fois l'optimum est localisé, quelques générations plus tard, toute la population converge à l'intérieur du domaine faisable, attirée par l'optimum. Dans l'exemple présenté dans la figure 5.5, l'optimum est trouvé à la génération 52 et la population a convergé dans $\mathcal{F}$ à la génération 90.



(a) Population à la génération 1 et 5
$\tau_1(1) = 0.31$, $\tau_1(2) = 0.25$
$\alpha_1(1) = 1$, $\alpha_2(1) = 10.05$
$\tau_5(1) = 0.99$, $\tau_5(2) = 0.98$
$\alpha_1(5) = 0.9$, $\alpha_2(5) = 7.55$

(b) Population à la génération 32 et 40
$\tau_{32}(1) = 0.94$, $\tau_{32}(2) = 0.63$
$\alpha_1(32) = 0.069$, $\alpha_2(32) = 46.21$
$\tau_{40}(1) = 0.1$, $\tau_{40}(2) = 0.74$
$\alpha_1(40) = 0.032$, $\alpha_2(40) = 31.56$

FIG. 5.5 – *Population de G8 à la génération 1-5 (a) et 32-40 (b).*

5.3.4 Cas 3: Quelques contraintes ne sont pas actives au niveau de l'optimum

Le cas où seulement une proportion des contraintes du problème sont actives au niveau de l'optimum global est similaire au premier cas (cas de G6) car la solution se trouve sur la frontière du domaine faisable. En fait, nous avons choisi d'étudier ce cas afin d'expliquer l'apport bénéfique de l'utilisation d'un cœfficient de pénalisation propre à chaque contrainte dans le cas où quelques unes ne sont pas actives à l'optimum.

La fonction G4 est soumise à 6 contraintes dont seulement 2 sont actives à l'optimum. La figure 5.6 montre l'évolution des différents cœfficients de pénalité correspondant aux 6 contraintes dans l'échelle logarithmique. On voit bien que les 4 cœfficients α_2, α_3, α_4 et α_5 correspondant aux 4 contraintes non actives diminuent d'une façon continue et régulière au long de l'évolution, et n'ont donc aucun effet sur le processus de recherche. ASCHEA se concentre ainsi sur les contraintes restantes, dont les cœfficients (α_1 et α_6) suivent le même phénomène que celui décrit pour le cas de la résolution de la fonction G6. Les individus de la population ne subissent de pression que des cœfficients de pénalité des deux contraintes actives.

α_1 et α_6 prennent au début des valeurs plus ou moins grandes afin de forcer les individus vers l'espace faisable. Ensuite, c'est la procédure d'augmentation/diminution qui prend place pour une meilleure exploration de la frontière.

On remarque aussi que la taille des intervalles d'oscillation des cœfficients de pénalisation diminue au fur et à mesure des générations, reflétant le rapprochement progressive de la population des frontières de C1 et C6. Ceci prouve que le degré de précision de l'exploration des frontières augmente au long de l'évolution. Le fait que l'optimum a été localisé assez rapidement (génération 332) n'empêche pas la continuation de l'exploration des frontières jusqu'à la fin de l'évolution, et avec une précision croissante.

Cependant, pour ce cas test, l'amélioration de la meilleure solution peut se stabiliser pendant

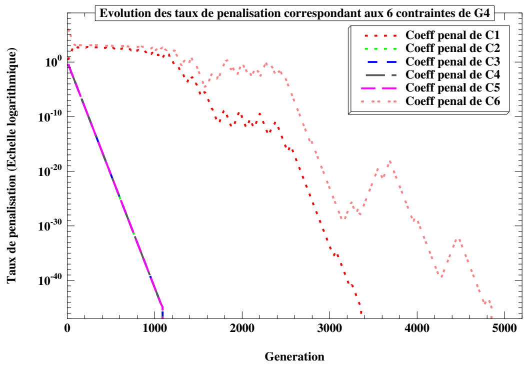


FIG. 5.6 – *Courbes d'évolution des médianes (sur 31 essais) des différents cœfficients de pénalisation correspondant aux 6 contraintes du cas test G4. Toutes les courbes sont confondues sauf celles de C1 et C6.*

quelques dizaines de générations à cause de la complexité de la surface de la frontière. Les courbes (a) et (b) de la figure 5.7 montrent que le meilleur faisable et le meilleur global sont confondus entre les générations 100 et 300. A ce moment, le degré d'homogénéité de la population est assez élevé et le processus de recherche est ralentit. Si la pénalisation était statique ou croissante, l'algorithme a des très faibles chances d'éviter la convergence prématurée. Cependant, grâce au mécanisme adaptatif d'ASCHEA, le cœfficient de pénalisation α_1 a enregistré une grande diminution entre les générations 275 et 388, permettant de générer des infaisables dans des régions non encore visitées. L'algorithme profite alors de leur présence pour croiser quelques uns avec les faisables, et stimuler le processus d'exploration. Le résultat est le passage de la meilleure performance de -30621.6 à la génération 300 à -30665.5 à la génération 332, correspondant à l'optimum global (courbe (c) et (d) de la figure 5.7. Entres autres, le meilleur de la population n'est plus confondu avec le meilleur faisable (c'est un nouveau point appartenant au l'espace irréalisable du côté de C1), et la diversité de la population est retrouvée.

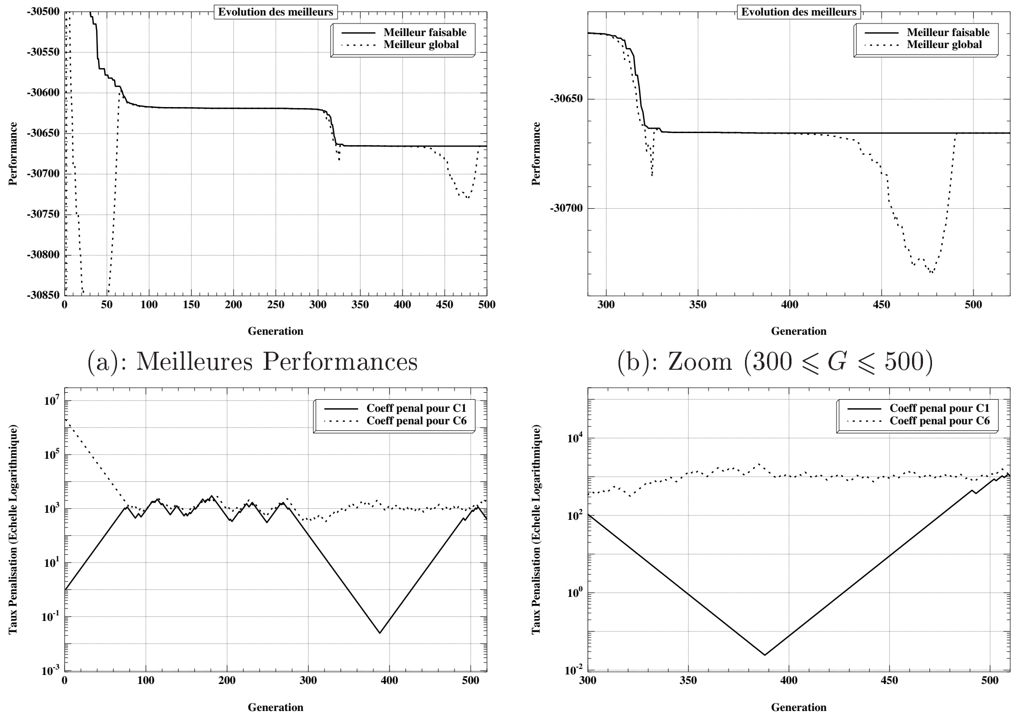
Pendant ce temps, le cœfficient de pénalité de C6 varie avec des taux faibles, permettant à l'algorithme de se concentrer sur la frontière de C1.

Cet exemple montre l'intérêt de l'utilisation d'un cœfficient de pénalisation propre à chaque contrainte dans les cas où:

- quelques contraintes ne sont pas actives au niveau de l'optimum,
- une fraction des contraintes actives forment une surface de frontière complexe plus difficile à explorer que celles des autres contraintes actives.

5.3.5 Résultats pour l'ensemble des cas test

Après avoir étudié dans la section précédente le comportement de l'algorithme dans la résolution de 3 fonctions de référence de différents types, on étend dans cette section l'étude à l'ensemble des 11 cas test proposées dans [87] et [73] (voir Annexe A). Pour chaque cas, 31 essais ont été effectués dans les mêmes conditions expérimentales décrites dans la section 5.3.1. Comme dans les expériences du chapitre précédent, pour les problèmes avec des contraintes d'égalité (G3, G5 et G11), les équations ont été remplacées par des inéquations de la forme: $h_j(\overrightarrow{x}) \leqslant \epsilon$ avec ϵ un nombre



(a): Meilleures Performances (b): Zoom ($300 \leqslant G \leqslant 500$)

(c): cœfficients de pénalisation α_1 et α_6 (d): Zoom ($300 \leqslant G \leqslant 500$)

FIG. 5.7 – *Courbes d'évolution des meilleures performances et des cœfficients de pénalisation durant 500 générations d'un essai de résolution du cas test G4. Les courbes (a) et (b) correspondent aux performances du meilleur faisable et du meilleur global pour ce test. Les courbes (c) et (d) correspondent à l'évolution des valeurs des cœfficients de pénalisation α_1 et α_6 en échelle logarithmique.*

positif très petit (fixé à 10^{-16} pour la suite). On note qu'avec cette transformation le domaine de faisabilité de la contrainte correspondante est largement plus grand que le domaine de faisabilité réel, ce qui peut causer l'échec de la technique si le nouvel espace contient des optima locaux. Dans le chapitre 6, nous proposons des stratégies de prise en compte des contraintes d'égalité plus adéquates et qui peuvent être généralisées à tout type de problème.

Les résultats des différentes expériences sont résumés dans le tableau 5.2. Toutes les solutions trouvées sont faisables. L'optimum exact a été trouvé pour 7 problèmes parmi les 11 étudiés.

L'amélioration de la qualité des résultats est prouvée pour la majorité des tests. Par exemple, pour G1, l'algorithme n'a pas réussit à trouver l'optimum exact seulement pendant deux essais sur 31, à comparer avec 7 échecs sur 31 essais avec la première version. De même pour le cas de G4, ASCHEA a mené tous les essais avec succès alors qu'il y a eu 16 échecs pour $\tau_{target} = 0.5$ avec la définition précédente de la fonction de pénalisation.

L'amélioration la plus significative est celle du cas G10, où aucune méthode de prise en compte des contraintes (chapitre 2) n'a réussi à s'approcher de très près de l'optimum global, exceptée

Fonction	valeur de l'Opt.	Opt. trouvé			Après 5000 gen		
		Nb. succès	Meilleur (nb gen)	Moy. (nb gen)	Meilleur	Médiane	Moy.
G1	-15	29	262	330	**-15**	-15	-14.84
G2	0.803619	–	–	–	0.785	0.56	0.59
G3*	1.0	–	–	–	0.999998	0.999979	0.99989
G4	-30665.5	31	95	217	**-30665.5**	-306665.5	-30665.5
G5*	5126.4981	–	–	–	5126.5	5128	5141.65
G6	-6961.81	31	92	150	**-6961.81**	-6961.81	-6961.81
G7	24.3062	–	–	–	24.3323	24.6162	24.6636
G8	0.095825	31	16	39	**0.095825**	0.095825	0.095825
G9	680.630	1	4567	4567	**680.630**	680.635	680.641
G10	7049.33	–	–	–	7061.13	7193.11	7497.434
G11*	0.75	31	27	37	**0.75**	0.75	0.75

TAB. 5.2 – *Résumé des résultats pour les 11 cas test, donnant pour chacun la meilleure, la médiane et la moyenne des meilleures solutions trouvées pour les 31 essais, après 5000 générations. Quand l'optimum est trouvé au moins une fois, les 3 premières colonnes donnent le nombre de succès, ainsi que le nombre de générations minimal et moyen mis par l'algorithme pour trouver l'optimum pour les essais munis avec succès. Les fonctions G2 et G3 ont été testées avec, respectivement, 20 et 10 variables*

la méthode du *"Stochastic ranking"*[106], qui a donné 7054.31 comme meilleure solution, mais la médiane est à 7559.19.

Ce problème nécessite de l'algorithme des grandes capacités d'exploration. Celle de l'opérateur de mutation logarithmique proposée dans le chapitre 3 a réussi à accomplir une très bonne performance en donnant 7259.64 comme valeur médiane, et 7066.56 comme meilleure valeur. Toutefois, cet opérateur effectue une exploration globale de l'espace de recherche, sans une concentration spéciale sur la frontière. Le fait d'explorer les alentours des frontières de faisabilité de chaque contraintes grâce à l'utilisation d'un coefficient de pénalisation propre pour chacune, a permis à ASCHEA d'améliorer les résultats en donnant 7193.11 comme valeur médiane et 7061.13 comme meilleure solution. Nous signalons que la valeur 7193.11 et la meilleure solution médiane jamais enregistrée pour ce problème (voir tableau 2.4 du chapitre 2).

On remarque aussi une légère amélioration au niveau de la qualité des résultats de G9, où le taux d'erreur est passé de $1.03.10^{-3}\%$ avec la version précédente à $0.73.10^{-3}\%$ avec la nouvelle version.

Pour le cas de G7, malgré que ASCHEA a réussi de s'approcher un peu plus de la solution globale (où il a donné 24.3323 comme meilleure solution), on remarque une légère baisse au niveau de la qualité de la solution médiane qui a passé de 24.505 à 24.616. Cependant, étant donné que l'optimum de G7 est sur la frontière de $\mathcal{F}$, une valeur plus petite de τ_{target}, égale à 0.4, a permis de donner 24.42 comme valeur médiane et 24.31 comme meilleure solution. Ce résultat est parmi les meilleurs enregistrés pour ce cas test.

Par ailleurs, malgré l'optimisation de sa performance, ASCHEA n'a pas réussi à améliorer la qualité des résultats pour le cas test G2. En effet, ce dernier a une multitude d'optima locaux, imposant l'utilisation d'une procédure de nichage. Dans la section suivante, nous ajoutons à ASCHEA une procédure de nichage afin qu'il puisse prendre en compte telles fonctions.

5.4 Introduction d'une procédure de nichage

Pour éviter les pièges d'attraction causés par les nombreux optima locaux pouvant entourer l'optimum global (cas de la fonction G2), une procédure de nichage a été ajoutée à l'algorithme afin d'assurer une meilleure exploration de l'espace de recherche. La procédure choisie est l'éclaircissement élitiste introduit par Pétrowski [93], dont le principe est de sélectionner, parmi l'ensemble des individus de la population, des groupes de "gagnants" autorisés à survivre, choisis selon leurs performances et leurs dispositions dans l'espace de recherche (voir chapitre 1, section 1.6.4).

Cependant, cette procédure affecte une performance nulle à tous les "perdants" (individus non sélectionnés pour survivre), ce qui ne s'adapte pas avec le mode de sélection $(\mu + \lambda)$ des stratégies d'évolution. En effet, si le nombre de "gagnants" est inférieur à μ, l'algorithme complète la population avec des individus considérés comme "perdants". Ayant une performance nulle, les individus sélectionnés sont choisi d'une manière complètement aléatoire, ce qui contredit le principe de la sélection déterministe.

De plus, l'éclaircissement élitiste ne s'adapte pas non plus avec la sélection ségrégationnelle d'ASCHEA. En effet, même si le nombre des "gagnants" est supérieur à μ mais il n'y a pas $\mu * \tau_{select}$ faisables parmi eux, on tombe dans le même problème. ASCHEA sera obligé de compléter les individus faisables parmi les "perdants", mais d'une façon aléatoire, puisque leurs performances sont nulles.

Pour ces différentes raisons, nous avons modifié la procédure d'éclaircissement en vue de l'adapter aux particularités d'ASCHEA. Pour ce faire, nous avons choisi de créer deux classes: la classe des meneurs (notée *classe_meneurs* pour la suite) et la classe des suiveurs (notée *classe_suiveurs* pour la suite). Les critères de décision des meneurs sont les mêmes que la procédure d'éclaircissement d'origine. Tout individu jugé meneur est classé dans la liste des meneurs. De même, les individus suiveurs sont classés ensemble dans la *classe_suiveurs* au lieu de leur affecter des performances nulles. Les μ individus de la génération suivante sont sélectionnés d'une manière déterministe dans la classe des meneurs. S'il n'y a pas assez de meneurs, les individus manquants sont complétés parmi les suiveurs ayant les meilleures performances.

Le nouvel algorithme de la procédure d'éclaircissement est comme suit:

Debut
trier la population (par ordre décroissant de la performance)
Pour i=1 **jusqu'à** i= $(\mu - 1)$
  Si $\overrightarrow{x_i} \notin$ *classe_suiveurs*
    ajouter $\overrightarrow{x_i}$ à *classe_meneurs*
    nbgagants=1
    Pour j=i+1 **jusqu'à** j= $\mu - 1$
    Si $\overrightarrow{x_j} \notin$ *classe_suiveurs* et $distance(\overrightarrow{x_i}, \overrightarrow{x_j}) < r_e$
      Si nbmeneurs <capacité_niche
        ajouter $\overrightarrow{x_j}$ à *classe_meneurs*
        nbmeneurs ← nbmeneurs+1
    sinon
      ajouter $\overrightarrow{x_j}$ à *classe_suiveurs*
Fin

avec:

- $distance(\overrightarrow{x_i}, \overrightarrow{x_j})$: retourne la distance euclidienne entre les points $\overrightarrow{x_i}$ et $\overrightarrow{x_j}$,

- r_e: le rayon d'éclaircissement,
- *capacité_niche*: capacité de chaque niche (nombre maximum de meneurs dans une même niche).

Quand cette procédure d'éclaircissement est appliquée, la sélection ségrégationnelle est légèrement modifiée, tout en gardant le même principe. Elle commence toujours par sélectionner une proportion τ_{select} des individus faisables, mais si l'ensemble des meneurs ne contient pas cette proportion en complet, elle termine le reste en sélectionnant les meilleurs faisables parmi les suiveurs. Ainsi, ASCHEA garde son principe de base qui est de maintenir un degré de faisabilité minimum dans la population.

L'application de cette procédure pour le cas test G2 a nettement amélioré la qualité des résultats retournés par l'algorithme. Le tableau suivant montre la différence entre les solutions données par ASCHEA avant et après l'ajout de la procédure d'éclaircissement. Ces résultats ont été obtenus avec une capacité de niche égale à 5 et un rayon d'éclaircissement égale à 0.08.

	Avant éclaircissement	Après éclaircissement rayon=0.08, capacité=5
Meilleur	0.785	0.802483
médiane	0.56	0.796397
Moy.	0.59	0.788221

TAB. 5.3 – *Résultats pour le cas test G2 avec et sans l'éclaircissement, donnant la meilleure performance, la performance médiane et moyenne sur les 31 tests.*

Le passage de la valeur médiane de 0.56 à 0.79 pour le cas test G2 prouve que la mauvaise qualité des résultats donnés par les expérimentations préalables est due essentiellement à l'absence de nichage. En effet, la multitude d'optima locaux de G2 nécessite une telle procédure pour éviter la convergence prématurée.

Pour cette deuxième série d'expériences pour G2, plusieurs rayons d'éclaircissement on été testés, en quête du meilleur. La courbe 5.8 illustre la performance médiane ainsi que la meilleure performance obtenues après 5000 générations d'évolution, avec des rayons d'éclaircissement de 0.01, 0.08, 0.1, 0.2 et 0.3. La capacité de chaque niche a été limité à 5 pendant la première série d'essais, et elle a été augmentée à 8 pour une deuxième série.

On remarque que les meilleurs résultats sont trouvés en utilisant un rayon d'éclaircissement égale à 0.08, que ce soit avec une capacité de niche de 5 ou de 8. Avec une capacité égale à 8, ASCHEA a donné 0.8036 comme valeur maximale, qui ne diffère que de $2.36.10^{-3}\%$ de la meilleure solution connue (0.803619). La qualité de ces résultats se détériore progressivement avec l'augmentation de la valeur de r_e. Toutefois, la pire performance a été enregistrée avec $r_e = 0.01$. En fait, cette valeur est trop petite pour la procédure de nichage, qui dégrade alors la performance de l'algorithme. En effet, une petite valeur de r_e ne permet pas la diversification de la population, spécialement dans la première phase de l'évolution. Vue la petite taille des niches, le nombre de points dans chaque niche est très faible. Ainsi, la majorité des individus sont considérés comme des meneurs, et c'est plutôt la sélection standard d'ASCHEA qui décide des survivants. L'éclaircissement ne commence à avoir d'effet qu'à une phase avancée de l'évolution, où la population commence à converger vers la zone du meilleur courant. Si le nichage n'a eu aucun effet pendant les premières générations, alors toute la population va converger vers la même zone. On risque alors d'avoir une seule niche regroupant tous les individus. En appliquant l'éclaircissement

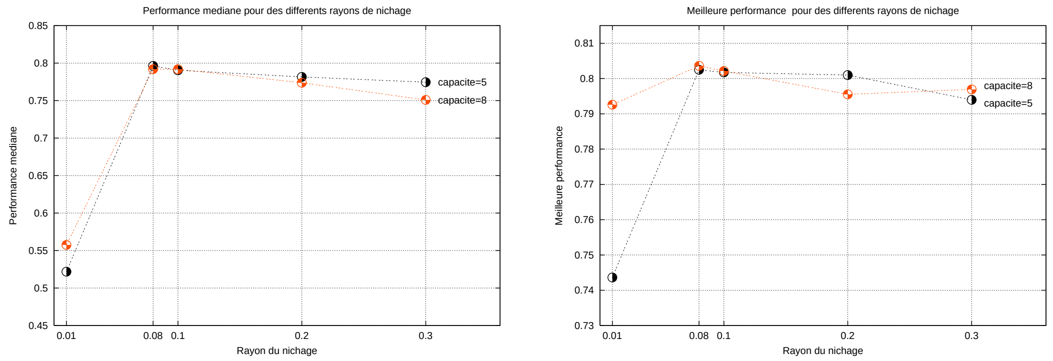
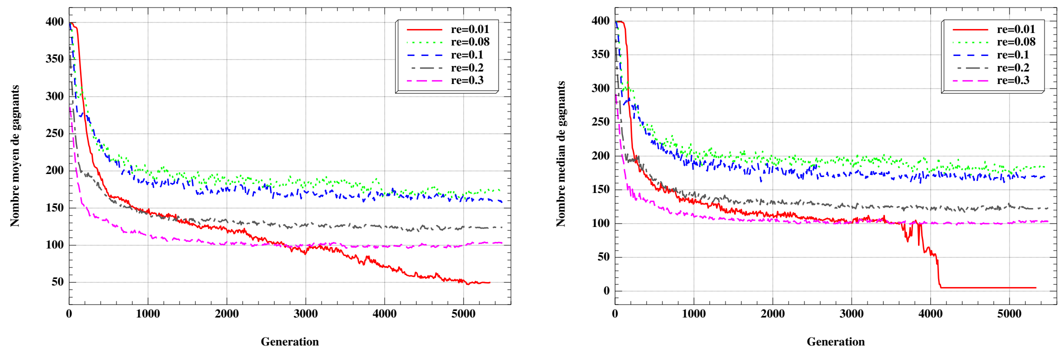


Fig. 5.8 – *Performance médiane (à gauche) et meilleure performance (à droite) données par ASCHEA avec la procédure d'éclaircissement modifiée, pour les rayons de nichage 0.01, 0.08, 0.1, 0.2 et 0.3, et les capacités des niches 5 et 8.*

modifié sur ce groupe, la majorité des individus seront considérés comme des suiveurs. Le nombre de meneurs se limitera alors à la capacité autorisée pour chaque niche. C'est ce qui est passé dans le cas où le rayon de nichage est égale à 0.01. La figure 5.9 montre bien comment le nombre de meneurs devient très faible à partir de la génération 4000, où il devient égale à 5 (capacité de chaque niche). Ainsi, il n'y a plus qu'une seule niche dans la population et la diversité est perdue.



(a) Nombre moyen de meneurs (b) Nombre médian de meneurs

Fig. 5.9 – *Évolution du nombre moyen (a) et médian (b) des meneurs dans la population après application de l'éclaircissement modifié, avec une capacité de niche égale à 5 et des différents rayons de nichage.*

5.4.1 Adaptation du rayon de nichage

Trouver le rayon d'éclaircissement idéal pour un problème donné n'est pas une tâche simple. En effet, avec une valeur très grande, le nombre de suiveurs est très élevé, vue que la surface d'une niche devient grande. Par contre, un rayon trop petit a tendance d'annuler l'effet du nichage, puisque la majorité des individus sont considérés comme des meneurs. Entre autre, une certaine valeur de r_e peut être idéale pendant les premières générations et trop grande pour une phase avancée de l'évolution, quand la population commence à converger.

Pour ajuster r_e manuellement, il faut tester plusieurs valeurs jusqu'à ce qu'on trouve une valeur qui donne un résultat satisfaisant. Afin d'éviter cette étape, et vu l'apport positif de l'adaptativité à la performance d'ASCHEA, nous avons eu l'idée d'utiliser un rayon adaptatif pour l'éclaircissement. L'objectif de cette stratégie d'adaptation est de créer un certain équilibre entre les meneurs et les suiveurs, tout au long de l'évolution. Si le nombre de suiveurs est très élevé, on réduit r_e. Par contre, on augmente r_e si on juge qu'il y a trop de meneurs dans la population.

Soit $\tau_s(t)$ le taux de suiveurs après application de la procédure d'éclaircissement à la génération t, et $\tau_m(t)$ le taux des meneurs ($\tau_m(t) = 1 - \tau_s(t)$). Soit τ_{clear} le taux maximal de suiveurs, réciproquement meneurs, autorisé dans la population courante ($0 < \tau_{clear} < 0.5$). La procédure d'ajustement de r_e est alors comme suit:

`si` $\tau_s > \tau_{clear}$

  $r_e(t+1) = r_e(t)/fact_r$

`sinon si` $\tau_m(t) > \tau_{clear}$

  $r_e(t+1) = r_e(t) * fact_r$

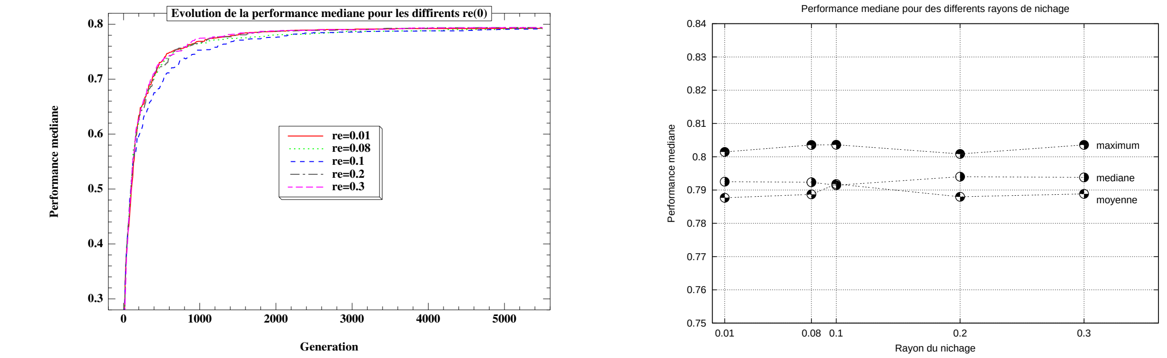
avec $fact_r$ est un paramètre positif supérieur à 1. Par mesure de simplicité, $fact_r$ prendra la même valeur que le facteur d'adaptation des cœfficients de pénalité ($fact_r = fact = 1.1$). La valeur initiale du rayon d'éclaircissement $r_e(0)$ est fixé par l'utilisateur. Il est possible de la définir automatiquement avec une procédure de recherche dichotomique, mais cette procédure est coûteuse. Nous avons donc préféré la définir manuellement.

Comme avec la version précédente de l'éclaircissement, des expériences de 31 tests ont été effectuées avec un rayon d'éclaircissement adaptatif, dans les même conditions expérimentales. Le paramètre τ_{clear} a été fixé à 0.4. Afin de visualiser l'effet de l'adaptativité de r_e sur la qualité des résultats, les mêmes valeurs de r_e essayées avec un rayon de nichage statique ont été prises comme des valeurs de départ ($r_e(0) \in \{0.01, 0.08, 0.1, 0.2, 0.3\}$).

La courbe 5.10(a) illustre l'évolution de la valeur médiane des meilleures performances sur 31 tests, données par l'algorithme en partant de l'une des valeurs initiales $r_e(0)$ testées. La première observation qu'on déduit de cette figure est que les courbes correspondant aux différentes valeurs de $r_e(0)$ sont presque confondues. On en conclue que la dépendance de l'efficacité de la procédure d'éclaircissement à la valeur du rayon de nichage a largement diminué en adaptant ce dernier au cours de l'évolution. Même la valeur 0.01 qui a donné des résultats de qualité médiocre avec la version sans adaptation, a réussi, avec l'adaptation, à donner des résultats d'une qualité presque équivalente à ceux obtenus avec les autres rayons.

La courbe 5.11(b) montre que les résultats enregistrés avec les différentes valeurs de $r_e(0)$ sont très proches. Par exemple, les meilleures performances sont comprises entre 0.80086 (pour $r_e = 0.2$) et **0.803614** (pour $r_e = 0.1$), et la performance médiane varie entre 0.07923 (pour $r_e = 0.08$) et 0.794 (pour $r_e = 0.2$). On note qu'aucune des méthodes présentées dans le chapitre 2 qui ont essayé de résoudre ce problème n'a réussi à atteindre la qualité de la meilleure performance atteinte par ASCHEA. La meilleure solution est de 0.803515, et elle a été donnée par la méthode de classement stochastique (chapitre 2, section 2.3.7).

