Table des matières

Introduction

Le travail présenté ici concerne l'optimisation sous contraintes dans le contexte de l'optimisation des formes en Mécanique des Solides. Nous nous restreindrons aux méthodes d'optimisation stochastiques que sont les Algorithmes Génétiques.

Les méthodes d'optimisation, peuvent se diviser en deux classes : l'approche déterministe et l'approche stochastique. Dans les deux cas il s'agit de méthodes itératives, pour lesquelles l'étape élémentaire consiste à déterminer le ou les points de l'espace de recherche à examiner à l'étape $n+1$ en fonction du ou des points connus à l'étape n.

Les méthodes déterministes utilisent des informations sur les valeurs au point courant de la fonction, des contraintes, ainsi que de leur gradient. Les méthodes stochastiques ne nécessitent aucune hypothèse de régularité (continuité, différentiabilité) portant sur la fonction à optimiser ou les contraintes. En particulier, il suffit de disposer d'une description algorithmique (calculable) de l'objectif pour que les approches stochastiques soient applicables. En Pratique, les méthodes stochastiques utilisent des tirages aléatoires pour déterminer les points de l'espace à explorer à l'étape $n+1$.

Les avantages et inconvénients des deux types de méthodes sont connus : les méthodes déterministes de type gradient garantissent une convergence robuste ... vers l'optimum local le plus proche. Les méthodes stochastiques n'offrent que des résultats de convergence en probabilité, avec une vitesse de convergence généralement très lente, mais sont par contre capables de localiser, quand il existe l'optimum global, même pour des fonctions ayant de nombreux optima locaux.

L'optimisation de formes recherche une configuration géométrique d'un objet répondant à un cahier des charges concernant ses propriétés physiques. Les fonctions à optimiser et les contraintes sont généralement obtenus à l'aide de simulations numériques. D'autre part, la définition de l'espace de recherche lui-même est le résultat d'un compromis : si les formes sont décrites par un petit nombre de paramètres, le problème d'optimisation est plus facile, mais l'ensemble des solutions possibles est limité. Si par contre on élargit l'espace de recherche, le problème d'optimisation devient très difficile, voire insoluble par des méthodes déterministes. Certains problèmes d'optimisation de formes donnent cependant naissance à des problèmes bien posés, pour lesquels la solution existe, est unique dans un espace suffisamment grand, et peut être approchée par une méthode de type gradient. C'est le cas de l'optimisation de domaine et de la méthode d'optimisation par homogénéisation de structures élastiques.

Cependant, dans de nombreux cas, les problèmes mathématiques posés par les problèmes d'optimisation de formes sont définis sur des espaces non-standard, par des fonctions et des contraintes peu régulières et possédant de nombreux optima locaux. Dans ce contexte, les méthodes stochastiques sont tout indiquées. Parmi les travaux antérieurs concernant l'utilisation des algorithmes génétiques en Mécanique des Structures, beaucoup ont concerné l'optimisation d'assemblages de barres. Goldberg et Samtani (1986), Hajela(1992), Lin et Hajela(1992) et Schoenauer et Wu (1995) ont résolus des problèmes de dimensionnement de section ; Hajela et Al(1993) , Grierson et Pak (1993) et Wu (1996) ont abordé le problème de l'optimisation topologique de treillis de barres. Des résultats intéressants ont également été obtenus pour l'optimisation des empilements de matériaux composites par Leriche et Haftka (1993), Leriche (1994). Dans le domaine de l'optimisation topologique de formes, les premiers travaux utilisant les algorithmes génétiques sont dus à Jensen(1992) et à Chapman, Saitou et Jakiela (1994). Citons enfin dans ce domaine Ghaddar, Maday et Patera (1994): cette approche, fondée sur le recuit simulé, a tout d'abord permis de définir un espace de description des formes polygonales possédant les propriétés d'approximation souhaitées. Ces résultats ont été appliqués à l'optimisation de la section droite d'une barre en maximisant son moment d'inertie, ainsi qu'à l'optimisation de la forme d'un radiateur.

Nos travaux se situent dans la continuation de ces travaux antérieurs, dans le cadre théorique proposé par Ghaddar, Maday et Patera (1994) et partant d'une approche par algorithmes génétiques semblable à celle proposée par Chapman, Saitou et Jakiela (1994). L'originalité de nos travaux du point de vue des algorithmes génétiques réside dans la prise en compte des spécificités du problème de l'optimisation de formes au niveau de l'algorithme lui-même : ainsi, une étude approfondie du codage et des opérateurs de croisement a amené la définition d'opérateurs spécifiques. Du point de vue de l'optimisation de forme, cette utilisation des algorithmes génétiques a permis d'obtenir de nombreux résultats originaux, tels les premiers résultats d'Optimum Design en élasticité non-linéaire.

Le plan de la thèse est le suivant : le problème de l'Optimisation de Formes est introduit au premier chapitre, et les méthodes déterministes de variation de domaine et d'homogénéisation sont rappelées brièvement. Les algorithmes génétiques, qui sont au cœur de ce travail, sont détaillés dans le chapitre 2. Le chapitre 3 décrit la résolution que nous avons effectuée d'un problème d'optimisation de formes paramétrique validant l'approche algorithmes génétiques dans le domaine de l'optimisation de forme en général. Le chapitre 4 rappelle le cadre de travail théorique défini par Ghaddar, Maday et Patera (1994) pour l'optimisation de forme topologique. Le chapitre 5 est la description du codage et des opérateurs que nous avons imaginés pour pallier aux déficiences des opérateurs génétiques standards. Ces opérateurs sont validés sur le problème simple de l'optimisation de la section d'une barre. Le problème d'Optimum Design en élasticité est abordé dans le chapitre 6 : le modèle est décrit, puis différents types de performance sont présentés et comparés. Par ailleurs, la méthode des algorithmes génétiques est comparée en détail avec la méthode d'homogénéisation. Enfin, le chapitre 7 présente les principaux résultats obtenus dans le domaine de l'Optimum Design : obtention de multiples solutions quasi-optimales, optimisation d'une structure en fonction de plusieurs configurations de chargement, ou pour résister à un chargement sur la frontière inconnue, et enfin les premiers résultats en élasticité non-linéaire. La dernière partie tire les conclusions de ce travail, et ouvre des voies pour des recherches ultérieures.

Chapitre 1

Optimisation de formes

Dans ce chapitre nous posons le problème de l'Optimisation de Formes et donnons une idée de l'état de l'art en terme de méthodes spécifiques d'optimisation utilisées dans la littérature. Nous détaillons les trois instances du problème que sont – de la plus simple à la plus complexe – l'optimisation des dimensions, qui se résume à un problème d'optimisation paramétrique, l'optimisation de domaine et l'optimisation topologique. Nous donnons les grandes lignes des méthodes classiques de résolution de ces deux problèmes que sont la méthode de variation de domaine et la méthode d'homogénéisation. Les limitations de cette dernière nous conduisent à utiliser des méthodes stochastiques (recuit simulé et algorithmes génétiques). La présentation détaillée de ces méthodes sera faite au chapitre suivant, compte-tenu d'une part de leur relative nouveauté et d'autre part de ce que les algorithmes génétiques, ainsi que leur adaptation au problème de l'optimisation de formes, constituent la méthode utilisée dans cette thèse.

1.1 Généralités

L'Optimisation de Formes cherche une description géométrique d'un objet de dimension 2 ou 3 qui doit optimiser une fonction généralement appelée fonction coût tout en respectant un certains nombre de contraintes. La fonction coût et/ou les contraintes sont liées au comportement physique de l'objet considéré.

Le problème-type que nous considérerons dans toute cette thèse est le problème de la minimisation du poids d'une structure solide avec des contraintes sur son comportement mécanique.

Un exemple classique est donné figure1: le problème de la plaque-console (cantilever dans la littérature anglo-saxonne) consiste à minimiser le poids d'une plaque contenue dans le domaine défini par le rectangle tout en imposant que le déplacement maximum de cette plaque sous un chargement donné (ici une force appliquée en un point donné) ne dépasse pas une valeur limite imposée. Nous détaillerons toutes les hypothèses mécaniques dans le chapitre 6.

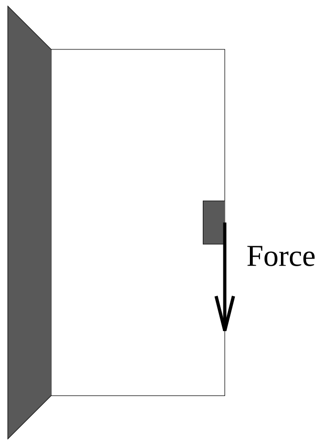


Figure 1 : *Exemple type de la plaque console(“cantilever”)*

On distingue en optimisation de formes trois grandes familles de problèmes, suivant les degrés de liberté laissés à l'optimisation : *l'optimisation des dimensions*, *l'optimisation de structure* et enfin *l'optimisation topologique de forme*. Nous allons en donner les traits caractéristiques et les illustrer par trois exemples sur le problème de la plaque console.

1.1.1 Optimisation des dimensions

Elle consiste à déterminer les dimensions géométriques d'une classe de formes données ; il faut par exemple trouver l'épaisseur d'une coque, la taille d'une barre dans un treillis, ou encore le rayon d'une tige métallique dans un assemblage.
L'optimisation des dimensions permet seulement la variation d'un certain nombre de paramètres de contrôle. Elle est par conséquent limitée.

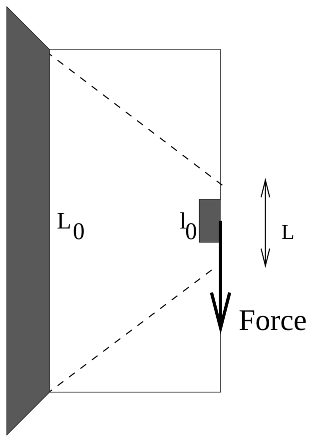


Figure 2 : *Optimisation dimensionnelle à un seul paramètre L.*

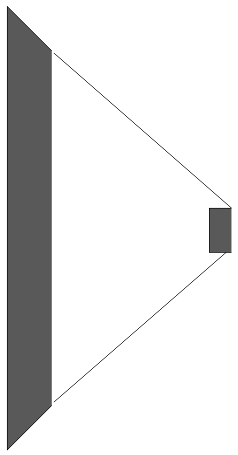


Figure 3 : *solution possible*

Exemple
On considère le problème de minimisation du poids de la plaque console décrite figure 2. Dans cette partie la structure est supposée de forme trapèzoïdale, et on doit trouver la longueur optimale L. Dans ce cas on a à résoudre le problème suivant :

$$\left\{\begin{array}{ll} \min\limits_{L} J(L) & = \frac{1}{2}(L+L_0), \quad l_0 \leq L \leq L_0 \\ D_{Max} < D_{Lim} & \end{array}\right.$$

si D_{Max} est le déplacement de la plaque sous la charge $\vec{F}$ et D_{Lim} est la valeur imposée que ne doit pas dépasser ce déplacement.

Une solution possible est représentée figure 3. Remarquons que dans ce cas très simple, compte-tenu de la monotonie de D_{Max} par rapport à L, une méthode de résolution par dichotomie peut traiter le problème : on cherche L tel que $D_{Max} = D_{Lim}$. Il reste bien entendu à préciser comment est calculé D_{Max}, ce qui sera détaillé au chapitre 6.

1.1.2 Optimisation de domaine

Encore appelée optimisation géométrique [51] [53], elle consiste à rajouter par rapport à la première classe de problèmes d'autres variables qui permettent des mouvements du contour plus importants. Les variables (encore appelées degrés de liberté) du problème peuvent être simples (trouver le rayon d'une tige) ou beaucoup plus complexes (par exemple pour trouver des points de contrôle qui déterminent une surface).

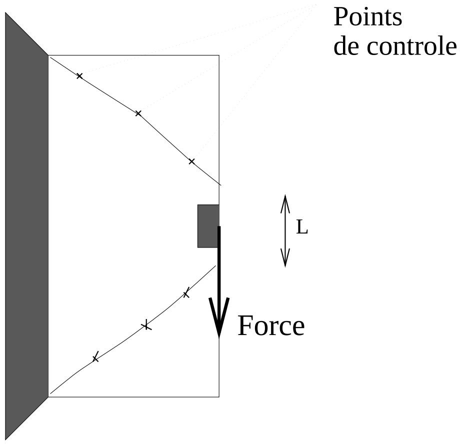


Figure 4 : *Optimisation géométrique*

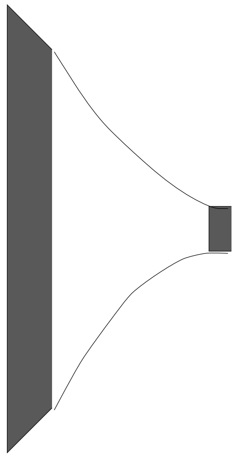


Figure 5 : *Solution possible correspondante*

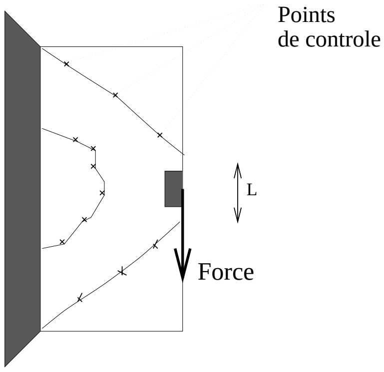


Figure 6 : *Exemple de forme de départ avec un trou*

Exemple

Reprenons la plaque console (figure 1) L est toujours variable mais la position des six points de contrôle aussi (figure 4).
L'optimisation de forme telle quelle, entendons par là l'optimisation géométrique, est la plus courante, la plus traitée. Elle est utilisée principalement dans l'industrie automobile ou aéronautique.
Il convient de remarquer la relation primordiale qui existe entre la représentation d'un contour et l'optimisation de forme. Il existe une multitude de façons de représenter un contour (lignes, arcs, splines ...). Chaque représentation dépend évidemment du problème considéré. Dans toutes les méthodes de résolution des problèmes d'optimisation, on note le découplage qu'il y a entre la représentation du contour et la méthode d'analyse proprement dite [13]. La principale méthode utilisée pour résoudre les problèmes d'optimisation géométrique est la méthode de variation de domaines.

Les méthodes de variation de domaine

- **Généralités**

Les méthodes classiquement utilisées en optimisation géométrique de formes sont les méthodes de variations de domaine [13] encore appelées "Analyse de sensibilité". Elles nécessitent le calcul de la dérivée de la fonction coût et des contraintes par rapport aux variables de contrôle (voir Cea [13], Pironneau [53]), qui sont utilisées dans des méthodes classiques de résolution de problèmes d'optimisation sous contraintes (ex. méthode d'Uzawa).
Leur principe est simple : il consiste à trouver des informations quantitatives sur la façon dont la réponse d'une structure est affectée par de petites modifications des variables qui définissent cette structure.
Ainsi, en partant d'une forme initiale donnée, on la modifie de façon continue jusqu'à l'obtention d'une forme respectant les objectifs précités.

- **Principe**

Le problème s'écrit de la manière suivante:

$$\begin{aligned} & min \quad J(\boldsymbol{u},\boldsymbol{h}) && (1.1.1)\\ \psi_i(\boldsymbol{u},\boldsymbol{h}) & = 0 \quad ; i=1,2,....,m, && (1.1.2)\\ \psi_i(\boldsymbol{u},\boldsymbol{h}) & <= 0 \quad ; i=m+1,m+2,....,k. && (1.1.3) \end{aligned}$$

avec :

- J fonctionnelle à minimiser ;
- $\boldsymbol{u} = \{u_1, u_2, ..., u_m\}$ vecteur composé de m variables d'état ;
- $\boldsymbol{h} = \{h_1, h_2, ..., h_n\}$ vecteur de n variables de contrôle ;

Les formes vérifiant les équations 1.1.3 et 1.1.3, c'est-à-dire celles qui respectent les contraintes, sont dites formes admissibles.

Le problème qui nous intéresse ici est l'étude de l'influence d'une variation des variables de contrôle sur les propriétés de la structure. On considère à la fois les deux types de variables, $\boldsymbol{u}$ et $\boldsymbol{h}$, et leurs variations respectives $\boldsymbol{u} + \delta\ \boldsymbol{u}$ et $\boldsymbol{h} + \delta\ \boldsymbol{h}$. En linéarisant les relations entre les variations de la fonction coût et les variations des contraintes, en fonction des vecteurs $\delta\ \boldsymbol{u}$ et $\delta\ \boldsymbol{h}$ on obtient :

$$\begin{aligned} \delta J &= (\partial J/\partial \boldsymbol{u})\delta\boldsymbol{u} + (\partial J/\partial \boldsymbol{h})\delta\boldsymbol{h} \\ \delta\psi_i &= (\partial\psi_i/\partial \boldsymbol{u})\delta\boldsymbol{u} + (\partial\psi_i/\partial \boldsymbol{h})\delta\boldsymbol{h}, \quad i=1,2,....,k. \end{aligned} \tag{1.1.4}$$

Les équations linéarisées correspondant à (1.1.3) sont de la forme :

$$A\delta\boldsymbol{u} \quad + \quad B\delta\boldsymbol{h} = 0; \tag{1.1.5}$$

Où:

$$A = \begin{bmatrix} \partial\psi_1/\partial u_1 & .... & \partial\psi_1/\partial u_n \\ ...., & .... & .... \\ ...., & .... & .... \\ \partial\psi_m/\partial u_1 & .... & \partial\psi_m/\partial u_n \\ \end{bmatrix}, \qquad B = \begin{bmatrix} \partial\psi_1/\partial h_1 & .... & \partial\psi_1/\partial h_n \\ ...., & .... & .... \\ ...., & .... & .... \\ \partial\psi_m/\partial h_1 & .... & \partial\psi_m/\partial h_n \\ \end{bmatrix}$$

Il y a deux manières de résoudre ce problème :

- **Première approche**
  Les matrices A, B et les quantités $\partial J/\partial \boldsymbol{u}$, $\partial J/\partial \boldsymbol{h}$, $\partial\psi_i/\partial \boldsymbol{u}$, $\partial\psi_i/\partial \boldsymbol{h}$ doivent être calculés avec les valeurs non perturbées des variables.
  Si A est non singulière la variation des variables d'état s'écrit :

$$\delta\boldsymbol{u} \;=\; -A^{-1}B\delta\boldsymbol{h}. \tag{1.1.6}$$

  En remplaçant la dernière relation dans les expressions (1.1.4), on obtient les variations δJ et $\delta\psi_i$, $(i = m+1, ..., k)$ en fonction de $\delta\boldsymbol{h}$.

$$\begin{array}{rcl} \delta J & = & [-(\partial J/\partial\boldsymbol{u})A^{-1}B + (\partial J/\partial\boldsymbol{h})]\delta\boldsymbol{h} \\ \delta\psi_i & = & [-(\partial\psi_i/\partial\boldsymbol{u})A^{-1}B + (\partial\psi_i/\partial\boldsymbol{h})]\delta\boldsymbol{h} \\ i & = & m+1, m+2, ..., k. \end{array} \tag{1.1.7}$$

  Ce qui donne la fonctionnelle J et les contraintes ψ en fonction des variables de contrôle.

- **Seconde approche**
  La méthode la plus généralement utilisée pour éliminer la dépendance des fonctionnelles par rapport à $\delta\boldsymbol{u}$ est légèrement différente.
  En remarquant que $\delta\boldsymbol{u}$ apparaît dans toutes les expressions à travers les termes $(\partial J/\partial\boldsymbol{u})\delta\boldsymbol{u}$ et $(\partial\psi_i/\partial\boldsymbol{u})\delta\boldsymbol{u}$, il semble naturel d'essayer de les éliminer. Pour cela on considère $k+1$ vecteurs λ^i tels que :

$$\begin{array}{rcl} A^T\lambda^0 & = & (\partial J/\partial\boldsymbol{u})^T, \\ A^T\lambda^i & = & (\partial\psi_i/\partial\boldsymbol{u})^T, \\ i & = & 1, 2, ..., k. \end{array} \tag{1.1.8}$$

  Les termes de l'équation (1.1.7) comportant les dérivées partielles par rapport à $\boldsymbol{u}$ sont calculées à partir des valeurs courantes des vecteurs $\boldsymbol{u}$ et $\boldsymbol{h}$, c'est-à-dire non perturbées.
  En considérant la transposée des deux membres de l'équation (1.1.8) et en multipliant par $\delta\boldsymbol{u}$ il vient:

$$\begin{array}{rcl} (\lambda^0)^T A\delta\boldsymbol{u} & = & (\partial J/\partial\boldsymbol{u})\delta\boldsymbol{u}, \\ (\lambda^i)^T A\delta\boldsymbol{u} & = & (\partial\psi_i/\partial\boldsymbol{u})\delta\boldsymbol{u}, \\ i & = & 1, 2, ..., k. \end{array} \tag{1.1.9}$$

  Grâce à 1.1.5 on obtient

$$\begin{array}{rcl} -(\lambda^0)^T B\delta\boldsymbol{h} & = & (\partial J/\partial\boldsymbol{u})\delta\boldsymbol{u}, \\ -(\lambda^i)^T B\delta\boldsymbol{h} & = & (\partial\psi_i/\partial\boldsymbol{u})\delta\boldsymbol{u} \\ i & = & 1, 2, ..., k. \end{array} \tag{1.1.10}$$

En remplaçant $(\partial J/\partial \boldsymbol{u})$, $\delta \boldsymbol{u}$ et $(\partial \psi_i/\partial \boldsymbol{u})\delta \boldsymbol{u}$ par leurs valeurs dans l'équation (1.1.4) on a :

$$\delta J = -(\lambda^0)^T B\delta \boldsymbol{h} + (\partial J/\partial \boldsymbol{h})\delta \boldsymbol{h} \tag{1.1.11}$$
$$\delta \psi_i = -(\lambda^i)^T B\delta \boldsymbol{h} + (\partial \psi_i/\partial \boldsymbol{h})\delta \boldsymbol{h} \tag{1.1.12}$$

Ce qui peut encore s'écrire sous la forme :

$$\delta J = [\nabla_h J]^T \delta \boldsymbol{h} \tag{1.1.13}$$
$$\delta \psi_i = [\nabla_h \psi_i]^T \delta \boldsymbol{h} \tag{1.1.14}$$

avec

$$[\nabla_h J] = (\partial J/\partial \boldsymbol{h})^T - B^T \lambda^0 \tag{1.1.15}$$
$$[\nabla_h \psi_i] = (\partial \psi_i/\partial \boldsymbol{h})^T - B^T \lambda^i \tag{1.1.16}$$

Les composantes des vecteurs $[\nabla_h J]$ et $[\nabla_h \psi_i]$ sont appelées *coefficients de sensibilité* des contraintes et de la fonction coût par rapport aux variables de contrôle.
Les coefficients de sensibilité sont très utiles car ils contiennent des informations sur la façon dont un changement sur la forme peut affecter la fonction coût et les contraintes. Pour plus de détails sur les méthodes de variations de domaine, se référer à (Sokolowski-Zolesio) [34] Murat-Simon [51] Pironneau [53].
Le calcul des valeurs de ces coefficients permet de voir quelles variables sont les plus importantes c'est-à-dire celles dont la modification a une grande influence sur les propriétés de la structure.

- **Discussion**

  Un inconvénient majeur de ces méthodes réside dans leur incapacité à modifier la topologie de la solution initiale, c'est-à-dire qu'on ne peut ni ajouter ni supprimer de trous. Ainsi, à partir de la plaque de la figure 4, il est possible de trouver la solution de la figure 5 mais pas la solution de la figure 7 qui nécessite de deviner la topologie en partant par exemple de la solution initiale décrite figure 6.

  Dans la partie suivante nous allons considérer les méthodes utilisées en optimisation topologique de formes.

1.1.3 Optimisation topologique

Les problèmes d'optimisation topologique, où l'ouvert solution n'est pas une déformée continue d'un l'ouvert de référence, sont parmi les plus difficiles en Optimisation de Formes. C'est le plus général des trois types d'optimisation de formes. Il concerne aussi bien les modifications de la topologie que de la forme ou des dimensions.
Toute forme incluse dans le domaine initial peut être a priori considérée comme solution

potentielle du problème de l'optimisation topologique de forme; en particulier, il peut y avoir apparition ou disparition de trous.
Exemple

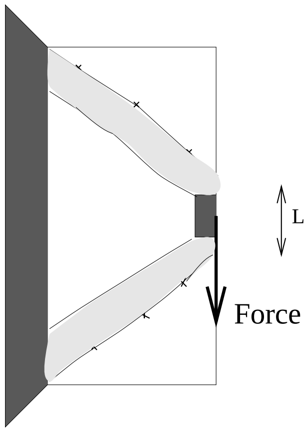


Figure 7 : *Exemple de solution en optimisation topologique*

Si l'on reprend le problème (figure 1), quelle que soit la solution initiale on doit arriver à la figure 7.

Il existe à notre connaissance deux familles de méthodes pour aborder ces problèmes : Les méthodes d'homogénéisation et les méthodes fondées sur des techniques d'optimisation stochastique. Nous présentons d'abord la première, la seconde étant l'objet de cette thèse.

Les méthodes d'homogénéisation en élasticité linéaire

- **Généralités**
  En optimisation topologique, où la topologie de la solution est inconnue, les méthodes d'homogénéisation sont les seules méthodes déterministes connues et utilisées. Ces méthodes ont été introduites en optimisation topologique par Bendsoe et Kikuchi(1988).
  Le problème se généralise en considérant non plus des variables booléennes matière/vide mais en utilisant une densité de matière sur un ouvert englobant la solution. A la fin de cette optimisation déterministe, la densité courante est forcée vers la valeur 1 ou 0 ce qui représentera la matière présente ou absente. Toutefois, cette approche nécessite la construction d'un opérateur homogénéisé décrit par Allaire et Kohn(1993)[4], et par Allaire, Bonnetier, Francfort, Jouve(1996) [2]. Nous en présentons brièvement les grandes lignes.

- **Principe**
  Soit Ω un domaine borné constitué d'un matériau élastique isotrope de loi de Hooke A.
  On suppose que sur une partie du contour $\partial\Omega$ de Ω est appliqué un chargement f

donné.
Les formes (ou topologies) dites *admissibles* sont obtenues en enlevant un sous-ensemble $H \in \Omega$ de trous dans Ω ; les trous étant en fait la limite dégénérée d'un matériau dont la loi de Hooke tend vers zéro.
La loi de Hooke d'un matériau isotrope est définie pour toute matrice symétrique ε par :

$$A\varepsilon = 2\mu\varepsilon + \lambda(tr\varepsilon)I \tag{1.1.17}$$

où:
I désigne la matrice identité (en dimension 2 ou 3), λ et μ les constantes de Lamé.
On a le système des équations d'élasticité

$$\left\{\begin{array}{rcll} \sigma & = & A\varepsilon(\vec{u}) & , \\ div\sigma & = & 0 & \text{dans } \Omega \setminus H \\ \sigma \cdot \vec{n} & = & \vec{f} & \text{sur } \partial\Omega, \end{array}\right. \tag{1.1.18}$$

avec

$$\varepsilon(\vec{u}) \quad = \quad 1/2(\nabla\vec{u} + \nabla\vec{u}^t) \tag{1.1.19}$$

$\vec{u}$ étant le déplacement de la structure, $\varepsilon(\vec{u})$ et σ sont des tenseurs symétriques respectivement de déformation et de contrainte.
La compliance est définie comme le travail des forces extérieures. Elle est donnée par :

$$\begin{aligned} c(\Omega \setminus H) &= \int_{\partial\Omega} f \cdot u \\ &= \int_{\Omega\setminus H} < Ae(\vec{u}), e(\vec{u}) > \\ &= \int_{\Omega\setminus H} < A^{-1}\sigma, \sigma > \end{aligned} \tag{1.1.20}$$

Le but des méthodes d'homogénéisation est de :
trouver H (ensemble des trous) qui minimise le poids avec une contrainte sur la compliance, ou la compliance avec une contrainte sur le poids.
En introduisant un paramètre de pénalisation $\lambda > 0$ on se ramène à un problème de minimisation sans contrainte :

$$Min_H[c(\Omega \setminus H) + \lambda \mid \Omega \setminus H \mid] \tag{1.1.21}$$

Sachant que :
$\mid \Omega \setminus H \mid$ est le poids de la structure (Ω privé des trous)

Allaire et Kohn [4] [10] montrent que le problème de minimisation (1.1.21) est équivalent au problème suivant :

$$Min_H \int_{\Omega} J(\tau)dx, \tag{1.1.22}$$

avec

$$J(\tau) = \begin{cases} 0 & \text{si } \tau = 0 \\ < A^{-1}\tau, \tau > +\lambda & \text{sinon} \end{cases} \tag{1.1.23}$$

parmi les contraintes admissibles τ vérifiant

$$\begin{cases} div\tau = 0 & \text{dans } \Omega \\ \tau \cdot \vec{n} = \vec{f} & \text{sur } \partial\Omega. \end{cases} \tag{1.1.24}$$

Ils introduisent ensuite la formulation relaxée de (1.1.22). Le problème de minimisation devient dans ce cas :

$$Min_{A^* \in G_\theta} \int_\Omega < A^{*-1}\tau, \tau > +\lambda\theta \tag{1.1.25}$$

pour τ admissible vérifiant 1.1.24.
Ou $\theta(x) \in [0, 1]$ est la densité de matière au point x et
G_θ est l'ensemble des lois de Hooke que l'on peut construire par homogénéisation en mélangeant du matériau A et du vide en proportion respective θ et $1 - \theta$.
Pour toutes les démonstrations se référer à [4].
Des résultats de convergence (Kohn [44], Allaire [4]) font de l'homogénéisation une méthode robuste et sûre.

Discussion

Il faut noter cependant que plusieurs difficultés liées à l'utilisation de ces méthodes subsistent :

- cette approche est limitée au cadre strict de l'élasticité linéaire ; l'utilisation de tout autre modèle mécanique doit passer par la définition de l'opérateur homogénéisé correspondant ;

- la solution homogénéisée doit être projetée sur l'espace des solutions réelles, c'est-à-dire, ne comportant que des zones de matière de densité 1 ou des zones de vide de densité 0 ; cette étape de pénalisation pose d'une part des difficultés techniques, d'autre part elle nous fait sortir du cadre théorique (il n'y a plus de résultat d'existence après les itérations pénalisées).

- il est impossible à l'heure actuelle de prendre en compte plus d'un unique chargement dans le processus d'optimisation. On ne peut donc pas calculer la forme optimale devant résister à plusieurs configurations données ;

- la fonction coût et les contraintes, doivent être différentiables ; on ne peut pas imposer une valeur limite à ne pas dépasser pour une grandeur physique liée au comportement de la structure recherchée (correspondant à une contrainte dans L^∞ non-différentiable) ; ainsi, pour respecter strictement une contrainte comme $D_{Max} < D_{Lim}$ (voir section 6.3.4), la méthode d'homogénéisation doit procéder par "tâtonnement" et proposer deux solutions qui encadrent ce déplacement limite ;

- il est impossible de traiter des problèmes pour lesquels des conditions aux limites sont imposées sur une frontière inconnue de la solution (par exemple une pression uniforme).

Pour remédier à ces difficultés, une possibilité est d'utiliser des méthodes d'optimisation stochastiques.

Ces méthodes travaillent sur des espaces quelconques, et sont capables d'optimiser des fonctionnelles non différentiables, voire discontinues.

De plus, n'ayant besoin que des valeurs de J (la fonctionnelle à minimiser), elles peuvent s'affranchir des limitations de l'homogénéisation. C'est l'objet de cette thèse de le démontrer.

Chapitre 2

Les méthodes d'optimisation stochastiques

2.1 Introduction

Nous présentons dans ce chapitre les méthodes d'optimisation stochastiques que nous utiliserons dans la suite pour les problèmes d'élasticité.
Cette présentation nous semble nécessaire du fait de la relative nouveauté des AGs dans le domaine de l'analyse numérique. Elle se veut générale et sans lien immédiat avec le problème de l'Optimisation de Formes.

2.2 Les algorithmes génétiques

2.2.1 Généralités

Introduits par Holland en 1975 [32], puis popularisés par, entre autres, Goldberg [22], Michalewicz [48], les Algorithmes génétiques (AGs) sont basés sur le paradigme naturel de l'évolution darwinienne des populations.
Dans la nature, les individus d'une population donnée sont souvent en compétition, que ce soit pour la survie (recherche de nourriture par exemple) ou pour le choix d'un compagnon. Seuls les mieux adaptés d'entre eux survivent et se reproduisent. En d'autres termes, les gènes des individus performants vont se retrouver dans de plus en plus d'individus au fil des générations. La combinaison des bons caractères des différents ancêtres peut conduire à des individus très adaptés, dont l'adaptation est encore plus grande que celle de chacun des parents. C'est une description simplifiée de la façon dont les espèces évoluent de manière à être de plus en plus adaptées à leur environnement.
Les Algorithmes génétiques utilisent une analogie directe, quoique caricaturale, de ce comportement naturel pour résoudre des problèmes d'optimisation. L'adaptation est alors mesurée en terme de valeurs de la fonction à optimiser. Cette analogie explique la spécifité du vocabulaire utilisé pour les algorithmes génétiques. Nous allons par conséquent

commencer par introduire quelques définitions, et les grandes lignes de l'AG de base.
Soit le problème modèle de maximisation d'une fonction :
$Max F(x)$, $x \in E$

On appelle *individu* un point de E c'est-à-dire une solution possible au problème donné. Très souvent (particulièrement dans le cadre original de Holland où $E = \{0,1\}^n$, un individu est également appelé *chromosome*, constitué de *gènes* (les bits 0/1). Les AGs utilisent une *population* d'individus représentant chacun une solution possible du problème donné, c'est donc un P-uplet $(X_1, ..., X_p) \in E^P$. A chaque individu on attribue une *performance* évaluant le mérite de cet individu en tant que solution possible du problème. Généralement, il s'agit directement de la valeur $F(x)$.

Les individus sont choisis, ou encore *sélectionnés* en fonction de leur performance : les individus les moins adaptés ont moins de chance d'être sélectionnés pour se reproduire et donc disparaissent naturellement.

Les meilleurs individus ont une plus grande probabilité de se reproduire par *croisement* avec un autre individu de la population. Cette opération donne naissance à des nouveaux individus dont les gènes sont issus de ceux de leurs parents.
En favorisant le croisement des meilleurs, les parties les plus intéressantes de l'espace de recherche sont ainsi explorées.

Chaque individu de la population a une certaine probabilité de *muter*. La *mutation* est une perturbation aléatoire d'un individu et favorise l'émergence d'individus possédant des caractéristiques génétiques nouvelles.

Toute une population de solutions possibles pour un problème est donc produite en choisissant les meilleurs individus par l'opérateur sélection, puis en appliquant les opérateurs croisement et mutation. Cette nouvelle population remplace les parents, et contient une large proportion des caractéristiques des bons individus de la *génération* précédente. Ce processus itéré sur plusieurs générations doit permettre de trouver les meilleurs caractéristiques de plus en plus souvent dans la population. Ces caractéristiques sont mélangées ou échangées avec d'autre bonnes caractéristiques. Si ce processus s'est bien déroulé, la population doit converger vers une solution optimale du problème.

Nous allons maintenant décrire les différentes étapes de l'algorithme génétique classique.

2.2.2 L'Algorithme génétique canonique

Notations et définitions

On cherche à maximiser une fonction positive réelle donnée f sur l'ensemble $E = \{0,1\}^n$ (cf Holland [32])
L'algorithme fait évoluer une population d'individus à travers plusieurs générations par des opérations que nous schématisons par la figure 8 et que nous détaillons ensuite.

