Un état des lieux de l'optimisation évolutionnaire et de ses implications en sciences pour l'ingénieur

Rodolphe Le Riche, Marc Schoenauer, Michèle Sebag

Un état des lieux de l'optimisation évolutionnaire et de ses implications en sciences pour l'ingénieur

Rodolphe Le Riche
CNRS UMR 5146 et Ecole des Mines de St-Etienne
leriche@emse.fr

Marc Schoenauer
TAO et INRIA Futurs
Marc.Schoenauer@inria.fr

Michèle Sebag
TAO et CNRS UMR 8623
Michele.Sebag@lri.fr

Résumé

Les algorithmes évolutionnaires (AEs) ont aujourd'hui 40 ans d'existence ; depuis 15 ans, ils ont fait l'objet d'applications nombreuses et variées dans le domaine des sciences de l'ingénieur.

Cet article fait un état des lieux de l'optimisation évolutionnaire. Il commence par une introduction à l'optimisation évolutionnaire. Puis un fil conducteur historique résume les principaux résultats théoriques sur les AEs (les schémas, le "No Free Lunch THeorem", la corrélation performance-distance, la variance de formae, l'adaptation des pas de mutations) et montre un changement de perception des AEs : à l'utopie des AEs optimiseurs universels succèdent les AEs sur-mesure. L'article se termine par une présentation des AEs modernes les plus efficaces. L'optimisation paramétrique (les stratégies d'évolution en particulier), non-paramétrique et les couplages entre AEs et heuristiques y sont discutés. Des exemples en conception optimale et en identification de lois de comportement illustrent notre propos.

Table des matières

1 Objectifs de cet article

Les algorithmes évolutionnaires (AEs), dont les algorithmes génétiques sont les représentants les plus connus, ont aujourd'hui 40 ans ; depuis 15 ans, ils ont été intensivement appliqués aux problèmes d'optimisation mal posés dans le domaine des sciences de l'ingénieur. A ce stade de leur histoire, il ne s'agit plus d'un axe de recherche à la mode, nourri par une métaphore biologique. L'expérience acquise permet un certain recul sur les avantages et les limites des algorithmes évolutionnaires, sur les leçons à tirer des applications couronnées de succès (et des autres), et sur les spécificités des AEs par rapport aux autres approches d'optimisation globale.

Cet article dresse un état des lieux partiel de l'optimisation évolutionnaire. Il commence par une introduction aux AEs (section 2) résumant les principaux concepts. En section 3, un fil historique – l'utopie de l'optimiseur universel, sa fin et la spécialisation des AEs – est suivi pour présenter et discuter la théorie des schémas, le théorème du No Free Lunch et quelques outils pour la conception d'AEs sur-mesures. Dans la section 4, nous faisons un point sur les optimiseurs évolutionnaires paramétriques généralistes représentés principalement par les stratégies d'évolution (ES). L'histoire de l'adaptation du pas des mutations gaussiennes nous guidera depuis la règle des $1/5^{\text{ièmes}}$ jusqu'à la méthode la plus aboutie aujourd'hui d'adaptation de la matrice de covariance (CMA-ES), en passant par

l'auto-adaptation. Une sous-partie sera aussi consacrée aux méthodes par estimation de densités (EDAs), discussion qui est motivée par l'intérêt pédagogique de la comparaison entre EDAs, ES, autres AEs et l'optimisation stochastique en général. Enfin, la section 5 traite de l'optimisation évolutionnaire non-paramétrique à travers des exemples issus de la mécanique.

Principales notations

$\cdots \sim \ldots$	...suit la loi ...
$\propto$	Proportionnel à.
$*$	Symbole désignant n'importe quel élément de $\{1, \ldots, A\}$.
A	Cardinalité de l'alphabet.
AG(s)	Algorithmes Génétiques.
AE(s)	Algorithmes Evolutionnaires.
C	Matrice de covariance pour la mutation multi-normale.
CMA-ES	Covariance Matrix Adaptation - ES.
$d(x, y)$	Distance entre x et y.
ES	Evolution Strategies (stratégies d'évolution).
f, f_i	Fonction(s) coût(s) à minimiser.
F	Performance à maximiser (fitness).
$\bar{F}(\Omega)$	Moyenne de la performance F sur tous les éléments d'un ensemble Ω.
FDC	Fitness Distance Correlation.
g	Contrainte d'optimisation, $g(x) \leq 0$.
λ	Nombre d'enfants.
μ	Taille de la population (de parents).
n	Nombre de variables.
$\mathcal{N}(\mu, C)$	Loi (multi)-normale de moyenne μ et covariance C.
p_c	Probabilité de croisement par point enfant.
p_m	Probabilité de mutation par composante.
$p_s(x)$	Probabilité de sélection de x.
p^t	Densité des bons points à l'itération t.
$P(\omega)$	Probabilité de l'évènement ω.
$\mathcal{P}^t$	Population à l'itération t.
$\mathcal{S}$	Espace des variables d'optimisation (Search space).
ρ	Nombre d'individus soumis au croisement.
σ, σ_i	Pas des mutations multi-normales (σ_i^2 val. propres de C).
τ	Taille des tournois (sélection).
$U[0, 1]$	Loi uniforme entre 0 et 1.
$V(x)$	Voisinage de x.
x	Un point dans $\mathcal{S}$
x_i	i-ième composante de x.
x^*, $\hat{x}^*$	Un optimum global et son estimation.

2 Introduction aux algorithmes évolutionnaires

2.1 Contexte

On considèrera dans cet article le problème (mono-objectif) d'optimisation globale

$$\left\{ \begin{array}{l} x^* \in \mathcal{S} \\ x^* = Argmin_{x \in \mathcal{S}} f(x) \end{array} \right. \quad (1)$$

où f est la *fonction objectif* ou *fonction coût*, à minimiser, définie sur l'espace de recherche $\mathcal{S}$ et à valeurs dans $\mathbb{R}$. Alternativement, nous formulerons parfois le problème comme la maximisation de la performance (aussi appelée adaptation

ou *fitness*) $F : \mathcal{S} \rightarrow \mathbb{R}^+$, qui correspond à un problème d'optimisation équivalent[1]. La seule hypothèse faite sur $\mathcal{S}$ est qu'il s'agit d'un espace topologique, i.e. sur lequel est définie une notion de voisinage. Cette hypothèse est nécessaire pour définir la notion de *solutions locales* du problème d'optimisation, i.e. de points $\bar{x} \in \mathcal{S}$ tels que :

$$\begin{cases} \exists V(\bar{x}), \text{ voisinage de } \bar{x}, \\ \bar{x} = Argmin_{x \in V(\bar{x}) \cap \mathcal{S}} f(x) \end{cases} \tag{2}$$

Par opposition, les solutions de (1) sont appelées *solutions globales* du problème d'optimisation.

Nous distinguerons deux grandes classes de tels problèmes d'optimisation : lorsque l'espace de recherche $\mathcal{S}$ vérifie l'une des relations d'inclusion ci-dessous, où $\mathcal{D}$ désigne un ensemble discret fini :

$$\mathcal{S} \subset \mathbb{R}^n \qquad \text{ou } \mathcal{S} \subset \mathcal{D}^m \qquad \text{ou } \mathcal{S} \subset \mathbb{R}^n \times \mathcal{D}^m, \qquad n, m \in \mathbb{N} \tag{3}$$

on parlera d'*optimisation paramétrique*. Les composantes d'un vecteur $x \in \mathcal{S}$ sont les valeurs prises par les variables d'optimisation, prises dans un espace $\mathcal{S}$ continu (un compact de $\mathbb{R}^n$) ou discret ($\mathcal{D}^n$) ou mixte (continu et discret). Dans le cas contraire (il n'existe aucun entier n ou couple d'entiers (n, m) tels que (3) soit vérifiée, on parlera d'*optimisation non paramétrique*.

Dans le cas paramétrique, pour s'affranchir de problèmes liés à la numérisation, on suppose que les optima globaux x^* de (1) sont tels que $f(x^*) > -\infty$ et, dans les cas continus, que les optima globaux ne sont pas isolés[2], d'une part pour ne pas se focaliser sur des problèmes trompeurs très particuliers, et d'autre part pour permettre des démonstrations de convergence vers l'ensemble des optima globaux.

Le sujet de cet article est la résolution de (1) au moyen d'algorithmes évolutionnaires[3] (AEs).

Les AEs ne font pas d'hypothèses particulières sur les propriétés de f ou $\mathcal{S}$. En particulier, dans la formulation continue, on n'impose ni différentiabilité, ni continuité, ni régularité (i.e., f Lipschitzienne) ni convexité sur f. Cette absence d'hypothèse n'est pas une caractéristique propre des AEs, mais une propriété des optimiseurs stochastiques, e.g., le recuit simulé [85]. En conséquence, les algorithmes d'optimisation stochastique et particulièrement les AEs ont un très large champ d'application. Le prix à payer est celui d'un temps de calcul généralement élevé, dû au grand nombre d'évaluations de la fonction performance nécessaire pour trouver des optima de bonne qualité.

Notons que l'efficacité des AEs (en termes de qualité des solutions trouvées et de vitesse de convergence) est très dépendante du problème considéré ainsi que de la mise en œuvre de l'algorithme (cf. Section 3.3).

2.2 Structure générale d'un optimiseur évolutionnaire

2.2.1 La métaphore darwinienne

Les algorithmes évolutionnaires sont des méthodes d'optimisation stochastiques dans lesquelles un ensemble de points dans $\mathcal{S}$ est produit à chaque itération au

[1] La fonction F est définie par ré-écriture de f ; par exemple

$$F(x) = -f(x) + \max_{y \in \mathcal{S}} f(y) + \varepsilon, \ \varepsilon > 0$$

[2] En termes mathématiques, on ne cherche que les optima globaux essentiels pour lesquels,

$$f^* \ = \ \min\{y \mid \forall \varepsilon > 0 \ , \ v(x \in S | f(x) < y + \varepsilon) \ > \ 0\}$$

où v est une mesure d'ensemble, par exemple $v(E) \ = \ \text{Volume}(E)/\text{Volume}(\mathcal{S})$.

[3] Plus exactement, la résolution de (1) **entre autres** au moyen d'optimiseurs évolutionnaires car, comme nous le verrons en Section 6, l'utilisation conjointe d'autres optimiseurs est courante.

individu, phénotype	x
chromosome, génotype	codage de x
gène	une composante du codage de x
allèle	valeur affectée à un gène (à une composante)
fitness, performance, adaptation	$F(x)$
population	$\mathcal{P} = \{x^1, \ldots, x^\mu\} \in \mathcal{S}^\mu$
génération	passage de la population $\mathcal{P}^t$ à l'instant t à $\mathcal{P}^{t+1}$

TAB. 1 – La métaphore darwinienne.

moyen d'opérations stochastiques (cf. exemple en FIG. 1).

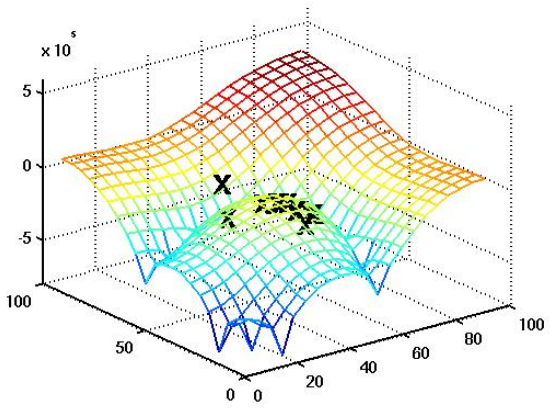


FIG. 1 – Les optimiseurs évolutionnaires utilisent un ensemble (une "population") de points à chaque itération. Dans cet exemple, la fonction performance présente des optima locaux. Elle est issue d'un problème de conception mécanique : la maximisation de la rigidité A_{11} d'une plaque composite pénalisée pour que le module de Poisson soit $0.48 \leq \nu \leq 0.52$, en fonction des variables x_1 et x_2 qui sont les orientations des fibres du matériau, cf. [59].

Il est usuel d'introduire les AEs au moyen de la métaphore "darwinienne" : Un point x de $\mathcal{S}$ est assimilé à un *individu* d'une espèce. La théorie (très simplifiée) de Darwin est que les espèces évoluent en étant soumises à deux mécanismes, les *variations aveugles* lors de la reproduction, et la *sélection naturelle* qui favorise les individus les plus adaptés, permettant ainsi à terme l'émergence d'espèces adaptées à leur environnement. Lorsque les individus sont des points de $\mathcal{S}$ et que l'adaptation à l'environnement est quantifiée par la valeur de la fonction objectif f, une telle évolution peut alors être vue comme la résolution du problème d'optimisation (1). Les principaux termes de cette métaphore sont récapitulés en TAB. 1. Plus généralement, on trouvera dans la littérature des AEs des allusions fréquentes à des concepts de la biologie, de la génétique et de l'évolution (e.g., les niches de population [99], les chromosomes diploïdes [55], ...). Nous utiliserons souvent cette métaphore car elle facilite l'explication et peut même stimuler l'intuition. Cependant, nous garderons présent à l'esprit qu'en aucun cas elle ne peut justifier un quelconque choix algorithmique.

2.2.2 Squelette d'un algorithme évolutionnaire

La FIG. 2 décrit l'organigramme d'un algorithme évolutionnaire type, commun à la plupart des instances classiques d'AEs, tels que les algorithmes génétiques (AG, cf. Section 3.1), les stratégies d'évolution (ES, cf. Section 4.2) et les algorithmes par estimation de densités (EDA, cf. Section 4.1). Pour simplifier la présentation cependant, l'organigramme de la FIG. 2 ne rend compte explicitement ni des AEs dont la population est de taille variable [131], ni des AEs utilisant une mémoire (ou archive, [93]), ni des AEs dont la population est structurée en sous-populations (par exemple dans les implémentations parallèles, [101], ou pour la prise en compte des contraintes, [97, 109]).

L'état de la recherche à l'instant t est contenu dans les μ individus de la population $\mathcal{P}^t$ et leurs performances. La population initiale, $\mathcal{P}^0$, peut être initialisée par tirage aléatoire dans $\mathcal{S}$, ou par tirage en contrôlant certaines mesures de la population (comme le nombre de 1 dans les chaînes binaires, [78]), ou enfin en choisissant certains individus (ce qui revient à injecter de la connaissance a priori). Les itérations sont composées d'opérations de sélection et/ou de remplacement des individus, et d'opérations de variation. Des exemples de procédures de sélection, remplacement, croisement et mutation sont donnés dans les paragraphes 2.2.3 et 2.2.5.

Le principe d'un optimiseur évolutionnaire est le suivant : la sélection et le remplacement sont des tirages parmi les individus de $\mathcal{P}^t$ biaisés par la fonction performance de manière à favoriser les individus ayant une meilleure performance (sélection "naturelle"). Les individus sélectionnés aussi appelés *parents* sont soumis aux opérateurs de "variation aveugle" (i.e. indépendants de la performance), le croisement et la mutation dans les AEs les plus classiques (voir section 2.2.5) ou la mise à jour de la distribution p^t puis son échantillonnage dans les EDAs (voir section 4.1), de manière à produire λ nouveaux points, les *enfants*. Le rôle du croisement est de permettre aux enfants d'hériter rapidement des caractéristiques bénéfiques de leurs parents (*your brain and my beauty*) ; le rôle de la mutation est d'assurer la diversité de la population et l'exploration de l'espace de recherche (voir Section 2.3). Comme l'illustre la FIG. 3, les opérateurs de variation travaillent sur les génotypes, i.e., dans $\mathcal{S}$, et sont spécifiques à l'espace de recherche ; les opérateurs d'initialisation, de croisement et de mutation sont dits *opérateurs génotypiques*. A l'inverse, la sélection et le remplacement sont des opérations basées uniquement sur les phénotypes (les f) et sont indépendantes de la représentation du problème traité.

Les opérateurs de variation définissent des transitions stochastiques dans $\mathcal{S}^\mu$, transitions qui sont biaisées vers les bons individus par les opérateurs phénotypiques. Trois critères d'arrêt peuvent être composés pour stopper l'optimisation : un nombre maximum de calculs de la fonction performance (ou tout autre mesure du temps d'exécution), un nombre maximum d'itérations sans amélioration du meilleur point trouvé et une diversité minimale de la population[4].

Pour compléter cette présentation, les paragraphes suivants donnent des exemples d'opérateurs de sélection et de variation usuels, sans entrer dans le détail d'une analyse entre différentes versions possibles. Bien qu'il existe des versions standards d'algorithmes évolutionnaires (e.g., CMA-ES pour l'optimisation paramétrique continue, cf. Section 4.2), dès lors que le problème sort du cadre de l'optimisation paramétrique, il n'est d'autre choix que de créer un AE spécifique au problème considéré à partir d'opérateurs génotypiques définis sur l'espace de recherche considéré et intégrés dans l'organigramme modèle de FIG. 2. Les exemples donnés ici peuvent donc être vus comme des composants à assembler. Cet article fournit ainsi, dans

[4]La diversité de la population peut être mesurée dans l'espace des f, e.g., $\sigma_f / f^{\text{ref}}$, où σ_f est une estimation de l'écart type des f de $\mathcal{P}^t$ et f^{ref} est une valeur de normalisation. Alternativement, la diversité peut être mesurée dans $\mathcal{S}$.

l'ensemble de ses sections, quelques règles pour construire un algorithme efficace.

Le lecteur désirant compléter cette introduction peut se référer aux livres introductifs aux AEs suivants : [56] est un livre historique, même si l'optimisme issu de la théorie des schéma est depuis apparu peu fondé (cf. paragraphe 3.1). Une excellente introduction aux AEs modernes, et toujours d'actualité, est fournie par [102]. Mais l'ouvrage le plus complet et le plus à jour à recommander aujourd'hui pour une introduction en profondeur aux AEs est le livre de Eiben et Smith (2003) [45]. Citons enfin le tout récent livre de K. DeJong [76] qui donne une vision unifiée du domaine.

$t \leftarrow 0$.
Initialiser la population $\mathcal{P}^0 \subset \mathcal{S}^\mu$.
Evaluer $\mathcal{P}^0$ (calculs des f).
Tant que continuer$(\mathcal{P}, t)$
    $t \leftarrow t+1$
    Sélection.
    Variation (croisement et mutation, ou mise à jour puis échantillonnage d'une distribution p^t) $\Rightarrow \lambda$ enfants.
    Evaluation des enfants.
    Remplacement de certains parents par certains enfants.
Fin.

FIG. 2 – Organigramme générique d'un algorithme évolutionnaire avec μ "parents" et λ "enfants". $\mathcal{P}^t$ est la population à l'itération t. Dans le cas des EDAs (cf. Section 4.1), p^t est une estimation de la densité des meilleurs points à l'itération t.

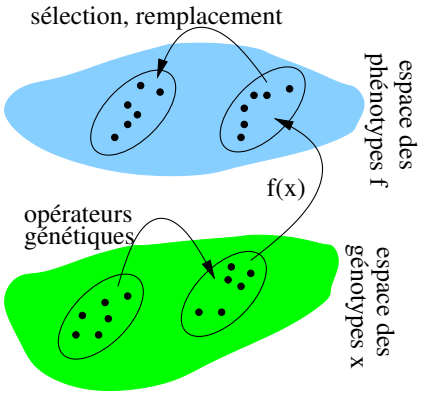


FIG. 3 – Dans le cadre de la métaphore évolutionnaire, l'espace d'application des opérateurs de variation, et l'espace de recherche sur lequel est calculée la fonction objectif (et donc dans lequel a lieu la sélection) sont qualifiés de "génotypiques" et de "phénotypiques", respectivement (d'après Lewontin, 94).

2.2.3 Opérations phénotypiques : sélection et remplacement

La sélection effectue un tirage biaisé par la performance (la fonction coût f) dans une certaine population. L'idée de la sélection en probabilité est qu'il peut être bénéfique pour l'optimisation de garder une probabilité non nulle de sélectionner des individus de performance moyenne voire mauvaise. En effet, dans les problèmes

difficiles à optimiser, il y a généralement une mauvaise corrélation entre la valeur de la performance et la distance aux optima (cf. Section 3.4.1). Il peut donc y avoir un avantage à ne pas sélectionner de manière déterministe à partir de f.

L'exemple le plus classique de sélection est le **tournoi** de taille τ. Son pseudo-code est le suivant :

```
SelectionTournoi(P)     % En général, P ⊂ S^μ,
                        % population de parents
   Choisir τ individus par tirage uniforme sans remise dans P
   Retourner l'individu de performance optimale
Fin
```

La procédure est répétée dans les AEs où plusieurs individus sont sélectionnés. Par exemple, dans les algorithmes génétiques, on appliquerait[5] `SelectionTournoi` λ fois sur $\mathcal{P}^t$ (tirages avec remises) pour produire λ enfants.

La sélection par tournoi présente deux intérêts. D'une part, elle n'est pas sensible aux *valeurs* de performance des individus en compétition, mais uniquement à l'ordre induit. Cela permet de conserver une pression de sélection[6] constante même quand, par exemple, les individus de la population deviennent de plus en plus semblables. D'autre part, on peut régler la pression de sélection par l'intermédiaire de τ : si $\tau = \mu$, seul le meilleur individu est sélectionné. A l'extrême inverse, si $\tau = 1$, la sélection par tournoi devient un tirage aléatoire. $\tau = 2$ est une valeur recommandée en moyenne [9]. Il existe de nombreuses autres variantes de sélections. Nous citerons celles basées sur la valeur de la performance (e.g., la roue de loterie des AGs, [69]), et celles dont le différentiel de pression de sélection entre individus suit une loi de Boltzmann à la manière du recuit simulé [100].

Le **remplacement**, comme son nom l'indique, remplace certains des parents par certains des enfants en fonction de leurs performances f respectives. Les stratégies d'évolution (ES, cf. Section 4.2) n'utilisent que le remplacement, et non la sélection, pour biaiser la recherche. Dans les ES, le remplacement est déterministe et peut prendre deux formes, notées (μ,λ)-ES et (μ+λ)-ES. Dans le remplacement (μ,λ)-ES, les μ meilleurs individus parmi les λ enfants[7] constituent la population suivante $\mathcal{P}^{t+1}$. $\mathcal{P}^t$ n'est pas conservée. Il est ainsi possible que le meilleur individu de la population $t+1$ soit moins bon que le meilleur individu de la population t. Notons que la perte accidentelle du meilleur individu courant est difficile à supporter pour l'utilisateur, particulièrement si le coût d'évaluation de la performance est élevé[8]. En revanche, il s'agit d'un trait favorisant l'optimisation globale ou l'auto-adaptation (cf. section 4.2.2) : il faut accepter que la meilleure performance puisse décroître d'une population à la suivante pour avoir des garanties d'atteindre l'optimum global. Le remplacement (μ+λ)-ES construit $\mathcal{P}^{t+1}$ en gardant les μ meilleurs individus de $\mathcal{P}^t$ et des λ enfants. Ainsi, si les meilleurs individus de $\mathcal{P}^t$ restent compétitifs par rapport aux enfants, ils seront conservés (c'est la propriété d'"élitisme"). (μ+λ)-ES est typiquement recommandé lorsque le coût de calcul de la fonction performance

[5] Historiquement, les AGs utilisent non pas le tournoi mais une sélection proportionnelle à la performance (tirage à la roulette, *roulette wheel*) [69]. Voir Section 3.1.

[6] La pression de sélection est l'espérance du nombre de descendants dans la population $\mathcal{P}^{t+1}$ du meilleur individu de $\mathcal{P}^t$. Elle est liée au temps de généralisation (take-over time, [8]), qui est l'espérance du nombre d'itérations pour que le meilleur individu d'une population initiale ait entièrement rempli la population sous la seule application de la sélection (sans opérateurs de variation).

[7] Il faut que le nombre d'enfants λ soit supérieur au nombre de parents μ. En pratique, le ratio recommandé est $\lambda/\mu = 7$ [9].

[8] Dans le cas où le meilleur individu de $\mathcal{P}^{t+1}$ est moins bon que celui de $\mathcal{P}^t$, on emploie souvent l'heuristique dite d'*élitisme*, remplaçant le pire individu de la population $\mathcal{P}^{t+1}$ par le meilleur de $\mathcal{P}^t$.

est élevé, mais avec plus de risque de ne donner comme solution qu'un optimum local.