La courbe 5.11 illustre le nombre médian des meneurs dans la population pour les différentes valeurs de $r_e(0)$ testées. On remarque que, pour toutes les valeurs de $r_e(0)$, le nombre de meneurs



(a): Courbe d'évolution de la performance médiane (b): Performance moyenne, médiane et maximale

FIG. 5.10 – *Performances obtenues pour le cas test G2 avec l'éclaircissement élitiste modifié et rayon de nichage adaptatif. Les valeurs initiales du rayon d'éclaircissement sont: 0.01, 0.08, 0.1, 0.2 et 0.3, et la capacité des niches est limitée à 5*

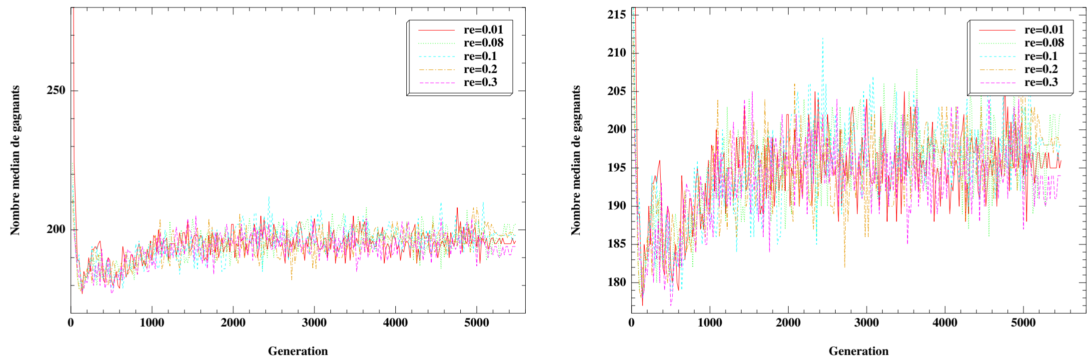


FIG. 5.11 – *Évolution du nombre médian des meneurs dans la population avec l'application de l'éclaircissement modifié, pour une capacité de niche égale à 5 et des différents rayons de nichage. La courbe de droite est un zoom de celle à gauche pour un nombre de meneurs compris entre 175 et 215.*

oscille entre 180 et 210. On rappelle que la procédure d'éclaircissement est appliquée avant la sélection, donc le nombre de meneurs est mesuré sur l'ensemble des parents et des enfants, qui comptent 400 individus pour nos expérimentations.

5.4.2 Extension des tests

Le nichage est destiné en premier lieu aux fonctions multimodales afin d'identifier les optima locaux et éviter la convergence vers l'un d'eux. Toutefois, il peut être appliqué aussi à tout type de problèmes dans le but de maintenir la diversité de la population. Nous avons alors essayé d'appliquer la procédure d'éclaircissement avec rayon adaptatif, définie dans ce chapitre, sur quelques cas test de la section 5.3. Nous avons sélectionné les cas test où ASCHEA a échoué à localiser l'optimum pour un essai ou plus parmi l'ensemble des 31 essais effectués dans la section 5.3. Les fonctions sélectionnées sont alors G1, G3, G5, G7, G9 et G10. Les conditions expérimentales restent inchangées. Les paramètres de l'éclaircissement sont définis comme suit: $\tau_{clear} = 0.4$, $r_e(0) = 0.1$, capacité_niche=5 et 8. Le tableau 5.4 illustre les résultats obtenus pour les différents cas test.

Fonction	Opt.	capacité=5			Capacité=8		
		Meilleur	Médiane	Moyenne	Meilleur	Médiane	Moyenne
G1	-15	-15	-14.9999	-14.987	-15	-15	-14.96
G3	1	1	0.999997	0.99994	1	0.999995	0.99997
G5	5126.49	5126.5	**5126.5**	**5126.72**	5126.5	**5126.5**	**5126.53**
G7	24.3	24.34	24.52	24.59	24.32	24.51	24.59
G9	680.63	680.63	680.691	680.725	680.633	680.7	680.73
G10	7049.33	**7049.51**	7120.25	7212.21	**7049.42**	7272.19	7615.2

TAB. 5.4 – *Résultats pour les cas test G1, G3, G5, G7, G9 et G10 avec l'éclaircissement élitiste modifié et rayon adaptatif, obtenus avec une capacité de niche égale à 5 (les 3 colonnes de gauche) et 8 (les 3 colonnes de droite).*

Etant donnée que la caractéristique de base d'ASCHEA est d'assurer la diversité de la population, l'introduction de l'éclaircissement pour plus de diversité n'a pas enregistré une très grande amélioration de la qualité des résultats, sauf pour les cas test G5 et G10.

On note une amélioration au niveau de la performance moyenne des fonctions G1 et G3, mais pas au niveau de la performance médiane. Pour le cas test G1, l'éclaircissement a aidé l'algorithme à éviter les optima locaux entourant l'optimum global. Sans le nichage, ASCHEA a enregistré 29 succès sur les 31 essais, et pour les deux autres essais, il est tombé dans le piège de la convergence prématurée, où il a donné -12.4531 pour les 2 échecs. Avec le nichage, il a réussi à éviter cet optimum local et s'approcher de très près l'optimum global, mais il a enregistré une baisse au niveau du degré de précision.

Ceci est dû au fait que la procédure d'éclaircissement est basée sur un classement selon la performance des individus, sans tenir compte de leur faisabilité. Ainsi, le meilleur faisable n'occupe pas toujours le premier rang, et il peut être éliminé au cours de l'éclaircissement, s'il n'est pas classé meneur. Un moyen pour augmenter la probabilité de survie des meilleurs faisables est d'augmenter la capacité des niches. La courbe 5.12 montre que la dégradation de la qualité de la meilleure solution faisable est fréquente avec une capacité égale à 5, spécialement au début de l'évolution, alors qu'elle est beaucoup moins fréquente avec une capacité égale à 8. De plus, les 3 colonnes de droite du tableau 5.4 montrent que la performance médiane de la fonction G1 s'est améliorée, ainsi que le taux de précision des résultats. Cette amélioration est remarquée aussi pour le cas de G7, mais pas pour les autres tests.

L'éclaircissement a nettement amélioré les résultats du problème G5, où ASCHEA a réussi à approcher de très près l'optimum dans tous les essais. Ce test est considéré difficile parce qu'il regroupe des contraintes d'égalité non linéaires et d'inégalité linéaires en même temps, et peu de méthodes de prise en compte de contraintes ont pu donner des résultats satisfaisants. En donnant des résultats qu'aucune autre technique n'a réussi à atteindre, ASCHEA prouve sa grande robustesse et son habilité de prendre en compte tout type de contrainte.

Par ailleurs, la meilleure amélioration est enregistrée pour le cas test G10. Ce dernier est un

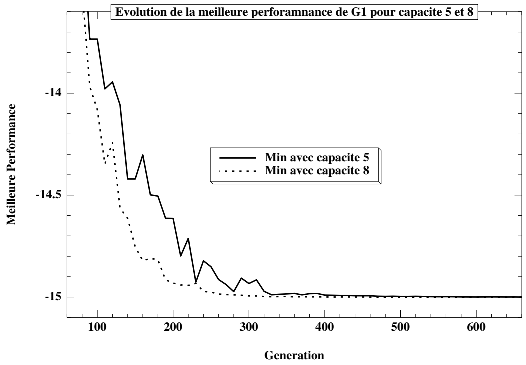


FIG. 5.12 – *Courbes d'évolution de la performance de la meilleure solution faisable (sur 31 essais) pour le cas test G1, avec l'éclaircissement modifié et une capacité des niches de 5 et 8.*

problème prouvé difficile, nécessitant de l'algorithme des grandes capacités d'exploration. L'introduction de l'éclaircissement a permis à ASCHEA de donner la meilleure performance jamais enregistrée pour ce problème. Il a réussi à donner 7049.42, c. à d. une erreur de seulement $1.28.10^{-3}\%$ par rapport à l'optimum global.

5.5 Conclusion

Dans la version originale d'ASCHEA, la fonction de pénalisation utilise un seul cœfficient pour toutes les contraintes du problème, et la faisabilité de chaque individu est calculée par rapport à l'ensemble des contraintes.

L'unicité du cœfficient de pénalisation peut être gênante pour le processus de recherche si le degré de rigidité des contraintes est variable. Nous avons alors introduit dans ce chapitre une nouvelle définition de la fonction de pénalisation, utilisant un cœfficient de pénalisation propre à chaque contrainte. Chaque cœfficient est adapté au cours de l'évolution selon le degré de faisabilité de la population par rapport à la contrainte correspondante.

Les expérimentations réalisées sur les mêmes fonctions de référence que le chapitre précédent ont montré une amélioration remarquable au niveau de la qualité des résultats, spécialement pour les fonctions soumises à plusieurs contraintes, dont une proportion est active au niveau de l'optimum. Avec cette nouvelle stratégie de pénalisation, ASCHEA a pu orienter le processus de recherche vers les zones des contraintes les plus difficiles à satisfaire, et assurer ainsi une meilleure exploration dans leurs zones.

Par ailleurs, en vue de prendre en compte les fonctions multimodales, une technique de nichage a été introduite à ASCHEA. Elle est inspirée de l'éclaircissement élitiste de Pétrowski [93], mais elle a été modifiée, afin de l'adapter aux spécificités d'ASCHEA. En outre, dans le but d'éviter une recherche manuelle lourde du meilleur rayon de nichage pour chaque problème, nous avons ajouté à la technique d'éclaircissement une stratégie d'adaptation de son rayon au cours de l'évolution, qui vise le maintient de l'équilibre entre le nombre de meneurs et celui des suiveurs au cours de la phase de sélection.

L'expérimentation de cette procédure a nettement amélioré la qualité des résultats pour la fonction multimodale G2 et pour les deux fonctions difficiles G5 et G10.

Cependant, malgré la grande robustesse d'ASCHEA dans la résolution de l'ensemble des cas test de référence, sa stratégie de traiter les contraintes d'égalité en les transformant en inégalités ne peut pas être généralisée. En effet, cette stratégie suppose des connaissances à priori sur les contraintes. Dans le chapitre suivant, nous introduisons à ASCHEA une nouvelle stratégie de prise en compte des contraintes d'égalité, basée, comme toutes les composantes d'ASCHEA sur l'adaptativité. Elle transforme une équation en un couple d'inéquations. Le domaine de faisabilité définie par ces deux inéquations est ensuite réduit progressivement, dans le but de l'approcher de l'espace faisable de mesure nulle.

Chapitre 6

ASCHEA: Prise en compte des contraintes d'égalité

Résumé

Dans les deux chapitres précédents, nous avons introduit ASCHEA: un algorithme pour l'optimisation sous contraintes, basé sur une approche adaptative pour ajuster les cœfficients de pénalisation, permettant de maintenir la diversité de la population. Il utilise aussi une sélection ségrégationnelle pour maintenir un certain degré de faisabilité, et un croisement basé une stratégie de séduction/sélection pour stimuler l'exploration de la frontière du domaine faisable. ASCHEA a montré une grande robustesse dans la résolution de plusieurs cas test de référence.

Cependant, pour prendre en compte les contraintes d'égalité, ASCHEA se contente de les transformer en inégalités. Malgré les bons résultats obtenus, cette technique ne peut pas être généralisée, car elle suppose des connaissances à priori sur les contraintes (c.f. chapitre 5).

Ce chapitre propose des solutions originales pour prendre en compte les contraintes d'égalité. Chaque équation est transformée en un couple d'inéquation, définissant un domaine de faisabilité initial d'une taille moyennement petite. Il est ensuite réduit progressivement au cours de l'évolution, afin de l'approcher le maximum possible de l'espace de mesure nulle. Deux stratégies sont proposées, une par ajustement dynamique et l'autre par ajustement adaptatif.

L'étude expérimentale des deux approches sur 3 cas test de référence a donné des résultats de bonne qualité, où la violation des contraintes d'égalité est presque nulle.

6.1 Introduction

Dans les deux chapitres précédents, nous avons introduit l'algorithme ségrégationnel adaptatif pour l'optimisation sous contraintes: ASCHEA. Pour prendre en compte les contraintes, ASCHEA possède trois composantes principales. La première est une fonction de pénalisation adaptative, dont le but est de maintenir la diversité de la population, en favorisant les individus faisables ou infaisables, selon le degré de faisabilité de la population. Les deux autres composantes s'appliquent pendant les opérations de sélection et de croisement, afin de garder un certain degré de faisabilité (grâce à la sélection ségrégationnelle) et stimuler l'exploration de la frontière de l'espace faisable (grâce à la stratégie de séduction/sélection). La fonction de pénalisation a été optimisée dans le chapitre 5, en utilisant un cœfficent propre à chaque contrainte. Cette modification a amélioré la robustesse d'ASCHEA. Entre autres, une procédure de nichage a été introduite comme option à ASCHEA, en vue de traiter les fonctions multimodales.

Tous ces ingrédients ont donné à ASCHEA une capacité étonnante dans la résolution de plusieurs cas test de référence, spécialement les problèmes avec contraintes d'inégalité, quelques soit leur type et quelques soit la topologie de l'espace faisable.

Grâce à sa grande capacité à explorer la frontière du domaine faisable, ASCHEA réussi à prendre en compte avec succès les contraintes d'égalité en les transformant en inégalités. Toutefois, la manière dont ces contraintes sont traitées ne peut pas être généralisée, du fait que le domaine de faisabilité de la contrainte transformée devient largement plus grand que celui défini par l'égalité. Ainsi, si le nouvel espace de faisabilité contient des optima locaux ayant une performance meilleure que l'optimum global, on risque la convergence vers une solution infaisable.

Pour permette à ASCHEA d'être appliqué sur tout genre de problème, une amélioration de sa technique pour la prise en compte des contraintes d'égalité s'impose. Cette technique doit être généralisable et doit s'adapter avec toute forme d'espace faisable. Ce chapitre est consacré à cet objectif. Deux approches de prise en compte des contraintes d'égalité sont proposées et comparées.

L'idée générale est de partir, pour une contrainte d'égalité donnée h_j, d'un espace faisable $\mathcal{F}_{h_j}(0)$ assez grand, limité par deux contraintes d'inégalité. Cet espace est ensuite réduit progressivement aux cours de l'évolution, dans le but de l'approcher au maximum de l'espace faisable réel, de mesure nulle.

Pour se faire, deux stratégies sont proposées. La première est une approche par ajustement dynamique, où l'espace est réduit selon un schéma choisi préalablement. Au contraire, dans la deuxième approche, le domaine $\mathcal{F}_{h_j}(t)$ est réduit adaptativement selon l'état courant de la population. L'idée est de garder quelques individus faisables (appartenant à $\mathcal{F}_{h_j}(t+1)$ après la réduction), afin qu'ils contribuent à attirer les autres individus vers le nouveau domaine de faisabilité.

Une étude expérimentale comparative des deux approches est présentée dans la deuxième partie de ce chapitre, illustrant des résultats obtenus avec 3 fonctions de référence.

6.2 Prise en compte des contraintes d'égalité

Dans la première version d'ASCHEA, les contraintes d'égalité $h_j(\overrightarrow{x}) = 0$ sont transformées en des contraintes d'inégalité $h_j(\overrightarrow{x}) \leqslant \epsilon$, où ϵ est un nombre positif très petit. Cette technique repose sur la capacité d'ASCHEA d'explorer efficacement la frontière. Mais elle peut échouer si on ne sais pas de quel côté de la surface de l'égalité il faut chercher. Elle peut échouer aussi en présence de plusieurs équations de différents types, ou à cause de l'existence d'optima locaux dans le nouvel espace de faisabilité défini par l'inégalité.

Nous proposons dans cette section une nouvelle méthode pour prendre en compte les contraintes d'égalité. L'idée de base est toujours de transformer les égalités en inégalités, mais aussi de réduire progressivement l'espace faisable de la contrainte en question. Ainsi, une contrainte d'égalité est considérée comme un couple de contraintes d'inégalité:

$$-\epsilon_j(t) \leqslant h_j(\overrightarrow{x}) \leqslant \epsilon_j(t),$$

avec ϵ_j un nombre positive dont la valeur est mise à jour automatiquement par l'algorithme au fur et à mesure de l'évolution.

Le but est de diminuer $\epsilon_j(t)$ afin qu'elle ait une très petite valeur à la fin de l'évolution. Ceci permettra à la population de s'approcher, au fur et à mesure des générations, de la surface de la contrainte d'égalité correspondante, tout en visitant des différentes régions de l'espace de recherche.

A chaque fois qu'une grande proportion des individus réussissent à joindre l'espace faisable $\mathcal{F}_{h_j}(t)$, ce dernier est réduit d'un certain taux, par diminution de la valeur $\epsilon_j(t)$ d'une certaine quantité. Les nouveaux individus infaisables sont alors forcés vers l'espace faisable courant grâce à la pénalisation et le croisement entre faisables et infaisables.

Pour diminuer la valeur de $\epsilon_j(t)$ (et réduire ainsi $\mathcal{F}_{h_j}(t)$), nous proposons deux stratégies. La première est une approche par ajustement dynamique: $\epsilon_j(t)$ est diminué d'un facteur constant dès que suffisamment d'individus sont faisables. Pour la deuxième approche, $\epsilon_j(t)$ est ajusté adaptativement selon l'état courant de la population: la diminution de $\epsilon_j(t)$ doit préserver la faisabilité d'une proportion fixée d'individus, ce qui devrait permettre d'attirer plus facilement les nouveaux individus infaisables à l'intérieur de $\mathcal{F}_{h_j}(t)$.

Ces deux approches sont présentées ci-dessous en détails, et testées sur quelques problèmes de référence.

6.2.1 Ajustement dynamique

L'idée de l'ajustement dynamique est de diminuer la valeur de $\epsilon_j(t)$ à des moments de l'évolution d'un certain taux fixé préalablement. On réduit alors progressivement et régulièrement l'espace faisable de la contrainte d'égalité correspondante. La population est ainsi attirée progressivement vers l'espace défini par $[-\epsilon_j(t), \epsilon_j(t)]$ $(\mathcal{F}_{h_j}(t))$.

A chaque fois qu'une proportion τ_{reduct} des individus soit à l'intérieur de $(\mathcal{F}_{h_j})$, la taille de ce dernier est réduite d'un certain taux $fact_\epsilon$. Le schéma de mise à jour de ϵ_j est comme suit:

$$\begin{array}{ll} \text{si } (\tau_t(j) > \tau_{reduct}) & \epsilon_j(t+1) = \epsilon_j(t)/fact_\epsilon \\ \text{sinon} & \epsilon_j(t+1) = \epsilon_j(t) \end{array}$$

où τ_{reduct} et $fact_\epsilon$ sont des paramètres de la méthode définis par l'utilisateur. Nous rappelons que $\tau_t(j)$ est la proportion d'individus de la population respectant la contrainte j à la génération t.

Au début de l'évolution, on autorise un taux d'erreur (violation de l'égalité) assez grand, en affectant des valeurs "élevées" à $\epsilon_j(0)$ $(\simeq 0.1, 0.01, \cdots)$. Ceci permettra à l'algorithme de trouver des faisables facilement. Dès que quelques faisables sont générés, le reste de la population est attiré rapidement vers $\mathcal{F}(t)$ grâce à la pénalisation et au croisement.

Une fois qu'une grande proportion (supérieure à τ_{reduct}) de la population est faisable, on réduit $\mathcal{F}_{h_j}(t)$ des deux côtés. Ce phénomène se répète jusqu'à la fin de l'évolution (figure 6.1).

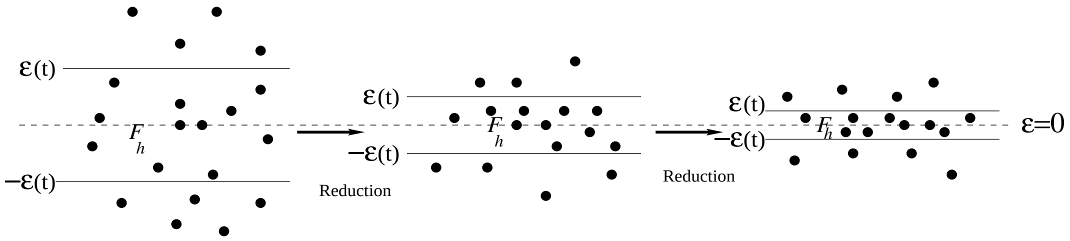


FIG. 6.1 – *Réduction progressive de l'espace faisable d'une contrainte d'égalité avec l'approche par ajustement dynamique.*

La valeur de τ_{reduct} doit être supérieure à τ_{target} (taux limite d'adaptation de la pénalisation), afin de permettre au cœfficient de pénalité de la contrainte correspondante d'augmenter et de diminuer normalement selon le schéma défini par l'équation 5.4 du chapitre 5. En effet, si $\tau_{reduct} \leqslant \tau_{target}$, le taux de faisabilité de la population pour les contraintes d'égalité ne dépassera jamais τ_{target}. Dans ce cas, les cœfficients de pénalité correspondants sont augmentés continuellement au cours de l'évolution, et leur utilité décroît en parallèle. Car, avec des cœfficients de pénalisation très élevés, le but principal d'ASCHEA, qui de maintenir la diversité de la population en favorisant à certains moments les infaisables, n'est plus assuré, du fait que ces derniers sont fortement pénalisés.

6.2.2 Ajustement adaptatif

Avec l'approche par ajustement dynamique, la réduction de l'espace faisable $\mathcal{F}_{h_j}(t)$ se fait selon un schéma fixe tout au long de l'évolution, où $\epsilon_j(t)$ est diminué sans tenir compte du nouvel état de la population, après la réduction de son espace de faisabilité. Si la population est très proche de la frontière de $\mathcal{F}_{h_j}$ à la génération t, la réduction de l'espace faisable peut causer la perte de tous les faisables pour la contrainte correspondante à la génération $t + 1$. Ceci risque d'avoir des conséquences négatives sur le bon fonctionnement d'ASCHEA. Si l'algorithme arrive à générer rapidement des individus dans le nouvel espace faisable, aucune difficulté ne se pose. Par contre, si l'algorithme perd le chemin vers le nouvel espace faisable, il risque de retourner des solutions infaisables à la fin de l'évolution. Entre autres, si la nouvelle frontière est éloignée de la population, l'algorithme peut avoir des difficultés à y attirer la population pendant un grand moment de l'évolution. Pendant ce temps, les cœfficients de pénalité continuent d'augmenter. On se retrouve donc, dans une phase un peu avancée de l'évolution, avec des cœfficients de pénalisation très élevés. Ils deviennent donc incapables de favoriser dans certains cas les infaisables, puisque ces derniers sont fortement pénalisés. Entre autres, des cœfficients de pénalité très élevés risquent de forcer les infaisables vers une zone unique dans l'espace. A ce moment, la pénalisation ne joue plus aucun rôle puisque les degrés de violation sont similaires.

Pour essayer d'éviter ce risque, nous proposons dans cette section une deuxième approche pour mettre à jour ϵ_j, basée sur un ajustement adaptatif. L'idée est d'adapter ϵ_t de telle façon qu'il y ait toujours des faisables dans la population pour la contrainte correspondante. Pour se faire, ϵ_j est diminué d'une certaine quantité tout en permettant à une proportion des individus faisables de garder leur faisabilité.

La réduction de $\mathcal{F}_{h_j}(t)$ se fait alors selon la distance des individus situés dans son domaine par rapport à ses bords. Le but est que les individus les plus loin de la frontière restent faisables, et ceux qui sont proches deviennent infaisables après le rapprochement des bords vers l'intérieur (figure 6.2). Pour augmenter la chance de génération d'enfants très proches de la surface définie par l'égalité $h_j(\overrightarrow{x}) = 0$, nous avons choisi de garder des individus faisables répartis sur les deux côtés de l'espace limités par la surface de l'égalité (figure 6.2). Le croisement de ces individus entre eux devrait stimuler l'exploration de la frontière réelle. Par conséquent, le taux de rapprochement de chaque bord de $\mathcal{F}_{h_j}$, défini par une des deux inégalités (après transformation de h_j), dépendra du nombre d'individus situés entre le bord et la surface de la contrainte correspondante définie par $h_j(\overrightarrow{x}) = 0$. On utilisera alors deux paramètres ϵ_j^+ et ϵ_j^- [1], correspondant, respectivement, au bord limitant la zone "positive" de $\mathcal{F}_{h_j}$ $(h_j(\overrightarrow{x}) > 0)$ et au bord limitant la zone "négative" de $\mathcal{F}_{h_j}$ $(h_j(\overrightarrow{x}) < 0)$.

Soit $C_j(t)$ la contrainte correspondante à la $j^{\text{ème}}$ contrainte d'égalité transformée selon les

1. Des expériences on été réalisées en utilisant un ϵ_j unique pour chaque contrainte on donné des résultats d'une qualité inférieure à celle des résultats obtenus avec l'approche proposée

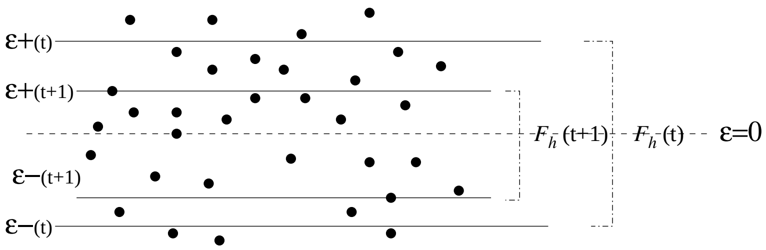


FIG. 6.2 – *Réduction adaptative de l'espace faisable $\mathcal{F}_{h_j}(t)$, de telle façon qu'une proportion des individus appartiennent à $\mathcal{F}_{h_j}(t+1)$.*

valeurs courantes de $\epsilon_j^-(t)$ et $\epsilon_j^+(t)$:

$$C_j(t) : \epsilon_j^-(t) \leqslant h_j(\overrightarrow{x}) \leqslant \epsilon_j^+(t)$$

Le calcul des nouvelle valeurs de $\epsilon_j^-(t+1)$ et $\epsilon_j^+(t+1)$ pour la contrainte h_j se fait comme suit.

Soient $(h_j^+)_i,\ i = 1, \cdots n_{\mathcal{F}}^+$, les valeurs positives de la fonction h_j dans la population courante tel que $0 \leqslant (h_j^+)_i \leqslant \epsilon_j^+(t)$, et $(h_j^-)_i,\ i = 1, \cdots n_{\mathcal{F}}^-$ les valeurs négatives de la fonction h_j tel que $\epsilon_j^-(t) \leqslant (h_j^-)_i \leqslant 0$. La variable $n_{\mathcal{F}}^+$ indique le nombre d'individus respectant $C_j^+(t) : 0 \leqslant h_j(\overrightarrow{x}) \leqslant \epsilon_j^+(t)$ et $n_{\mathcal{F}}^-$ indique le nombre d'individus respectant $C_j^-(t) : \epsilon_j^-(t) \leqslant h_j(\overrightarrow{x}) \leqslant 0$.

On classe les valeurs de h_j^+ selon un ordre croissant et celles de h_j^- selon un ordre décroissant. On obtient alors les deux listes suivantes:

$$\begin{aligned} \mathcal{L}_h^+ &= ((h_j^+)_1, (h_j^+)_2, \cdots, (h_j^+)_{n_{\mathcal{F}}^+}) \\ \mathcal{L}_h^- &= ((h_j^-)_1, (h_j^-)_2, \cdots, (h_j^-)_{n_{\mathcal{F}}^-}) \end{aligned}$$

Si on suppose qu'on veut qu'une proportion $\tau_{equality}$ des individus faisables par rapport à $C_j(t)$ restent faisables après l'ajustement de $\epsilon_j^-(t)$ et $\epsilon_j^+(t)$, les valeurs de ces derniers sont alors adaptées selon la valeur de l'élément limitant cette proportion dans chaque liste (figure 6.3). Soient $indice1$ et $indice2$ le numéro de cet élément dans, respectivement, la liste $\mathcal{L}_h^+$ et la liste $\mathcal{L}_h^-$. Chaque numéro est défini par la partie entière du produit du nombre d'éléments de la liste correspondante par la proportion $\tau_{equality}$:

$$\begin{aligned} indice1 &= E(n_{\mathcal{F}}^+ \times \tau_{equality}) \\ indice2 &= E(n_{\mathcal{F}}^- \times \tau_{equality}) \end{aligned}$$

$\epsilon_j^+(t)$ et $\epsilon_j^-(t)$ prennent alors les valeurs des violations des éléments d'ordre $indice1$ et $indice2$ dans, respectivement, $\mathcal{L}_h^+$ et $\mathcal{L}_h^-$.

Afin de considérer le cas où une des listes $\mathcal{L}_h^+$ ou $\mathcal{L}_h^-$ est vide (ce cas se présente si tous les individus faisables respectent soit $C_j^+(t)$ soit $C_j^-(t)$), ainsi que le cas où il n'y a pas assez d'éléments dans la liste $\mathcal{L}_h^+$ (respectivement $\mathcal{L}_h^-$) permettant de garder au moins un faisable vérifiant $C_j^+(t+1)$ (respectivement $C_j^-(t+1)$) après l'adaptation de $\mathcal{F}_{h_j}(t)$, un test sur la valeur de l'indice calculé est ajouté. Entre autres, comme pour le cas de l'ajustement dynamique, la réduction de l'espace

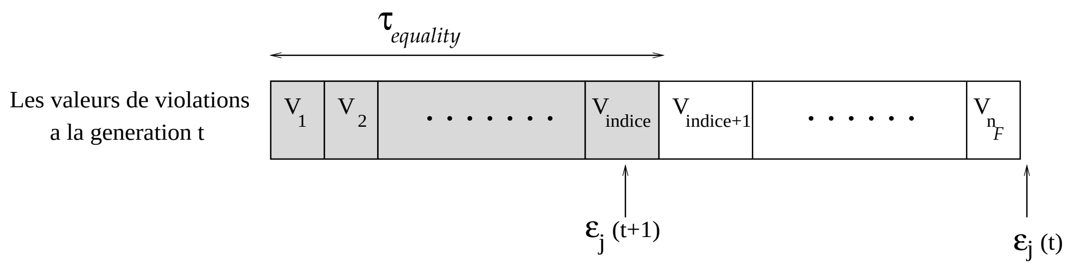


FIG. 6.3 – *Mise à jour de $\epsilon_j(t)$ avec l'approche par ajustement adaptatif, tel que les individus ayant les violations $V_1, V_2, \cdots, V_{indice}$ gardent leur faisabilité.*

faisable ne se fait que si le taux de faisabilité de la population pour la contrainte j est supérieure à une certaine limite τ_{reduct}.