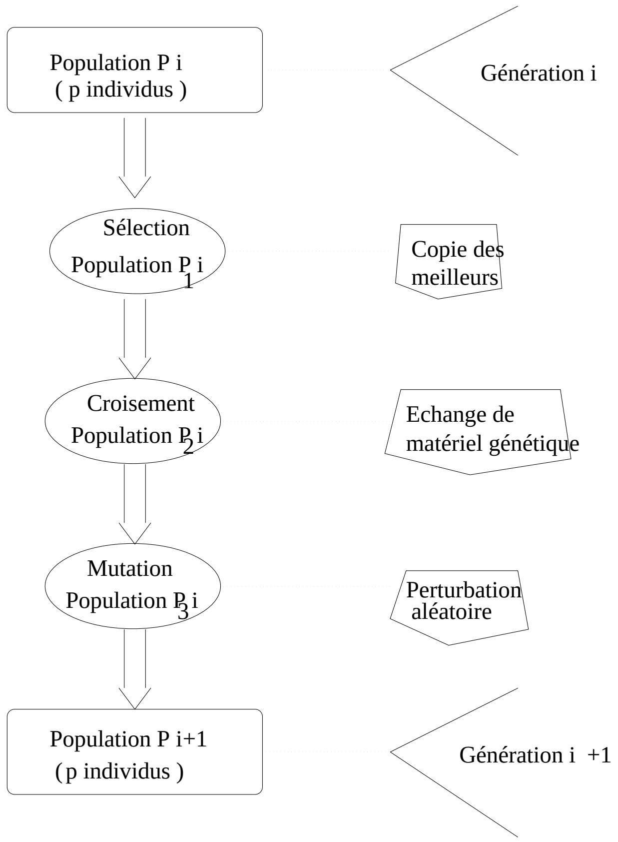


Figure 8 : *l'algorithme génétique de base*

2.2.3 Les opérateurs classiques

La reproduction joue un rôle fondamental pour passer de la population P_i (population de la génération G_i, $i \geq 1$) à la population P_{i+1} de la génération suivante. C'est lors de cette phase que l'on choisit les individus qui seront combinés deux à deux pour donner d'autres individus à la génération suivante .
Les parents sont choisis aléatoirement suivant une méthode qui favorisera les meilleurs d'entre eux. C'est la phase dite de *sélection*, où apparaît le principe Darwinien de l'évolution. Certains individus peuvent être choisis plusieurs fois tandis que d'autres ne seront jamais choisis. Il existe de nombreuses méthodes de sélection en fonction de la performance. Nous présentons plus en détail la sélection par roulette, utilisée dans les travaux originaux de Holland, et qui illustre bien les phénomènes mis en oeuvre.

La sélection par roulette

La sélection par roulette consiste à copier les individus proportionnellement à leur performance. Sachant qu'en fin de compte le nombre total de copies est égal à la taille de la population qui est fixe. Le comportement des AGs dépend énormément de cette phase

de sélection c'est-à-dire du choix des individus qui vont se croiser et se reproduire.
Les individus sont donc sélectionnés proportionnellement à leur performance comme le préconise Holland [32].
Soit $(x_1, ..., x_P)$ la population à la génération t. La performance de chaque individu est :

$$F(x_i), \quad i = 1, ... \quad , P. \tag{2.2.1}$$

Chaque individu x_i est copié en $Fils(x_i)$ fois, sachant que $Fils(x_i)$ est une variable aléatoire vérifiant

$$Esp(Fils(x_i)) \quad = \quad \frac{F(x_i)}{\bar{F}}. \tag{2.2.2}$$

Avec:

$$\bar{F} \quad = \quad \frac{\sum_{j=1}^{P} F(x_j)}{P} \tag{2.2.3}$$

La taille de la population restant constante, on a de plus :

$$\sum_{i=1}^{P} Fils(x_i) \quad = \quad P \tag{2.2.4}$$

L'espérance du nombre de fils d'un individu n'est en général pas un entier. Il faut ensuite convertir ce nombre en entier sans que le résultat soit trop biaisé. Plusieurs méthodes existent dont la meilleure est sans doute [8] *L'Echantillonage stochastique universel.*

Exemple

Pour fixer les idées supposons que l'on ait une population de 4 individus (x_1, x_2, x_3, x_4). Les performances des individus sont consignées dans le tableau 2.2.3 :

individu:	x_1	x_2	x_3	x_4
performance:	1.2	0.6	0.2	0.4

Table : 2.2.3

On évalue la performance de chaque individu par rapport à la performance moyenne de l'ensemble de la population.
La sélection par la roulette consiste à représenter les individus proportionnellement à leur performance [32], en lançant la boule 4 fois (taille de la population) de suite on obtient les enfants. On recopie un élément chaque fois que la boule s'arrête dans le secteur de la roulette qui lui correspond.

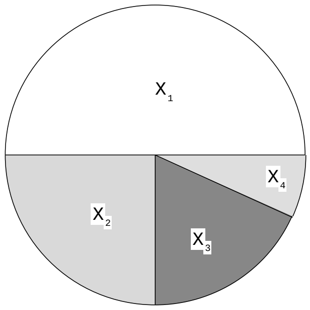


Figure 9 : *Sélection par la roulette*

Cette méthode de sélection est biaisée et ce biais et d'autant plus important que la taille de la population est petite.
On verra dans la suite d'autres méthodes de sélection plus performantes (section 2.2.8).

Une fois sélectionnés, les meilleurs individus ont une certaine probabilité de se reproduire par les opérations de croisement et de mutation.

Le croisement

L'opérateur de croisement transforme des couples d'individus, en recombinant certaines de leur parties. L'idée de base est de tenter d'améliorer ainsi la performance des enfants.

$$\left\{\begin{array}{lcl} I & = & xxxxxxxxxxx \\ J & = & yyyyyyyyyyy \end{array}\right. \overset{Pcrois}{\Longrightarrow} \left\{\begin{array}{lcl} I' & = & xxxxxyyyyyy \\ J' & = & yyyyyxxxxxx \end{array}\right.$$

(I, J) est le couple parents sélectionné et (I', J') le couple enfants obtenu. L'exemple présenté est un croisement *à un point* [22]. C'est l'un des croisements les plus utilisés. Un même point de coupure est choisi aléatoirement sur les deux chaînes de bits et on permute les deux chaînes à partir de ce point.
Il existe des croisements à 2, 3 ou n points, le principe reste le même. Pour le croisement à *2 points* par exemple, deux sites de coupures sont aléatoirement choisis et la permutation se fait ensuite.

$$\left\{\begin{array}{lcl} I & = & xxxxxxxxxxx \\ J & = & yyyyyyyyyyy \end{array}\right. \overset{Pcrois}{\Longrightarrow} \left\{\begin{array}{lcl} I' & = & xxxyyyyxxxx \\ J' & = & yyyxxxxyyyy \end{array}\right.$$

Dans le cas de problèmes discrets le croisement à découpage de chromosome précédemment décrit est parfaitement adapté. Dans tous les cas, on applique l'opérateur de croisement aux individus avec une certaine probabilité.
Si l'opération de croisement n'est pas appliquée, les enfants sont obtenus juste en copiant les parents.

La mutation

Elle est caractérisée par des perturbations génétiques aléatoires, destinées à diversifié les éléments de la population : en l'absence de toute mutation aucune caractéristique génétique nouvelle, c'est-à-dire qui n'était pas précédemment dans la population, ne peut apparaître. Tout comme le croisement, elle est appliquée avec une certaine probabilité.
Les individus sont ainsi aléatoirement déplacés dans E. L'idée est d'éviter les minima locaux.

$$\left\{ I \;=\; xxxxxxxxxxx \right. \overset{Pmut}{\Longrightarrow} \left\{ I' \;=\; xxxx\bar{x}xxxxxx \right.$$

Après toutes ces opérations nous obtenons la population P_{i+1} de la génération $i+1$.

2.2.4 Les paramètres

Même sur l'AG canonique, il existe un nombre important de paramètres à fixer : c'est le cas de la taille de la population $Taille$, la probabilité de croisement P_{cross}, la probabilité de mutation P_{Mut}, le critère d'arrêt (lorsque la solution n'est pas connue). Tous les opérateurs évolués que nous verrons (section 2.2.8) nécessiteront en contrepartie l'introduction de nouveaux paramètres. Pratiquement, les problèmes posés par la mise en place d'un bon algorithme génétique se trouve principalement dans l'ajustement de tous ces paramètres. La mise en place de ces paramètres critiques demande un temps de calcul non négligeable. De plus, la méthodologie nécessite des moyennes sur plusieurs essais indépendants pour valider l'approche stochastique.

Nous allons maintenant présenter des résultats heuristiques et théoriques expliquant le bon comportement des AGs.

2.2.5 Exploration et Exploitation

Tout algorithme d'optimisation doit comporter 2 phases pour trouver l'optimum global d'un problème.
Une phase *d'exploration*, qui permet d'explorer (échantillonner) de nouvelles régions de l'espace de recherche.
Une phase *d'exploitation*, qui utilise la connaissance que l'on a des parties de l'espace déjà visitées, et aide à trouver les meilleurs points dans les meilleures régions déjà explorées.
L'algorithme doit trouver un bon compromis entre ces deux phases.
La méthode de la recherche complètement aléatoire est très performante pour l'exploration mais ne fait pas d'exploitation du tout, tandis que la méthode du gradient est très bonne pour l'exploitation alors que la partie exploration est très peu développée. La combinaison de ces deux méthodes peut donner de très bons résultats, seulement il est difficile de trouver la juste mesure entre la part d'exploration et celle d'exploitation : quand peut-on juger qu'on a suffisamment exploité les résultats avant d'explorer à nouveau l'espace de

recherche ?
Holland [32] a montré que les algorithmes génétiques combinent ces deux phases. Si en théorie ceci est vérifié, en pratique des problèmes se posent. En effet, dans ses travaux, Holland suppose entre autre que :
La population est infinie, que la performance doit refléter l'utilité de la solution et que les gènes d'un chromosome ne doivent pas interagir de façon significative (problème dit de l'épistasie) c'est-à-dire que la fonction doit être séparable en terme de fonction de variables réelles.

La première des hypothèses ne peut pas être réalisée.
Une conséquence est que, même en l'absence de toute *pression de sélection* (c'est-à-dire pour une performance complètement constante), les individus de la population vont converger vers un point de l'espace solution. Ceci s'explique de façon très simple par l'accumulation d'erreurs stochastiques.
Si un gène prédomine par hasard dans une génération donnée, il est équiprobable qu'il devienne encore soit plus prédominant à la génération suivante soit moins influant. Si sa prédominance augmente au cours de plusieurs générations, étant donné que la population est finie, le gène peut se retrouver sur beaucoup d'individus et ce gène une fois fixe ne bouge plus si on ne considère que l'opération de croisement, qui ne peut pas changer les gènes. Au fil des générations les autres gènes se fixent de cette façon. Ce phénomène appelé *dérive génétique* est la principale source de convergence prématurée des AGs.
Les deux autres hypothèses peuvent quant à elles être vérifiées par certaines fonctions tests mais dans le cas réel elles sont rarement satisfaites.

2.2.6 Le théorème des schémas

La question qui se pose naturellement est de savoir comment les AGs obtiennent des résultats aussi intéressants étant donnée la simplicité des opérateurs décrits précédemment.

La plupart des recherches menées sur les algorithmes génétiques ont consisté jusqu'à présent à essayer de trouver des lois empiriques. Il n'y a pas vraiment de théorie générale expliquant pourquoi les AGs ont de telles propriétés. Toutefois on avance plusieurs hypothèses qui peuvent expliquer partiellement le succès des AGs et nous aider à implémenter un bon algorithme génétique.

Le *théorème des schémas* de Holland(75) est la première explication du bon comportement des algorithmes génétiques.
Il s'agit de chercher d'abord des similitudes entre les chaînes dans la population, ensuite de trouver des relations de cause à effet entre ces similitudes et les performances élevées, ce qui fournit de nouvelles informations permettant de mieux guider la partie *exploration* de l'algorithme.
Considérons une chaîne de caractère A de longueur l : $A = a_1a_2a_3....a_l$ avec $a_i \in \{0, 1\}$

Définition 2.2.1 ***(Schéma)***

Un schéma H *est défini comme un sous-ensemble de chaînes qui ont en commun des caractéristiques génétiques.*
Il peut être représenté en codage binaire par une chaîne de caractère de {0,1,#}. Le caractère # représente les caractéristiques qui ne sont pas communes.

Définition 2.2.2 ***(Instance)***
On dit qu'une chaîne $A = a_1a_2.....a_l$ *est une* instance *d'un schéma* $H = b_1b_2\#.....b_l$ *si :* $\forall i \in \{0, ..., l\}$ *tel que* $b_i \neq \#$ *on a* $a_i = b_i$.

On dit que le chromosome A (chaîne) est contenu dans H.

Définition 2.2.3 ***(Ordre d'un schéma)***
L'ordre *d'un schéma est le nombre de caractères fixes c'est-à-dire les caractères autres que # qu'il contient. Il est noté* $o(H)$.

Définition 2.2.4 ***(Longueur utile d'un schéma)***
La longueur utile *d'un schéma, noté* $l(H)$, *est la distance entre les deux caractères les plus externes autre que #.*

Exemple 2.2.5
$H = 1\#1\#0$ *est un schéma qui représente toutes les chaînes de caractères de longueur 5 avec 1 en première et troisième position et 0 en dernière. La longueur utile du schéma ici* $l(H) = 5$ *et l'ordre* $o(H) = 3$.
Comme exemple d'instance on a :
$A = 11110$, $B = 10110$,.....

Nous allons considérer maintenant les effets des différents opérateurs de l'algorithme sur les schémas.

La reproduction

L'opération de reproduction influe de manière assez simple sur les schémas.
Soit $A_i = a_1a_2a_3a_4a_5$ une chaîne de caractère, $a_i \in \{0, 1\}$. On se donne un ensemble de chaînes $A = (A_j)_{j\in[1,n]}$ qui forment la population $A(t)$ à la génération t, et un schéma H défini sur l'alphabet {0, 1, # }, # étant le symbole indifférent.
On suppose qu'à la génération t il y a $m = m(H, t)$ exemplaires de H contenu dans $A(t)$.
L'opérateur de reproduction copie les chaînes proportionnellement à leur performance.
Une chaîne A_i est sélectionnée avec une probabilité $p_i = f_i/\bar{f}_s$.
Ou f_i est la valeur de la performance pour la ième chaîne de caractère A_i . $\bar{f}_s = \frac{1}{n}\sum_{i=1}^{n} f_i$ est la valeur moyenne de la performance pour les individus de toute la population $A(t)$.
Pour une population de taille n, on a l'équation d'adaptabilité (ou d'évolution) :

$$Esp(m(H, t + 1)) = \frac{f(H)}{\bar{f}_s} m(H, t) \qquad (2.2.5)$$

qui donne l'espérance du nombre de représentants du schéma H dans la population $t + 1$, $f(H)$ étant la performance moyenne des chaînes représentant le schéma H (instances de

H) à la date t c'est-à-dire la valeur moyenne de la fonction performance pour les individus du schéma H. Ainsi, tous les schémas d'une population se développent ou dépérissent en fonction de leur moyenne de schéma $f(H)$ du fait seul de l'opérateur de reproduction. Si un schéma H reste au dessus de la moyenne d'une valeur $c\bar{f}_s$ (c étant une constante), l'équation d'évolution se réécrit :

$$Esp(m(H,t+1)) = (1+c)m(H,t); \tag{2.2.6}$$

L'influence du croisement

L'opération de reproduction n'influe pas sur l'exploration de nouvelles régions de l'espace de recherche. Aucun nouveau point n'est exploré et on ne fait que copier les individus. C'est ici qu'apparaît le rôle du croisement qui permet un échange d'information entre les chaînes.
On a vu précédemment que le croisement est un processus aléatoire de probabilité p_c ; la décision pour l'exécuter est prise par un générateur de nombres aléatoires uniformément distribués entre 0 et 1 ; si le nombre généré est inférieur à p_c le croisement a lieu, et dans ce cas on détermine l'endroit sur la chaîne de longueur l où l'opération a lieu.
Pour le théorème des schémas l'influence du croisement est considérée uniquement sous l'angle négatif : vu la capacité du croisement à détruire les schémas, on va compter ceux qui sont détruits et borner ce nombre. Voyons comment l'opération de croisement peut détruire un schéma particulier H, en partant d'un exemple concret.

Exemple 2.2.6

On considère deux chaînes A_1 et A_2 en codage binaire de longueur $l = 10$, on génère un point de coupure $k = 4$;

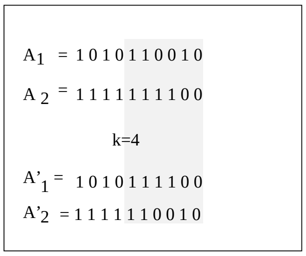


Figure 10 : Exemple de croisement

Considérons la chaîne A_1 ci dessus et deux schémas H et H' qui la représentent

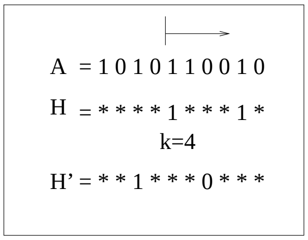


Figure 11 : Influence du croisement sur les schémas

Dans ce cas de figure le schéma H' pourra être détruit car des *positions fixes* sont situées de part et d'autre du point de coupure. Par contre, le schéma H survivra obligatoirement car ses deux alèles (positions) fixes se trouvent sur le même côté par rapport à la position de croisement, mais il ne sera présent que dans l'une des deux chaînes enfants.
On peut donc quantifier l'effet du croisement sur la vie d'un schéma particulier.
La probabilité P_d pour qu'un schéma soit détruit est donnée par la probabilité d'avoir la position de croisement entre la première et la dernière position fixe c'est-à-dire entre 1 et 0 pour H'.

$$p_d \quad = \quad l(H)/(l-1) \tag{2.2.7}$$

Si l'opération est faite avec la probabilité de croisement p_c, on obtient la relation suivante :

$$p_d \quad \leq \quad p_c l(H)/(l-1). \tag{2.2.8}$$

Si le croisement est fait entre deux chaînes qui représentent le même schéma il ne sera jamais détruit. Une borne inférieure sur la probabilité de survie p_s du schéma s'en déduit de manière immédiate :

$$p_s \quad \geq \quad 1 - p_c \left[\frac{l(H)}{(l-1)} \right] \tag{2.2.9}$$

Comme la reproduction et le croisement sont des opérations indépendantes, on peut estimer le nombre de représentants du schéma H dans la génération $t+1$ par rapport à la génération t comme le produit du nombre espéré par la seule reproduction avec la probabilité de survie p_s après 1 croisement. En tenant compte de la reproduction et du croisement on obtient après t générations :

$$m(H,t) \quad \geq \quad (1+c)^t \left\{1 - p_c \left[\frac{l(H)}{(l-1)}\right]\right\}^t m(H,0). \tag{2.2.10}$$

Le rôle de la mutation

De la même manière, on considère maintenant l'opérateur de mutation qui est un changement aléatoire de certains éléments de la chaîne. Cet opérateur est appliqué avec une probabilité p_m. En introduisant des éléments nouveaux, la mutation prévient la perte de caractéristiques génétiques. En fait, tout élément de la chaîne a une probabilité de survie de $(1-p_m)$ et comme les mutations sont indépendantes, la probabilité de survie du schéma est : $(1-p_m)^{o(H)}$ si $o(H)$ est l'ordre du schéma.

$$m(H,t+1) \geq m(H,t)(1+c)\left\{1 - p_c \left[\frac{l(H)}{(l-1)}\right]\right\}(1-p_m)^{o(H)} \tag{2.2.11}$$

Théorème 2.2.7 ***("des schémas")***

La probabilité de survie d'un schéma après les opérations de reproduction, croisement et mutation s'obtient par:

$$m(H,t+1) \geq m(H,t)(1+c)\left\{1 - p_c \left[\frac{l(H)}{(l-1)}\right] - o(H)p_m\right\}. \tag{2.2.12}$$

Ainsi, un schéma Hde faible *longueur définie*, dont la performance moyenne est supérieure à la moyenne de la population, verra son nombre augmenter d'une génération à l'autre.
La théorie des schémas explique le succès ou l'échec des AGs suivant la manière dont les instances sont traitées. On remarque que ce "théorème" ne considère les opérateurs que pour leurs actions destructrices de schémas. Il faut une étude plus fine pour tenir compte des effets créateurs de nouveaux schémas (voir section 5.4)
Pour une population de taille n, Holland a montré qu'un algorithme génétique traite un nombre de schémas de l'ordre de n^3. C'est la théorie du *parallélisme implicite*. C'est-à-dire que l'évaluation de la performance d'un chromosome est également l'évaluation de tous les schémas auxquels il appartient.

Les individus d'une population ont une certaine probabilité de se reproduire et d'avoir des enfants. Le nombre de telles opportunités pour un individu donné est fonction de sa performance. Les meilleurs individus contribuent donc plus de leurs gènes pour la génération suivante. La bonne performance d'un individu est due au fait qu'il possède de bonnes instances. En transmettant beaucoup de ses instances à la génération suivante, on augmente la possibilité de trouver de meilleures solutions.

Holland [32] a montré que la meilleure façon d'explorer l'espace de recherche est d'allouer des essais de reproduction aux individus proportionnellement à leur performance par rapport au reste de la population. De cette manière les bons schémas reçoivent un nombre d'essais croissant exponentiellement en fonction des générations.

La principale critique formulée contre ce "théorème" est de ne pas prendre en compte la performance moyenne du schéma effectivement observée dans la population (voir Radcliffe [55]).

l'hypothèse de la construction des blocs

Selon Goldberg la puissance des AGs réside dans leur capacité à trouver de bons blocs constructifs (encore appelés *briques élémentaires*), et de les recombiner.

Définition 2.2.8 ***(briques élémentaires)***

> *Ce sont en fait des schémas de petite longueur définie qui consistent en des bits qui mis ensemble conduisent à une bonne performance.*

Ces schémas courts d'ordre faible et très bien adaptés conduisent à une meilleure performance quand on les met dans un individu.
Les *briques élémentaires* sont sélectionnées, recombinées et reselectionnées pour former des chaînes encore plus adaptées.

2.2.7 Modélisation par chaînes de Markov

Soit P_i la population à l'instant i, et P la taille de la population, $X_i = ({X_i}^1, ..., {X_i}^P)$ est un vecteur dont les P composantes sont des chaînes de longueur N sur {0,1}. On a vu que pour passer de X_i à X_{i+1} on suit le schéma :

$$X_i \overset{\text{Séléction}}{\longrightarrow} Y_i \overset{\text{Croisement}}{\longrightarrow} Z_i \overset{\text{Mutation}}{\longrightarrow} X_{i+1}$$

La suite X_i est une chaîne de Markov d'espace d'état $(\{0,1\}^N)^P$, la loi de X_i est déterminée de manière unique par la donnée de la loi de X_0 et du mécanisme de transition précédemment décrit.
Les principaux résultats théoriques de convergence sont :

- Un résultat très général de convergence globale, au sens de la convergence faible des mesures de probabilités (Zhigljavsky [68])

- Des résultats partiels basés sur les algorithmes génétiques en tant que chaînes de Markov (Davis [19] et Vose [66])

- Un résultat de convergence global basé sur la théorie de Freidlin-Wentzell des perturbations stochastiques de systèmes dynamiques (Cerf [14])

  On part d'un *système non perturbé* : c'est-à-dire d'un AG sans mutation ni croisement, à sélection *caricaturale.*
  On perturbe ensuite ce système sachant que cette *perturbation* consiste en : une sélection proportionnelle, une mutation et un croisement à un point. Sous des hypothèse très techniques concernant la manière dont ces perturbations convergent

vers 0 de manière liée, Cerf [14] montre qu'il existe une taille de population P^* appelée *taille critique* (calculable à partir de F et des définitions des perturbations) telle que, si $P > P^*$,

$$\forall x \in E^P \lim_{i \to \infty} P(P_i \subset \mathcal{F}^* / P_0 = x) = 1$$

F est la fonction performance et $\mathcal{F}^*$ est l'ensemble des maxima globaux de F.

Pour d'autres détails sur les principaux modèles théoriques des AGs voir les travaux de Vose [66] et plus récemment de G. Rudolph [57].

2.2.8 Les opérateurs de sélection évolués

Pourquoi de nouveaux opérateurs ?

Deux phénomènes motivent l'introduction de méthodes de sélection plus évolués : la *convergence prématurée* et la *convergence trop lente.*

Un problème courant pour les AGs est que les gènes d'individus relativement performants, mais non optimaux, dominent rapidement dans la population provoquant ainsi une convergence prématurée sur un maximum local.
C'est souvent le cas dans les premières générations où peu d'individus sont performants, l'écart type des performances des individus est très élevé. L'espérance du nombre de fils souhaités pour le meilleur individu, encore appelée *pression sélective*, est très importante. Les meilleurs individus peu nombreux se reproduisent beaucoup. Il est donc important de pouvoir contrôler la pression sélective.

A l'inverse, la convergence peut être trop lente (stagnation) : Après de nombreuses générations la population a en grande partie convergé mais peut ne pas avoir localisé précisément le maximum global (cas d'un très faible gradient autour du maximum). La moyenne de la fonction performance est dans ce cas relativement élevée. Ainsi il peut n'y avoir qu'une assez faible différence entre le meilleur individu et les individus moyens.
La pression sélective est très faible dans ce cas et les individus se reproduisent tous de manière à peu près identique. Il est alors très difficile d'atteindre un maximum global.
Une fois que la population a convergé, la capacité des AGs à trouver de meilleures solutions est presque nulle. En effet, le croisement d'individus presque identiques ne produit pas grand chose de neuf. Seul l'opérateur de mutation reste pour explorer de nouvelles parties de l'espace. Ce qui devient en fait une simple recherche aléatoire.

Les mêmes techniques sont utilisées pour remédier à ces deux types de problèmes. Le *théorème des schémas* stipule que l'on doit accorder des essais de reproduction à un individu proportionnellement à sa fonction mérite. Mais ce faisant on a une convergence prématurée de l'algorithme car la population n'est pas infinie. Pour rendre optimal l'algorithme génétique, on doit modifier la méthode de sélection des parents.
La manière de modifier le processus de sélection doit contrôler le nombre de tentatives de reproduction de chaque individu pour qu'il ne soit ni trop élevé ni trop faible. Il faut donc comprimer la marge de variation de la fonction mérite de manière à ne pas avoir brusquement un individu trop performant. Nous présentons les différentes techniques de mise à l'échelle utilisées, la sélection par le rang et les tournois.

la pression sélective

Définition 2.2.9 ***(Pression sélective)***

> *On appelle* pression de sélection ρ *l'espérance du nombre de copies souhaitées pour le meilleur individu.*

L'individu le plus performant aura donc (en espérance) ρ descendants. On cherche généralement une pression sélective fixée car, si ρ est trop faible, il n'y a pas de sélection du tout. Si par contre ρ est trop forte il y a une convergence prématurée vers probablement un maximum local.

La mise à l'échelle linéaire

Les algorithmes génétiques travaillent en terme de maximisation. Or, maximiser la fonction f ou $\alpha f + \beta$ est théoriquement équivalent (pour $\alpha > 0$) mais très différent pour les AGs ; f est donc remplacée par la fonction performance transformée $f^{'} = \alpha f + \beta$ dans la phase de sélection pour une génération donnée.
Soit :

$$\bar{f} \quad = \quad \frac{\sum_{I \in P} f(I)}{n} \tag{2.2.13}$$

et

$$f_{max} = max f(I); I \in P \tag{2.2.14}$$

$\bar{f}$ est la moyenne d'adaptation brute et $\bar{f}'$ est la moyenne d'adaptation transformée.
α et β sont choisis de telle sorte que $\bar{f} = \bar{f}'$, ce qui assure que chaque individu d'adaptation moyenne a un nombre de copie attendu égal à un et que $f'_{max} = \rho\bar{f}$, cette dernière relation permet de contrôler le nombre de descendants de l'individu ayant l'adaptation brute maximale. Ces deux relations déterminent α et β en fonction de ρ, f_max et $\bar{f}$. ρ est généralement choisi entre 1.0 et 2.0 .

Exemple 2.2.10

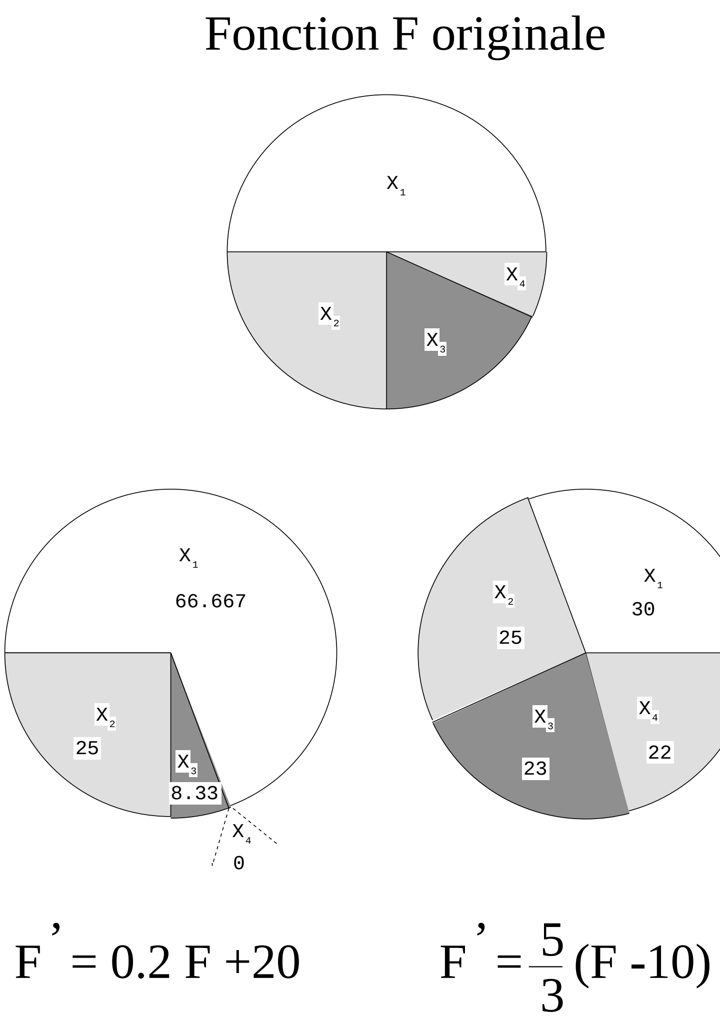


Figure 12 : Echelle et Pression sélective

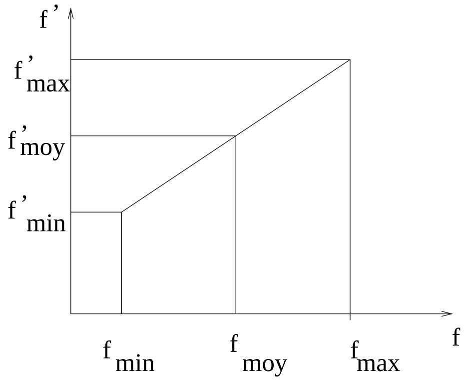


Figure 13 : *mise à l'échelle linéaire*

La mise à l'échelle est la technique la plus répandue pour éviter une trop rapide convergence ou une uniformisation de la population. On peut l'utiliser sous deux formes :

On fixe le nombre maximum de reproduction pour tous les individus à 2.0 par exemple (un individu ne peut pas avoir plus de deux copies dans ce cas).
Ce qui s'obtient en soustrayant une constante à la performance brute puis en divisant par la moyenne des performances ajustées.
Soustraire une valeur augmente le rapport entre la performance maximale et la moyenne des performances. On doit de plus faire attention à ce que la fonction performance ne prenne pas de valeurs négatives.
La mise à l'échelle tend à réduire les variations de la performance en début d'exécution ralentissant ainsi la convergence tout en augmentant la quantité d'exploration. Cependant, la présence d'un seul individu extrêmement performant, avec par exemple une performance 10 fois plus grande que la performance de n'importe quel autre individu, peut conduire à une surcompression de la fonction mérite. Si par exemple l'échelle des performances est réduite de façon à ce que le rapport maximum des performances sur la moyenne des performances est de l'ordre de 2:1. Dans ce cas tous les autres membres de la population auront leur performance regroupée autour de 1. Bien que l'on ait évité une convergence prématurée, les variations de la fonction performance sont tellement réduites que l'on risque de s'éloigner du meilleur.

Il existe une autre méthode qui est une variante de la précédente, la seule différence est sur le terme à soustraire.
Dans cette méthode, on considère la performance minimale au cours de chaque génération, la valeur effectivement soustraite est la valeur minimale observée pendant les n générations précédentes (n fixé).
Avec cette méthode la pression sélective, c'est-à-dire le rapport entre le nombre maximum d'essais de reproduction autorisés et le nombre moyen, varie au cours de l'exécution et varie également d'un problème à l'autre. La présence d'un individu trop mauvais peut provoquer une sous-expansion tandis que quelques individus trop performants peuvent toujours provoquer une convergence prématurée puisqu'ils n'influencent pas le taux d'échelle appliqué.

Le problème de ces deux types de méthodes est que l'intensité de la compression est dictée par un seul individu le meilleur ou le pire. Elles seront donc peu efficace si la performance de cet individu est trop élooignée de la performance moyenne. .

La mise à l'échelle exponentielle

La performance transformée est donnée par:

$$f' = f^{k(n)} \tag{2.2.15}$$

k est fonction des générations et n représente la génération courante. [22]

$$k(n) = \left(\frac{s^*}{s_0}\right)^{p_1} \left[\tan\left(\frac{n}{N+1}\frac{\pi}{2}\right)\right]^{\left(p_2\left(\frac{s_0}{s^*}\right)^{\alpha}\right)} \tag{2.2.16}$$

sachant que :
s^*, s^0, p_1, p_2 sont des paramètres à ajuster dans $]0,1[$. (voir Goldberg [22]).
Discussion :

- Si $k \simeq 1$ la transformation est inopérante.

- Si $k \simeq 0$ il y a réduction des écarts de la performance, aucun individu n'est vraiment favorisé et l'algorithme génétique se comporte comme un simple algorithme de recherche aléatoire, qui correspond à une exploration de l'espace.

- Si $k > 1$ les écarts sont exagérés et seuls les bons individus sont sélectionnés.

k varie donc des faibles valeurs vers de fortes valeurs au fur et à mesure que le nombre de générations augmente (voir Goldberg[22]).

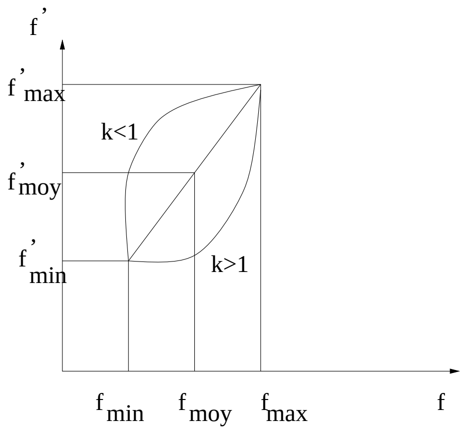


Figure 14 : *mise à l'échelle exponentielle*

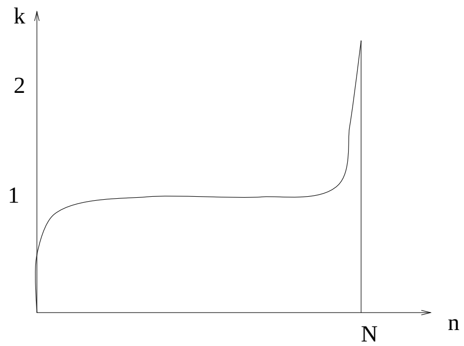


Figure 15 : *évolution de k en fonction des générations*

Sélection par le rang

Une autre méthode également très utilisée est la sélection par le rang, méthode qui surmonte le problème de la trop grande influence d'un seul individu.
Dans ce cas de figure la performance n'est pas modifiée mais les individus sont classés selon leur performance de départ. La probabilité pour qu'un individu se reproduise dépend du rang de l'individu. Cela peut être fait linéairement [7] ou exponentiellement [18].
Ceci donne un résultat analogue à celui de la mise à l'échelle puisque le rapport (de la performance maximum sur la moyenne) est normalisé à une valeur fixée indépendamment des valeurs de la fonction performance.
Toutefois, cette méthode assure aussi que les nouvelles performances d'individus moyens s'étalent régulièrement. Dans ce cas l'influence d'un ou de deux individus extrêmes sera négligeable. En effet, la méthode ne tient pas compte de l'amplitude de l'écart entre leur performance et celles des autres éléments de la population. Le nombre d'essais de reproduction alloué aux k meilleurs individus sera toujours le même, quelle que soit la performance de départ de ceux qui sont meilleurs ou moins bons.

Dans ce cas la surcompression de la performance du meilleur individu n'est plus un problème.

$$Prob(rang) = q(1-q)^{rang-1} \tag{2.2.17}$$

représente la probabilité pour qu'un certain rang soit tiré, avec $q \in]0,1[$.