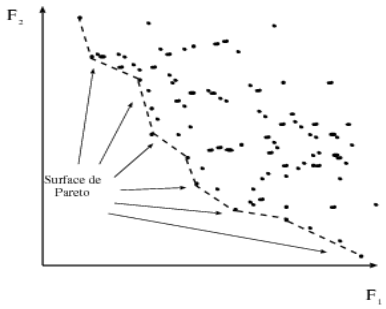
2.2.4 Opérations phénotypiques : sélection multi-objectifs

De nombreux problèmes du monde réel sont en fait des problèmes multi-critères, dans lesquels on cherche à optimiser plusieurs objectifs contradictoires. On appelle *front de Pareto* du problème l'ensemble des points de l'espace de recherche tels qu'il n'existe aucun point qui est strictement meilleur qu'eux sur tous les critères simultanément. Il s'agit de l'ensemble des meilleurs compromis réalisables entre les objectifs contradictoires. C'est cet ensemble de points qui constituent **les** solutions du problème d'optimisation multi-objectifs. Or, la souplesse des algorithmes évolutionnaires a permis la mise au point de méthodes efficaces de résolution de problèmes multi-objectifs, au point que ce domaine constitue aujourd'hui une niche riche de succès pour ces algorithmes. On consultera pour les détails les deux ouvrages [38, 31] et les actes des conférences spécifiquement dédiées à ce domaine [140, 50, 24]. Nous allons rapidement ici rappeler les principes de l'optimisation multi-objectifs évolutionnaire, qui peuvent se résumer en une phrase : un algorithme évolutionnaire multi-objectifs est un algorithme tel que défini en section 2.2 (voir Figure 2), mais dans lequel les étapes darwiniennes utilisent la notion de dominance au sens de Pareto en lieu et place de la comparaison usuelle des performances dans le cas mono-objectif.

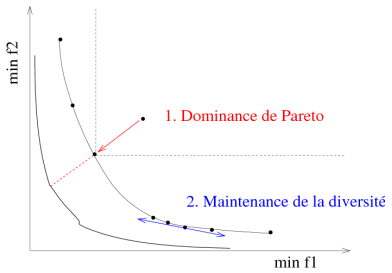
Dominance au sens de Pareto Soient $f_1, \ldots, f_n$ les objectifs à minimiser sur l'espace de recherche $\mathcal{S}$.
Définition : Soient x et y deux points de $\mathcal{S}$. On dira que x *domine par* y au sens de Pareto, noté $x \succ y$ si

$$\forall i \in [1, n], f_i(x) \leq f_i(y)$$
$$\exists i_0 \in [1, n], f_i(x) < f_i(y)$$

La figure 4-a donne un exemple de front de Pareto : l'ensemble de l'espace de recherche est représenté dans l'espace des objectifs, et les points extrémaux pour la relation de dominance au sens de Pareto forment le front de Pareto du problème (notez que l'on n'est pas toujours dans une situation aussi régulière que celle présentée figure 4, et que le front de Pareto peut être concave, discontinu, ...)



(a) Les points extrémaux de l'espace de recherche forment le front de Pareto du problème.



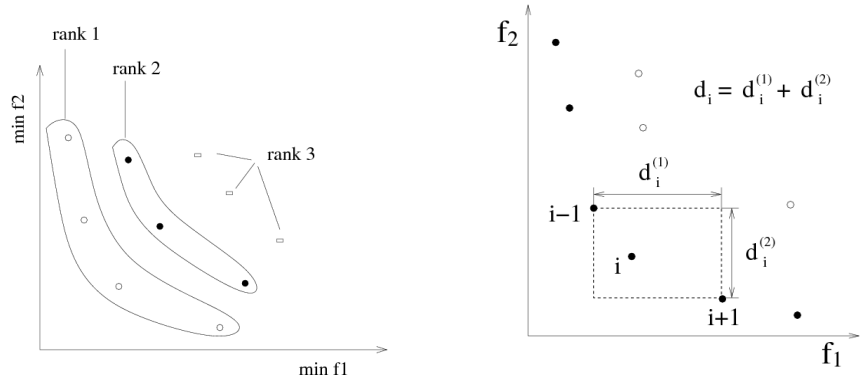
(b) Comparaison au sens de Pareto, en deux temps.

FIG. 4 – *Bases de l'optimisation évolutionnaire au sens de Pareto.*

Un exemple de sélection de Pareto Pour faire d'un algorithme évolutionnaire un algorithme multi-objectifs, il suffit, comme nous l'avons dit plus haut, de rem-

placer les procédures de sélection (sélection et remplacement de la Figure 2) par des procedures basées sur la notion de dominance au sens de Pareto. Cependant, la relation d'ordre définie par la dominance étant une relation d'ordre partiel, il faut rajouter une procédure de choix secondaire entre individus non comparables au sens de Pareto : afin de favoriser la couvertue par la population du front de Pareto, on utilise un critère basée sur la diversité.

Nous n'allons ici donner qu'un exemple d'une telle procédure, une des plus populaires aujourd'hui, renvoyant à [38, 31] pour une présentation exhaustive.



(a) Ranking de Pareto.

(b) Mesure de surpeuplement.

FIG. 5 – *Les deux critères de comparaison de l'algorithme NSGA-II.*

La procédure utilisée par l'algorithme NSGA-II [39] est basée sur la notion de tournoi, et compare deux individus de la manière suivante.

- On compare tout d'abord leur *rang de Pareto*, défini itérativement de la manière suivante (voir Figure 5-a) : les individus non-dominés de la population courante sont dits de rang 1. Ils sont retirés de la population, et la procédure est itérée pour obtenir les rang 2, 3, ... L'individu avec le rang le plus faible est préféré à l'autre. Cependant, même ainsi relaxée, la comparaison au sens de Pareto est un ordre partiel, et on fait appel en cas d'ex-aequo à un critère basé sur la diversité.
- la *distance de peuplement* est définie comme suit (voir Figure 5-b) :
  - pour chaque objectif i, on ordonne la population par valeurs croissantes de l'objectif
  - pour chaque individu p, on définit d_i, *distance de peuplement partielle selon l'objectif* i comme la somme des distances de p à ses deux plus proches voisins dans la liste ordonnée
  - la distance de peuplement totale D_p est donnée par la somme, sur l'ensemble des objectifs, des distances partielles.

  On choisit alors celui des deux individus qui a la distance de peuplement la plus faible, i.e., qui est le plus isolé dans l'espace des objectifs.

Un exemple de résultats obtenus avec cet algorithme est donné section 5.1.5.

2.2.5 Opérations de variations génotypiques

La description générale qui précède est illustrée dans le cas de trois espaces de recherche usuels, concernant respectivement l'optimisation de variables réelles, de variables discrètes, et de permutations. Le lecteur ne perdra pas de vue que les AEs

permettent de proposer des opérateurs de variation spécifiques à un problème et un espace de recherche ad hoc (cf. Section 5). Par opérateurs de variation, nous entendons ici un ensemble croisement, produisant x', puis mutation produisant un "enfant" x^e (les densités p^t sont traitées en Section 4.1). L'opérateur de croisement s'applique à un ensemble de ρ points préalablement sélectionnés, soit directement par l'opérateur de sélection, soit préalablement par le remplacement (les ρ points sont alors tirés au hasard dans la population courante $\mathcal{P}^t$). Certains AEs n'utilisent que la mutation et pas le croisement. Dans ces cas, x' est la copie d'un individu sélectionné.

L'objectif général du croisement est de recombiner des caractéristiques de points préalablement sélectionnés pour créer de nouveaux points. En particulier, on demande au croisement de transmettre aux enfants les caractéristiques communes aux ρ individus sélectionnés (propriété de "respect", [113]). L'objectif de la mutation est d'effectuer des perturbations stochastiques de la population courante. Sans la mutation, l'action combinée de la sélection (ou/et du remplacement) et du croisement aboutit à une uniformisation rapide de la population autour d'un point dépendant de la population initiale, ce qui ne peut pas constituer un processus d'optimisation globale. Une telle uniformisation de $\mathcal{P}$ hors des optima globaux est appelée **convergence prématurée**.

Variables continues

Considérons tout d'abord les variables réelles, $x \subset \mathbb{R}^n$. Un exemple de croisement entre ρ points $x^1, \ldots, x^\rho$ préalablement sélectionnés produit le point x' suivant[9]

$$x'_i \;=\; \sum_{j=1}^{\rho} \beta_j x_i^j \quad , \quad i = 1, n \tag{4}$$

avec

$$\beta_j \geq 0 \quad , \quad \sum_{j=1}^{\rho} \beta_j = 1 \left(\text{ par expl., } \beta_j = \frac{\alpha_j}{\sum_{k=1}^{n} \alpha_k} \quad , \quad \alpha_k \sim U[0,1] \right) . \tag{5}$$

Notons que x' se trouve ici dans l'enveloppe convexe des ρ parents ; ceci peut, en réduisant l'enveloppe convexe de la population d'une génération à la suivante, compromettre les chances de trouver l'optimum global. Pour éviter ce comportement contractant du croisement, il est conseillé de tirer α_k uniformément dans $[-.5, 1.5]$ [102].

x' est ensuite muté pour donner un point "enfant", x^e, typiquement au moyen d'une loi multi-normale centrée sur x' et de covariance C,

$$x^e \;=\; x' + \mathcal{N}(0, C) \tag{6}$$

Le choix de C, qui induit un pas moyen et des directions privilégiées dans $\mathcal{S}$ est discuté en Section 4.2. Si les composantes sont mutées indépendamment, C est une matrice diagonale avec $C_{ii} \;=\; \sigma_i^2$ [9].

Variables discrètes

Soit maintenant le problème en variables discrètes où chaque x_i, $i = 1, n$, appartient à un alphabet {1,2,...,A}. Le croisement entre $x^1,\ x^2\ , \ldots, x^\rho$ peut prendre la forme suivante (c'est le **croisement uniforme** ou "ρ-ary discrete recombination" de [126]). Avec la probabilité p_c, faire

$$x'_i \;=\; x_i^j \qquad \text{où } j \text{ est tiré uniformément dans } \{1, ..\rho\} \tag{7}$$

[9]Cet opérateur de croisement est donné ici sous une forme générale. Quand $\rho = 2$, il correspond au "panmictic generalized intermediate recombination" de [9]. Quand $\beta_j = 1/\rho$, il correspond au "global intermediate recombination" de [126].

Si le croisement n'a pas lieu (avec probabilité $1 - p_c$), copier un des ρ individus en x'. Chaque composante x'_i peut ensuite être mutée indépendamment avec une probabilité p_m, par exemple,

$$x^e_i \;=\; j \quad , \quad j \neq x'_i \quad , \text{ avec probabilité } 1/(\text{A-1}) \; . \tag{8}$$

Ceci correspond à une mutation sans corrélation entre composantes (les mutations corrélées sont le sujet des EDAs, cf. par exemple [107, 59]).

Permutations
Enfin, considérons brièvement l'optimisation de permutations. Les permutations sont des listes ordonnées de n éléments distincts. Il y a $n!$ permutations possibles. L'exemple le plus simple d'optimisation de permutations est le problème du voyageur de commerce qui consiste en la détermination du plus petit tour reliant n villes données. Les permutations apparaissent dans de nombreux autres problèmes d'ordonnancement et présentent des difficultés de mise en œuvre particulières pour l'optimisation évolutionnaire : il devient non trivial de proposer un **codage** des permutations et un croisement produisant des permutations valides. Par exemple, si l'on code les permutations sous la forme d'une liste ordonnée, puis on cherche à croiser les deux permutations suivantes (l'une est écrite en majuscules, l'autre en minuscules) par le croisement uniforme des alphabets discrets, on pourrait par exemple obtenir

$$\begin{aligned} x^1 = &\quad \texttt{[A B C D E F G H I]} \\ x^2 = &\quad \texttt{[d f e b a g c h i]} \\ x' = &\quad \texttt{[A f e D a g G H I]} \end{aligned}$$

qui n'est pas un tour valide (deux a, pas de b, ...). Un des plus anciens et plus simples opérateurs de croisement pour les permutations codées sous forme de liste est le croisement ordonné de Davis [36] : *i)* On sélectionne deux points de coupures au hasard dans les parents x^1 et x^2. *ii)* Les éléments de x^1 situés entre les points de coupures sont copiés dans x'. *iii)* A partir du second point de coupure et en retournant au début de la liste quand la fin est atteinte, trouver les prochains éléments de x^2 qui n'apparaissent pas dans x'. Les copier dans x', toujours en tournant à partir du second point de coupure.

$$\begin{aligned} &i) \quad x^1 = \quad \texttt{[A B C | D E F G | H I]} \\ &i) \quad x^2 = \quad \texttt{[d f e | b a g c | h i]} \\ &ii) \quad x' = \quad \texttt{[- - - | D E F G | - -]} \\ &iii) \quad \text{éléments non copiés : } \texttt{h i b a c} \\ &iii) \quad x' = \quad \texttt{[b a c | D E F G | h i]} \end{aligned}$$

Ce croisement préserve les positions relatives et absolues de x^1 entre les points de coupures et les positions relatives de x^2 hors des coupures. Il existe de nombreux autres opérateurs de croisement pour les permutations. Par exemple, l'opérateur de recombinaison de coins ("edge recombination", [138]) paraît particulièrement adapté au problème du voyageur de commerce car il préserve les arcs (portions de tours dont dépend la performance totale).

La mutation la plus simple d'une permutation est le retournement de l'ordre entre deux positions prises au hasard (opérateur 2-opt de [98]), par exemple les positions 3 et 6,

$$x^e = \quad \texttt{[b a | F E D c | G h i]} \; .$$

2.2.6 Caractéristiques des optimiseurs évolutionnaires

Les méthodes d'optimisation évolutionnaires possèdent donc quatre caractéristiques :

1. L'état d'un AE à une itération donnée est représenté par une population de points, par opposition aux méthodes procédant par évolution d'un point unique (e.g., programmation mathématique [105], recherche gloutonne, recuit simulé). Ceci confère aux AEs une plus grande fiabilité dans les fonctions où il existe des optima locaux puisque la recherche a lieu dans le volume de $\mathcal{S}$, par opposition aux trajectoires dans $\mathcal{S}$ que parcourent les optimiseurs à un point. Par contre, cette fiabilité a un coût numérique, celui de l'évaluation de μ points plutôt qu'un, et il n'est pas évident qu'à même nombre d'évaluations de f, une méthode évolutionnaire soit plus efficace que plusieurs recherches locales avec redémarrage aléatoire. Il existe cependant des fonctions pour lesquelles l'évolution d'une population conduit à une convergence plus rapide que l'évolution d'un point unique. Comparons par exemple dans $\mathbb{R}^n$ un $(\mu+\lambda)$-ES, $\mu > 1$, et un (1+1)-ES avec mutation gaussienne et sans croisement sur une fonction avec topologie en tunnel telle que la fonction tunnel-sphère de la Fig. 6 : dès que l'(1+1)-ES a atteint le tunnel, il est condamné à progresser en suivant ce tunnel car tout autre point aurait une moins bonne performance. Cette progression peut être extrêmement longue si le tunnel est étroit. Le $(\mu+\lambda)$-ES peut lui progresser hors du tunnel, ce qui est beaucoup plus facile, tant que ses μ individus ne sont pas dedans. Notons que les topologies "en tunnel" ne sont pas purement académiques, mais reflètent bien les fonctions coût pénalisées de l'optimisation sous contraintes, l'intérieur du tunnel correspondant aux régions de satisfaction des contraintes. Pour un même nombre d'évaluations de la fonction performance, l'intérêt des approches à base de populations par rapport aux méthodes fondées sur l'évolution d'un seul point a été prouvé pour l'optimisation de fonctions bruitées [4] et montré empiriquement pour l'auto-adaptation (cf. [21] et Section 4.2.2).
2. Les opérateurs de variation, opérateurs génotypiques, doivent être conçus en rapport avec le choix du codage des solutions, l'espace des génotypes. Le seul exemple explicite de codage que nous ayons vu jusqu'à présent est l'optimisation de permutations (cf. Section 2.2.5) où les permutations étaient représentées comme des listes ordonnées. Auparavant, nous avions implicitement codé les variables réelles comme un vecteur de $\mathbb{R}^n$ et les variables discrètes comme des listes dans un alphabet de cardinalité A. D'autres représentations existent, par exemple celles basées sur des codages binaires (cf. Section 3.1). Il est fréquent et recommandé d'explorer plusieurs représentations pour un même problème ; l'avantage des AEs est que le changement de représentation ne remet pas en cause l'approche d'optimisation, mais demande simplement le développement d'opérateurs d'initialisation, de croisement et de mutation adaptés.
3. Les AEs se déplacent dans $\mathcal{S}$ par des transitions en probabilités[10]. Ainsi, deux exécutions d'un AE sont susceptibles de ne pas donner le même résultat. De plus, les AEs ont besoin d'un certain nombre de calculs de points avant que la machinerie des probabilités ne puisse se mettre en place, ce qui induit un coût numérique qui peut paraître exorbitant dans certaines applications. Pour donner un ordre de grandeur, on peut estimer qu'il n'y a pas d'optimisation évolutionnaire à moins de 1000 calculs de la fonction coût. Par contre, les sauts en probabilités sont le moteur par lequel les AEs convergent asymptotiquement (i.e., quand $t \to \infty$) vers l'ensemble des optima globaux (sous des

[10] Les AEs peuvent être vus comme des chaînes de Markov sur $\mathcal{S}^\mu$ où les opérateurs de sélection, remplacement et variation définissent des transitions en probabilités entre les états que sont les populations [135].

conditions sur les intensités minimales de sélection et mutation et la taille de population, cf. [27, 122]).

4. Les AEs sont des optimiseurs d'ordre 0, i.e., ils n'utilisent que la valeur de la fonction coût, mais ni son gradient, $\partial f/\partial x_i$, ni son hessien. Cette dernière caractéristique étend le champ d'application des AEs aux nombreux problèmes pratiques où soit le gradient n'existe pas, soit il n'est pas connu.

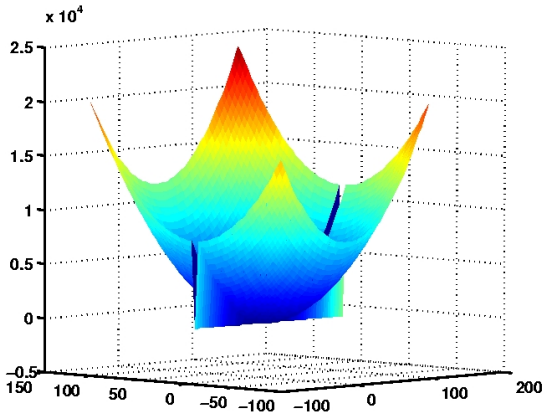


FIG. 6 – La fonction tunnel-sphère (à minimiser) en 2D. $f(x) = x^T x$ si $x \in \mathcal{T}$, $f(x) = -\exp(-||x||/10)$ sinon, où le tunnel est $\mathcal{T} = \{x \in \mathcal{S} \mid -0.5 \leq x_i \leq 0.5\ ,\ i \in [2, n]\}$. Tous les points à l'intérieur du tunnel ont une meilleure fonction coût que ceux hors du tunnel. La population permet à l'AE de progresser hors du tunnel, même si un point est dedans, ce qui facilite sa progression vers $x^* = 0$.

2.3 Réglage d'un AE : le compromis exploration-exploitation

Tout optimiseur global réalise un compromis entre l'exploration de nouvelles régions de $\mathcal{S}$ et l'exploitation de connaissances déjà acquises sur $\mathcal{S}$. Les connaissances acquises correspondent typiquement à l'identification des meilleures régions parmi les régions déjà partiellement explorées. L'exploration est nécessaire à la détermination d'optima globaux mais elle implique également de nombreux calculs de f. L'archétype de la recherche exploratoire est l'échantillonnage aléatoire de $\mathcal{S}$ avec conservation du meilleur point (méthodes de Monte-Carlo). L'exploitation conduit à une convergence accélérée, mais qui aboutit souvent à des optima locaux (convergence prématurée).

L'ensemble des paramètres des AEs règlent le compromis exploration-exploitation réalisé par l'algorithme. L'accroissement de l'intensité de mutation (p_m, valeurs propres de C) et de la taille de population (μ) contribue à augmenter le comportement exploratoire. Réciproquement, l'accroissement de la pression de sélection (τ), du nombre d'enfants (λ) et de la probabilité de croisement (p_c) rend l'AE plus exploitateur. Dans les AEs, un réglage trop en faveur de l'exploitation se traduit typiquement par une convergence prématurée.

Il n'existe pas de réglage optimal d'un AE en général car le bon réglage dépend de la fonction f optimisée et de l'objectif de l'optimisation (trouver un optimum global à ...% près après ...calculs de f et avec ...% de chances). En règle générale, plus la fonction f est difficile à optimiser pour l'AE, plus l'accent doit être mis sur l'exploration et vice versa. En ce qui concerne l'effet du nombre de dimensions n sur les paramètres de l'AE, des études théoriques et empiriques indiquent les ordres de grandeurs suivants :

- Taille minimale de population pour des alphabets binaires, $\mu \sim \mathcal{O}(n)$ [52, 27].
- Probabilité de mutation pour des alphabets binaires, $p_m \sim \mathcal{O}(1/n)$ [7].
- Pas des mutations multi-normales, $\sigma_i \sim \mathcal{O}(1/n)$, où σ_i^2 sont les valeurs propres de C [125].

3 Une perspective historique

Les algorithmes évolutionnaires regroupent des approches aux noms et aux origines différentes :

- Les algorithmes génétiques ont été imaginés et analysés par J. Holland en 1975 [69] en vue, dans un premier temps, de modéliser l'adaptation. Ils ont été utilisés comme optimiseurs par K. DeJong [42], puis popularisés par D. Goldberg avec la parution de son livre [56]. On lira dans cette perspective l'article et le récent ouvrage de K. DeJong [41, 76].
- Les stratégies d'évolution ont été mises au point par deux élèves ingenieurs à Berlin en 1965, I. Rechenberg [119] et H.-P. Schwefel dont l'ouvrage de référence date de 1981 [125]. Th. Bäck est un des premiers à avoir fait le lien entre stratégies d'évolution et algorithmes génétiques en 1995 [9].
- La programmation évolutionnaire date du début des années 60, et est née en Californie avec les travaux de L.J. Fogel [49]. Tout d'abord centrée sur l'évolution d'automates à états finis, cette technique a ensuite été généralisée à des représentations quelconques par D. B. Fogel au début des années 80 [47]. D. B. Fogel est par ailleurs le premier à avoir utilisé le terme "Evolutionary Computation" dans son livre [48].
- La programmation génétique est apparue plus récemment, et tout d'abord comme une forme particulière d'algorithmes génétiques qui faisait évoluer des programmes représentés sous forme d'arbre [33]. Elle a surtout été popularisée par les travaux de J. Koza (92) [88, 89, 90].
- L'optimisation statistique, ou EDAs (voir section 4.1), enfin, est apparue plus récemment avec les travaux de S. Baluja [11] et H. Mühlenbein [107].

Ces différents AEs ont aujourd'hui, après plus de 35 ans d'histoire et des milliers d'applications dans tous les domaines, en grande partie fusionné, et un des principaux artisans de cette unification a été Z. Michalewicz avec son livre [102] dans lequel il démontre bien que le principe de ces algorithmes (qu'il appelle encore "Evolution Programs" en 1992) peut s'appliquer à toute représentation. Les AEs partagent aujourd'hui plus de quatre conférences internationales annuelles ou bisannuelles (GECCO et CEC organisées depuis les USA, PPSN et Evo* organisées en Europe) ainsi qu'une conférence française à vocation internationale, *Evolution Artificielle*. Il existe trois revues exclusivement consacrées au sujet (*Evolutionary Computation, IEEE Trans. on Evolutionary Computing* et *Genetic Programming and Evolvable Machines*) qui sont dédiées au domaine, mais on trouvera aussi de nombreuses contributions dans *Complex Systems, Journal of Heuristics, BioSystems, Natural Computing, J. of Global Optimization*, et *Theoretical Computer Science (C)*. L'histoire des AEs peut être découpée en deux grandes périodes, la première qui a duré jusqu'au milieu des années 90 où l'on était à la recherche d'un optimiseur universel, performant sur tous types de problèmes, suivie de la période contemporaine de spécialisation des AEs durant laquelle la communauté a pris conscience de la nécessité de prise en compte de connaissances du domaine.