Le schéma final de mise à jour de $\epsilon_j^+(t)$ est alors:

$$\begin{array}{ll} \text{Si } (\tau_t(j) > \tau_{reduct}) & \text{et } (indice1 > 0) \\ \quad \text{alors} & \epsilon_j^+(t+1) = (h_j^+)_{indice1} \\ \quad \text{sinon} & \epsilon_j^+(t+1) = \epsilon_j^+(t) \end{array}$$

De même pour $\epsilon_j^-(t)$:

$$\begin{array}{ll} \text{Si } (\tau_t(j) > \tau_{reduct}) & \text{et } (indice2 > 0) \\ \quad \text{alors} & \epsilon_j^-(t+1) = (h_j^-)_{indice2} \\ \quad \text{sinon} & \epsilon_j^-(t+1) = \epsilon_j^-(t) \end{array}$$

6.3 Etude Expérimentale

Afin de pouvoir expliquer graphiquement le comportement de l'algorithme, une fonction à deux dimensions à été choisie (G11), soumise à une seule contrainte d'égalité avec deux optima possibles. La définition de cette fonction est donnée ci-dessous et les positions des deux optima sont présentées dans la figure 6.4.

- **Minimiser** $G11(\overrightarrow{x}) = x_1^2 + (x_2 - 1)^2$,
  sous la contrainte non linéaire suivante:
  $x_2 - x_1^2 = 0$,
  et les bornes:
  $-1 \leqslant x_i \leqslant 1,\ i = 1, 2.$

Son optimum global est $\overrightarrow{x}^* = (\pm 0.70711, 0.5)$, avec $G11(\overrightarrow{x}^*) = 0.75000455$.

De plus, nous avons testé notre approche sur deux autres fonctions de référence G3 et G5, dont la définition est donnée dans l'annexe A. La fonction G3 est soumise à une contrainte d'égalité non linéaire et $G3(\overrightarrow{x}^*) = 1$, et la fonction G5 est soumise à 3 contraintes d'égalité non linéaires et 2 contraintes d'inégalité linéaires et $G5(\overrightarrow{x}^*) = 5126.49$.

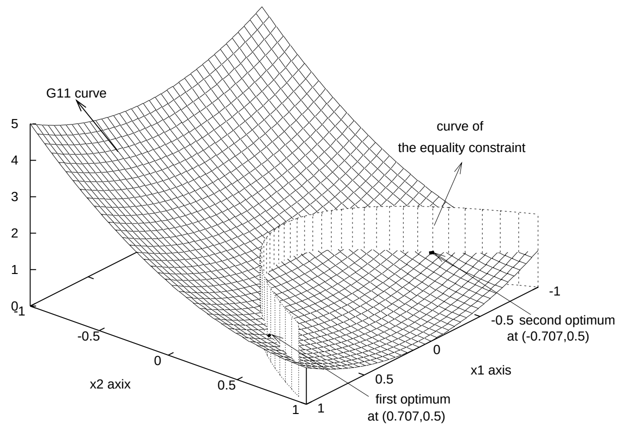


FIG. 6.4 – *Surface de la fonction G11. La courbe de la contrainte d'égalité est rendue visible artificiellement.*

6.3.1 Conditions expérimentales

Comme pour les deux chapitres précédents, ASCHEA utilise toujours un ES(100+300) avec la sélection ségrégationnelle décrite dans le paragraphe 4.2.3 du chapitre 4. Aucun changement n'est effectué pour les paramètres standards: probabilité de mutation et de croisement égale à 0.9, la stratégie de mutation est celle auto-adaptative anisotrope avec une déviation standard initiale de 0.03 et des taux d'adaptation local et global de 0.1 et 1, respectivement. Les paramètres spécifiques à ASCHEA gardent les mêmes valeurs que celle utilisées par le chapitre 5:

- $\tau_{select} = 0.3$
- $\tau_{target} = 0.5$
- $fact = 1.1$

Le paramètre $fact_\epsilon$ de l'approche par ajustement dynamique a été fixé à 1.01. Nous avons évité d'utiliser des valeurs plus grandes afin de ne pas réduire brusquement la taille de l'espace faisable, et perdre ainsi tous les faisables.

En ce qui concerne le paramètre τ_{reduct} utilisé dans les deux approches et le paramètre $\tau_{equality}$ de l'approche par ajustement adaptatif, plusieurs valeurs ont été testées en quête des meilleures approximations. 31 essais ont été effectués pour chaque configuration, avec un nombre de générations limité à 5000.

6.3.2 Ajustement Dynamique

Les résultats donnés par l'approche par ajustement dynamique sont d'une qualité très satisfaisante, où l'idée de diminuer progressivement la taille de l'espace faisable s'est avérée efficace. En effet, les taux de violation atteints à la fin de l'évolution sont très petits. L'étude expérimentale du cas test G11 a montré que, si les paramètres τ_{reduct} et $fact_\epsilon$ sont bien ajustés, l'approche dynamique est robuste. L'objectif général de cette stratégie, consistant dans la réduction progressive de l'espace faisable de la contrainte d'égalité du problème testé, est bien accompli. En effet, la valeur de $\epsilon(t)$ arrive jusqu'à 10^{-12}, ce qui est considéré comme un taux d'erreur très faible. La figure 6.5

montre l'évolution de la valeur de $\epsilon(t)$ au cours de l'évolution, illustrant la réduction progressive de l'espace faisable correspondant.

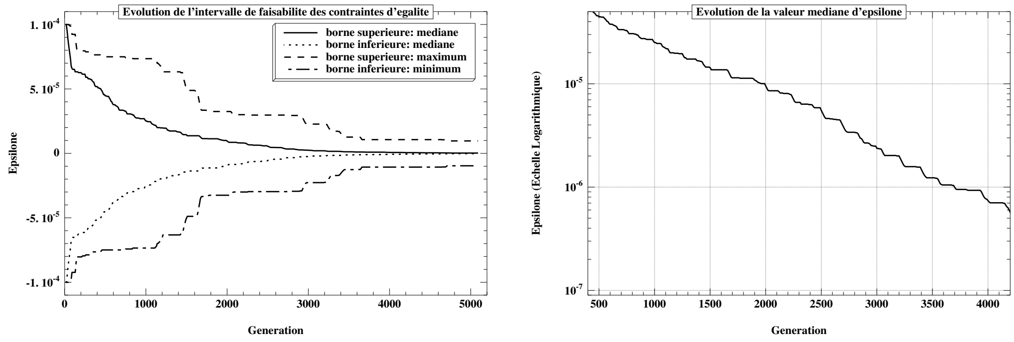


FIG. 6.5 – *Evolution de l'intervalle de faisabilité médian et maximal au cours de la résolution de G11 avec l'approche par ajustement dynamique. La courbe de droite présente la valeur médiane de $\epsilon_j(t)$ à l'échelle logarithmique.*

La figure 6.6 montre un exemple d'évolution de l'espace faisable de G11 entre les générations 25 et 45, avec $\tau_{reduct} = 0.7$ et $fact_\epsilon = 1.01$. La courbe de droite, qui est un zoom de celle à gauche, montre comment les frontières de $\mathcal{F}(25)$ se rapprochent pour donner, à la génération 45, l'espace $\mathcal{F}(45)$ légèrement plus petit. Grâce à la petite valeur de $fact_\epsilon$, ce rapprochement se fait avec des petits pas, afin d'essayer de garder quelques individus faisables et de permettre aux individus infaisables de rejoindre facilement le domaine de $\mathcal{F}_{h_j}$ après chaque réduction.

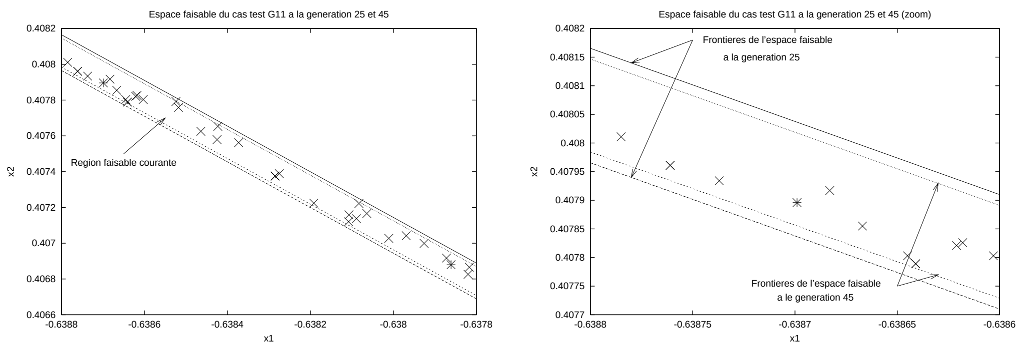


FIG. 6.6 – *Exemple de réduction de l'espace faisable du cas test G11 entre les générations 25 et 45. La courbe à droite est un zoom de celle à gauche pour* $x \in [-0.6388, -0.6386]$.

La qualité des résultats ainsi que la violation de la contrainte h de G11 sont dépendantes de la valeur de τ_{reduct}. Le tableau 6.1 montre que les valeurs de violation augmentent avec l'augmentation de τ_{reduct}, où leur moyenne sur les 31 essais passe de $7.43.10^{-11}$ avec $\tau_{reduct} = 0.6$ à $1.29.10^{-4}$ avec $\tau_{reduct} = 0.9$. Par contre, la performance de la meilleure solution est d'autant meilleure que τ_{reduct} est plus élevé.

En fait, la fréquence de modification de $\epsilon(t)$ diminue avec l'augmentation de τ_{reduct}, vu que la population met plus de temps à atteindre le taux de faisabilité demandé. L'espace faisable final est alors plus grand, donnant à l'algorithme la possibilité de trouver une solution plus performante, mais qui peut être plus loin de l'optimum global.

Dans le cas de problèmes réels, la satisfaction des contraintes est nécessaire. En effet, on préfère avoir une bonne solution qui ne viole pas les contraintes que d'avoir une très bonne solution mais qui viole légèrement les contraintes. En considérant cet objectif, et selon les résultats du tableau 6.1, nous pouvons conclure que le meilleur taux pour le paramètre τ_{reduct} est de 0.6.

Il est possible que les autres valeurs testées arrivent à atteindre la même performance que celle observée avec $\tau_{reduct} = 0.6$, mais il leur faudra beaucoup plus de générations pour avoir un espace de faisabilité aussi petit que celui obtenu avec la valeur 0.6. Nous avons alors effectué des expériences en fixant la violation maximale permise à 10^{-12}, afin d'évaluer le nombre de générations nécessaires pour chacune des configurations pour atteindre ce degré de précision. Les résultats de ces expériences sont présentés dans le tableau 6.2.

τ_{reduct}	Performance			Violation			
	Min	Med	Moy	Min	Med	Moy	Max
0.6	0.75	0.75	0.7519	$3.19.10^{-12}$	$2.11.10^{-11}$	$7.43.10^{-11}$	$9.55.10^{-10}$
0.7	0.749998	0.75	0.7515	$3.18.10^{-9}$	$6.82.10^{-8}$	$3.81.10^{-7}$	$2.76.10^{-6}$
0.8	0.749952	0.75	0.7509	$3.74.10^{-7}$	$2.6.10^{-6}$	$7.87.10^{-6}$	$4.76.10^{-5}$
0.9	0.7493	0.749953	0.7517	$3.37.10^{-6}$	$4.66.10^{-5}$	$1.29.10^{-4}$	$6.98.10^{-4}$

TAB. 6.1 – *Résultats du cas test G11 pour les différentes valeurs de τ_{reduct} testées. Les 3 colonnes de gauche illustrent la meilleure performance, la performance médiane et moyenne, alors que les 4 colonnes de droite donnent les valeurs de violation minimale, médiane, moyenne et maximale enregistrées sur les 31 essais.*

Le tableau 6.2 montre que le nombre moyen de générations nécessaires pour atteindre la précision 10^{-12} est presque doublé avec l'augmentation de la valeur de τ_{reduct}, alors que l'amélioration de la performance moyenne est relativement négligeable. De plus, avec $\tau_{reduct} = 0.9$, l'algorithme n'a réussi à atteindre cette précision que pour 16 essais parmi 31. Pour les autres essais, il a perdu tous les faisables et il n'était pas capable de retrouver le chemin vers $\mathcal{F}$. En fait, au fur et à mesure des générations, la population devient de plus en plus homogène, et son déplacement vers l'espace faisable courant de plus en plus lent, jusqu'à ce que la convergence soit définitive et le mouvement de la population devient négligeable. A ce moment, l'ajustement de $\mathcal{F}(t)$ induit une perte non récupérable de tous les faisables. Il est donc nécessaire d'accélérer légèrement la vitesse de réduction de $\mathcal{F}(t)$ afin de pouvoir atteindre le taux de violation prévu avant la convergence définitive de la population. Pour cette raison, nous maintenons le choix de la valeur 0.6 comme une bonne approximation du paramètre τ_{reduct}.

Résultats pour G3 et G5

L'approche par ajustement dynamique a montré une grande robustesse dans la résolution du cas test G11. Pour confirmer cette robustesse, nous étendons l'étude expérimentale à deux autres cas test de référence G3 et G5 (voir Annexe A). Le premier est soumis à une seule contrainte d'égalité, alors que le deuxième est soumis à 3 contraintes d'égalité et 2 contraintes d'inégalité simultanément.

La valeur de τ_{reduct} retenue est celle qui a donné la plus petite violation pour le cas de G11: $\tau_{reduct} = 0.6$.

τ_{reduct}	Nb succès	Nb Générations			Performance		
		Min	Moy	Max	Min	Med	Moy
0.6	31	5 252	6 392	8 670	0.75	0.75	0.75126
0.7	31	9 087	13 144	31 244	0.75	0.75	0.75087
0.8	31	17 153	23 065	37 773	0.75	0.75	0.75049
0.9	16	30 430	36 234	49 005	0.75	0.75	0.75032

TAB. 6.2 – *Résultats obtenus pour le cas test G11 en limitant la violation maximale à* 10^{-12}. *Ils donnent le nombre de générations minimal, moyen et maximal pour atteindre ce degré de précision ainsi que la performance minimale, médiane et moyenne pour les essais munis avec succès.*

Quand l'algorithme retourne des faisables à la fin de l'évolution, les résultats sont de bonne qualité. Nous avons alors ajouté à la condition d'arrêt de l'algorithme un critère sur le degré de faisabilité de la population. Une fois que le nombre maximum de générations est atteint, il doit y avoir au moins un faisable dans la population pour que l'algorithme puisse s'arrêter, sinon il continue jusqu'à ce qu'il en trouve. Ceci nous a permis de calculer les meilleures performances et les performances moyennes et médianes des meilleures solutions faisables sur les 31 essais pour les cas test G3, G5 et G11. Leurs valeurs sont illustrées dans les 3 premières colonnes du tableau 6.3. Les 4 autres colonnes donnent la violation minimale, maximale, moyenne et médiane des meilleures solutions obtenues à la suite de l'ensemble des 31 essais. Pour le cas de G5, la violation est calculée en faisant la somme des violations des différentes contraintes h_j à la génération t.

Les résultats obtenus sont d'une très bonne qualité, où la valeur médiane de G3 n'est qu'à 7.10^{-4}% de l'optimum global, et celle de G5 est de 5.3% de l'optimum, ce qui peut être considéré comme une bonne performance, vue la difficulté du problème. Entre autres, la violation médiane pour les deux cas test est au dessous de 10^{-6}.

On note que pour le cas de G5, seules les méthodes de pénalités dynamiques de Joines et Houk (chapitre 2, section 2.3.3) et celle du classement stochastique de Runarsson et Yao (chapitre 2, section 2.3.7) ont pu atteindre une erreur maximale de 10^{-4}. En réduisant l'erreur jusqu'à 10^{-9}, ASCHEA a accompli un meilleure performance.

Fonction	Opt.	Performance			Violation			
		Meill	Med	Moy	Min	Med	Moy	Max
G3	1	1	0.999993	0.9932	9.18^{-9}	$1.98.10^{-7}$	$4.37.10^{-5}$	$2.66.10^{-4}$
G5	5126.49	5126.52	5154.13	5168.9	$1.6.10^{-9}$	$6.18.10^{-7}$	$2.61.10^{-4}$	0.008
G11	0.75	0.75	0.75	0.7519	$3.19.10^{-12}$	$2.11.10^{-11}$	$7.43.10^{-11}$	$9.55.10^{-10}$

TAB. 6.3 – *Résultats des cas test G3, G5 et G11 obtenus avec l'ajustement dynamique, donnant, pour chaque fonction, la moyenne la médiane et la meilleure performance (les 3 premières colonnes), ainsi que la violation minimale, médiane, moyenne et maximale sur les 31 essais (les 4 dernières colonnes).*

La difficulté que rencontre ASCHEA avec l'approche dynamique est de garder un taux de faisabilité minimal tout au long de l'évolution. En effet, souvent, après la réduction de l'espace faisable, tous les individus faisables deviennent infaisables.

La figure 6.7 montre comment le degré de faisabilité médian de la population de G11 atteint la valeur nulle à plusieurs reprises. Toutefois, grâce à la petite valeur du paramètre $fact_\epsilon$ (1.01) permettant de réduire l'espace faisable très lentement, l'algorithme réussi à générer rapidement des enfants dans le nouvel espace faisable. Cependant, des valeurs plus élevées de $fact_\epsilon$ peuvent engendrer la perte définitive des faisables. C'est le cas de G5 avec $fact_\epsilon > 1.01$ (e.g. $fact_\epsilon = 1.02$), où après quelques opérations de réduction de $\mathcal{F}(t)$, les faisables disparaîssent et ne réapparaîssent plus.

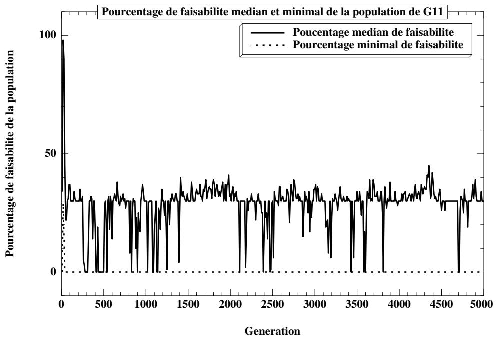


FIG. 6.7 – *Pourcentage médian et minimal (sur 31 essais) de faisabilité de la population au cours de la résolution de G11 avec l'ajustement dynamique ($\tau_{reduct} = 0.6$).*

Quand le degré de faisabilité de la population s'annule, le croisement basé sur la séduction/sélection n'est plus appliqué, et la réapparition des individus faisables devient dépendante principalement de la mutation. Avec la mutation auto-adaptative, la génération d'enfants dans le nouvel espace de faisabilité devient difficile si la population n'est pas proche de la nouvelle frontière ou si les déviations standards sont trop petites. L'utilisation d'un facteur de réduction faible permet de garder une distance réduite entre la population et la frontière. Cependant, à une phase avancée de l'évolution, l'algorithme met plusieurs générations pour générer des enfants faisables, et parfois il devient incapable de retrouver le chemin de $\mathcal{F}(t)$.

Pour aider la population à rejoindre l'espace faisable après chaque réduction, nous avons remplacé la mutation auto-adaptative anisotrope utilisée dans la première série d'expériences avec la mutation logarithmique présentée dans le chapitre 3. Grâce à sa capacité à générer des grandes perturbations, l'opérateur de mutation logarithmique permet de produire quelques enfants plus éloignés de leurs parents que ceux produits par la mutation auto-adaptative (voir section 3.3.4, chapitre 3). Avec un peu de chance, une proportion des ces enfants peut appartenir au nouvel espace faisable, et aide ainsi le reste de la population à la rejoindre. Nous avons alors effectué des tests pour les 3 fonctions G3, G5 et G11 en utilisant la mutation logarithmique, avec $fact_\epsilon = 1.01$ et $fact_\epsilon = 1.05$. Le nombre de composantes mutées a été limité à 2 et l'ordre de précision à 10^{-16}. Le tableau 6.4 résume les résultats obtenus.

L'approche par ajustement dynamique est devenue plus performante avec l'utilisation de la mutation logarithmique, où la plus grande violation enregistrée pour les 3 cas test avec $fact_\epsilon = 1.01$ est de $1.69.10^{-8}$. Cependant, le degré de précision des solutions médianes des fonctions G3 et G5 s'est légèrement détérioré. Par contre, tous les essais ont été menés avec succès pour le cas de G11, où l'optimum a été localisé à chaque test, et la violation ne dépasse pas 3.10^{-11}.

Comme prévu, l'utilisation de la mutation logarithmique a permis de maintenir un certain taux de faisabilité de la population, grâce à une réapparition immédiate des faisables après chaque

Fonction	Opt.	Performance			Violation			
		Meill	Med	Moy	Min	Med	Moy	Max
		$fact_\epsilon = 1.01$						
G3	1	0.999974	0.99991	0.99989	2.29^{-11}	$2.68.10^{-10}$	$2.94.10^{-10}$	$5.94.10^{-10}$
G5	5126.49	5126.67	5163.06	5175.77	$5.77.10^{-10}$	10^{-8}	$9.67.10^{-9}$	$1.69.10^{-8}$
G11	0.75	0.75	0.75	0.75	$2.28.10^{-13}$	$1.33.10^{-11}$	$1.41.10^{-11}$	$2.99.10^{-11}$
		$fact_\epsilon = 1.05$						
G3	1	0.999835	0.99892	0.99845	0	0	0	0
G5	5126.49	5126.51	5147.89	5152.97	$5.86.10^{-14}$	$4.43.10^{-13}$	$3.07.10^{-12}$	$3.27.10^{-11}$
G11	0.75	0.75	0.75	0.75012	0	2.43^{-19}	$4.08.10^{-19}$	$2.05.10^{-18}$

TAB. 6.4 – *Résultats des cas test G3, G5 et G11 obtenus avec l'ajustement dynamique utilisant la mutation logarithmique, avec* $fact_\epsilon = 1.01$ *et* 1.05*. Les les 3 premières colonnes donnent, pour chaque fonction, la moyenne, la médiane et la meilleure performance, alors que les 4 dernières colonnes donnent la violation minimale, médiane, moyenne et maximale sur les 31 essais.*

réduction de $\epsilon(t)$. La figure 6.8 montre que le pourcentage de faisabilité médian de la population de G11 ne descend pas au dessous de 25%. Ceci a permis d'appliquer plus fréquemment l'opération de réduction, permettant ainsi d'atteindre des taux de violation très petits.

Cependant, avec l'augmentation de $fact_\epsilon$, l'utilisation de la mutation logarithmique ne permet plus de maintenir des faisables dans la population, vue la très petite taille $\mathcal{F}(t)$ atteinte à un certain moment de l'évolution. La figure 6.8 montre que le taux de faisabilité médian pour le cas test G11 s'annule à la génération 2480. Aucun faisable n'a pu être produit à la suite. Nous précisons que les résultats donnés dans le tableau 6.4 pour $fact_\epsilon = 1.05$ sont calculés à partir des dernières solutions faisables obtenues pour chaque essai. Ils montrent comment les valeurs de violation sont presque négligeables pour l'ensemble des fonctions, mais la performance de l'algorithme a diminué par rapport à celle observée avec $fact_\epsilon = 1.01$.

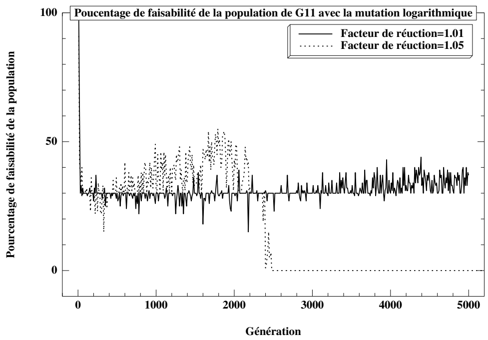


FIG. 6.8 – *Pourcentage médian (sur 31 essais) de faisabilité de la population au cours de la résolution de G11 avec l'approche par ajustement dynamique associée à la mutation logarithmique, pour* $fact_\epsilon = 1.01$ *et* $fact_\epsilon = 1.05$.

L'approche par ajustement dynamique a enregistré une grande robustesse dans la résolution d'un ensemble de cas test de référence soumis à des contraintes d'égalité. Cependant, elle ne tient pas compte de la faisabilité de la population après la réduction de $\epsilon(t)$. Au cours de la résolution des cas test G3, G5 et G11, l'algorithme a perdu tous les faisables à plusieurs moments de l'évolution. L'utilisation de la mutation logarithmique a permis de diminuer la fréquence d'annulation du degré de faisabilité de la population avec $fact_\epsilon = 1.01$, mais elle augmente avec l'augmentation de $fact_\epsilon$. Par contre, la précision de la performance des solutions obtenues avec la mutation logarithmique est inférieure à celle des solutions obtenues avec la mutation auto-adaptative. En effet, la capacité de la mutation auto-adaptative anisotrope de raffiner les meilleures solutions est supérieure à celle de la mutation logarithmique.

Dans le but d'assurer un taux de faisabilité minimal de la population indépendamment de la stratégie de mutation utilisée, nous proposons d'ajuster $\epsilon_j(t)$ selon un schéma adaptatif.

6.3.3 Ajustement Adaptatif

L'approche par ajustement adaptatif nécessite le choix de deux paramètres τ_{reduct} et $\tau_{equality}$. Ces deux paramètres décident du taux d'adaptation des valeurs de $\epsilon_j^+(t)$ et $\epsilon_j^-(t)$. Afin de chercher la meilleure combinaison possible permettant en même temps une violation minimale et une localisation précise de l'optimum global, nous avons effectué des expériences sur la fonction G11 en utilisant plusieurs valeurs. Celles de τ_{reduct} varient entre 0.6 et 0.9 et celles de $\tau_{equality}$ varient entre 0.2 et 0.8. Nous rappelons que la valeur de τ_{reduct} doit être supérieure à τ_{target} (degré de faisabilité minimal pour diminuer la pénalisation), afin d'éviter le risque d'avoir des cœfficients de pénalisation continuellement croissants. 31 essais ont été effectués pour chaque configurations. Les paramètres ϵ^+ et ϵ^- ont été initialisés à 0.001 et -0.001 respectivement.

Les figures 6.9 et 6.10 illustrent en dimension 3 les résultats de ces expériences. La première figure donne, pour chaque valeur de τ_{reduct} testée, la violation médiane en fonction des générations et de $\tau_{equality}$. On remarque que pour toutes les configurations, la violation médiane varie entre 10^{-5} et 10^{-20}. La violation minimale est de 0, et elle a été atteinte avec toutes les configurations, exceptées celles où $\tau_{reduct} = 0.9$ associée à $\tau_{equality}$ variant entre 0.5 et 0.8. La violation maximale a été enregistrée avec $\tau_{reduct} = 0.9$ et $\tau_{equality} = 0.7$, et elle est de $3.1.10^{-4}$. Avec des telles valeurs, la violation des contraintes est suffisamment petite pour être négligée.

Pour toutes les expériences, l'algorithme trouve une solution ayant une performance très proche de celle de l'optimum ($\simeq 0.75$). Cependant, le degré de précision diffère d'une configuration à une autre, selon les valeurs de τ_{reduct} et $\tau_{equality}$. La figure 6.10 illustre la distance euclidienne médiane de la meilleure solution à l'optimum en fonction de $\tau_{equality}$ et des générations, et ceci pour chaque valeur de τ_{reduct} testée. Toutes les distances médianes varient entre 10^{-3} et 10^{-6}. Les meilleures sont celles données par $\tau_{reduct} = 0.9$ avec $\tau_{equality} = 0.8$, où la médiane est de 10^{-5}. Les distances minimales se rapprochent de 10^{-6}, une valeur atteinte par la majorité des configurations.

Pour décider de la meilleure combinaison de τ_{reduct} et $\tau_{equality}$, il faut prendre en compte en même temps la violation et la distance à l'optimum. Les meilleures valeurs sont celles qui permettent de s'approcher le plus possible de l'optimum tout en violant le moins possible la contrainte d'égalité.

Si on considère les résultats de l'étude expérimentale de G11, les meilleures performances sont données par $\tau_{reduct} = 0.9$, mais les violations minimales sont données par $\tau_{reduct} = 0.6$. Avec $\tau_{reduct} = 0.6$, l'opération de réduction de l'espace faisable est plus fréquente qu'avec $\tau_{reduct} = 0.9$, ce qui explique la différence des violations médianes données par les deux taux à la fin de l'évolution, présentées dans la figure 6.9.