Exemple 2.2.11

Reprenons l'exemple introduit pour illustrer la sélection avec la roulette puis repris dans la mise à l'échelle, 2.2.3. La taille de la population est de 4,

individu	*performance brute*	*nouveau rang*	*probabilité*
1	*2.5*	*2*	*0.158*
2	*1.8*	*3*	*0.144*
3	*5*	*1*	*0.4*
4	*0.7*	*4*	*0.0864*

Certains résultats [18] [7] semblent prouver que la sélection par le rang mène à de meilleures solutions que les techniques de mise à l'échelle citées précédemment.
L'inconvénient majeur du classement est que le comportement est le même pour les premières ou les dernières générations.
En outre il requiert un tri des individus qui peut être très coûteux en temps calcul dans le cas de grandes populations.

Sélection par tournoi

Cette méthode choisit les individus susceptibles de se reproduire sans recalculer explicitement une fonction performance modifiée.
Soit $n \in IN$ la taille du tournoi. Pour sélectionner un individu on choisit uniformément

dans la population n individus, et on sélectionne le meilleur des n. Le tournoi de taille 2 et la sélection par le rang ont la même espérance.

Utiliser de grands tournois a pour effet d'augmenter la pression sélective car les individus inférieurs à la moyenne ont peu de chances de gagner un tournoi alors que ceux supérieurs à cette moyenne ont de grandes chances de gagner.
Ce processus peut être généralisé pour des pressions sélectives plus petites au *tournoi probabiliste*. Dans ce cas de figure, le meilleur individu gagne le tournoi avec une probabilité $p \in]0.5, 1[$. Utiliser de plus petites valeurs pour p a pour effet de baisser la pression de sélection, les individus inférieurs à la moyenne auraient alors plus de chance de remporter le tournoi face à d'autres qui sont supérieurs à cette moyenne.
En ajustant la taille du tournoi ou la probabilité de gagner, la pression de sélection peut être rendue arbitrairement grande ou petite.

Des comparaisons ont été menées dans [26] entre ces différentes méthodes, la sélection proportionnelle, le classement de la performance, les tournois, et bien d'autres méthodes encore. De Jong a montré dans le cas du tournoi binaire que l'espérance est la même que pour la sélection par le rang, mais que la variance est plus importante [20]. On peut donc en conclure, pourvu que les paramètres soient bien choisis, que toutes ces méthodes, mis à part la sélection proportionelle brute, qui est vraiment mauvaise, sont à peu près équivalentes en ce sens que tous les résultats obtenus grâce à elle sont bons. Il n'y a donc pas une meilleure méthode à proprement parler.
Une tendance récente favorise cependant le tournoi, pour sa facilité d'implémentation : il n'y a pas de calcul de coefficients, ni de tri, mais seulement des comparaisons.

2.2.9 Le Partage

Le partage (*sharing*) est un autre opérateur évolué des AGs.
Pourquoi?
Il est utilisé pour éviter le regroupement d'individus performants, et assurer une certaine diversité dans la population et empêcher ainsi une convergence prématurée. Le partage vise à trouver tous les maxima. Cette technique est illustrée dans l'exemple de Goldberg (voir Figure 16).

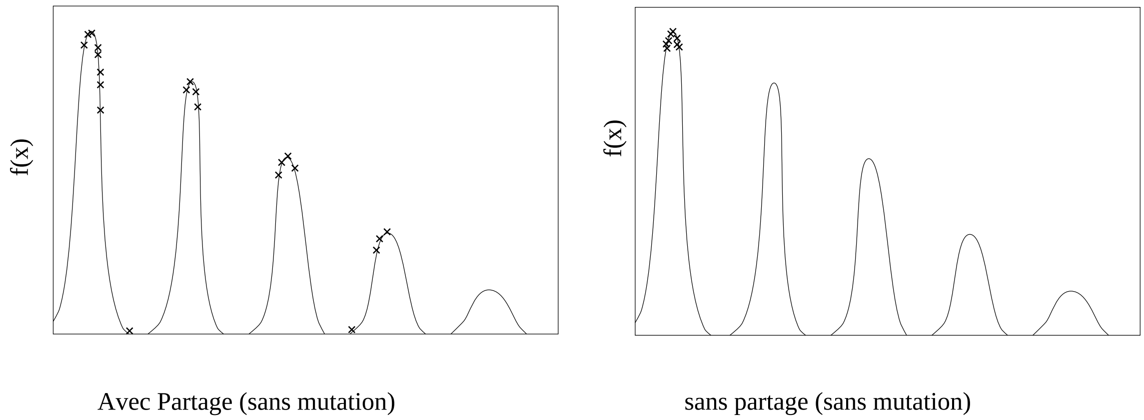


Figure 16 : *Exemple de fonction à pics décroissants avec ou sans partage*

Principe

Le partage évite l'agrégation des individus dans une région donnée en pénalisant les individus qui ont trop de voisins proches. La notion de *voisinage* entre deux individus I et $I^{'}$ est définie par :

$$\Pi(I, I^{'}) = \left(1 - \left(\frac{d(I, I^{'})}{\delta}\right)^{\alpha}\right)^{+} \qquad (2.2.18)$$

où :
$d(I, I^{'})$ est la *distance* entre deux individus. Cette distance mesure la dissimilarité entre les individus I et $I^{'}$. Une distance fréquemment utilisée est la *distance de Hamming* qui mesure la distance entre deux chaînes de caractères en comptant le nombre de bits différents.

δ est un paramètre dépendant du problème traité, il est fixé par l'utilisateur. Il permet de délimiter le voisinage.

α permet d'ajuster l'intensité de la pénalisation que l'on applique à la performance d'un individu.

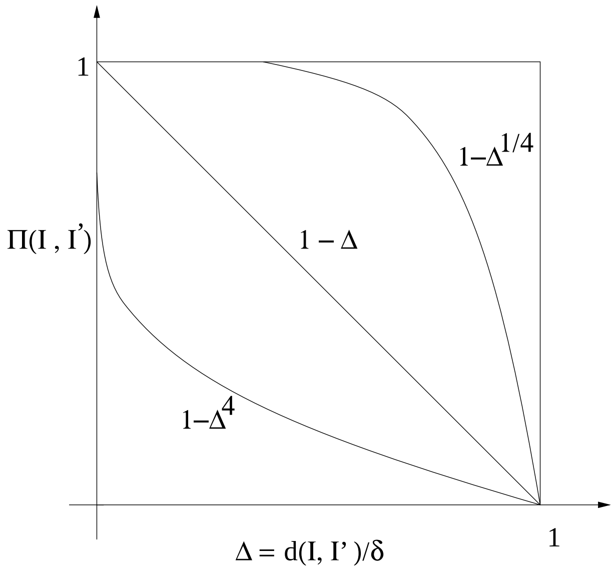


Figure 17 : *Allure de la fonction partage*

Pour effectuer le partage, la performance $F(I)$ de I est remplacée par :

$$F^{'}(I) \quad = \quad \frac{F(I)}{\sum_{I' \in P} \Pi(I, I')} \tag{2.2.19}$$

La nouvelle performance est inversement proportionnelle au nombre de voisins d'un individu.

Exemple 2.2.12

On considère le cas de trois individus et pour α fixé égal à 1.

Figure 18 : Exemple de partage

On a :

$$\begin{array}{rll} d(1,2) > \delta & \longrightarrow \Pi(1,2) & = \Pi(2,3) = 0 \\ d(1,3) << \delta & \longrightarrow \Pi(1,3) & \simeq 1 \end{array} \tag{2.2.20}$$

Dans ce cas la performance modifiée devient :

$$\left\{ \begin{array}{rcl} F^{'}(1) & = & \dfrac{F(1)}{2} \\ F^{'}(2) & = & F(2) \\ F^{'}(3) & = & \dfrac{F(3)}{2} \end{array} \right. \tag{2.2.21}$$

2.2.10 Le codage

Définition, problématique

Dans la théorie originale de Holland, encore prônée par un petit nombre d'irréductibles, il suffisait de ramener un problème d'optimisation dans $\{0,1\}^n$, puis de lancer la machine génétique pour obtenir une solution au problème initial. L'opération consistant à se ramener dans l'espace des chaînes de bits $\{0,1\}^n$ était appelée *codage*.

Par extension, on considère aujourd'hui deux espaces : l'espace où est posé le problème, et sur lequel il est possible de calculer les valeurs de la fonction à optimiser F, ou espace *phénotypique* et l'espace où travaille les opérateurs génétiques (croisement, mutation) ou espace *génotypique*. Le codage est le passage du phénotype au génotype (le décodage étant l'opération inverse).

Importance du codage

Un bon codage est un codage qui favorise la formation des briques élémentaires qui seront recombinées par l'opérateur de croisement.

Deux questions importantes, et qui restent ouvertes, se posent naturellement :

- Est-il possible en général de trouver des codages qui vérifient les hypothèses de la construction de briques élémentaires (voir section : 2.2.10) ?

- Dans le cas où il ne serait pas possible de trouver de tels codages, peut-on modifier l'algorithme génétique de façon à améliorer sa performance, et dans ce cas, comment?

Le codage binaire universel

Malgré toutes les critiques qui lui ont été adressées, le codage le plus utilisé reste le codage binaire. Nous allons en décrire un exemple, dans la mesure où il est représentatif d'un état d'esprit maintenant dépassé, et qui peut sembler surprenant aux spécialistes de la manipulation des nombres réels que sont les mathématiciens appliqués.
La définition d'un codage binaire de longueur n fixée pour tous les réels x d'un intervalle $[a, b]$ se déroule comme suit. Pour coder $x \in [a, b]$, on calcule

$$x_{bin} = \left[\frac{x-a}{b-a}2^n\right]$$

x_{bin} est un entier de $[0, 2^n - 1]$. Son expression en base 2 est un codage binaire obtenu dans $\{0, 1\}^n$ L'entier n détermine la précision du codage.
Dans le cas du codage binaire, s'il y a plusieurs paramètres, on a plusieurs formes de représentation possibles. Pour la plus courante, les paramètres sont représentés par une seule chaîne. Le premier paramètre occupe les n_0 premières places de la chaîne, le paramètre suivant les n_0 places suivantes et ainsi de suite. Dans ce genre de codage une chaîne représente effectivement une solution possible du problème. Les opérations standards du génétique seront faites sur une seule chaîne.
Exemple d'un problème à trois paramètres :

$$\xrightarrow{n_0}\xrightarrow{n_0}\xrightarrow{n_0}$$

1 0 1 0 1 0 1 1 1 1 1 1 0 0 0 0 0 1

Figure 19 : *représentation en une chaîne unique*

Les autres formes de codage

- Un chromosome est une suite de symboles. Classiquement, ces symboles sont binaires, donc chaque symbole a une *cardinalité* (nombre de caractères dans l'alphabet) de deux. Des symboles de plus grande cardinalité ont parfois été utilisés. Le problème reste ouvert pour savoir laquelle des représentations permet d'avoir le plus grand nombre de schémas et ainsi un plus grand degré de *parallélisme implicite.*

- Pour un codage binaire on a 2^l chaînes de longueur l possibles, et 3^l schémas. Pour un codage utilisant un alphabet de cardinal k si l' est la longueur du codage on a $k^{l'}$ chaînes et $(k+1)^{l'}$ schémas; il est facile de montrer que de tous les codages possibles, c'est l'alphabet binaire qui présente le nombre maximal de schéma par bit d'information. Et comme ce sont les similarités (schémas) qui sont à la base de la méthode d'exploration, il est tout naturel de maximiser leur nombre quand on met au point un codage.

- Cependant Goldberg dans sa théorie sur l'alphabet virtuel [24] a montré que pour les alphabets de grande cardinalité, chaque symbole converge très vite (en l'espace de quelque générations) ce qui laisse évidemment peu de valeurs possibles, et revient tout simplement à dire que chaque symbole a en fait une faible cardinalité. Ceci explique les bons résultats de certains codages non binaires.
  Par ailleurs, une représentation non binaire peut avoir beaucoup d'avantages quand les paramètres utilisés sont numériques, ce qui est souvent le cas dans les problèmes réels. Il peut donc être très intéressant d'utiliser des chromosomes dont chaque symbole représente directement un nombre réel. Cette représentation directe des paramètres réels nécessite de définir des opérateurs spécifiques.
  Ainsi a été défini le *croisement barycentrique* que l'on utilise pour les problèmes de type continus. Il consiste à choisir deux gènes $P_1(i)$ et $P_2(i)$ dans chacun des parents à la même position i, et à définir les gènes correspondants $C_1(i)$ et $C_2(i)$ chez les enfants par combinaison linéaire :

$$\left\{ \begin{array}{lcl} C_1(i) & = & \alpha P_1(i) + (1-\alpha) P_2(i), \\ C_2(i) & = & (1-\alpha) P_1(i) + \alpha P_2(i), \end{array} \right. \tag{2.2.22}$$

  Le coefficient α est tiré au hasard uniformément dans $[0, 1]$. On utilisera ce type de croisement dans le chapitre 3.

  La mutation généralement utilisée en cas de codage réel est la mutation gaussienne des stratégies d'évolution [61] (que nous présenterons également au chapitre 3). On pourra aussi se reporter au livre de Michalewicz [49].

2.2.11 La fonction Performance ou comment évaluer un individu

Principe
A chaque solution on associe une fonction performance qui décrit le mérite de l'individu que le chromosome représente. Dans le cas de l'optimisation par exemple, la performance est souvent la valeur de la fonction à optimiser. Ce n'est cependant pas toujours le cas.
Construction de la fonction mérite
Cette fonction est très importante pour les AGs, au même titre que le codage. Beaucoup d'études ont été menées pour optimiser les autres paramètres des AGs mais il s'avère qu'ils ont peu d'impact sur leur efficacité.
Grefenstette [27] montre qu'il existe un ensemble de paramètres idéal pour les opérateurs classiques de l'AG , mais il conclut que le mécanisme de base des algorithmes génétiques est si robuste que dans d'assez grands intervalles le choix des paramètres n'est pas critique. Ce qui est primordial étant la fonction mérite et le codage.

Pour que les AGs se comportent bien, nous devons trouver une manière de construire des fonctions mérite ne comportant pas trop de maxima locaux ou présentant un maximum

global isolé.
La construction de la fonction performance peut être assez évidente pour certains problèmes, dans le cas où l'on sait exactement ce que l'on veut optimiser (voir par exemple dans le chapitre 5 l'optimisation du moment d'inertie d'une barre en flexion).
En optimisation combinatoire, si l'on considère par exemple le problème du voyageur de commerce [28] (problème comportant beaucoup de contraintes), de nombreux points dans l'espace de recherche représentent des chromosomes invalides et donc ont une valeur nulle. C'est aussi le cas pour la mise en place d'emploi du temps pour des classes [9]. Des professeurs, des classes et des salles doivent être associés. La plupart des allocations de classe et de professeurs dans une salle peut violer plusieurs contraintes à savoir une salle occupée par deux classes en même temps, ou une classe ou un professeur se trouvant dans plusieurs salles au même moment, ou encore une classe se retrouvant avec un emploi du temps ne comportant pas tous les cours qu'elle est supposée recevoir.

- Pour que l'algorithme génétique soit performant dans ces cas de figures il faut construire une fonction mérite qui puisse évaluer la performance d'un mauvais individu en fonction de la capacité de ce mauvais individu à guider l'algorithme vers un bon chromosome. Dans ce genre de problèmes on doit savoir où se trouvent les bons chromosomes pour donner à des points voisins de bonnes valeurs, aux points éloignés des mauvaises valeurs.

- D'autres méthodes ont été suggérées pour calculer la fonction mérite lorsque le but doit être tout ou rien. On peut par exemple considérer des *sous-buts* [17]et récompenser les chromosomes qui atteignent ces sous-buts. Pour le problème des emplois du temps par exemple on peut récompenser toutes les classes qui ont leurs cours bien alloués.

- Une autre approche [22] consiste à utiliser des fonctions de *pénalisation* qui indiquent à quel point un chromosome est mauvais et construit la fonction mérite comme la différence entre une constante et la pénalisation.
  Quant à la construction de ces fonctions de pénalisation [35] il semble évident que les meilleures méthodes sont celles qui évaluent la gravité de la violation des contraintes et les pires sont celles qui indiquent simplement le nombre de contraintes violées. On peut également construire de bonnes fonctions de pénalisation en considérant le coût de la réparation, c'est-à-dire ce que coûterait de transformer un chromosome non faisable en un chromosome faisable.

- Une autre façon de procéder est de considérer une fonction approchée dans le cas où la vraie fonction est très difficile ou très lente à calculer [9]. Si une fonction beaucoup plus rapide et plus simple peut être construite et si cette nouvelle fonction donne approximativement la valeur de la fonction mérite, les AGs peuvent trouver pour un temps CPU donné, un meilleur chromosome que celui que l'on aurait eu (pour le même temps)avec la vraie fonction mérite.
  Ainsi, si la nouvelle fonction est 10 fois plus rapide on aura 10 fois plus d'évaluations de la fonction mérite dans le même temps.
  Une évaluation approchée de 10 points dans l'espace de recherche peut parfois être

plus avantageuse qu'un unique calcul exact en un seul point. Les AGs sont suffisamment robustes pour converger malgré la perturbation que représente l'approximation [22].

Remarques

Pour tout problème donné il faut définir un codage, une performance, et des opérateurs adaptés. C'est ce que nous verrons dans le chapitre 5.

2.3 La méthode du recuit simulé

2.3.1 Généralités

Introduite par Kirkpatrick et al. [58] cette méthode d'optimisation utilise l'analogie entre l'optimisation et la thermodynamique du recuit d'un solide.
Le processus de recuit consiste à chauffer le solide jusqu'à la fusion puis à le refroidir jusqu'à la cristallisation. Pour obtenir un état final de cristal parfait le refroidissement doit être conduit de façon à éviter des minima locaux de l'état énergétique qui induisent des imperfections.
Le comportement du corps solide pendant les variations de la température peut seulement être expliqué par le comportement du système de l'équilibre thermique, car le nombre d'atomes est très élevé ($= 10^{23}/cm^3$) ; l'équilibre énergétique est caractérisé par la distribution de Boltzmann qui donne la probabilité qu'un solide soit à l'état énergétique i avec l'énergie E_i à la température T

$$P_i(X = i) \quad = \quad \frac{1}{Z(T)} exp(-\frac{E_i}{K_B T}). \tag{2.3.23}$$

où X est une variable stochastique représentant l'état actuel du solide, et $Z(T)$ une fonction appelée fonction de partition définie par :

$$Z(T) \quad = \quad \sum_{j=1}^{n} exp(-\frac{E_j}{K_B T}) \tag{2.3.24}$$

et n représente le nombre d'états énergétiques possibles.
Pour simuler l'évolution d'un solide vers l'équilibre thermique à une température donnée T, Métropolis et al [52] ont introduit un algorithme basé sur la technique de Monte-Carlo.

A l'itération $K + 1$, une petite perturbation est imposée au système, par exemple le déplacement d'une particule, le changement énergétique correspondant ΔE est calculé.

- Si $\Delta E \leq 0$,
  le déplacement est accepté et la configuration est enregistrée comme point de départ de la prochaine itération.

- Si $\Delta E > 0$,
  la décision est traitée en termes probabilistiques. Soit $P(\Delta E) = exp(-\frac{\Delta E}{K_b T})$ Si $P(\Delta E) > \delta$ la configuration est acceptée pour initialiser la prochaine itération, sinon, elle est rejetée.
  δ est un nombre aléatoire uniformément généré dans [0,1].
  Ce processus permet d'éviter les minima locaux.

En répétant cette procédure plusieurs fois, le mouvement thermique des particules est simulé. Par un nombre d'itérations très élevé on doit atteindre l'équilibre thermique.
Le terme $P(\Delta E)$ signifie que le système évolue vers la distribution de Boltzman, on tend ainsi vers l'équilibre thermique. Kirkpatrick et al [58] en faisant l'analogie entre l'optimisation et le processus physique de recuit ont tiré plusieurs conclusions :

La détermination de l'état de basse énergie d'un système est un problème d'optimisation. On a en effet une équivalence entre la solution d'un problème d'optimisation à l'itération K et l'état énergétique d'un système physique à l'itération K dans l'algorithme de Metropolis. La fonction coût ou objectif joue le rôle de l'énergie.

En principe, dans les problèmes d'optimisation il n'y a pas de paramètre température "naturelle" mais la température peut être simulée simplement par un paramètre de contrôle qui jouera ce rôle.
Cela permet la résolution de problèmes d'optimisation. L'algorithme décrit en question est *le recuit simulé.*

Dans l'algorithme du recuit classique le processus commence avec une température élevée et des perturbations aléatoires sont réalisées sur chaque variable.
Pour chaque perturbation aléatoire une fonction coût est évaluée puis comparée avec sa valeur antérieure. Si on obtient une amélioration, la solution courante et la valeur de la fonction courante sont sauvegardées. Dans le cas contraire, la solution courante n'est sauvegardée que si la probabilité de Boltzmann est plus grande qu'un nombre aléatoire uniformément généré dans [0,1].
Cette procédure est répétée pendant un certain nombre de cycles jusqu'à ce que l'état de quasi équilibre soit atteint.
Puis, la température est réduite et une nouvelle itération est exécutée.

On montre que l'ensemble des points générés converge vers un minimum global quand la température tend vers 0.

2.3.2 Les paramètres critiques

Le principal problème rencontré pour trouver le minimum global en un temps fini, est à la détermination de l'ensemble des paramètres qui gouvernent la convergence de l'algorithme, qu'on appelle encore *le schéma de refroidissement.* Ces paramètres sont la valeur initiale du paramètre de contrôle (la *température initiale* T_0), le *facteur de réduction* de la température (F_r), le *nombre d'itérations* (longueur de la chaîne de Markov L_M) pour chaque valeur de température et le *critère d'arrêt.*
Comment déterminer T_0 ?
Dans le processus physique le solide doit être chauffé jusqu'à ce qu'il fonde pour que

dans la phase liquide les atomes ou particules puissent se déplacer et atteindre l'équilibre thermique.
Dans l'algorithme du recuit de base, le paramètre de contrôle initial doit être suffisamment élevé pour permettre à toutes les transitions d'être acceptées. Ce qui permet ainsi (on l'espère) de localiser l'endroit ou la zone où se situe le minimum global. Le paramètre T_0 varie d'un problème à l'autre.
Le taux d'acceptation des transitions approximativement égal à 1 dans le premier cycle peut être pris comme critère pour la détermination du facteur T_0. En pratique[58], on commence par une petite valeur et on la multiplie par un facteur supérieur mais proche de 1 jusqu'à l'obtention d'un taux d'acceptation proche de 1.
Le taux doit être calculé après un nombre minimum d'itérations.

Comment déterminer F_r ?

La valeur du facteur de réduction F_r, responsable de la réduction du paramètre de contrôle entre deux itérations consécutives, doit être inférieure mais proche de 1. Le refroidissement doit être conduit comme pour le processus physique (pour éviter les minima locaux). En général on choisit F_r dans l'intervalle [0.8 ; 0.99] [1][58].

Comment déterminer la longueur de la chaîne de Markov L_M?

Le nombre d'itérations ou longueur de la chaîne de Markov doit être suffisamment grand pour que l'équilibre thermique soit atteint (c'est-à-dire pour que le rapport entre le nombre d'acceptation n_a et le nombre de rejet n_r soit égal à 1) [58]. Dans le cas contraire on risque d'obtenir des minima locaux ou, comme dans le processus physique, d'avoir des structures en cristal présentant des imperfections locales.
Le critère d'équilibre thermique est en général trop sévère car il nécessite souvent un grand nombre d'itérations et est généralement remplacé par la notion de *quasi équilibre* [65].
On essaie de mettre à jour le paramètre de contrôle après un nombre minimal d'itérations et quand le rapport $\frac{n_a}{n_r} \in [0.8; 1.25]$ [65].

Comment déterminer le critère d'arrêt ?

On arrête le processus de recherche du minimum quand des amélioration sensibles ne sont plus réalisées ou quand le paramètre de contrôle est inférieur à une certaine limite. Un critère d'arrêt couramment utilisé [64] est :

$$\frac{< E > -E_{min}}{< E >} < \varepsilon$$

où $< E >$ est la valeur moyenne de la fonction à minimiser, E_{min} étant la solution courante, et ε représente la précision souhaitée.

Chapitre 3

Exemple d'optimisation géométrique

Dans ce chapitre nous présentons un exemple d'optimisation géométrique : "Peut-on entendre la forme du tambour ?" Ce problème inverse a été résolu par une méthode de variation de domaine par Hutchinson & Niordsen [33]. Nous présentons tout d'abord leurs résultats à titre de référence. Nous détaillons ensuite deux approches basées sur les algorithmes génétiques (voir également [42]), insistant sur la représentation utilisée, les opérateurs spécifiques mis au point, et une méthode adaptée de réduction de l'espace de recherche. Nous comparons finalement nos résultats à ceux de [33] sur le problème du tambour harmonique.

3.1 "Peut-on entendre la forme du tambour ?"

Le problème auquel nous nous interressons ici est celui de l'identification d'une forme de tambour ayant un son donné ("Peut-on entendre la forme du tambour ?"). Plus précisément, il s'agit de trouver un domaine de IR^2 tel que certaines valeurs propres de l'opérateur de Laplace (avec conditions aux limites de Dirichlet) aient des valeurs données. Détaillons tout d'abord le problème direct.

3.1.1 Le problème direct:

Soit Ω un ouvert de IR^n de classe $\mathcal{C}^\infty$, de frontière $\partial\Omega$. Dans Ω, l'opérateur $-\Delta$ avec conditions aux limites de type Dirichlet est défini de $\mathcal{D}(-\Delta_{\Omega,D})$ dans $L^2(\Omega)$, avec $\mathcal{D}(-\Delta_{\Omega,D}) = \Big\{u \in H^1_0(\Omega), \Delta\, u \in L^2(\Omega)\Big\}$.

Cet opérateur est auto-adjoint positif, et d'inverse compact. Il existe donc une base Hilbertienne de $L^2(\Omega)$, $\Big\{u_i \in \mathcal{D}(-\Delta_{\Omega,D}); i \geq 1\Big\}$ et une suite de réels strictement positifs $(\lambda_i)_{i\geq 1}$ telles que

$$\begin{cases} 0 < \lambda_1 \leq \lambda_2 \leq \ldots \lambda_i \leq \ldots < +\infty, \\ -\Delta\, u_i \;=\; \lambda_i\, u_i \text{ dans } \Omega,\ i = 1, 2, \ldots, \\ u_i = 0 \text{ sur } \partial\Omega,\ i = 1, 2, \ldots. \end{cases}$$

On notera $\Lambda_\Delta(\Omega)$ le spectre de Ω. Les valeurs propres λ_i sont caractérisées par le principe de Min-Max.

Remarquons enfin que si l'on change Ω en $\alpha\Omega$, pour une paramètre α positif, les valeurs propres de $-\Delta$ sont divisées par un facteur α^2. Il est donc en particulier toujours possible d'imposer une valeur donnée à lapremière valeur propre de l'opérateur de Laplace sur un domaine de forme quelconque en dilatant ce domaine par un facteur bien choisi. Cette remarque est à la base de la méthode de réduction de l'espace de recherche que nous introduirons section 3.5.

3.1.2 Le problème inverse

Dans ce chapitre, nous nous proposons de résoudre le problème inverse posé comme suit.

Etant donné un ensemble fini Λ de valeurs réelles positives, trouver un domaine Ω telle que ces valeurs soient les premières valeurs propres de l'opérateur de Laplace sur Ω, c'est-à-dire

$$\begin{cases} \Lambda \subset \Lambda_\Delta(\Omega) \\ \forall \mu \in \Lambda_\Delta(\Omega), \mu \notin \Lambda \Rightarrow \mu \geq \lambda \ \ \forall \lambda \in \Lambda \end{cases}$$

Le problème de l'unicité d'une forme de spectre donné est un problème encore ouvert. Il existe cependant des éléments de réponse dans le cas d'un ouvert de frontière régulière ($\mathcal{C}^\infty$), et des contre-exemples dans le cas d'un ouvert de frontière seulement $\mathcal{C}^0$. Il en découle que le problème inverse dans la classe des ouverts de frontière $\mathcal{C}^0$ est un problème mal posé ... ce qui n'empèche pas de chercher *une* forme ayant un spectre *se rapprochant le plus* possible d'un ensemble de valeurs positives données.

3.1.3 Le tambour harmonique

Le problème du tambour harmonique est le cas particulier le plus connu du problème général présenté ci-dessus. Un tambour est dit *harmonique* si ses plus basses fréquences de vibration sont dans les rapports 2:3:3:4 : si par exemple la première fréquence de vibration, ou *fondamentale* est un Do, la suivante (double) doit être un Sol, suivie ensuite par un Do a l'octave supérieur. Le problème est ici de trouver la forme de tambour présentant ces caractéristiques, dont le son est "harmonieux" à l'oreille.

En termes de spectre de l'opérateur $-\Delta$, il faut que les premières valeurs propres vérifient les relations suivantes :

$$\sqrt{\frac{\lambda_2}{\lambda_1}} = \sqrt{\frac{\lambda_3}{\lambda_1}} = \frac{3}{2}; \text{et } \sqrt{\frac{\lambda_4}{\lambda_1}} = 2$$

Compte-tenu de la propriété de proportionnalité des valeurs propres par dilatation du domaine (voir section 3.1.1 ci-dessus), on peut se fixer lapremière valeur propre, et se ramener au problème inverse classique de déterminer la forme ayant pour deuxième, troisième et quatrième valeurs propres les valeurs voulues.

3.2 Approche par variation de domaine

La méthode classique que nous allons détailler est basée sur des petites variations du domaine Ω. Soit ε un petit paramètre, et Ω_ε le domaine perturbé. La ième valeur propre de $-\Delta_{\Omega_\varepsilon,D}$ s'obtient alors par une petite correction (de l'ordre de ε) de la ième valeur propre de $-\Delta_{\Omega,D}$. Cette formule d'approximation (formule de Hadamard) est le point essentiel de l'algorithme de Hutchinson-Niordson :

A partir d'un domaine Ω_0 donné, on calcule les quatre premières valeurs propres et les vecteurs propres associés à ce domaine. Ces valeurs propres permettent, en inversant la formule d'Hadamard, de trouver la déformation du domaine à effectuer pour se rapprocher des valeurs propres désirées. Ce processus est répété pendant pendant un certain nombre d'itérations.

Formule d'Hadamard

Soit $(s, \vec{n})$ une paramétrisation locale du bord $\partial\Omega$ du domaine Ω supposé régulier (de classe C^∞;

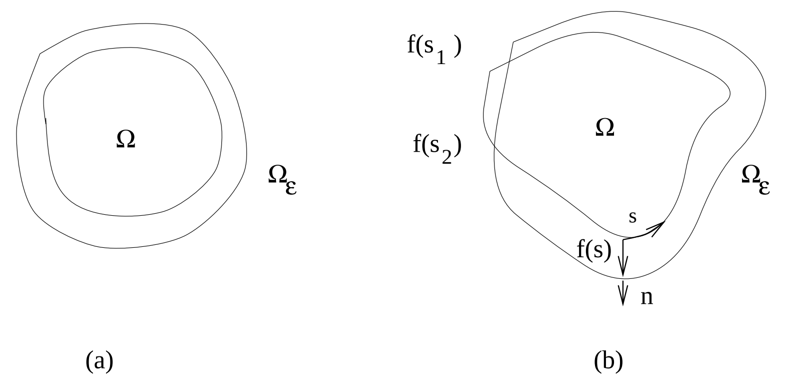


Figure 20 : *variation du domaine*

Soit λ_i la ième valeur propre de $-\Delta_{\Omega,D}$ et u_i le vecteur propre associé. On considère le domaine Ω_ε, pour un petit paramètre positif ε, obtenu à partir de Ω de la façcon suivante :

$$\Omega^\varepsilon \;=\; \Omega + \varepsilon f(s)\vec{n}(s),$$

où f est une fonction régulière à valeurs réelles, périodique de période la longueur de la frontière $\partial\Omega$. Ainsi, la frontière $\partial\Omega^\varepsilon$ de Ω_ε est donné par

$$\partial\Omega^\varepsilon = \partial\Omega + \varepsilon f(s)\vec{n}(s).$$

Par soucis de simplicité d'exposé uniquement, nous faisons l'hypothèse que f est strictement positive. Cette restriction n'est pas nécessaire pour établir la formule d'approximation. On note par λ_i^ε et u_i^ε la ième valeur propre et le ième vecteur propre de l'opérateur

$-\Delta_{\Omega_\varepsilon,D}$. $(\lambda_i^\varepsilon, {u_i}^\varepsilon(x))$ vérifient donc l'équation de Helmholtz suivante :

$$\left\{\begin{array}{rcll} \Delta {u_i}^\varepsilon + {\lambda_i}^\varepsilon {u_i}^\varepsilon & = & 0 & \textit{dans } \Omega^\varepsilon, \\ {u_i}^\varepsilon & = & 0 & \textit{sur } \partial\Omega^\varepsilon. \end{array}\right. \tag{3.2.1}$$

Comme ${u_i}^\varepsilon$ est de classe $\mathcal{C}^\infty$, nous pouvons écrire

$${u_i}^\varepsilon\,|_{\partial\Omega} \;=\; {u_i}^\varepsilon\,|_{\partial\Omega^\varepsilon} + \varepsilon f(s)\frac{\partial {u_i}^\varepsilon}{\partial n}\,|_{\partial\Omega} + 0(\varepsilon^2). \tag{3.2.2}$$

Ce qui donne

$${u_i}^\varepsilon(x)|_{\partial\Omega} = -\varepsilon f(s)\frac{\partial {u_i}^\varepsilon}{\partial n}\,|_{\partial\Omega} + O(\varepsilon^2). \tag{3.2.3}$$

D'autre part, en développant u_i^ε en puissance de ε on obtient :

$${u_i}^\varepsilon(x) = {u_i}^{(0)}(x) + \varepsilon {u_i}^{(1)}(x) + ... \quad \text{dans } \Omega, \tag{3.2.4}$$

où ${u_i}^{(0)}$, ${u_i}^{(1)}$ sont deux fonctions régulières définies dans Ω. Et de la même manière, nous pouvons écrire un développement de ${\lambda_i}^\varepsilon$ de la forme :

$${\lambda_i}^\varepsilon \;=\; {\lambda_i}^{(0)} + \varepsilon {\lambda_i}^{(1)} + ... \tag{3.2.5}$$

Pour ces deux développements asymptotiques, nous admettons que $\lambda_i^{(0)} = \lambda_i$ et $u_i^{(0)} = u_i$. Les équations (3.2.3) et (3.2.4) conduisent à

$${u_i}^{(0)}(x) + \varepsilon {u_i}^{(1)}(x)\,|_{\partial\Omega} \;=\; -\varepsilon f(s)\frac{\partial {u_i}^{(0)}}{\partial n}, \tag{3.2.6}$$

$${u_i}^{(0)}(x)\,|_{\partial\Omega} \;=\; 0, \tag{3.2.7}$$

ou encore,

$${u_i}^{(1)}(x)\,|_{\partial\Omega} \;=\; -f(s)\frac{\partial {u_i}^{(0)}}{\partial n}, \tag{3.2.8}$$

$${u_i}^{(0)}(x)\,|_{\partial\Omega} \;=\; 0. \tag{3.2.9}$$

De plus, les fonctions $u_i^{(0)}$ et $u_i^{(1)}$ sont solutions des deux problèmes suivants :

$$\left\{\begin{array}{rcll} \Delta {u_i}^{(0)} + {\lambda_i}^{(0)} {u_i}^{(0)} & = & 0 & \textit{dans } \Omega, \\ {u_i}^{(0)} & = & 0 & \textit{sur } \partial\Omega, \end{array}\right. \tag{3.2.10}$$

et

$$\left\{\begin{array}{rcll} \Delta {u_i}^{(1)} + {\lambda_i}^{(1)} {u_i}^{(0)} + {\lambda_i}^{(0)} {u_i}^{(1)} & = & 0 & \textit{dans } \Omega, \\ {u_i}^{(1)} & = & -f(s)\dfrac{\partial {u_i}^{(0)}}{\partial n} & \textit{sur } \partial\Omega, \end{array}\right. \tag{3.2.11}$$

En considérant le dernier problème portant sur ${u_i}^{(1)}$, en multipliant par ${u_i}^{(0)}$ et en intégrant, nous obtenons :

$$\int_\Omega (\Delta {u_i}^{(1)} + {\lambda_i}^{(0)} {u_i}^{(1)}) {\bar{u_i}}^{(0)} \quad = \quad -{\lambda_i}^{(1)} \int_\Omega \mid {u_i}^{(0)} \mid^2 . \tag{3.2.12}$$

de plus, nous avons :

$$\begin{aligned}
\int_\Omega (\Delta u_i^{(1)} + \lambda_i^0 u_i^{(1)}) {u_i}^{(0)} &= \int_\Omega \Delta u_i^{(1)} {u_i}^{(0)} + \lambda_i^0 \int_\Omega \Delta u_i^{(1)} {u_i}^{(0)} \\
&= \int_{\partial\Omega} \frac{\partial u_i^{(1)}}{\partial n} {u_i}^{(0)} - \int_{\partial\Omega} u_i^{(1)} \frac{\partial u_i^{(0)}}{\partial n} + \int_\Omega u_i^{(1)} \Delta {u_i}^{(0)} + \lambda_i^0 \int_\Omega u_i^{(1)} {u_i}^{(0)} \\
&= \int_{\partial\Omega} \frac{\partial u_i^{(1)}}{\partial n} {u_i}^{(0)} - \int_{\partial\Omega} u_i^{(1)} \frac{\partial u_i^{(0)}}{\partial n}
\end{aligned}$$

L'équation 3.2.12 peut encore se mettre sous la forme :

$$-\int_{\partial\Omega} \frac{\partial {u_i}^{(1)}}{\partial n} {\bar{u_i}}^{(0)} + \int_{\partial\Omega} {u_i}^{(1)} \frac{\partial {u_i}^{(0)}}{\partial n} \quad = \quad -{\lambda_i}^{(1)} \int_\Omega \mid {u_i}^{(0)} \mid^2 . \tag{3.2.13}$$

De :

$${u_i}^{(0)} \mid_{\partial\Omega} = 0,$$

et

$$\int_\Omega \mid {u_i}^{(0)} \mid^2 = 1,$$

il vient

$${\lambda_i}^{(1)} \quad = \quad -\int_{\partial\Omega} {u_i}^{(1)} \frac{\partial {u_i}^{(0)}}{\partial n}, \tag{3.2.14}$$

soit finalement:

$${\lambda_i}^{(1)} \quad = \quad -\int_{\partial\Omega} f(s) \left| \frac{\partial {u_i}^{(0)}}{\partial n} \right|^2 (s)\, ds. \tag{3.2.15}$$

L'approximation cherchée de la valeur propre λ_i^ε de l'ordre de ε^2 est donc donnée alors par la formule suivante

$${\lambda_i}^{\varepsilon} \quad = \quad \lambda_i + \varepsilon \int_{\partial\Omega} f(s) \left| \frac{\partial u_i}{\partial n} \right|^2 (s)\, ds + 0(\varepsilon^2). \tag{3.2.16}$$

3.2.1 Mise en oeuvre numérique

Soit donc un ensemble $\Lambda = (\lambda_i)_{i=1}^N$ formé de N valeurs propres: il s'agit de trouver un domaine Ω dont les N premières valeurs propres de $-\Delta_{\Omega,D}$ sont exactement les $(\lambda_i)_{i=1}^N$.