3.1 Le mythe de l'algorithme génétique, optimiseur universel

La recherche d'un optimiseur universel depuis les années 70 jusqu'au milieu des années 90 s'est articulée autour de trois thèmes, la représentation, les rôles respectifs

du croisement et de la mutation, et les paramètres optimaux de l'algorithme.

L'algorithme incarnant le mieux cette période, et d'ailleurs celui qui retenait le plus l'attention, était l'algorithme génétique (AG, [69, 75, 56]). L'AG classique a pour espace de recherche l'ensemble des chaines binaires de dimension n, $\mathcal{S} \equiv \{0,1\}^n$. Ce choix, comme nous le verrons ci-dessous, s'appuie sur la théorie des schémas. La sélection est effectuée proportionellement à la performance F. La probabilité de sélection de $x^i \in \mathcal{P}^t$ est

$$p_s(x^i) \;=\; \frac{F(x^i)}{\sum_{x^j \in \mathcal{P}^t} F(x^j)} \;. \tag{9}$$

Deux individus parents sont tirés avec remise dans la population courante $\mathcal{P}^t$ avec probabilité p_s. Leur croisement génère deux autres points qui sont ensuite mutés et ajoutés à la population enfant. Ce cycle sélection/croisement/mutation est répété $\mu/2$ fois jusqu'à compléter la population $\mathcal{P}^{t+1}$. Dans la version élitiste de l'AG, le meilleur individu de la population précédente est recopié dans $\mathcal{P}^{t+1}$ (il remplace un des enfants au hasard, avant évaluation). Le croisement est le plus souvent un croisement à un point dans lequel un point de rupture est choisi aléatoirement dans les chaînes parents et les sous-portions de chaînes sont échangées. Par exemple,

$$\begin{array}{l}[0 \Updownarrow 1\ 1\ 1] \\ [1 \Updownarrow 1\ 0\ 0]\end{array} \;\Rightarrow\; \begin{array}{l}[0\ 1\ 0\ 0] \\ [1\ 1\ 1\ 1]\end{array} \;.$$

La mutation est celle décrite pour les variables discrètes en Section 2.2.5.

L'analyse des performances des AGs s'appuie sur le théorème des schémas [69]. Un **schéma** $\mathcal{H}$ est un sous-ensemble de $\{0,1\}^n$ que l'on note en utilisant le symbole $*$ pour signifier "0 OU 1". Par exemple,

$$\mathcal{H} \;=\; [0\ *\ 1\ *\ 1] \;=\; \{[0\ 0\ 1\ 0\ 1],\ [0\ 1\ 1\ 0\ 1],\ [0\ 1\ 1\ 1\ 1],\ [0\ 0\ 1\ 1\ 1]\} \;.$$

L'ordre du schéma, $o(\mathcal{H})$, est le nombre de bits fixés ($o(\mathcal{H}) \;=\; 3$ dans l'exemple). La longueur du schéma $\ell(\mathcal{H})$, est la distance maximale entre bits fixés ($\ell(\mathcal{H}) \;=\; 4$ dans l'exemple). Notons $N(\mathcal{H},t)$ le nombre d'individus de la population $\mathcal{P}^t$ appartenant au schéma $\mathcal{H}$, et $\bar{F}(\Omega)$ la performance moyenne des individus de Ω. Le théorème des schémas montre que l'espérance du nombre d'individus de $\mathcal{H}$ dans la population $\mathcal{P}^{t+1}$ croît en fonction du nombre d'individus de $\mathcal{H}$ dans $\mathcal{P}^t$, de l'avantage en performance de $\mathcal{H}$, et décroit en fonction de l'ordre et de la longueur de $\mathcal{H}$, qui le rendent respectivement plus fragile sous les effets de la mutation et du croisement [69] :

$$E[N(\mathcal{H},t+1)] \;\geq\; N(\mathcal{H},t)\frac{\bar{F}(\mathcal{H}\cap\mathcal{P}^t)}{\bar{F}(\mathcal{P}^t)}\left[1 - p_c\frac{\ell(\mathcal{H})}{n-1}\right][1-p_m]^{o(\mathcal{H})} \;. \tag{10}$$

Le terme $\frac{\bar{F}(\mathcal{H}\cap\mathcal{P}^t)}{\bar{F}(\mathcal{P}^t)}$ modélise les effets d'amplification de la sélection par tirage à la roulette. Le terme $\left[1 - p_c\frac{\ell(\mathcal{H})}{n-1}\right]$ modélise les effets destructeurs du croisement à un point, si le point de coupure intervient entre les bits fixés du schéma. Par exemple, $\mathcal{H} \;=\; [1\ *\ 0\ *]$ est perdu par croisement de $[1\ 0 \Updownarrow 0\ 1]$ et $[0\ 1 \Updownarrow 1\ 1]$. Enfin, le terme $[1-p_m]^{o(\mathcal{H})}$ modélise les effets destructeurs de la mutation ; si la mutation d'un individu porte sur l'un des bits fixés l'individu sort du schéma.

Ce théorème a donné naissance à l'idée de **briques élémentaires** (ou "building blocks"), qui sont des schémas courts ($o(\mathcal{H})$ et $\ell(\mathcal{H})$ petits) et performants. Pour des schémas courts et une faible probabilité de mutation p_m, les deux derniers termes de l'Equation (10), qui correspondent aux destructions de schémas par croisement et mutation, peuvent être négligés. Si un schéma est durablement performant, i.e.,

$$\frac{\bar{F}(\mathcal{H}\cap\mathcal{P}^{t+k})}{\bar{F}(\mathcal{P}^{t+k})} > 1+\varepsilon \quad , \quad \varepsilon > 0 \;,\; k = 0 \ldots T \tag{11}$$

alors le nombre de représentants des briques élémentaires dans la population croit rapidement (géométriquement),

$$E[N(\mathcal{H}, t+T)] > N(\mathcal{H}, t)(1+\varepsilon)^T \ . \tag{12}$$

A la condition que la mutation soit peu utilisée, les AGs permettraient donc l'émergence rapide des briques élémentaires dans la population, que le croisement devrait ensuite assembler pour mener à x^*. De plus, chaque individu est le représentant d'un grand nombre de schémas (2^n dans un alphabet binaire, $*$ ou sa valeur de bit en chaque position) dont la performance est estimée à travers un unique calcul de fonction coût. Il y aurait donc une économie de calculs sous-jacente aux AGs, un "parallélisme implicite" [69].

3.2 Les débats des années 90

Les schémas et les arguments séduisants qui en découlent ont donné jusque dans les années 90 l'espoir d'obtenir des algorithmes performants sur tous les problèmes en moyenne. Mais, rapidement, les contradictions entre résultats empiriques et théorie des AGs ont donné lieu à des débats sur la représentation et l'importance respective des opérateurs de croisement et mutation.

3.2.1 La représentation

Un argument souvent donné [69, 56] est que le codage binaire maximise la proportion des schémas instanciés par un individu et les effets du parallélisme implicite. Cependant, il a été montré par Antonisse que cet argument est faux[11] [3].

Il semble que les partisans du codage binaire aient été intuitivement attirés par l'idée qu'un alphabet de cardinalité minimale réalise une décomposition maximale[12] de l'information présente dans x. Puisqu'il n'existe pas d'entité informative plus petite que le bit, l'alphabet binaire permet de révéler toutes les briques élémentaires d'un problème. La difficulté est de trouver parmi les nombreux ($2^n!$) codages binaires possibles, celui favorisant les schémas performants les plus compacts (les briques élémentaires). A cet effet, des travaux ont été réalisés qui visent à faire optimiser les codages par l'AE lui même en codant la signification du gène sur l'allèle. Nous citerons l'inversion [69] et les AGs désordonnés ("Messy GAs", [53, 54]).

En général cependant, les représentations "naturelles", adaptées au problème, se sont avérées plus efficaces que les codages binaires. Pour un problème donné, les briques élémentaires ont souvent une définition évidente. Il est préférable de les manipuler directement à travers un codage ad hoc. Descendre à une échelle de description plus fine (binaire) fait perdre cette connaissance du problème. Ainsi, les stratégies d'évolution (ES, [125, 9, 66]) qui manipulent directement des vecteurs de nombres réels sont les AEs les plus performants pour l'optimisation de variables continues quoique les nombres réels puissent naturellement être représentés par des chaines binaires. En optimisation topologique en mécanique du solide, de multiples représentations ont été comparées dans [82, 83, 61] (cf. aussi Section 5.1). Les

[11] En effet, le nombre de schémas auquel un individu donné appartient est 2^n. Dans un alphabet de cardinalité A, le nombre de schémas est $(A+1)^n$; la fraction des schémas illustrée par un individu serait donc maximisée pour $A = 2$. L'erreur est ici qu'il y a plusieurs symboles $*$ dans un alphabet de cardinalité $A > 2$ si l'on veut compter tous les sous-espaces. Par exemple, pour $A = 3$, on a $*_{0,1}$, $*_{0,2}$,$*_{1,2}$,$*_{0,1,2}$. De plus, il faut réaliser la comparaison entre cardinalités à taille d'espace $\mathcal{S}$ constante, $2^{n_2} = A^{n_A}$. Sous ces deux corrections, les individus d'alphabets de cardinalité $A > 2$ échantillonnent une plus grande proportion du nombre total de sous-espaces (schémas) que dans le cas binaire.

[12] Par décomposition maximale, nous entendons le codage non redondant ayant le plus grand nombre de composantes, n. Si la représentation est vue comme une application $\psi : \mathcal{C} \to \mathcal{S}$, où $\mathcal{C}$ est l'espace des codages, un codage est non redondant quand ψ est bijective.

représentations binaires y sont moins efficaces que, par exemple, celles basées sur des cellules de Voronoï car elles transmettent mal les caractéristiques des topologies lors du croisement et elles ne permettent pas de raffiner localement la description de forme pour un n fixé.

3.2.2 Croisement versus mutation

L'autre débat de fond dans la communauté des AEs dans les années 80 à 90 est celui de l'importance relative du croisement et de la mutation.

Le Théorème des schémas (Equation (10)) décrit la croissance des briques élémentaires sous l'hypothèse d'une utilisation marginale de la mutation, dont le rôle est uniquement d'éviter la convergence prématurée. Le croisement est le principal opérateur de recherche assurant la recombinaison des briques élémentaires[13]. Forts de cette analyse, les algorithmes génétiques (AGs) ont des probabilités de mutation de l'ordre de $p_m = 0.01$ (en $\mathcal{O}(1/n)$) et des probabilités de croisement $0.6 \leq p_c \leq 1$.

Au contraire, les stratégies d'évolution et la programmation évolutionnaire [49, 47] utilisent principalement la mutation pour localiser x^*. Les premières versions d'ES et de programmation évolutionnaire dans les années 60 et 70 ne comportaient d'ailleurs pas de croisement. De la même manière que les AGs regardent la mutation comme un opérateur de réparation (ré-injection de briques élémentaires perdues du fait des erreurs d'échantillonnage dans la population), les ES considèrent le croisement comme un opérateur de réparation. Celui-ci sert à extraire les similarités des bons individus de la population pour réparer l'excès de bruit introduit par les mutations, ce qui augmente la vitesse de convergence [18, 20].

3.3 La fin du mythe

Vers 1995, les retours d'expériences et des éléments d'analyse montrent qu'un optimiseur universel ne peut pas exister.

3.3.1 Les retours d'expérience

L'essor des AEs dans les années 90 a induit un grand nombre d'études et d'applications dont les résultats ont souvent été en désaccord, au moins partiel, avec la théorie des schémas. Par exemple, dans [95, 96], une optimisation des paramètres de l'AG par essais-erreurs est effectuée et conclut, pour l'optimisation de stratifiés composites, que des populations petites ($\mu \approx 10$) et d'importantes probabilités de mutations produisent le meilleur AG. Petit à petit, l'idée de spécialisation de l'AE en fonction du problème considéré s'impose.

3.3.2 Critique des schémas

Dans le même temps, il devenait clair que deux hypothèses manquaient à la théorie des schémas pour qu'elle puisse s'appliquer. D'une part, l'estimation de la performance des bons schémas dans la population courante, $\bar{F}(\mathcal{H} \cap \mathcal{P}^t)$, est une approximation de leur performance $\bar{F}(\mathcal{H})$ dont la précision diminue avec la dispersion de $F(x),\ x \in \mathcal{H}$. Le théorème des schémas n'est descriptif que pour les problèmes à faible variance de schémas. Cette discussion est poursuivie en Section 3.4.2.

D'autre part, dans les problèmes difficiles, non-linéaires, les bons schémas ne contiennent pas nécessairement l'optimum global x^*. L'hypothèse selon laquelle la recombinaison de schémas courts et performants mène à l'optimum est presque une hypothèse de linéarité. En pratique, les problèmes difficiles à optimiser sont

[13] A noter cependant que ce rôle constructif du croisement n'est absolument pas pris en compte par le théorème des schémas.

trompeurs [137] : pour les alphabets binaires, un problème est trompeur à l'ordre k si tous les schémas d'ordre k de meilleure performance moyenne ne contiennent pas x^*. Par exemple, un problème à 3 bits tel que $\bar{F}([0\ *\ *]) > \bar{F}([1\ *\ *])$, $\bar{F}([*\ 0\ *]) > \bar{F}([*\ 1\ *])$, $\bar{F}([*\ *\ 0]) > \bar{F}([*\ *\ 1])$, et $x^* = [1\ 1\ 1]$ est trompeur à l'ordre 1.

3.3.3 Le théorème du No Free Lunch

En 1995, un résultat théorique est venu conforter les observations selon lesquelles il n'y avait pas un AE meilleur que les autres en général, mais plutôt un AE optimal par problème, le théorème du "No Free Lunch", (NFL, [139]). Wolpert et McReady montrent qu'*en moyenne, sur tous les problèmes d'optimisation, le comportement de n'importe quel algorithme est le même.* Ce résultat est établi sur des problèmes en variables discrètes mais de cardinalité arbitraire. L'interprétation directe du NFL est que, ce qu'un algorithme d'optimisation a gagné sur un problème (par rapport à tous les autres algorithmes) est perdu sur un autre problème. Bien entendu, il faut se garder d'interprêter naïvement le NFL en concluant qu'il est inutile d'améliorer les méthodes d'optimisation puisqu'elles ne feront jamais mieux que, par exemple, une recherche aléatoire. Ce théorème est en effet établi sous l'hypothèse que tous les problèmes d'optimisation possibles sont également probables. Dès le début de cet article, nous sommes sortis de ce cadre en nous limitant aux problèmes sans optima isolés (optima non essentiels, cf. Section 2.1). Les optimiseurs ne sont pas tous équivalents sur une classe de fonctions donnée. La recherche en optimisation, pour être pertinente, doit lier l'algorithme au problème.

3.4 Outils pour la conception des AEs

Une fois que l'idée d'adapter l'algorithme au problème a émergé, il est devenu nécessaire de pouvoir mesurer a priori l'adéquation entre les composants d'un AE (choix du codage, de la procédure d'initialisation et des opérateurs de variation) et la fonction à optimiser f. Bien sûr, il est toujours possible d'analyser cette adéquation a posteriori, par exemple en moyennant plusieurs optimisations. Cependant, l'analyse a priori est plus à même de faire comprendre quels mécanismes d'optimisation conviennent à une fonction donnée. Nous décrivons ci-après deux types d'analyse de performance a priori[14].

3.4.1 La relation entre l'AE et la performance

Un optimiseur évolutionnaire est composé d'un codage des x, d'une procédure d'initialisation de $\mathcal{P}^0$ et d'opérateurs de variation. Il est instructif de confronter la distribution des points échantillonnés par chacune de ces composantes à la distribution des performances associées.

Un outil simple pour juger de la qualité d'une procédure de codage et d'initialisation est de tracer $f(x)$ en fonction de la distance $d(x, , \hat{x}^*)$ entre x et le meilleur point $\hat{x}^*$ d'un échantillon tiré au moyen de cette procédure [77, 78]. En l'absence de connaissance a priori, on préfèrera un codage qui permette une bonne couverture du plan (f, d).

Un exemple est donné sur la FIG. 7 ; le but de l'optimisation est de retrouver une chaine s de $\{0,1\}^{900}$, la fonction $F(x)$ donnant le nombre de bits communs à x et s. De plus, le nombre de 0 figurant dans la chaine s est fixé à 100. Deux

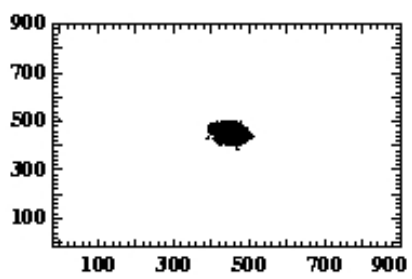
[14]Notons qu'il existe une troisième approche basée sur l'estimation de l'**épistasis**, c'est à dire l'évaluation de l'écart entre f et sa plus proche fonction décomposable, $f_d(x) = \sum_{i=1}^{n} f_i(x_i)$, [121]. L'idée est ici que plus il existe de couplages entre variables dans f, plus l'optimisation par un AE est difficile. En fait, tout dépend de l'aptitude des opérateurs de variation à identifier et décrire ces couplages.

procédures d'initialisation sont comparées sur la Figure : à gauche, chaque bit est indépendamment mis à 0 ou 1 avec probabilité 1/2. Le nombre total de 1 suit une loi binomiale et pour $n = 900$, 99% des chaînes générées ont un nombre de 1 compris entre 411 et 489. A droite, la probabilité d'apparition d'un 1 à un bit est choisie suivant $U[0, 1]$ indépendamment pour chaque individu. Le graphe montre que, pour ce problème, la seconde procédure est la plus appropriée puisqu'une plus grande diversité de (d,f) est obtenue.

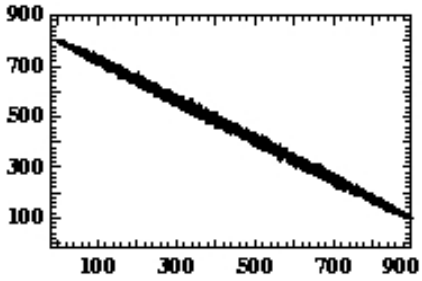
Il existe des mesures synthétisant les graphes (d,f), en particulier la corrélation performance-distance ou FDC (pour Fitness-Distance Correlation), [74]),

$$FDC = \frac{1/N \sum_{i=1}^{N} (f(x^i) - \bar{f})(d(x^i, \hat{x}^*) - \bar{d})}{\sigma_d \sigma_f} \tag{13}$$

où N est la taille de l'échantillon, $\bar{f}$, σ_f, $\bar{d}$ et σ_d sont les moyennes et écarts-types empiriques de f et d sur l'échantillon, respectivement. La fonction $OneMax$, qui associe à une chaine x le nombre de bit à 1, a une FDC de 1 par construction, ce qui correspond à un problème d'optimisation facile (chaque amélioration de la fonction coût correspond à un rapprochement de x^* et la fonction est unimodale). L'initialisation par chaîne (à droite sur la FIG. 7) capte bien la facilité du problème. On notera que la FDC ne donne pas d'informations fiables sur la difficulté de l'optimisation d'un problème par AE, sauf pour les valeurs extrêmes (1 et -1) indiquant que le problème est facile.



(a) FDC=-0.156133 pour une initialisation uniforme avec $Pr(x_i = 1) = 1/2$



(b) FDC=-0.999255 pour une initialisation avec $Pr(x_i = 1) = p_x; p_x \sim U(0, 1)$

FIG. 7 – Performance vs distance au meilleur individu d'un échantillon de 6000 points pour deux procédures d'initialisation (d'après [78]). On constate une distribution plus large avec l'initialisation par chaîne, ce qui se traduit ensuite par une convergence plus rapide vers x^*. La FDC empirique proche de -1 indique un problème facile.

Le même type de démarche est applicable au choix de couples (codage, croisement) ou (codage, mutation). Dans ces cas, il est pertinent de regarder la relation entre f du meilleur parent et f du meilleur enfant. Différentes moyennes résumant cette relation sont discutées dans [77] mais, comme pour la FDC, le graphe complet est le plus instructif.

3.4.2 La variance de formae

Une version généralisée des schémas de la Section 3.1 sont les **formae** ou classes d'équivalences générées par la relation "a la même caractéristique que" appliquée aux points de $\mathcal{S}$ [113]. Si la caractéristique en question est la valeur de certains bits

d'une chaîne binaire de taille fixée, les formae coïncident avec les schémas. Mais les formae s'appliquent également aux codages de tailles variables (dont nous parlons dans la partie 5 sur l'optimisation non paramétrique). De manière générale, les caractéristiques sont les entités élémentaires portées par le codage et recombinées par l'opérateur de croisement. Cette généralisation étant faite, les concepts utilisés ici d'ordre et de performance de formae sont similaires à ceux introduits pour les schémas en Section 3.1.

Si l'on accepte la conjecture selon laquelle les AEs procèdent par recombinaison des caractéristiques élémentaires portées par les bons individus de la population (les "building blocks"), il devient clair que l'efficacité de l'AE dépend de la possibilité d'identifier ces bonnes caractéristiques à partir d'un échantillon limité d'individus, la population. Soit $\mathcal{H}$ l'une des formae contenant l'optimum global x^*. On peut espérer que l'optimisation par un AE sera d'autant plus aisée que la performance moyenne de $\mathcal{H}$ est supérieure à celle des autres formae de même ordre (possédant le même nombre d'éléments) et ne contenant pas x^*. Mais il est également nécessaire, pour que $\mathcal{H}$ soit détecté et propagé dans la population, que la variance de la performance de $\mathcal{H}$ ne soit pas trop grande. Un principe de choix de codage pour un AE est donc de minimiser la variance des formae : voir par exemple [114], dans lequel les variances de formae de plusieurs codages du voyageur de commerce sont comparées et une bonne corrélation entre faible variance et rapidité d'optimisation par AE est constatée.

4 Optimiseurs évolutionnaires paramétriques : estimations de densités et stratégies d'évolution

Tous les algorithmes évolutionnaires paramétriques définissent à chaque itération t une densité de probabilité p^t qui sera échantillonnée pour produire les λ nouveaux points. Les algorithmes basés sur les opérateurs de croisement définissent ces densités implicitement : p^t n'est pas calculé et les enfants sont produits par le truchement des opérateurs de variation. Quand il existe une expertise sur le problème traité, comme c'est le cas par exemple en optimisation de plaques composites [95, 96], travailler directement sur les opérateurs de variation et le codage peut aider à mieux prendre en compte les connaissances a priori. Par contre, les optimiseurs évolutionnaires paramétriques généralistes bénéficient de l'utilisation explicite de p^t qui, à travers une meilleure formalisation, permet de mieux les contrôler.

Trois familles de méthodes d'optimisation paramétriques sont présentées ci-après, les algorithmes par estimation de densités, les ES auto-adaptatifs et les ES adaptatifs. Toutes ces méthodes ont en commun le fait d'apprendre et d'exploiter à chaque itération une densité paramétrée par un vecteur θ, $p^t(x) \equiv p(x|\theta^t)$ (ou $p^t(x) \equiv p(x|\theta^t(x))$ dans le cas des ES auto-adaptatifs). La différence entre ces familles de méthodes réside dans la manière dont la densité est mise à jour. Pour les algorithmes à estimation de densité, l'objectif est d'apprendre la densité maximisant la probabilité des bons points à t. Les ES adaptatifs utilisent la dynamique des bons points aux itérations successives. Dans les ES auto-adaptatifs enfin, les θ sont optimisés en même temps que les x, la qualité des θ étant évaluée indirectement à travers la performance des x qui les portent, et de leur descendance.

4.1 Algorithmes à estimation de densités

Bien que les EDAs ne soient pas aujourd'hui parmi les méthodes évolutionnaires les plus efficaces en moyenne [84], leur intérêt théorique est grand [92, 51] et justifie de leur consacrer cette section. Les EDAs permettent de situer l'optimisation évolutionnaire par rapport à d'autres approches de l'optimisation globale comme

l'optimisation stochastique et l'optimisation Bayésienne. La comparaison entre EDAs et ES est aussi révélatrice des différents mécanismes possibles d'optimisation stochastique.