Une réduction très rapide de $\mathcal{F}(t)$ ne donne pas à l'algorithme le temps nécessaire pour bien explorer l'espace faisable avant de l'ajuster. Par contre, un grand intervalle de temps entre deux

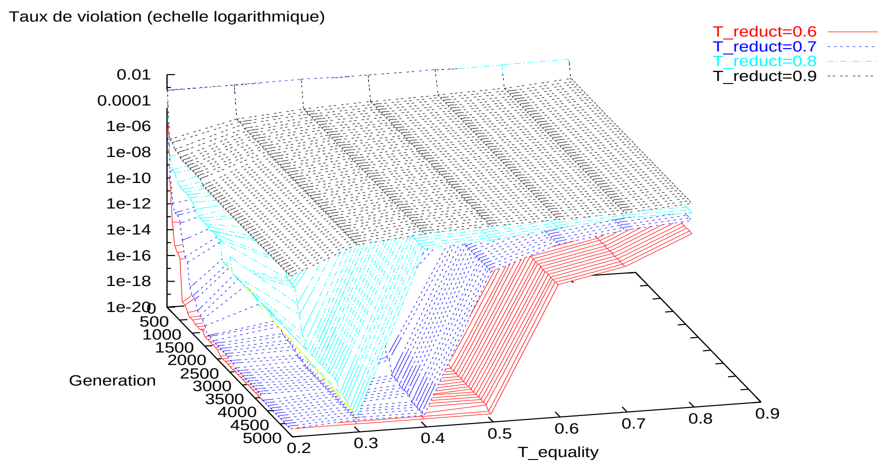


FIG. 6.9 – *G11: Courbes tridimensionnelles des taux de violation médians (sur 31 essais) en fonction des générations et de* $\tau_{equality}$*, pour les différentes valeurs de* τ_{reduct} *testées (*0.6, 0.7, 0.8, *et* 0.9*).*

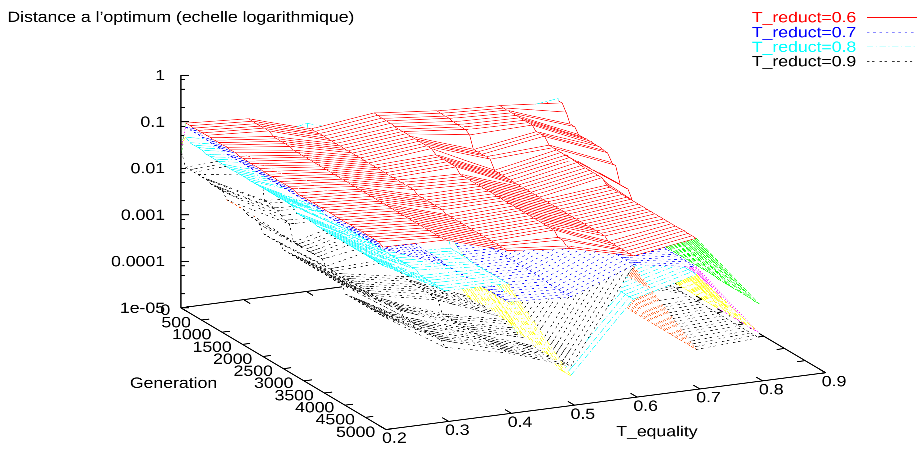
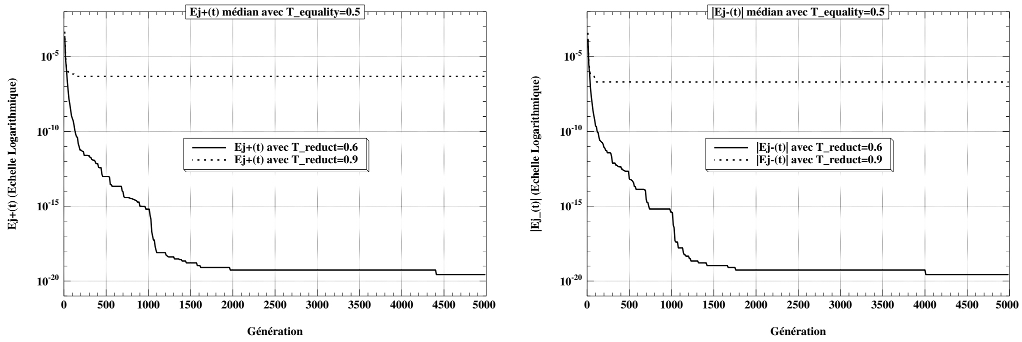


FIG. 6.10 – *G11: Courbes tridimensionnelles des distances médianes (sur 31 essais) des meilleures solutions à l'optimum global, enregistrées avec les différentes valeurs de* τ_{reduct} *et de* $\tau_{equality}$ *testées.*

opérations d'ajustement, même s'il permet à l'algorithme de bien visiter les différentes régions de $\mathcal{F}(t)$, risque de causer une convergence prématurée ou de donner des violations élevées à la fin de l'évolution.



(a) G11: $\epsilon_j^+(t)$ (b) G11: $|\epsilon_j^-(t)|$

FIG. 6.11 – *Courbe des valeurs médianes de $\epsilon_j^+(t)$ (a) et $|\epsilon_j^-(t)|$ (b) sur 31 essais pour le cas test G11 pour $\tau_{reduct} = 0.6$ et 0.9 avec $\tau_{equality} = 0.5$. La variable $\epsilon_j^-(t)$ a été transformée en sa valeur absolue afin de pouvoir la présenter à l'échelle logarithmique.*

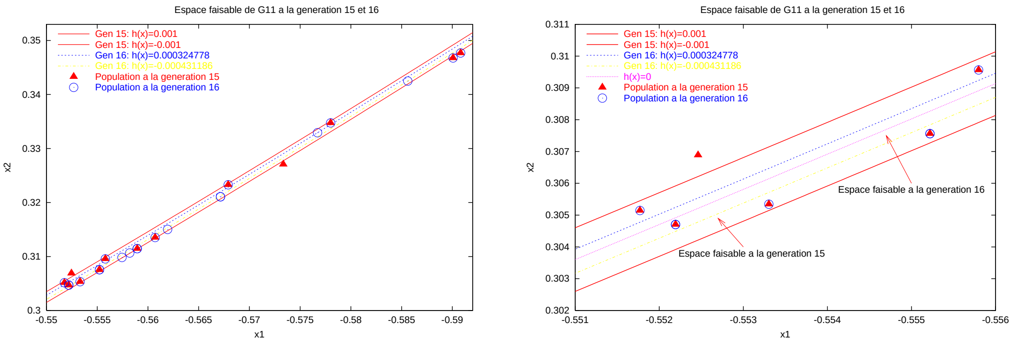
La figure 6.11 montre la différence entre l'évolution des valeurs de $\epsilon_j^+(t)$ (courbe (a)) et de $\epsilon_j^-(t)$ (courbe (b)) obtenues avec $\tau_{reduct} = 0.6$ et $\tau_{reduct} = 0.9$, pour $\tau_{equality} = 0.5$. On y voit que l'intervalle de faisabilité obtenu avec $\tau_{reduct} = 0.9$ est largement supérieur à celui obtenu avec $\tau_{reduct} = 0.6$. La première configuration a permis d'enregistrer les meilleures distances à l'optimum (figure 6.10), mais, pour quelques essais, la performance de la meilleure solution est meilleure que celle de l'optimum connu. En effet, la meilleure performance obtenue sur les 31 essais est de 0.749918, alors que la performance optimale est de 0.75. En fait, le meilleur individu a réussi de s'approcher de très près de la localisation de l'optimum, mais du côté du domaine infaisable (qui est considéré comme faisable avec les valeurs utilisées de $\epsilon_j^+(t)$ et de $\epsilon_j^-(t)$).

Ainsi, pour le choix de la meilleure configuration, nous prendrons en compte d'abord le taux de violation, ensuite la distance à l'optimum. La valeur 0.9 pour le paramètre τ_{reduct} ne sera pas retenue, vue la valeur élevée de la violation médiane enregistrée avec l'ensemble des configurations utilisant cette valeur. Entre autres, $\tau_{reduct} = 0.6$ n'a pas permis à l'algorithme de bien s'approcher de l'optimum, malgré qu'elle a donné les plus petites violations, à cause de la grande vitesse de réduction de l'espace faisable (figure 6.11). C'est donc parmi les valeurs 0.7 et 0.8 que sera choisie la bonne approximation de τ_{reduct}.

En considérant simultanément les taux de violation, les distances à l'optimum et les performances des meilleures solutions, c'est la configuration composée de $\tau_{reduct} = 0.7$ et $\tau_{equality} = 0.3$ qui a donné les meilleurs résultats. Nous supposons alors que les valeurs 0.7 et 0.3 constituent les meilleures approximations pour, respectivement, le paramètre τ_{reduct} et le paramètre $\tau_{equality}$ de l'approche par ajustement adaptatif.

Tous les résultats présentés dans la suite de ce chapitre sont obtenus avec les valeurs retenues de τ_{reduct} (0.7) et $\tau_{equality}$ (0.3).

La figure 6.12 montre un exemple de réduction de l'espace faisable de G11 entre les générations



(a) (b):Zoom

FIG. 6.12 – *Exemple d'évolution de l'espace faisable et quelques individus de la population entre les générations 15 et 16 au cours d'un essai de résolution de la fonction G11 avec l'approche par ajustement adaptatif.*

15 et 16 au cours d'un essai de résolution avec l'ajustement adaptatif. La borne supèrieure du domaine ($h(x) = \epsilon^+(15)$) et la borne inférieure ($h(x) = \epsilon^-(15)$) ne sont pas ajutsées de la même manière. Le pas de déplacement de la borne supèrieure est plus grand que celui de la borne infèrieure. Pour le ce cas de cet essai, cette différence est due au fait que le nombre d'individus appartenant à la zone limitée par les droites $h(x) = \epsilon^+(15)$ et $h(x) = 0$ est supèrieur au nombre d'individus appartenant à la zone limitée par les droites $h(x) = \epsilon^-(15)$ et $h(x) = 0$. En effet, il y avait 48 individus dans la première zone alors qu'il n'y avait que 32 dans la deuxième. Cette différence peut être due aussi à l'emplacement des individus dans chaque zone. Si les individus d'une des zones sont plus proches de la surface $h(x) = 0$ (doite sur la figure 6.12), alors le pas d'ajustement de la borne correspondante est plus grand. Dans les deux cas, l'ajustement des bords se fait de telle façon qu'on garde une proportion $\tau_{equality}$ d'individus dans chaque zone. Ainsi, un taux de faisabilité minimal est assuré au cours de l'évolution. Contrairement à l'approche par ajustement dynamique, le taux de faisabilité ne s'annule jamais (figure 6.13). L'objectif de la sélection ségrégationnelle d'ASCHEA est alors assuré, et la valeur du cœfficient de pénalisation oscille selon le phénomème observé au chapitre 5. Ceci permet d'accomplir le but d'ASCHEA qui est le maintient de la diversité de la population. Par conséquent, la procédure de réduction progressive de l'espace de faisabilité des contraintes d'égalité avec l'approche par ajustement adaptatif ne nuit pas au bon fonctionnement des autres composantes d'ASCHEA.

Résultats pour G3 et G5

Comme pour l'approche par ajustement dynamique, nous avons étendu l'étude expérimentale de l'appoche par ajustement adaptatif sur les cas test G3 et G5. Les paramètres de τ_{reduct} et $\tau_{equality}$ ont été fixés à, respectivement, 0.7 et 0.3, les valeurs retenues pendant la première étude de cette approche. Les paramètres ϵ_j^+ et ϵ_j^- pour le cas test G5 ont été initialisés à 0.1 et -0.1, vue la difficulté de trouver des faisables pour des valeurs plus petites. Par contre, pour le cas test G3, ils gardent les même valeurs initiales que celles utilisées pour le cas test G11 (0.001 et -0.001).

Le récapitulatif des résultats obtenus est donné par le tableau 6.6. Les meilleurs résultats sont ceux de G11, où la moyenne des 31 essais n'est qu'à 0.33% de l'optimum exacte, et la violation

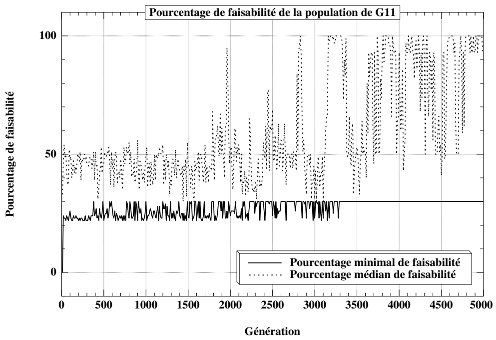


FIG. 6.13 – *Pourcentage minimal et médian (sur 31 essais) de faisabilité de la population au cours de la résolution de G11 avec l'approche par ajustement adaptatif.*

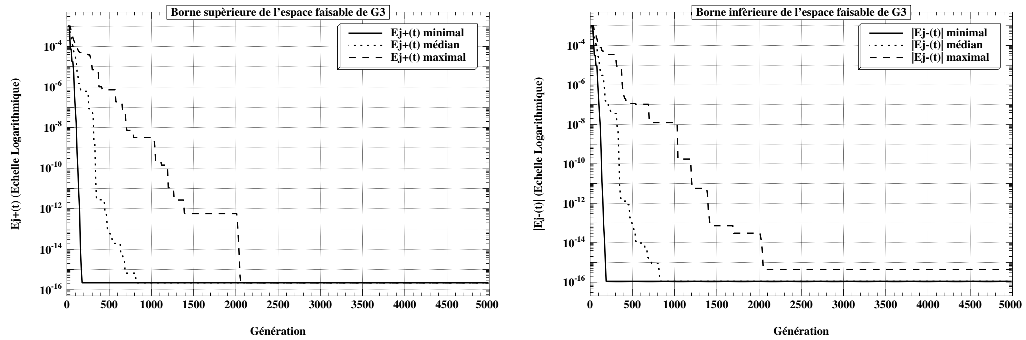
médiane n'est que de $5.42.10^{-20}$.

Fonction	Opt.	Performance			Violation			
		Meill	Med	Moy	Min	Med	Moy	Max
G3	1	0.999198	0.9335	0.8925	0	$2.22.10^{-16}$	$1.8.10^{-16}$	$2.22.10^{-16}$
G5	5126.49	5126.53	5159.19	5159.24	$1.1.10^{-6}$	$2.28.10^{-4}$	$4.5.10^{-3}$	0.0412
G11	0.75	0.75	0.75	0.7525	0	$5.42.10^{-20}$	$1.87.10^{-11}$	$4.46.10^{-11}$

TAB. 6.5 – *Résultats des cas test G3, G5 et G11 donnant, pour chaque fonction, la moyenne la médiane et la meilleure performance (les 3 premières colonnes), ainsi que la violation minimale, médiane, moyenne et maximale sur les 31 essais (les 4 dernières colonnes), obtenus avec l'ajustement adaptatif.*

Contrairement à ce qui est attendu, la robustesse de l'approche par ajustement adaptatif observée pendant la résolution de la fonction G11 n'est plus observée pour le cas de G3. Les résultats sont d'une qualité moyenne, où l'erreur entre la performance médiane et l'optimum exacte est de 6.65%. La dégradation de la capacité d'exploration d'ASCHEA pour ce problème est due essentiellement à la réduction très rapide de l'espace de faisabilité, limitant le temps permis pour la recherche dans le $\mathcal{F}(t)$ courant. La figure 6.14 illustre l'évolution des valeurs de $\epsilon_j^+(t)$(courbe (a)) et $\epsilon_j^-(t)$ (courbes (b)) pour G3. La valeur médiane atteint de très petites valeurs (10^{-16}) trop tôt dans l'évolution (avant la génération 1000), entraînant ainsi la population vers une convergence prématurée.

Pour ajuster la vitesse de diminution de la valeur de $\epsilon(t)$, une possibilité est d'augmenter la valeur du paramètre $\tau_{equality}$. Nous avons alors effectué des tests avec $\tau_{equality} = 0.8$ et $\tau_{equality} = 0.9$. Cette augmentation a permis d'améliorer les résultats, mais ils n'atteignent pas encore la qualité demandée. Avec $\tau_{equality} = 0.8$, la performance médiane s'est améliorée en passant de 0.9335 à 0.9866, tout en gardant des très faibles violations. Cependant, la vitesse de réduction de $\mathcal{F}(t)$ reste élevée, contrairement à ce qui est observé pour le cas de G11, où la valeur 0.8 n'a pas réussi à réduire suffisamment les violations. Nous avons alors effectué d'autres tests avec $\tau_{equality} = 0.9$. Cette valeur a permis d'améliorer remarquablement la performance médiane, mais la performance de la meilleure solution obtenue est meilleure que celle de l'optimum. En effet, avec $\tau_{equality} = 0.9$, la violation maximale s'élève à 3.10^{-4} alors qu'elle n'est que de 2.10^{-12} avec $\tau_{equality} = 0.8$ et



(a) G3: $\epsilon_j^+(t)$ (b) G3: $|\epsilon_j^-(t)|$

FIG. 6.14 – *Courbe des valeurs minimales, médianes, et maximales de $\epsilon_j^+(t)$ (a) et $|\epsilon_j^-(t)|$ (b) sur 31 essais pour le cas G3. La variable $\epsilon_j^-(t)$ a été transformée en sa valeur absolue afin de pouvoir la présenter à l'échelle logarithmique.*

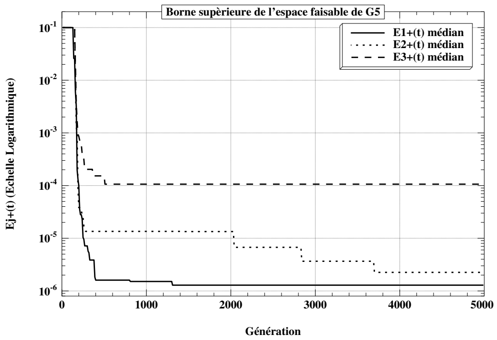
de $2.22.10^{-16}$ avec $\tau_{equality} = 0.3$. Nous déduisons de cette variabilité des résultats la dépendance de l'approche par ajustement adaptatif au paramètre $\tau_{equality}$. Ce dernier nécessite d'être ajusté pour chaque problème. Pour éviter le passage par cette étape, une solution est de demander à l'algorithme de l'ajuster automatiquement selon l'état courant de la population et les valeurs de $h_j(\overrightarrow{x})$. Une solution possible est de diminuer ou augmenter la valeur de $\tau_{equality}$ selon le taux de décroissance de $\epsilon_j(t)$ observé après chaque ajustement. Si ce taux est élevé, on augmente $\tau_{equality}$. Par contre, si la différence entre $\epsilon_j(t)$ et $\epsilon_j(t+1)$ est très petite, on diminue $\tau_{equality}$.

$\tau_{equlity}$	Performance			Violation			
	Meill	Med	Moy	Min	Med	Moy	Max
0.8	0.999837	0.986647	0.9760	0	$4.44.10^{-16}$	$1.28.10^{-13}$	2.10^{-12}
0.9	1.001	0.991999	0.989	$2.22.10^{-16}$	$6.66.10^{-14}$	$1.31.10^{-5}$	3.10^{-4}

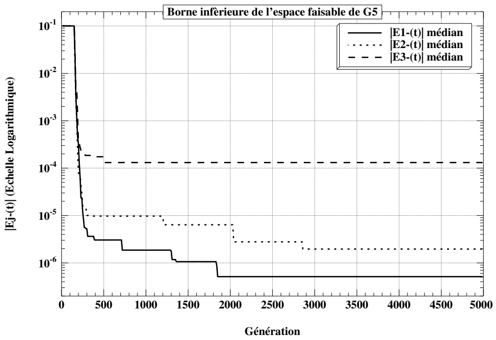
TAB. 6.6 – *Résultats du cas test G3 obtenus avec l'approche par ajustement adaptatif pour $\tau_{equality} = 0.8$ et $\tau_{equlity} = 0.9$.*

L'approche par ajustement adaptatif a eu aussi des difficultés dans la résolution de la fonction G5, qui est soumise à 3 contraintes d'égalité en même temps. Cette difficulté réside dans l'incapacité de la technique proposée de garantir un certain degré de faisabilité de la population pour les 3 contraintes simultanément, après l'ajustement de l'espace faisable de chacune. Pendant la mise à jour de ϵ_j, seule la contrainte j est prise en compte. Ainsi, le risque de perdre les individus respectant toutes les contraintes est élevé. C'est ce qui est passé pendant les 31 essais de résolution de la fonction G5, où le degré de faisabilité de la population τ_t devient nul après quelques centaines de générations, c'est à dire après plusieurs opérations de réduction des ϵ_j. Dans la majorité des cas, l'algorithme n'est plus capable de retrouver le chemin de l'espace faisable, dont la mesure est devenue très petite. Les résultats présentés dans le tableau 6.6 correspondent aux dernières solutions faisables trouvées.

Cependant, le degré de faisabilité de la population par rapport à chaque contrainte ne s'annule jamais, ce qui explique la continuation de la diminution des valeurs médianes des $\epsilon_j(t)$ présentées dans la figure 6.15, et ceci même après la perte totale des faisables.



(a) G5: $\epsilon_j^+(t)$



(b) G5: $|\epsilon_j^-(t)|$

FIG. 6.15 – *Courbes des valeurs médianes des* $\epsilon_j^+(t)\,(j = 1,2,3)$ *(a) et* $|\epsilon_j^-(t)|\,(j = 1,2,3)$ *(b) correspondant aux différentes contraintes d'égalité du cas test G5.*

Le fait de traiter chaque contrainte d'égalité séparément dans le cas de G5 a causé la division de la population en groupes, où chaque groupe tente de minimiser le taux de violation d'une des contraintes. Si la distance entre ces groupes est grande, la génération d'enfants appartenant à l'espace de faisabilité de toutes les contraintes devient très difficile.

Pour essayer d'éviter ce type de convergence et préserver la diversité de la population, dans le but de garder une répartition équilibrée des individus sur tout l'espace faisable courant, nous avons effectué des expériences en utilisant la procédure d'éclaircissement élitiste avec rayon adaptatif présentée dans la section 5.4 du chapitre 5. Le rayon d'éclaircissement a été initialisé à 0.1 et la capacité de chaque niche est limitée à 4.

Performance			Violation			
Meill	Med	Moy	Min	Med	Moy	Max
5126.5	5140.53	5187.31	$2.1.10^{-11}$	$1.44.10^{-5}$	2.10^{-3}	0.025

TAB. 6.7 – *Résultats du cas test G5 obtenus avec l'approche par ajustement adaptatif appliquée avec la procédure d'éclaircissement élitiste à rayon adaptatif.*

L'utilisation de l'éclaircissement a permis à ASCHEA de retourner une solution faisable pour 23 essais parmi les 31 réalisés (à comparer avec 0 essais sans le nichage). Pour les autres tests, il ne retrouve plus le chemin vers l'espace faisable qu'à une phase assez avancée de l'évolution, généralement après avoir dépassé les 4000 générations. Les résultats donnant les performances et les violations des dernières solutions faisables sont résumés dans le tableau 6.7. On remarque que l'éclaircissement a permis aussi d'améliorer la performance et la violation médiane. Toutefois, il n'annule pas la probabilité de perdre tous les faisables pendant l'évolution. Il est donc nécessaire d'améliorer l'approche par ajustement adaptatif pour garantir le maintient d'un certain degré de faisabilité par rapport à toutes les contraintes au cours de la réduction de l'espace faisable de chacune.

6.3.4 Discussion

La stratégie de prise en compte des contraintes d'égalité par réduction progressive de l'espace faisable, que ce soit par ajustement dynamique ou adaptatif, s'est avérée efficace, et constitue une direction de recherche prometteuse dans ce domaine. Les résultats préliminaires sont d'une qualité très satisfaisante, spécialement si on les compare avec des résultats donnés par d'autres méthodes de la littérature. En effet, la majorité des méthodes tolère une violation de 10^{-3} (e.g. la méthode de "Homomorphous mapping" (chapitre 2, section 2.5.3)), exceptée la méthode de classement stochastique (chapitre 2, section 2.3.7) qui tolère une violation maximale de 10^{-4}, et celle de de pénalités dynamiques (chapitre 2, section 2.3.3) qui a réussi à réduire l'erreur pour le cas test G5 à 10^{-4}.

Par contre, contrairement à ce qui était attendu, les résultats obtenus avec l'approche par ajustement dynamique, quand elle est appliquée avec des faibles facteurs de réduction, sont meilleurs que ceux obtenus avec l'approche par ajustement adaptatif, précisément pour les cas test G3 et G5. L'inconvénient de l'ajustement dynamique est le risque de perdre tous les faisables dans la population. Ce risque est réduit en utilisant la mutation logarithmique, mais la précision des solutions est meilleure avec la mutation auto-adaptative anisotrope. Pour remédier à ce problème, une solution possible est de combiner les deux types de mutation. Cette idée sera expérimentée prochainement.

La stratégie de la réduction de l'espace faisable adaptativement est très robuste, mais nécessite quelques améliorations. Elle réussit à maintenir la faisabilité de la population pour des problèmes soumis à une seule contrainte d'égalité, mais échoue en présence de plusieurs contraintes. La stratégie d'ajustement de la frontière de l'espace faisable de chaque contrainte doit être modifiée afin de prendre en compte la faisabilité de la population par rapport à toutes les contraintes simultanément.

Entre autres, elle présente une grande dépendance à ses paramètres τ_{reduct} et $\tau_{equality}$, qui contrôlent le taux d'ajustement de $\mathcal{F}(t)$. Nous pensons qu'il est possible de réduire cette dépendance en utilisant un $\tau_{equality}$ adaptatif. A chaque opération d'ajustement de $\epsilon(t)$, si la différence entre $\epsilon(t)$ et $\epsilon(t+1)$ est grande, alors on augmente $\tau_{equality}$. Par contre, si cette différence est très petite, on diminue $\tau_{equality}$. Ainsi, on peut contrôler le vitesse de décroissance de $\epsilon_j(t)$.

6.4 Conclusion

Nous avons présenté dans ce chapitre une méthode originale de prise en compte des contraintes d'égalité. Après la transformation classique de chaque égalité en un couple d'inégalité, définissant un espace de faisabilité $\mathcal{F}_{h_j}$, la taille de ce dernier est réduite progressivement au cours de l'évolution, permettant de minimiser en parallèle le taux de violation de la contrainte h_j correspondante.

Deux approches sont alors proposées. La première est basée sur un ajustement dynamique, où l'espace $\mathcal{F}_{h_j}$ est réduit d'un certain taux fixe chaque fois que le degré de faisabilité de la population pour la contrainte h_j atteint un certain seuil. Avec un faible facteur de réduction de $\mathcal{F}_{h_j}$, cette approche a donné des résultats très satisfaisants dans la résolution de 3 fonctions de référence. Cependant, à plusieurs moments de l'évolution, le degré de faisabilité de la population s'annule. Et avec des valeurs plus élevées pour le facteur de réduction, l'algorithme risque de perdre définitivement le chemin de l'espace faisable.

Pour éviter ce risque, nous avons proposé l'approche par ajustement adaptatif, où $\mathcal{F}_{h_j}$ est ajusté de telle façon qu'une certaine proportion des individus faisables gardent leur faisabilité après l'adaptation de $\mathcal{F}_{h_j}$. Cette approche a montré une grande robustesse dans la résolution d'un cas test soumis à une seule contrainte d'égalité, mais elle a échoué quand la fonction est soumise à plusieurs contraintes d'égalité simultanément, puisqu'elle ne prend en compte qu'une seule contrainte à la fois pendant l'opération d'adaptation.

Dans des prochains travaux, nous tenterons d'améliorer cette technique dans le but de prendre en compte l'ensemble des contraintes d'égalité pendant l'ajustement de l'espace faisable de chacune. La nouvelle frontière de $\mathcal{F}_{h_j}$ dépendra alors de la localisation des individus respectant toutes des contraintes. Le deuxième objectif de ces travaux est de permettre à l'approche par ajustement adaptatif d'ajuster automatiquement son paramètre $\tau_{equality}$.

Deuxième partie

Algorithmes Evolutionnaires: Application réelle

Chapitre 7

Optimisation d'une lame de phases pour un système laser

Résumé

Dans le but d'améliorer le processus de fusion nucléaire, les physiciens cherchent aujourd'hui à concevoir un système laser permettant de concentrer l'intensité sur la capsule élémentaire à irradier tout en ayant une distribution uniforme et une perte d'énergie minimale. Ceci revient à contrôler la forme du champ d'irradiation sur le plan focal du système optique.

Pour atteindre cet objectif, la *lame de phases*, un élément optique fondamental traversé directement par le faisceau laser, doit être façonnée d'une manière précise. La forme de ce composant est complexe et doit respecter certaines contraintes de fabrication.

Le but de cette application est de concevoir le profil bi-dimensionnel de la lame de phases permettant de rapprocher le profil d'intensité généré au plan focal d'une super gaussienne d'ordre élevé.

La simulation numérique de l'irradiation à travers le système optique incorporant certaines contraintes techniques requiert un double passage par la *Transformée de Fourier*, dont le coût computationnel dépend de la discrétisation choisie. Ainsi, dans le cas d'une approche paramétrique (lame de phases représentée par une matrice 2D $N \times N$), la complexité des solutions augmente quadratiquement avec N. Pour cette raison, et en parallèle de la représentation paramétrique, une représentation alternative est testée basée sur les arbres de la programmation génétique. Dans celles-ci, le génotype est indépendant de la discrétisation choisie.

Outre la description détaillée du choix de la configuration du problème ainsi que celle des deux approches testées, nous présentons une analyse des résultats obtenus. Ces derniers sont de très bonne qualité, en comparaison avec ceux donnés par d'autres techniques d'optimisation. Cependant, contrairement à ce qui était attendu, l'approche paramétrique a montré plus de robustesse dans la résolution de ce problème que l'approche fonctionnelle. Une étude comparative est présentée dans la dernière partie de ce chapitre.

7.1 Introduction

De nos jours, une des grandes directions de recherche dans la production d'énergie est de créer une nouvelle source d'énergie viable, non polluante et qui peut remplacer les autres sources polluantes comme le pétrole. Les efforts de certains chercheurs sont actuellement concentrés sur la simulation du processus de fusion qui est l'origine de l'énergie des étoiles, tel le soleil. Ce processus n'est, en fait, que la fusion d'éléments d'hydrogène, sous des conditions extrêmes de températures et de pression. Pour approcher ce mécanisme (sur terre), on a besoin d'une grande énergie qui stimule le processus de fusion au sein d'une capsule élémentaire. La source actuellement testée est celle de l'énergie laser.

Une des nécessités des circuits directs et indirects des systèmes lasers, en vue de favoriser la fusion, est une irradiation uniforme de l'énergie sur le foyer du système laser qui n'est autre que la capsule des atomes d'hydrogène à fusionner. La non uniformité des radiations est causée surtout par les variations spatiales dans le champ de lame des phases à travers laquelle passe le faisceau laser, et qui s'accumulent pendant la propagation du rayon à travers la lentille laser et l'air. Ce problème peut être réglé en changeant les propriétés de convergence de la lame de phases utilisée dans le système.

Entre autres, le processus de fusion ne se déclenche qu'avec des températures très élevées (de l'ordre de plusieurs dizaines de millions de degrés par une densité de $20g/cm^3$). Actuellement, avec la puissance du laser utilisé, pour atteindre une telle température, il faut maintenir le processus d'irradiation pendant un temps suffisant (la durée nécessaire pour que les éléments du combustible deviennent *plasma*[1]). D'où la nécessité de minimiser la perte d'énergie aux alentours de la capsule.

Il y a quelques années, les meilleures lames de phases qui ont été conçues avec des méthodes heuristiques permettaient une certaine uniformité de l'énergie sur la partie du plan focal à illuminer mais ne permettaient pas une convergence maximale de l'énergie. En effet, la perte d'énergie était estimée de 20% à 25% [25, 38, 39].
Les travaux les plus récents dans ce domaine, utilisant les techniques de *Recuit Simulé*, ont permis de réduire cette perte à moins de 10%, mais le temps de calcul en CPU reste élevé [76] (120h de temps CPU sur un ordinateur de type Cray YMP-2).