On part d'un domaine initial Ω_0 sur lequel on connaît explicitement les premières valeurs propres et vecteurs propres de l'opérateur $-\Delta_{\Omega_0,D}$ (e.g. un cercle). A partir de la formule d'approximation établie dans la section précédente, nous allons déformer de manière continue le domaine Ω_0. Deux cas de figures peuvent se présenter : soit on obtient un domaine Ω solution, soit on est sûr après un certain nombre d'itérations de ne jamais obtenir de solution au problème en partant de la forme Ω_0.

Trouver Ω à partir de Ω_0, revient à trouver $\varepsilon f(s)$ t.q.

$$\delta\,\lambda_i \;=\; \varepsilon \int_{\partial\Omega} f(s) \left|\frac{\partial u_i}{\partial n}\right|^2 (s)\, ds, \tag{3.2.17}$$

où $\delta\,\lambda_i$ est la différence entre λ_i et la ième valeur propre associée à Ω_0 et son ième vecteur propre.

On définit pour chacune des N fonctions propres u_i, une *fonction influence* g_i de la manière suivante :

$$g_i(s) \;=\; \left(\frac{\partial u_i}{\partial n}|_{\partial\Omega}\right)^2 (s). \tag{3.2.18}$$

Si les N fonctions d'influence associées avec cet ensemble de valeurs propres sont linéairement indépendantes alors la variation de forme $\varepsilon f(s)$ qui doit induire une telle variation sur les valeurs propres peut être cherchée sous la forme :

$$\varepsilon f(s) \;=\; \sum_{i=1}^{N} f_j g_j(s), \tag{3.2.19}$$

où les f_j sont les coordonnées de εf dans $(g_j)_{j=1}^N$, à déterminer. En utilisant la formule d'approximation, nous obtenons

$$\varepsilon f(s) \;=\; -\sum_{i=1}^{N}\sum_{j=1}^{N} {C_{ij}}^{-1} \delta\lambda_i g_j(s), \tag{3.2.20}$$

où C_{ij} est la matrice symétrique donnée par :

$$C_{ij} = \int_{\partial\Omega} g_i(s) g_j(s) ds; \tag{3.2.21}$$

Lorsque les fonctions d'influence ne sont pas indépendants - comme c'est le cas pour le cercle unité - cette méthode ne marche pas.
Cette méthode a été utilisée par Hutchinson [33]. Les résultats obtenus nous serviront de test de validation pour notre approche, que nous présentons maintenant.

3.3 Approche génétique

3.3.1 Représentation

Nous allons chercher une solution au problème dans l'ensemble des formes *étoilées*, construites à partir d'un ensemble fini de rayons donné. Une telle forme étoilée est alors définie de manière unique par la donnée des N rayons. Un individu I pour l'algorithme génétique est alors un vecteur de IR^N défini par la donnée de N longueurs de rayons $(r_1, r_2, ..., r_N)$, le domaine Ω t́ant étoilé construite à partir des $(r_1, r_2, ..., r_N)$ (voir Figure 22).

En termes d'algorithme génétique, nous utiliserons le codage réel décrit dans la section 2.2.10.

3.3.2 La performance

La performance d'un individu $(r_1, r_2, ..., r_N)$ sera une fonction des valeurs propres du domaine Ω contruit à partir de ces rayons. Le détail du calcul de la performance à partir des valeurs propres sera détaillée pour chaque problème. Mais dans tous les cas, on peut schématiser l'algorithme génétique de la manière suivante :

$$(r_1, r_2, ...r_n) \longrightarrow forme = \Omega \longrightarrow \lambda_1, \lambda_2, \lambda_3 \longrightarrow \text{performance pour l'A.G.} \qquad (3.3.22)$$

Pour résoudre le problème direct, c'est à dire pour calculer les n valeurs propres $\lambda_1, \lambda_2, ..., \lambda_n$ d'un domaine Ω donné, nous utilisons la méthode des éléments finis (méthode de Lanczo de la bibliothèque Modulef.

L'organigramme de l'algorithme est donné par la Figure 21. On utilisera une sélection basée sur la roulette avec une mise à l'échelle de facteur 2 décrits dans le chapitre 2. Nous allons maintenant préciser les opérateurs génétiques que nous avons utilisés.

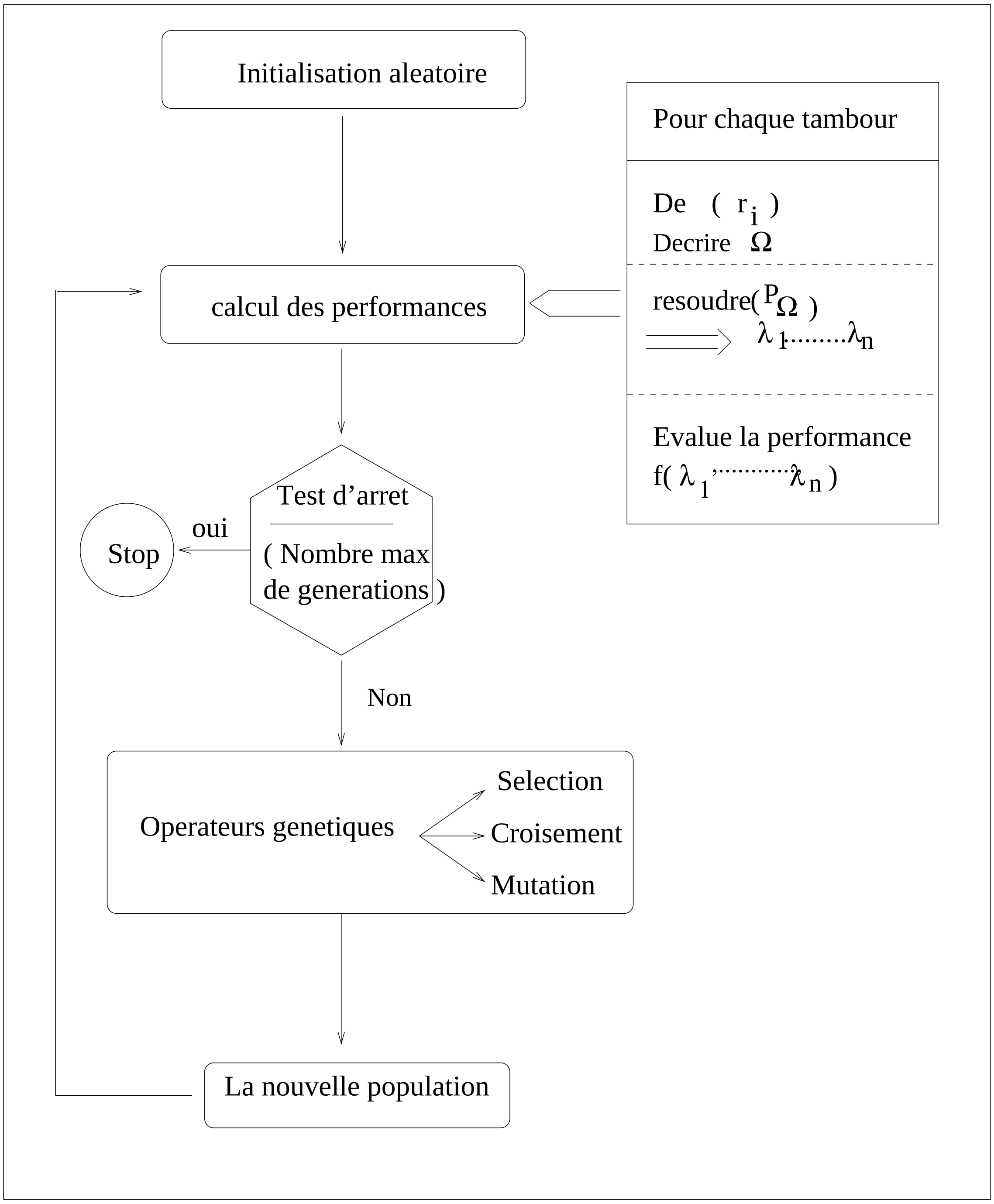


Figure 21 : *l'organigramme*

3.3.3 Les opérateurs génétiques

Un individu pour l'algorithme génétique est donc un ensemble de rayons : $(r_1, r_2, ..., r_N)$ (voir Figure 22)

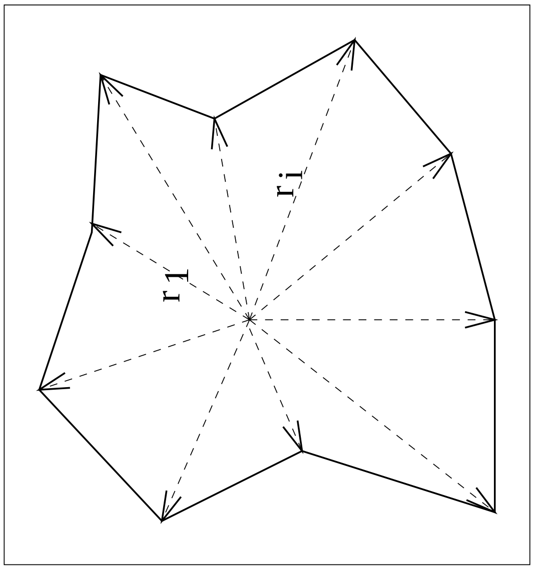


Figure 22 : *Modélisation d'un tambour*

Les principes de bases cités dans (Radcliffe citeRadcliffe) concernant les opérateurs génétiques nous conduisent à mettre en oeuvre des opérateurs de croisements non standards.

Le Croisement

Nous avons défini trois type de croisement, dont nous allons donner des exemples sur le couple de parents représenté Figure 23.

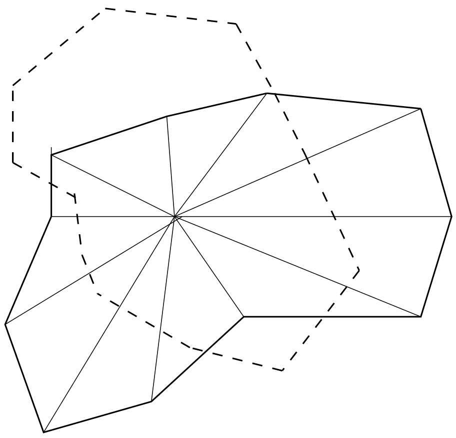


Figure 23 : *Les parents*

Croisement par inversion

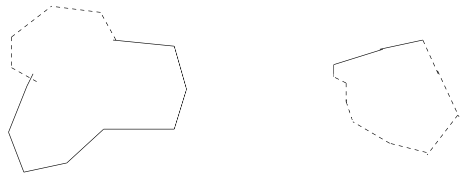


Figure 24 : *Inversion*

Le croisement par inversion considère les points correspondants aux deux plus petits rayons. Les rayons des parents sont échangés entre ces deux rayons (Figure 24).

Croisement extrême

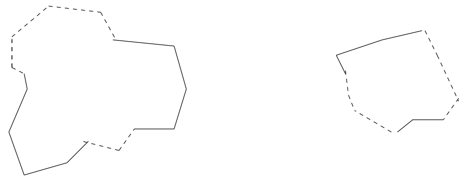


Figure 25 : *Extreme*

Le croisement extrême prend comme enfants les deux formes extrèmales construites à partir des parents pour la relation d'inclusion (Figure 25).

Croisement par combinaison

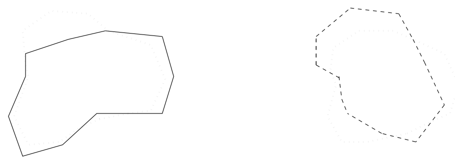


Figure 26 : *Combinaison*

On peut également combiner linéairement les valeurs des rayons (voir figure 26, rejoignant le croisement classique en algorithem génétique à codage réel [55, 49] :

$$(r_i), (r_i^{'}) \overset{Croisement}{\longrightarrow} \quad (\alpha * r_i + (1-\alpha) * r^{'}{}_i), \quad ((1-\alpha) * r_i + \alpha * r^{'}{}_i) \quad \text{avec } \alpha \in [0,1] \tag{3.3.23}$$

r_i est le i^{eme} rayon du premier parent, et $r^{'}{}_i$ le i^{eme} rayon du second parent (voir un exemple Figure 26).

La Mutation

Nous avons utilisé la mutation standard des nombres réels inspirée par les stratégies d'évolution ([61]) :

$$r_i \overset{mutation}{\longrightarrow} \quad r_i \; + \; N(0, \sigma) \tag{3.3.24}$$

$N(0, \sigma)$ étant la loi Gaussienne normale centrée en 0 de variance σ^2 ([61, 49]).

Le Sharing

Afin d'éviter une convergence prématurée de l'agorithme, nous avons mis en oeuvre la technique du partage (sharing) décite section 2.2.9. Toutefois, nous n'avons pas utilisé la distance euclidienne standard, mais défini une nouvelle distance plus appropriée au problème. Nous utiliserons comme distance entre deux individus la surface de la différence symétrique entre les deux formes qu'ils représentent (voir figure 27)

$$d(A, B) = surf(A \Delta B) = surf(A \cup B - A \cap B) \tag{3.3.25}$$

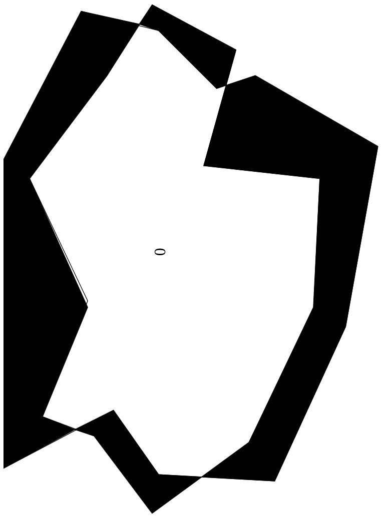


Figure 27 : *Distance spécifique*

3.4 Construction d'une forme connaissant une partie du spectre

Le premier des problèmes auquel nous nous sommes attaqué est le problème inverse aux valeurs propres : trouver la forme ayant des valeurs propres données. Dans un souci de simplicité, et afin de permettre une analyse précise des résultats, nous sommes partis d'une forme connue (le cercle unité) dont nous avons calculé les premières valeurs propres. L'objectif était donc de retrouver la forme de référence (le cercle) à partir de ces valeurs propres.

3.4.1 La fonction Performance

La fonction performance doit mesurer la proximité en terme de spectre entre la forme de référence et la forme définie par l'individu courant.
On considère donc une forme de référence Ω_0, forme étoilée définie par N rayons $r_1^0, r_2^0, \ldots, r_N^0$,

et les n premières valeurs propres de l'opérateur $-\Delta$ sur Ω_0 $\lambda_1^{obj}, \ldots, \lambda_n^{obj}$.
Soit une forme étoilée Ω de rayons $r_1, \ldots, r_N$. Si les n premières valeurs propres de l'opérateur $-\Delta$ sur Ω sont $\lambda_1, \ldots, \lambda_n$, on définit la performance de Ω par :

$$\mathcal{F}(\Omega) = 1 - \frac{1}{n}\sqrt{\sum_{i=1}^{n}\left(\frac{\lambda_i - \lambda_i^{obj}}{\lambda_i^{obj}}\right)^2} \tag{3.4.26}$$

3.4.2 Tests et résultats

Compte-tenu du caractère stochastique de l'algorithme, il ne faut rien déduire d'un unique calcul. C'est pourquoi tous les résultats présentés sont des moyennes sur 15 calculs indépendants (c'est-à-dire utilisant 15 populations aléatoires différentes comme point de départ).
Pour cet exemple, la forme de référence Ω_0 ici le cercle unité ($r_i{}^0 = 1, \forall i \in [1, N]$).

Génération	Num	Pcross	Pmut	σ	rmin	rmax
300	100	0.6	0.05	0.05	0.8	1.1

Table 3.4.2: les paramètres de l'algorithme.

La Table 3.4.2 donne les paramètres utilisés pour cet exemple.

- Génération est le nombre maximal de générations que nous utilisons pour suivre l'évolution de la population.
- Num est le nombre d'individus (formes étoilées) de la population.
- Pcross représente la probabilité de croisement entre deux individus.
- σ^2 est la variance de la mutation gaussienne des rayons. Ce paramètre caractérise la "force" de lamutation des formes, et a été ajusté par essais-erreur.
- *rmin* et *rmax* représentent respectivement le rayon minimal et maximal.

La Table 3.4.2 compare les trois premières valeurs propres obtenues en utilisant notre méthodes avec les valeurs de références, et donne l'erreur $1 - \mathcal{F}(\Omega)$ correspondante (rappelons qu'il s'agit d'une moyenne évaluée sur 15 tests).

Valeurs propres	Forme de référence	Meilleure moyenne	Erreur
λ_1	5.9883	5.987286	2.705659600E-08
λ_2	15.901	15.90049	
λ_3	16.221	16.22468	

Table 3.4.2: Analyse des trois premières valeurs propres

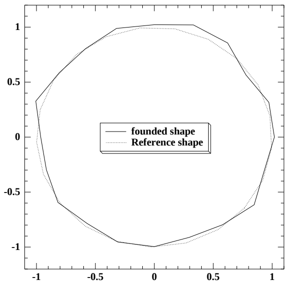


Figure 28 : *La meilleure forme obtenue.*

Les valeurs propres de la forme de référence et celles de la meilleure forme trouvée sont très proche bien que les formes correspondantes soient légèrement différentes (voir figure 28) Ceci peut résulter de la discrétisation relativement grossière que nous avons considérée.

3.5 Construction du tambour "harmonique"

Le tambour "harmonique" est caractérisé par un rapport de proportionnalité entre les fréquences correspondants aux 4 premiers modes propres : $\sqrt{\lambda_1}/2 = \sqrt{\lambda_2}/3 = \sqrt{\lambda_3}/3 = \sqrt{\lambda_4}/4$. Hutchinson et Niordsen [33] ont montré pour ce cas de tambour que les 4 premiers modes propres étaient suffisants pour déterminer grossièrement la forme du tambour.
La forme solution n'est pas connue exactement: il n'est plus question ici de comparer les valeurs propres d'une forme donnée à celles d'une forme de référence. Nous avons donc choisi la fonction performance suivante pour notre algorithme génétique :

$$f(\lambda_1, \lambda_2, \lambda_3, \lambda_4) = 100 - \left((\sqrt{\lambda_2} - \frac{3}{2}\sqrt{\lambda_1})^2 + (\sqrt{\lambda_3} - \frac{3}{2}\sqrt{\lambda_1})^2 + (\sqrt{\lambda_4} - 2\sqrt{\lambda_1})^2 \right)$$

Comme dans l'exemple précédent, le calcul de la performance requiert le calcul des valeurs propres, qui sera ici effectué par appel à la bibliothèque Modulef.

Remarque: La propriété d'harmonicité est invariante par dilatation. Les valeurs propres d'une forme Ω et les valeurs propres de la forme dilatée $\alpha\Omega$, pour $\alpha \in IR^{*+}$, vérifient une relation de proportionnalité.

$$\begin{array}{ccc} \Omega & \longrightarrow & \alpha\Omega \\ r_i & \longrightarrow & \alpha * r_i \\ \lambda_j & \longrightarrow & \frac{1}{\alpha^2} * \lambda_j \end{array}$$

Cette remarque va nous permettre de réduire la taille de l'espace de recherche par une normalisation des formes. Nous proposons dans la fin de ce chapitre deux méthodes de normalisation.

3.5.1 Normalisation moyenne

Nous imposons aux formes que nous manipulons d'avoir un rayon moyen égal à 1, c'est à dire de respecter la contrainte

$$\frac{1}{N}\sum_{i=1}^{N} r_i = 1$$

Pour ce faire, nous appliquons à toutes les formes crées au cours de l'algorithme (durant l'initialisation ou par application des opérateurs génétiques la transformation suivante:

$$r_i \longrightarrow \frac{r_i}{\sum_{i=1}^{N} r_i}$$

Résultats

Les notations sont les mêmes que celles définies en section 3.2. la Table 3.5.1 résument les paramètres que nous utilisons, et la Table 3.5.1 donne les quatre premières valeurs propres obtenues, ainsi que les rapports des valeurs propres (qui devrianet être égaux).

Génération	Num	Pcross	Pmut	σ	rmin	rmax
600	100	0.6	0.05	0.05	0.7	1.4

Table 3.5.1 : Les paramètres de l'AG.

λ_1	λ_2	λ_3	λ_4	$\sqrt{\lambda_1}/2$	$\sqrt{\lambda_2}/3$	$\sqrt{\lambda_3}/3$	$\sqrt{\lambda_4}/4$
5.9883	14.612	16.829	25.809	1.2235	1.2741	1.3674	1.2701

Table 3.5.1 : Valeurs propres obtenues

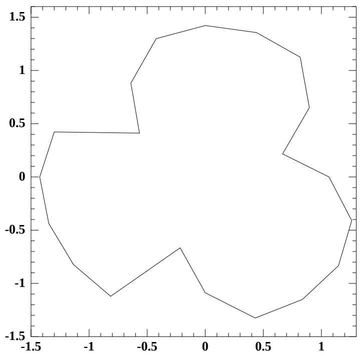


Figure 29 : *Le meilleur tambour obtenu à la génération 600*

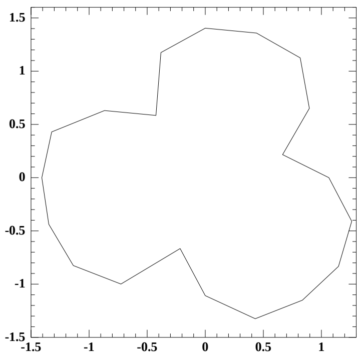


Figure 30 : *Le meilleur tambour obtenu à la génération 700*

Les résultats de cette méthode peuvent être améliorés en considérant un plus grand nombre de générations (voir les figures 29et 30). Nous nous heurtons à l'un des problèmes connus des algorithmes génétiques qui est le choix d'un critère d'arrêt. Le critère d'arrêt que nous avons utilisé est un nombre maximal de générations autorisé, et son choix est délicat; si on le choisit trop petit on n'atteint pas l'optimum, s'il est au contraire trop grand le coût de calcul, déjà important, devient insurmontable.

3.5.2 Normalisation par la première valeur propre

Une deuxième possibilité de normalisation est d'imposer la première valeur propre:

$$\lambda_1 = \nu_0$$

où ν_0 est fixé (nous l'avons pris égal à la première valeur propre du cercle unité), λ_1 est la première valeur propre de la forme de travail. Pour respecter cette containte, nous appliquons ici encore une trandformation à toutes les formes crées au cours de l'évolution. Cette transformation est donnée par les équations suivantes :

$$\begin{aligned} \alpha &= \frac{\lambda_1}{\nu_0} \\ r_i &= r_i \times \sqrt{\alpha} \\ \lambda_i &= \frac{\lambda_i}{\alpha} \end{aligned}$$

En utilisant les mêmes notations que précédemment, et avec les mêmes paramètres de l'AG que dans la section 3.5.1, nous avons obtenus les résultats présentés Table 6 et Figure 31. La figure 32 représente un tracé reconstitué du tambour harmonique obtenu par la méthode de variation de domaine de [33] décrite section 3.2.

λ_1	λ_2	λ_3	λ_4	$\sqrt{\lambda_1}/2$	$\sqrt{\lambda_2}/3$	$\sqrt{\lambda_3}/3$	$\sqrt{\lambda_4}/4$
5.9883	12.029	13.889	22.368	1.2235	1.1561	1.2423	1.1824

Table 6: Valeurs propres obtenues par la deuxième méthode de normalisation;

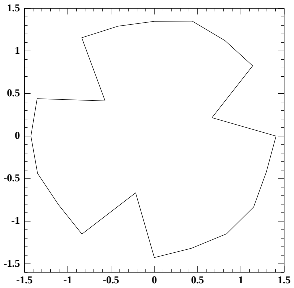


Figure 31 : *Le meilleur tambour obtenue à la génération 600.*

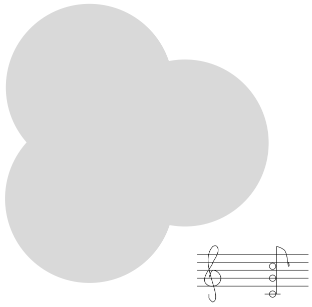


Figure 32 : *Le tambour harmonique (une esquisse)*

3.5.3 Discussion

D'une manière générale, la première méthode de renormalisation des formes a permis d'obtenir des formes apparemment plus proches de la forme en trèfle. Notons que la première méthode de normalisation est intrinsèque : seule la forme est prise en compte, ce qui n'est pas le cas dans la deuxième méthode, où la normalisation utilise la première valeur propre, dont le calcul est inévitablement entaché d'erreur.

Cependant, les résultats obtenus montrent que l'algorithme est sur la bonne voie, se rapprochant de la forme de trèfle espérée. Certes, nous n'obtenons jamais exactement le bon rapport $\frac{\lambda_i}{\lambda_1}$. Mais la raison principale en est sans doute que "le vrai tambour" a une forme régulière ce qui ne peut être approchée que très grossièrement par une forme polygonale avec 20 côtés. Le problème (ouvert) est de savoir à quelle point la solution obtenue est optimale dans l'espace de recherche considéré.

3.6 Conclusion

Nous avons proposé deux méthodes pour modéliser un tambour connaissant des propriétés sur son spectre en utilisant les AGs. Les résultats obtenus sont satisfaisants mais, il reste des problèmes non négligeables :

- La discrétisation utilisée est trop grossière (seulement 20 rayons décrivent le tambour). C'est probablement la principale raison pour les piètres figures obtenues en dépit des relativement bonnes valeurs propres obtenues pour le premier exemple. De plus l'unicité d'une forme pour un spectre donné n'est pas vérifié pour des formes non régulières (voir Gordon 89). Ce qui peut aussi expliquer la difficulté du problème dans l'espace des formes polygonales.

- L'autre inconvénient est le temps de calcul que cette méthode utilise si on utilise 600 générations de 100 individus, il faut à peu près 10 heures de temps CPU sur une station HP710.

En résumé, la résolution par algorithmes génétiques du problème inverse aux valeurs propres que nous avons présenté dans ce chapitre a montré a montré la faisabilité de l'approche génétique pour les problèmes d'optimisation de forme en mécanique des solides. Les résultats obtenus pourraient être probablement améliorés, d'une part en considérant une discrétisation plus fine (les puissances de calcul se sont accrues de plus d'un facteur 10 depuis la fin de l'étude présentée dans ce chapitre), d'autre part en utilisant les derniers raffinements des méthodes d'optimisation par évolution artificielle, comme par exemple l'idée de mutation adaptative pour les variables réelles ionspirées par l'école des Stratégies d'évolution, où la variance des mutations gaussiennes est déterminée "gratuitement" par l'évolution elle-même.

Mias le temps nous a manqué pour mener à bien ces expériences, d'autant qu'il n'est pas question pour les algorithmes génétiques de tenter de lutter contre les méthodes déterministes (telles la méthode de variation de domaine de [33]) lorsque ces dernières s'appliquent, mais bien de résoudre des problèmes jusqu'alors innaccessibles pour les autres méthodes. C'est ce que nous allons tenter de montrer dans le chapitre 5, sur des problèmes d'Optimum Design en calcul de structures, après avoir précisé notre cadre de travail au chapitre 4.

Chapitre 4

Un cadre théorique

Dans ce chapitre, nous commençons par présenter rapidement le cadre de travail introduit par Ghaddar, Maday et Patera (1994) [21] : Définition des espaces d'approximation, résultats de convergence et d'approximation pour le problème-test de l'optimisation de la forme de la section d'une barre droite. Nous étendons ensuite ces résultats au cadre de l'élasticité linéaire.

4.1 Formulation mathématique

Nous présentons dans cette section le cadre mathématique défini dans [21], commençant par introduire la problème-test – et donc la fonction objectif.

4.1.1 Le problème test

Pour étudier la résistance d'une barre de section droite S en (x,y) de longueur l (suivant z) à un moment exercé, posons :

$$I_{yy} = \int_S (y - y_0)^2 dxdy$$

I_{yy} est le moment d'inertie de la barre par rapport à la direction x et, (x_0, y_0) sont les coordonnés du centre de gravité O de la barre, définies par :

$$(x_0, y_0) = \frac{\int_S (x,y) dxdy}{\int_S dxdy}$$

le but est de maximiser I_{yy}, tout en minimisant le poids (i.e. le volume) de la pièce et le nombre de "coins" que peut avoir cette pièce (la notion de coin sera définie formellement dans la section suivante). Il faut de plus respecter les contraintes suivantes :

- il faut que Ω soit connexe, avec $\Omega \subset \Omega^{max}$ (Ω^{max} domaine de travail fixé).
- Ω doit avoir partout une épaisseur suffisante, pour des raisons de faisabilité technologique.

Le problème posé dans [21] est le suivant :

$$Min_{\Omega \subset \Omega_{Max}} C(\Omega) = \alpha_0 A + \alpha_1 P + \alpha_2 C + \alpha_3 I_{yy}, \tag{4.1.1}$$

oú A est l'aire de la section droite de la barre, P représente le périmètre de cette section, C est le nombre de coins et $\alpha_0, \ldots, \alpha_3$ sont des constantes positives.
La prise en compte des contraintes technologiques (épaisseur minimum, nombre de coin) et numériques (discrétisation du domaine de travail) conduit à la définition d'espaces originaux [21] que nous allons rappeler maintenant.

4.1.2 Définitions et hypothèses

On suppose que $\Omega_{Max} = [0, L] \times [0, L] \subset IR^2$. Soit

$$X_M^1 = \{S \in [0, L]^2 \text{tel que} S = \bar{\mathring{S}}\};$$

On suppose de plus que ∂S, la frontière de S, est composée de K courbes polygonales γ^j $(1 \leq j \leq K)$. Pour chaque courbe fermée γ^j, on notera m_j le nombre de sommets de γ^j, et $\boldsymbol{a_i^j}$ $(1 \leq i \leq m^j)$ les sommets de γ_j. On fait les hypothèses suivantes :

- Les bords sont rectilignes et les m^j sommets sont distincts : $\boldsymbol{a_i^j} = (a_i{}^j, b_i{}^j)$ vérifient la condition de rectilinéarité

$$(b_{i+1}{}^j - b_i{}^j)(a_{i+1}{}^j - a_i{}^j) = 0;$$

- Les coins sont de "vrais" coins :

$$(b_{i+1}{}^j - b_{i-1}{}^j)(a_{i+1}{}^j - a_{i-1}{}^j) \neq 0;$$

- Pour toute courbe $\gamma_j = \bigcup[\boldsymbol{a_i^j}, \boldsymbol{a_{i+1}^j}], 1 \leq i \leq m^j$ la condition suivante (courbe simple) est vérifiée :

$$(\boldsymbol{a_i^j}, \boldsymbol{a_{i+1}^j}) \bigcap (\boldsymbol{a_k^j}, \boldsymbol{a_{k+1}^j}) = 0, \text{ si } i \neq k$$

- Les γ_j sont d'intersection vide ;

$$\gamma^k \bigcap \gamma^j = \emptyset, \text{ si } k \neq j.$$

- Tout élément de X_M^1 a tout au plus un total de M coins.

$$V(S) = \sum_{k=1}^{K} m^k \leq M.$$

- Le contour de $S (\in X_M^1)$ est orienté de façon à ce que la normale unitaire $\boldsymbol{n}$, définie partout sauf aux coins, soit orientée vers l'intérieur du domaine. Elle peut ainsi être définie en tout point du contour en prenant comme valeur de la normale au coin la moyenne des normales des cotés voisins. Le choix de l'orientation de la normale à l'intérieur du domaine rend celui de γ_k unique.

Il est alors possible de définir lesnotions d'épaisseur et de coépaisseur d'un élément S de X_M^1 :

- à chaque coté $[\boldsymbol{a_i}^{\boldsymbol{k}}, \boldsymbol{a_{i+1}}^{\boldsymbol{k}}]$ de S est associée une famille de rectangles de largeur τ décrite par les sommets ordonnés $(\boldsymbol{a_i}^{\boldsymbol{k}}, \boldsymbol{a_{i+1}}^{\boldsymbol{k}}, \boldsymbol{b_{i+1}}^{\boldsymbol{k}}(\tau), \boldsymbol{b_i}^{\boldsymbol{k}}(\tau))$
avec $\boldsymbol{b_i}^{\boldsymbol{k}}(\tau) = \boldsymbol{a_i}^{\boldsymbol{k}} + \tau \boldsymbol{n}(\frac{1}{2}(\boldsymbol{a_i}^{\boldsymbol{k}}, \boldsymbol{a_{i+1}}^{\boldsymbol{k}}))$

- De même, à chaque coin $\boldsymbol{a_i}$ de S est associée une famille de carrés de coté τ et de diagonale $[\boldsymbol{a_i}^{\boldsymbol{k}}, \boldsymbol{a_i}^{\boldsymbol{k}} + \tau\sqrt{2}\boldsymbol{n}]$.

L'*épaisseur* $E(S)$ est définie comme étant le maximum de tous les τ tels que tout rectangle ou carré de coté τ soit dans S pour tout k,$1 \leq k \leq K$ pour tout i $1 \leq i \leq m^k$. La *coépaisseur* $\bar{E}(S)$ est définie comme étant l'épaisseur de la fermeture du complémentaire de S dans $[0, L]^2$. L'épaisseur et la coépaisseur d'un élément S de X_M^1 sont strictement positives.