Les algorithmes à estimation de densités (EDAs, pour "Estimation of Distribution Algorithms") fouillent l'espace de recherche $\mathcal{S}$ en échantillonnant et en mettant à jour itérativement une distribution p^t représentant les meilleurs points connus. Les EDAs ont d'abord été introduits dans des espaces discrets, la densité p^t servant à remplacer les opérateurs de variation (algorithme "Population Based Incremental Learning", PBIL, dans [11], "Univariate Marginal Distribution Algorithm", UMDA, dans [106]). L'organigramme type d'un EDA est le suivant.

1. Initialisation de la densité $p^0(x)$, $t = 0$.
2. Création de λ points par échantillonnage de $p^t(x)$.
3. Evaluation de F pour les λ points.
4. Sélection des μ meilleurs ($\mu < \lambda$).
5. Mise à jour de p^{t+1} au moyen des μ meilleurs points[a] et de $p^t(x)$.
6. Si continuer, $t \leftarrow t + 1$, aller en 2.

[a] Dans certaines variantes, la mise à jour utilise également l'information contenue par les *pires* points de l'échantillon courant [12, 129].

La densité p^t permet d'exprimer les connaissances acquises au cours de l'évolution [34] ; en particulier les individus $(x, F(x))$ fournissent un échantillon de la fonction F (voir cf. FIG. 8), et les hypothèses faites sur la corrélation spatiale de F permettent d'encadrer la valeur de $F(x)$ en fonction de la distance de x aux points connus. Si $\hat{F}^*$ est la valeur de la performance au delà de laquelle un point est considéré comme bon (par exemple la meilleure performance vue à l'itération t), la densité p^t peut être interprétée comme

$$p^t(x) \;=\; P(x|F(x) > \hat{F}^*) \;=\; \frac{P(F(x) > \hat{F}^*|x)P(x)}{P(F(x) > \hat{F}^*)} \tag{14}$$

où $P(x)$ est une densité a priori. Si $P(x)$ est uniforme, $p^t(x) \;\propto\; P(F(x) > \hat{F}^*|x)$. L'équation (14) a aussi l'intérêt de faire le lien entre EDAs et optimisation stochastique [123, 73]. Dans [73] par exemple, le point tiré à l'itération $t + 1$ est celui qui maximise la probabilité $P(F(x) > \hat{F}^*|x)$.

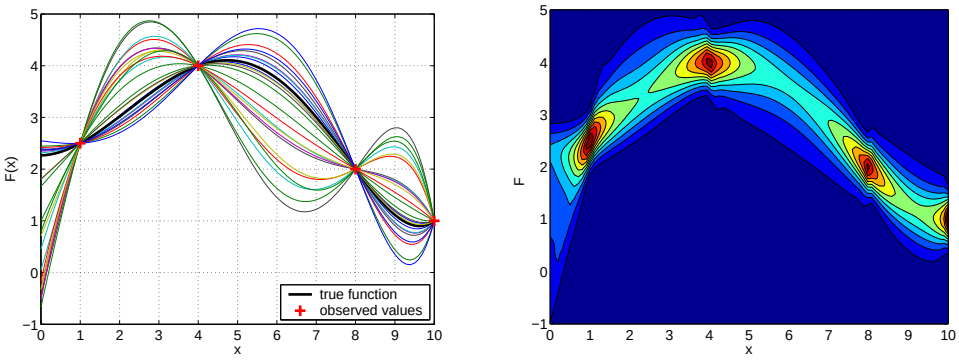
Les EDAs peuvent également être vus comme des méthodes Bayésiennes. Si l'on considère que la sélection définit une probabilité de sélection sur $\mathcal{S}$, p_s, et dans l'*idéalisation d'une population infinie*, le cumul des étapes 2 et 4 de l'organigramme des EDAs produit p^{t+1} suivant

$$p^{t+1}(x) \propto p_s^t(x).p^t(x) \;. \tag{15}$$

La FIG. 9 illustre la convergence d'un EDA dans le cas théorique où p^t est non-paramétrique et $f(x)$ est supposée connue sur tout $\mathcal{S}$, ce qui est équivalent à une population infinie.

En pratique, les densités p^t sont paramétrées par θ, $p^t(x) \;\equiv\; p(x|\theta^t)$, et les populations sont finies, ce qui induit deux types de difficultés.

Premièrement, il est nécessaire pour que l'EDA localise rapidement x^* et que sa densité converge vers une distribution uniforme sur les x^*, que sa structure lui permette de décrire les couplages entre variables x_i. Sur la FIG. 10 par exemple, les variables x_1 et x_2 sont supposées indépendantes, ce qui ne permet pas à $p(x_1, x_2) = p_1(x_1)p_2(x_2)$ de faire coincider la région de x^* et la région de plus haute probabilité, même quand un grand nombre de points échantillonnent toute la région de $\mathcal{S}$ de



(a) Fonctions compatibles avec les observations.

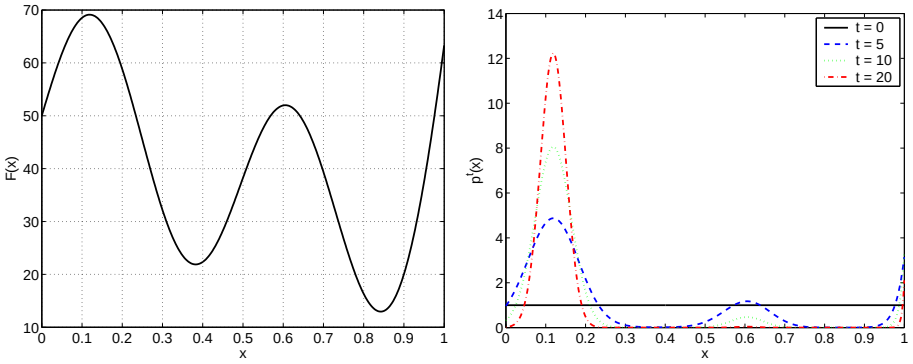
(b) Contours de $p(F|x)$. La combinaison d'une probabilité a priori sur $F(\mathbf{x})$ et des observations donne une distribution sur $\mathcal{S}$.

FIG. 8 – Représentation de l'incertitude sur les régions non échantillonnées comme une distribution $p(F|x)$ (d'après [58]).

haute performance. Cette contrainte de prise en compte des couplages a fait succéder aux premiers EDAs, fondés sur des distributions produit (PBIL [11], UMDA [106]), des algorithmes utilisant des dépendances plus complexes (e.g., FDA [107] ou BOA [110] qui utilise les réseaux Bayésiens).

Cependant, et nous arrivons maintenant à la seconde difficulté, estimer une densité demande un échantillon d'autant plus grand que l'espace $p(.|\theta)$ des distributions choisi est riche[15]. Dans les EDAs, l'échantillon servant à estimer $p(.|\theta)$ est composé des μ meilleurs points parmi les λ tirages. Numériquement, la construction de cet échantillon coûte λ calculs de f à chaque itération. Si le nombre de points μ est trop faible par rapport à la richesse de $p(.|\theta)$, $p(.|\theta^t)$ dépendra beaucoup des tirages réalisés et l'algorithme aura un comportement instable. C'est le compromis biais-variance en estimation [43, 134]. Il y a donc un arbitrage à réaliser entre la complexité de p^t et la taille des populations (donc le coût de l'optimisation). A une extrémité de cet arbitrage, certains EDAs s'appuient sur des distributions simples : par exemple PBIL_c [128] utilise une loi multi-normale sans corrélations pour les espaces continus. DDOA [59] emploie deux densités où les variables sont indépendantes dans deux espaces différents, le cumul des densités introduisant des couplages entre variables. Au contraire, les EDAs qui optent pour des structures de dépendances entre variables plus ambitieuses doivent, au delà du maximum de vraisemblance, recourir à des techniques de régularisation de l'estimation : les deux grandes approches sont la minimisation de risques régularisés et la validation croisée (cf. [43]). L'algorithme IDEA [23] modélise les densités jointes de variables continues au moyen de produits de densités conditionnelles. Pour estimer p^t, IDEA minimise la vraisemblance régularisée (pénalisée) par le nombre d'arcs du graphe des probabilités conditionnelles. Notons enfin que, pour stabiliser le comportement des EDAs, il est courant d'effectuer une combinaison linéaire entre les deux dernières estimations

[15]La richesse ou complexité d'un espace de fonctions reflète l'indépendance des valeurs qu'elle peut prendre sur un échantillon ; ainsi un échantillon de deux points suffit à décrire une fonction linéaire en dimension 2. Formellement, la notion de complexité d'une classe de fonctions est capturée par la dimension de Vapnik-Chervonenkis. Le lecteur intéressé est renvoyé à [134, 127] pour plus de détails.



(a) Exemple de fonction 1D à maximiser avec trois maxima locaux en $x = 0.11$, $x = 0.61$, et $x = 1.00$. L'optimum global est en $x = 0.11$. $F(x) = 50 - 60x^2 + 50x^5 + 20x^{10} + 20\sin(40x/\pi)$

(b) Convergence de la distribution $p(x)$ vers l'optimum global $x = 0.11$. Le processus est $p^{t+1}(x) \propto p_s^t(x)p^t(x) \propto f(x)p^t(x)$

FIG. 9 – Illustration de la convergence théorique d'un EDA (d'après [58])

de θ, et l'ancienne densité,

$$p^t(x) = p(x|(1-\alpha)\theta^t + \alpha\theta^{t-1}) \quad , \quad \alpha \in [0,1[\ . \tag{16}$$

Comme nous l'avons dit au début de la discussion sur les EDAs, ceux-ci ne sont pas, en moyenne, des optimiseurs très efficaces. Leur principe, qui est d'apprendre à chaque itération la probabilité de présence de bons points dans tout $\mathcal{S}$, fait implicitement l'hypothèse qu'une grande partie de $\mathcal{S}$ a été explorée, ce qui est irréaliste ou très coûteux dès les moyennes dimensions ($n > 4$ environ). De plus, cette densité est une photographie statique de ce que l'on sait de $\mathcal{S}$ à l'instant t, elle n'encourage pas l'exploration de nouvelles régions sinon avec de faibles probabilités (queues de distributions). Les optimiseurs efficaces, tels que les ES adaptatifs, ne prétendent pas estimer une densité de présence en tout point de $\mathcal{S}$. En exploitant la dynamique des populations sur plusieurs itérations ils peuvent, par exemple, "allonger la foulée" dans des directions où les derniers déplacements se sont avérés favorables.

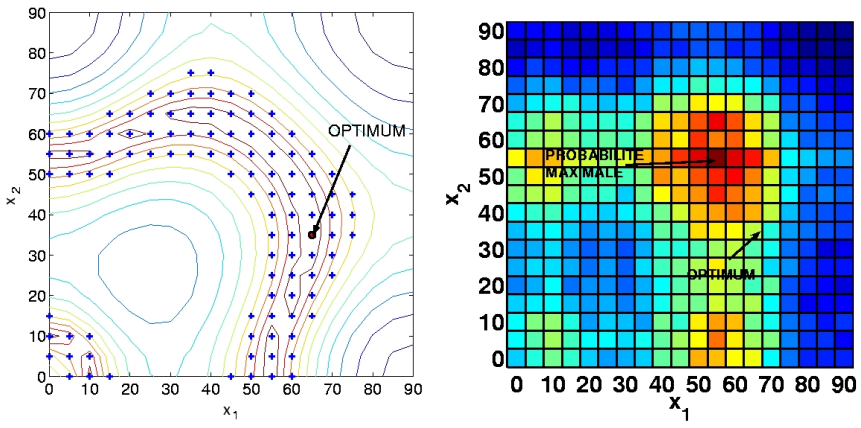
4.2 Stratégies d'évolutions

L'évolution historique des ES illustre parfaitement les différentes manières de régler les paramètres des algorithmes évolutionnaires en général, puisque le réglage de la mutation gaussienne est passé par toutes les étapes, de fixée à adaptative puis à auto-adaptative avant de revenir vers l'adaptatif.

Les ES sont des algorithmes évolutionnaires pour l'optimisation paramétrique continue ($\mathcal{S} = \mathbb{R}^n$, ou un sous-ensemble de $\mathbb{R}^n$).

4.2.1 La mutation gaussienne

L'opérateur principal des ES est la mutation gaussienne, qui ajoute à l'individu un bruit normal centré. La distribution gaussienne la plus générale en dimension n est la distribution multivariée $\mathcal{N}(m, C)$, de moyenne m et de **matrice de covariance** C, une matrice $n \times n$ définie positive. La distribution de probabilité



(a) Courbes de niveaux de f et points sélectionnés

(b) Lignes de niveaux du produits des marges

FIG. 10 – Points sélectionnés (a) et lignes de niveaux du produit des lois marginales (b) pour $(x_1, x_2) \in \{0, 5, 10, \ldots, 85, 90\}^2$ sur un problème de conception de composites en vibrations (d'après [59]). Quand la distribution des points sélectionnés est représentée par des variables indépendantes (produit des marges, $p(x_1, x_2) = p_1(x_1)p_2(x_2)$), les zones de hautes probabilités ne coincident pas forcément avec les zones de haute performance.

correspondante est

$$\Phi(X) = \frac{\exp(-\frac{1}{2}(X-m)^t C^{-1}(X-m))}{\sqrt{(2\pi)^n |C|}}$$

où $|C|$ est le déterminant de C.

Mais il est en fait plus parlant d'écrire la mutation gaussienne d'un vecteur $x \in \mathbb{R}^n$ sous la forme suivante :

$$x \rightarrow x + \sigma \mathcal{N}(0, C)$$

i.e., de distinguer un facteur d'échelle σ, aussi appelé le **pas** de la mutation, des directions principales de la distribution données par la matrice de covariance C. Par exemple, le cas le plus simple est quand C est la matrice identité : les variables sont alors mutées indépendamment par ajout de bruit gaussien de variance σ^2.

Ajuster les paramètres d'une ES revient alors à rechercher les meilleures valeurs du pas et de la matrice de covariance au cours de l'évolution. Les premiers essais ont concerné le cas isotrope dans lequel le seul degré de liberté est le pas σ.

4.2.2 Adaptation du pas

Le pas de la mutation gaussienne donne en fait l'échelle à laquelle l'algorithme "voit" la fonction objectif. Pour simplifier, supposons que la fonction objectif soit la fonction x^2 en dimension 1, et que l'algorithme soit un (1+1)-ES (un unique parent donne naissance à un enfant, et le meilleur des 2 devient le parent de la génération suivante). Alors, la distance moyenne entre deux individus successifs est proportionnelle à σ. Ceci a deux conséquences : d'abord, si le point de départ est d_0, il faudra en moyenne d_0/σ itérations pour atteindre le voisinage de 0. Ensuite, la précision maximale que l'on peut alors espérer est elle aussi proportionelle à σ. Ces

arguments conduisent naturellement à l'idée d'un paramètre *adaptatif*, c'est-à-dire ici par exemple proportionnel à la distance à la solution.

Ce type d'argument a été développé en détail dans la théorie du *taux d'amélioration* (progress rate) par Schwefel [125], puis par Beyer [20]. Récemment, Auger [6] a donné une preuve formelle des résultats sur le "progress rate" ainsi qu'une preuve de convergence de cet algorithme adaptatif ... totalement inutilisable puisqu'en pratique on ne connait pas, bien sûr, la distance à l'optimum !

Mais il est une autre information qui est elle disponible durant l'évolution, c'est le *taux de succès*, i.e., la proportion de mutations qui ont donné lieu à une amélioration de la qualité de la solution. Or, cette information donne indirectment des renseignements sur le pas courant : cette observation de Rechenberg [120] a donné la "règle des 1/5", premier exemple connu de paramètre *adaptatif*, qui dit qu'il faut augmenter σ quand le taux de succès est supérieur à 0.2 (on est trop loin de l'optimum et on fait des pas, certes dans la bonne direction, mais trop petits), et le diminuer dans le cas inverse (on "tourne autour" de l'optimum sans arriver à améliorer la performance). Cette règle dérive d'études théoriques sur des fonctions simples, mais a été généralisée à toute fonction ... avec des succès divers. En effet, elle est facilement mise en défaut sur des fonctions ayant de nombreux optima locaux, ou même sur des fonctions convexes pour lesquelles les axes principaux ne sont pas les axes des coordonnées : dans ce cas, une matrice de covariance non diagonale est nécessaire pour espérer une convergence fine. Quoi qu'il en soit, bien que plus du tout utilisée aujourd'hui, la règle des 1/5 n'en reste pas moins une étape importante dans l'évolution de la pratique des ES.

4.2.3 Auto-adaptation dans ES

L'étape importante suivante a été l'invention de la mutation *auto-adaptative* : ici, les paramètres de la mutation (le pas σ, voire même la matrice de covariance C elle-même), sont attachés à chaque individu, et soumis eux aussi à ... des mutations. On distingue trois variantes, suivant les paramètres de mutation utilisés : la mutation *isotrope* n'utilise qu'un pas σ par individu, la matrice de covariance étant alors l'identité ; la mutation *non-isotrope* attache à chaque individu un vecteur de pas $(\sigma_1, \ldots, \sigma_n)$ et la matrice de covariance est alors la matrice diagonale ayant σ_i^2 en i-ème position sur la diagonale. Les différentes variables sont ici encore mutées indépendamment, mais avec des variances différentes ; enfin la mutation *corrélée* attache à chaque individu une matrice de covariance complète. Muter un individu revient alors à muter les paramètres de la mutation, puis ensuite à muter les variables du problème (l'individu lui-même) avec les nouveaux paramètres de mutation. Les détails peuvent être trouvés dans [125, 10, 45].

L'idée sous-jacente est que l'étape de sélection, qui pourtant n'utilise que la valeur de performance des individus, va également sélectionner les individus ayant les paramètres de mutation les plus adaptés à la région de l'espace de recherche où ils se trouvent (e.g., un grand pas dans les régions de faible gradient et inversement). En effet, de deux individus ayant des performances identiques, mais des paramètres de mutation différents, celui qui a des paramètres de mutation adaptés à la fonction performance va donner en moyenne des enfants plus performants que l'autre, et la sélection va ensuite impitoyablement éliminer les enfants ayant les moins bons paramètres de mutation, du moins sur plusieurs générations. C'est ce qui a fait dire que les paramètres de la mutation sont "optimisés gratuitement" – et la mutation auto-adaptative a longtemps été l'état-de-l'art en terme d'optimisation évolutionnaire paramétrique continue [9].

Mais on peut se demander plus précisément ce que sont de "bons" paramètres de mutation dans une région donnée de l'espace de recherche. Cette question a été discutée rapidement dans la section précédente dans le cas d'un unique paramètre,

le pas σ de la mutation en dimension 1. Des arguments similaires en remplaçant la fonction "sphère" ($\sum x_i^2 \equiv X^t X$) par la fonction "ellipsoïde" ($\frac{1}{2} X^t H X$ pour une matrice définie positive H) montrent que la matrice de covariance optimale doit être proportionnelle à H^{-1}. Or, dans la variante corrélée de la mutation auto-adaptative (une matrice de covariance est attachée à chaque individu), le pas s'adapte finalement au pas optimal, [9, 40], mais la matrice de covariance qui est apprise par l'algorithme n'est pas l'inverse de la hessienne de la fonction objectif [5].

4.2.4 CMA-ES : retour à l'adaptation déterministe

Mais même lorsque l'auto-adaptation fonctionne, elle fonctionne lentement : si le pas initial n'est pas proportionnel à la distance à l'optimum, dans le cas simple de la mutation isotrope, il faut un certain nombre de générations avant qu'il le devienne et que l'algorithme devienne efficace.

C'est cette observation qui a conduit Hansen et Ostermeier à proposer une méthode déterministe d'adaptation du pas [67], puis de l'ensemble des paramètres de la mutation gaussienne [65], revenant ainsi à une méthode **adaptative** d'ajustement des paramètres de la mutation. Pour en donner une idée intuitive, si deux mutations réussies successives sont allées dans la même direction, il faut probablement agrandir le pas pour aller plus vite. Finalement, la méthode complète, appelée CMA-ES (*Covariance Matrix Adaptation*) fut proposée et surtout étudiée en profondeur (et ses paramètres par défaut soigneusement réglés) dans [66]. Enfin, une amélioration pour la mise à jour de la matrice de covariance en grande dimension fut proposée dans [64], et une variante pour l'optimisation multi-objectifs vient d'être proposée [70].

Aujourd'hui, CMA-ES est sans contestation possible la meilleure méthode évolutionnaire pour l'optimisation paramétrique continue. Elle a en particulier été grand vainqueur sur les méthodes stochastiques (autres ES, mais aussi PSO, Differential Evolution, et autres EDAs) pour l'ensemble de la compétition organisée lors de la conférence CEC'05 [133].

4.2.5 Conclusions sur ES

Ce bref résumé de l'histoire des ES illustre bien l'expérience générale acquise au cours des deux dernières décennies à propos du réglage des paramètres des AEs :

- Il est très difficile de régler des paramètres statiques – et il est possible, sinon fréquent, qu'il n'existe pas de valeur fixe optimale, comme c'est le cas pour le pas de la mutation.
- Les méthodes adaptatives, qui utilisent de l'information sur l'état courant de l'évolution, sont aussi bonnes que l'information qu'elles utilisent ! Le taux de succès est de l'information de bas niveau, liée très indirectement seulement au paramètre qu'il s'agit d'adapter, et la règle des 1/5 qui en découle est facilement prise en défaut. A l'inverse, CMA-ES utilise de l'information de haut niveau pour adapter de maniere sophistiquée tous les paramètres de la mutation, avec un succès beaucoup plus tangible.
- Les méthodes auto-adaptatives sont efficaces quand elles sont applicables, c'est-à-dire lorsque la sélection, qui n'utilise que la performance, élimine les mauvais paramètres avec les mauvais individus. Elles sont alors plus performantes que les méthodes statiques et les méthodes adaptatives de base, mais rien ne peut battre une bonne méthode adaptative ...

5 Optimiseurs évolutionnaires non paramétriques

Jusqu'à présent, cet article a essentiellement traité le cas de l'optimisation paramétrique (les individus sont des vecteurs de variables continues et/ou discrètes), en soulignant qu'il existe de nombreuses autres méthodes d'optimisation paramétrique ; comparativement, l'avantage des AEs par rapport aux autres méthodes est de s'adapter plus aisément aux spécificités de l'espace de recherche et/ou de la fonction performance − au prix d'un coût de calcul élevé.

En revanche sur le terrain de l'optimisation non-paramétrique, les AEs ont beaucoup moins de concurrents et ont connu des succès spectaculaires : voir les travaux en programmation génétiques de Koza [88, 90], et la session *"Human-competitive achievements of Evolutionary Algorithms"* tous les ans lors de la conférence GECCO ; l'ensemble des travaux présentés dans cette session illustre les avancées permises par l'optimisation non paramétrique avec les AEs.

Il existe en effet de très nombreux problèmes réels d'optimisation qui ont été modélisés sous forme d'optimisation paramétrique, souvent au prix de simplifications abusives, simplement parce que le scientifique ou l'ingénieur en charge de la modélisation ne connaissait que des méthodes d'optimisation paramétriques. Pourtant, il est souvent plus payant d'obtenir des solutions peut-être pas tout à fait optimales (du fait du caractère stochastique des AEs) pour un modèle exact que des solutions exactes (avec garanties sur l'erreur par exemple, pour les méthodes de type gradient) pour un modèle ... faux[16]. Cette situation n'est pas sans rappeler les débuts des algorithmes d'optimisation, lorsqu'on se ramenait à un problème linéaire qu'on savait résoudre, mais dont la solution était peu en rapport avec le problème non-linéaire du départ.

Or, comme il a été brièvement dit en section 2, pour lancer un algorithme évolutionnaire sur un espace de recherche $\mathcal{S}$, il "suffit" de disposer d'opérateurs de variation (et d'une initialisation, souvent négligée) sur $\mathcal{S}$ dont la sémantique est en rapport avec le problème [112]. Ceci permet de "sortir du cadre" et d'explorer des espaces de recherche jusque-là hors de portée du praticien. On peut même alors quitter le domaine de la stricte optimisation en se dirigeant vers le domaine de l'*évolution sans fin* ("open ended evolution"), dans laquelle des solutions sont améliorées en permanence suivant l'évolution de l'environnement. Une caractéristique importante des AEs prend ici tout son sens, celle d'être des algorithmes *"any time"*, c'est-à-dire capables à tout instant de donner la meilleure solution trouvée jusqu'alors.