Dans ce travail, nous proposons une autre technique pour concevoir une lame de phase utilisant le paradigme évolutionnaire. Deux techniques ont été testées dans la résolution de ce problème: la première est basée sur une représentation paramétrique des inconnues alors que la deuxième est basée sur une représentation fonctionnelle. Ce travail a été réalisé en collaboration avec **Alain Racine**, Doctorant au C.M.A.P, spécialisé dans la programmation génétique. Ces travaux ont fait l'objet de deux publications [96] [108].

Nous commencerons d'abord par introduire rapidement la fusion nucléaire. Ensuite, nous détaillerons la modélisation et la formulation du problème, suivies par une illustration de l'ensemble des tests réalisés et de leurs résultats. La dernière partie de ce chapitre est une comparaison entre les deux approches, paramétrique et fonctionnelle, avec interprétation des résultats.

Pour le reste du chapitre, nous introduisons les notations suivantes:

- LPC: Lame de phases continues.
- FT: Transformée de Fourier (*Fourier Transform*). Par mesure de simplicité, on utilisera la même notation pour désigner la transformée de Fourier rapide (*Fast Fourier Transform*).

7.2 La Fusion Nucléaire

7.2.1 Le processus de fusion

La réaction de fusion est le mariage de noyaux légers qui donne naissance à des noyaux de taille moyenne (cf. figure 7.1), et s'accompagne d'une très forte libération d'énergie. Ce processus, à l'état

1. *plasma: peut être vu comme un 4^{eme} état de la matière: gaz ionisé*

naturel, existe dans les environnements extrêmement chauds que sont les étoiles, tel le soleil. Cette réaction est possible dans les étoiles grâce à l'état de grande agitation des noyaux (due à leur très haute température), ce qui permet aux atomes de même signe de se rapprocher et de franchir la *barrière répulsive* pour fusionner.

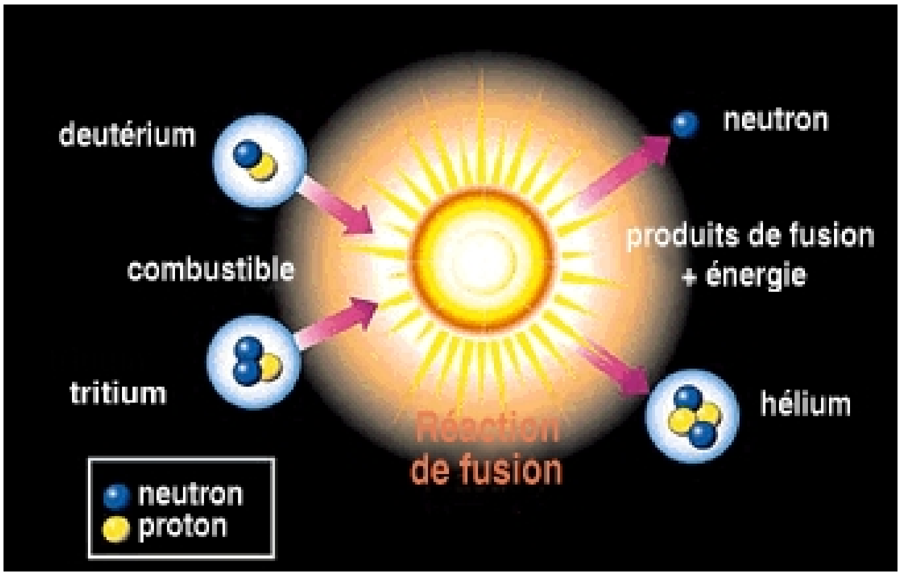


FIG. 7.1 – **La fusion nucléaire**.

7.2.2 La fusion sur Terre

L'homme cherche à maîtriser les réactions de fusion sur terre afin de récupérer cette fabuleuse énergie. Il a réussi à maîtriser celle-ci dans les bombes nucléaires de type H mais pas encore pour produire l'électricité. Pour une application civile de la fusion thermonucléaire, la réaction la plus étudiée est la fusion de deux noyaux d'isotopes d'hydrogène, le détérium et le tritium qui s'agglomèrent en donnant un noyau plus lourd, celui de l'hélium. Cependant, la source d'énergie (réacteur de fusion) est très difficile à réaliser. Un tel réacteur de fusion doit vérifier les trois conditions suivantes:

- une forte densité de particule (pour garantir un taux de collision élevé);
- une très haute température du plasma (sinon la vitesse des particules ne serait pas assez élevée pour permettre de franchir la *barrière de répulsion* électrostatique);
- un temps de confinement long (pour conserver le plasma chaud assez longtemps pour qu'une quantité suffisante de combustible fusionne).

Le confinement inertiel

Le confinement inertiel est une technique de confinement du plasma qui suppose la compression d'un grain de combustible par balayage de toute sa surface par des faisceaux lasers, pour que la compression et l'augmentation de sa température et de la densité de particule conduise à la fusion thermonucléaire. Le principe de ce processus est schématisé dans la figure 7.2.

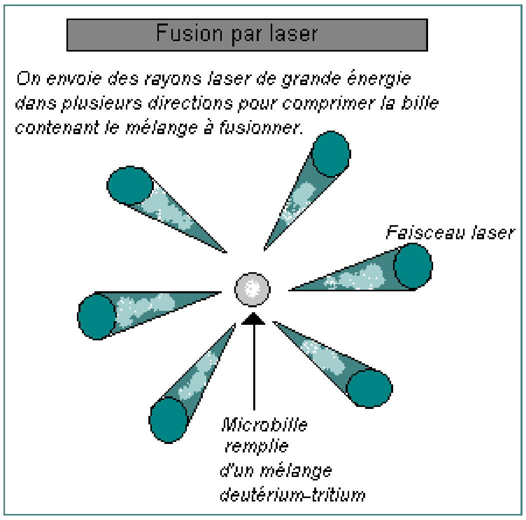


FIG. 7.2 – **Le confinement inertiel**.

La fusion par le laser est étudiée dans quelques laboratoires dans le monde. Pour certains, les pulses lasers sont prévus pour délivrer, au total, quelque 200kJ en moins d'une nanoseconde sur chaque grain de combustible. C'est une puissance délivrée d'environ 2 1014 W durant le pulse; c'est en gros 100 fois la capacité de puissance électrique mondiale.

Afin d'accomplir cet objectif, le système laser doit vérifier plusieurs conditions, dont:

- illuminer uniformément la particule,
- concentrer l'énergie sur la particule ($\simeq$ minimiser la perte d'énergie).

7.3 Modélisation du Système Optique

Considérons un faisceau laser défini par son champ électrostatique scalaire. Le problème consiste à concevoir une lame de phases continues (autrement dit une forme de lentille) pour produire le profil d'irradiation désiré dans le plan focal de la lentille convergente du dispositif optique.

Sans la lame de phases continues (LPC), le profil d'intensité dans le plan focal correspond à une fonction *sinus cardinal* et s'apparente donc à un pic d'intensité très énergétique au point focal (figure 7.3).

Dans le but d'induire le phénomène de fusion par confinement inertiel, il faut illuminer la cible se trouvant au plan focal de la lentille convergente à la fois intensément mais aussi uniformément. Ainsi, une LPC est ajoutée en aval de la lentille convergente de sorte que sa forme -à déterminer- modifie le profil d'irradiation.

Un système laser est déterminé essentiellement par 4 composantes:

- un diaphragme dont la pupille permet de calibrer le rayon incident,

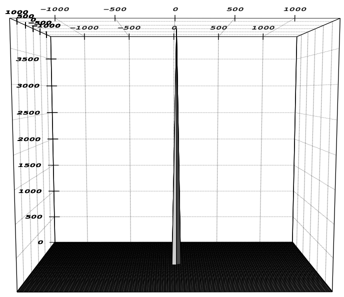


FIG. 7.3 – *Le profil du champ d'intensité au plan focal sans la LPC*

- une lame de phases continues qui permet de contrôler la convergence et l'uniformité du rayon laser,
- une lentille pour diriger le faisceau incident vers l'objet à illuminer,
- le plan focal comportant le point sur lequel on veut concentrer l'énergie laser et qui permet de mesurer le profil d'intensité.

Le figure 7.4 montre le schéma du dispositif expérimental.

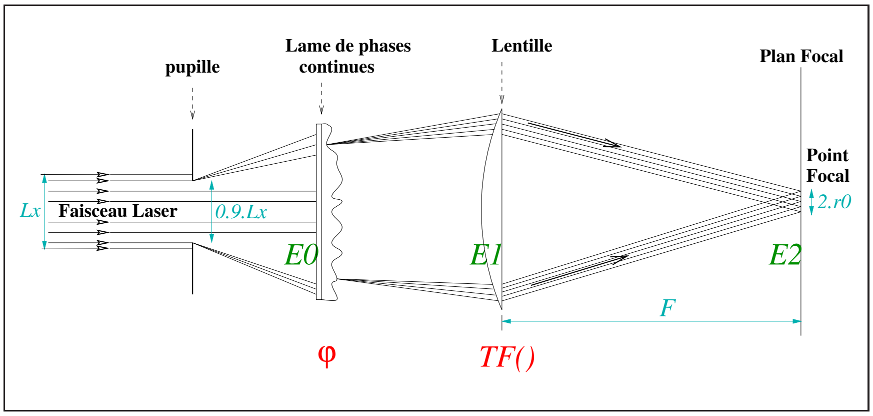


FIG. 7.4 – **Le système Laser**. *$L_x = 20cm$ est le diamètre du faisceau laser incident, $F = 8m$ est la longueur du foyer et $r_0 = 300\mu m$ est le rayon du point focal. La longueur d'onde du laser est de $350\eta m$*

7.3.1 Formulation Analytique

Dans un tel système, on considère 3 champs énergétiques qui se créent au fur et à mesure de la propagation du faisceau laser (voir figure 7.4):

- $E_0(x, y)$: champ électrostatique au niveau de la LPC,
- $E_1(x, y)$: champ électrostatique au niveau de la lentille convergente,
- $E_2(x, y)$: champ d'intensités sur le plan focal.

Le premier champ généré par le diaphragme est calculé comme suit (cf Annexe B pour plus de détails sur les calculs optiques):

$$E_0(x,y) = e^{-\pi\left(\frac{\sqrt{(x^2+y^2)}}{R}\right)^M} \tag{7.1}$$

où :

- R: diamètre du diaphragme ($R = 0.9 \times L_x$),
- M: indice de diffraction de la pupille du diaphragme ($M = 8$).

Le champ électrostatique généré par la LPC est le produit du champ E_0 par l'exponentiel de la lame $\varphi(x,y)$:

$$E_1(x,y) = E_0(x,y).e^{i\varphi(x,y)} \tag{7.2}$$

En fin, le champ électromagnétique E_2 produit sur le plan focal n'est autre que la transformée de Fourier du champ électrostatique sur la lentille (E_1):

$$E_2(x,y) = \mathrm{FT}(E_1(x,y)) \tag{7.3}$$

Finalement, l'intensité lumineuse relative au plan focal obtenue à partir du champ E_2 (c.f. Annexe B) est:

$$I(x,y) = (E_2(x,y))^2 = (\mathrm{FT}(E_1(x,y)))^2 \tag{7.4}$$

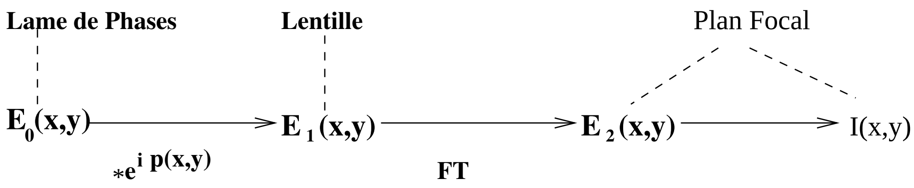


FIG. 7.5 – *Calcul de l'intensité lumineuse sur le plan focal*

7.3.2 Le Problème d'Optimisation

Le but de la conception d'une lame de phases est qu'une petite cible circulaire de rayon r_0 –bien plus large que le pic obtenu en absence de la LPC (figure 7.3)– placée dans le plan focal soit illuminée le plus uniformément possible par le faisceau laser (voir figure 7.6). Dans notre cas, le profil d'intensité désiré a la forme d'une *super gaussienne* de révolution d'ordre 8 centrée sur le point focal. Plus exactement, c'est l'intensité radiale moyenne (figure 7.6) du profil objectif bi-dimensionnel qui doit être proche de l'hyper-gaussienne (i.e. $t(r) = exp[-(\frac{r}{r_0})^8]$).

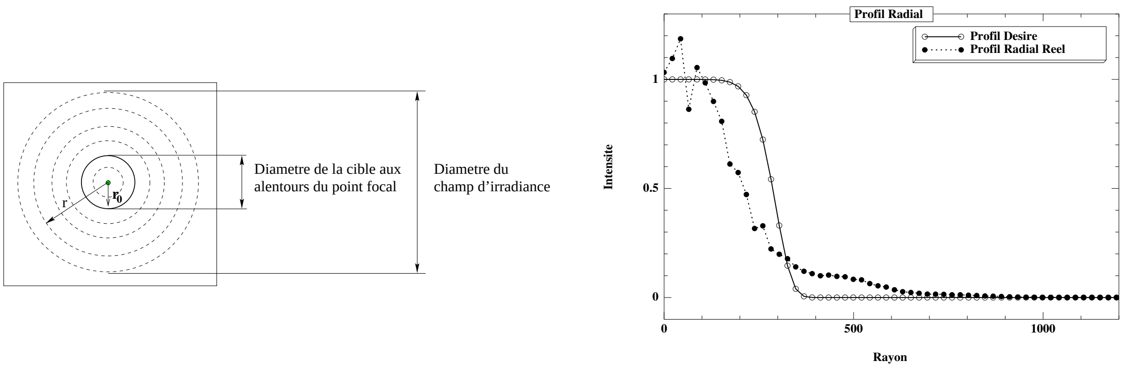
Le profil radial $\overline{I}(r)$ du champ d'intensité obtenu à partir de $I(x,y) \equiv I(r,\theta)$ est donc donné par:

$$\overline{I}(r) = \frac{1}{2\pi r}\int_0^{2\pi} I(r,\theta)\ d\theta$$

L'objectif est donc de déterminer une lame de phases dont le profil radial d'illumination au plan focal s'approche autant que possible d'une *super gaussienne.*

La fonction de coût à minimiser $\mathcal{E}rr$ est alors (après normalisation de $\overline{I}(r)$ en $\widetilde{I}(r)$):

$$\mathcal{E}rr = \int_0^\infty \left[t(r) - \widetilde{I}(r)\right]^2 dr \tag{7.5}$$



(a) La cible au plan focal. (b) Profils radiaux désiré et obtenu.

FIG. 7.6 – *Le problème d'optimisation: approcher le profil radial réel du profil désiré.*

7.3.3 Les Contraintes de Fabrication

La LPC à concevoir doit respecter certaines contraintes techniques de fabrication. En effet, les fabricants ne sont pas capables de produire des LPC aux formes trop rugueuses. Pour assurer un aspect lisse acceptable pour les différentes formes des lames de phases, une pratique générale est de filtrer les hautes fréquences spatiales de la lentille dans l'espace des fréquences, c'est à dire après l'application de la FT. Une transformée de Fourier inverse permet ensuite de ramener le signal filtré dans l'espace cartésien.

Ceci nous amène à considérer une nouvelle inconnue $Z(x, y)$, dans l'espace cartésien, et de calculer $\varphi(x, y)$ en filtrant les hautes fréquences dans $\tilde{Z}$ correspondant à Z après transformation vers l'espace des fréquences. Deux stratégies de détermination de la phase (φ) sont possibles:

– soit on part d'une lame de phases initiale "graduée" et continue (φ_0) à laquelle on rajoute un bruit filtré (Z); dans ce cas, l'objectif serait de déterminer le bruit Z:

$$1^{er} \text{ cas:} \quad Z(x,y) \underset{FT}{\longrightarrow} \tilde{Z} \times Filtre \underset{FT^{-1}}{\longrightarrow} \hat{Z}(x,y) + \varphi_0(x,y) \longrightarrow \varphi(x,y)$$

– soit de partir d'une solution aléatoire (Z) dont on optimisera la forme au cours de l'évolution:

$$2^{eme} \text{ cas:} \quad Z(x,y) \underset{FT}{\longrightarrow} \tilde{Z} \times Filtre \underset{FT^{-1}}{\longrightarrow} \varphi(x,y)$$

Dans les deux cas, pour que le rayon de corrélation soit toujours inférieur au rayon de la tâche focale, il est nécessaire de filtrer les hautes fréquences dues à l'aspect aléatoire des inconnues. Pour se faire, la variable $\tilde{Z}$ dans l'espace des fréquences est multipliée par un filtre, dont une partie des composantes sont nulles, ce qui permet d'annuler les fréquences à effet négative.

Le profil du filtre utilisé est illustré dans la figure 7.7, avec une exagération au niveau des coordonnées z pour assurer la visibilité. Avec un tel filtre, toutes les hautes fréquences dans de l'espace Fourier sont mises à zéro.

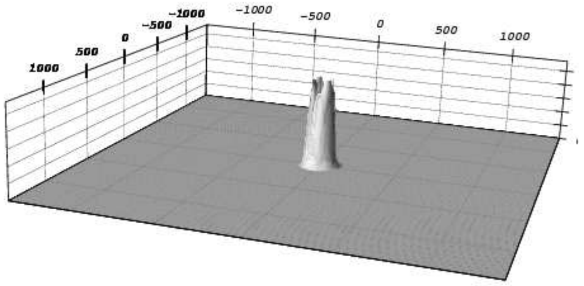


FIG. 7.7 – *L'allure du filtre utilisé pour annuler les hautes fréquences dans l'espace des fréquences.*

7.3.4 Discrétisation

Le but global de cette application est de trouver une bonne structure de la LPC qui permet de réaliser les objectifs décrits antérieurement. Dans le cas discret, ceci revient à trouver tous les éléments d'une matrice numérique (φ) représentant le masque de la lame de phases à calculer sur un domaine carré. La taille de cette matrice dépend du degré de discrétisation N adopté (32×32, 64×64, 128×128, 256×256, ...). Selon les modalités du calcul de φ données dans le paragraphe précédent, l'inconnue à déterminer est la matrice $Z(x, y)$ $(x, y \in \{1, 2, \cdots, N\})$.

La figure 7.6 indique les modalités de calcul de l'erreur en cas discret. Pour chaque rayon sur le plan focal, on détermine l'intensité radiale moyenne et on la compare avec la valeur optimale correspondante donnée par l'hyper-gaussienne (figure 7.6 (b)). La somme au carré des erreurs calculés constitue la fitness de la solution Z du départ.

Pour toute la suite, la mesure de la performance est donnée en pourcentage. Ainsi, si on considère L_f le rayon du champ d'irradiance sur le plan focal, alors la fitness devient:

$$\mathcal{E}rr = \sum_{r=0}^{r \leqslant L_f} \left[t(r) - \widetilde{I}(r) \right]^2 \times 100 \tag{7.6}$$

7.3.5 Domaine de recherche

La calcul de $\varphi(x, y)$ présenté dans la section 7.3.3 suggère une simplification de l'espace de recherche: au lieu de travailler sur la matrice inconnue Z dans l'espace cartésien, il est possible de travailler directement dans l'espace de Fourier et ne considérer que les variables non nulles de $\tilde{Z}$ après filtrage ($\tilde{Z} \times Filtre$). On évite ainsi d'optimiser toute une partie de la matrice $\tilde{Z}$ qui sera mise à zéro pendant le filtrage. Ainsi, on obtient des gains sur deux plans: sur le plan computationnel, puisqu'on évite la première FT, et sur le plan de la complexité puisqu'on a beaucoup moins d'inconnues à déterminer. En effet, le nombre de termes non nuls dans $\tilde{Z} \times Filtre$ est largement inférieur à celui de Z, puisqu'ils ont été, pour la majorité, annulés par le filtre.

Le nombre de valeurs non nulles ($nval$) dans $\tilde{Z} \times Filtre$ est indiqué dans le tableau 7.1 pour chaque valeur N de discrétisation. Ce tableau montre clairement comment $nval$, présentant le nombre de paramètres à optimiser, augmente quadratiquement avec le degré de discrétisation N. Donc, même si on ne considère pas le calcul de la performance, la complexité de l'algorithme, étant très sensible au nombre d'inconnus, augmente rapidement avec l'augmentation de N. Pour cette raison, on s'est limité à une discrétisation de 256, afin d'assurer la convergence du système. Par exemple, pour N=512, $nval$ devient 9112, ce qui représente un très grand nombre d'inconnues pour la capacité actuelle des machines.

$N = 32$	$\longrightarrow$	$nval = 32$
$N = 64$	$\longrightarrow$	$nval = 148$
$N = 128$	$\longrightarrow$	$nval = 560$
$N = 256$	$\longrightarrow$	$nval = 2284$

TAB. 7.1 – *Nombre des termes non nuls dans $\tilde{Z} \times Filtre$ pour différentes valeurs du paramètre de discrétisation N. $nval$ augmente quadratiquement avec la discrétisation N.*

7.3.6 La Fonction de Performance

A partir des considérations précédentes, on peut établir les modalités de calcul de la fonction de coût (cf. figure 7.8). Le génotype est l'ensemble des termes non nuls dans l'espace de Fourier. La matrice de Fourier filtrée est obtenue en remplissant les éléments d'une matrice par des zéros si l'élément de même rang dans le filtre est nul, sinon par la composante correspondante dans le génotype. La comparaison entre le profil radial obtenu sur le plan focal et le profil désiré est donné par l'équation 7.6.

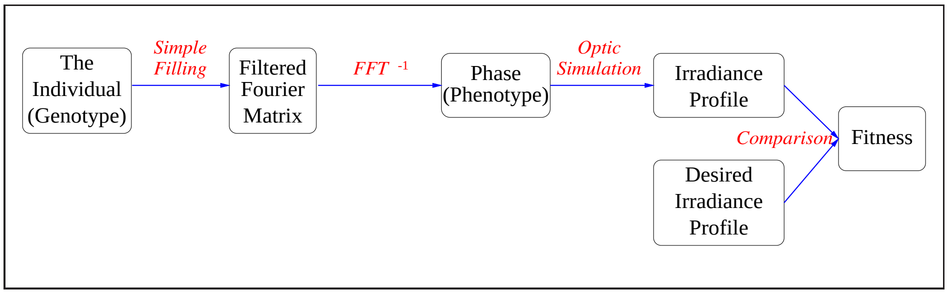


FIG. 7.8 – *Calcul de la fitness*

7.4 Choix de l'algorithme évolutionnaire

Vue la nature du problème (problème à variables réelles), nous avons pensé que les stratégies d'évolution avec auto-adaptation sont les algorithmes les plus adaptés à notre application.

Cependant, des tests ont tout de même été réalisés utilisant deux types d'algorithmes avec différentes configurations. Le premier type est un algorithme génétique réel standard, avec remplacement générationnel et une sélection par tournoi (chapitre 1, section 1.5.2). La mutation choisie pour le GA est la mutation Gaussienne avec ajustement de la déviation selon la règle 1/5 de Rechenberg [97].

Le deuxième type d'algorithme utilise les stratégie d'évolution avec la sélection $(\mu + \lambda)$-ES. Pour la mutation des variables, c'est la mutation Log-Normale de Schwefel [121] qui a été choisie, avec auto-adaptativité des déviations standards (chapitre 1, section 1.3.2).

Des séries d'expériences de 11 tests chacune ont été effectuées pour chaque algorithme avec deux configurations différentes: deux stratégies de mutation pour les ES (isotrope et anisotrope) et deux valeurs pour la taille du tournoi de l'opérateur de sélection des GA.

En ce qui concerne l'initialisation de la première population, chaque variable de chaque individu est générée aléatoirement dans l'intervalle $[-1500, 1500]$. Il est intéressant de noter que des expérimentations antérieures, avec un intervalle d'initialisation largement plus petit ($[-1, 1]$) ont révélé une convergence beaucoup plus lente de l'algorithme, spécialement au début de l'évolution, due aux très grands taux d'erreur de la population de départ [96].

La stratégie d'hybridation choisie est le croisement global discret (chapitre 1, section 1.3.1), appliqué aussi bien pour les variables x_i que pour les déviations standards σ_i pour le cas de la mutation auto-adaptative anisotrope [9]. Une autre stratégie de croisement a été testée une fois que le type d'algorithme évolutionnaire idéal est défini. Son principe et ses résultats sont présentés dans la deuxième partie de cette section.

L'ensemble des paramètres génétiques des deux types d'algorithmes définis pour l'expérimentation sont donnés dans les tableaux 7.2 et 7.3. Vu que ces expériences sont destinées au choix du type d'algorithme qui s'adapte avec notre problème, nous avons limité le degré de discrétisation à 64, afin de simplifier les tests. La probabilité de mutation pour le GA a été fixé à 0.1. Toutefois, des valeurs plus élevées ont été testées, mais les résultats correspondant ne sont pas présentés parce que leur qualité ne diffère pas beaucoup de celle des résultats présentés.

Les algorithmes génétiques ont trouvé beaucoup de difficulté à résoudre un tel type de problème. La figure 7.9, illustrant l'évolution de la performance moyenne donnée par les deux algorithmes, montre que l'évolution du GA se stabilise dès les premières générations. Le changement de la taille du tournoi n'améliore pas concrètement la performance de l'algorithme. La meilleure fitness donnée par le GA est de 5.363% avec le tournoi de taille 5, ce qui constitue un taux d'erreur très élevé et inacceptable pour notre problème. Cette mauvaise qualité est due en partie à la stratégie de remplacement des GA (remplacement générationnel), qui limite la durée de vie d'un individu à une génération. Si aucun des enfants générés n'a une performance meilleure que celle des parents, l'évolution commence à se stabiliser. Par contre, la sélection déterministe offre la possibilité au meilleurs individus de survivre pendant plusieurs générations, et de guider ainsi la population vers leurs zones. De plus, l'utilisation d'une déviations standard unique pour toute la population au cours de la mutation ne permet pas un ajustement précis des solutions, nécessaire pour ce type de problème.

Entre autres, la mutation auto-adaptative isotrope, malgré qu'elle montre une meilleure exploration de l'espace de recherche que celle anisotrope dans une première phase de l'évolution, elle

TAB. 7.2 – *Les paramètres de* **ES**

Taille de la Population: μ	50
Nombre d'enfants: λ	100
Mode de Sélection	$(\mu + \lambda)$
Les paramètres de mutation	
Type de mutation	1- Auto-adaptative isotrope
	2- Auto-adaptative anisotrope
Déviation standard initiale.: σ_0	0.3
Taux d'adaptation global : τ [2]	1
Taux d'adaptation local : τ_L [2]	0.7 (pour l'auto-adaptative anisotrope)
Probabilité de mutation	1
Les paramètres du croisement	
Type de croisement	discret global
Probabilité de croisement	1
Intervalle d'initialisation	$[-1500, 1500]$

[2] Ces paramètres sont modifiés selon le nombre *nval* de paramètres à optimiser [9].

TAB. 7.3 – *Les paramètres de* **GA**

Taille de la Population:	50
Mode de Sélection	tournoi
Taille	$\{5, 10\}$
Les paramètres de mutation	
Type mutation	Gaussienne adaptative (règle 1/5)
Déviation standard initiale.: σ_0	0.3
facteur de la règle 1/5:	1.1
Probabilité de mutation	0.1
Les paramètres du croisement	
Type de croisement	global discret
Probabilité de croisement	1
Intervalle d'initialisation	$[-1500, 1500]$
Type de remplacement	Générationnel

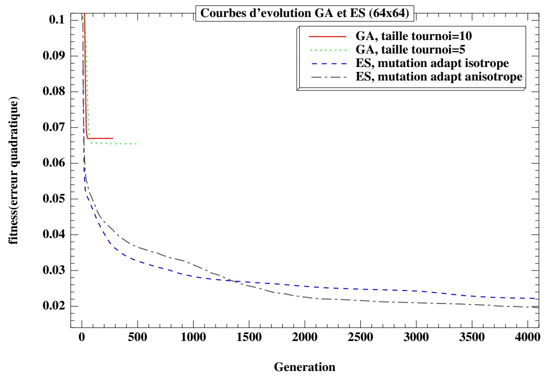


FIG. 7.9 – *Evolution de la moyenne (sur 11 tests) des meilleures performances données par GA et ES. L'opérateur de sélection par tournoi du GA a été testé avec les tailles 5 et 10, et ES avec les mutation auto-adaptatives isotrope et anisotrope.*

montre une convergence plus lente dans une phase plus avancée. Cette différence n'est pas très significative en observant la courbe 7.9. Nous avons alors effectué le test de Student afin de voir le taux de confiance de l'hypothèse de différence des deux séries d'expériences. La courbe 7.10 illustre l'évolution de la moyenne empirique de la différence observée. On remarque qu'au début de l'évolution les performances des deux opérateurs de mutation sont presque similaires, mais à partir de la génération 1800, les deux séries d'essais deviennent statistiquement différents avec une confiance de 95%, et qui atteint 99% à la génération 4200. Selon ce test, nous pouvons confirmé que l'opérateur de mutation auto-adaptative anisotrope est plus robuste que celui de la mutation auto-adaptative isotrope pour le cas de cette application.

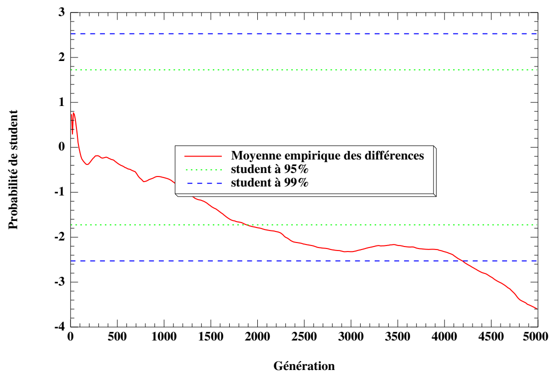


FIG. 7.10 – *Moyenne empirique des différences observées entre les séries d'essais effectuées avec les mutations auto-adaptatives isotrope et anisotrope, et les intervalles de confiance selon le test de Student.*

Suite aux observations déduites des expérimentations avec les deux types d'algorithme et les différentes mutations, ce sont les stratégies d'évolution avec mutation auto-adaptative anisotrope qui ont été retenues pour la suite de l'étude expérimentale dans ce travail.