On peut ainsi définir, pour tous t et $\bar{t}$ positifs, les espaces $X_M^2(t, \bar{t})$ et $X_M^3(t, \bar{t})$ par

$$X_M^2(t, \bar{t}) = \{S \in X_M^1 \mid E(S) \geq t, \bar{E}(S) \geq \bar{t}, (t, \bar{t}) \leq t_{max}\}, et$$

$$X_M^3(t, \bar{t}) = \{S \in X_M^2(t, \bar{t}) \mid S \text{ connexe}\}$$

Les espaces X_M^i $(i \in \{1, 2, 3\})$, munis de la distance δ définie par

$$\delta(A, B) = \text{Aire } (A \Delta B), \text{ avec } A \Delta B = (A \bigcup B) \setminus (A \bigcap B),$$

sont des espaces métriques.
Le problème se ramène alors à :

$$Min_{S \subset X_M^3(t,\bar{t})} C(S), \text{ avec}$$

$$\begin{aligned}
C(S) &= \alpha_0 \left[\sum_{k=1}^{K} \sum_{i=1}^{m_k} \frac{1}{2} ({b_{i+1}}^k + {b_i}^k)({a_{i+1}}^k + {a_i}^k) \right] \\
&+ \alpha_1 l \left[\sum_{k=1}^{K} \sum_{i=1}^{m_k} \{({b_{i+1}}^k + {b_i}^k)^2 ({a_{i+1}}^k + {a_i}^k)^2\}^{\frac{1}{2}} \right] \\
&+ \alpha_2 \sum_{k=1}^{K} m_k + \alpha_3 f \left(\left[\sum_{k=1}^{K} \sum_{i=1}^{m_k} \frac{1}{3} ({b_i}^k - y_S)^3 ({a_{i+1}}^k + {a_i}^k) \right] / I_0 \right)
\end{aligned}$$

si

$$(x_G, y_G) = \frac{1}{Aire(S)} \left(\sum_{k=1}^{K} \sum_{i=1}^{m_k} \frac{{b_i}^k}{2} ({a_{i+1}}^k - {a_i}^k)^2, \sum_{k=1}^{K} \sum_{i=1}^{m_k} \frac{({b_i}^k)^2}{2} ({a_{i+1}}^k + {a_i}^k) \right)$$

est le centre de gravité de S.

4.1.3 Résultats théoriques

Les résultats obtenus dans [21] sur le problème de minimisation ci-dessus sont les suivants:

- Le problème de minimisation de la fonction coût C a au moins une solution dans $X^3_M \in \Omega$.

  Cette démonstration repose sur le fait que (X^3_M, δ) est séquentiellement compact et que C est semi continue inférieurement sur (X^3_M, δ)

- Résultats d'approximation dans le cas du problème discret et estimation d'erreur possible

Nous allons maintenant présenter la transposition de ces résultats au cadre de l'élasticité linéaire.

4.2 Résultats en élasticité linéaire

4.2.1 Rappel

En utilisant les espaces de formes précédemment introduits, nous allons montrer que les fonctions que nous serons amenés à utiliser par la suite (voir Chapitre 5) sont semi continues inférieurement dans ces espaces.

La première fonction coût que nous avons utilisée (voir section 6.3.4) prend en compte la surface de la forme considérée, ainsi que le déplacement maximal de cette structure soumise à un chargement donné. Elle est donnée par :

$$C(A) = Aire(A) + \alpha(|u_A|_{L^\infty(\Gamma_1)} - u_{Lim})^+ \tag{4.2.2}$$

Sa minimisation entraine celle de l'aire de la forme A, pénalisée pour prendre en conpte des contraintes sur le déplacement u_A de cette structure, qui ne doit pas dépasser la valeur $u_{Lim} \geq 0$, donnée.
La seconde fonction coût est basée sur la notion de compliance introduite section 1.1.3, et est donnée par :

$$C(A) = Aire(A) + \alpha \int_{\Gamma_1} f u_A da \tag{4.2.3}$$

où f est l'intensité de la force appliquée sur une partie Γ_1 de ∂A bord du domaine. u_A est le déplacement maximal obtenu. Minimiser cette fonction revient à minimiser le poids et le travail des efforts externes.

Dans les ceux cas, nous allons chercher à montrer la semi-continuité inférieure de la fonction C, i.e. :

$$\begin{array}{l} \text{Soit} A, B \in X^3_M(t, \bar{t}) \\ \forall \varepsilon > 0,\ \exists \mu > 0 \text{ tel que } \delta(A, B) \leq \mu \text{ alors } C(B) \geq C(A) - \varepsilon \end{array}$$

Les deux fonctions coût décrites ci-dessus font intervenir le déplacement u_A de la structure soumise à un chargement donné. Nous nous p[laçons dans le cadre de l'élasticité linéaire, et ce déplacement est obtenu comme solution du problème classique :

La structure $A \subset IR^2$ est supposée constituée par un matériau élastique. Le déplacement u_A de A soumise à des forces f sur la partie $\partial\Gamma_1$ de sa frontière et fixée sur la partie $\partial\Gamma_2$ de sa frontière satisfait les équations [16]

$$P_A \begin{cases} -div\sigma = 0 & \text{dans } A\backslash T, \\ \sigma \cdot n = 0 & \text{sur } \partial A\backslash(\Gamma_1 \cup \Gamma_2), \\ \sigma \cdot n = f & \text{sur } \Gamma_1, \\ u_A = 0 & \text{sur } \Gamma_2. \end{cases} \tag{4.2.4}$$

où :

$$\sigma = K\epsilon(u_A) = \lambda tr\varepsilon 1_E + 2\mu\varepsilon \tag{4.2.5}$$

$$\epsilon(u) = \frac{1}{2}(\nabla u + \nabla u^T) \tag{4.2.6}$$

$$T = \{T_i, i \in IN\} \tag{4.2.7}$$

K est le tenseur d'élasticité (loi de Hooke), ϵ est le tenseur des déformations linéarisées, et T est l'ensemble des trous.

Toute fois, dans un souci de lisibilité, nous allons présenter les résultats de continuité dans le cadre plus simple où u_A est solution du problème de Laplace avec conditions de Dirichlet dans A.

4.2.2 Position du problème

Soit A un domaine dans $X_M^3(t, \bar{t})$, et u_A la solution du problème S_A :

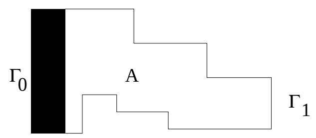


Figure 33 : *Domaine quelconque*

$$S_A \begin{cases} \Delta u = 0 & \text{dans } A, \\ \dfrac{\partial u}{\partial n} = 0 & \text{sur } \partial A\backslash(\Gamma_1 \cup \Gamma_0), \\ \dfrac{\partial u}{\partial n} = f & \text{sur } \Gamma_1, \\ u = 0 & \text{sur } \Gamma_0. \end{cases} \tag{4.2.8}$$

soit $C(A) = C_1(A) + C_2(A)$

Où $C_1(A) = \int_{\Gamma_1} f u_A da$ et $C_2(A) = Aire(A)$

On veut montrer que cette fonction est semi continue inférieurement. La semi continuité de C_2 résulte immédiatement de la définition de la distance δ. Il suffit donc de montrer la semi continuité de C_1 dans $X_M^3(t, \bar{t})$, i.e. :

$$\forall \varepsilon, \exists \eta \; \forall B \;, \; \delta(A, B) < \eta \Longrightarrow C_1(B) \geq C_1(A) - \varepsilon. \tag{4.2.9}$$

soit encore

$$C_1(A) - C_1 B) = \int_{\Gamma_1} f u_A da - \int_{\Gamma_1} f u_B da \leq \varepsilon \tag{4.2.10}$$

Pour ce faire, nous allons considérer deux domaines A fixé et B variable, avec u_A et u_B solutions respectifs de S_A et S_B Toujours par souci de simplicité, nous allons supposer que le contour Γ_1 où sont appliquées les charges est le même.
Les déplacements u_A et u_B vérifient respectivement les systèmes suivants :

$$S_A \begin{cases} \Delta u_A = 0 \quad \text{dans } A, \\ \dfrac{\partial u_A}{\partial n_A} = 0 \quad \text{sur } \partial A \backslash (\Gamma_1 \cup \Gamma_0), \\ \dfrac{\partial u_A}{\partial n_A} = f \quad \text{sur } \Gamma_1, \\ u_A = 0 \quad \text{sur } \Gamma_0. \end{cases} \qquad S_B \begin{cases} \Delta u_B = 0 \quad \text{dans } B, \\ \dfrac{\partial u_B}{\partial n_B} = 0 \quad \text{sur } \partial B \backslash (\Gamma_1 \cup \Gamma_0), \\ \dfrac{\partial u_B}{\partial n_B} = f \quad \text{sur } \Gamma_1, \\ u_B = 0 \quad \text{sur } \Gamma_0. \end{cases} \tag{4.2.11}$$

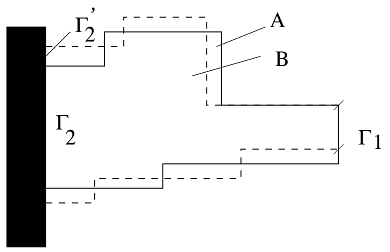


Figure 34 : *deux domaines*

Posons $\Omega = (A \cap B)$ alors $u = u_A - u_B$ vérifie le problème suivant :

$$S_{A \cap B} \begin{cases} \Delta u = 0 \quad \text{dans } A \cap B = \Omega, \\ \dfrac{\partial u}{\partial n} = g \quad \text{sur } \partial (A \cap B) \backslash \Gamma_0, \\ u = 0 \quad \text{sur } \Gamma_0. \end{cases} \tag{4.2.12}$$

où la fonction g est donnée par :

$$g = \begin{cases} \dfrac{\partial u_A}{\partial n_B} & \text{sur } \partial B \cap (\partial(A \cap B) \backslash \Gamma_0), \\ \\ -\dfrac{\partial u_B}{\partial n_A} & \text{sur } \partial A \cap (\partial(A \cap B) \backslash \Gamma_0), \\ \\ 0 & \text{sur } \partial A \cap \partial B. \end{cases} \tag{4.2.13}$$

Pour éviter les problèmes au coins, définissons la normale n à $\partial(A \cap B)$ de manière faible par dualité.

Pour toute fonction Φ suffisamment régulière,

$$\int_\Gamma \frac{\partial u}{\partial n}\Phi = \int_\Omega \Delta u \Phi + \int_\Omega \nabla u \nabla \Phi.$$

Pour $v \in V = \{v = (v_i) \in H^1(\Omega) \text{ tel que } v = 0 \text{ sur } \Gamma_0\}$
Dans ce cas,

$$\int_\Omega \Delta u v = \int_\Gamma g v - \int_\Omega \nabla u \nabla v$$

et donc pour v=u, il vient

$$0 = \int_\Gamma g u - \int_\Omega (\nabla u)^2 \tag{4.2.14}$$

Utilisant l'inégalité de Poincarré, on obtient :

$$\int_\Gamma g u = \int_\Omega (\nabla u)^2 \geq C \mid\mid u \mid\mid^2_{H^1(\Omega)} \tag{4.2.15}$$

où C ne dépend que de la constante de lipschitz du domaine, soit ici de $(t, \bar{t})$.

L'évaluation de $\int_{\Gamma_1} f(u_A - u_B)$ nous donne :

$$\begin{aligned} \mid \textstyle\int_{\Gamma_1} f(u_A - u_B) \mid = \mid \int_{\Gamma_1} f u \mid &\leq |f|_{0,\Gamma_1} |u|_{0,\Gamma_1} \\ &\leq |f|_{0,\Gamma_1} |u|_{0,\Gamma} \\ &\leq C' |f|_{0,\Gamma_1} \mid\mid u \mid\mid_{H^1(\Omega)} \end{aligned}$$

où C' est une constante qui ne dépend que de t et $\bar{t}$ (voir Grisvard [30]). La relation obtenue par l'inégalité de Poincarré (voir 4.2.15) donne alors :

$$C \mid\mid u \mid\mid^2{}_{H^1(\Omega)} \leq \mid \textstyle\int_\Gamma g u da \mid \leq |g|_{0,\Gamma} |u|_{0,\Gamma} \leq C' |g|_{0,\Gamma} \mid\mid u \mid\mid_{H^1(\Omega)}$$

Soit encore :

$$\mid\mid u \mid\mid^1_H (\Omega) \leq C'' |g|_{0,\Gamma}.$$

Il ne reste donc plus qu'à évaluer $|g|_{0,\Gamma}$ (avec $g = \dfrac{\partial(u_A - u_B)}{\partial n}$).

Sachant que $\dfrac{\partial u_A}{\partial n_A} = 0$, l'évaluation de $\displaystyle\int_{\partial B} \frac{\partial u_A}{\partial n}$ ne pose pas de problème loin des coins de $(\partial A \cap \partial B)$: tant que l'on est pas au voisinage d'un coin, $\dfrac{\partial u_A}{\partial n}$ est proche de $\dfrac{\partial u_A}{\partial n_A}$ et donc dans ce cas:

$$\frac{\partial u_A}{\partial n} = O(|A - B|)$$

Par contre, plusieurs problèmes se posent au voisinage des coins “intersections”. Nous allons détailler un cas de figure illustré Figure 35.

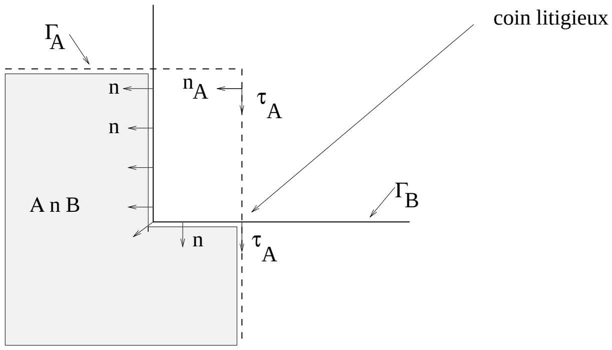


Figure 35 : *le problème des coins*

Au voisinage d'un coin de $(\partial A \cap \partial B)$ on a :

$$\frac{\partial u_A}{\partial n} = \frac{\partial u_A}{\partial \tau_A} \text{ lorsque } n = n_B.$$

Cette dérivée tangentielle peut être très grande lorsque les les efforts exercés sont importants, on a donc pas forcément $\dfrac{\partial}{\partial n}(u_A - u_B) = O(|A - B|)$. Il faut donc traiter le problème autrement. Pour cela, introduisons la proposition suivante :

Proposition 4.2.1

$$|| \, u_B \, ||_{H^1(B)} \leq C|f|_{0,\Gamma_1} \leq C$$

Considérons maintenant les deux espaces V et V_1 définis par :

$V_1 = \{u \in H^1(A \cap B) \; ; \; u = 0 \text{ sur } \Gamma_0\}$
$V = \{u \in H^1([0, L]^2) \; ; \; u = 0 sur \text{ sur } \Gamma_0\}$
$[0, L]^2$ est le compact qui contient A et tous les B est qui est définie dans $X^3_M(t, \bar{t})$.

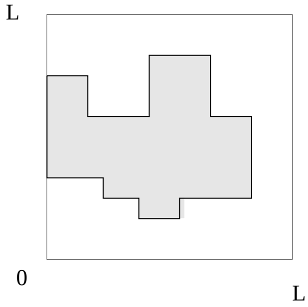


Figure 36 : *Prolongement*

Introduisons enfin le théorème de prolongement de Caldérone

Theorem 4.2.2

Soit U un ouvert lipschitzien de IR^N. Alors il existe un opérateur de prolongement $P : H^k(U) \longrightarrow H^k(IR^N)$.

$$|| \, Pu \, ||_{H^k(IR^N)} \leq C_k^{'} \, || \, u \, ||_{H^k(U)}$$

où $C_k^{'}$ ne dépend que de la k et de la constante de lipschitz du domaine.

Remarque

On peut donc trouver un $C_3^{'}$ uniforme tel que :
$|| \, u_B \, ||_{H^1(IR^2)} \leq C_3^{'} \, || \, u_B \, ||_{H^1(B)}$.
Si on confond u_B avec son prolongement Pu_B, et en utilisant la proposition 4.2.1, on peut donc trouver une constante C_4 telle que:

$$\forall B \in X_M^3(t, \bar{t}) \text{ tel que } \Gamma_0 \cup \Gamma_1 \subset \partial B$$

$$|| \, u_B \, ||_{H^1(IR^2)} \leq C_4 \tag{4.2.16}$$

On peut écrire maintenant que pour tout v dans V

$$\int_{\Gamma_1} fv = \int_A \nabla u_A \nabla v$$

et que

$$\int_{\Gamma_1} fv = \int_B \nabla u_B \nabla v$$

pour $v = u_A - u_B$

$$\int_{\Gamma_1} f(u_B - u_A) = \int_{A \cap B} \nabla u_A \cdot \nabla(u_B - u_A) + \int_{A-(A\cap B)} \nabla u_A \cdot \nabla(u_B - u_A) \tag{4.2.17}$$

de la même façon il vient :

$$\int_{\Gamma_1} f(u_B - u_A) = \int_{A \cap B} \nabla u_B \cdot \nabla(u_B - u_A) + \int_{B-(A\cap B)} \nabla u_B \cdot \nabla(u_B - u_A) \tag{4.2.18}$$

La différence des équations 4.2.18 et 4.2.17 donne :

$$\int_{A\cap B} \nabla(u_B - u_A)\cdot\nabla(u_B - u_A) = \int_{A-(A\cap B)} \nabla u_A \cdot \nabla(u_B - u_A) - \int_{B-(A\cap B)} \nabla u_B \cdot \nabla(u_B - u_A)$$

d'où :

$$\underbrace{\int_{A\cap B} \nabla(u_B - u_A)\cdot\nabla(u_B - u_A)}_{(1)} + \underbrace{\int_{B-(A\cap B)} \nabla(u_B - u_A)\cdot\nabla(u_B - u_A)}_{(2)}$$

$$-\underbrace{\int_{A-(A\cap B)} \nabla u_A \cdot \nabla(u_B - u_A)}_{(3)} + \underbrace{\int_{B-(A\cap B)} \nabla u_A \cdot \nabla(u_B - u_A)}_{(4)} = 0$$

Nous allons estimer chacun des termes de l'égalité précédente.
On a déjà,

$$(3) : \int_{A-(A\cap B)} \nabla u_A \cdot \nabla(u_B - u_A) \leq c_0 \mid\mid u_A \mid\mid_{H^1(A-A\cap B)} \mid\mid (u_B - u_A) \mid\mid_{H^1(A-A\cap B)}$$

$$(4) : \int_{B-(A\cap B)} \nabla u_A \cdot \nabla(u_B - u_A) \leq c_1 \mid\mid u_A \mid\mid_{H^1(B-A\cap B)} \mid\mid (u_B - u_A) \mid\mid_{H^1(B-A\cap B)}$$

En considérant le prolongement de $(u_B - u_A)$ dans $H^1(IR^2)$ on a par 4.2.16

$$\mid\mid (u_B - u_A) \mid\mid_{H^1(IR^2)} \leq 2C_4$$

Soit maintenant $\varepsilon > 0$

$$\exists \eta < 1 \text{ tel que } |\Omega'| < \eta \Longrightarrow \mid\mid u_A \mid\mid_{H^1(\Omega')} \leq \varepsilon'.$$

D'où : pour $\delta(A,B) < \eta$ on a $Aire(A - A\cap B) < \eta$ et $Aire(B - A\cap B) < \eta$

$$\mid\mid u_A \mid\mid_{H^1(A-A\cap B)} \mid\mid (u_B - u_A) \mid\mid_{H^1(IR^2)} \leq \varepsilon' C_4$$

et

$$\mid\mid u_A \mid\mid_{H^1(B-A\cap B)} \mid\mid (u_B - u_A) \mid\mid_{H^1(IR^2)} \leq \varepsilon' C_4.$$

D'autre part

$$\int_{B\setminus(A\cap B)} \nabla(u_B - u_A)\cdot\nabla(u_B - u_A) \geq 0$$

donc :

$$\int_{A\cap B} \nabla(u_B - u_A)\cdot\nabla(u_B - u_A) \leq 2(\varepsilon' C_4)$$

On obtient ensuite par l'inégalité de Poincarré

$$c \mid\mid u_B - u_A \mid\mid^2_{H^1(A\cap B)} \leq \int_{A\cap B} \nabla(u_B - u_A)\cdot\nabla(u_B - u_A) \leq \varepsilon_1$$

et donc

$$\mid \int_\Gamma f(u_B - u_A)\mid \leq C|f|_{0,\Omega} \mid\mid u_B - u_A \mid\mid_{H^1(A\cap B)} \leq \varepsilon$$

Ce qui assure la continuité de $\int_\Gamma fu$

4.2.3 Le problème d'élasticité

Soit $A \in I\!R^2$ constitué par un matériau élastique. On considère le problème P_A suivant :

$$P_A \begin{cases} -div\sigma = 0 & \text{dans } A\backslash T, \\ \sigma\cdot n = 0 & \text{sur } \partial A\backslash(\Gamma_1\cup\Gamma_2), \\ \sigma\cdot n = \mathcal{F} & \text{sur } \Gamma_1, \\ u = 0 & \text{sur } \Gamma_2. \end{cases} \tag{4.2.19}$$

où :

$$\sigma = K\epsilon = \lambda tr\varepsilon 1_E + 2\mu\varepsilon \tag{4.2.20}$$

$$\epsilon(u) = \frac{1}{2}(\nabla u + \nabla u^T) \tag{4.2.21}$$

$$T = \{T_i, i \in I\!N\} \tag{4.2.22}$$

K est le tenseur d'élasticité (loi de Hooke), ϵ est le tenseur des déformations linéarisées, et T est l'ensemble des trous.

Remarques

- Pour montrer la continuité de la fonction coût le raisonnement est analogue au précédent, l'inégalité de Korn étant valable pour un domaine lipschtzien (voir Oleinik [45]).
- Nous avons admis la semi continuité de la deuxième fonction performance.

Chapitre 5

Codage et Opérateurs

Dans ce chapître, nous nous intéressons spécifiquement à la représentation d'une forme et à la construction des opérateurs génétique (voir [43]).
Le choix de l'espace de représentation est l'un des principaux problèmes des AGs. Le codage binaire a longtemps été le seul utilisé, arguments a l'appui [24]. Ces arguments ont cependant maintenant été battus en brèche [5, 55], et la tendance actuelle est plus d'adapter le codage au problème que de chercher à tout prix à utiliser le codage binaire. L'exemple le plus frappant est le codage de paramètres réels [36, 49] où les algorithmes génétiques rejoignent les stratégies d'évolution [61]. Mais de nombreux autres domaines d'application utilisent des représentations adaptées (il existe par exemple de nombreuses représentations pour le problème du voyageur de commerce [54]).

Enfin, même en ne considérant que le cadre du codage binaire, pour un problème donné, il existe différents codages de coder conduisent à des comportements différents pour l'algorithme génétique. Par exemple, l'utilisation du codage Gray (deux nombres consécutif ont un bit d' écart) peut modifier complètement les résultats des algorithmes génétiques [12].

Dans ce chapitre nous considérons le problème du choix d'une bonne représentation pour l'Optimisation Topologique de Formes : nous avons choisi un tableau de bits pour représenter une forme dans un domaine donné.
L'approche représentation par tableau de bits a déjà été utilisée par Jensen [37], Chapman et al. [15] pour le problème particulier du cantilever. D'une part il s'agit de l'approche la plus “naturelle” compte-tenu de la nécessité d'utiliser la méthode des éléments finis (voir chapitre 6). D'autre part, elle s'intègre a priori parfaitement dans le cadre traditionnel des algorithmes génétiques décrit au chapitre 2.

Nous allons néanmoins faire apparaître la nécessité de définir des opérateurs de croisement spécifiques pour le problème de l'optimisation des formes: une matrice de bits n'est pas, du point de vue des algorithmes génétiques, équivalent à une chaîne de bits. Enfin, les difficultés que rencontrent souvent les algorithmes génétiques en fin d'optimisation rendront utile la définition d'un opérateur de mutation, sorte de méthode de descente locale.

La validité des idées développées dans ce chapitre sera testé sur le même problème d'optimisation de la section droite d'une barre pour maximiser son moment d'inertie (voir Maday-Patera [21]). En effet, les comparaisons expérimentales effectuées nécessitent de très nombreux calculs de fonction performance. Or l'évaluation de la fonction performance pour ce problème est informatiquement parlant plus simple que celui des problèmes d'optimisation de structures élastiques, tels celui de la plaques cantilever que nous verrons dans la section 6. Ce qui nous permettra de faire plus simplement des comparaisons sur les différents paramètres qui gouvernent les algorithmes génétiques.

Ce problème test est défini dans la section 5.1. Il nous permettra de préciser les difficultés rencontrés lors du décodage, ou passage de l'espace génotypique (le tableau de bits) à l'espace phénotypique (ou est évaluée une forme). Nous présenterons ensuite dans la section 5.2 la représentation des formes sur une grille régulière nous permettant de définir l'espace génotypique, ou encore l'espace de recherche pour l'algorithme génétique. Dans la section 5.4 nous discutons à priori, puis montrons expérimentalement la faible performance des opérateurs de recombinaison chaînes de bits classiques, ce qui nous conduira à définir des opérateurs plus adaptés. De même, on définira également deux sortes d'opérateurs de mutation (section 5.5). Une stratégie mixte concernant un changement d'opérateur durant l'évolution va se dégager, et une discussion concernant les résultats suivra.

5.1 Choix du problème

La difficulté majeure posée par l'utilisation des méthodes d'optimisation stochastique en optimisation de structure réside dans les calculs de la performance (ou fonction coût). En effet, un seul calcul de la fonction performance pour les problèmes d'optimisation de structures en élasticité présentés au chapitre 6 nécessite une analyse par élément fini de la forme (voir également [37, 15, 41]). Or un calcul par éléments finis est coûteux (entre 0.2 et 2 secondes par analyse pour une station HP haut de gamme (94), même avec une discrétisation est grossière). C'est pour cela que, pour les tests et comparaisons que nous souhaitons faire, nous avons choisis un problème test très simple : l'optimisation de la section droite d'une barre pour maximiser son moment d'inertie suivant une direction donnée tout en minimisant le poids.
Ce problème a été introduit est utilisé pour les mêmes raisons dans [21] pour des méthodes basée sur le recuit simulé. Ce choix est également motivé par l'étude théorique menée dans [21] et que nous avons rappelé dans le chapitre précédent (4). Ceci nous permet de nous placer dans les bons espaces d'existence et d'approximation de solutions.

5.1.1 Le problème test

La fonction que l'on cherche à maximiser, pour étudier la résistance de la barre à un moment exercée, est le rapport entre le moment d'inertie est le poids de la barre (ou encore l'aire de la section droite).

Les Hypothèses

Les hypothèse mécaniques sont les suivantes:

- la section droite de la barre est constante.
- On ne spécifie pas comme les moments sont appliqués à la barre.
- Le domaine de travail est le carré unité $[0,1] \times [0,1]$, et la forme "*parfaite*" solution du problème est la forme en "H".
- Le moment d'inertie de la barre par rapport à la direction x est donnée par :

$$I_{yy} = \int_S (y - y_G)^2 dxdy$$

S en (x,y) est la section droite de la barre.
(x_G, y_G) représente les coordonnés du centre de gravité G de la section droite S.
avec

$$(x_G, y_G) = \frac{\int_S (x,y)dxdy}{\int_S dxdy}$$

Les solutions de ce problème sont les barres de section en "H" parfaits (la position exacte de la barre du "H" étant indifférente).

5.2 Génotypes et phénotypes

5.2.1 Le génotype

Que ce soit pour se placer dans le cadre de l'étude théorique d'approximation de [21], où dans le cadre des simulations numériques utilisées dans la plupart des problèmes d'optimisation de formes en mécanique, la recherche de la forme optimale passe par une discrétisation du domaine de travail, ou du moins de chacune des formes considérées durant le processus d'optimisation. D'autre part, l'utilisation d'une discrétisation différente pour chaque forme considérée introduirait un bruit numérique dù au remaillage qui pourrait, en fin d'optimisation, oblitérer totalement les différences de comportement entre deux formes voisines, rendant non significatifs les résultats de l'optimisation.

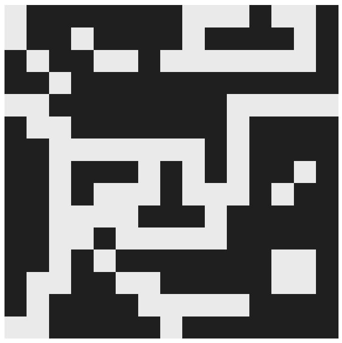
Une représentation "naturelle" d'une forme contenue dans un domaine de travail régulier passe donc par la discrétisation de ce domaine, puis par l'attribution à chaque élément de cette discrétisation d'une valeur matière ou solide, i.e. 0 ou 1. Or il se trouve que cette représentation semble également entrer parfaitement dans le cadre historiques des algorithmes génétiques [32, 22]. Ceci explique que tous les travaux antérieurs utilisant les méthodes stochastiques pour les mêmes types de problèmes ([31, 21, 37, 15, 41]) aient utilisé cette représentation. Nous verrons cependant (section 5.3 que cette apparence simplicité de la représentation cache en fait quelques pièges.

5.2.2 Du génotype au phénotype

Le décodage semble lui aussi tout à fait simple.

```
100000001110110
100100001000010
010011011111110
001000000000000
110000000011111
011000000010000
001111111010000
001000101010010
001011101110100
001111000100000
001010000000110
011001100000110
010000111110000
110000010000000
```

(a)



(b)

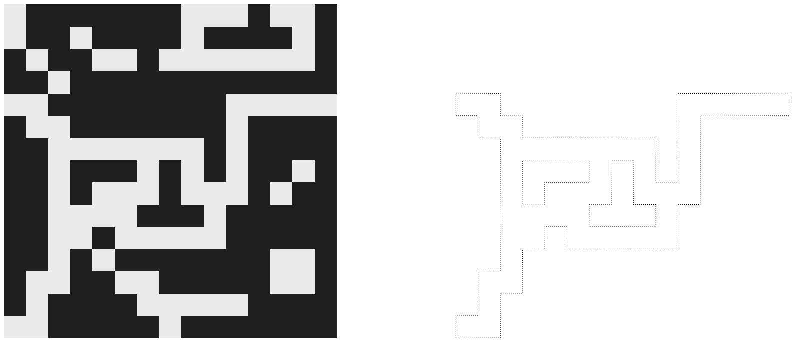
Figure 37 : *a* - Le génotype en tableau de bits. *b* - Le phénotype.
Représentation d'une forme : l'AG s'applique sur le tableau de bits (a) la performance est calculée à partir du phénotype (b).

La figure (37-a) donne, pour un maillage 15×15, le tableau de bits et la forme correspondante. Ce tableau est constitué de "0" ou de "1" suivant que l'élément correspondant est de la matière ou du vide. Ce qui est représenté dans le phénotype (37-b) le gris symbolisant la matière et, le noir le vide.
Cette simple étape de décodage pose en fait des problèmes du point de vue mécanique au niveau de la définition d'une structure mécaniquement cohérente.

5.2.3 Du phénotype à la structure

Déterminer la composante principale

Une fois le phénotype (une forme du domaine de travail) obtenu, on détermine tout d'abord la composante connexe principale de la forme en question(38-b). Puis on recherche les trous (vide inclus dans la composante principale).
La composante trouvée constitue notre structure et l'analyse mécanique sera faite sur elle.
Le choix de la méthode de détermination de la connection est basé sur des considérations mécaniques.



(a) (b)

Figure 38 : *a* - Le phénotype. *b* - La vraie structure.
la performance est calculée à partir du phénotype (a), mais le moment d'inertie dépend uniquement de la composante connectée (b)

Analyse géométrique

Nous avons choisi de considérer comme éléments connectés deux éléments qui ont une face commune, deux éléments n'ayant qu'un coin en commun sont considérés comme disconnectés (voir figure 39)

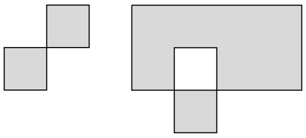


Figure 39 : *exemple d'éléments non connectés*

Ce choix nous permet d'obtenir des formes a priori plus stables, qui peuvent par conséquent supporter un moment de flexion où de torsion. Cela évite l'apparition de singularités dans le modèle mécanique aux "coins de liaisons".

5.2.4 La fonction performance

Calculer la performance revient à maximiser le rapport $\frac{Resitance}{Poids}$ [15] et ceci indépendamment de l'amplitude de l'effort de flexion exercé sur la barre.

Le moment d'inertie de la masse de la section droite est pris comme mesure de la résistance tandis que la surface de la matière connectée représente le poids. En résumé, la performance d'une forme peut être donnée par:

$$F = \frac{I_{xx}}{Poids_{connecte}}$$

I_{xx} représente le moment d'inertie de la barre, et $Poids$ est le poids de la section droite. Il reste cependant ici deux problèmes importants :

- Pour qu'une forme soit viable, c'est-à-dire que les contraintes mécaniques du problèmes soient respectées, il faut que les deux bords verticaux de la section droite soient connectés.
- Seul la matière réellement reliée aux deux bords verticaux rentre en compte dans le calcul du moment d'inertie. Les parties non connectées doivent être enlevées avant le calcul du moment voir (38-b).

Pour remplir la première condition la performance d'une forme qui ne relie pas les deux bords est mis à 0 (c'est la "death penalty" voir Baeck [6]).
Par contre, il y a plusieurs façons de résoudre le second problème comme nous allons le voir maintenant.

5.2.5 Composantes non connectées

Une fois la structure principale déterminées il reste comme nous l'avons vu sur la figure 38(a) des composante non reliées à la composante principale. Cette matière restante peut jouer un rôle sur l'émergence de nouvelles structures. Par contre, la convergence de l'algorithme doit aboutir à des structures débarrassées de cette matière "inutile". Nous avons considéré 3 façons de prendre en compte cette matière.

La réparation

Cette opération consiste à enlever, à la fois du génotype et du phénotype les composantes ou parties ne rentrant pas en compte dans la structure principale. Mais, cette méthode conduit très vite l'algorithme sur un optimum local car privé d'un matériel génétique intéressant les individus vont très vite converger vers le moins mauvais. Quelques résultats expérimentaux (figure 40) confirment confirment ce point de vue.

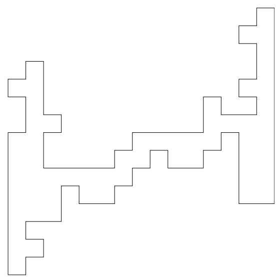


Figure 40 : *Optimum local obtenu par la réparation. Aire = 2.36 Inertie = 2.941562*

5.2.6 La réparation "provisoire"

Dans ce cas de figure, on garde le génotype tel quel (les parties non connectées ne sont pas enlevées) mais, par contre pour évaluer la performance, on considère uniquement la composante connectée : les composantes inutiles sont ignorées dans le calcul de la performance. Ce processus nous permet de garder le matériel génétiquement intéressant (contrairement au cas précédent). Par contre, cette méthode introduit une dégénérescence dans la représentation (plusieurs génotypes éventuellement très différents correspondant à un même phénotype). En d'autres termes, deux individus complètement différents peuvent avoir la même performance. La dégénérescence a en général des effets néfastes sur la convergence de l'algorithme [54], comme le confirment nos résultats numériques (figure 40). L'AG a beaucoup de mal à converger au milieu de tous ces plateaux de génotypes équivalents en performance.