Cette section se propose d'illustrer les capacités des AEs à traiter des problèmes d'optimisation non-paramétriques sur trois exemples. Deux concernent la conception optimale de structures mécaniques, le troisième l'identification de lois de comportement de matériaux nouveaux.

5.1 Optimisation topologique

5.1.1 Contexte mécanique

Le contexte de cette partie est l'optimisation topologique de structures. Le problème consiste à trouver la forme optimale d'une structure contenue dans un domaine donné (i.e., la répartition de matière dans ce domaine).

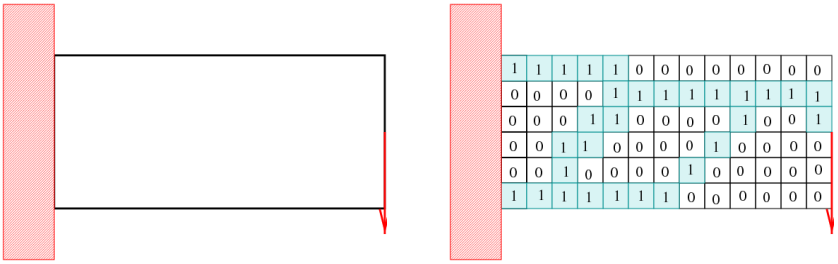
La notion d'optimalité est définie à partir de critères mécaniques d'une part (par exemple, une borne sur le déplacement maximal de la structure soumise à un chargement donné, ou des valeurs de fréquences propres maximales, ou une

[16] Selon Herbert Simon [130], "*In complex real-world situations, optimization becomes approximate optimization since the description of the real world is radically simplified until reduced to a degree of complication that the decision maker can handle.*
Satisficing *seeks simplification in a somewhat different direction, retaining more of the detail of the real-world situation, but settling for a satisfactory, rather than approximate best, decision.*"

combinaison de critères mettant en jeu la rigidité et le comportement vibratoire) et la masse de la structure d'autre part. A noter que les algorithmes évolutionnaires pourraient aussi de prendre en compte d'autres critères comme, par exemple, des coûts de fabrication fonctions du nombre de trous à percer dans la forme ou du nombre de "coins", ...

Le modèle mécanique utilisé ici est, sauf mention explicite du contraire, celui de l'élasticité linéarisée bidimensionnelle en contraintes plane, et les matériaux considérés sont linéaires et isotropes (cf., e.g. [30]). Toutes les données mécaniques sont adimensionnelles (e.g., le module d'Young vaut toujours 1) et les effets de la gravité sont négligés.

Un test très populaire pour l'optimum design et celui de la plaque-console (cantilever) : le domaine de calcul est un rectangle, la plaque est encastrée sur la partie verticale gauche de la frontière (déplacements nuls imposés) et le chargement consiste à appliquer une force ponctuelle verticale au milieu de la frontière verticale de droite. La Figure (11) montre le domaine de calcul pour le cantilever 2×1.



(a) La plaque console 2×1 (b) Représentation par tableau de bits

FIG. 11 – *Le problème de la plaque-console* 2×1 *et une représentation par tableau de bits ("bitarray") d'une structure sur un maillage régulier (ici un maillage* 13×6*).*

5.1.2 La fonction de performance

Nous allons dans la suite considérer la problème test de la plaque console, dans lequel les deux caractéristiques à optimiser sont, d'une part, la masse de la structure, et d'autre part son déplacement sous l'action d'un chargement unique, en l'occurrence une force ponctuelle appliquée au milieu du côté vertical droit.

Préliminaires mécaniques

Quelle que soit l'approche retenue, quelques précautions mécaniques sont indispensables. En effet, surtout lors des premières générations, de nombreuses structures dans la population ne seront pas mécaniquement admissibles – le point d'application de la force n'étant pas connecté à la frontière de déplacement imposé. Il ne faut donc pas tenter de calculer le comportement mécanique de telles formes, sous peine d'erreurs graves au sein du solveur utilisé.

Cependant, une astuce technique permet d'éviter d'avoir à pré-trier ces formes (avec les calculs de topologie que cela suppose) : au lieu de considérer que ce qui n'est pas la structure est vide, on le considère comme constitué d'un matériau très mou (de module d'Young de plusieurs ordres de grandeur inférieur à la "matière"). Les structures non connectées deviennent ainsi admissibles, mais donneront un déplacement très important - ce qui devrait permettre de les éliminer lors de la sélection. De plus, plus la structure sera proche d'une structure connectée, moins ce déplacement calculé sera important, et il y aura donc "gratuitement" une pression sélective vers les

structures admissibles. De plus, cette astuce permet de ne pas régénérer le maillage pour chaque structure, ce qui est un gain de temps énorme.

Il faudra cependant être très prudent avec cette approche si l'on se place dans le cadre de l'optimisation modale : en effet, des modes propres apparaissent, qui ne font intervenir que le "vide" et il faut donc les repérer et les éliminer.

Un problème sous contrainte

L'approche la plus classique consiste à considérer que l'on cherche à minimiser le poids de la structure sans que le déplacement maximal sous l'action du chargement imposé ne dépasse une valeur limite. On est ainsi face à un problème d'optimisation sous contrainte. Nous nous proposons de traiter ce problème par une technique de pénalisation adaptative.

Pour une structure donnée, appelons P son poids et D_{Max} son déplacement maximal (calculé par éléments finis) sous le chargement imposé. Soit D_{Lim} la valeur maximale autorisée pour D_{Max}. L'approche par pénalisation consiste à se ramener à un problème sans contrainte en minimisant

$$P + \alpha |D_{Max} - D_{Lim}|^+$$

c'est-à-dire en pénalisant le poids dés que le déplacement maximal dépasse la valeur limite autorisée (x^+ désigne la partie positive de x).

L'ajustement du paramètre α (qui peut être vu comme un multiplicateur de Lagrange pour la contrainte associée) n'est pas facile. Une méthode de pénalisation statique, avec α constant, peut donner de très bons résultats mais nécessite un ajustement très fin de la valeur de α. En effet, une valeur trop petite pourra donner des solutions du problème pénalisé qui ne respectent pas la contrainte. Une valeur trop grande, au contraire, interdira à la recherche de sortir de l'espace admissible, et ce manque de "raccourcis" peut se révéler fatal à l'exploration globale de l'espace de recherche.

Une stratégie tentante est alors de partir de petites valeurs, autorisant l'exploration de la zone infaisable, puis de l'augmenter progressivement afin d'assurer la faisabilité des solutions dans les dernière générations. Mais une telle pénalisation dynamique requiert une bonne intuition initiale pour déterminer une stratégie efficace, qui dépend certes du problème, mais également du l'état de la population à un instant donné.

Les algorithmes évolutionnaires permettent justement l'adaptation de certains paramètres aux conditions locales rencontrées lors de l'évolution : on parle alors de techniques adaptatives.

L'approche adaptative proposée ici a pour but de garder dans la population une proportion minimale d'individus satisfaisant les contraintes ainsi qu'une proportion minimale qui les violent. En effet, comme il est fréquent dans les problèmes sous contraintes intéressants, la solution cherchée se trouve probablement proche de la frontière du domaine admissible : intuitivement, si l'on est loin de cette frontière (i.e., le déplacement maximal est bien plus petit que la limite autorisée), on peut sans doute enlever de la matière sans violer la contrainte (attention, même pour le problème continu, il n'existe pas de certitude que la solution soit exactement sur la frontière du domaine admissible).

Notons $\Theta^k_{feasible}$ la proportion d'individus qui satisfont les contraintes à la génération k, et Θ_{inf} et Θ_{sup} deux paramètres donnés par l'utilisateur. Les faibles valeurs des paramètres de pénalisation favorisent les individus qui violent les contraintes (et inversement). Pour maintenir $\Theta^k_{feasible} \in [\Theta_{inf}, \Theta_{sup}]$, nous proposons la règle de mise à jour suivante :

$$\alpha_{k+1} = \begin{cases} \beta \cdot \alpha_k & \text{si } \Theta^k_{feasible} < \Theta_{inf} \\ (1/\beta) \cdot \alpha_k & \text{si } \Theta^k_{feasible} > \Theta_{sup} \\ \alpha_k & \text{sinon} \end{cases} \quad (17)$$

avec $\beta > 1$. Les paramètres choisis par l'utilisateur dans cette méthode sont Θ_{inf}, Θ_{sup}, β et la valeur initiale α_0. Après quelques tests de robustesse, les valeurs suivantes ont été choisies et seront utilisées dans toutes les simulations présentées dans cet article : $\beta = 1.1, \Theta_{inf} = 0.4$, et $\Theta_{sup} = 0.8$.

Notons que les variations de α ne sont pas monotones, et il n'y a donc pas de garantie *a priori* que le meilleur individu de la population satisfasse les contraintes. Il peut même arriver que la population ne contienne aucun individu admissible – même si dans ce cas, l'augmentation régulière de la valeur de α doive favoriser les individus violant le moins les contraintes, menant à l'émergence rapide d'individus admissibles.

Un problème multi-objectifs

En fait, le problème considéré, même dans le contexte simple de cette section, est un problème multi-objectifs : on cherche à minimiser **à la fois** le poids et le déplacement maximal sous l'action du chargement. On peut traiter directement ce problème en utilisant les procédures de sélection spécifiques au problèmes multi-objectifs (voir section 2.2.4). Les résultats d'une telle approche seront présentés en section 5.1.5. Attention cependant : du fait de l'astuce employée afin d'éviter les calculs de connectivité (section 5.1.2), les structures "vides" font partie du front de Pareto (elles sont de poids minimal). Il convient donc de limiter l'exploration dudit front aux valeurs de poids (ou de déplacements) intéressantes.

5.1.3 Une représentation paramétrique : les tableaux de bits

Le point le plus crucial dans la construction d'un algorithme évolutionnaire est le choix de la représentation.

Ainsi qu'il l'a été dit dans la section 5.1.2, le calcul de la performance d'une structure passe par une résolution d'un problème mécanique par éléments finis, et implique donc une discrétisation (un maillage) de la structure. De plus, si tout se passe bien ("convergence" de la méthode), nous serons amenés à comparer les comportements mécaniques de structures très proches les unes des autres. Des considérations numériques montrent qu'il est alors préférable d'utiliser des maillages identiques, afin de ne pas ajouter de bruit de remaillage aux valeurs (proches) des comportements mécaniques.

Ces considérations ont naturellement débouchées sur l'idée de partir d'un maillage donné pour définir une représentation binaire "naturelle", appelée tableau de bits, ou *bitarray* dans [82] : elle est associée au maillage du domaine de design qui est utilisé pour calculer le comportement mécanique de toutes les structures. A chaque élément du maillage on attribue une valeur 0 ou 1, la valeur 1 signifiant que cet élément contient de la matière, et 0 qu'il est vide (cf. Figure 11-b).

Cette représentation, "naturelle" pour des spécialistes des éléments finis, a été utilisée dans tous les premiers travaux appliquant les algorithmes évolutionnaires à l'optimisation tolopogique de formes [71, 29, 28], qui ont utilisé le fait que cette représentation peut être vue comme équivalente à la représentation en chaînes de bits des algorithmes génétiques classiques. Toutefois, il a été clairement montré dans [83] qu'il est préférable d'adapter les opérateurs de croisement à la "bi-dimensionalité" des tableaux de bits.

De nombreux résultats numériques ont été présentés à l'aide de la représentation en tableaux de bits dans [80, 79, 83]. Ils ont cependant tous été améliorés par l'utilisation des représentations non-structurées (voir section 5.1.4). Citons parmi ces

travaux des résultats d'optimisation topologique en élasticité non-linéaire (modèle des grands déplacements et matériau linéaire), premiers résultats de ce type, hors de portée (à l'époque) des méthodes classiques. La souplesse des algorithmes évolutionnaires est ici encore mise en évidence, puisqu'il suffit de changer de solveur du problème direct pour pouvoir attaquer le problème inverse. Parmi les autres succès rapportés dans [83], citons encore l'optimisation avec conditions aux limites sur la frontière inconnue (et non pas sur la frontière du domaine de définition), comme par exemple dans le cas de l'optimisation d'un dôme sous-marin soumis à la pression hydraulique.

Malgré son succès dans la résolution de problèmes d'optimisation topologique de formes [82, 83], la représentation "bitarray" souffre d'une profonde limitation liée à la dépendance de la complexité de l'algorithme avec celle du maillage associé. En effet, la taille d'un individu (le nombres de bits nécessaires pour le décrire) est égale à la taille du maillage. Malheureusement, des résultats théoriques [27] comme des constatations expérimentales [52] indiquent que la taille critique de population nécessaire pour atteindre la convergence augmente au moins linéairement avec la taille des individus. De plus, les populations plus nombreuses nécessitent souvent un plus grand nombre de générations pour converger. Il est donc clair que cette approche doit restreindre son domaine d'application à de grossiers maillages bidimensionnels, alors que les ingénieurs ont besoin de fins maillages tridimensionnels !

Ces considérations conduisent à la recherche de représentations plus compactes, dont la complexité ne dépend pas de celle de la discrétisation. Pour obtenir une représentation indépendante de la complexité une ultime étape consiste à la faire évoluer elle-même en l'ajustant par AEs.

5.1.4 Représentation de Voronoï

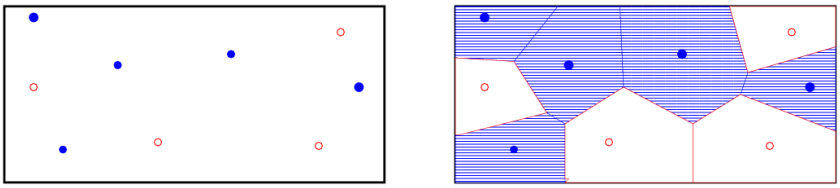
Cette section définit la *représentation de Voronoï*, qui dépasse les limitations de la représentation par tableaux de bits.

Diagrammes de Voronoï

Considérons un nombre fini de points $V_0, \ldots, V_N$ (les *sites de Voronoï*) dans un domaine borné donné de $\mathbb{R}^n$ (le domaine de travail). A chaque site V_i on associe l'ensemble de tous les points du domaine de travail pour lesquels le site de Voronoï le plus proche est V_i. On nomme cet ensemble *cellule de Voronoï*. Le *diagramme de Voronoï* est la partition du domaine définie par les cellules de Voronoï. Chaque cellule est un sous-ensemble polyédral du domaine de travail (cf. [22]).

Génotype et initialisation

Considérons maintenant une liste – de longueur variable – de sites de Voronoï, chaque site étant étiqueté 0 ou 1. Le diagramme de Voronoï correspondant, dans lequel chaque cellule est étiquetée comme le site qu'elle contient, représente une partition du domaine de travail en deux sous-ensembles(cf. Figure (14)).



(a) Le génotype : une liste de sites de Voronoï.

FIG. 12 – *La représentation de Voronoï sur un domaine rectangulaire* 2×1.

La procédure d'**initialisation** consiste en un tirage aléatoire uniforme du nombre de sites de Voronoï (compris entre 1 et un nombre maximum donné par l'utilisateur), des sites de Voronoï dans la structure, et du label booléen vide/matériau.

Décodage
D'un point de vue pratique, la performance de toutes les structures est évaluée en utilisant un maillage de taille fixe. Une partition décrite par un diagramme de Voronoï peut facilement se projeter sur un maillage donné : un élément appartient à l'une ou l'autre des catégories (matériau ou vide) en fonction du label de la cellule de Voronoï dans laquelle se trouve son centre de gravité, mais il y a indépendance entre le nombre de sites et la taille du maillage.

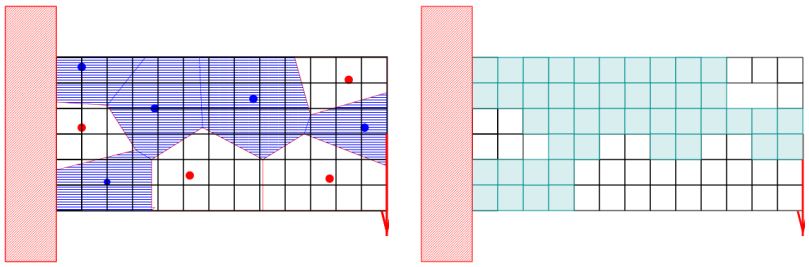


FIG. 13 – *la représentation de Voronoï sur le problème de la plaque-console* 2×1 *et la structure correspondante (répartition vide/matériau)*

Opérateurs de variation
Les opérateurs de variation pour la représentation de Voronoï sont définis de la manière suivante :

- L'opérateur de **croisement** échange les sites de Voronoï sur une base géométrique. De ce point de vue, il est similaire à l'opérateur de croisement spécifique décrit dans [81]. La Figure (14) est un exemple d'application de cet opérateur.
- L'opérateur de **mutation** est choisi de manière aléatoire, avec des poids définis par l'utilisateur, parmi les opérateurs suivants :
  - la mutation de *déplacement* effectue une mutation Gaussienne sur les coordonnées des sites. Comme dans les stratégies d'évolution (voir section TAB. 4.2.3), la mutation adaptative est utilisée : une déviation standard est associée à chaque coordonnée de chaque site de Voronoï, et elle subit une mutation log-normale avant d'être utilisée pour la mutation Gaussienne des coordonnées correspondantes.
  - la mutation de *label* change aléatoirement l'attribut booléen d'un site.
  - les mutations *ajout* et *suppression* sont des opérateurs spécifiques aux représentations de longueur variable qui respectivement ajoutent ou suppriment aléatoirement un site dans la liste.

Conditions expérimentales

Sauf indication contraire, les expériences numériques présentées dans la suite utilisent les paramètres suivants : évolution standard de type Algorithme Génétique : sélection basée sur le rang et remplacement de tous les parents par leur descendance), avec des population de 80 individus ; au plus 40 sites de Voronoï par individu ; taux de croisement de 0.6 et taux de mutation de 0.3 par individu ; les poids relatifs de chaque type de mutation sont de 0.5 pour la mutation de déplacement, les

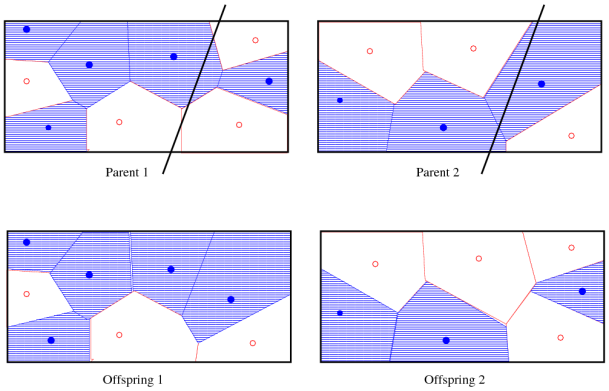
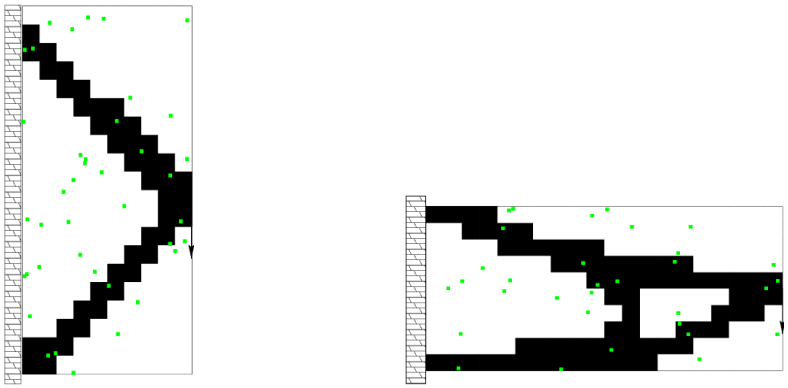


FIG. 14 – *Un opérateur de croisement pour la représentation de Voronoï : une droite aléatoire est tracée dans chaque diagramme et les sites sont échangés de part et d'autre de cette droite.*

autres types de mutations se partageant en parts égales les 0.5 restants ; le nombre maximum de générations est de 2000 et le critère d'arrêt de l'algorithme est de 300 itérations successives sans amélioration du meilleur individu ; tous les graphiques sont les résultats de 21 calculs indépendants avec les mêmes paramètres ; les temps CPU sont donnés pour des processeurs Pentium III à 800MHz sous Linux.

Premiers résultats

L'approche a été validée sur les problèmes-test de cantilevers de dimension 1×2 et 2×1, discrétisées respectivement selon un maillage régulier de 10×20 et 20×10, avec des limites respectives sur le déplacement maximal de 20 et 220. La Figure 15 montre les meilleures structures obtenues pour les deux cas-tests. Le coût d'une génération est de moins d'une seconde.



(a) : poids=0.215, 35 sites (b) : poids=0.35, 32 sites

FIG. 15 – *Les deux meilleurs individus pour la représentation de Voronoï. Temps CPU = 1s/génération. (a) maillage* 10×20*,* D_{lim}*=20 (b) maillage* 20×10*,* D_{lim}*=220.*

Le cantilever 10×1

Le problème du cantilever 10×1 (discrétisé par un maillage régulier de 200×20 éléments) est délicat à traiter avec une représentation de type bitarray à cause d'une difficulté supplémentaire par rapport au cas précédent : la plupart des structures générées aléatoirement au cours du processus d'initialisation ne connectent pas le point d'application de la force avec la frontière encastrée. Une procédure d'initialisation particulière est utilisée, dans laquelle le poids moyen des structures aléatoires peut être contrôlé (cf. [78] pour les détails). De plus, le nombre maximal de sites par individus a été augmenté à 120 et les meilleurs résultats sont obtenus pour des populations de 120 individus. La Figure 16 montre un des résultats les plus significatifs pour une contrainte $D_{lim} = 12$.

FIG. 16 – *Structure optimale sur le maillage* 200×20 *pour le cantilever* 10×1. *D_{lim}=12, D_{max}=11.99, poids=0.445. Temps CPU = 28s/génération.*

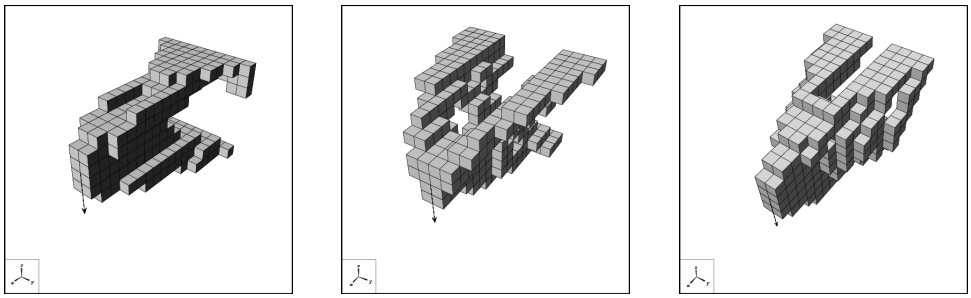
Un problème 3D

Nous présentons maintenant les premiers (à notre connaissance) résultats 3D en optimisation topologique de formes par algorithmes évolutionnaires.

Le domaine de travail est un parallélépipède rectangle et le problème présente une symétrie permettant de ne discrétiser que la moitié du domaine par un maillage à $16 \times 7 \times 10$ éléments. Le plan en $x = 0$ (au fond sur la Figure 17) est encastré, et une force verticale est appliquée au milieu de la face opposée.

Pour ce problème plus complexe que les cas précédents, quelques paramètres doivent être modifiés : la taille de la population est de 120 individus et le nombre maximum de site de Voronoï par individu vaut 120.

La Figure 17 montre que l'algorithme a été capable de trouver quelques solutions acceptables ... en quelques jours de calcul (les analyses par éléments finis de problèmes 3d sont beaucoup plus coûteuses que pour les cas 2d à nombre de mailles égal). De plus, elle démontre la capacité connue des AEs de trouver des solutions quasi-optimales multiples pour un même problème, certaines étant assez originales par rapport à celle obtenue par la méthode d'homogénéisation [1].