Un autre opérateur de croisement a été aussi testé, basé sur un croisement *diagonal bi-dimensionnel*, qui est une généralisation du croisement 1-point dans le cas d'une seule dimension [66]. Dans ce cas, le génôme n'est pas considéré comme une liste linéaire de paramètres mais comme une matrice 2D. Deux points distincts sont choisis aléatoirement sur les deux parents, définissant la droite du croisement (équivalente au point de croisement en 1-D). Les enfants sont obtenus en permutant les deux parties séparées par la droite des deux parents [65] (figure 7.11).

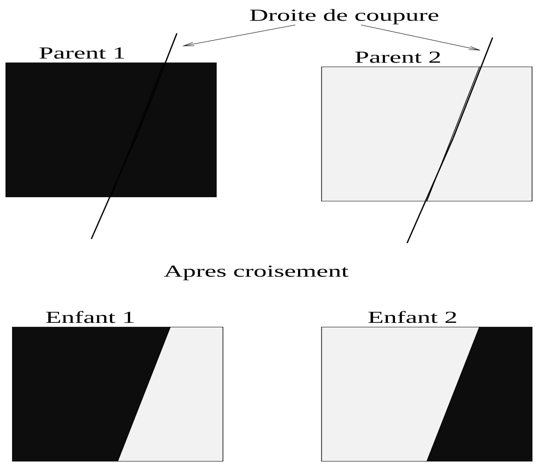


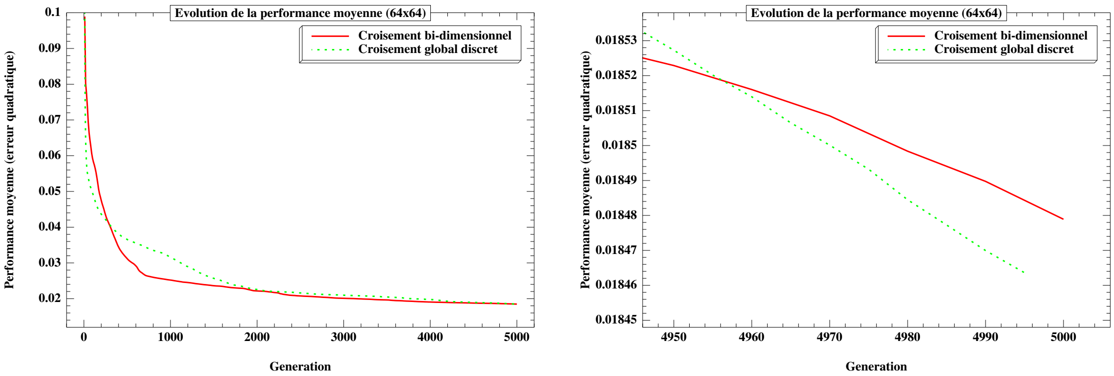
FIG. 7.11 – *Croisement diagonal bi-dimensionnel*

Appliqué aussi bien sur les variables du génotype que sur les déviations standard de la mutation auto-adaptative, cet opérateur a montré une grande robustesse, en favorisant une descente rapide du taux d'erreur, spécialement pendant la première phase de l'évolution (figure 7.12, courbe (a)). Toutefois, à la fin de l'évolution, cette stratégie n'a pas donné de meilleurs résultats que le croisement discret global. En effet, la moyenne des 11 tests obtenue avec le croisement global est légèrement meilleure que celle obtenue avec le croisement bi-dimensionnel (figure 7.12 courbe (b)). Cette différence n'est pas très significative, mais étant donné que le croisement global discret est plus simple à appliquer, il a été retenu pour la suite des expérimentations.

7.5 Choix de la phase initiale

Dans la section 7.3.3, nous avons vu que, pour chercher la forme d'une LPC optimale, deux choix se présentent: soit on part d'une forme aléatoire, soit on part d'une solution initiale φ_0. Pour le deuxième cas, il est nécessaire de définir l'allure de φ_0.

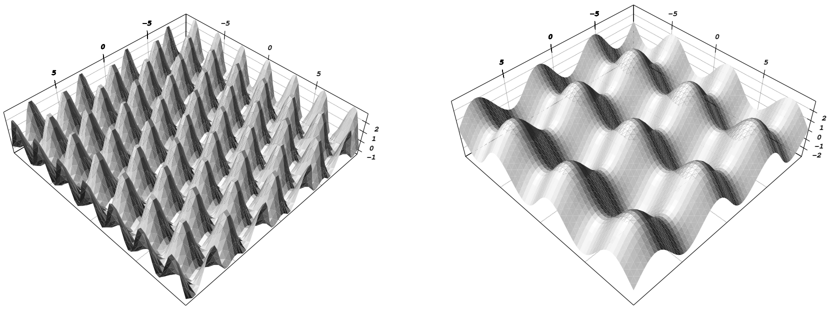
L'expérience préalable des physiciens dans ce domaine les a conduit à l'hypothèse que la forme optimale des lames de phases se rapproche d'une surface périodique continue à laquelle on ajoute un bruit déterminé. L'objectif de l'introduction des termes spatiaux périodiques est la diffraction du faisceau incident en plusieurs petits faisceaux. Ainsi, on n'a plus un seul point focal sur le champ de convergence de la lentille mais un ensemble de point focaux, ce qui empêche la formation du pic d'intensité présentée dans la figure 7.3 et garantit une distribution de l'énergie sur toute



(a) (b): Zoom

FIG. 7.12 – *Evolution de la moyenne (sur 11 tests) des meilleures performances données par ES avec le croisement global discret et le croisement diagonal bi-dimensionnel.*

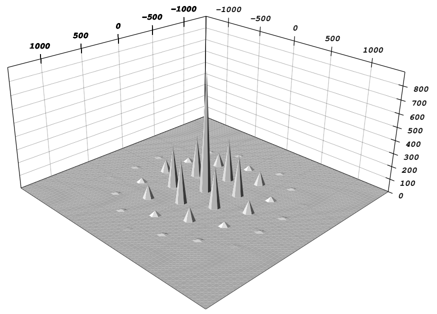
la tâche focale. Toutefois, les termes périodiques n'assurent pas une distribution uniforme et sans perte de l'énergie. D'où la nécessité d'introduire des composantes aléatoires (bruit) dans la surface de la phase, afin de contrôler le profil d'irradiation. Donc, la tâche de l'algorithme se résume en la détermination de ce bruit. En considérant cette hypothèse, on a décidé de partir de deux types de LPC initiales (φ_0): l'une est une superposition de 3 cosinus et l'autre est une superposition de 2 sinus (voir figure 7.13), et d'essayer, en parallèle, de partir d'une surface complètement aléatoire dont on optimisera la forme durant l'évolution.



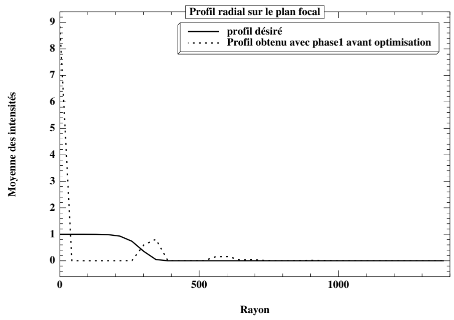
phase1: superposition de 3 cosinus phase2: superposition de 2 sinus

FIG. 7.13 – *Les formes des LPC initiales φ_0 choisies (avec N=64).*

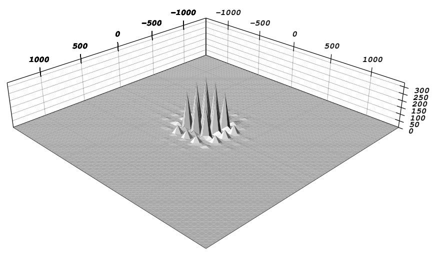
Le résultat du calcul du champ d'intensité au plan focal en suivant les étapes donnés dans le sous-paragraphe 7.3.1, et en partant de l'une des deux LPC initiales proposées, explique les fondements de l'hypothèse des physiciens. En effet, le pic d'intensité observé dans l'absence d'une lame de phases (figure 7.3) a disparu, et a été remplacé par une dispersion plus large de l'énergie. La figure 7.14 montre que la LPC initiale 1 (superposition de 3 cosinus) a permis de répartir l'énergie sur une grande partie de la tâche focale (figure 7.14 (a) et (b)), mais les pics d'énergie sont moins intenses avec la deuxième LPC initiale (superposition de 2 sinus) (figure 7.14 (b) et (d)). Les courbes (e) et (f) de la figure 7.14 illustrent un exemple de champs d'intensité donné par une phase générée aléatoirement. La répartition de l'énergie est moins uniforme qu'avec *phase1* et *phase2*, mais l'erreur est plus petite. On note que plus l'intervalle d'initialisation de ses éléments est petit, plus l'erreur initiale est élevée.



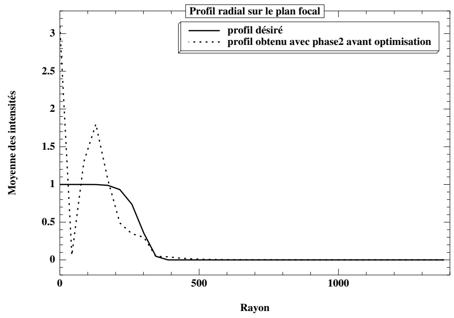
(a): Champs d'intensité obtenu avec *phase1*



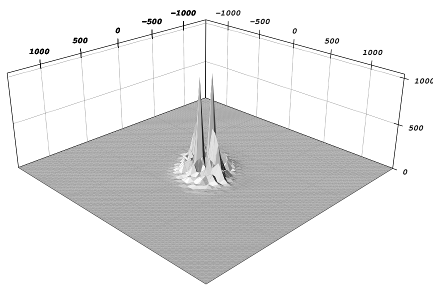
(b): Profil radial obtenu avec *phase1*
$erreur = 67665.5\%$



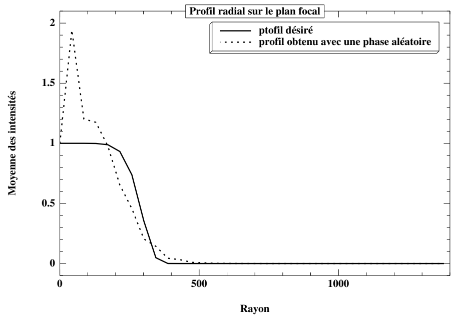
(c): Champs d'intensité obtenu avec *phase2*



(d): Profil radial obtenu avec *phase2*
$erreur = 638.46\%$



(e): Champs d'intensité obtenu avec une phase aléatoire



(f): Profil radial obtenu avec une phase aléatoire
$erreur = 114.97\%$

FIG. 7.14 – *Les profils des champs d'intensités au plan focal et les profila radiaux donnés par la phase1, la phase2 et une phase aléatoire avant l'évolution, avec N=64.*

Afin de décider du meilleur choix de φ_0, des tests comparatifs ont été réalisés, utilisant l'algorithme décrit dans la section précédente.

Pour cette deuxième phase expérimentale, la discrétisation choisie est toujours 64×64, afin de réduire le temps de calcul. Des discrétisations plus élevées seront testées une fois la bonne configuration (type de la LPC initiale) a été fixée.

11 essais ont été réalisés avec un nombre de générations limité à 5000 pour les trois configurations suivantes:

1. φ_0 est une superposition de *3 cosinus* $\longrightarrow \varphi = \varphi_0 + \hat{Z}$,
2. φ_0 est une superposition de *2 sinus* $\longrightarrow \varphi = \varphi_0 + \hat{Z}$,
3. Pas de LPC initiale $\varphi_0 \longrightarrow \varphi = \hat{Z}$.

7.5.1 Résultats obtenus avec les différentes configurations et comparaison

L'hypothèse donnée par les physiciens s'est avérée non valable quand on utilise les algorithmes évolutionnaires. En effet, l'algorithme a plus de difficulté de raffiner la surface d'une LPC donnée que de partir d'un ensemble aléatoire de lames et de l'optimiser. Cette observation est déduite des résultats donnés dans les figures 7.15 et 7.16 .

La courbe de droite de la figure 7.15 illustre la meilleure fitness enregistrée pour les 3 configurations. On constate que plus le degré de graduation (nombre de termes périodiques) de la surface de départ est élevé, plus l'algorithme a du mal à converger vers une solution acceptable. Avec la première configuration (superposition de *3 cosinus*), l'erreur reste très élevée ($\simeq$ 19%), et au bout de 500 générations, l'amélioration devient négligeable. Avec une surface initiale moins graduée (superposition de *2 sinus*), la performance est nettement meilleure, mais n'atteint toujours pas le seuil souhaité. Une erreur pareille (la meilleure performance enregistrée est de 9.88%) est inacceptable pour les physiciens du laser.

Cependant, avec la troisième configuration (phase de départ aléatoire), l'algorithme atteint des résultats particulièrement satisfaisants (performance inférieure à 2.5%) (courbe 7.15).

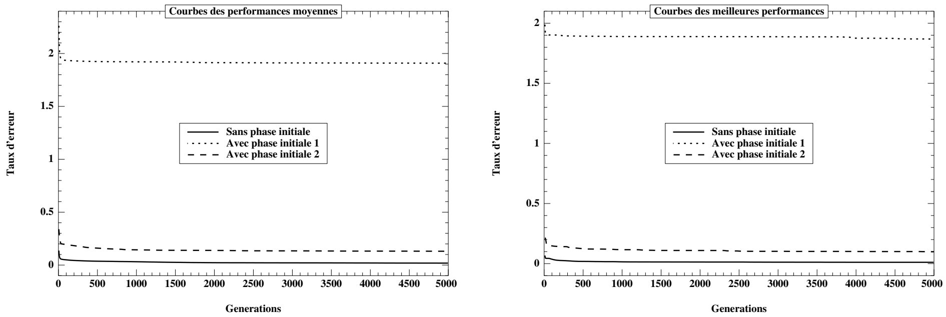


FIG. 7.15 – *Courbes des moyennes (à gauche) et des meilleurs (à droite) taux d'erreur donnés par l'algorithme en partant d'une superposition de 3 cosinus (phase1), d'une superposition de 2 sinus (phase2) ou d'une surface aléatoire (sans phase initiale).*

Les meilleurs profils radiaux sont présentés dans la figure 7.16. L'interprétation de cette courbe ne fait que confirmer les conclusions générées à partir des courbes précédentes. En effet, le profil donné par la première configuration (φ_0: *3 cosinus*) est d'une qualité médiocre. Il est tellement loin du profil désiré (*super gaussienne*) que le maintient de cette configuration pour le reste de notre étude ne peut pas être envisagé.

Par ailleurs, la comparaison entre la troisième configuration (sans phase initiale) et la deuxième configuration (φ_0: *2 sinus*) montre que les profils d'intensité résultants se rapprochent bien du profil désiré, mais celui donné par la deuxième est de meilleure qualité.

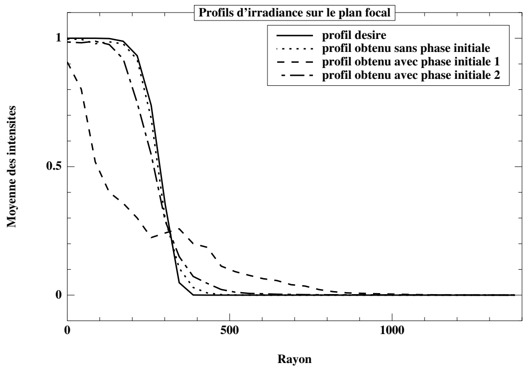


FIG. 7.16 – *Courbes des profils radiaux des meilleurs solutions données (sur 11 essais) avec les 3 configurations après 5000 générations d'évolution.*

Les profils des meilleures lames de phases trouvées par l'algorithme pour les 3 configurations sont présentés dans les figures 7.19 (pour la première configuration), 7.17 (pour la deuxième configuration) et 7.18 (pour la troisième configuration).

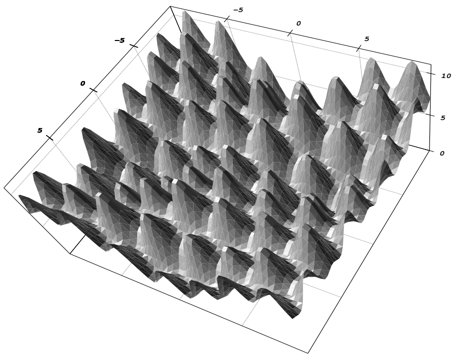


FIG. 7.17 – *Meilleure LPC trouvée avec la première configuration (phase initiale 1: superposition de* 3 cosinus*).*

La constatation qui peut être faite en observant les 3 lames de phases est que, pour une discrétisation égale à 64, plus la phase initiale est multimodale, plus l'algorithme a des difficultés à la raffiner et plus le profil d'intensité s'éloigne du profil désiré. En effet, avec une LPC de départ plus irrégulière, le bruit Z qui doit être ajouté pour contrôler les caractéristiques optiques de la

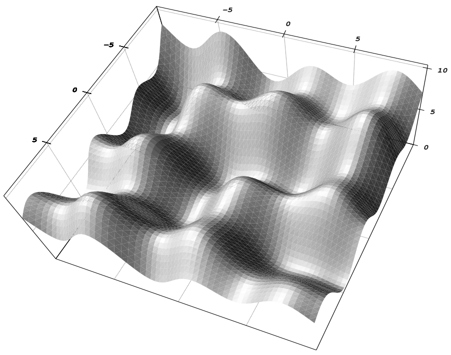


FIG. 7.18 – *Meilleure LPC trouvée avec la deuxième configuration (phase initiale 2: superposition de* 2 sinus*).*

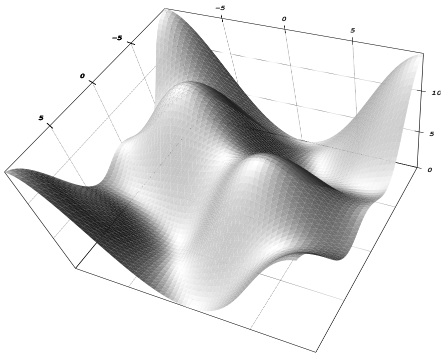


FIG. 7.19 – *Meilleure LPC trouvée avec la troisième configuration (sans phase initiale).*

LPC doit être précis et adapté à la forme de la surface donnée. Ainsi, l'augmentation du degré de complexité de φ_0 est accompagnée en parallèle par une réduction de l'espace de recherche de Z et d'un ralentissement de la vitesse de convergence de l'algorithme.

Pour toutes ces raisons, pour tout le reste des expérimentations, c'est la troisième configuration qui est maintenue: aucune LPC initiale prédéfinie.

7.6 Résultats obtenus pour différentes discrétisations

Une fois le choix de la configuration initiale a été fixé, d'autres expérimentations ont été réalisées pour les 4 discrétisations proposées (32, 64, 128 et 256). Comme dans la section précédente, pour chaque cas, 11 essais ont été effectués avec un nombre de générations limité à 5000.

Les résultats trouvés sont en général très satisfaisants. En effet, l'erreur varie entre 0.6526% et $2,494\%$ pour N=32 et 4% et 4.5% pour N=256. La courbe 7.20 illustre l'erreur moyenne, maximale et minimale enregistrées sur 11 tests après 5000 générations pour les différentes discrétisations. Nous ne présentons pas des discrétisations plus élevées (512,1024,...), parce que, à partir de 256, les calculs deviennent très coûteux (courbe 7.25). Par exemple, pour N=512, le temps moyen d'une génération est de 560 secondes. Avec un tel temps computationnel et avec la capacité actuelle des machines, nous ne pouvons pas lancer des expériences pour des grandes valeurs de N.

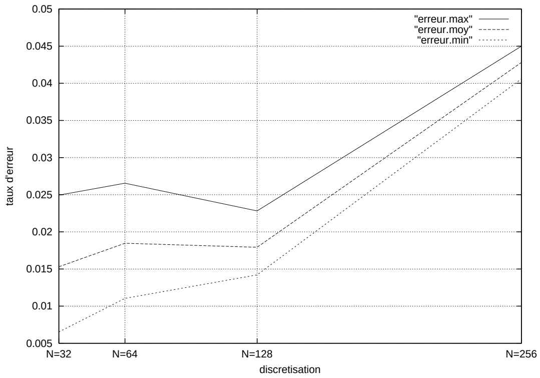
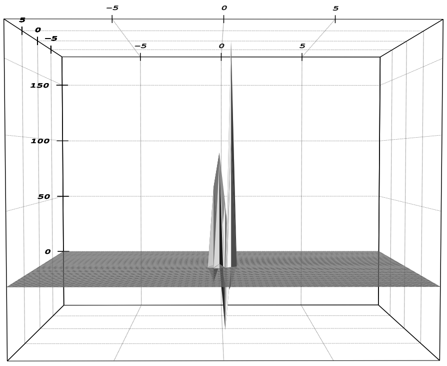


FIG. 7.20 – *Erreur moyenne, maximale et minimale pour N=32, 64, 128 et 256.*

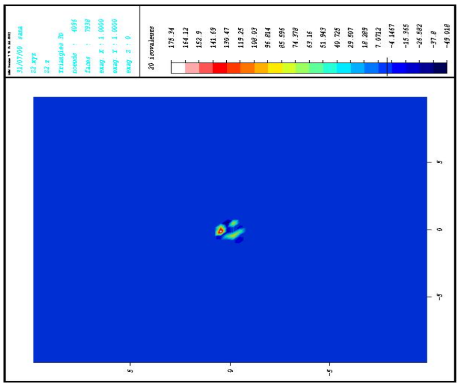
La figure 7.21 représente les détails des résultats correspondant à la meilleure solution obtenue avec N=64. En fait, la solution réelle donnée par ES n'est qu'un tableau de termes réels correspondant aux densités des éléments non nuls dans l'espace Fourier. Converti en une matrice 2D de taille $N \times N$, on retrouve la matrice $\tilde{Z} \times filtre$ illustrée par la courbe C1 de la figure 7.21.

En ramenant la solution vers l'espace cartésien avec une FT inverse, on obtient la LPC qu'on cherche à déterminer, représentée par la courbe C3. La courbe C4 montre l'allure de la LPC en iso-valeurs.

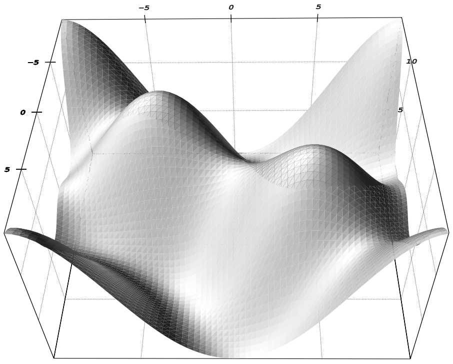
Le champ d'irradiation généré par φ sur le plan focal est illustré par la courbe C5. On voit clairement la concentration de l'énergie dans la tâche focale à illuminer (diamètre=$600\mu m$). Les différents pics d'intensité correspondent au différents faisceaux d'énergie diffractés par la lentille. Le nombre des faisceaux ainsi que la distance de séparation latérale entre les pics dépendent directement du nombre des termes périodiques de la LPC et de la longueur d'onde du laser utilisé.



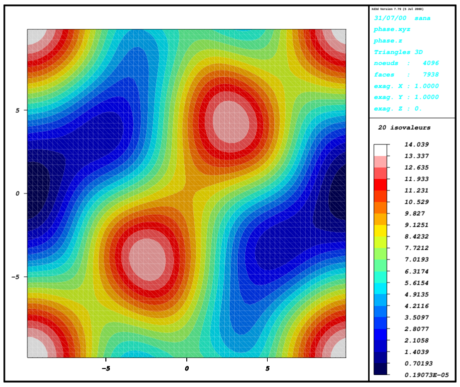
C1: le génotype (dans l'espace des fréquences)



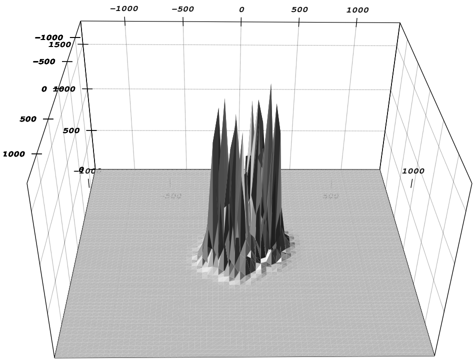
C2: Le génotype en iso-valeurs



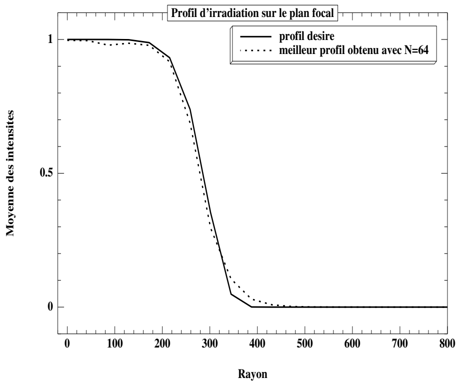
C3: Profil de la LPC obtenue



C4: La LPC en iso-valeurs (D=20cm)



C5: Le champ d'intensité sur la tâche focale



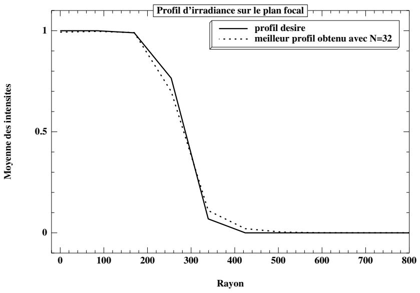
C6: Profils d'intensités obtenu et désiré

FIG. 7.21 – *Détails des résultats correspondant à la meilleure solution obtenue avec pour N=64, présentant le passage du génotype au calcul du profil d'intensité. C1 et C2 présentent le profil du génotype respectivement dans l'espace des fréquences et en iso-valeurs. Après application de la FT inverse, on obtient la LPC illustrée par C3 et C4 (en iso-valeurs). C5 présente le champ d'intensité donné par cette LPC, et C6 illustre le profil d'intensité calculé avec l'équation 7.4.*

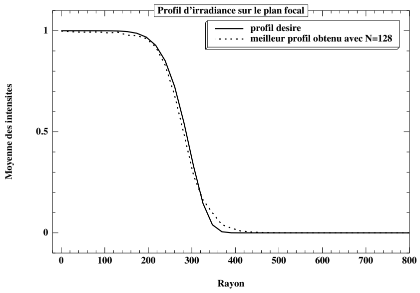
En faisant les moyennes radiales des intensités dans le plan focal, on obtient le profil d'intensité représenté par la courbe C6. A première vue, le profil d'irradiation obtenu semble se confondre avec la *super gaussienne*. Cependant, il déborde un peu de l'enveloppe désirée du côté inférieur de la courbe ($300\mu m < rayon \leqslant 400\mu m$).

En fait, plus de 99% de l'énergie est contenu dans l'enveloppe. L'erreur causée par un léger manque d'uniformité au niveau de la distribution d'énergie à l'intérieur de l'enveloppe ($rayon < 300\mu m$) ne pose pas un grand problème pour le bon fonctionnement du système. Par contre, le débordement de l'énergie hors l'enveloppe, même avec des très petites quantités, peut être une cause de doute sur l'efficacité du dispositif expérimental. D'où la nécessité de réduire au maximum cette erreur.

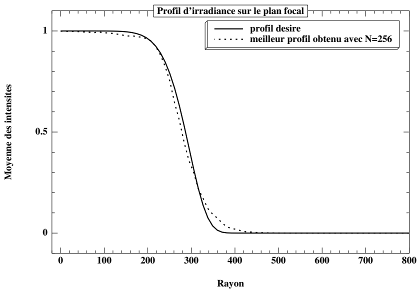
La figure 7.22 illustre le meilleur profil radial trouvé pour les degrés de discrétisation 32, 128 et 256, alors que la figure 7.23 montre l'aspect des lames de phases correspondant aux meilleurs résultats pour N=32 et N=128. Celle obtenue avec N=256 n'est pas représentée pour des raisons techniques.



Meilleur profil obtenu avec N=32
erreur=0.6526%, perte=0.2126%



Meilleur profil obtenu avec N=128
erreur=1.1%, perte=0.745%



Meilleur profil obtenu avec N=256
erreur=4%, perte=1.455%

FIG. 7.22 – *Meilleurs profils radiaux (sur 11 essais) pour les discrétisations (32, 128 et 256). Pour chaque cas, on indique le taux d'erreur (défaut d'uniformité entre le profil obtenu et le profil désiré) ainsi que le taux de perte d'énergie (l'erreur hors la cible à illuminer).*

Avec une discrétisation égale à 32, le plus petit taux d'erreur enregistré sur 11 essais après 5000 générations est de 0.6526%. Une performance pareille peut être un “excellent résultat” vu que le taux d'erreur s'approche de zéro. Cependant, la LPC correspondante n'est pas une solution réalisable physiquement. En effet, les degrés de discrétisation 32 et 64 ne sont pas assez précis pour

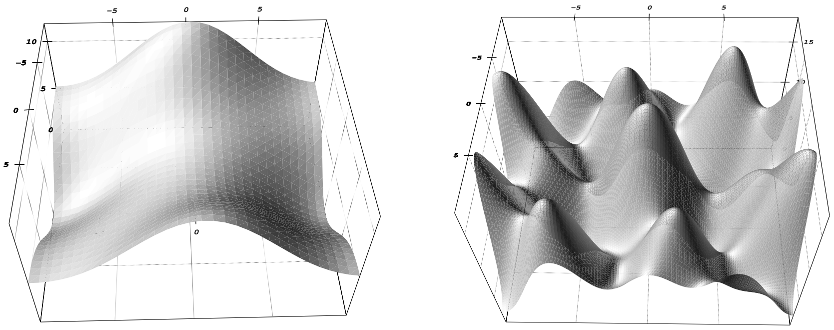


FIG. 7.23 – *Meilleures LPC obtenues pour les discrétisations 32 (à gauche) et 128 (à droite).*

la physique optique, et l'écart entre les calculs théoriques et réels peut être très grand.