La pénalisation de la performance

La troisième méthode utilise une modification de la fonction performance pour favoriser la disparition progressive de la matière non connectée.
On calcule d'abord le moment d'inertie sans tenir compte des parties non reliées. On introduit ensuite un paramètre de pénalisation relativement à la surface non connectée. Pour cette troisième méthode la performance est donc donnée par :

$$F = \frac{I_{xx}}{poids_{connecte} + \alpha poids_{nonconnecte}}$$

I_{xx} représente toujours le moment d'inertie de la barre, $poids_{connecte}$ est l'aire de la section droite, $poids_{nonconnecte}$ est l'aire restante, et α est le paramètre de pénalisation.
Nous avons essayé d'ajuster le facteur de pénalisation α de façon à ce que la forme de la barre ne soit pas trop modifiée tout en diminuant sensiblement la surface en trop. Trouver le bon α n'est pas aisé car :

- α trop grand nous enlèvera bien sûr très vite toute la surface en trop mais pénalisera aussi le rapport $Moment/Aire$.
- De même α très petit perd son utilité car on aura un bon rapport $Moment/Aire$ mais la surface en trop ne sera que très peu pénalisée. Ce problème sera détaillé sur l'application Optimum Design (voir section 6.4.2)

Nous venons de voir le rapport entre le codage et la performance, et la nécessité de prise en compte des phénomènes mécaniques lors de la définition de la performance (ceci sera à nouveau montré au chapitre 7 sur des problèmes d'optimisation de structures élastiques). Nous allons dans le paragraphe suivant étudier la relation entre le codage et les opérateurs génétiques que nous allons utiliser dans les chapitres suivants.

5.3 Chaînes de bits vs matrice de bits

A priori, il existe une équivalence formelle entre le génotype défini au paragraphe précédent (voir figure 37) représenté par un tableau de bits, et une chaîne de bits (voir par exemple la représentation machine d'une telle matrice). Seulement, nous allons voir que ces deux types de représentation sont loin d'être équivalentes quand on utilise les opérateurs de croisement standard de la théorie des algorithmes génétiques.

Cette section est consacrée à la démonstration de la proposition suivante:

$$100001100001111111100001100001100001 \overset{Different}{\neq} \begin{array}{|c|c|c|c|c|c|} \hline 1 & 0 & 0 & 0 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 & 0 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 0 & 0 & 0 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 & 0 & 1 \\ \hline \end{array}$$

représentation en chaînes de bits   représentation en tableau de bits

Nous allons mettre en évidence, tout d'abord en utilisant l'analyse des schémas (voir chapitre 2), puis par des expérimentations numériques, cette différence entre ces deux représentation.

5.3.1 L'analyse des schémas

Supposons que l'on utilise un codage en chaînes de bits à la place d'un codage en tableau de bits. Si nous considérons l'opérateur de croisement on remarque après l'analyse des schéma que ce codage induit le problème suivant :
Soit H un schéma 2D correspondant à une barre verticale de longueur L. Dans une représentation bit-string classique H peut être vue comme un chaîne de longueur $L \times M$ (si le tableau a pour dimension $N \times M$). Il s'en suit que H sera beaucoup plus perturbé par les croisement 1D à 1-point où 2-points que ne le serait un schéma correspondant cette fois ci à une barre horizontale de même longueur.
Plus généralement si l'on utilise des tableau de bits pour des chaînes de bits, les croisement à n points introduisent un fort biais contre l'émergence de briques élémentaires verticales ([22]).
Pour nous fixer les idées prenons l'exemple de deux schéma a et b.

Exemple 5.3.1

```
1..............    ..............1
1..............    1..............
1..............    ...............
...............    ...............
...............    ...............
```

(a) (b)

Figure 41 : Différences entre des instances de longueur définies en 1 ou 2D

Le schéma (a) a une taille de 3 en 2D et de 21 en 1D. L'émergence de ce schéma qui peut être très intéressant pour un problème 2D ne signifie rien pour le codage bits string

est à toutes les chance d'être perturbés par les croisement 1D classiques.
Par contre, le schéma (b) a une taille de 15 en 2D et une longueur de 2 en 1D. Bien qu'il ne présente aucun intérêt pour un problème 2D, il a toutes les chances d'être préservé par les croisements 1D classiques.

Quantifier le biais introduit par l'utilisation des croisements de type n points reste à faire. Il faudrait par exemple étudier la variance des performances sur les schémas, comme fonction de la longueur des schémas (dans la suite de ce qui a été fait dans [54] pour évaluer divers représentation du problème du voyageur de commerce).
Néanmoins, l'inefficacité (prévisible) des opérateurs de croisements à n points (confirmé à posteriori par les résultats numériques de la section 5.6). nous conduit à introduire des opérateurs spécifiques plus adaptés a notre problème 2D.

5.4 Opérateurs de croisement

Nous définissions dans cette sections des opérateurs adaptés au problème de l'optimisation de formes, et montrons leur utilité *a posteriori* par des expérimentations numériques comparatives.

5.4.1 Le croisement

Nous commençons par étudier les opérateurs de croisement pour la représentation introduite dans la section précédente.
Dans toute la suite de ce paragraphe, pour illustrer les divers opérateurs définis et utilisés, nous considérons toujours le même un couple symbolique de parents (voir figure 42).
Dans la représentation des enfants issus de ce couple, la couleur sert uniquement à montrer quel matériel génétique (bits) vient de quel parents, indépendamment de la valeur 0/1 (vide/matière) des bits.

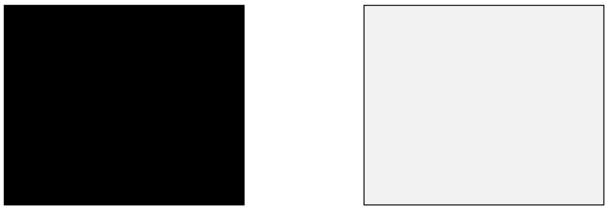


Figure 42 : *Les parents*

Les opérateurs mono-dimensionnels

Pour évaluer le biais introduit par l'utilisation d'un tableau de bits en tant que chaînes de bits, nous avons d'abord utilisé des opérateurs de croisement standard. C'est-à-dire ceux définis dans l'algorithme génétique de base, introduits dans le chapître 3. Ces opérateurs seront appelés mono-dimensionnels. Nous avons considéré les trois types d'opérateurs de croisement 1D suivant.

- *Le croisement à un point*
  La forme est représentée par une chaîne de "0" et de "1", un point est aléatoirement choisi sur cette chaîne, c'est le point de coupure, les enfants sont obtenus en échangeant les deux parties droites des parents (voir 43).

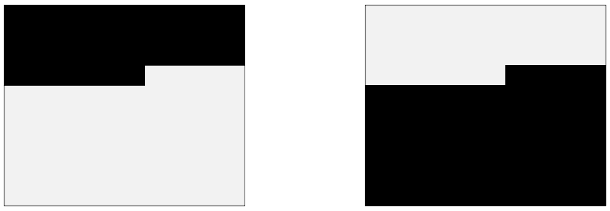


Figure 43 : *exemple d'un croisement un point 1D*

- *Le croisement à deux points*
  Deux points sont aléatoirement choisis sur la chaîne, on permute les deux parties comprises entre ces 2 points chez les parents. Les enfants sont le résultat de cette opération(voir Figure 44).

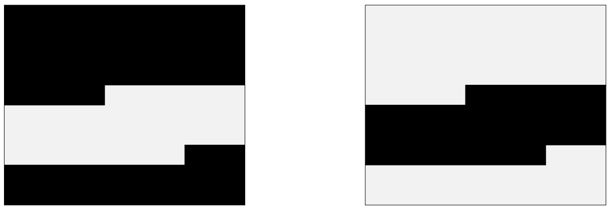


Figure 44 : *exemple d'un croisement deux point 1D*

- *Le croisement uniforme*
  Chaque bit d'un enfant est construit en choisissant aléatoirement entre les deux bits correspondants des parents. Les choix pour des positions différentes sont indépendants (voir figure 45).

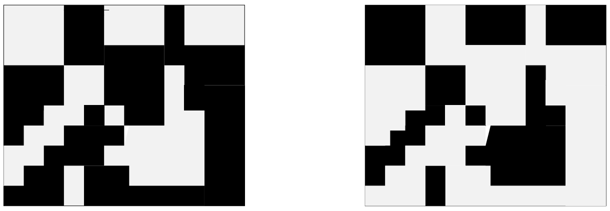


Figure 45 : *les enfants par croisement uniforme*

Comme on l'a déjà vu dans la section (précédente 5.3.1) les croisements à un où 2 points induisent un biais géométrique: seules des bandes horizontales sont échangées. Par contre, le croisement uniforme qui lui ne souffre pas de ce biais, il détruit (ou respecte!) tous les schémas indépendamment de la géométrie.

Le croisement bidimensionnel

Pour palier à la limite des opérateurs 1D, nous avons utilisé trois formes de croisement 2D.

- *Le croisement diagonal*
  L'idée de base de ce croisement est de généraliser en dimension 2 le très populaire croisement à 1-point dans le cas d'une dimension.
  On choisit aléatoirement deux points distincts qui seront situés au même endroit sur les deux parents. Ces deux points vont générer une droite (ou diagonale), Les enfants sont obtenus en permutant les deux parties correspondantes des deux parents (figure 46).

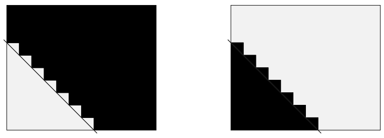


Figure 46 : *Le croisement diagonale*

- *Le croisement bidiagonal*
  Ce croisement est une variante du précédent. Deux droites qui seront les mêmes sur les deux parents sont trouvées aléatoirement. Les enfants sont obtenus en permutant les parties comprises entre les deux droites chez les deux parents(voir figure 47).

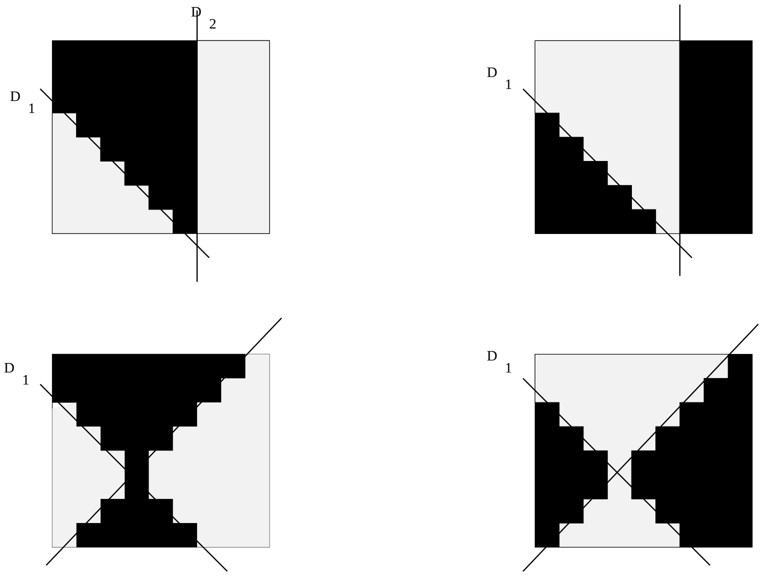


Figure 47 : *Deux exemples de croisement bidiagonale*

- *Le croisement par blocs*
  Deux lignes horizontales h1 et h2 et deux lignes verticales v1 et v2, sont aléatoirement générées ce qui subdivise chaque parent en 9 cases. On cherche ensuite aléatoirement le nombre et la position des cases à permuter chez les parents, cette méthode a été utilisée par Jensen [37] et a depuis été généralisée à la dimension N par Moon [40].

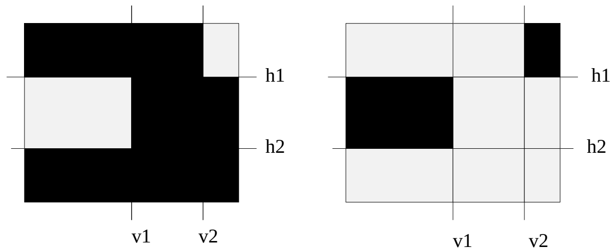


Figure 48 : *Exemple de croisement 2-blocks*

5.5 Les opérateurs de mutation

On a utilisé 3 types de mutation :

5.5.1 La mutation Classique

C'est la mutation standard des algorithmes génétiques: chaque individu a une probabilité de muter, et, pour chaque individu appelé à muter, chaque bit a une certaine probabilité de muter. Cette mutation n'introduit aucun biais géométrique particulier.

5.5.2 La mutation générale

C'est une mutation introduite pour garder une certaine diversité dans la population, et empêcher ainsi une convergence prématurée de l'algorithme. Les opérateurs de mutation usuels ne tiennent pas compte de l'uniformité éventuelle de la population dans certaines zones des formes considérées. Nous avons repris une mutation dite *générale* [11], c'est-à-dire prenant en compte les caractéristiques de la population dans son ensemble. Pour ce type de mutation chaque bit a une certaine probabilité de muter, et cette probabilité est fonction de la variance des valeurs de tous les bits à la même position dans toute la population. Le processus est le suivant :
Au début de chaque génération la diversité des valeurs des bits à chaque position de la population est déterminée, et stockée dans une *matrice de valeurs moyennes* (D_i).
$D(i)$, la valeur moyenne des bits à la position i, est calculée comme suit:

$$D(i) = \frac{1}{N} \sum_{j=1}^{j=N} bit_i(j) \tag{5.5.1}$$

$bit_i(j)$ est la valeur du ième bit dans le jème individu de la population de taille N.
Une diversité de 0.0 (resp. 1) signifie que tous les bits de la population à cette position valent 0 (resp. 1), tandis qu'une diversité de 0.5 signifie que la moitié des bits est à 1 et l'autre moitié à 0.
Plus faible est la diversité à un endroit dans une population, plus grande doit être la probabilité de mutation à cet endroit : cet opérateur impose de fort taux de mutation aux endroits qui ont déjà convergé.

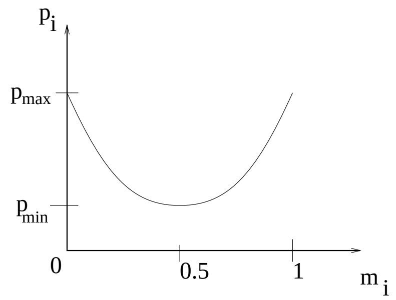


Figure 49 : *probabilité de mutation*

La probabilité de mutation est donc contrôlé par 2 paramètres: $Pmin$ est le taux de mutation minimal, $Pmax$ le taux de mutation maximal.

La probabilité de mutation de chaque position est donnée par une parabole qui passe par $Pmax$ quand la diversité vaut 0.0 ou 1.0, et qui prend une valeur minimal $Pmin$ quand la diversité est de 0.5 (voir figure 49):

$$Pmut(i) = 4.0\left[(Pmax - Pmin)D(i)^2 - (Pmax - Pmin)D(i)\right] + Pmax. \qquad (5.5.2)$$

Il faut noter que l'utilisation de cette méthode introduit deux paramètres (à choisir par l'utilisateur) au lieu du paramètre habituel. Cette méthode nous permet d'éviter une convergence prématurée de l'algorithme ce qui n'est pas négligeable, mais elle ralentit aussi la convergence :il faut trouver un compromis.

5.5.3 La mutation des composantes

Ce troisième opérateur de mutation est un opérateur local conçu spécialement pour les problèmes d'optimisation de formes, permettant des petites modifications localisée sur le contour de la forme. Cette approche utilise la topologie trouvée. L'idée de base de cette mutation est que, pour une topologie donnée, c'est-à-dire pour un nombre donné de trous dans la forme il doit être aisé de trouver la forme optimale pour cette topologie en bougeant légèrement le contour. Nous utiliserons principalement cette mutation en fin d'évolution c'est-à-dire une fois que les caractéristiques grossières de la forme sont trouvées nous avons besoin de raffiner la forme en enlevant ou en rajoutant des pixels. Concrètement, les pixels entourant la topologie ont une plus grande probabilité de muter que les autres pixels. Ce qui nous permet d'accélérer la convergence en fin d' évolution (voir les résultats de la section 7.5).

5.6 Résultats comparatifs

5.6.1 Les conditions expérimentales

Pour tous les tests effectués dans cette thèse, l'algorithme génétique utilisé (CMAPX) utilise un schéma de générations. Les paramètres sont les suivants :

- Type de sélection La sélection utilisée est la roue de loterie avec une mise à l'échelle linéaire pour atteindre une pression sélective de 2.0.
- Taille de la population La taille de la population notée Gen est de 90 sauf mention contraire.
- Probabilité de croisement et de mutation On verra plus loin les valeurs utilisées pour les probabilités de croisement et de mutation (p_{mut}, p_{cross}).
- Autres Notations
  - $Pcross$: la probabilité de croisement pour un couple donné.
  - $Pmut_{forme}$: la probabilité pour qu'une forme mute .

  - $Pmut_{pix}$:la probabilité pour qu'un bit d'une forme donnée mute (mutation standard).

  - $Pmut_{max}$ et $Pmut_{min}$: les probabilités maximale et minimale pour qu'un bit d'une forme donnée mute (mutation générale).

- Test d'arrêt L'algorithme s'arrête après 100 générations sans aucune amélioration sur la meilleure performance obtenue, ou encore après 2000 générations.

- Validation

  - Tous les tests sont effectués sur des tableaux de bits de 15×15 décrits ci dessous. Tous les résultats présentés sont des moyennes de 20 tests indépendants (20 points de départ différents du générateur de nombres pseudo-aléatoires).
  - Le problème test présenté dans la section (5.1.1) est très simple, presque tous les tests ont aboutit à une forme très proche de la forme en "H" attendu. Il n'est donc pas intéressant comparer le taux de succès (optimum atteint). Pour la comparaison des différents opérateurs nous chercherons plutôt à mesurer la vitesse de convergence c'est-à-dire que nous donnerons les courbes montrant la progression de la moyenne sur les 20 essais des meilleures performances en fonction des générations.

5.6.2 Le problème

Reprenons le problème test que nous avons assez brièvement introduit dans les paragraphes précédents. On souhaite optimiser le moment d'inertie d'une poutre soumis à des efforts de flexion. On veut donc maximiser le moment d'inertie tout en minimiser le poids de la section droite de la poutre. On se fixe comme hypothèses que :

- Le domaine de travail est le carré $[0, 1] \times [0, 1]$ subdivisé en un tableau de 15×15 éléments qui sont soit de la matière soit du vide.

- on suppose que la section droite est constante pour toute la poutre.

- On ne spécifie pas comment les moments sont appliqués à la poutre.

Le résultat attendu est de la forme (figure 50) avec une performance de $1,54008$

Figure 50 : *forme de la section droite en "H" attendu*

Il y a donc 15 solutions possibles suivant la position de la barre horizontale.

5.6.3 Comparaison des divers opérateurs de croisement

En l'absence de toute mutation nous avons voulu voir l'influence du type de croisement sur l'évolution des performances.

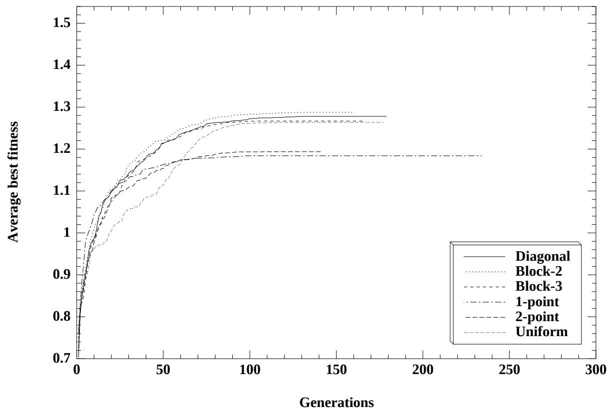


Figure 51 : *Les opérateurs de croisement*

Une première remarque sur les mauvais résultats général dû à l'utilisation du seul opérateur de croisement : la forme optimale est rarement atteinte. La population devient très vite homogène ce qui explique que toutes les courbes de la figure (51) s'arrêtent brutalement entre les générations 150 et 250 bien que l'on ait autorisé 2000 générations. Ceci se porte en faux par rapport aux théorie historiques des AGs qui considéraient la mutation comme un opérateur mineur.

Mauvaises performance pour les opérateurs 1D

On remarque que les opérateurs 1D classiques (1-point et 2-points) sont distancés aussi bien par le croisement uniforme que par les opérateurs 2D. Pour les opérateurs 2D Les résultats obtenus sont effectivement ceux escomptés d'après le théorème des schémas (voir la section 5.3.1), et confirmé par l'étude **tableau de bits $\neq$ chaîne de bits**

Comportement du croisement uniforme

La deuxième remarque concerne la bonne performance du croisement uniforme.
En regardant plus précisément la courbe on se rend compte que ce croisement qui au départ débute mal, rattrape vite les autres “bons” croisements dans les dernières générations. Ce phénomène est observé dans tous les autres résultats voir (section 6.3).

Au début de l'évolution, le croisement uniforme perturbe de la même manière tous les schémas quelque soit leur ordre. Tandis que les autres croisements même les 1D préservent des parties assez importantes de l'espace.

Mais, au fur et à mesure que la population converge et que des briques de base de plus en plus grandes sont construites ces effets perturbatrices disparaissent, ce croisement ne perturbe plus les schémas car le croisement de parties identiques chez deux parents ne changent rien chez les enfants quelque soit le type de croisement utilisé. Aucun croisement ne peut perturber une partie qui a déjà totalement convergé. Mais par contre, le croisement uniforme permet de garder une certaine diversité pour les autres parties. Et, donc à la fin on a d'aussi bons résultats que pour les croisements spécifiques 2D.

5.6.4 Comparaison des divers opérateurs de mutation

Dans ce paragraphe on suppose maintenant que la probabilité de croisement est nulle, et que l'on utilise le seul opérateur de mutation. Nous allons comparer les trois types de mutation définies dans la section 5.5

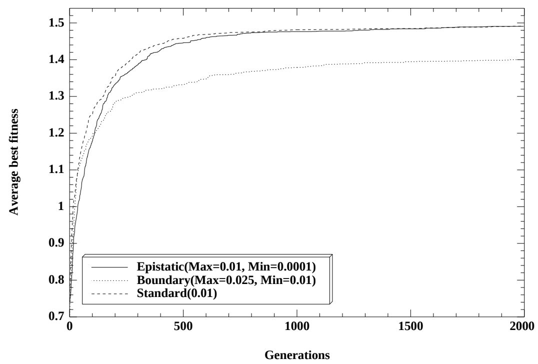


Figure 52 : *Comparaison de la mutation standard, générale et composante*

en considérant la figure 52 on remarque :

- Les résultats obtenus sont bien meilleurs que ceux obtenus par simple croisement et sans la mutation (voir figure 51) ce qui montre encore une fois le rôle de la mutation (ces résultats sont confirmés par les résultats en élasticité (chapitre 7)
- Comme l'on s'y attendait, la mutation du contour est le moins bon. Ce qui s'explique tout simplement car cet opérateur est un opérateur d'optimisation local et favorise donc une convergence prématurée. Toutefois, quand on regarde plus précisément on

se rend compte que cette mutation trouve exactement la solution "optimale" quand il trouve le bon optimum local, et ceci, plus vite que les deux autres mutations qui ont beaucoup de mal à ajuster les derniers pixels. Ce résultat sera confirmé pour le problème du vélo que nous verrons dans les problèmes de chargements multiples (voir section 7.5). Ce qui suggère que la mutation des composantes doit être principalement utilisée en fin d'évolution. Son intérêt réside surtout dans l'affinage des composantes. Appliqué avant la convergence cet opérateur perturbe la recherche du meilleur.

- En l'absence de tout croisement, il n'existe aucune différence notable entre la mutation standard et la générale si l'on considère les meilleurs résultats obtenues en 2000 générations. Ils atteignent respectivement les performances moyennes (et les écarts type) de 1.49083(0.00772) et 1.4913(0.01523). De plus, la mutation générale ralentit visiblement la convergence en tout début d'évolution c'est effectivement le résultat escompté.

5.6.5 Meilleurs résultats

Les résultats présentés jusqu'ici ont utilisé un seul opérateur génétique c'est-à-dire soit le croisement soit la mutation. Les résultats que nous présentons dans cette section par contre concernent à la fois ces deux opérateurs. Nous considérons une mutation et un croisement donnés avec une certaine probabilité pour chaque évolution. Vu les résultats obtenus aux paragraphe précédents (section 5.6.3 5.6.4), nous ne considérerons pas ici les opérateurs de croisement 1D (1-point et 2-points) pas plus que la mutation des composantes.

Gen	*pop*	standard P_m	générale (p_{max}, p_{min})
2000	90	0.001	(0.005,0.0001)

Table 5.6.5 Les paramètres

	mutation standard	mutation générale
uniforme	1.48805 (0.00784)	1.50936 (0.00702)
diagonal	1.49773 (0.00869)	1.49499 (0.00912)
2-block	1.49203 (0.01138)	1.50914 (0.00872)
3-block	1.50352 (0.01123)	1.5197 (0.0059)

Table 5.6.5 Moyenne des valeurs finales de la performance (écart type) sur 20 tests.

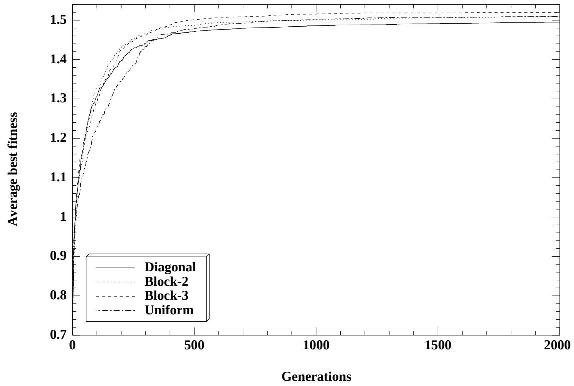


Figure 53 : *Mutation - Croisements 2D*

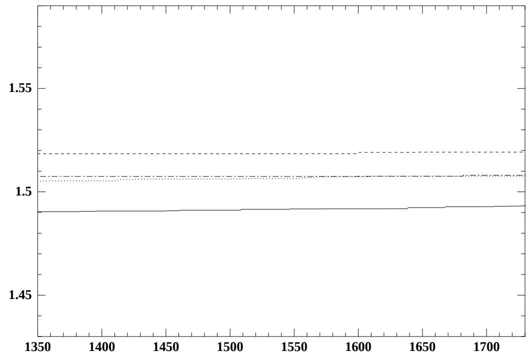


Figure 54 : *meilleure performance moyenne sur 20 tests, zoom sur la fin d'évolution*

Le tableau 5.6.5 présente les meilleurs résultats obtenus pour le problème test. Les probabilités de croisement et de mutation sont consignées sur le tableau 5.6.5. Ils ont été choisis après de nombreux tests. Dans presque tous les tests, les résultats obtenus par la mutation dite générale sont sensiblement meilleurs que ceux obtenus par la mutation classique. Le croisement à 3-blocs de son coté semble être le meilleur des croisements utilisés. De plus, l'association de ces deux opérateurs ne donne pas seulement les meilleurs résultats mais le fait aussi avec la plus petite déviation. La figure 53 montre les résultats obtenus pour la mutation générale suivant les opérateurs de croisement utilisés. Une fois de plus on remarque le mauvais début pris par le croisement uniforme avant de rattraper les autres opérateurs comme pour les tests de la section précédente (section 5.6.3). Ceci est confirmé par tous les tests quelque soit l'opérateur de mutation ou le taux de mutation appliqué.

A ce point tous les résultats suggèrent l'utilisation d'une stratégie mixte. Le croisement 3-blocs couplé avec la mutation générale semble être le meilleur choix. Mais, la bonne performance du croisement uniforme en fin d'évolution montre qu'il est judicieux de l'utiliser plutôt en fin d' exécution. De plus, le fait que la mutation des composantes soit locale suggère d'utiliser cette mutation pour "polir" la meilleure solution trouvée dans les dernières générations.

Stratégie de changement d'opérateurs

Nos tentatives ont consisté à commencer avec les paramètres utilisés pour les courbes de la figure 52 en utilisant le croisement 3-blocs et la mutation générale, puis après un nombre donné de générations, de remplacer ces opérateurs par le croisement uniforme et la mutation locale. Les figures suivantes (55 et 56) montrent les résultats obtenus en fonction de la génération où l'on change les opérateurs. La question qui se pose naturellement dans ce cas et : **quand** peut on passer d'une stratégie à une autre ? Nous avons essayé de permuter les stratégies en fonction des générations

- Nous avons d'abord considéré un changement assez rapide.

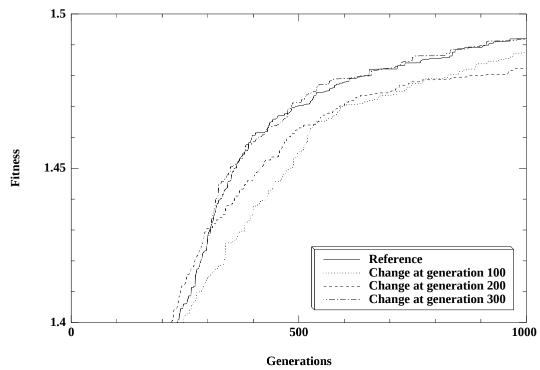


Figure 55 : *changement de stratégie avant où à 300 générations : trop rapide*

On se rend compte (voir Figure 55) qu'un changement précoce ici au bout de 100 ou 200 génération conduit à une dégradation. Les deux courbes correspondantes sont en dessous de la courbe de référence, et ne la rattrapent jamais.

- Si on change d'opérateurs à la génération 300, il ne se passe rien. Il n'y a aucun changement c'est-à-dire qu'on a ni amélioration ni dégénérescence (voir figure 55).

-

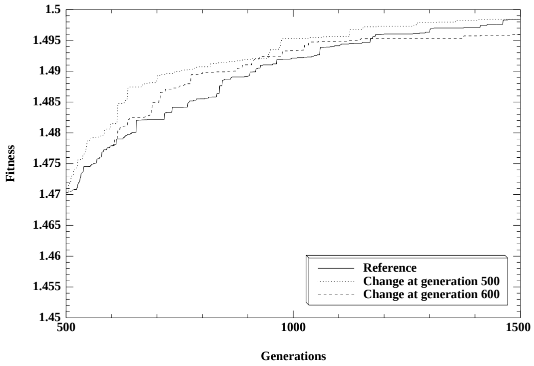


Figure 56 : *changement de stratégie entre 500 et 600 générations : amélioration passagère.*

Si par contre on attend 500 ou 600 générations avant de changer de stratégie, on remarque qu'il y a une amélioration temporaire (voir Figure 56).
Cette amélioration n'est pas significative, l'augmentation est relativement faible et la courbe de référence rattrape assez vite les meilleures courbes. Toutes les courbes finissent à peu près au même endroit. Ce qui peut s'expliquer selon Spears [63] : Le mérite proviendrait plus du changement d'opérateurs que des opérateurs eux mêmes.
Il serait cependant nécessaire de faire des tests sur des problèmes plus complexes pour voir la signification et l'interprétation des résultats ci dessus. Toutefois, des tests systématiques sont difficiles à conduire pour des raisons de temps de calcul, sur des problèmes où il est besoin de faire une étude d'éléments finis .

5.7 Conclusion

5.7.1 Sur les opérateurs

On a montré dans ce chapître la faiblesse d'une utilisation des AGs en boite noire : la représentation standard en bits strings n'est pas adapté pour représenter les phénotypes dans les problèmes 2D.
Le biais géométrique induit par les croisements à 1- ou 2-points influent négativement sur les performances. "l'isotropie" du croisement uniforme permet d'obtenir de bons résultats dans toutes les dimensions. Toutefois, les résultats obtenus par les opérateurs spécialement conçus pour le problèmes 2D ont donné les meilleurs résultats.
Deux opérateurs de mutation ont été étudiés, en plus de la mutation classique "bit-flip". La mutation générale qui tend à garder une certaine diversité le long des générations doit être utilisé avec précaution. Dans certains cas elle pourrait tout aussi bien empêcher toute convergence. Une seconde mutation a été mis en place pour permettre une optimisation de forme locale.
Les résultats numériques suggèrent que la meilleure stratégie pourrait être une stratégie mixte avec en début un croisement 2D et une mutation classique ou générale et, le croise-

ment uniforme couplé avec la mutation des composantes en fin d'exécution.
La question importante restant à savoir à quel moment on peut passer d'un opérateur à l'autre. Les premières résultats utilisant le changement après un nombre fixé de générations montrent que la solution doit être cherchée de façon évolutive.

- On pourrait par exemple coder dans les individus les opérateurs les plus adaptés pour cet individu pour imiter ce qui est fait en "stratégies évolutives" pour ajuster la mutation pour des chromosomes à valeurs réelles (voir Schwefel [61]) ou dans les AGs pour décider s'il vaut mieux appliquer un croisement 2D ou le croisement uniforme comme le suggère Spears [67].

- Une autre approche serait de faire des tests de temps à autre au fur et à mesure des générations. Ces tests permettraient de décider des opérateurs et paramètres qui seraient les plus adaptés pour les quelques prochaines générations sur la base d'une analyse statistique (voir Greffenstette [29]) ou encore en utilisant des techniques d'apprentissage comme dans [62].

5.7.2 Sur le codage

Nous n'avons utilisé qu'un seul codage au cours des travaux de cette thèse, et avons cherché à adapter les opérateurs et la fonction performance au codage plutôt que l'inverse. Nous discuterons les mérites et inconvénients de cette approche au vu des résultats sur des problèmes d'optimisation de formes en élasticité (chapitre 7).

Chapitre 6

La fonction performance

Ce chapitre est consacré à la définition de la fonction performance pour le problème de l'optimisation topologique de formes en calcul de structures. A partir du même but, obtenir la forme de poids minimal mais suffisamment "solide" pour certaines conditions d'utilisation (certains chargements), plusieurs approches sont possibles. Ainsi les méthodes d'homogénéisation minimisent la *compliance*, ou travail des forces extérieures, dont le principal intérêt est d'être différentiable. Les approches par algorithmes génétiques antérieures utilisaient une définition de la rigidité basée sur le rapport entre le poids de la structure et son déplacement maximal sous le chargement imposé. Mais un cahier des charges définissant une structure mécanique requiert le plus souvent que cette structure ait un déplacement maximal ne dépassant pas une valeur imposée sous un chargement donné. Nous avons été ainsi amenés à définir le problème de l'optimisation topologique de formes comme un problème d'optimisation sous contraintes.
Quelle que soit la performance utilisée, une étape obligatoire est la simulation du comportement mécanique de la structure sous le chargement imposé : Nous introduisons (section 6.1) les modèles classiques que nous avons utilisés (élasticité linéaire et non linéaire en grandes déformations). Nous présentons ensuite en détail dans la section 6.2 les différentes fonctions performances utilisées dans la littérature, en les discutant au vu d'expériences numériques. La fin du chapitre est consacrée à l'optimisation sous contraintes dans le cadre de l'optimisation topologique de formes. Il existe plusieurs possibilités de prise en compte des contraintes par les algorithmes génétiques. Nous présenterons une approche par pénalisation, pour laquelle le problème est la détermination des poids relatifs de la fonction objectif et des termes de pénalisation en cas de violation des contraintes. Une étude numérique nous permettra de dégager une stratégie optimale pour les problèmes futurs (dans la section 6.4.2).

Tout au long de ce chapitre, nous utiliserons pour les diverses expériences numériques le problème test de la plaque cantilever auquel nous appliquerons les résultats du chapitre précédent : représentation en tableau de bits, croisement par blocs, mutation générale, puis "composante".

6.1 Le modèle mécanique

Nous rappelons ici succinctement le cadre mécanique dans lequel nous nous plaçons, renvoyant à [16] pour une présentation détaillée et complète du modèle théorique, et à [39] pour les modèles numériques.

6.1.1 Le modèle linéaire

Sauf mention contraire, tous les tests numériques que nous avons effectués utilisent le modèle classique de l'élasticité linéaire bidimensionnelle en contraintes planes. On néglige les effets de la gravité et on se place sous l'hypothèse des petites déformations. Le problème d'élasticité peut se formuler comme suit :

soit $\Omega \subset IR^2$ un ouvert borné.

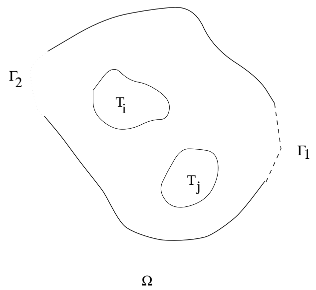


Figure 57 : *Le domaine*

On considère le problème P_Ω suivant :

$$P_\Omega \begin{cases} -div\sigma = 0 & \text{dans } \Omega \backslash T, \\ \sigma \cdot n = 0 & \text{sur } \partial\Omega \backslash (\Gamma_1 \cup \Gamma_2), \\ \sigma \cdot n = F & \text{sur } \Gamma_1, \\ u = u_0 & \text{sur } \Gamma_2. \end{cases} \tag{6.1.1}$$

où :

$$\sigma = K\epsilon \tag{6.1.2}$$

$$\epsilon(u) = \frac{1}{2}(\nabla u + \nabla u^T) \tag{6.1.3}$$

$$T = \{T_i, i \in IN\} \tag{6.1.4}$$

K est le tenseur d'élasticité (loi de Hooke), ε est le tenseur des déformations linéarisé et T est l'ensemble des trous T_i .