(a) : poids 0.152, 103 sites (b) : poids 0.166, 109 sites (c) : poids 0.157, 112 sites

FIG. 17 – *Trois résultats pour le cantilever tridimensionnel sur un maillage* $16 \times 7 \times 10$ *sur la moitié du domaine, avec la même contrainte sur le déplacement maximal. Temps CPU = 4 à 5mn/génération.*

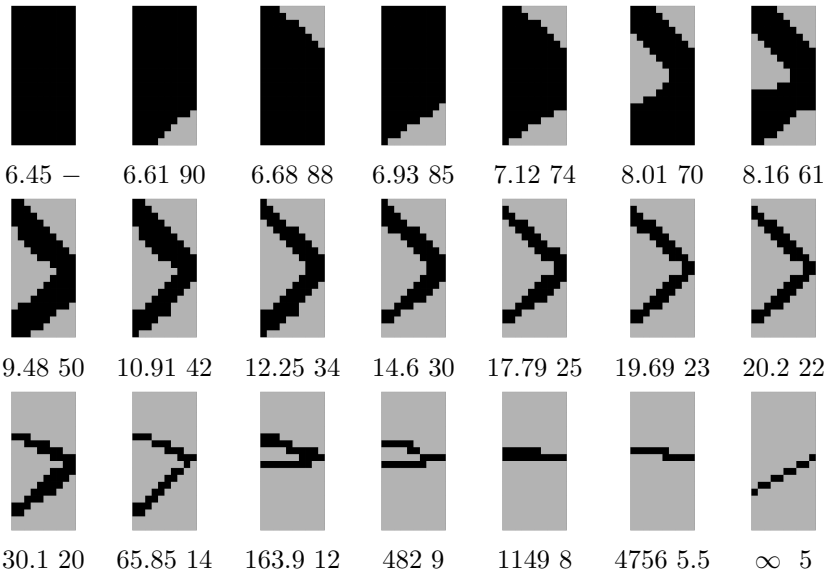


FIG. 18 – 21 compromis optimaux au sens de Pareto pour le problème de la plaque console 10×20. Le déplacement maximal et la masse (en %) sont indiqués sous chaque structure. La dernière structure atteint les limites de la modélisation.

5.1.5 Résultats multi-critères

Les résultats présentés jusqu'à présent concernaient la minimisation du poids de la structure avec une contrainte sur le déplacement maximal en présence d'un (ou plusieurs) chargements donnés. Nous considérons maintenant la minimisation simultanée du poids **et** du ou des déplacements maximaux sous un ou plusieurs chargements (section 5.1.2). En utilisant les techniques d'optimisation de Pareto (voir section 2.2.4), on peut obtenir **en un seul essai** un échantillonnage du front de Pareto du problème. La figure 18 présente ainsi les structures les plus significatives obtenues en trois essais de algorithme évolutionnaire multi-objectifs NSGA-II (voir section 2.2.4) pour le problème test du cantilever 10×20, classées par poids décroissant, et la figure 19 le front de Pareto correspondant. Les résultats sont détaillés dans [62]. En particulier, ils sont comparés avec les résultats obtenus par l'approche sous contrainte de déplacement maximal : lorsque les déplacements maximaux sont proches, les deux structures obtenus par les deux méthodes sont très similaires, et ont des comportements mécaniques et un poids très proches. Sachant que l'approche multi-objectifs permet d'obtenir pour le même prix l'ensemble du front de Pareto, l'avantage est clairement au multi-objectifs !

A noter cependant que de nombreuses structures sans intérêt font partie du front de Pareto (la structure pleine par exemple est optimale en terme de rigidité !) et qu'il faut en général en tenir compte dans l'algorithme pour éviter que de telles solutions n'envahissent la population. D'autre part, l'analyse des résultats d'algorithmes multi-objectifs pour des problèmes comportant plus de 3 ou 4 objectifs tient de la gageure, et il est souvent difficile d'en extraire des informations utiles sans avoir recours à des techniques de fouille de données [108].

Conclusions sur l'optimisation tolopogique de formes

Dans le domaine de l'optimisation tolopogique de formes, les algorithmes évolutionnaires

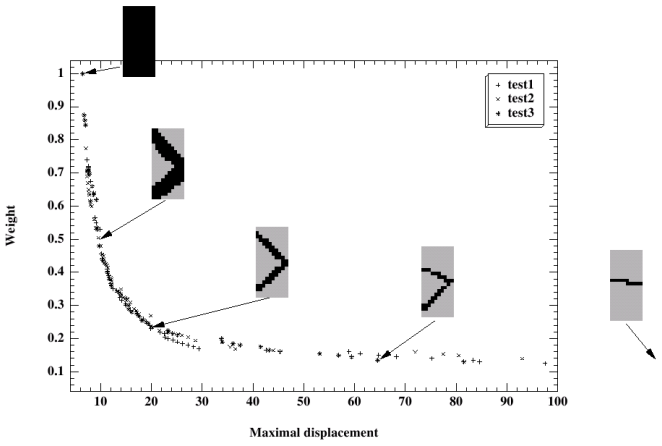


FIG. 19 – Trois fronts de Pareto pour le problème de la plaque console 10×20 obtenus indépendamment après 400 générations de 300 individus (test1, test2 et test3).

manipulant des représentations non-paramétriques permettent de dépasser les limites des méthodes déterministes classiques pour explorer des espaces de recherche jusque là hors d'atteinte pour les concepteurs. Pour un même problème, des représentations différentes peuvent donner des résultats de nature fondamentalement différente. Par exemple, pour le problème étudié dans cette section, le lecteur intéressé trouvera dans [63] d'autres représentations non-paramétriques, et dans [44] un exemple de représentation de formes à l'aide de programmes VRML, évoluant suivant les principes de la programmation génétique (voir section 5.3.1). Ainsi, au-delà de la "simple" optimisation, les algorithmes évolutionnaires se révèlent de fabuleux outils d'exploration, donnant naissance à ce qu'on peut sans doute appeler la *créativité assistée par ordinateur* [15, 16].

5.2 Conception optimale de stratifiés composites

Nous donnons ici un exemple d'algorithme évolutionnaire spécialement conçu pour optimiser les empilements dans les matériaux composites [96]. Les matériaux composites que nous considérons sont ceux obtenus par superposition de plis, chaque pli étant composé de fibres, typiquement en verre ou en graphite, alignées et tenues par une matrice, le plus souvent une résine époxyde. La structure considérée est une plaque simplement supportée, soumise à des charges compressives dans le plan Nx et Ny, ce qui est illustré sur la FIG. 20. Le problème de l'optimisation de stratifiés composites est la détermination du nombre et de l'orientation des plis. Pour des raisons de fabrication, les orientations possibles des plis prennent des valeurs dans $\{0°, +45°, -45°, 90°\}$. Pour des raisons de symétrie de comportement du stratifié, on impose que l'empilement soit symétrique par rapport à son plan moyen et qu'il y ait autant de plis à $+\theta°$ qu'à $-\theta°$. Ces contraintes sont directement prises en compte dans le codage en ne codant qu'une moitié de l'empilement (l'autre moitié étant obtenue par symétrie) et en définissant comme allèle une paire $x_i = \pm\theta°$, $\theta \in \{0, 45, 90\}$. En notant par les indices le nombre de plis adjacents dans la même direction, et comme les plis à $+90°$ et $-90°$ sont équivalents, l'ensemble des valeurs des allèles est $x_i \in \{0_2, \pm 45, 90_2\}$. Par exemple, le vecteur de variables $x = [\pm 45, 90_2, \pm 45]$ correspond au stratifié à 12 plis de séquence $[45°/-45°/90°/90°/+45°/-45°/-45°/+45°/90°/90°/-45°/+45°]$.

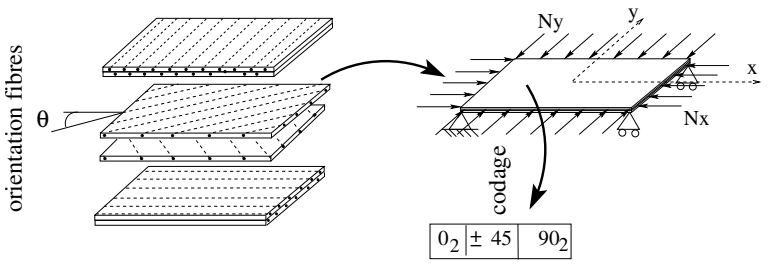


FIG. 20 – Codage d'une plaque composite, d'après [96, 111]. Une plaque est composée d'un empilement de plis dans lesquels les fibres du matériau ont une direction θ. Il est usuel d'imposer un empilement symmétrique par rapport au plan moyen et le même nombre de plis dans la direction θ et $-\theta$. Ces contraintes sont directement prises en compte dans le codage, en ne décrivant qu'une moitié de stratifié et en groupant les plis par "allèles" $\pm\theta$. Le nombre d'allèles, variable, permet de changer l'épaisseur du stratifié.

Le problème de la conception du stratifié est formulé comme

$$\begin{cases} \min n \ , \\ \text{en changeant } x \in \{0_2, \pm 45, 90_2\}^n \text{ (dont } n) \ , \\ \text{tel que} \quad \lambda_{\text{fl}}(x) \geq 1 \ , \quad \lambda_{\text{r}}(x) \geq 1 \quad \text{et} \quad n_c(x) \leq 4 \ . \end{cases} \tag{18}$$

L'inclusion de n dans les variables rend le problème non paramétrique. λ_{fl} est le facteur de charge critique de flambement, λ_{r} le facteur de charge critique de rupture (critère des déformations principales) et n_c est le nombre de plis contigus de même orientation. Les deux premières contraintes assurent que le stratifié ne se rompe pas par flambement ou rupture sous le chargement (Nx, Ny). La limitation du nombre de plis contigus de même orientation est une règle métier visant à réduire l'endommagement de la matrice. Ces trois contraintes d'optimisation sont prises en compte par pénalisation statique. La formulation détaillée de la pénalisation, les valeurs des cas de chargement et les propriétés des matériaux peuvent être trouvés dans [96]. Nous les omettons ici pour ne pas alourdir inutilement notre propos. [17] peut être consulté pour les modèles de flambement et de rupture.

Les problèmes d'optimisation de séquence d'empilement sont non-linéaires et susceptibles d'avoir des optima locaux car il y a surabondance de variables (le nombre de plis peut être important, de l'ordre de la centaine). De plus, la nature discrète et non-paramétrique de $\mathcal{S}$ nous invite à aborder le problème par algorithmes évolutionnaires. De multiples versions d'AEs pour l'optimisation de composites ont été proposées dans, e.g., [111, 95, 96, 86, 87, 132]. Nous résumons ici les comparaisons effectuées dans [96].

Le critère utilisé pour comparer les différents AEs est le **coût** de la recherche, défini comme le nombre moyen d'évaluations de f pour atteindre une solution de performance proche[17] de x^* avec 80% de chances. Le prix est estimé par moyenne sur 200 runs indépendants pour trois cas de chargements (Nx, Ny), soit 600 optimisations par version de l'algorithme. La contribution de chaque composant de l'AE (mutation, croisement, ...) à la performance de l'algorithme dépend des autres composants, ce qui peut se comprendre, en première approximation, à travers l'équilibre global entre exploration et exploitation (cf. Section 2.3). En toute rigueur, il est faux de discuter de l'effet d'un opérateur sans parler de son interaction avec les autres. Cependant, essayer toutes les combinaisons ou résoudre le méta-problème d'optimi-

[17] Proche signifie avec le même nombre de plis et $\min(\lambda_{\text{fl}}, \lambda_{\text{r}}) > 0.999 \min(\lambda_{\text{fl}}(x^*), \lambda_{\text{r}}(x^*))$.

Parent 1 :	$[x_1^1$	x_2^1	x_3^1	x_4^1	$\square$	$\square$	$\square]$
Parent 2 :	$[x_1^2$	x_2^2	x_3^2	x_4^2	x_5^2	x_6^2	$x_7^2]$
X1-thin, $ic = 2$:	$[x_1^1$	x_2^1	x_3^2	x_4^2	x_5^2	x_6^2	$x_7^2]$
X1-thick, $ic = 5$:	$[x_1^1$	x_2^1	x_3^1	x_4^1	$\square$	x_6^2	$x_7^2]$
soit,	$[x_1^1$	x_2^1	x_3^1	x_4^1	x_6^2	$x_7^2]$	
X2-thin, $ic = 1$ et 3 :	$[x_1^1$	x_2^2	x_3^2	$x_4^1]$			
X2-thick, $ic = 1$ et 6 :	$[x_1^1$	x_2^2	x_3^2	x_4^2	x_5^2	$x_6^2]$	

FIG. 21 – Exemples de croisements pour les stratifiés composites. ic sont les indices des points de coupures.

sation des paramètres de l'AE (e.g., [57]) paraissait d'un coût numérique trop important ici. La démarche adoptée dans [95, 96] a été de trouver, par énumérations successives sur chaque composante de l'AE, un optimum local dans l'espace des AEs (opérateurs et leurs réglages). L'AE de départ de cette méta-optimisation empirique est un algorithme génétique élitiste avec des réglages classiques (sélection par le rang, population de $\mu = 100$ individus, croisement à deux points et $p_c = 0.6$, $p_m = 0.01$). L'AE à l'arrivée a une population de $\mu = 8$ individus, et ses opérateurs de variation ont été adaptés. Il a un prix de recherche de 1450 analyses pour des stratifiés optimaux de 48 plis (12 allèles, donc une taille d'espace $\mathcal{S}$ supérieure à $3^{12} = 531441$). L'AE explore moins de 0.2% de l'espace de recherche. Nous discutons maintenant le choix des opérateurs pour l'optimisation de séquences d'empilement.

Croisement

Cinq croisements ont été comparés, X1-thin, X1-thick, X2-thin, X2-thick et le croisement uniforme. Ces croisements utilisent deux parents et produisent un enfant. Lorsque les deux stratifiés parents n'ont pas la même épaisseur, le chromosome du stratifié le plus fin est complété d'allèles vides, $\square$, jusqu'à avoir la même taille que le stratifié épais. Lorsqu'un vide apparaît à l'intérieur d'un empilement, il est effacé et la séquence compactée. Les croisements X1-... sont des croisements à un point de coupure. X1-thin est un croisement où le point de coupure est pris dans la partie pleine du stratifié le plus mince. Ainsi, le stratifié enfant a toujours l'épaisseur d'un des deux parents. Dans X1-thick, le point de coupure est pris dans la partie pleine du stratifié le plus épais. Le stratifié produit peut avoir toutes les épaisseurs entre, et y compris celles, des deux parents. Les croisements X2-... suivent la même logique avec deux points de coupures. Le croisement uniforme a été décrit en Section 2.2.5. Des exemples de croisements sont donnés en FIG. 21.

Le croisement le plus performant sur cette classe de problèmes est X1-thick. Contrairement aux X...-thin, il mélange bien les épaisseurs des parents puisqu'il est capable de produire des stratifiés ayant toutes les épaisseurs intermédiaires entre celles des parents. De plus, il ne possède qu'un point de coupure ce qui permet de conserver d'importantes parties de séquences des parents intègres lors de la recombinaison. Que cette dernière propriété soit un avantage doit être mis en relation avec l'usage intensif de l'opérateur de permutation (voir ci-après).

Mutation(s)

On peut imaginer trois types de perturbations locales d'une séquence d'empilement : un changement d'orientation des plis individuels, un changement de la position des plis et un changement de l'épaisseur du stratifié. Il est important de séparer ces perturbations au sein de la mutation pour permettre le réglage de l'AE. Pour s'en convaincre, considérons un contre-exemple. L'application directe des opérateurs standards aux composites passerait par l'utilisation de la mutation discrète de l'équation (8) sur un codage des séquences où $x_i \in \{0_2, \pm 45, 90_2, \square\}$, $i = 1, n$, avec n fixé a priori (cadre paramétrique). Si tous les allèles ont la même probabilité d'apparition par mutation $(1/(A-1))$, toutes les probabilités d'évènements de la mutation deviennent interdépendantes et fonctions de l'épaisseur du stratifié et de n. Par exemple à n fixé, plus un stratifié est mince, plus il a de $\square$ dans son chromosome, et plus il a de chances d'épaissir par mutation (mutation de $\square$ en un allèle plein, 0_2, ± 45, ou 90_2).

L'opérateur de mutation prend donc trois formes dans l'optimisation de séquences d'empilement : la variation d'épaisseurs, la variation d'orientation des plis et la variation de l'ordre dans lequel les plis sont empilés. C'est l'opérateur de "permutation" qui réalise cette dernière opération en inversant la position de deux allèles avec la probabilité p_p par individu :

$$\begin{array}{rl} \text{Avant permutation :} & [x_1 \quad x_2 \quad x_3 \quad x_4 \quad \ldots \quad x_n] \\ \text{Permutation de 2 et 4 :} & [x_1 \quad x_4 \quad x_3 \quad x_2 \quad \ldots \quad x_n] \end{array}$$

Pour l'optimisation de séquences d'empilement, l'opérateur de permutation est très important. Il est pratiqué sur tous les enfants, $p_p = 1$. Il permet en effet de modifier la réponse du stratifié vis à vis des plis contigus (n_c dans l'équation (18)) et du flambement (λ_{fl}), sans changer sa réponse vis à vis de la rupture (λ_{r}) ou du nombre de plis qui sont deux fonctions insensibles à la position des plis dans la séquence. En termes de paysages de fonctions, la permutation déplace les x parallèlement aux lignes de niveau des critères n et λ_{r}. Une fois que la contrainte sur λ_{r} est satisfaite, la permutation donne l'avantage de pouvoir explorer intensément les autres critères tout en restant dans le domaine faisable de λ_{r}.

La probabilité empirique optimale de changement d'orientation des allèles est de 0.01 par allèle. De même, les probabilité optimales d'ajout et de retrait d'un allèle sont de 0.05 par individu. Les opérateurs de mutations et de permutation sont appliqués indépendamment sur le même individu.

5.3 Identification de lois de comportement

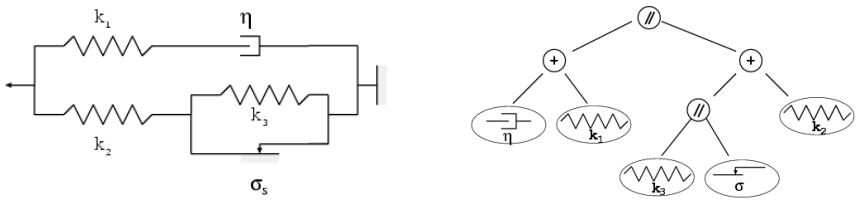
Nous allons illustrer les possibilités de la programmation génétique sur deux exemples issus de la mécanique du solide et concernant l'identification de lois de comportement à partir de données expérimentales [124, 117].

5.3.1 Identification de modèles rhéologiques

La résolution d'un problème commence par le choix de l'espace de recherche, ici l'espace des lois de comportement considéré. Pour diverses raisons (insuffisance des données disponibles, besoin de certification), l'expert attache une grande importance à ce que la loi obtenue ne soit pas une boite noire, mais puisse être inspectée et validée manuellement.

Pour cette raison, l'espace choisi a été celui des lois rhéologiques (lois de comportement dynamique uni-dimensionnel), en se limitant au couplage série et parallèle d'éléments purement élastiques (les ressorts), plastiques (les patins) et visqueux

(les amortisseurs). La FIG. 22.(a) décrit un modèle rhéologique proposé pour le polypropylène.



(a) (b)

FIG. 22 – Un modèle rhéologique (a) et sa représentation arborescente (b)

L'identification paramétrique vise à déterminer les coefficients d'un modèle donné ; dans le cas de la FIG. 22 il s'agit de déterminer les raideurs k_1, k_2 et k_3 des ressorts, le seuil σ du patin et la viscosité η de l'amortisseur. Notons que ce problème d'identification paramétrique est un problème d'optimisation mal posé (non convexe et admettant de nombreux optima locaux).

Un problème clairement plus complexe est celui de l'identification non paramétrique, cherchant à la fois la forme du modèle et les coefficients de celui-ci. A notre connaissance, il n'existe pas d'approche d'optimisation autre que la programmation génétique permettant d'explorer l'espace des modèles rhéologiques, et nous allons voir comment.

La programmation génétique définie par Koza [88, 90] étend les principes des AEs aux espaces d'arbres. L'espace d'arbres initialement considéré était celui des programmes Lisp, et le but était la synthèse de programmes ; en effet, pour peu que l'on sache définir la performance de tout programme, la synthèse de programme se ramène à un problème d'optimisation dans l'espace des programmes : trouver le programme de performance optimale. Un programme se définit comme un arbre, construit à partir d'un ensemble de noeuds ou opérateurs (while do, repeat until, if then else,...) et un ensemble d'opérandes (expressions et instructions ; le langage Lisp permet justement de considérer de la même manière expressions et instructions, ces dernières renvoyant une valeur).

Le fait de considérer un espace de représentation ou espace génotypique $\mathcal{S}$ particulier dans le cadre d'un AE demande de définir des opérateurs génotypiques adaptés : un opérateur d'initialisation, permettant d'échantillonner $\mathcal{S}$, un opérateur de croisement (défini sur $\mathcal{S}^\rho$ et à valeurs dans $\mathcal{S}$) et un opérateur de mutation (défini sur $\mathcal{S}$ et à valeurs dans $\mathcal{S}$).

La procédure d'initialisation procède de manière récursive : étant donné un ensemble de noeuds $\mathcal{N}$ et un ensemble d'opérandes ou de terminaux $\mathcal{T}$, on choisit un élément de $\mathcal{N} \bigcup \mathcal{T}$; si cet élément q est un noeud d'arité k, on rappelle k fois la même routine, qui produit les sous-arbres $s_1, \ldots, s_k$ et on renvoie l'arbre $q(s_1, \ldots, s_k)$; sinon, l'élément q est un terminal, et la routine renvoie q. Un choix délicat est de définir la profondeur maximum d_{max} des arbres.

La procédure d'initialisation la plus conseillée (*ramped half and half*, [13]) impose qu'une moitié des individus soit des arbres "pleins" (à tout niveau $d > 1$, le noeud choisi est un opérateur tiré dans $\mathcal{T}$).

Le croisement met généralement en jeu deux parents [13] ; un sous-arbre est choisi dans chacun des parents, et ces sous-arbres sont échangés (FIG. 23). La mutation classique, ou structurelle, consiste à remplacer l'un des sous-arbres du parent courant, par un nouveau sous-arbre issu de la routine d'initialisation.

L'identification de modèles rhéologiques par programmation génétique repose sur le fait que les modèles rhéologiques peuvent être vus comme des espaces d'arbres (FIG. 22.(b)). Ces arbres sont définis à partir de deux opérateurs parallèle et série

```
Routine Initialisation(𝒩, 𝒯, profondeur d)
  Si d == 1,
    Tirer q dans 𝒯
    Return q
  Sinon
    Tirer q dans 𝒩 ⋃ 𝒯
    Si q ∈ 𝒩 d'arité k
      For i = 1..k
        s_i = Initialisation(𝒩, 𝒯, profondeur d − 1)
      return q(s_1 ..., s_k)
    Else return q
```

TAB. 2 – Initialisation en programmation génétique, à partir d'un ensemble $\mathcal{N}$ de noeuds et $\mathcal{T}$ de terminaux.

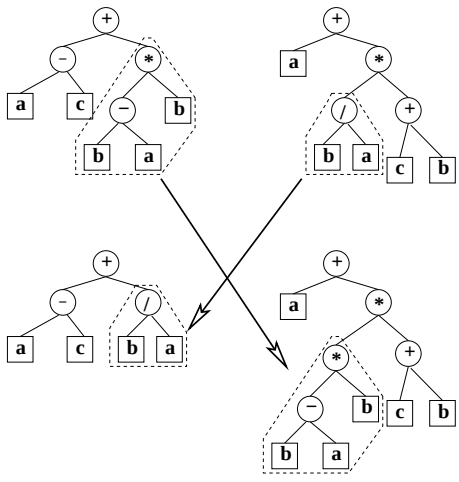


FIG. 23 – Croisement en programmation génétique

($\mathcal{N} = \{//, \Sigma\}$) et de trois terminaux ($\mathcal{T} = \{ressort,\ patin,\ amortisseur\} \times \mathbb{R}^+$) représentant un élément (ressort, patin ou amortisseur) associé à un coefficient réel (respectivement la raideur du ressort, le seuil du patin ou la viscosité de l'amortisseur).