Les solutions commencent à être concrètement acceptable à partir de N=128. Avec cette discrétisation, ES a montré une très grande robustesse en arrivant à atteindre un taux d'erreur de 1.42% en 5000 générations, en moins de 30h de temps CPU avec un Pentium II 350 Mhz, à comparer avec 120 heures de Cray YMP-2 pour une discrétisation de 64×64 pour enregistré une erreur de 7% [76]. On note que la fitness continue à s'améliorer tant que l'algorithme tourne. En effet, le meilleur résultat obtenu avec N=128 passe de 1.42% à la génération 5000 à 0.68% à la génération 20000, ce qui est considéré comme une très bonne performance. Cette amélioration est illustrée dans la courbe 7.24.

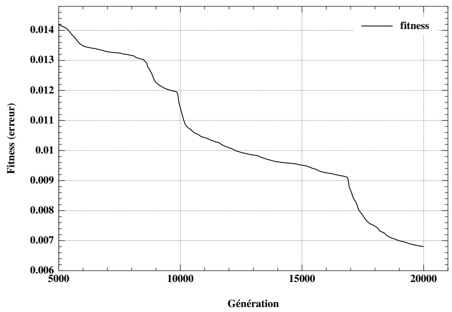


FIG. 7.24 – *Evolution de la fitness de la meilleure solution trouvée avec N=128, durant une extension du temps de calcul à 20 000 générations.*

Pour la discrétisation 256×256, le meilleur résultat enregistré est de 4%. Cette performance est considérée comme robuste, vu le nombre de paramètres à déterminer (2284). Par contre, le temps CPU augmente énormément ($\simeq$ 112h pour 5000 générations).

De point de vue physique, la réalisabilité des LPC augmente avec le degré de discrétisation (l'écart entre les calculs théoriques et la réalité diminue avec l'augmentation du degré de discrétisation). Cependant, la croissance rapide de la complexité et du temps CPU avec l'augmentation de N (figure 7.25) ne permet pas d'effectuer des tests avec des discrétisations plus grandes que 256. En effet, pour N=512, le nombre d'inconnues s'élève à 9112 et le temps CPU pour une génération s'élève à 560 secondes. Pour cette raison, une approche alternative dont la taille des solutions est indépendante de la discrétisation a été testée. La section suivante résume l'apport de cette technique.

Avec la deuxième approche, les LPC sont présentées par une fonctionnelle 2D, codées sous formes d'arbres. Contrairement à la représentation paramétrique, la taille de l'espace de recherche est indépendante de N. Par contre, au niveau computationnel, on est toujours obligé de passer par la représentation paramétrique pour le calcul de l'inverse de la transformée de Fourier rapide. On ne gagnera pas beaucoup donc au niveau temps CPU, mais essentiellement au niveau complexité.

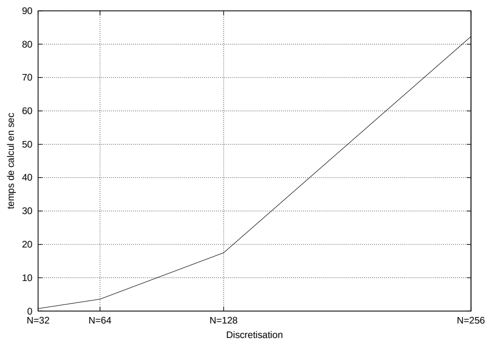


FIG. 7.25 – *Courbe d'évolution de temps CPU moyen par génération selon le degré de discrétisation N*

7.7 Comparaison de la Représentation Paramétrique vs la Représentation Fonctionnelle (GP)

Les résultats obtenus avec la représentation paramétrique sont très satisfaisants, mais l'algorithme reste incapable de traiter des discrétisations très élevées, et ceci malgré les simplifications techniques permettant de réduire l'espace de recherche (recherche dans l'espace de fréquentiel). La complexité des calculs augmente quadratiquement selon N (figure 7.25) et le domaine de recherche s'élargie très rapidement.

Une solution possible pour dépasser cette difficulté est de représenter autrement les solutions. Les arbres de **GP** peuvent être envisagés comme une représentation alternative, où on optimise directement la fonctionnelle 2D. Avec la représentation fonctionnelle, la taille de la solution (*génotype*) est complètement indépendante de la discrétisation choisie.

7.7.1 La Représentation Fonctionnelle (GP)

On rappelle que la programmation génétique est basée sur la manipulation de structures arborescentes dynamiques (cf. chapitre 1, section 1.4), souvent considérés comme des programmes [69, 70].

L'idée dans ce cas est de représenter directement le champs d'inconnues $\tilde{Z} \times Filtre$ comme un arbre. L'objectif avec cette nouvelle représentation est que le problème d'optimisation soit indépendant de la discrétisation.

Les valeurs des paramètres de GP utilisés pour l'expérimentation sont résumées dans le tableau 7.4. Une explication détaillée de leurs significations est donnée au chapitre 1, section 1.4.

Ensemble des noeuds	$\{+, -, *, sin\}$
Ensembles des terminaux	$\{x, y, \mathbb{R}\}$
Fenêtre de discrétisation	$[-32, 31]^2$
Taille de la population	**100**
Mode de sélection	**Tournoi (taille=10)**
Remplacement	**Générationnel avec élitisme**
Probabilité de croisement	**0.6**
Probabilité de mutation	**0.4**
Types de mutations	
	Promotion d'une branche
	Remplacement aléatoire d'une branche
	Permutation d'un noeud ou d'un terminal
	Mutation gaussienne des terminaux réels
Une combinaison de **3** opérateurs de mutation est utilisée à chaque mutation	
Profondeur maximale d'un individu	**14**
Profondeur minimale à l'initialisation	**3**
Intervalle d'initialisation des terminaux réels	**[-100,100]**
Déviation pour la mutation gaussienne des réels	**10**

TAB. 7.4 – *Les paramètres de GP*

7.7.2 Les résultats comparatifs ES vs GP

La figure 7.26 trace les meilleures performances ainsi que leurs moyennes sur 11 tests pour les deux approches et pour chaque degré de discrétisation. La première constatation est que les résultats sont comparables et particulièrement satisfaisants.

Toutefois, concernant la comparaison entre les deux approches paramétrique et fonctionnelle, certains résultats surprenants ont été révélés par l'expérimentation.

Dans les premières générations, **ES** et **GP** atteignent rapidement une performance raisonnable, ce qui peut être imputé au large intervalle d'initialisation pour **ES** et aux opérateurs génétiques de **GP** qui s'avèrent efficaces pour cette phase d'exploration. En effet, les mutations et croisement **GP** permettent à l'algorithme d'effectuer des grands sauts dans l'espace de recherche, puisque une petite mutation au niveau du génotype peut engendrer une modification radicale au niveau du phénotype.

Cependant, dans une phase plus avancée de l'évolution, tandis que **ES** continue d'améliorer la performance du meilleur individu, **GP** montre d'importantes difficultés pour raffiner les solutions et ainsi atteindre un plus faible taux d'erreur.

Avec **ES**, malgré une convergence qui tend à devenir assez lente, l'erreur continue de diminuer constamment. Ce phénomène peut être expliqué par la mutation auto-adaptative qui permet une exploration locale aux alentours des meilleurs. En effet, l'adaptation des déviations standards de

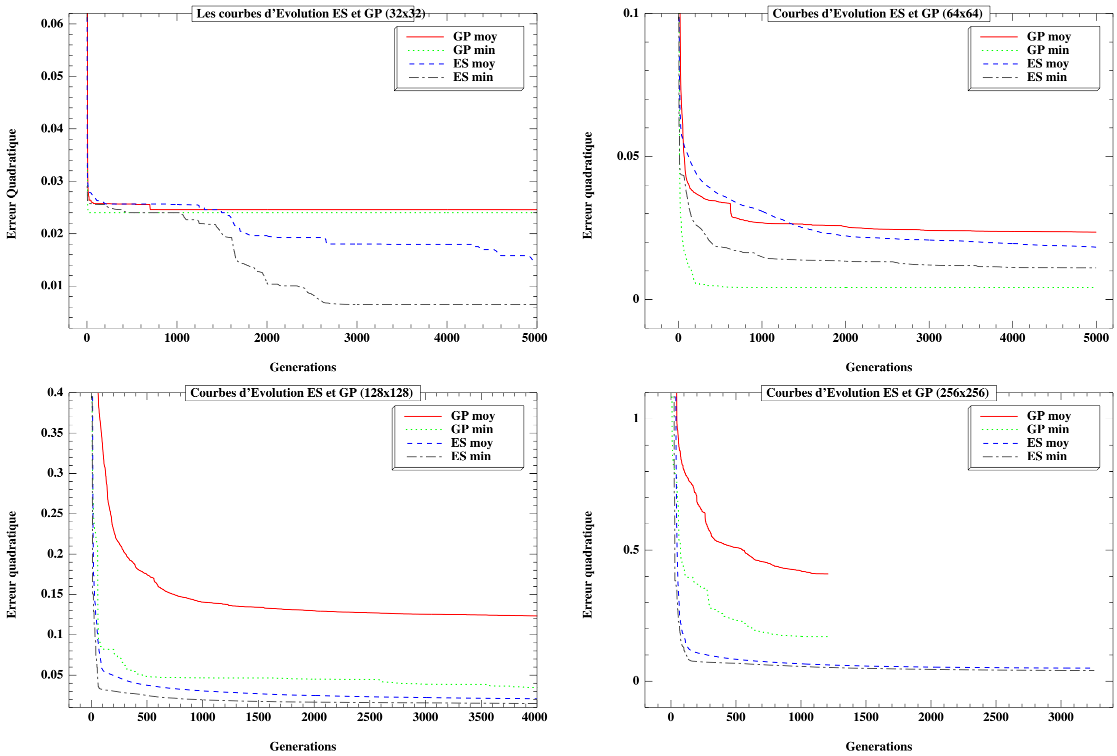


FIG. 7.26 – *Courbes d'évolution de la performance moyenne (sur 11 tests) et de la meilleure performance pour* **GP** *et* **ES** *pour N=32, 64, 128 et 256.*

la mutation permet de diminuer progressivement le taux de modifications des solutions pendant la mutation ce qui assure une bonne exploitation des meilleures solutions au cours de l'évolution.

D'un autre côté, **GP**, après une très efficace exploration de l'espace de recherche, semble avoir des difficultés pour raffiner la solution et améliorer sa performance. En fait, les opérateurs **GP** montrent un léger manque de précision et une certaine incapacité de régler finement le comportement phénotypique de l'individu à partir de sa représentation génotypique [5].

Un autre point qui renforce cette explication est que les meilleurs résultats sont obtenus à partir d'un tournoi de taille relativement importante (taille 10 sur une population de 100). Cette forte pression sélective oriente l'exploration de GP aux alentours des meilleurs individus pour améliorer le processus de raffinement des solutions. Cependant, cette approche réduit rapidement la diversité génétique de la population, et GP se retrouve alors entrain d'explorer juste une petite région de l'espace de recherche, ce qui favorise la convergence locale de l'algorithme.

7.7.3 Optimisation des constantes de la solution GP

Le manque de précision des solutions GP obtenues à la convergence nous a incité à chercher un moyen pour raffiner les meilleurs individus. Nous avons pensé qu'il est possible d'augmenter le taux de précision d'un arbre en optimisant les valeurs de ses terminaux réels. Par exemple, la meilleure solution GP obtenue avec la discrétisation 128×128 a 84 constantes et une performance (erreur quadratique entre le profil désiré et le profil obtenu) de 3.7752%. Vu le nombre des constantes, il y a une grande chance d'améliorer la performance de la solution en ajustant légèrement leurs valeurs.

Les premiers essais réalisés utilisent l'algorithme (1+1)-ES avec mutation auto-adaptative isotrope et auto-adaptative anisotrope (chapitre 1, section 1.3.2). Cependant, tous ces essais ont mené à l'échec, malgré la multitude de configurations essayées (plusieurs valeurs pour la déviation standard initiale et les taux de son adaptation). Ceci est due à la difficulté de trouver la bonne déviation standard initiale et de fait que les mutations auto-adaptatives mutent toutes les composantes de l'individu, ce qui n'est pas optimal pour une technique de raffinement d'un arbre.

Pour résoudre ce problème, il fallait changer la stratégie de mutation utilisée. Nous avons alors opté pour la mutation logarithmique [94, 48] présentée dans le chapitre 3, parce qu'elle ne demande pas un choix précis de la déviation standard, et le nombre de composantes mutées est décidé par l'utilisateur.

Des expérimentations sont alors réalisées en mutant 2, 3 ou 4 composantes sur l'ensembles les 84 constantes de la solution GP traitée. L'intervalle des perturbations générées a été fixé à $[-100, 100]$ qui est l'intervalle d'initialisation des terminaux réels de l'algorithme GP.

21 tests ont été effectués pour chaque cas, qui ont tous réussi à améliorer la performance de la solution traitée. La figure 7.27 illustrent les courbes d'évolution de la médiane et de la meilleure performance sur 21 essais, pour Nb_Comp=2,3 et 4, avec Nb_Comp est le nombre de composantes mutées.

A partir de Nb_Comp=4, le degré d'amélioration de la fitness commence à décroître. Pour cette raison, nous ne présentons pas des valeurs plus grandes de Nb_Comp.

La meilleure performance enregistrée est donnée par la mutation de 3 composantes par génération, mais en moyenne, l'algorithme est plus performant en mutant seulement 2 composantes. Pour la meilleure performance, l'erreur est passée de 3.7752% à 3.7086% au bout de 5000 générations.

L'amélioration moyenne est faible, mais continue à diminuer tant que l'algorithme tourne. La courbe 7.28 illustrant un test pour Nb_Comp=2, montre la fitness continue de descendre pendant 20000 générations d'évolution, où elle passe de 3.7752% à 3.66%.

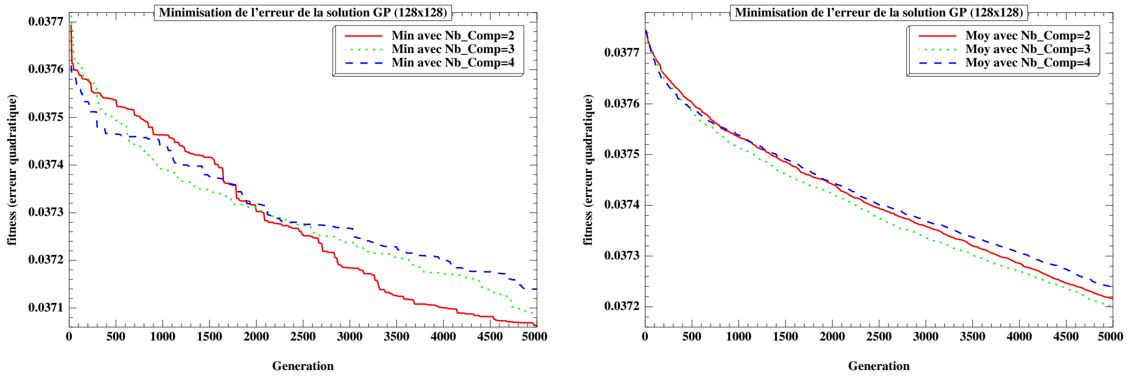


FIG. 7.27 – *Courbes des meilleures améliorations (à gauche) et des améliorations moyennes (à droite) de la performance de la solution GP, résultant des expérimentations de raffinement de ses terminaux réels avec l'algorithme (1+1)-ES.*

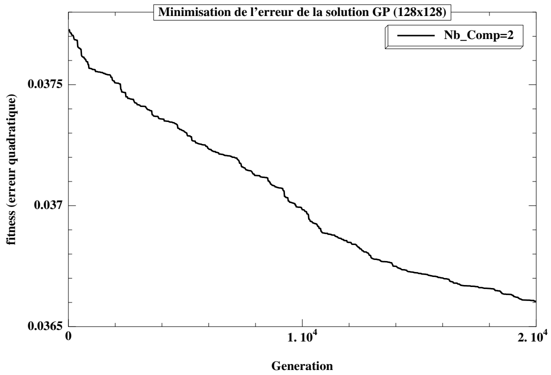


FIG. 7.28 – *Evolution de la performance de la solution GP avec le raffinement de ses constantes pendant 20000 générations (Nb_comp=2).*

Cependant, cette amélioration reste insuffisante pour amener les solutions GP au niveau des performances de celles données par l'approche paramétrique. L'optimisation locale des meilleures solutions améliore leur qualité, mais il est nécessaire d'optimiser aussi l'exploration globale de l'algorithme GP.

7.8 conclusion

Dans ce chapitre, nous avons traité un problème réel difficile avec les algorithmes évolutionnaires. L'objectif de cette application est de calculer la forme des Lames de Phases de type Mégajoules pour un système laser conçu pour la fusion nucléaire.

Dans la première partie de ce travail, le problème a été résolu avec une approche paramétrique en utilisant les stratégies d'évolution. Avec cette présentation, les solutions sont des matrices de nombres réels qui représente le masque de la lame des phases à déterminer dans l'espace fréquentiel.

Une étude préliminaire a été réalisée dans le but de déterminer la configuration idéale pour le problème. Une fois la bonne modélisation a été choisie, des tests ont été réalisés avec un degré de discrétisation des LPC croissant. Les résultats sont de très bonne qualité en comparaison avec ceux de l'état de l'art, mais la complexité du problème ainsi que le temps de calcul augmente quadratiquement avec le degré de discrétisation, et les ES rencontrent certaines difficultés à manipuler un très grand nombre d'inconnues.

Une autre approche a été alors testée indépendante du degré de discrétisation, basée sur une représentation fonctionnelle des individus, utilisée dans la programmation génétique GP. Comme pour la première approche, GP a donné des résultats très satisfaisants. Cependant, contrairement à ce qui a été prévu, les solutions obtenues avec les ES sont meilleures et plus précises que celles données par GP. Ceci peut être expliqué par la faiblesse de l'algorithme GP mis en place à ajuster les terminaux réels de l'arbre GP, et ainsi optimiser localement les meilleures solutions.

Une tentative d'optimiser les valeurs des terminaux réels de la meilleure solution GP pour $N = 128$ avec un $(1 + 1) - ES$ a été mené avec succès et a réussi à améliorer sa performance. Toutefois, cette amélioration reste insuffisante pour que la qualité des solutions GP arrive au niveau de celles données par l'approche paramétrique. Il est donc nécessaire d'améliorer aussi l'exploration globale de notre algorithme GP.

Une autre solution serait d'utiliser une population de très grande taille (Dans [69], J. Koza utilise des populations entre 800 et 1000 individus et dans [71], jusqu'à 40 000 individus), mais cette solution nécessite un ensemble de machines assez puissantes.

Indépendamment de l'approche adoptée, les algorithmes évolutionnaires ont réussi à concevoir un élément primordial dans le système laser dédié au *confinement Inertiel*. Ils ont enregistré une meilleure performance (aussi bien au niveau temps de calcul qu'au niveau qualité des résultats) que les méthodes décrites dans la littérature.

Annexe A

Les Fonctions tests pour l'optimisation sous contraintes

Dans cet annexe, on donne la description des 11 cas test, de G1 à G11, utilisés dans la partie expérimentale de l'optimisation sous contraintes.

•**Minimiser**

$G1(\overrightarrow{x}) = 5x_1 + 5x_2 + 5x_3 + 5x_4 - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=1}^{1} 3x_i,$

sous les contraintes suivantes:

$2x_1 + 2x_2 + x_{10} + x_{11} \leqslant 10,$
$2x_1 + 2x_3 + x_{10} + x_{12} \leqslant 10,$
$2x_2 + 2x_3 + x_{11} + x_{12} \leqslant 10,$
$-8x_1 + x_{10} \leqslant 0,$
$-8x_2 + x_{11} \leqslant 0,$
$-8x_3 + x_{12} \leqslant 0,$
$-2x_4 - x_5 + x_{10} \leqslant 0,$
$-2x_6 - x_7 + x_{11} \leqslant 0,$
$-2x_8 - x_9 + x_{12} \leqslant 0,$

et les bornes:

$0 \leqslant x_i \leqslant 1$ pour $1 \leqslant i \leqslant 9$,
$0 \leqslant x_i \leqslant 100$ pour $10 \leqslant i \leqslant 12$ et $0 \leqslant x_{13} \leqslant 1.$

Le problème a 13 variables et 9 contraintes linéaires. La fonction G1 est quadratique et son minimum global est $\overrightarrow{x^*} = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$, dont la valeur est: $G1(\overrightarrow{x^*}) = -15$.
Seulement six des noeufs contraintes sont actives au niveau de l'optimum global.

• **Maximiser**

$G2(\overrightarrow{x}) = |\frac{\sum_{i=1}^{n} cos^4(x_i) - 2\prod_{i=1}^{n} cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} ix_i^2}}|,$

sous les contraintes suivantes:

$\prod_{i=1}^{n} x_i \geqslant 0.75$, $\sum_{i=1}^{n} x_i \leqslant 7.5n$,

et les bornes:

$0 \leqslant x_i \leqslant 10$ pour $1 \leqslant i \leqslant n.$

La fonction G2 est non linéaire et son maximum global est inconnu. Pour $n = 20$, la meilleur solution trouvée est: $G2(\overrightarrow{x^*}) = 0.803619$, pour lequel la deuxième contrainte est presque active (10^{-8}).

- **Maximiser**

$G3(\overrightarrow{x}) = (\sqrt{n})^n . \prod_{i=1}^n x_i$,

sous les contraintes suivantes:

$\sum_{i=1}^n x_i^2 = 1$,

et les bornes:

$0 \leqslant x_i \leqslant 1$ pour $1 \leqslant i \leqslant n$.

L'optimum global de G3 est $x^*{}_i = 1/\sqrt(n), 1 \leqslant i \leqslant n$, où $G3(\overrightarrow{x}^*) = 1$.

- **Minimiser**

$G4(\overrightarrow{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141,$

sous les contraintes suivantes:

$0 \leqslant 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 \leqslant 92$
$90 \leqslant 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 \leqslant 110$
$20 \leqslant 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 \leqslant 25,$

et les bornes:

$78 \leqslant x_1 \leqslant 102$, $33 \leqslant x_2 \leqslant 45$, $27 \leqslant x_i \leqslant 45$ pour $i = 3, 4, 5$.

La solution optimale est $\overrightarrow{x}^* = (78, 33, 29.995256, 45, 36.775812)$, où $G4(\overrightarrow{x}^*) = -30665.5$. Deux contraintes sont actives au niveau de l'optimum, la première (borne supèrieure) et la troisième (borne infèrieure).

- **Minimiser**

$G5(\overrightarrow{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + 0.000002/3x_2^3$,

sous les contraintes suivantes:

$x_4 - x_3 + 0.55 \geqslant 0$, $x_3 - x_4 + 0.55 \geqslant 0$,
$1000sin(-x_3 - 0.25) + 1000sin(-x_4 - 0.25) + 894.8 - x_1 = 0$,
$1000sin(x_3 - 0.25) + 1000sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$
$1000sin(x_4 - 0.25) + 1000sin(x_4 - x_3 - 0.25) + 1294.8 = 0$,

et les bornes:

$0 \leqslant x_i \leqslant 1200$ pour $i = 1, 2$
et $-0.55 \leqslant x_i \leqslant 0.55$ pour $i = 3, 4$

La meilleure solution connue pour ce problème est $\overrightarrow{x}^* = (679.9453, 1026.067, 0.1188764, -0.3962336)$, avec $G5(\overrightarrow{x}^*) = 5126.4981$.

- **Minimiser**

$G6(\overrightarrow{x}) = (x_1 - 10)^3 + (x_2 - 20)^3$,

sous les contraintes non linéaires suivantes:

$(x_1 - 5)^2 + (x_2 - 5)^2 - 100 \geqslant 0$,
$-(x_1 - 6)^2 - (x_2 - 5)^2 + 82.81 \geqslant 0$,

et les bornes:

$13 \leqslant x_1 \leqslant 100$ et $0 \leqslant x_2 \leqslant 100$.

La solution globale connue est $\overrightarrow{x}^* = (14.095, 0.84296)$, avec $G6(\overrightarrow{x}^*) = -6981.81388$.
Les deux contraintes du problème sont actives aux alentours de l'optimum.

- **Minimiser**
  $G7(\overrightarrow{x}) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$,

sous les contraintes suivantes:

$105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 \geqslant 0$,
$-3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2x_3^2 + 7x_4 + 120 \geqslant 0$,
$-10x_1 + 8x_2 + 17x_7 - 2x_8 \geqslant 0$,
$-x_1^2 - 2(x_2 - 2)^2 + 2x_1x_2 - 14x_5 + 6x_6 \geqslant 0$,
$8x_1 - 2x_2 - 5x_9 + 2x_{10} + 12 \geqslant 0$,
$-5x_1^2 - 8x_2 - (x_3 - 6)^2 + 2x_4 + 40 \geqslant 0$,
$3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} \geqslant 0$,
$-0.5(x_1 - 8)^2 - 2(x_2 - 4)^2 - 3x_5^2 + x_6 + 30 \geqslant 0$,

et les bornes:

$-10.0 \leqslant x_i \leqslant 10.0, \quad i = 1, \ldots, 10$.

Ce problème a 3 contraintes linéaires et 5 autres non linéaires. La fonction G7 est quadratique et a comme optimu global
$\overrightarrow{x}^* = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927)$
où $G7(\overrightarrow{x}^*) = 24.3062091$. Six (sur huit) contraintes sont actives aux alentours de l'optimum global (toutes à part les deux dernières).

- **Maximiser**
  $G8(\overrightarrow{x}) = \frac{\sin^3(2\pi x_1)\cdot\sin(2\pi x_2)}{x^3\cdot(x_1+x_2)}$,

sous les contraintes suivantes:

$x_1^2 - x_2 + 1 \leqslant 0$,
$1 - x_1 + (x_2 - 4)^2 \leqslant 0$

et les bornes:

$0 \leqslant x_1 \leqslant 10$ et $0 \leqslant x_2 \leqslant 10$.

La fonction G8 a plusieurs optima locaux, les plus hauts pics se trouvent au long de l'axe des x.
L'optimum connu se trouve à l'intérieur du domaine faisable et a comme coordonnées:
$\overrightarrow{x}^* = (-1.737459, -0.167763)$, avec $G8(\overrightarrow{x}^*) = 0.095825$.

- **Minimiser**
  $G9(\overrightarrow{x}) = (x_1-10)^2+5(x_2-12)^2+x_3^4+3(x_4-11)^2+10x_5^6+7x_6^2+x_7^4-4x_6x_7-10x_6-8x_7$,

sous les contraintes suivantes:

$127 - 2x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 \geqslant 0$,
$282 - 7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 \geqslant 0$,
$196 - 23x_1 - x_2^2 - 6x_6^2 + 8x_7 \geqslant 0$,
$-4x_1^2 - x_2^2 + 3x_1x_2 - 2x_3^2 - 5x_6 + 11x_7 \geqslant 0$

et les bornes:

$-10.0 \leqslant x_i \leqslant 10.0, i = 1, \ldots, 7$.

Ce problème a 4 contraintes non linéaires. La fonction G9 est aussi non linéaire et son minimum global est:
$\overrightarrow{x}^* = (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227)$,
où $G9(\overrightarrow{x}^*) = 680.6300573$.
Deux (sur quatre) contraintes sont actives au niveau de l'optimum global (la première et la dernière).

- **Minimiser**
  $G10(\overrightarrow{x}) = x_1 + x_2 + x_3$,

sous les contraintes suivantes:

$1 - 0.0025(x_4 + x_6) \geqslant 0$,
$1 - 0.0025(x_5 + x_7 - x_4) \geqslant 0$,
$1 - 0.01(x_8 - x_5) \geqslant 0$,
$x_1 x_6 - 833.33252 x_4 - 100 x_1 + 83333.333 \geqslant 0$,
$x_2 x_7 - 1250 x_5 - x_2 x_4 + 1250 x_4 \geqslant 0$,
$x_3 x_8 - 1250000 - x_3 x_5 + 2500 x_5 \geqslant 0$,

et les bornes:

$100 \leqslant x_1 \leqslant 10000,\ 1000 \leqslant x_i \leqslant 10000,\ i = 2, 3,\ 10 \leqslant x_i \leqslant 1000,\ i = 4, \ldots, 8.$

La problème a trois contraintes linéaires et trois autres non linéaires. La fonction G10 est linéaire et son minimum global est: $\overrightarrow{x}^* = (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.9799, 286.4162, 395.5979)$, où $G10(\overrightarrow{x}^*) = 7049.330923$. Toutes les contraintes sont actives au niveau de l'optimum global.

- **Minimiser**
  $G11(\overrightarrow{x}) = x_1^2 + (x_2 - 1)^2$,

sous les contraintes non linéaires suivantes:

$x_2 - x_1^2 = 0$,

et les bornes:

$-1 \leqslant x_i \leqslant 1,\ i = 1, 2.$

Deux solutions globales sont connues pour ce problème qui sont: $\overrightarrow{x}^* = (\pm 0.70711, 0.5)$, avec $G11(\overrightarrow{x}^*) = 0.75000455$.

Description des cas test

Une description générale des 11 cas test a été par Micheliwicz ...donnant pour chaque fonction son type, le nombre et le type des contraintes à les quelles elle est soumise, le nombre de contraintes actives aux alentours de l'optimum global et le pourcentage des points faisables sur un ensemble de 1 000 000 points générés aléatoirement dans l'espace de recherche.

Fonction	nb variables	Type de f	ρ	IL	EN	IN	a
G1	13	quadratique	0.0111%	9	0	0	6
G2	k	non linéiare	99.8474%	0	0	2	1
G3	k	polynomiale	0.0000%	0	1	0	1
G4	5	quadratique	52.1230%	0	0	6	2
G5	4	cubique	0.0000%	2	3	0	3
G6	2	cubique	0.0066%	0	0	2	2
G7	10	quadratique	0.0003%	3	0	5	6
G8	2	non linéaire	0.8560%	0	0	2	2
G9	7	polynomiale	0.5121%	0	0	4	2
G10	8	linéaire	0.0010%	3	0	3	6
G11	2	quadratique	0.0000%	0	1	0	1

TAB. A.1 – *Résumé des 11 cas test. Le ration* $\rho = |F|/|S|$ *est déterminé manuellement en générant 1 000 000 points aléatoires dans l'espace de recherche* S *et en testant à chaque fois si le point appartient à* F *ou non. Pour G2 et G3, le nombre de variables* k *a été fixé à 50. IL, EN et IN repésentent le nombre des inégalités linéaires, des équations et des inéaglités non linéaires, respectivement.*

Annexe B

Des notions d'optique pour l'application du sytème laser

Dans cet annexe, on décrit quelques notions élémentaires dans le domaine de la physique du laser, et qui peuvent être utiles pour la compréhension de certains calculs réalisés dans l'application du système laser, spécialement pour le calcul des champs d'intensité. On définit esssentiellement l'onde électromagnétique et la représentation scalaire de l'intensité lumineuse.