Lorsque l'on veut résoudre un problème d'élasticité faisant intervenir de grandes déformations le modèle linéaire décrit ci-dessus n'est plus valide.

6.1.2 Le modèle non linéaire

Nous considérons à présent le problème de l'élasticité en contraintes planes dans le contexte des grandes déformations. Le matériau obéit encore à une loi linéaire, mais on prend en compte les effets non linéaires dus à l'hypothèse des grandes déformations. Une description détaillée du modèle théorique peut être trouvée dans [16]. Le modèle EF utilisé et les détails de l'implémentation (itérations quasi-Newton sur la non-linérité) sont détaillés dans [39] [39].
Les équations considérées sont :

$$\begin{cases} -div(I+\nabla u)\sigma = 0 & \text{dans } \Omega \\ (I+\nabla u)\sigma \cdot n = 0 & \text{sur } \partial\Omega - (\Gamma_1 \cup \Gamma_2) \\ (I+\nabla u)\sigma \cdot n = F(u) & \text{sur } \Gamma_1 \\ u = u_0 & \text{sur } \Gamma_2 \end{cases} \tag{6.1.5}$$

Avec la loi de comportement $\sigma = K\epsilon$ et $\epsilon = \dfrac{(I+\nabla u)^T(I+\nabla u) - I}{2}$.
Pour nos calculs nous avons considérés une loi de comportement linéaire isotrope (loi de St Venant-Kirchoff).

Exemple de fonction F

Certaines conditions aux limites, comme la condition de pression, peuvent introduire une non-linéarité supplémentaire. Lorsque, la pression est toujours normale à la surface de la configuration déformée, la condition aux limites de pression qui s'écrivait en élasticité linéaire :

$$\sigma \cdot n = -pn \quad \text{sur } \Gamma_1.$$

devient en non linéaire :

$$(Id+\nabla u)\sigma \cdot n = -p[det(Id+\nabla u)](Id+\nabla u)^{-1} \cdot n \quad \text{sur } \Gamma_1$$

6.1.3 Le problème test

Dans le cadre théorique défini ci-dessus, nous détaillons maintenant les étapes du calcul de performance. Nous illustrerons note démarche sur le problème test de la plaque cantilever présenté chapitre 1. Nous le rappelons ici pour des raisons de complétude et donnons également les paramètres utilisés, choisis d'après les résultats des chapitres précédents.

Considérons un domaine de travail discrétisé en 10×20 pixels constitués soit de matière soit de vide (figure 1).On suppose que la plaque a son côté gauche encastrée, le point d'application de la force est le point (10,10).
La forme supposée optimale est la plaque en “>” (figure 58). Cette supposition est basée sur les résultats de la méthode d'homogénéisation décrite paragraphe 1.1.3 dont on trouvera une discussion détaillée paragraphe 6.3.2.

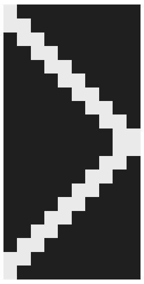


Figure 58 : *La forme attendue*

Pour tous les tests de ce chapitre et du chapitre suivant nous utilisons :
le croisement par blocs appliqué avec un taux de 0.6 (section 5.4.1).
la mutation générale avec une probabilité ($P_{Max} = 0.1, P_{Min} \in \{0.05, 0.005\}$) P_{max} et P_{Min} étant les probabilités maximale et minimale de transformer un pixel (section 5.5.2).

Les valeurs de ces paramètres ont été choisis après de nombreuses expériences numériques.

6.2 Calcul de la fonction performance

La fonction performance doit quantifier la valeur d'une forme donnée par rapport aux objectifs de poids minimal et de "solidité". Elle met donc en œuvre, outre le poids de la structure, immédiat à obtenir par analyse géométrique, son comportement mécanique sous un chargement donné. Ce comportement peut être simulé par une méthode d'éléments finis. Toutefois, cette approche par éléments finis ne pourra être effectuée que sur les formes statiquement stables et là encore une analyse géométrique préalable est nécessaire. Les résultats de cette simulation et le poids de la structure servent conjointement à quantifier la valeur de la forme

Les deux principales étapes du calcul de la fonction performance, sont résumés dans la figure 59.

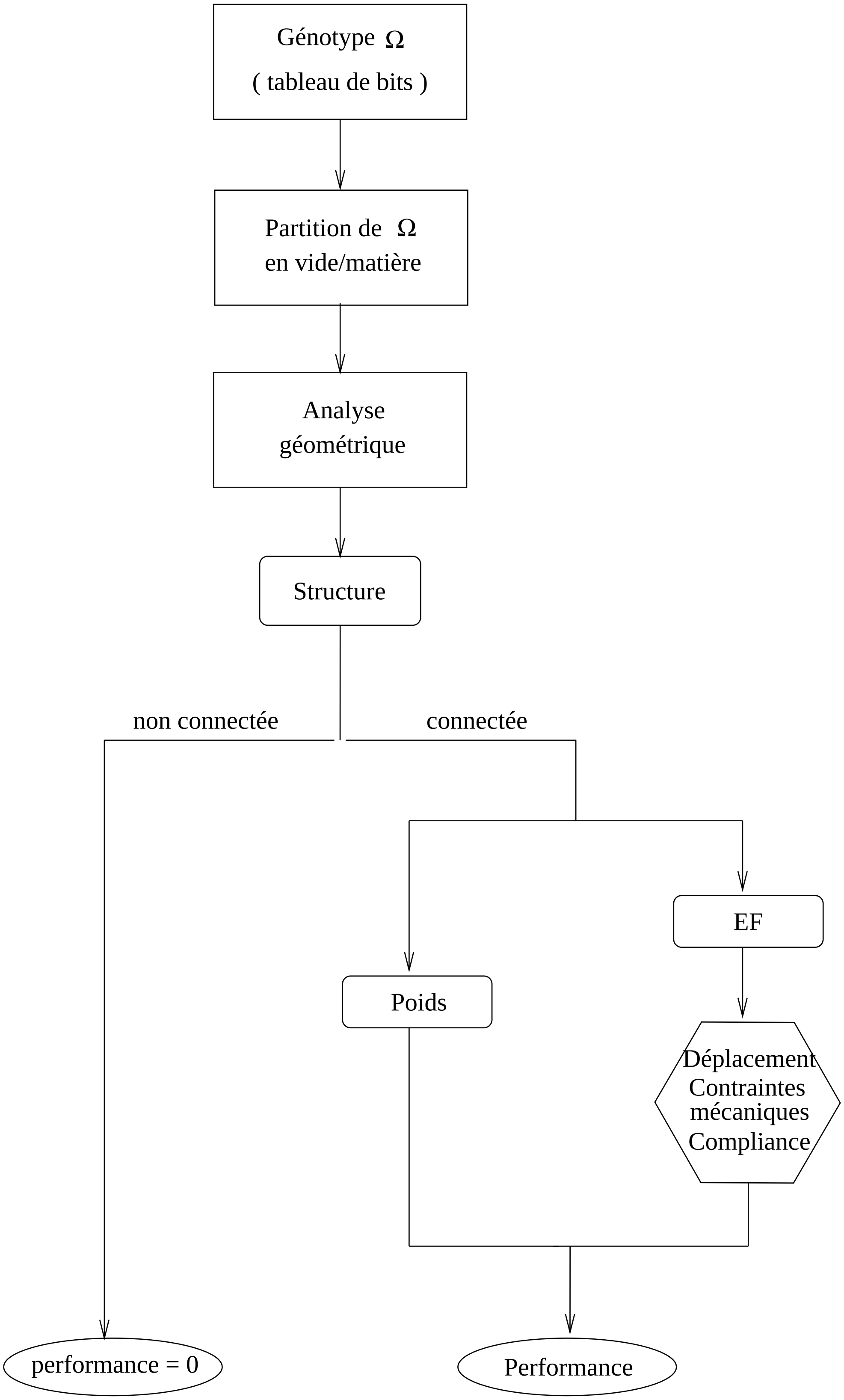


Figure 59 : *Décodage et calcul de la performance.*

6.2.1 Analyse géométrique

Considérons le problème de la plaque cantilever représenté figure 60. Le chargement est appliqué au milieu du bord droit de la plaque, le bord gauche étant fixé.

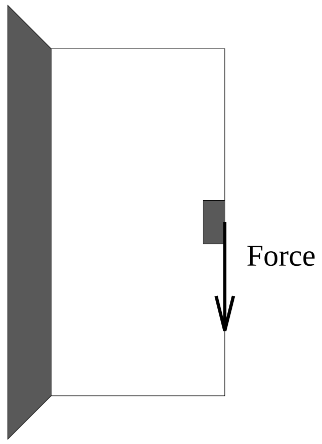


Figure 60 : *Plaque encastrée*

On impose l'existence de matière aux points où sont appliqués les chargements. Il faut ensuite que ces points de chargement soient connectés au bord fixe par de la matière pour que le problème mécanique ait un sens. Cette étude de connection est effectuée de la même manière que pour le problème de l'optimisation de la section droite d'une barre présenté dans le chapitre précédent (section 5.2.3). En particulier, le même traitement est utilisé pour éliminer les structures non connectées : elles leur alloue une performance nulle, indépendamment de toute propriété mécanique, et ne sont donc pas prises en compte lors de l'étape suivante de sélection darwinienne.

Il faut noter que, à la différence de Chapman et al. [15] on n'impose pas de la matière sur certaines parties du contour où la plaque est supposée fixée, . Le processus d'optimisation choisira lui-même l'endroit où "accrocher" la structure sur le bord vertical. Ceci permet une plus grande flexibilité dans la résolution du problème d'optimisation, comme nous le verrons lors de la comparaison avec la méthode d'homogénéisation (paragraphe 6.3.2). Cela permet aussi, dans certains cas, de pouvoir proposer des solutions quasi-optimales diversifiées comme nous l'illustrerons sur le problème du cantilever modifié (section 7.2).

6.2.2 Simulation du comportement mécanique

Après identification de la composante matérielle reliant le point d'application du chargement au bord fixe (si elle existe), on utilise une analyse par éléments finis sur cette composante. La méthode des éléments finis utilisée est détaillée dans [39].
Notons encore, par rapport aux travaux antérieurs, que c'est la "vraie" structure qui est

utilisée pour le calcul par éléments finis, contrairement à la fois à Jensen (1992) et Chapman et al. (1994) qui considèrent la totalité du domaine pour l'analyse EF, en assignant un très faible module de Young aux éléments représentant du vide. Bien que les résultats des deux approches ne diffèrent pas de manière significative, utiliser la vraie structure nous permet de prendre en compte plus simplement des forces appliquées sur le contour inconnu (comme pour le problème du dôme sous pression présenté section 7.6), cette approche a en outre l'avantage de diminuer sensiblement le temps de calcul de la méthode des éléments finis puisque la taille de la matrice est réduite.

Notons enfin que cette approche nous permet de considérer des problèmes de chargements multiples (i.e. devant résister à différents chargements). Il suffit en effet d'effectuer autant d'analyses E.F que de cas de chargement et de les prendre en compte (d'une manière à préciser) dans le calcul final de la performance. De tels problèmes seront abordées en section 7.5.

Une fois que l'on dispose du poids de la structure et des résultats de la (ou des) simulation(s) numérique(s) de son comportement mécanique, on est en mesure de calculer une performance pour cette structure. Plusieurs expressions analytiques de la performance sont discutées dans les prochaines sections.

6.3 Les fonctions performance

6.3.1 Les problèmes

Le problème général de l'optimisation de structure est un problème d'optimisation multicritères : on cherche à minimiser le poids de la structure tout en conservant de "bonnes" propriétés mécaniques pour les cas de chargement considérés. Ce problème peut se formaliser de plusieurs manières, que nous allons examiner plus en détail, sur le cas-test présenté figure 1.

6.3.2 Minimiser la compliance

Une mesure différentiable de la raideur d'une structure soumise à un chargement donné est basée sur la *compliance*, ou travail des forces extérieures
$C = \int_{\Gamma_1} Fu.$
Le problème de l'optimisation de structure devient alors : minimiser le poids et la compliance.

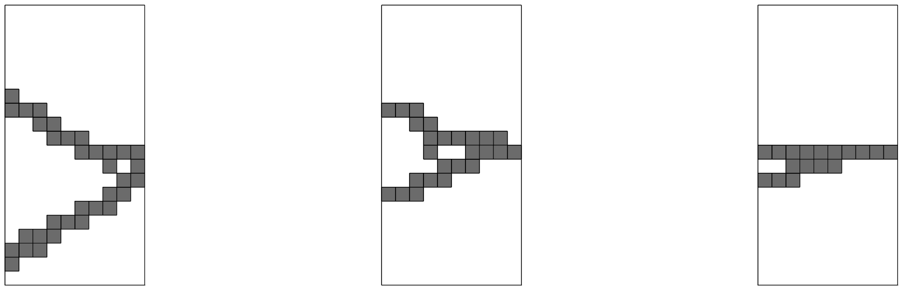
Les méthodes déterministes d'optimisation de forme par des techniques de gradient utilisent pour la plupart la compliance. Ainsi, la méthode d'homogénéisation minimise la fonction $Poids + \alpha C$ (section 1.1.3). Une fonction performance possible pour l'approche par algorithmes génétiques est

$$P_C = \frac{1}{A + \alpha C} \tag{6.3.6}$$

où α est un paramètre réel à fixer par l'utilisateur, suivant que l'on veut favoriser plutôt les structures de poids minimum (α petit) ou de compliance faible, mais de poids plus important (α grand).

Comparaison AG-homogénéisation

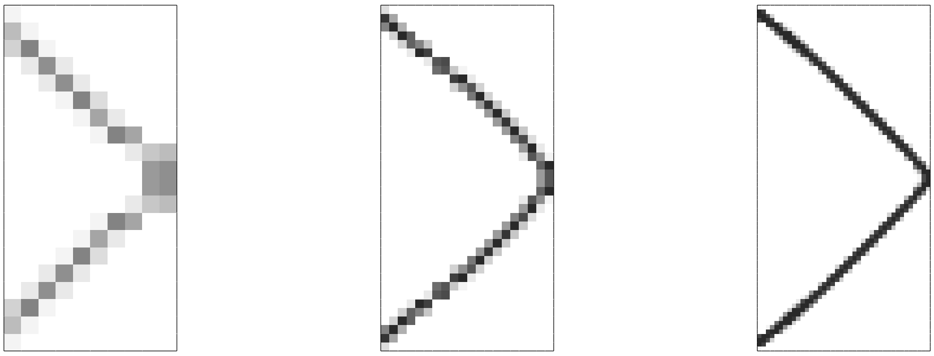
L'utilisation de la fonction performance donnée par l'équation 6.3.6 utilisant la compliance nous permet de comparer les approches par homogénéisation et Ags.
Les comparaisons sont faites sur le cantilever de référence 1×2 utilisant une discrétisation régulière de 10×20. Les figures 61 montrent les solutions obtenues par l'approche génétique pour différentes valeurs de α :(1, 0.1 et 0.01). La taille de la population est de 75. On fixe le nombre maximum de générations à 500. Les autres paramètres utilisés sont les mêmes que dans la section 7.2



(a) (b) (c)

Figure 61 : *Optimisation de la compliance par AGs pour un maillage grossier et différentes valeurs de alpha. (a):* $\alpha = 1$, *(b) :* $\alpha = 0.1$ *et (c) :* $\alpha = 0.01$

Les(figures 62) présentent les résultats obtenus par la méthode d'homogénéisation telle qu'elle est décrite en section 1.1.3, pour différents maillages et $\alpha = 1$.
Les résultats obtenus par ces deux approches diffèrent fortement. On remarque que pour la méthode d'homogénéisation la "projection" de la solution composite (densité de matière $\in [0, 1]$ dans chaque cellule) sur une solution "classique" (chaque cellule est pleine ou vide) est problématique lorsque le maillage est trop grossier.



(a) (b) (c)

Figure 62 : *Optimisation de la compliance par la méthode d'homogénéisation pour $\alpha = 1$, et différents maillages. (a) : 10 × 20 , (b) : 20 × 40 et (c) : 40 × 80*

En revanche, cette méthode trouve pour un maillage fin une solution et cette solution converge, lorsque l'on raffine le maillage, vers **LA** solution optimale continue. De plus, l'étape de pénalisation (projection sur l'ensemble des structures réalistes vide/matière) est de plus en plus facile lorsque le maillage devient très fin (figure 62(c)).

Tout se passe comme si l'algorithme génétique avait en quelque sorte contourné la difficulté en trouvant des solutions optimales, mais dans un espace de recherche plus petit alloué (maillage grossier).

6.3.3 Maximiser la raideur

Une autre approche consiste à utiliser la raideur de la structure en considérant le déplacement maximal de cette structure sous le chargement prescrit. Le gros inconvénient de cette approche pour les méthodes classiques est que le déplacement maximal n'est pas une grandeur différentiable. En revanche, ce n'est pas un problème pour les méthodes ne nécessitant que le calcul de valeurs de la fonction à optimiser telles les algorithmes génétiques.

Comme dans le chapitre 5 l'aire A de la matière connectée représente le poids de la structure. Si on suppose comme dans Jakiela [15] que la raideur de la pièce est inversement proportionnelle au déplacement maximal D_{Max} du point d'application de la force, maximiser la résistance de la plaque tout en minimisant son poids peut se formaliser par le problème de la maximisation de la fonction

$$P_R = \frac{Raideur}{Poids}. \tag{6.3.7}$$

sachant que la raideur est inversement proportionnel à D_{Max}, la performance devient

$$P_R = \frac{1}{D_{Max} A} \tag{6.3.8}$$

Résultats Utilisant de la raideur

La meilleure forme avec un rapport raideur sur Poids maximal donne (figures 63 et ../Ticite/chaprealbest.ps01) une forme légèrement différente de la forme supposée parfaite (figure 58)

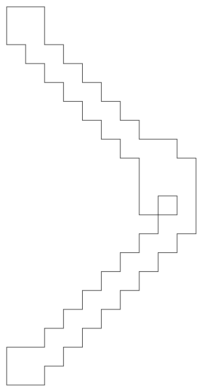


Figure 63 : *A:0.44*D_{Max}*:0.19* P_R*:11.91 La forme la plus légère obtenue n'a pas la meilleure performance*

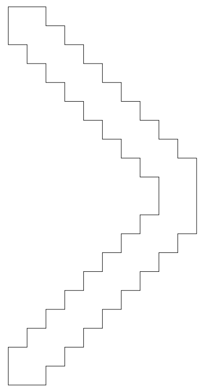
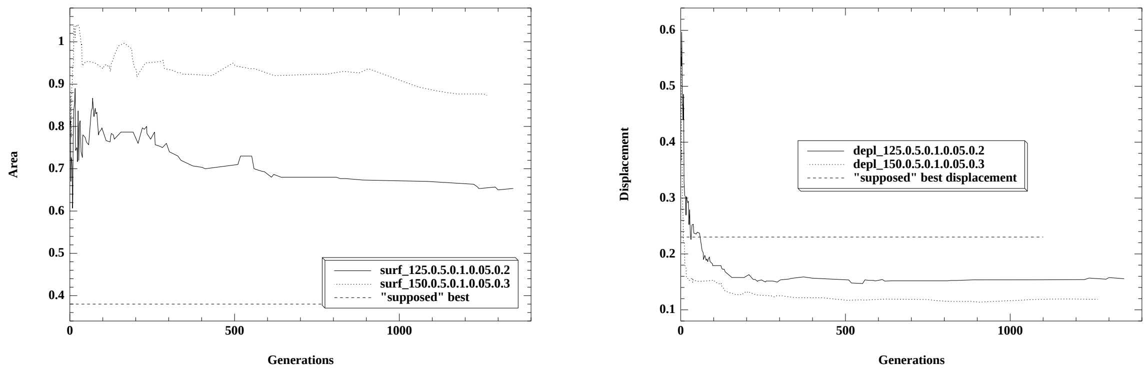


Figure 64 : *A:0.56* D_{Max}*:0.14* P_R*:12.61 meilleure performance mais plus lourde.*

La forme la plus légère obtenue donnant une bonne performance est la forme montrée figure 63. La forme de la figure 64 a une meilleure performance bien qu'elle soit moins légère. Si l'on compare l'évolution de la surface des formes obtenues en fonction des générations (figure 65(a)) et que l'on prend comme forme de référence la plaque supposée parfaite (figure 58), on obtient les courbes suivantes :

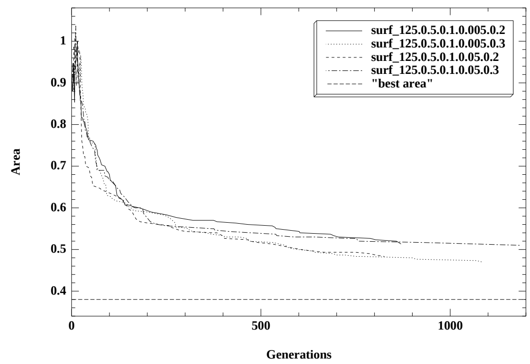


(a) (b)

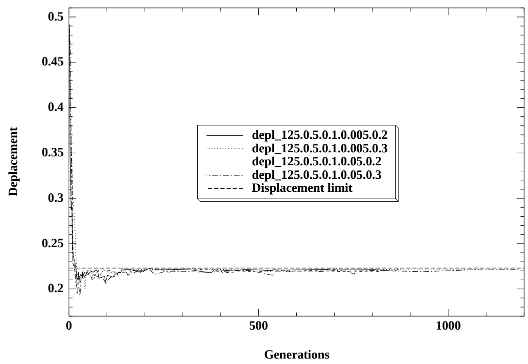
Figure 65 : *Evolution de la surface et des déplacements.*

Les résultats représentés figures 65 ont été obtenus avec la fonction performance utilisant la raideur. Ces résultats sont des moyennes sur 20 tests utilisant respectivement des

populations de taille 125 (en traits pleins) et 150 (en pointillés).
En reprenant les mêmes tests mais en utilisant cette fois ci la performance pénalisée (équation 6.3.9)pour des populations de taille 90, 100, 125 et 150, on obtient les figures 72.



(a)



(b)

Figure 66 : *évolution de la surface et des déplacements pour* $D_{Lim} = 0.2215$

On remarque que les surfaces des structures obtenues par la performance utilisant la raideur sont plus importantes que celle de la “forme parfaite” par contre, les déplacements sont effectivement plus faibles, ce qui fait de la structure trouvée (par analyse de la raideur) une structure très stable mais également très épaisse.

6.3.4 Minimiser le poids sous contraintes

Une approche sans doute plus réaliste du problème du déplacement maximal est de considérer l'optimisation de structures comme un problème de minimisation (du poids) sous contraintes (sur le déplacement maximal). En effet, les cahiers des charges des bureaux d'étude contiennent plus facilement des spécifications sur le déplacement maximal sous un chargement donné, faciles à vérifier *a posteriori*, que sur la compliance ou la raideur définie dans la section précédente. C'est l'approche que nous avons adoptée.
Il existe de nombreuses approches des problèmes d'optimisation sous contraintes par algorithmes d'évolution artificielle. Un aperçu de ces méthodes peut être trouvé dans [50]. Néanmoins, la méthode la plus employée car la plus facile à mettre en œuvre reste la méthode de pénalisation : la performance est donnée par la fonction objectif, éventuellement *pénalisée* en cas de violation des contraintes :

$$P = \frac{1}{A + \alpha(D_{Max} - D_{Lim})^+} \tag{6.3.9}$$

D_{Max} est le déplacement maximal du point d'application de la force, D_{Lim} est le déplacement-limite imposé, A représente l'aire de la vraie structure (i.e. celle de la composante connectée utilisée par les EF), x^+ signifie que l'on prend la partie positive de x, et enfin

α est un paramètre réel positif appelé *facteur de pénalisation*. La détermination de ce paramètre reste la difficulté majeure de cette approche. Si des principes généraux peuvent être appliqués [35], la méthode d'essai-erreur reste la plus utilisée en pratique. Nous verrons dans la section 6.4.2 comment nous avons tenté de résoudre cette difficulté.

Discussion

Parmi les trois types de fonctions performance que nous avons considérés nous remarquons que :

- ni la performance prenant en compte la compliance ni celle basée sur la raideur ne permettent un contrôle précis du déplacement maximal en charge de la structure optimale, qui est généralement une donnée importante du cahier des charges.
- la performance utilisant la raideur ne nous donne pas la structure la plus légère possible mais une structure présentant un bon rapport résistance sur poids (section 6.3.3).
- la performance utilisant une approche par pénalisation proportionnelle à la violation de la contrainte pose le problème du réglage du paramètre de pénalisation. Pour de petites valeurs de α, la meilleure solution risque de ne pas respecter la contrainte imposée, alors que de grandes valeurs de α risquent d'interdire certaines régions de l'espace de recherche pouvant constituer des sortes de “raccourcis” pour l'optimisation : la solution trouvée peut ne pas être optimale.

Dans toute la suite de cette thèse, nous utiliserons néanmoins une fonction performance prenant en compte le déplacement limite imposé D_{Lim}.

En fait, compte-tenu des problèmes posés par les composantes de matière non connectées avec la structure principale (voir la discussion section 5.2.5), la fonction performance utilisée est la suivante:

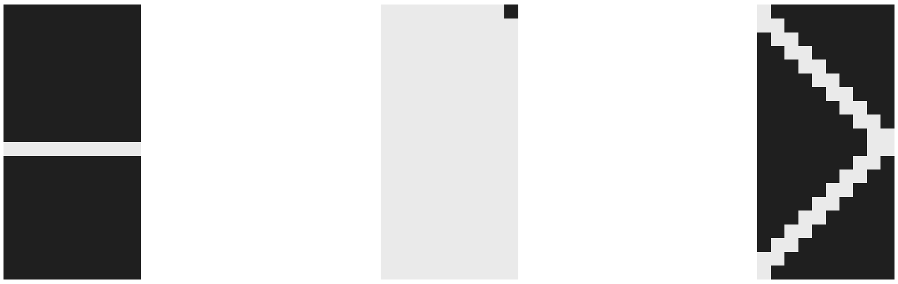
$$\frac{1}{A_{co} + \epsilon A_{nc} + \alpha(D_{Max} - D_{Lim})^{+}} \tag{6.3.10}$$

où A_{co} est l'aire de la structure réelle, $Aire_{nc}$ est l'aire totale du matériel non connecté à la composante principale et ε un (petit) paramètre positif,
D_{Max} et D_{Lim} sont respectivement le déplacement maximal de la structure (calculé par éléments finis) et la limite imposée sur ce déplacement, α est le paramètre de pénalisation à ajuster. La fin de ce chapitre est consacrée à une étude plus détaillée de cette fonction performance.

6.4 La performance pénalisée

6.4.1 Influence de D_{Lim}

En fonction du déplacement limite imposé, la structure optimale a des topologies qui peuvent être très différentes. Pour l'exemple de la plaque console discrétisée en 10×20, on peut ainsi tendre vers les trois topologies représentées par les figures 67.



(a) (b) (c)

Figure 67 : *(a:D_{Max}=46.64) La forme la plus légère possible mais avec un D_{Lim} autorisé très grand. (b:$D_{Max} = 0.06$ la forme la plus lourde on s'autorise un D_{Lim} très petit (c:$D_{Max} = 0.27$) la forme à peu près en ">" un D_{Lim} intermédiaire*

Observons le comportement de l'algorithme pour chacun de ces cas.

Grand déplacement limite ($D_{Lim} = 46.64$)

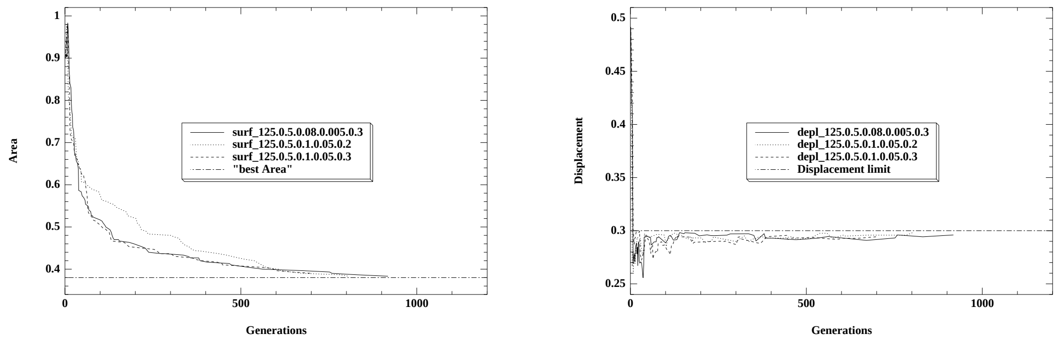
Nous permettons à la plaque de grands déplacements. La plaque "barre" se rapprochant de la (figure 67(a)) est effectivement obtenue au bout de quelques itérations (voir figure 68).

Figure 68 : $D_{Max} = 9.63$, $D_{Lim} = 46.64$, *Aire =0.17*

Déplacement limite intermédiaire

Suivant les valeurs de D_{Lim} on obtient une plaque en forme de ">" plus ou moins renforcée.

- $D_{Lim} = 0.30$

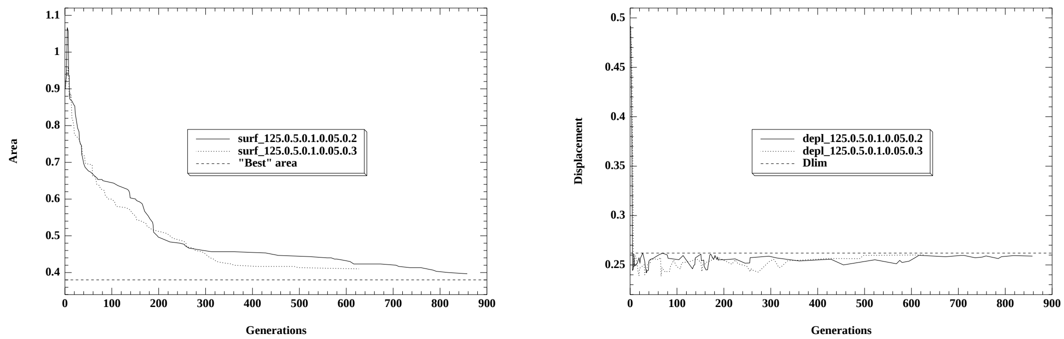


(a) (b)

Figure 69 : *évolution de la surface et des déplacements pour $D_{Lim} = 0.30$*

La figure (69a) représente l'évolution de la surface du meilleur individu en fonction des générations. Chaque courbe est une moyenne de 20 tests et les différentes courbes obtenues sont comparées avec une courbe de référence qui est l'aire de la plaque en ">" supposée parfaite. De manière analogue, la figure 69(b) donne l'évolution des déplacements de la meilleure plaque au cours des générations, moyennés sur 20 essais indépendants. Les courbes obtenues sont comparées avec le déplacement limite imposé.
Il se dégage de ses figures que les contraintes sont respectées et la surface tend vers la surface minimale espérée.

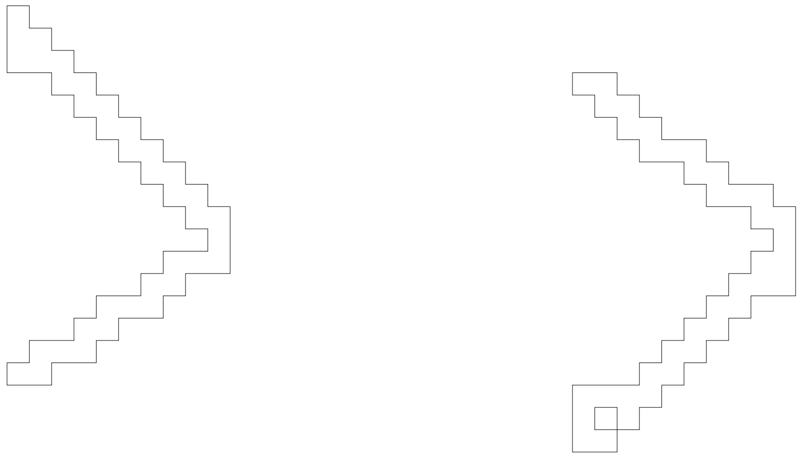
- $D_{Lim} = 0.262$
  On s'impose une limite sur le déplacement plus faible que précédemment et on refait les mêmes expériences. L'évolution des meilleurs individus (moyennés sur 20 essais indépendants) au cours des générations est représentée par les figures 70.



(a) (b)

Figure 70 : *évolution de la surface et des déplacements pour $D_{Lim} = 0.262$*

La figure 70(b) montre que les meilleurs respectent les contraintes. Il faut noter cependant que chaque population intermédiaire contient des individus ne respectant pas les contraintes sur le déplacement. Les figures 71(a) et (b) présentent les meilleurs formes obtenues dans ce cas.

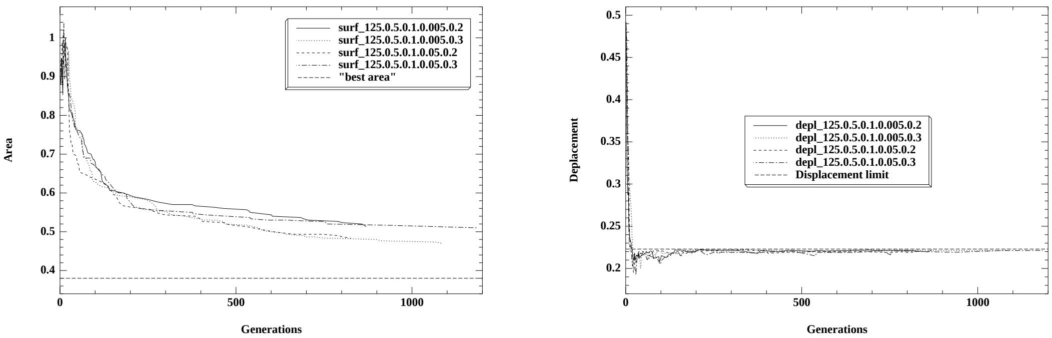


(a) (b)

Figure 71 : *type de structures obtenues*

Tous les tests n'ont bien sûr pas donnés ces formes optimales, ce qui explique pourquoi la courbe des moyennes des meilleures surfaces n'a pas atteint la valeur minimale espérée.

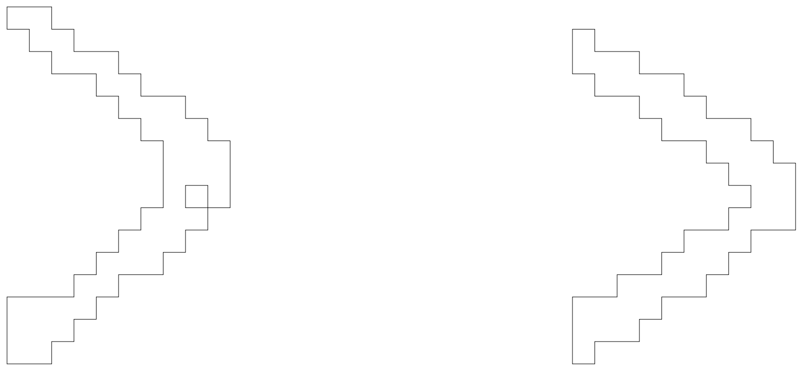
- $D_{Lim} = 0.2215$ On s'impose un déplacement limite encore plus petit, et on refait les mêmes tests que dans les parties précédentes.



(a) (b)

Figure 72 : *évolution de la surface et des déplacements pour* $D_{Lim} = 0.2215$

L'analyse des courbes d'évolution nous montre que les contraintes sont parfaitement respectées, mais les moyennes des meilleures surfaces sont loin de la forme en “>” parfaite. Un bref coup d'oeil sur les structures obtenues (figures 73 (a) et (b)) confirme que les meilleurs formes obtenues pour ce D_{Lim} sont lourdes.



(a) (b)

Figure 73 : *deux exemples de structures assez épaisses obtenues pour* $D_{Lim} = 0.2215$

Déplacement limite petit($D_{Lim} = 0.06$)

Nous obtenons cette fois ci une plaque très épaisse mais pas complètement pleine (figure 74).