Notons que la programmation génétique et les codages non paramétriques remettent en cause dans une certaine mesure l'hypothèse dite de forte causalité [120], selon laquelle les bons individus se trouvent souvent au voisinage des bons individus[18] : le fait de modifier un seul opérateur (e.g. remplaçant un noeud série par un noeud parallèle) conduit en général à un individu de phénotype entièrement différent.

Il faut ainsi distinguer les modifications mineures (les coefficients des ressorts, patins ou amortisseurs sont légèrement modifiés) des modifications structurelles (la nature d'un terminal ou d'un opérateur est modifiée). Il faut en particulier faire suivre toute modification structurelle d'une étape d'ajustement des coefficients. Supposons en effet qu'un individu x doté d'une structure nouvelle apparaisse ; la chance pour que ses coefficients lui permettent d'avoir une performance acceptable est voisine de zéro, si les autres individus de la population ont déjà bénéficié de quelques générations pour ajuster les coefficients adaptés à leur structure. L'individu x doit ainsi disposer de "cours de rattrappage" pour être viable et survivre à la sélection. Pratiquement, deux opérateurs de mutation sont définis : la mutation structurelle procède tel que décrit ci-dessus, remplaçant un sous-arbre par un sous-arbre aléatoire ; la mutation scalaire procède par perturbation gaussienne des coefficients attachés à chaque élément. Le taux de mutation structurelle est d'un ordre de grandeur plus faible que celui de la mutation scalaire ; de plus, toute mutation structurelle et tout croisement sont suivis d'un nombre n_s ($n_s = 10$ dans les expériences faites) de mutations scalaires ; les n_s individus produits par la mutation scalaire sont évalués et le meilleur est retenu. L'augmentation du paramètre n_s conduit à une augmentation du temps de calcul ; mais sa diminution a pour effet que la structure des individus n'évolue plus après les premières générations, et que la programmation génétique se ramène alors à une optimisation paramétrique.

La partie la plus délicate est de programmer la fonction objectif F ; celle-ci demande en effet de simuler la loi de comportement d'un modèle rhéologique x quelconque sous les conditions de chargement expérimentales.

Or, un patin obéit à deux lois de comportement différentes selon que la contrainte courante est inférieure à la contrainte seuil ou non. Si un individu comprend k patins, la loi de comportement recouvre ainsi 2^k cas de figure. En pratique, à tout noeud i de l'arbre rhéologique sont associées deux variables contrainte σ_i et déformation ε_i. Pour chaque pas de temps t_j le système d'équations différentielles associées à chaque noeud est construit[19], le noeud racine portant les conditions de chargement expérimentales. Ce système est résolu par différences finies, et la performance est donnée par la différence euclidienne entre le comportement obtenu et le comportement expérimental.

Le comportement du meilleur individu, au fur et à mesure des générations, "fitte" le comportement observé (Fig. 24).

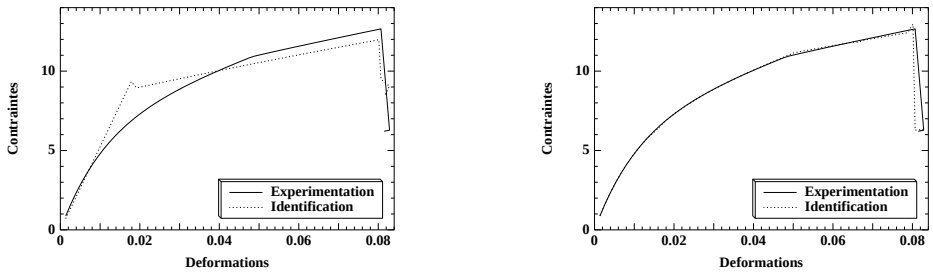
Le critère d'arrêt prend en compte le bruit de résolution numérique : l'évolution s'arrête lorsque la performance est du même ordre que la différence entre le comportement simulé et celui simulé pour une discrétisation temporelle d'un pas double.

Notons que la représentation ainsi définie est redondante : d'une part, l'opérateur

[18] L'absence de cette hypothèse caractérise une fonction performance aléatoire, pour laquelle la procédure d'optimisation adaptée est une méthode de Monte-Carlo.

[19] Ainsi l'équation associée à un patin de seuil σ_S est :

$$\sigma(t_j) < \sigma_S \quad OU \quad \dot{\varepsilon}(t_j) = 0$$



(a) 20 générations (b) 200 générations

FIG. 24 – Comportement du meilleur individu après 20 et 200 générations, moyenné sur 15 exécutions et comparé avec le comportement expérimental (en pointillé). La population génétique comprend 200 individus, le taux de croisement est .4, le taux de mutation de .2 ; le nombre de mutations scalaires après toute modification structurelle est de 10. Le coût de calcul est de 12 heures sur Pentium 166, pour 150 points d'expérience.

parallèle est commutatif ($q//q' \equiv q'//q$) ; et d'autre part, un modèle rhéologique peut en général être simplifié[20] (e.g. la mise en série de deux ressorts de raideurs k et k' est équivalent à un unique ressort de raideur $k + k'$).

En revanche, tout arbre rhéologique correspond à un modèle valide ; en d'autres termes, tout l'espace de recherche est admissible.

5.3.2 Identification de modèles dimensionnellement admissibles

Dans de nombreux cas cependant, l'espace de recherche contient des solutions non admissibles ou indésirables. Si les connaissances du domaine permettent d'éviter d'explorer les solutions indésirables et de restreindre ainsi l'espace de recherche, l'identification peut en être facilitée d'autant.

Ainsi, l'expert attend souvent des lois cohérentes du point de vue de la dimension des variables : on n'additionne pas des mètres et des kilogrammes pour obtenir des Newtons.

Dans le contexte de la programmation génétique, il faut alors remettre en cause le principe dit de *cloture de la représentation*, qui était vérifié en Lisp et selon lequel tout sous-arbre pouvait être argument d'un opérateur quelconque. Fondamentalement, la cloture de la représentation a pour avantage la simplicité des opérateurs

[20] La capacité de la programmation génétique à produire des arbres extrêmement longs et touffus, appelée *bloat*, constitue l'une ses limitations essentielles, abondamment étudiée (voir par exemple [91]). En effet, la croissance incontrôlée du code génétique avec le nombre de générations pénalise l'évolution à deux niveaux : en termes de taille mémoire d'une part, et en augmentant le coût d'évaluation d'autre part. C'est l'une des raisons pour lesquelles la programmation génétique fait en général évoluer des populations extrêmement vastes (jusqu'à plusieurs centaines de milliers d'individus [90]) pendant un faible nombre de générations (quelques centaines).

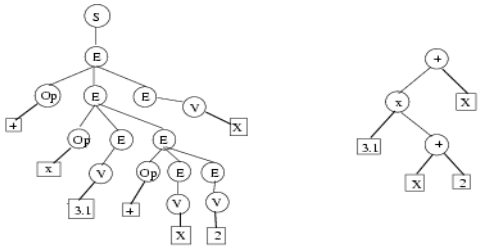
génotypiques, initialisation croisement et mutation ; mais elle conduit à une augmentation exponentielle de la taille de l'espace de recherche.

L'approche la plus utilisée pour restreindre l'espace de la programmation génétique en fonction des connaissances du domaine repose sur l'usage de grammaires hors contexte [136, 60]. Une grammaire hors contexte décrit les phrases admissibles d'un langage selon un quadruplet $\{S, N, T, P\}$, où S est le symbole initial (start), N et T respectivement l'ensemble des symboles non-terminaux et terminaux, et P l'ensemble des règles de production ; à chaque symbole non-terminal est associé une règle de production, donnant l'ensemble des règles de dérivation permettant de réécrire le symbole non-terminal. La grammaire décrivant l'ensemble des polynômes de deux variables X et Y (où le symbole $\mathcal{R}$ correspond à une constante réelle quelconque) est donné en Fig. 25.(a).

La production d'une expression bien formée procède en choisissant une des règles de dérivation associées au symbole S initial, et en réecrivant itérativement l'expression courante, remplaçant chacun des symboles non-terminaux par une des règles de dérivation associées, jusqu'à obtenir une expression terminale, i.e. ne contenant aucun symbole non-terminal. Ce procédé de réécriture définit un arbre de dérivation, vu comme une représentation équivalente (mais non unique) de l'expression terminale obtenue (Fig. 25.(b)).

La programmation génétique, opérant sur les espaces d'arbres, peut en particulier opérer sur les arbres de dérivation. La programmation génétique grammaticale (PGG) impose deux restrictions sur les opérateurs de variation [60]. En ce qui concerne l'opérateur de croisement, la restriction porte sur la sélection du sous-arbre dans le second parent : étant donné un sous-arbre sélectionné uniformément dans le premier parent, on impose que le sous-arbre sélectionné dans le second parent ait pour racine *le même symbole non terminal*. Le fait d'échanger les deux sous-arbres ainsi sélectionnés garantit l'obtention de deux expressions bien formées si les expressions initiales étaient bien formées. En ce qui concerne la mutation, le sous-arbre sélectionné dans le parent courant est simplement remplacé par un arbre de dérivation associé au même symbole. La programmation grammaticale généralise ainsi la programmation génétique fortement typée [136]. Dans les deux cas, le principe est le même : toute restriction de l'espace de recherche permettant de prévenir a priori la production d'expressions non pertinentes, améliore les performances.

$$
\begin{array}{rcl}
N & = & \{< E >, < O >, < V >\} \\
T & = & \{+,\ \times,\ X,\ Y,\ \mathcal{R}\ \} \\
P & = & \left\{ \begin{array}{ll}
S & := < E > ; \\
< E > & := < O >< E >< E > \\
 & \quad\ \mid < V >; \\
< O > & := \ + \mid \times \ ; \\
< V > & := \ X \mid Y \mid \mathcal{R} \ ;
\end{array} \right.
\end{array}
$$



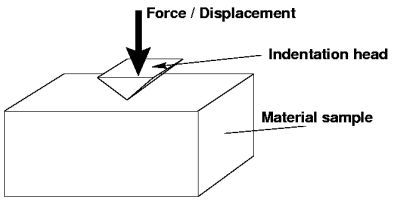
(a) Grammaire (b) Arbre de dérivation et Expression

Fig. 25 – Grammaire hors contexte des polynômes de deux variables

La cohérence dimensionnelle, sous réserve d'utiliser un nombre fini d'unités, peut être codée sous forme de grammaire hors contexte. A chaque unité considérée (ex., Newton) est associé un symbole non terminal et la règle de production associée spécifie toutes les façons possibles d'obtenir une expression de l'unité considérée (additionner ou soustraire deux expressions de type Newton ; multiplier une expression de type masse par une accélération ; etc) [116].

Ainsi, dans le cadre de l'identification de lois d'indentation (Fig. 26.(a)), trois

dimensions sont considérées (masse, longueur, temps). Chaque variable du problème est associée à son unité, décrite par le vecteur de ses exposants en les dimensions (Fig. 26.(b)). Le nombre d'unités possibles est limité en se limitant aux exposants entiers variant entre -2 et 2. Un symbole non-terminal est associé à chacune des 125 unités possibles, et les règles de production formant la grammaire dimensionnelle sont construites automatiquement.



(a)

Entité	masse	longueur	temps
Variables			
E (Force)	+1	+1	–2
K (elements élastiques)	+1	0	–2
C (elements visqueux)	+1	0	–1
t (temps)	0	0	+1
Solution			
u (déplacement)	0	+1	0

(b)

Fig. 26 – Identification de lois d'indentation

La réduction de l'espace de recherche est de plusieurs ordres de grandeur, et croit avec la profondeur des arbres autorisée (Fig. 5.3.2).

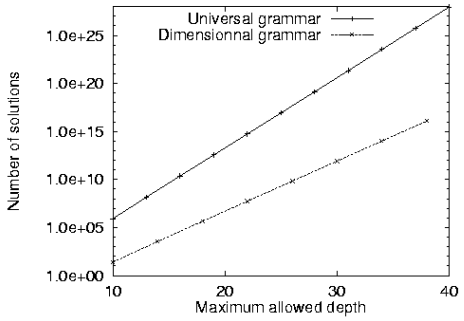


Fig. 27 – Réduction de l'espace de recherche due à la grammaire dimensionnelle

La grammaire dimensionnelle décrite ci-dessus comprend 125 symboles non-terminaux et des milliers de règles de dérivation. Il s'agit ainsi d'une grammaire de très grande taille comparé aux exemples de la littérature, ce qui a entrainé une difficulté particulière pour l'initialisation de la population. En effet, la sélection uniforme des règles de dérivation conduit à des expressions extrêmement longues, qui ne satisfont pas la limite sur la profondeur de l'arbre obtenu. L'initialisation devient ainsi une étape extrêmement coûteuse ; des centaines de milliers d'arbres doivent être générés pour produire une population de 200 individus de longueur admissible. De surcroît, l'initialisation débouche sur une population peu diversifiée, et l'évolution ne permet pas de compenser la pauvreté de la population initiale. En effet, ainsi que nous l'avons souligné à plusieurs reprises, la qualité de l'initialisation a des effets considérables sur la qualité de l'optimisation [35].

Une heuristique, inspirée des travaux sur l'échantillonnage uniforme de structures combinatoires [46], consiste à associer à chaque symbole non terminal N la profondeur minimale $d(N)$ d'un arbre terminal instanciant ce symbole. Ainsi, la profondeur associée au non-terminal Newton est de 2 (produit d'un coefficient élastique par un déplacement). De même, à chaque dérivation (e.g. opN_1, N_2) est associée sa

profondeur minimale ($1+max(d(N_1), d(N_2))$). Cette information est alors exploitée lors de l'initialisation pour satisfaire la contrainte de produire des arbres de profondeur inférieure à d_{max} : à une profondeur d, seules les dérivations D telles que $d(D) \leq d_{max} - d$ sont considérées.

Notons que l'usage de grammaires pour contraindre la programmation génétique (*grammar-based GP*) se prête bien aux approches de type estimation de distributions (section 4.1). En effet, une distribution sur l'espace de recherche est naturellement obtenue en associant à toute règle de dérivation de la grammaire une probabilité (*probabilistic context free grammar*) [118].

6 Les AEs comme méta-optimiseurs : couplage avec des heuristiques

Les algorithmes évolutionnaires sont des méthodes d'optimisation globales. La globalité est obtenue par les transitions probabilistes et la population, ce qui a pour conséquence que les AEs sont souvent considérés comme coûteux. Pour fixer les idées, on peut considérer qu'il n'y a pas de recherche évolutionnaire à moins de 1000 évaluations de la fonction performance.

Néanmoins, la structure des AEs offre des possibilités de couplage avec d'autre stratégies d'amélioration des solutions. Un tel couplage a un grand intérêt avec des méthodes d'optimisation locales ou des heuristiques d'amélioration rapides. Pour ne pas trop alourdir le texte, nous appellerons dans ce paragraphe toute procédure d'amélioration des solutions "heuristique", qu'elle soit mathématiquement fondée ou basée sur une expertise, qu'elle soit locale ou non. L'AE rend la recherche globale, l'heuristique contribue à son efficacité. On peut faire l'analogie avec les rôles respectifs de l'adaptation (l'AE) et l'apprentissage (l'heuristique) dans les espèces vivantes. La synergie entre AEs et heuristiques a été reconnue très tôt (voir par exemple [37], chap. 4) et a grandement contribué à la réussite des applications des AEs puisque, partant de la meilleure stratégie connue de résolution d'un problème donné, on peut souvent, si le temps de calcul le permet, l'améliorer en la couplant avec un AE.

Les couplages AEs-heuristiques prennent typiquement trois formes. L'heuristique peut être utilisée pour introduire des bons points de départ dans la population initiale. Elle peut également être ajoutée aux opérateurs de mutation. Enfin, elle peut améliorer les meilleurs individus de la dernière population. Ces possibilités sont résumées dans l'organigramme d'un EA ci-dessous.

$t \leftarrow 0$, `initialiser la pop.` [**avec une heuristique**]
`Evaluer la pop.` (f)
`Tant que continuer`
  $t \leftarrow t+1$
  `Sélection.`
  `Variations (croisement, mutation,` [**heuristique**]`).`
  `Evaluer les enfants.`
  `Remplacer certains parents par les enfants.`
`Fin tant que.`
[**heuristique sur les meilleurs individus**]

Lors du couplage entre un AE et une heuristique, il faut contrôler l'importance relative des deux (ou plus) méthodes dans le budget total d'évaluations de la fonction performance. A un extrême, si les ressources sont larges, il est possible d'utiliser les heuristiques jusqu'à leur convergence. C'est l'idée qui sous-tend les algorithmes génétiques mémétiques [68]. L'AE travaille alors dans le sous-espace des points de

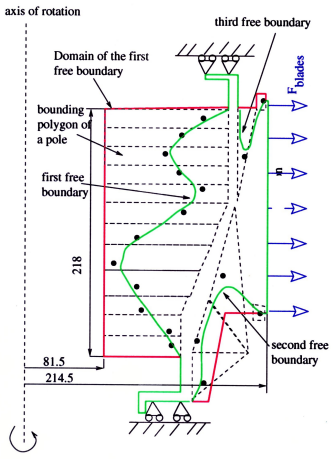


FIG. 28 – Codage de la forme d'un disque d'aubes de turbine (structure axi-symétrique) par B-splines (d'après [94]). Les pôles des B-splines sont contenus dans des polygones convexes.

convergence de l'heuristique. Si l'heuristique est un optimiseur local, c'est le sous-espace des optima locaux. A l'autre extrême, il est possible d'appliquer l'heuristique hors de l'AE, en post-traitement de la population finale. Entre ces deux extrêmes, il faut contrôler les appels à l'heuristique. Le premier critère reste les ressources en calculs. On doit aussi veiller à ce que l'heuristique n'induise pas une convergence prématurée, ce qui pourrait par exemple arriver si l'heuristique donne, trop tôt, un avantage trop important à certains individus de la population. Enfin, il peut arriver que l'heuristique produise des individus moins recombinables entre eux, auquel cas son taux d'utilisation devra être limité.

Un exemple de couplage entre AE et heuristique en optimisation de forme est donné dans [94]. Le problème d'optimisation est la minimisation du volume d'un disque d'aubes de turbines, avec des limites sur les contraintes de Von Mises et sur les déplacements. La forme est paramétrée au moyen de B-splines. Les positions des pôles des B-splines sont les variables d'optimisation, x, codées sous la forme d'un vecteur de nombres réels. Pour que les formes restent réalistes, chaque pôle est contraint de rester à l'intérieur de polygones convexes. Le problème est schématisé en FIG. 28.

L'heuristique d'amélioration des formes est la croissance biologique pénalisée. Le principe de la croissance biologique est d'ajouter ou d'enlever localement de la matière à la surface du solide dans les zones de fortes ou faibles contraintes de Von Mises. La pénalisation réalise une dilatation ou une contraction globale du solide quand les ajouts locaux de matière ne permettent plus de réduire ou d'augmenter les contraintes de Von Mises maximales. La FIG. 29 illustre l'effet de la croissance biologique sur un disque d'aube de turbine.

Plusieurs stratégies de couplages ont été comparées dans [94] : la croissance biologique est appliquée à la population initiale, au plus mauvais individu de la population courante ou à l'ensemble de la population, avec une fréquence donnée. Un individu enfant est soit généré par les opérateurs de variation usuels, soit par copie d'un parent et amélioration par croissance biologique. Les tests réalisés montrent qu'il est préférable d'appliquer l'heuristique à l'ensemble de la population. Dans ce cas, une bonne fréquence d'application était que, toutes les 9 générations, tous les individus de la population sont améliorés par une itération de croissance biologique. Cette stratégie couplée est plus efficace qu'un AE pur, que l'heuristique seule et

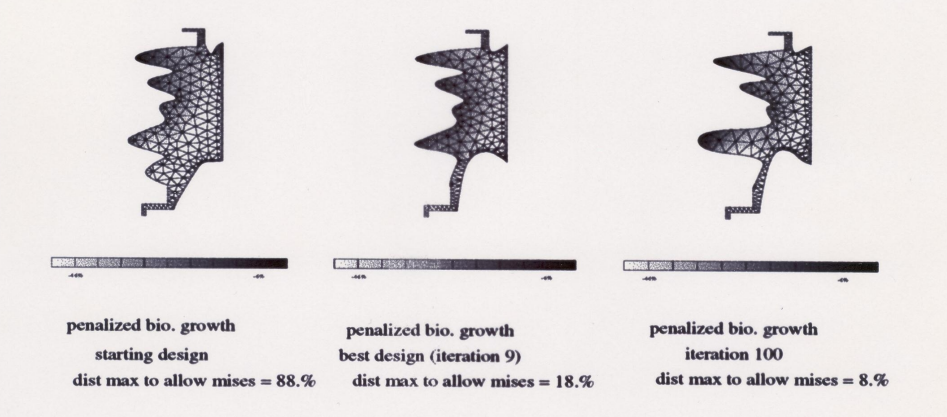


FIG. 29 – Effet de l'heuristique de "croissance biologique pénalisée" sur la forme d'un disque fan décrit par B-splines (d'après [94])

.

qu'une recherche aléatoire. Un exemple de design obtenu est donné en FIG. 30.

7 Conclusions : la mollesse et l'utilité des algorithmes évolutionnaires

Cet article a cherché à présenter les développements récents en optimisation évolutionnaire du point de vue des applications pour l'ingénieur. La perception des grandes possibilités ouvertes par les AEs a évolué. Dans une première époque exubérante l'inspiration darwinienne s'est accompagnée d'une recherche d'un optimiseur universel. Dans une seconde époque, il est apparu clairement que la différence entre la potentialité théorique d'une méthode, et ses capacités effectives, repose sur l'usage des connaissances du domaine. Le cœur des recherches s'intéresse ainsi à l'adéquation entre un optimiseur évolutionnaire et le problème d'optimisation posé.

Il commence aujourd'hui à exister des mesures de l'adéquation entre un AE et sa classe de fonctions (section 3.4). Les AE modernes sont adaptatifs (section 4).

Des sujets importants, où les AEs sont déjà des méthodes de référence ont été omis : l'optimisation de fonctions bruitées, l'optimisation sous contraintes, et la parallélisation. Le lecteur désireux de combler ces lacunes peut consulter, par exemple, [4, 19, 2], [103, 104, 14, 32] et [26, 25] sur chacun de ces sujets. Un autre aspect concerne l'optimisation des fonctions chères, où l'évolution est couplée avec l'apprentissage pour construire un modèle simplifié de la fonction objectif ; ce modèle simplifié permet de réduire considérablement le nombre d'appels à la fonction objectif (*surrogate optimisation*). Le lecteur intéressé se reportera à [115, 72].

Au sein de la discipline générale de l'optimisation, les algorithmes évolutionnaires occupent une place particulière. On ne peut qu'être frappé par la généralité de l'espace de recherche $\mathcal{S}$ dans lequel ils travaillent. La seule condition sur $\mathcal{S}$ est qu'il soit topologique, peu importe qu'il soit continu, discret, mixte, paramétrique ou non. Les AEs sont les méthodes d'optimisation les plus proches de la formulation du problème à résoudre puisqu'on peut y coder les variables et redéfinir les opérateurs de variations. Enfin, les AEs sont, en général, faciles à coupler avec d'autres optimiseurs tels que des méthodes de programmation mathématique ou des heuristiques propres au problème traité. Nous pensons que ces nombreux degrés de liberté, qui ont parfois valu aux AEs (et parfois à juste titre !) le qualificatif de méthodes "molles", sont au contraire les qualités qui leur permettent aujourd'hui déjà et demain

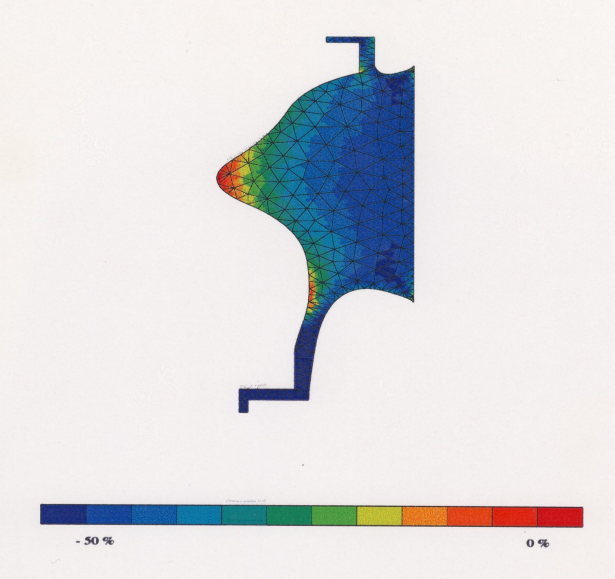


FIG. 30 – Design optimal de disque d'aubes de turbine obtenu par un couplage entre AE et croissance biologique pénalisée. La matière est excentrée par rapport à l'axe de rotation pour accroître la résistance en flexion du disque. Les contraintes sur la rupture de la pièce sont violées de moins de 1% [94].

davantage d'aborder les problèmes les plus difficiles (e.g., les problèmes en hautes dimensions sans gradients connus, les problèmes non-paramétriques, les problèmes bruités).