B.0.1 La lumière vue comme une onde électromagnétique

En optique ondulatoire, la lumière est modélisée par une *onde électromagnétique*, générée par une combinaison de deux champs:
– un champ *électrostatique* $\overrightarrow{E}(M,t)$
– un champ magnétique $\overrightarrow{B}(M,t)$
Une onde *monochromatique plane* est alors caractérisée par:

$$\overrightarrow{E}(\overrightarrow{r},t) = \overrightarrow{E_0}e^{i\omega t}e^{-i\overrightarrow{k}.\overrightarrow{r}}$$
$$\overrightarrow{B}(\overrightarrow{r},t) = \overrightarrow{B_0}e^{i\omega t}e^{-i\overrightarrow{k}.\overrightarrow{r}}$$

où:
– $\overrightarrow{r} = \overrightarrow{OM}$
– ω est la pulsation.
– k est le vecteur d'onde.. $\overrightarrow{k} = k.\overrightarrow{u}$, $\overrightarrow{u}$ donne la direction de propagation.
– $\overrightarrow{E_0}$ et $\overrightarrow{B_0}$ sont les amplitudes.
Les équations de Maxwell fournissent une relation simple entre $\overrightarrow{E}$ and $\overrightarrow{B}$ (voir figure B.1).

$$\overrightarrow{B}(\overrightarrow{r},t) = \frac{\overrightarrow{k}}{w} \wedge \overrightarrow{E}(\overrightarrow{r},t)$$

Ainsi nous pouvons assimiler une onde électromagnétique à un champ électrostatique.

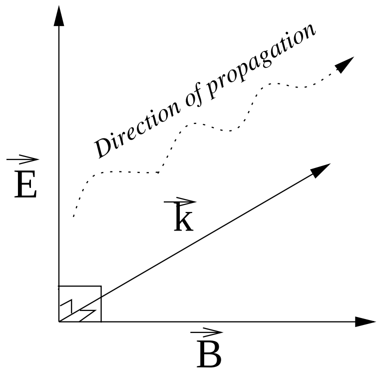


FIG. B.1 – *Une onde électromagnétique*

Dans ce cas, le champ électrostatique peut être défini par le champ scalaire $E(\vec{r})$:

$$\vec{E}(\vec{r}, t) = E(\vec{r})e^{i\omega t}\vec{u}$$

On néglige t et on ne considère que $\vec{E}(\vec{r})$ pour caractériser le champ électromagnétique.

B.0.2 L'intensité lumineuse

L'intensité de l'onde lumineuse peut être calculée à partir du champ scalaire

$$I(\vec{r}) = \gamma.|E(\vec{r})|^2$$

où γ est une constante. Pour l'application concernant le système laser, nous avons considéré $\gamma = 1$.

Liste des tableaux

Table des figures

Bibliographie

[1] Abadie, Carpentier, and Hensgen. Generalisation of the wolfe reduced gradient. *Note de l'électricité de France*, Avril 1967.

[2] M. Ahuactzin, E. Talbi, P. Bessiere, and E. Mazer. Using genetic algorithms for robot motion planning. In *Proceedings of European Conference on Artificial Intelligence*, 1992.

[3] J. M Alliot. *Techniques d'optimisation stochastique appliquées aux problemes du trafic aérien.* PhD thesis, ENA, 1996.

[4] Peter J. Angeline. Genetic programming's continued evolution. In Peter J. Angeline and K. E. Kinnear, Jr., editors, *Advances in Genetic Programming*, volume 2, pages 89–110. MIT Press, Cambridge, MA, USA, 1996.

[5] Peter J. Angeline. Subtree crossover: Building block engine or macromutation? In J. R. Koza and al., editors, *GP97: Proceedings of the* $2^n d$ *Annual Conf.*, pages 13–16. Morgan Kaufmann, jul 1997.

[6] Peter J. Angeline and J. B. Pollack. The evolutionary induction of subroutines. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, Bloomington, Indiana, USA, 1992. Lawrence Erlbaum.

[7] M. Avriel. *Nonlinear Programming - Analysis and Methods.* Prentice-Hall, 1976.

[8] T. Bäck, F. Hoffmeister, and H. P. Schwefel. A survey of evolution strategies. In R. K. Belew and L. B. Booker, editors, *Proceedings of the* 4^{th} *International Conference on Genetic Algorithms*, pages 2–9. Morgan Kaufmann, 1991.

[9] Th. Bäck and H. P. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–23, 1993.

[10] Th. Bäck and H. P. Schwefel. Evolutionary computation: An overview. 1996.

[11] James E. Baker. Adaptive selection methods for genetic algorithms. In J. J. Grefenstette, editor, *Proceedings of the* 1^{st} *International Conference on Genetic Algorithms*, pages 101–111. Laurence Erlbaum Associates, July 1985.

[12] James E. Baker. Reducing bias and inefficiency in the selection algorithm. In J. J. Grefenstette, editor, *Proceedings of the* 2^{nd} *International Conference on Genetic Algorithms*, pages 14–21, 1987.

[13] G. Bilchev and I. C. Parmee. Ant colony search vs. genetic algorithms. Technical report, Plymouth Engineering Design Centre, University of Plymouth, 1995.

[14] A. Colorni, M. Dorigo, and V. Maniezzo. Distributed optimization by ant colonies. In *Proceedings of the First European Conference on Artificial Life*, Paris, 1991. MIT Press/Bradford Book.

[15] Courant and Hilbert. Methods of mathematical physics. *Intersciences*, 1:34–36, 1953.

[16] N. L. Cramer. A representation for the adaptive generation of simple sequential programs. In J. J. Grefenstette, editor, *Proceedings of the* 1^{st} *International Conference on Genetic Algorithms*, pages 183–187, 1985.

[17] J. M. Daida, J. D. Hommes, S. J. Ross, and J. F. Vesecky. Extracting curvilinear features from SAR images of arctic ice: Algorithm discovery using the genetic programming paradigm. In T. Stein, editor, *Proceedings of IEEE International Geoscience and Remote Sensing*, pages 673–675, Florence, Italy, 1995. IEEE Press.

[18] G.B Dantzig. Application of the simplex method to a transportation problem. In NY T.C. Koopmans ed., Wiley, editor, *in Activity Analysis of Production and Allocation*, 1951.

[19] K. Deb. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 2000.

[20] K. Deb and D. E. Goldberg. An investigation of niche and species formation in genetic function optimization. In J. D. Schaffer, editor, *Proceedings of the* 3^{rd} *International Conference on Genetic Algorithms*, pages 42–50. Morgan Kaufman, 1989.

[21] K. A. DeJong. *The Analysis of the Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, University of Michigan, Ann Harbor, 1975. *Dissertation Abstract International*, 36(10), 5140B. (University Microfilms No 76-9381).

[22] K. A. DeJong and J. Sarma. On decentralizing selection algorithms. In L. J. Eshelman, editor, *Proceedings of the* 6^{th} *International Conference on Genetic Algorithms*, pages 17–23. Morgan Kaufmann, 1995.

[23] K .A. DeJong and W. M. Spears. An analysis of multi-point crossover. In G. J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 301–315, San Mateo, 1991. Morgan Kaufmann.

[24] K. A. DeJong and W. M. Spears. A formal analysis of the role of multi-point crossover in genetic algorithms. *Artificial Intelligence*, 5:1–26, 1992.

[25] S. N. Dixit, M. D. Feit, M. D. Perry, and H. T. Powell. Designing fully continuous phase screens for tailoring focal-plane irradiance profiles. *Optics Letters 21*, 21(21):1715–1717, November 1996.

[26] A. E. Eiben, C. H. M. Van Kemenade, and J. N. Kok. Orgy in the computer: multi-parent reproduction in genetic algorithms. In *European Conference on Artificial Life*, Granada, 1995.

[27] L. Eshelman and J. D. Schaffer. Real-coded genetic algorithms and interval-schemata. In L. D. Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 187–202, Los Altos, CA, 1993. Morgan Kaufmann.

[28] Faure et Huard. Résolution de programmes mathématiques à fonction non linéaire par la méthode du gradient réduit. *Revue Française de Recherche Opérationnelle*, (36):167–206, 1965.

[29] A. V. Fiacco and G. P. McCormick. Extensions of the sequential unconstrained minimisation technique for nonlinear programming. In *Congres of Management Sciences Institute*, February 1965.

[30] A. V. Fiacco and G. P.McCormick. The sequential unconstrained minimisation technique for nonlinear programming a primal dual method. *Management Science*, 10(2), 1964.

[31] R. Fletcher. *Practical Methode of Optimization (second edition).* Chichester, 1987.

[32] C. Floudas and P. Pardalos. *A Collection of Test Problems for Constrained Global Optimization Algorithms.* LNCS, 1987.

[33] D. B. Fogel. An analysis of evolutionary programming. In D. B. Fogel and W. Atmar, editors, *Proceedings of the 1^{st} Annual Conference on Evolutionary Programming*, pages 43–51, La Jolla, CA, 1992. Evolutionary Programming Society.

[34] D. B. Fogel. *Evolving artificial intelligence.* PhD thesis, University of California, San Diego, CA, 1992.

[35] D. B. Fogel. *Evolutionary computation. Toward a New Philosophy of Machine Intelligence.* Piscataway, NJ: IEEE Press, 1995.

[36] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution.* New York: John Wiley, 1966.

[37] C. M. Fonseca and P. J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, 1995.

[38] Laboratory for Laser Energetics. Distributed phase plates for super gaussian focal-plane irradiance profiles. *LLE Review*, 63:126–129.

[39] Laboratory for Laser Energetics. High-efficiency distributed phase plate generation and characterization. *LLE Review*, 65:1–8.

[40] A. Fukunaga and A. Stechert. Evolving nonlinear predictive models for lossless image compression with genetic programming. In John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba, and Rick Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 95–102, University of Wisconsin, Madison, Wisconsin, USA, 22-25 July 1998. Morgan Kaufmann.

[41] H. De Garis. Genetic programming : building artificial nervous systems using genetically programmed neural networks modules. In R. Porter and B. Mooney, editors, *Proceedings of the 7^{th} International Conference on Machine Learning*, pages 132–139. Morgan Kaufmann, 1990.

[42] D. E. Goldberg. *Genetic algorithms in search, optimization and machine learning.* Addison Wesley, 1989.

[43] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multi-modal function optimization. In J. J. Grefenstette, editor, *Proceedings of the 2^{nd} International Conference on Genetic Algorithms*, pages 41–49. Lawrence Erlbaum Associates, 1987.

[44] J. J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Trans. on Systems, Man and Cybernetics*, SMC-16:122–128, 1986.

[45] C. A. Grimes. Application of genetic techniques to the planning of railway track maintenance work. In A. M. S. Zalzala, editor, *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, GALESIA*, volume 414, pages 467–472, Sheffield, UK, 12-14 September 1995. IEE.

[46] E. LUTTON M. SCHOENAUER H. HAMDA, F. JOUVE and M. SEBAG. Unstructured representations in evolutionary topological optimum design. *IJAI, The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies, Special Issue on Creative Evolutionary Systems*, 200.

[47] A. B. Hadj-Alouane and J. C. Bean. A genetic algorithm for the multiple-choice integer program. Technical Report TR 92-50, Department of Industrial and Operations Engineering, The University of Michigan, 1992.

[48] S. Ben Hamida and A. Petrowski. The need for improving the exploration operators for constrained problems. In *Proceedings of the Congress On Evolutionary Computation*, LNCS, pages 1176–1183. Springer Verlag, 2000.

[49] S. Ben Hamida and M. Schoenauer. An adaptive algorithm for constrained optimization problems. In M. Schoenauer and al., editors, *Proceedings of the 6^{th} Conference on Parallel Problems Solving from Nature*, LNCS 1917, pages 529–538. Springer Verlag, 2000.

[50] Simon Handley. Classifying nucleic acid sub-sequences as introns or exons using genetic programming. In Christopher Rawlins, Dominic Clark, Russ Altman, Lawrence Hunter, Thomas Lengauer, and Shoshana Wodak, editors, *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology (ISMB-95)*, pages 162–169, Cambridge, UK, 1995. AAAI Press.

[51] W. E. Harush. Minima of functions of several variables with inequalities as side conditions, 1939. Master's thesis.

[52] D. Himmelblau. *Applied Nonlinear Programming*. McGraw-Hill, 1972.

[53] R. Hinterding and Z. Michalewicz. Your brains and my beauty: Parent matching for constrained optimization. In D.B. Fogel, editor, *Proceedings of the Fifth IEEE International Conference on Evolutionary Computation*, pages 810–815. IEEE Press, 1998.

[54] R. Hinterding, Z. Michalewicz, and A. E. Eiben. Adaptation in evolutionary computation: A survey. In T. Bäck, Z. Michalewicz, and X. Yao, editors, *Proceedings of the Fourth IEEE International Conference on Evolutionary Computation*, pages 65–69. IEEE Press, 1997.

[55] W. Hock and K. Schittkowski. *Test Examples for Nonlinear Programming Codes*, volume 187. Springer Verlag, 1981.

[56] J. H. Holland. Outline for a logical theory of adaptive systems. *Journal of the association of computing machinery*, 3, 1962.

[57] J. H. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, 1975.

[58] A. Homaifar, S. H.-Y. Lai, and X. Qi. Constrained optimization via genetic algorithms. *Simulation*, 62(4):242–254, 1994.

[59] J. Horn, D.E. Goldberg, and K. Deb. Long path problems. In Y. Davidor, H.-P. Schwefel, and R. Manner, editors, *Proceedings of the 3^{rd} Conference on Parallel Problems Solving from Nature*, pages 149–158. Springer Verlag, 1994.

[60] J. Horn, S. N. Nafpliotis, and D. E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In Z. Michalewicz, J. D. Schaffer, H.-P. Schwefel, D. B. Fogel, and H. Kitano, editors, *Proceedings of the First IEEE International Conference on Evolutionary Computation*, volume 1, pages 82–87. IEEE Press, 1994.

[61] J. Horn and SN. Nafpliotis. Multiobjective optimization using the niched pareto genetic algorithm. Technical report, Department of General Engineering, University of Illinois at Urbana-Champaign, Illinois, 1993.

[62] R. Toombs J. Reed and N.A. Barricelli. Simulation of biological evolution and machine learning. *Journal Theor. Biol.*, 17:319–342, 1967.

[63] J.A. Joines and C.R. Houck. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs. In Z. Michalewicz, J. D. Schaffer, H.-P. Schwefel, D. B. Fogel, and H. Kitano, editors, *Proceedings of the First IEEE International Conference on Evolutionary Computation*, pages 579–584. IEEE Press, 1994.

[64] L. Kallel. *Convergence des algorithmes génétiques: aspects spatiaux et temporels*. PhD thesis, Ecole Polytechnique, 1999.

[65] C. Kane. *Algorithmes génétiques et Optimisation topologique*. PhD thesis, Université de Paris VI, July 1996.

[66] C. Kane and M. Schoenauer. Topological optimum design using genetic algorithms. *Control and Cybernetics*, 25(5):1059–1088, 1996.

[67] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, (4):373–395, 1984.

[68] L. G. Khachian. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, (20):191–194, 1979.

[69] J. R. Koza. *Genetic Programming: On the Programming of Computers by means of Natural Evolution*. MIT Press, Massachussetts, 1992.

[70] J. R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Massachussetts, 1994.

[71] J. R. Koza, F. H Bennett III, D. Andre, and M. A. Keane. *Genetic Programming II: Darwinian Invention and Problem Solving*. Morgan Kaufmann Publishers, San Francisco, 1999.

[72] John R. Koza. Evolution of a computer program for classifying protein segments as transmembrane domains using genetic programming. In Russ Altman, Douglas Brutlag, Peter Karp, Richard Lathrop, and David Searls, editors, *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, pages 244–252. AAAI Press, 1994.

[73] S. Koziel and Z. Michalewicz. Evolutionary algorithms, homomorphous mapping and constrained parameter optimization. *Evolutionary Computation*, 7(1):19–44, 1999.

[74] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In J. Neyman, editor, *Proceedings of the second Berkely Symposium on Methematical Studies and Probability*, pages 481–492. University of California Press, 1951.

[75] R. G. Leriche, C. Knopf-Lenoir, and R. T. Haftka. A segragated genetic algorithm for constrained structural optimization. In L. J. Eshelman, editor, *Proceedings of the* 6^{th} *International Conference on Genetic Algorithms*, pages 558–565, 1995.

[76] Y. Lin, T. J. Kessler, and G. N. Lawrence. Design of continuous surface-relief phase plates by surface-based simulated annealing to achieve control of focal-plane irradiance. *Optics Letters*, 21(20):1703–1705, October 1996.

[77] Evelyne LUTTON. *Genetic Algorithms and Fractals - Algorithmes Génétiques et Fractales*. PhD thesis, Université Paris XI, Orsay, 1992. Dossier d'Habilitation à diriger des recherches.

[78] C.Y. Maa and M.A. Shanblatt. A two-phase optimization neural network. *IEEE Transactions on Neural Networks*, 3(6):1003–1009, 1992.

[79] S. W. Mahfoud. Crowding and preselection revisited. In R. Manner and B. Manderick, editors, *Proceedings of the* 2^{nd} *Conference on Parallel Problems Solving from Nature*, pages 27–36. Springer Verlag, 1992.

[80] Brij Masand. Optimising confidence of text classification by evolution of symbolic expressions. In Kenneth E. Kinnear, Jr., editor, *Advances in Genetic Programming*, chapter 21, pages 445–458. MIT Press, 1994.

[81] Z. Michalewicz. Genetic algorithms, numerical optimization and constraints. In L. J. Eshelman, editor, *Proceedings of the* 6^{th} *International Conference on Genetic Algorithms*, pages 151–158. Morgan Kaufmann, 1995.

[82] Z. Michalewicz. *Genetic Algorithms+Data Structures=Evolution Programs*. Springer Verlag, New-York, 1996. 3rd edition.

[83] Z. Michalewicz and N. Attia. Evolutionary optimization of constrained problems. In A.V. Sebald and L.J. Fogel, editors, *Proceedings of the* 3^{rd} *Annual Conference on Evolutionary Programming*, pages 98–108. World Scientific, 1994.

[84] Z. Michalewicz, D. Dasgupta, R. Leriche, and M. Schoenauer. Evolutionary algorithms for constrained engineering problems. *Computers & Industrial Engineering Journal*, 30(2), April 1996.

[85] Z. Michalewicz and C. Z. Janikow. Handling constraints in genetic algorithms. In R. K. Belew and L. B. Booker, editors, *Proceedings of the* 4^{th} *International Conference on Genetic Algorithms*, pages 151–157. Morgan Kaufmann, 1991.

[86] Z. Michalewicz and G. Nazhiyath. Genocop III: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints. In *Proceedings of the Second IEEE International Conference on Evolutionary Computation*, 1.

[87] Z. Michalewicz and M. Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, 4(1):1–32, 1996.

[88] H. Myung, J.-H. Kim, and D.B. Fogel. Preliminary investigation into a two-stage method of evolutionary optimization on constrained problems. In J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, editors, *Proceedings of the* 4^{th} *Annual Conference on Evolutionary Programming*, pages 449–463. MIT Press, 1995.

[89] Peter Nordin. A compiling genetic programming system that directly manipulates the machine code. In *Advances in Genetic Programming*, pages 311–332. 1994.

[90] J. Paredis. Coevolutionary constraint satisfaction. In Y. Davidor, H.-P. Schwefel, and R. Manner, editors, *Proceedings of the* 3^{rd} *Conference on Parallel Problems Solving from Nature*, pages 46–55. Springer-Verlag, LNCS 866, 1994.

[91] J. Paredis. Towards balanced coevolution. In M. Schoenauer and al., editors, *Proceedings of the* 6^{th} *Conference on Parallel Problems Solving from Nature*, LNCS, pages 529–538. Springer Verlag, 2000.

[92] I.C. Parmee and G. Purchase. The development of directed genetic search technique for heavily constrained design spaces. In *Proceedings of the Conference on Adaptive Computing in Engineering Design and Control*, pages 97–102. University of Plymouth, 1994.

[93] A. Petrowski. A clearing procedure as a niching method for genetic algorithms. In T. Fukuda, editor, *Proceedings of the Third IEEE International Conference on Evolutionary Computation*, pages 798–803. IEEE press, 1996.

[94] A. Petrowski and S. Ben Hamida. A logarithmic mutation operator to solve constrained optimization problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann, 1999.

[95] D. Powell and M. M. Skolnick. Using genetic algorithms in engineering design optimization with non-linear constraints. In S. Forrest, editor, *Proceedings of the* 5^{th} *International Conference on Genetic Algorithms*, pages 424–430. Morgan Kaufmann, 1993.

[96] A. Racine, S. Ben Hamida, and M. Schoenauer. Parametric coding vs genetic programming: A case study. In W.B. Langdon R. Poli P. Nodin and T. Fogarty, editors, *Proceedings of the Second European Workshop on Genetic Programming*, 1999.

[97] I. Rechenberg. *Evolution Strategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution*. Fromman-Hozlboog Verlag, Stuttgart, 1973.

[98] I. Rechenberg. *Evolution Strategie '94*, volume 1. Fromman-Hozlboog Verlag, Stuttgart, 1994.

[99] R.G. Reynolds, Z. Michalewicz, and M. Cavaretta. Using cultural algorithms for constraint handling in Genocop. In J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, editors, *Proceedings of the* 4^{th} *Annual Conference on Evolutionary Programming*, pages 298–305. MIT Press, 1995.

[100] R. T. Rockafellar. Convex functions and duality in optimization problems and dynamics. *Lecture Notes in Operations Research and Mathematical Economics*, (11):117–141, 1969.

[101] R. T. Rockafellar. Augmented lagrange muiltiplier functions and duality in nonconvex programming. *SIAM Journal On Control*, (12):268–285, 1974.

[102] E. Ronald. When selection meets seduction. In L. J. Eshelman, editor, *Proceedings of the* 6^{th} *International Conference on Genetic Algorithms*, pages 167–173. Morgan Kaufmann, 1995.

[103] Rosen. The gradient projection method for nonlinear programming, part i, linear constraints. *Journal of the Society for Industrial and Applied Mathematics*, 8:181–217, 1960.

[104] Rosen. The gradient projection method for nonlinear programming, part ii, nonlinear constraints. *Journal of the Society for Industrial and Applied Mathematics*, 9:514–532, 1961.

[105] Gerald P. Roston. *A Genetic Methodology for Configuration Design*. PhD thesis, Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213-3891, USA, December 1994.

[106] Thomas P. Runarsson and Xin Yao. Stochastic ranking for constrained evolutionary optimization. *Evolutionary Computation*, 4(3):284–294, September 2000.

[107] Conor Ryan, J. J. Collins, and Michael O Neill. Grammatical evolution: Evolving programs for an arbitrary language. In Wolfgang Banzhaf, Riccardo Poli, Marc Schoenauer, and Terence C. Fogarty, editors, *Proceedings of the First European Workshop on Genetic Programming*, volume 1391 of *LNCS*, pages 83–95, Paris, 14-15 April 1998. Springer-Verlag.

[108] A. Racine S. Ben Hamida and M. Schoenauer. Two evolutionary approaches to design phase plate for tailoring focal-plane irradiance profile. In C. Fonlupt J-K. Hao E. Lutton E. Ronald and M. Schoenauer, editors, *Artificial Evolution'99*, LNCS 1829, pages 266–276. Springer Verlag, 1999.

[109] N. Saravanan and D. B. Fogel. Learning strategy parameters in evolutionary programming: an empirical study. In A.V. Sebald and L.J. Fogel, editors, *Proceedings of the* 3^{rd} *Annual Conference on Evolutionary Programming*, pages 269–280, 1994.

[110] N. Saravanan, D. B. Fogel, and K. M. Nelson. A comparison of methods for self-adaptation in evolutionary algorithms. *Biosystems*, 36:157–166, 1995.

[111] D.J. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In J. J. Grefenstette, editor, *Proceedings of the* 1^{st} *International Conference on Genetic Algorithms*. Laurence Erlbaum Associates, 1985.

[112] D. Schlierkamp-Voosen and H. Mühlenbein. Strategy adaptation by competing subpopulations. In Y. Davidor, H.-P. Schwefel, and R. Manner, editors, *Proceedings of the* 3^{rd} *Conference on Parallel Problems Solving from Nature*, pages 199–208. Springer-Verlag, LNCS 866, 1994.

[113] M. Schoenauer and Z. Michalewicz. Evolutionary computation at the edge of feasibility. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Proceedings of the* 4^{th} *Conference on Parallel Problems Solving from Nature*, pages 245–254. Springer-Verlag, LNCS 1141, 1996.

[114] M. Schoenauer and Z. Michalewicz. Boundary operators for constrained parameter optimization problems. In Th. Bäeck, editor, *Proceedings of the* 7^{th} *International Conference on Genetic Algorithms*, pages 322–329. Morgan Kaufmann, 1997.

[115] M. Schoenauer and Z. Michalewicz. Sphere operators and their applicability for constrained parameter optimization problems. In B. Porto, editor, *Proceedings of the* 7^{th} *Annual Conference on Evolutionary Programming*, LNCS. Springer Verlag, 1998.

[116] M. Schoenauer, M. Sebag, F. Jouve, B. Lamy, and H. Maitournam. Evolutionary identification of macro-mechanical models. In P. J. Angeline and Jr K. E. Kinnear, editors, *Advances in Genetic Programming II*, pages 467–488, Cambridge, MA, 1996. MIT Press.

[117] M. Schoenauer and S. Xanthakis. Constrained GA optimization. In S. Forrest, editor, *Proceedings of the* 5^{th} *International Conference on Genetic Algorithms*, pages 573–580. Morgan Kaufmann, 1993.

[118] A.C. Schultz and J.J. Grefenstette. Improving tactical plans with genetic algorithms. In *Proceedings of IEEE Conference on Tools for AI*, pages 328–334. Morgan Kaufmann, 1990.

[119] H.-P. Schwefel. *Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik*. Technical University of Berlin, 1965. Dipl.-Ing. Thesis.

[120] H.-P. Schwefel. Numerische optimierung von computer-modellen mittels der evolutionsstrategie. *Interdisiplinary systems research*, 26, 1977.

[121] H.-P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, New-York, 1981. 1995 – 2^{nd} edition.

[122] H.-P. Schwefel. Collective phenomena in evolutionary systems. In *31*st *annual meeting of the int'l society for general system research*, volume 2, pages 1025–1033, Budapest, 1987.

[123] D.F. Shanno and K.H. Phua. Numerical experience with sequential quadratic programming algorithms for equality constrained nonlinear programming. *ACM, Transcations on Mathematical Software*, 15(1):49–63, 1989.

[124] A. Smith and D. Tate. Genetic optimization using a penalty function. In S. Forrest, editor, *Proceedings of the* 5^{th} *International Conference on Genetic Algorithms*, pages 499–503. Morgan Kaufmann, 1993.

[125] N. Srinivas and K Deb. Multi-objective optimization using non-dominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1995.

[126] P.D. Surry, N.J. Radcliffe, and I.D. Boyd. A multi-objective approach to constrained optimization of gas supply networks. In T. Fogarty, editor, *Proceedings of the AISB-95 Workshop on Evolutionary Computing*, volume 993, pages 166–180. Springer Verlag, 1995.

[127] P.D. Surry, N.J. Radcliffe, and I.D. Boyd. The comoga method: Constrained optimization by multi-objective genetic algorithms. *Control and Cybernetic*, 3(26), 1997.

[128] G. Syswerda. Uniform crossover in genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the* 3^{rd} *International Conference on Genetic Algorithms*, pages 2–9. Morgan Kaufmann, 1989.

[129] Astro Teller and Manuela Veloso. PADO: A new learning architecture for object recognition. In Katsushi Ikeuchi and Manuela Veloso, editors, *Symbolic Visual Learning*, pages 81–116. Oxford University Press, 1996.

[130] Bruce Tidor. An analysis of selection procedures with particular attention paid to proportional and boltzmann selection. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufman.

[131] R.J Vanderbei. *Linear Programming: Foundations and Extensions*. Kluwer Academic Publishers, Boston, 1996.

[132] D. Waagen, P. Diercks, and J. McDonnell. The stochastic direction set algorithm: A hybrid technique for finding function extrema. In D. B. Fogel and W. Atmar, editors, *Proceedings of the 1*st *Annual Conference on Evolutionary Programming*, pages 35–42. Evolutionary Programming Society, 1992.

[133] P. Wolfe. Convergence conditions for ascent methods. *SIAM Review*, (11):116–135, 1969.

[134] X. Yao and Y. Liu. Fast evolutionary programming. In L. J. Fogel, P. J. Angeline, and T. Bäck, editors, *Proceedings of the 5^{th} Annual Conference on Evolutionary Programming*, pages 451–460. MIT Press, 1996.

[135] X. Yin and N. Germay. A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization. In *Artificial Neural Nets and Genetic Algorithms*, pages 450–457, Wien, 1993. Springer Verlag.

[136] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation 8(2)*, pages 173–195, 2000.

[137] E. Zitzler and L. Thiele. Multi-objective optimization using evolutionary algorithms: A comparative case study. In A.-E. Eiben and T. B editors, *Proceedings of the* 5^{th} *Conference on Parallel Problems Solving from Nature*, pages 292–301. Springer-Verlag, LNCS 1498, 1998.

[138] E. Zitzler and L. Thiele. Multi-objective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[139] G. Zoutendijk. *Methods of Feasible Directions.* Elsevier, 1960.