Références

[1] G. Allaire, E. Bonnetier, G. Francfort, and F. Jouve. Shape optimization by the homogenization method. *Nümerische Mathematik*, 76 :27–68, 1997.

[2] P.J. Angeline. The effects of noise on self-adaptive evolutionary optimization. In L. J. Fogel, P. J. Angeline, and T. Bäck, editors, *Proceedings of the* 5^{th} *Annual Conference on Evolutionary Programming*, pages 433–439. MIT Press, 1996.

[3] J. Antonisse. A new interpretation of schema notation that overturns the binary encoding constraint. In J. D. Schaffer, editor, *Proceedings of the* 3^{rd} *International Conference on Genetic Algorithms*, pages 86–91. Morgan Kaufmann, June 1989.

[4] D. V. Arnold and H.-G. Beyer. Local performance of the $(\mu/\mu_i,\lambda)$-es in a noisy environment. In W. N. Martin and W. M. Spears, editors, *Foundations of Genetic Algorithms 6*, pages 127–141. Morgan Kaufmann, 2001.

[5] A. Auger. *Contributions théoriques et numériques à l'optimisation continue par algorithmes évolutionnaires*. PhD thesis, Université Paris 6, December 2004. in French.

[6] A. Auger, M. Schoenauer, and O. Teytaud. Local and global order 3/2 convergence of a surrogate evolutionary algorithm. In K. Deb et al., editor, *Proceedings of the Genetic and Evolutionary Conference 2004*. LNCS 3102 and 3103, Springer Verlag, 2004.

[7] T. Bäck. Optimal mutation rates in genetic search. In S. Forrest, editor, *Proceedings of the* 5^{th} *International Conference on Genetic Algorithms*, pages 2–8. Morgan Kaufmann, 1993.

[8] T. Bäck. Selective pressure in evolutionary algorithms : a characterization of selection mechanisms. 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 57–62, 1994.

[9] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford Univ. Press, New-York, 1996.

[10] T. Bäck, M. Schütz, and S. Khuri. A comparative study of a penalty function, a repair heuristic and stochastic operators with set covering problem. In J.-M. Alliot, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers, editors, *Artificial Evolution*, number 1063 in LNCS. Springer Verlag, 1995.

[11] S. Baluja. Population-based incremental learning : a method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, Carnegy Mellon University, Pittsburg, 1994.

[12] S. Baluja. An empirical comparison of seven iterative and evolutionary function optimization heuristics. Technical Report CMU-CS-95-193, Carnegie Mellon University, 1995.

[13] W. Banzhaf, P. Nordin, R.E. Keller, and F.D. Francone. *Genetic Programming — An Introduction On the Automatic Evolution of Computer Programs and Its Applications*. Morgan Kaufmann, 1998.

[14] S. BenHamida. *Algorithmes Évolutionnaires : Prise en Compte des Contraintes et Application Réelle*. PhD thesis, Université de Paris 11 – Orsay, 29 mars 2001.

[15] P. J. Bentley, editor. *Evolutionary Design by Computers*. Morgan Kaufman Publishers Inc., 1999.

[16] P. J. Bentley and D. W. Corne, editors. *Creative Evolutionary Systems*. Morgan Kaufmann, 2001.

[17] J.-M. Berthelot. *Matériaux Composites : comportement mécanique et analyse des structures*. Technique & Documentation, Paris, 3 edition, 1999.

[18] H.-G. Beyer. An alternative explanation for the manner in which genetic algorithms operate. *BioSystems*, 41 :1–15, 1997.

[19] H.-G. Beyer. Mutate large, but inherit small ! On the analysis of mutations in $(1, \lambda)$-ES with noisy fitness data. In Th. Bäck, A.E. Eiben, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of the* 5^{th} *Conference on Parallel Problems Solving from Nature*, pages 109–118. Springer Verlag, 1998.

[20] H.-G. Beyer. *The Theory of Evolution Strategies*. Springer, 2001.

[21] H.-G. Beyer and H.-P. Schwefel. Evolution strategies : a comprehensive introduction. *Natural Computing*, 1 :3–52, 2002.

[22] J.-D. Boissonnat and M. Yvinec. *Géométrie algorithmique*. Ediscience International, 1995.

[23] P. A. N. Bosman and D. Thierens. Expanding from discrete to continuous estimation of distribution algorithms : The IDEA. In M. Schoenauer et al., editor, *Proceedings of the* 6^{th} *Conference on Parallel Problems Solving from Nature*, pages 767–776. Springer-Verlag, LNCS 1917, 2000.

[24] A.H. Aguirre C.A. Coello Coello and E. Zitzler, editors. *Proceedings of Evolutionary Multi-Critrion Optimization'05*. Springer Verlag, LNCS 3410, 2005.

[25] S. Cahon, N. Melab, E-G. Talbi, and M. Schoenauer. ParaDisEO-based design of parallel and distributed evolutionary algorithms. In Pierre Liardet et al., editor, *Artificial Evolution 03*, pages 215–227. LNCS 2936, Springer Verlag, 2003.

[26] E. Cantu-Paz. *Efficient and Accurate Parallel Genetic Algorithms*. Kluwer Academic Pub, 2000.

[27] R. Cerf. *Une théorie asymptotique des algorithmes génétiques*. PhD thesis, Université de Montpellier II, France, 1994.

[28] C. D. Chapman and M. J. Jakiela. Genetic algorithm-based structural topology design with compliance and topology simplification considerations. *ASME Journal of Mechanical Design*, 118 :89–98, 1996.

[29] C. D. Chapman, K. Saitou, and M. J. Jakiela. Genetic algorithms as an approach to configuration and topology design. *Journal of Mechanical Design*, 116 :1005–1012, 1994.

[30] P. G. Ciarlet. *Mathematical Elasticity, Vol I : Three-Dimensional Elasticity*. North-Holland, Amsterdam, 1978.

[31] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, 2002.

[32] C. A. Coello Coello. Theoretical and numerical constraint handling techniques used with evolutionary algorithms : A survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, pages 1245–1287, 2002.

[33] N.J. 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. Laurence Erlbaum Associates, 1985.

[34] N. A. C. Cressie. *Statistics for spatial data*. Wiley, revised edition, 1993.

[35] J.M. Daida. Challenges with verification, repeatability, and meaningful comparison in genetic programming : Gibson's magic. In *Proceedings of the Genetic and Evolutionary Conference 99*, pages 1069–1076. Morgan Kaufmann, 1999.

[36] L. Davis. Applying adaptive algorithms to epistatic domains. In *Proc. Intl. Joint Conference on Artificial Intelligence*, 1985.

[37] L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[38] K. Deb. *Multi-objective optimization using genetic algorithms*. Wiley, 2001.

[39] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization : Nsga-ii. In M. Schoenauer et al., editor, *Proceedings of the* 6^{th} *Conference on Parallel Problems Solving from Nature*, pages 849–858. Springer-Verlag, LNCS 1917, 2000.

[40] K. Deb and H.-G. Beyer. Self-adaptive genetic algorithms with simulated binary crossover. *Evolutionary Computation*, 2001.

[41] K. A. DeJong. Are genetic algorithms function optimizers ? In R. Manner and B. Manderick, editors, *Proceedings of the* 2^{nd} *Conference on Parallel Problems Solving from Nature*, pages 3–13. North Holland, 1992.

[42] 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).

[43] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, 2 edition, 2001.

[44] M. Ebner. Evolutionary design of objects using scene graphs. In C. Ryan, T. Soule, M. Keijzer, E. Tsang, R. Poli, and E. Costa, editors, *Proc. EuroGP 2003*. Springer-Verlag, 2003.

[45] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer Verlag, 2003.

[46] P. Flajolet, P. Zimmerman, and B. Van Cutsem. A calculus for the random generation of labelled combinatorial structu res. *Theor. Comput. Sci.*, 132(1-2) :1–35, 1994.

[47] D. B. Fogel. *Evolving artificial intelligence*. PhD thesis, University of California, San diego, CA, 1992.

[48] D. B. Fogel. *Evolutionary Computation. Toward a New Philosophy of Machine Intelligence*. IEEE Press, Piscataway, NJ, 1995.

[49] L. J. Fogel. Autonomous automata. *Industrial Research*, 4 :14–19, 1962.

[50] C.M. Fonseca, P.J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors. *Proceedings of Evolutionary Multi-Critrion Optimization'03*. Springer Verlag, LNCS 2632, 2003.

[51] D. E. Goldberg. *The Design of Innovation : Lessons from and for Competent Genetic Algorithms*. Kluwer, 2002.

[52] D. E. Goldberg, K. Deb, and J. H. Clark. Genetic algorithms, noise, and the sizing of populations. Technical Report IlliGAL 90007, Univ. of Illinois at Urbana-Champaign, 1991.

[53] D. E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms : Motivations, analysis and first results. *Complex Systems*, 3 :493–530, 1989.

[54] D. E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms revisited : Nonuniform size and scale. *Complex Systems*, 4 :415–444, 1990.

[55] D. E. Goldberg and R. E. Smith. Nonstationary function optimization using genetic algorithms with dominance and diploidy. In *Proceedings of the* 2^{nd} *International Conference on Genetic Algorithms*, pages 59–68. Morgan Kaufmann, 1987.

[56] David E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, 1989.

[57] J. J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Trans. on Systems, Man and Cybernetics*, SMC-16(1) :122–128, 1986.

[58] L. Grosset. *Optimization of Composite Structures by Estimation of Distribution Algorithms*. PhD thesis, Univ. of Florida and Ecole Nationale Supérieure des Mines de Saint-Etienne, September 2004.

[59] L. Grosset, R. Le Riche, and R. T. Haftka. A double-distribution statistical algorithm for composite laminate optimization. *Structural and Multidisciplinary Optimization*, 31(1) :49–59, January 2006.

[60] F. Gruau. On using syntactic constraints with genetic programming. In P.J. Angeline and K.E. Kinnear Jr., editors, *Advances in Genetic Programming II*, pages 377–394. MIT Press, 1996.

[61] H. Hamda, F. Jouve, E. Lutton, M. Schoenauer, and M. Sebag. Compact unstructured representations in evolutionary topological optimum design. *Applied Intelligence*, 16 :139–155, 2002.

[62] H. Hamda, O. Roudenko, and M. Schoenauer. Multi-objective evolutionary topological optimum design. In I. Parmee, editor, *Evolutionary Design and Manufacture*, pages 121–132. Springer Verlag, 2002.

[63] H. Hamda and M. Schoenauer. Toward hierarchical representations for evolutionary topological optimum design. In J. Périaux et al., editor, *Eurodays 2000, in memoriam of B. Mantel*. John Wiley, 2002.

[64] N. Hansen, S. Müller, and P. Koumoutsakos. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolution Computation*, 11(1), 2003.

[65] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies : The covariance matrix adaption. In *Proceedings of the Third IEEE International Conference on Evolutionary Computation*, pages 312–317. IEEE Press, 1996.

[66] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2) :159–195, 2001.

[67] N. Hansen, A. Ostermeier, and A. Gawelczyk. On the adaptation of arbitrary normal mutation distributions in evolution strategies : The generating set adaptation. In L. J. Eshelman, editor, *Proceedings of the* 6^{th} *International Conference on Genetic Algorithms*, pages 57–64. Morgan Kaufmann, 1995.

[68] W. E. Hart, N. Krasnogor, and J.E. Smith. Recent advances in memetic algorithms. In *Series : Studies in Fuzziness and Soft Computing*, volume 166. Springer, 2005.

[69] John H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

[70] Christian Igel, Nikolaus Hansen, and Stefan Roth. Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 614(4), 2006.

[71] E. Jensen. *Topological Structural Design using Genetic Algorithms*. PhD thesis, Purdue University, November 1992.

[72] Y. Jin, M. Olhofer, and B. Sendho. A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*, 6(5) :481–494, 2002.

[73] D. Jones, M. Schonlau, and W. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13 :455–492, 1998.

[74] T. Jones. *Evolutionary algorithms, fitness landscapes and search*. PhD thesis, University of New Mexico, 1995.

[75] K. A. De Jong. *An analysis of the behavior of a class of genetic adaptative systems*. PhD thesis, University of Michigan, Ann Arbor, MI, 1975.

[76] K. A. De Jong. *Evolutionary Computation : a unified approach*. MIT Press, 2006.

[77] L. Kallel. *Convergence des algorithmes génétiques : aspects spatiaux et temporels*. PhD thesis, Ecole Polytechnique, Palaiseau, France, Feb. 1999.

[78] L. Kallel and M. Schoenauer. Alternative random initialization in genetic algorithms. In Th. Bäck, editor, *Proceedings of the* 7^{th} *International Conference on Genetic Algorithms*, pages 268–275. Morgan Kaufmann, 1997.

[79] C. Kane. *Algorithmes génétiques et Optimisation topologique*. PhD thesis, Université de Paris VI, July 1996.

[80] C. Kane, F. Jouve, and M. Schoenauer. Structural topology optimization in linear and nonlinear elasticity using genetic algorithms. In *Proceedings of the ASME 21st Design Automation Conference*. ASME, Boston, Sept. 1995.

[81] C. Kane and M. Schoenauer. Genetic operators for two-dimensional shape optimization. In J.-M. Alliot, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers, editors, *Artificial Evolution*, number 1063 in LNCS. Springer Verlag, Septembre 1995.

[82] C. Kane and M. Schoenauer. Topological optimum design using genetic algorithms. *Control and Cybernetics*, 25(5) :1059–1088, 1996.

[83] C. Kane and M. Schoenauer. Optimisation topologique de formes par algorithmes génétiques. *Revue Française de Mécanique*, 4 :237–246, 1997.

[84] S. Kern, N. Hansen, S. Müller, D. Büche, J. Ocenasek, and P. Koumoutsakos. Distributions in continuous evolutionary algorithms - review and comparison. *Natural Computing*, 3(1) :77–112, 2004.

[85] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220 :671–680, 1983.

[86] N. Kogiso, L.T. Watson, Z. Gürdal, and R.T. Haftka. Genetic algorithms with local improvement for composite laminate design. *Structural Optimization*, 7(4) :207–218, 1994.

[87] N. Kogiso, L.T. Watson, Z. Gürdal, R.T. Haftka, and Nagendra S. Design of composite laminates by a genetic algorithm with memory. *Mechanics of Composite Materials and Structures*, 1(1) :95–117, 1994.

[88] J. R. Koza. *Genetic Programming : On the Programming of Computers by means of Natural Evolution*. MIT Press, Massachusetts, 1992.

[89] J. R. Koza. *Genetic Programming II : Automatic Discovery of Reusable Programs*. MIT Press, Massachussetts, 1994.

[90] J. R. Koza and al. *Genetic Programming III : Automatic Synthesis of Analog Circuits*. MIT Press, Massachussetts, 1999.

[91] W. B. Langdon and W. Banzhaf. Genetic programming bloat without semantics. In Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, Juan Julian Merelo, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VI 6th International Conference*, volume 1917, pages 201–210, Paris, France, 2000. Springer Verlag.

[92] P. Larranaga and J. A. Lozano. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2001.

[93] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Archiving with guaranteed convergence and diversity in multi-objective optimization. In W.B. Langdon & al., editor, *Proceedings of the Genetic and Evolutionary Conference 2002*, pages 39–447. Morgan Kaufmann, 2002.

[94] R. Le Riche and G. Cailletaud. A mixed evolutionary/heuristic approach to shape optimization. *International Journal of Numerical Methods in Engineering*, 41 :1463–1484, 1998.

[95] R. Le Riche and R.T. Haftka. Optimization of laminate stacking sequence for buckling load maximization by genetic algorithm. *AIAA Journal*, 31(5) :951–970, May 1993.

[96] R. Le Riche and R.T. Haftka. Improved genetic algorithm for minimum thickness composite laminate design. *Composites Engineering*, 5(2) :143–161, 1995.

[97] R. Le Riche, C. Knopf-Lenoir, and R. T. Haftka. A segregated genetic algorithm for constrained structural optimization. In *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA95)*, pages 558–565. Morgan Kaufman, 1995.

[98] S. Lin and B. Kernighan. An efficient heuristic procedure for the traveling salesman problem. *Operations Res.*, 21 :498–516, 1973.

[99] S. W. Mahfoud. *Niching Methods for Genetic Algorithms*. PhD thesis, University of Illinois, Urbana Champaign, 1995. also Illigal Report 95001.

[100] S. W. Mahfoud and D. E. Goldberg. Parallel recombinative simulated annealing : a genetic algorithm. *Parallel Computing*, 21 :1–28, 1995.

[101] W. N. Martin, J. Liening, and J. P. Cohoon. *Evolutionary Computation 2 : Advanced Algorithms and Operators*, chapter Island (migration) models : evolutionary algorithms based on punctuated equilibria, pages 101–124. Inst. of Physics Editor, Bristol, UK, 2000. T. Bäck, D. B. Fogel and Z. Michalewicz, eds.

[102] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer Verlag, 3 edition, 1992-1996.

[103] Z. Michalewicz, D. Dasgupta, R. Leriche, and M. Schoenauer. Evolutionary algorithms for constrained engineering problems. *Computers & Industrial Engineering Journal*, 30(2), April 1996.

[104] Z. Michalewicz and M. Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, 4(1) :1–32, 1996.

[105] M. Minoux. *Mathematical programming : Theory and Algorithms*. Wiley, 1986.

[106] H. Mühlenbein. The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5 :303–346, 1998.

[107] H. Mühlenbein and T. Mahnig. *Theoretical aspects of evolutionary computing*, chapter Evolutionary algorithms : from recombination to search algorithms, pages 135–174. Springer, 2000.

[108] S. Obayashi. Pareto solutions of multipoint design of supersonic wings using evolutionary algorithms. In I.C. Parmee, editor, *Adaptive Computing in Design and Manufacture V*, pages 3–15. Springer-Verlag, 2002.

[109] J. Paredis. Co-evolutionary 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.

[110] M. Pelikan, D.E. Goldberg, and E. Cantu-Paz. Boa : the bayesian optimization algorithm. In D.E. Goldberg & al., editor, *Proceedings of the Genetic and Evolutionary Conference 1999*, pages 525–532. Morgan Kaufmann, 1999.

[111] Le Riche R. *Optimization of composite structures by genetic algorithms*. PhD thesis, Virginia Polytechnic Inst. and State University, Blacksburg, VA, USA, Oct. 1994.

[112] N. J. Radcliffe. Equivalence class analysis of genetic algorithms. *Complex Systems*, 5 :183–20, 1991.

[113] N. J. Radcliffe. Genetic set recombination. In Foundations of Genetic Algorithms 2, editor, *Foundations of Genetic Algorithms 2*, pages 203–220. Morgan Kaufmann, 1993.

[114] N. J. Radcliffe and P. D. Surry. Fitness variance of formae and performance prediction. In L. D. Whitley and M. D. Vose, editors, *Foundations of Genetic Algorithms 3*, pages 51–72. Morgan Kaufmann, 1995.

[115] A. Ratle. Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In Th. Bäck, G. Eiben, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of the* 5^{th} *Conference on Parallel Problems Solving from Nature*, pages 87–96. Springer-Verlag, LNCS 1498, 1998.

[116] A. Ratle and M. Sebag. Genetic programming and domain knowledge : Beyond the limitations of grammar-guided machine discovery. In M. Schoenauer et al., editor, *Proceedings of the* 6^{th} *Conference on Parallel Problems Solving from Nature*, pages 211–220. Springer-Verlag, LNCS 1917, 2000.

[117] A. Ratle and M. Sebag. Grammar-guided genetic programming and dimensional consistency : Application to non-parametric identification in mechanics. *Applied Soft Computing*, 1(3) :105–118, 2001.

[118] A. Ratle and M. Sebag. Avoiding the bloat with stochastic grammar-based genetic programming. In P. Collet, E. Lutton, M. Schoenauer, C. Fonlupt, and J.-K. Hao, editors, *Artificial Evolution'01*, pages 254–266. Springer Verlag, LNCS 2310, 2002.

[119] I. Rechenberg. *Cybernetic solution path of an experimental problem.* Royal Aircraft Establishment, Farnborough, Library Translation 1122, 1965.

[120] I. Rechenberg. *Evolutionsstrategie : Optimierung technischer System nach Principen der Biologischen Evolution.* Frommann-Holzboog, Stuttgart, 1973.

[121] C. Reeves and C. Wright. Epistasis in genetic algorithms : an experimental design perspective. In L. J. Eshelman, editor, *Proceedings of the* 6^{th} *International Conference on Genetic Algorithms*, pages 217–230. Morgan Kaufmann, 1995.

[122] G. Rudolf. Finite markov chain results in evolutionary computation : A tour d'horizon. *Fundamenta Informaticae*, 35(1-4) :67–89, 1998.

[123] F. Schoen. Stochastic techniques for global optimization : a survey of recent advances. *Journal of Global Optimization*, 1 :207–228, 1990.

[124] 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.

[125] H.-P. Schwefel. *Numerical Optimization of Computer Models.* John Wiley & Sons, New-York, 1981. 1995 – 2^{nd} edition.

[126] H.-P. Schwefel and G. Rudolf. Contemporary evolution strategies. In F. Morán et al., editor, *Proc. of the* 3^{rd} *Int. Conf. on Artificial Life*, pages 893–907. Springer, 1995. Lecture Notes in Artificial Intelligence 929.

[127] B. Schölkopf, C. Burges, and A. Smola. *Advances in Kernel Methods : Support Vector Machines.* MIT Press, Cambridge, MA, 1998.

[128] M. Sebag and A. Ducoulombier. Extending population-based incremental learning to continuous search spaces. In Th. Bäck, A.E. Eiben, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of the* 5^{th} *Conference on Parallel Problems Solving from Nature*, pages 418–427. Springer Verlag, 1998.

[129] M. Sebag, M. Schoenauer, and M. Peyral. Revisiting the memory of evolution. *Fundamenta Informaticae*, 38 :1–39, 1998.

[130] H. Simon. *Models of Bounded Rationality.* Cambridge, MIT Press, 1982.

[131] R. E. Smith and E. Smuda. Adaptatively resizing populations : algorithm, analysis and first results. *Complex Systems*, 9 :47–72, 1995.

[132] G. Soremekun, Z. Gürdal, R.T. Haftka, and L.T. Watson. Composite laminate design optimization by genetic algorithm with generalized elitist selection. *Computers & Structures*, 79(2) :131–143, 2001.

[133] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Technical Report KanGAL Report 2005005, IIT Kanpur, India, 2005. see http ://www.bionik.tu-berlin.de/user/niko/cec2005.html.

[134] V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[135] M. D. Vose. *The Simple Genetic Algorithm*. MIT Press, Cambridge, MA, 1999.

[136] P.A. Whigham. Inductive bias and genetic programming. In *IEEE Conference publications, n. 414*, pages 461–466, 1995.

[137] L. D. Whitley. Fundamental principles of deception in genetic search. In G. J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 221–241. Morgan Kaufmann, 1991.

[138] L. D. Whitley, T. Starkweather, and D. Shaner. *Handbook of Genetic Algorithms*, chapter Traveling salesman and sequence scheduling : quality solutions using genetic edge recombination, pages 350–372. Van Nostrand Reinhold, New York, 1991. L. Davis, ed.

[139] D. H. Wolpert and W. G. MacReady. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute, Santa Fe, NM, 1995.

[140] E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, and D. Corne, editors. *Proceedings of Evolutionary Multi-Critrion Optimization'01*. Springer Verlag, LNCS 1993, 2001.