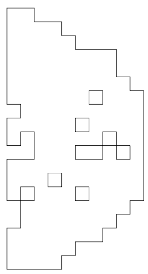


Figure 74 : $D_{Max} = 0.081$, $D_{Lim} = 0.06$, *Aire* = *1.33*

Au bout de 2000 générations (correspondant au test d'arrêt), les meilleures formes obtenues ne respectent toujours pas la contrainte imposée.

Discussion

Les résultats obtenus démontrent la grande influence du D_{Lim} imposé, (on remarque la grande différence entre les résultats obtenus pour $D_{Lim} = 0.26$ et pour $D_{Lim} = 0.22$). Cela confirme que les méthodes utilisant la compliance ou la raideur ne permettent pas un contrôle précis de D_{Lim}, or ce contrôle est nécessaire car une petite variation de D_{Lim} peut entraîner des différences non négligeables entre les structures obtenues (du moins autour d'une valeur critique).

Cependant, tous ces résultats ont été obtenus avec un paramètre de pénalisation fixé arbitrairement (ajusté par essai-erreur). Nous allons maintenant étudier l'influence que ce paramètre peut avoir, et les solutions envisagées pour contourner ce problème.

6.4.2 Le paramètre de pénalisation

Généralités

L'un des problèmes fondamentaux pour l'utilisation de fonctions pénalisées (et pas seulement pour les AGs) est la détermination du poids respectif de la fonction objectif et des pénalités, réglée dans notre cas par le paramètre de pénalisation α. Nous étudions dans cette section diverses stratégies de pénalisation.

Les expériences sont faites sur le même problème test de la plaque cantilever (voir section 6.3). Nous supposons que tous les paramètres (en dehors bien entendu du paramètre de pénalisation) sont constants, en particulier $D_{Lim} = 0.3$, correspondant à une valeur moyenne de la limite sur le déplacement maximal de la structure (voir section précédente).

Premiers résultats

Pour α fixé, étudions expérimentalement l'influence de ce paramètre sur la solution.

- α=1
  Les surfaces obtenues sont très légères mais la contrainte n'est pas respectée.
- $\alpha = 1000$
  Ce sont les meilleurs résultats obtenus : surfaces minimales et contraintes respectées.
- $\alpha = 100000$
  Les contraintes sont respectées. La surface des plaques optimales obtenues reste assez importante, mais en augmentant le nombre de générations on arrive à diminuer cette surface.

Les paramètres α choisis sont caricaturaux, mais les résultats démontrent que l'influence de α sur le problème d'optimisation n'est pas négligeable. Nous verrons dans la section suivante comment on peut arriver à diminuer l'impact de ce paramètre sur les résultats en le modifiant au cours des générations.

Choix d'un paramètre α évolutif

Il y a deux manières de faire évoluer α au cours des générations soit de manière exogène, en imposant un schéma d'évolution figé, soit de manière adaptative, en réglant α en fonction des résultats.

Dans les deux cas, l'idée est d'utiliser un paramètre relativement faible au début, pour ne pas empêcher l'émergence de topologies qui peuvent d'abord violer les contraintes, mais qui, au fur et à mesure des générations, pourront donner naissance à des individus respectant les contraintes. Ce facteur de pénalisation croît ensuite ensuite avec les générations de façon à arriver en fin d'exécution à des structures respectant les contraintes et de plus en plus légères.

- Croissance de α imposée en fonction des générations
  Une valeur moyenne de α est d'abord choisie, qui permet de déterminer une population de structures légères. Ensuite, au cours de l'évolution, on augmente la valeur de α géométriquement (toutes les n générations par exemple) pour assurer progressivement le respect des contraintes. De telles méthodes utilisant les AGs itérées ont été utilisées dans l'optimisation des treillis [60] aussi bien que dans d'autres domaines d'application ([38]).
  Le facteur de pénalisation utilisé à la génération i est donnée par :
  $$\alpha_i = \alpha_0 (M)^{[\frac{i}{n}]}.$$
  α_0 est la valeur au départ, M est le facteur multiplicatif . $[\frac{i}{n}]$ est la partie entière de $\frac{i}{n}$.

- α adaptatif en fonction de la population
  L'autre possibilité consiste à choisir le paramètre α en fonction de tous les individus composant la population, en tenant compte à la fois de leurs surfaces et du respect des contraintes. Tout comme dans le paragraphe précédent, α doit croître en fonction des générations. Le principe adopté pour construire ce paramètre α_i de la génération i est :
  $$\alpha_i = max(\alpha_{i-1}, \beta_i)$$
  sachant que β_i prend en compte toutes les contraintes violées et l'aire des structures.

Etude comparative

Nous présentons une étude expérimentale des trois approches proposées sur le problème de la plaque cantilever : α fixe, α croissant géométriquement et α adaptatif. La figure 75 montre l'évolution de la surface des meilleures structures obtenues par les méthodes de pénalisation utilisées. La figure 76 représente elle l'évolution de la fonction performance. Ces résultats sont toujours des moyennes sur 20 essais indépendants.

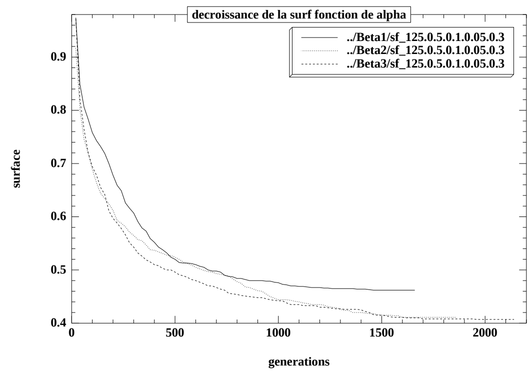


Figure 75 : *évolution de la surface*

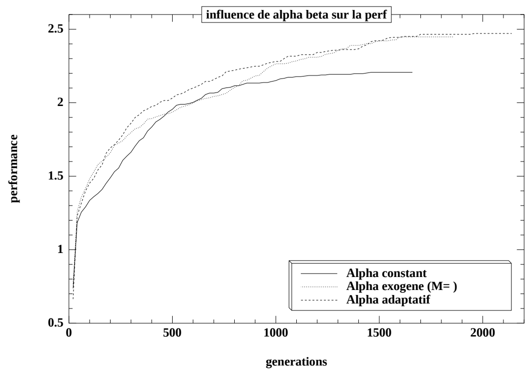


Figure 76 : *évolution de la performance*

Les résultats obtenus confirment que prendre α constant (même après une campagne d'essais intensifs pour déterminer une valeur optimale) donne de moins bons résultats qu'un paramètre qui évolue au cours du temps soit en fonction des générations soit encore en fonction de la population. On ne peut cependant pas dire laquelle des deux méthodes évolutives est la meilleure.
Pour essayer de voir laquelle de ces deux méthodes est vraiment la meilleure nous avons considéré une autre série de tests. Les figures (77 et 78) donnent les résultats de la comparaison des deux cas utilisant un paramètre α variable. Il y a un léger avantage pour le choix de α fonction de la population entière. Le choix de α qui augmente en fonction des générations est d'une certaine manière "aveugle" et ne tient pas compte de l'état de la population, ce qui peut expliquer ce désavantage.

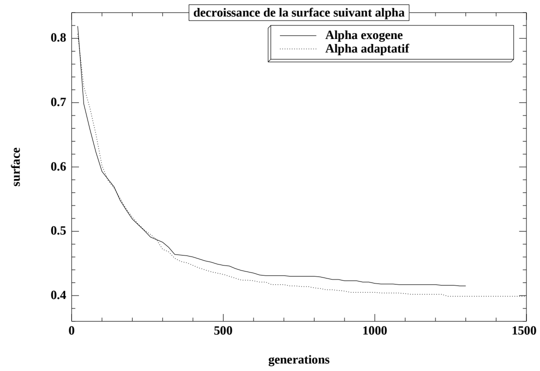


Figure 77 : *évolution de la surface*

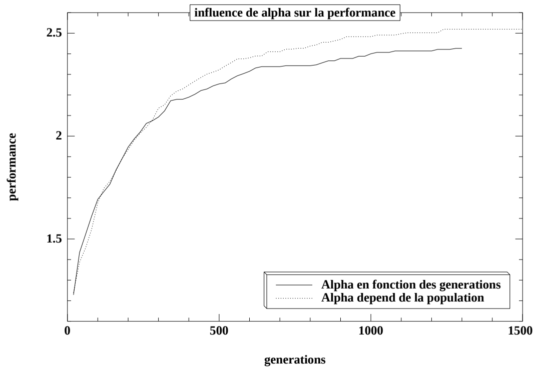


Figure 78 : *évolution de la performance*

6.5 Conclusion

L'étude de diverses formulations pour la fonction performance nous a conduit à ne plus considérer dans la suite que le problème de minimisation du poids sous contraintes. Les contraintes sont traitées par une méthode de pénalisation de la fonction objectif. Le point noir le plus important reste la détermination du paramètre de pénalisation α. Les expériences numériques effectuées montrent l'importance de ce paramètre, et l'avantage qu'il y a à ne pas le fixer durant l'évolution. Dans toutes la suite nous utiliserons le schéma d'évolution géométrique de α au cours des générations, qui s'est avéré sinon le plus performant mais en tout cas le plus robuste.

Nous avons maintenant entre les mains tous les outils nécessaires à une l'application réussie des techniques des algorithmes génétiques pour des problèmes difficiles d'optimisation topologique des formes en calcul des structures: c'est l'objet du chapitre suivant.

Chapitre 7

Résultats

Dans ce chapitre nous présentons les résultats originaux, du point de vue de l'Optimum Design, que nous avons obtenus par l'approche Algorithmes génétiques. Ce chapitre utilisera donc tous les résultats et mises au point détaillées dans les chapitres précédents : les opérateurs décrits dans le chapitre 5 et la fonction performance introduite au chapitre 6. Une instance de problème de plaque cantilever modifiée nous servira d'exemple de validation en section 7.2. Nous verrons ensuite l'application des algorithmes génétiques à des problèmes couramment étudiés en ingénierie et plus spécifiquement en optimisation topologique de structure.

Dans la section 7.4, la plaque cantilever est de nouveau modifiée pour mettre en valeur la capacité de l'approche génétique à résoudre les problèmes ayant des solutions optimales multiples (ou encore des solutions quasi-optimales).

Nous présenterons ensuite des problèmes plus complexes. Nous montrerons la facilité de mise en œuvre des problèmes de multi-chargements (où des contraintes sur les déplacements sous plusieurs cas de charge doivent être considérées) en section 7.5.

Nous verrons ensuite (section 7.6) un problème de chargement sur la frontière inconnue avec l'exemple du dôme sous pression. Ce type d'optimisation est aujourd'hui hors de portée des méthodes d'optimisation de forme par homogénéisation.
Dans la section 7.7 nous nous intéressons au modèle non linéaire en grands déplacements. Dans ce cas, les effets géométriques de la non linéarité du modèle peuvent conduire à des structures non viables à moins d'introduire des contraintes sur le champ de contrainte mécanique. Ici encore, il s'agit de résultats originaux, la résolution de problèmes d'optimisation topologique de forme dans ce contexte mécanique n'ayant à notre connaissance encore jamais été réalisée auparavant.

7.1 Conditions expérimentales

Pour tous les tests nous considérons une population de taille 125, le nombre de générations maximal est arbitrairement fixé à 1000. Notons qu'une exécution complète (1000 générations) nécessite environ 100000 analyses éléments finis, ce qui nécessite environ 6 heures de calcul sur une station HP 755 pour une discrétisation moyenne (par exemple 22×32, cf section 7.2).

Les opérateurs génétiques utilisés sont décrits dans les sections 5.4 et 5.5 : croisement par blocs (à 3 blocs) avec une probabilité de 0.6 et mutation générale avec des probabilités P_{max} et P_{min} de 0.005 et 0.0001 respectivement.

Les structures instables, dont la composante connectée contenant le point d'application n'est pas relié au bord fixe, ont une performance mise à zéro. La performance associée aux structures stables est celle définie au chapitre 6 par

$$P = \frac{1}{A_c + \varepsilon A_{nc} + \alpha (D_{Max} - D_{Lim})^+}$$

Le paramètre de pénalisation α suit une évolution géométrique d'un facteur 1.1 toutes les 10 générations (section 6.4.2).

7.2 Un autre problème de "cantilever"

On considère toujours une plaque encastrée mais on modifie les dimensions de la plaque par rapport aux chapitres précédents : la plaque est cherchée dans un domaine rectangulaire de dimensions 1.6×1.1. Elle est fixée par son bord gauche, une force est appliquée au milieu du bord opposé (figure 79).

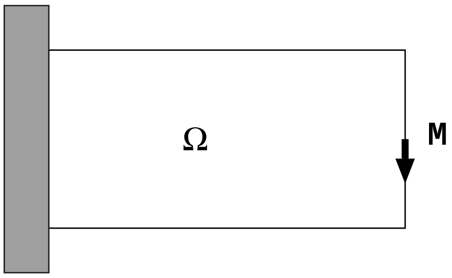
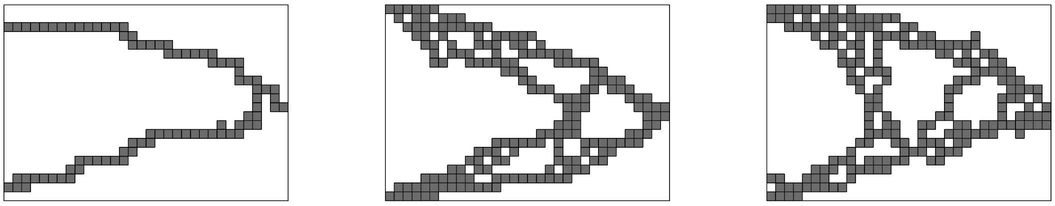


Figure 79 : *La plaque cantilever standard* 1.6×1

Résultats

Les figures 80 montrent des résultats obtenus sur la plaque cantilever 1.6×1.1 discrétisée selon un maillage régulier de 32×22.
Chacune des figures obtenues correspond à une valeur donnée de la contrainte sur le déplacement maximum D_{Lim}. Chacun des résultats présenté est le plus significatif des tests. Les résultats montrés ont été obtenus après 5 exécutions utilisant des populations initiales différentes.
Les résultats différents d'une exécution à l'autre soulignent ainsi l'aspect stochastique des algorithmes génétiques. On peut ainsi remarquer que la seconde structure (b) bien qu'étant optimisée avec utilisation d'une contrainte plus forte sur le déplacement, est à la fois plus légère et plus rigide que la troisième. Ces différences ne sont toutefois pas très importantes sur la plaque cantilever 1.6×1. D'autre part, un effet secondaire de

cet aspect stochastique est la possibilité d'obtenir plusieurs solutions quasi-optimales, qui sera illustrée plus en détails dans la section 7.5.



(a) (b) (c)

Figure 80 :

	(a)	*(b)*	*(c)*
D_{Lim}	*2.463*	*0.13*	*0.133*
D_{Max}	*2.439*	*0.13*	*0.133*
A	*0.2075*	*0.4675*	*0.49*

Plaques cantilever optimale pour différentes valeurs des contraintes sur le déplacement D_{Lim}. *La vraie valeur du déplacement maximal de la structure est* D_{Max}.

Une discrétisation plus fine pour le même problème donne par exemple comme résultat la figure 81.

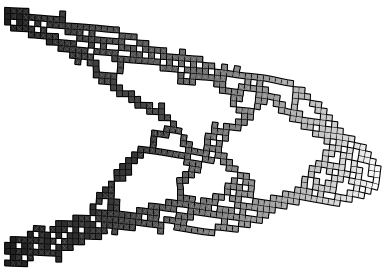


Figure 81 : *La déformée de la structure obtenue pour un maillage* 64×44 *pour la plaque* 1.6×1.1 $D_{Lim} = 0.18$, $D_{Max} = 0.178,\ A = 0.3725$

Les déplacements limites imposés ont été respectés. Les structures trouvées ont le même aspect général que celles obtenues par les méthodes d'homogénéisation (Allaire-Bonnetier-Franckort-Jouve)[2]. On notera la tendance de l'algorithme, lorsque l'on raffine le maillage, à générer de multiples petits trous. Or nous savons que la solution optimale au problème posé dans l'espace relaxé (cf section 1.1.3) est composée d'une infinité de micro-structures : l'algorithme génétique va dans la bonne direction.

7.3 L'exemple de la roue

Nous avons considéré un autre exemple d'application dans le même cadre avec des conditions aux limites différentes dont la solution devrait être une roue.

Domaine et chargement

Le problème est représenté par la figure (82). On souhaite trouver la forme optimale contenant le matériau imposé représenté en gris sur la figure (le point d'application de la force est supposée naturellement contenir de la matière, le point extremum du bord en bas à droite aussi). On impose comme conditions aux limites les conditions également représentées sur la figure 82 on suppose que le point en bas à droite est fixé en y et libre en x, et que tout point du bord gauche et fixé en x et libre en y (conditions de symétrie par rapport à Oy).

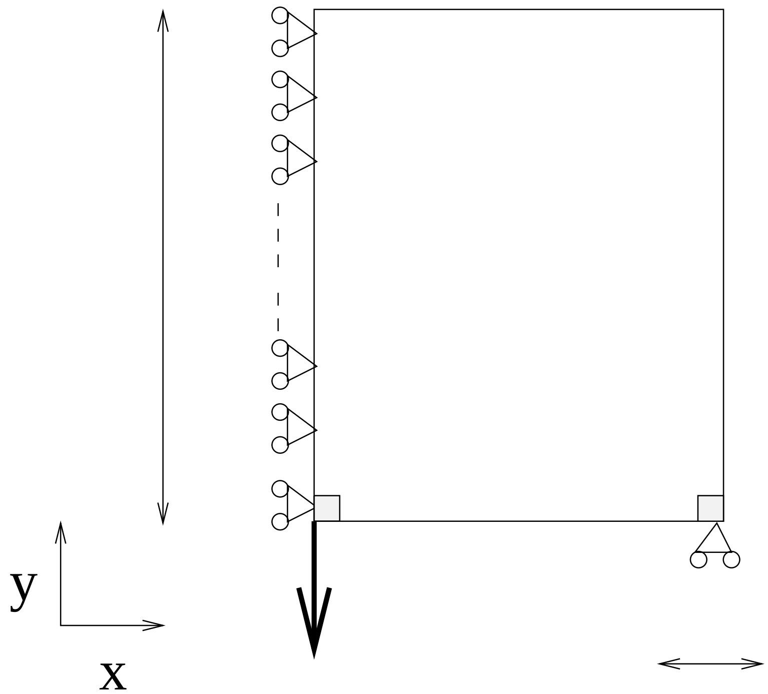
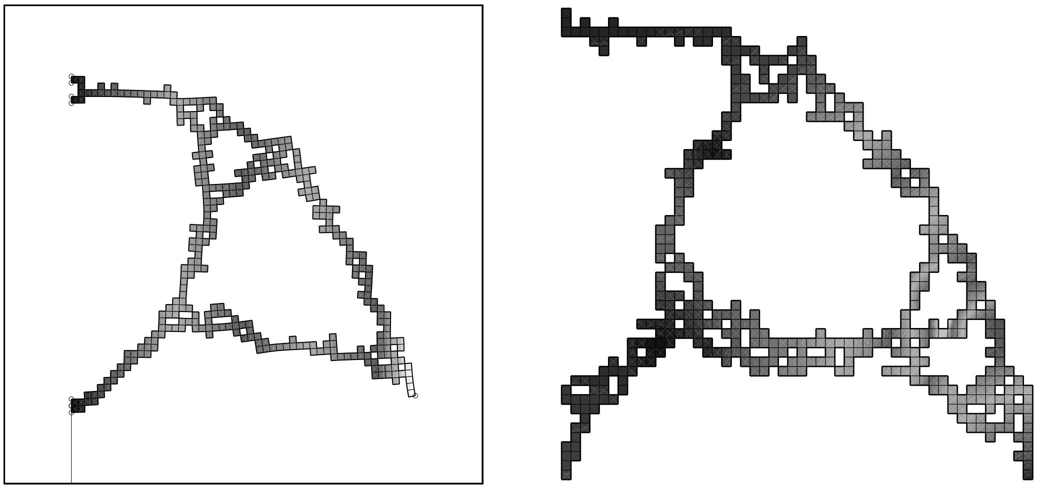


Figure 82 : *domaine et chargement*

On considère une population de taille 100.
Les solutions optimales satisfaisant les contraintes de déplacement sont représentées figures 83 et 84.



(a) (b)

Figure 83 : *Deux solutions optimales: imposition de la matière*

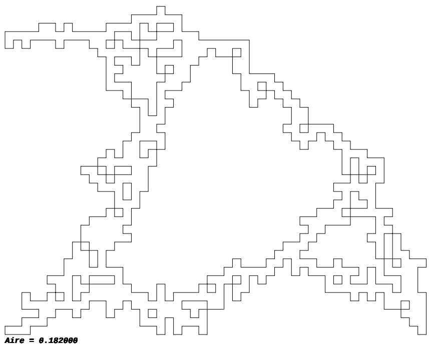


Figure 84 : *Un résultat sans imposer de la matière*

La figure 83(a) ressemble bien à un quart de roue, c'est la solution espérée. Par contre, la solution 83(b) montre que le résultat n'est pas symétrique, et si on complète la “roue” par symétrie on aboutit à une forme plus ovale que ronde. La question qui se pose est : a-t-on bien modélisé au départ le problème de la roue ? Utiliser une force verticale favorise une direction. Il serait sans doute judicieux de modifier le chargement (en considérant par exemple plusieurs conditions de chargement différentes). Nous avons cependant préféré valider la partie multi-chargement sur un problème plus courant en Optimum Design, le problème du vélo (section 7.5).

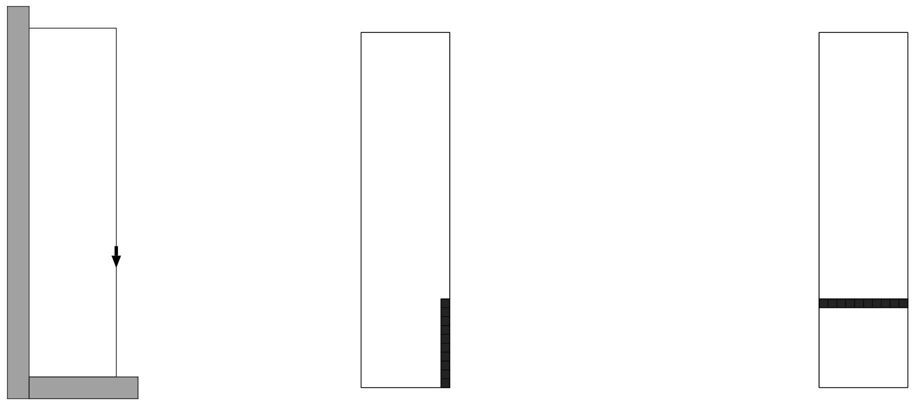
Pour les deux premières roues présentées ci dessus, on a imposée l'existence de matière au point en haut à gauche. Cette contrainte n'existait pas pour la troisième "roue" représentée figure 84 : l'absence de symétrie est accentué.

Remarques

Tous les problèmes présentés jusqu'ici se placent dans le cadre de l'élasticité linéaire et peuvent être résolu facilement par la méthode d'homogénéisation. Ils nous ont servis à valider notre approche. Le but des méthodes génétiques en Optimisation de Formes n'est pas de concurrencer les méthodes d'homogénéisation, qui restent plus efficaces de plusieurs ordres de grandeur pour les problèmes d'élasticité linéaire. Par contre, et c'est l'objet des sections suivantes, notre méthode permet de résoudre des problèmes que les méthodes classiques ne peuvent pas envisager.

7.4 Solutions multiples

Afin d'illustrer la capacité des AGs à trouver des solutions multiples, nous allons étudier le problème d'une plaque cantilever de dimension 1×4, discrétisée suivant un maillage régulier de 10×40, pour laquelle les contours gauche et bas sont fixes (voir figure 85(a))

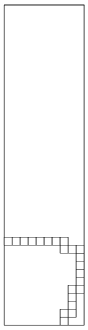
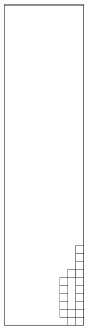
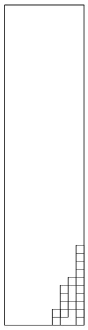
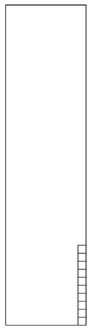


(a) (b) (c)

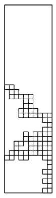
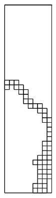
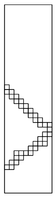
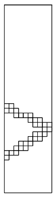
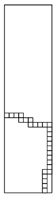
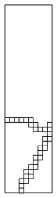
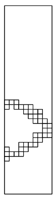
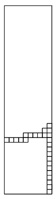
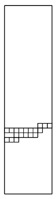
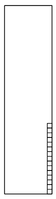
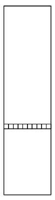
Figure 85 : *Le problème et deux solutions optimales*

Selon la hauteur du point d'application de la force et suivant la contrainte imposée sur le déplacement, le même problème peut avoir plusieurs solutions optimales. Un exemple simple de tels cas est montré figures 85 (b) et (c) : la force est appliquée au point (10,10) et, pourvu que l'on choisisse la contrainte sur le déplacement suffisamment grande, les deux structures (b) et (c) sont toutes deux optimales. De la même manière, si le point d'application est fixé et que l'on relaxe graduellement la contrainte sur le déplacement, il existe différentes solutions quasi optimales pour différentes valeurs de la limite sur le déplacement D_{Lim}. Nous présentons dans la suite différents résultats parmi les plus significatifs obtenus suivant le point d'application de la force.

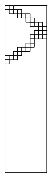
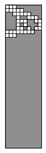
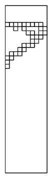
point d'application : (10,10)

				
D_{Lim}	0.4324	1.174		3.432
D_{Max}	0.2532	1.1704	1.1454	3.432
$Aire$	0.22	0.18	0.0.18	0.1

point d'application de la force : (10,15)

D_{Lim}	
0.144	 *Aire=0.74* D_{Max}*=0.143*  *Aire=0.52* D_{Max}*=0.14076*
0.32	 *Aire=0.38* D_{Max}*=0.294*
0.396	 *Aire=0.34* D_{Max}*=0.3919*  *Aire=0.27* D_{Max}*=0.3693*  *Aire=0.33* D_{Max}*=0.3927*
0.42	 *Aire=0.39* D_{Max}*=0.405*  *Aire=0.26* D_{Max}*=0.3415*
7.8	 *Aire=0.19* D_{Max}*=6.743*  *Aire=0.15* D_{Max}*=7.766*
46.7	 *Aire=0.1* D_{Max}*=46.6*

point d'application de la force : (10,35)

0.429	 Aire=0.33 D_{Max}=0.4061  Aire=0.41 D_{Max}= 0.428
0.559	 Aire=0.34 D_{Max}=0.5474

Les différents tests présentés illustrent parfaitement la capacité des algorithmes génétiques à localiser plusieurs solutions quasi-optimales.

Il existe en fait deux manières d'arriver à ce résultat :

- En lançant plusieurs fois l'algorithme à partir de points de départ différents, les solutions obtenues sont généralement légèrement différentes du fait du caractère aléatoire du processus.

- Par l'utilisation de la technique de partage (ou sharing) présentée à la section 2.2 qui permet de trouver plusieurs optima ou quasi-optima au cours d'un même essai.

Le seconde méthode est à la fois plus élégante et plus sûre, puisqu'elle ne dépend pas de parcours chanceux dans l'espace des populations. Par contre, elle nécessite la mise au point du facteur de partage, ce qui peut se révéler fastidieux (les quelques résultats heuristiques de la littérature [25] sont de peu d'aide pour notre problème). De plus, elle requiert un surcout en temps calcul puisqu'il faut calculer les distances deux à deux des individus de la population. C'est pourquoi les résultats présentés ici ont été obtenus par la première méthode ...

7.5 Les problèmes de chargements multiples

La plupart des situations réelles de fonctionnement des structures les soumettent à plusieurs cas de chargement. Nous nous sommes intéressés au cas du vélo : le cycliste pédalant sur son vélo sollicite le cadre de celui-ci de plusieurs manières très différentes suivant les conditions de terrain (plat, montée, descente), la vitesse à laquelle il souhaite de déplacer, le nombre de passagers qu'il transporte, ... Il est donc nécessaire de prévoir un cadre qui résiste dans toutes ces situations.

7.5.1 La fonction performance

Comme les AGs ne nécessitent que le calcul des valeurs de la fonction à optimiser, la prise en compte des chargements multiples se réduit ici à une modification de la fonction performance.
La nouvelle performance dépend du nombre de chargements considèrés. Nous l'avons choisie de la même forme que les fonction performances utilisées jusqu'ici, c'est-à-dire :

$$perf_{Multicharge} = \frac{1}{Aire + \sum_{i=1}^{n} \alpha_i ({D_{Max}}^i - {D_{Lim}}^i)^+}. \tag{7.5.1}$$

Sachant que α_i est le paramètre de pénalisation du déplacement pour le ième chargement. ${D_{Max}}^i$ est le déplacement maximal sous le chargement i, et ${D_{Lim}}^i$ le déplacement limite imposé toujours pour ce chargement. Les paramètres α_i varient au cours des générations suivant la méthode décrite dans la section 6.4.2.

7.5.2 Le problème du vélo

Domaine de travail

Ce problème du vélo étudié par Rasmussen et Olhoff(1992), [56] avec la méthode d'homogénéisation, a été repris par Chirehdast [46] et par Allaire, Jouve [3]. Le but est d'optimiser la rigidité d'une structure de bicyclette.
Une première formalisation possible du problème est donnée figure 86 : le gris représente la matière imposée, les conditions de chargement sont indiquées par des flèches (il y a ici 6 cas de chargement). Nous discrétisons le domaine de travail en un maillage régulier de 25×44 (le côté de chaque élément mesure 25 mm).
Les points (0, 100) et (1100, 100) sont respectivement un point roulement sans glissement et un point fixé.

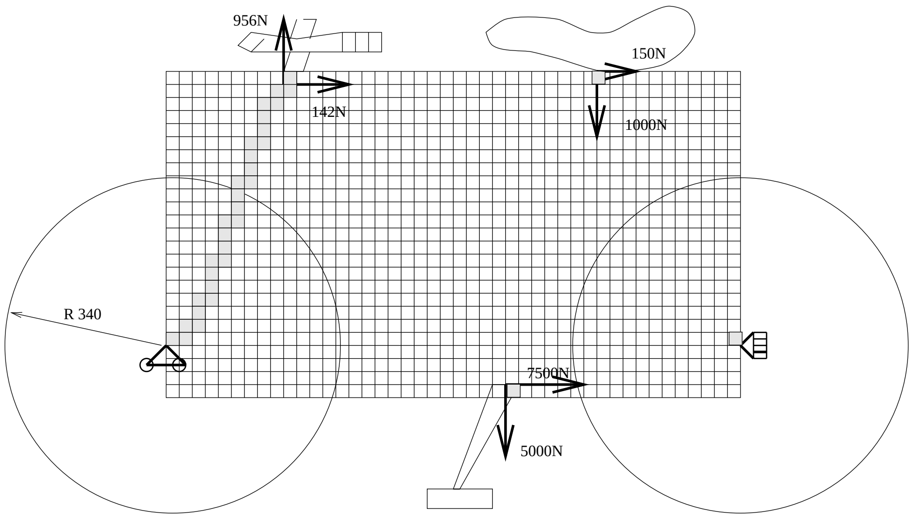


Figure 86 : *domaine de travail et chargement*

résultats

chge(F_x, F_y)	(142, 0)	(0, 956)	(1500, 0)	(0, -1000)	(7500, 0)	(0, -5000)
(D_{Lim}, D_{Max}	(1.1411,)	(3.979,)	((1.256,),)	(5.182,)	(11.932,)	(38.772,)

Table 7.5.2: Charges et déplacement limites imposés.

Le tableau 7.5.2 résume les charges appliquées, les déplacements limites imposés et les déplacement maximaux obtenus pour la structure optimale calculée et représentée figure 87

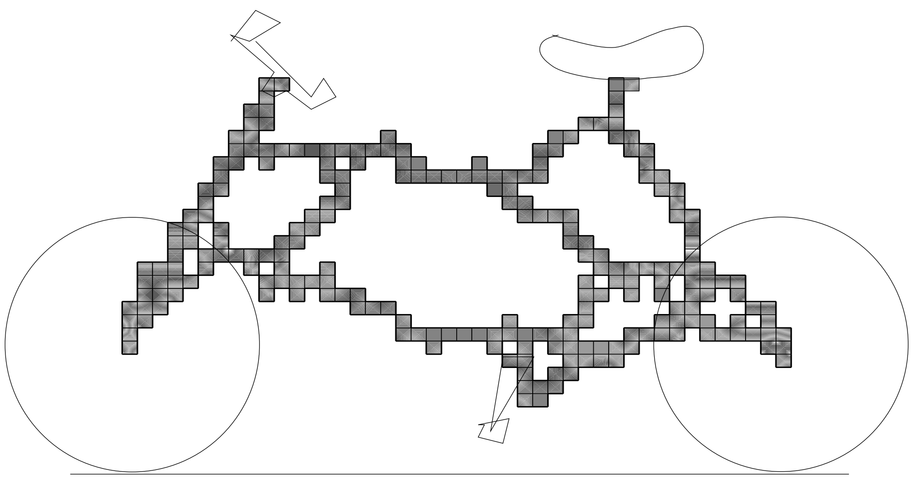


Figure 87 : *Seuls déplacements sont contraints.*

La structure trouvée est stable, et respecte les déplacements limites imposés mais on se rend compte qu'il peut y avoir des problèmes pour le champ de contraintes voir les éléments appartenant à la structure mais se touchant par un coin.

7.5.3 Un autre modèle de vélo

Le modèle que nous avons présenté dans la section précédente n'est pas très réaliste : il ne permet pas le passage de la roue ! On doit modifier le domaine de travail.

Le domaine de travail

Le nouveaau domaine de travail est défini figure 88. La matière est cette fois ci interdite dans la parties noires. On indique également sur cette figure les conditions aux limites, les chargements ainsi que les intensités correspondantes. Dans la première tentative, nous n'avons utilisé que les déplacements dus aux chargements successifs, ainsi la fonction performance utilisée est la même que celle décrite au paragraphe 7.5.1.

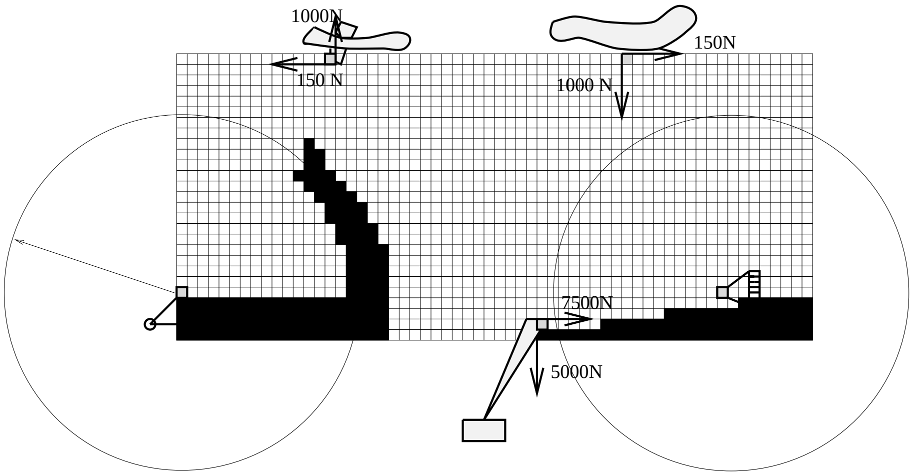


Figure 88 : *domaine de travail et chargement*

Premiers résultats

chge(F_x, F_y)	(150, 0)	(0, 1000)	(150, 0)	(0, -1000)	(7500, 0)	(0, -5000)
$(D_{Lim}, D_{Max}$	$(9.5, 6.8)$	$(136.5, 53.7)$	$(9.3, 9.23)$	$(32.0, 31.9)$	$(63.40, 59.7)$	$(269.8, 263.4)$
$(\sigma_{Lim}, \sigma_{Max})$	(,)	(,)	(,)	(,)	(,)	(,)

Table 7.5.3: charges, D_{Lim} imposés, et D_{Max} obtenus

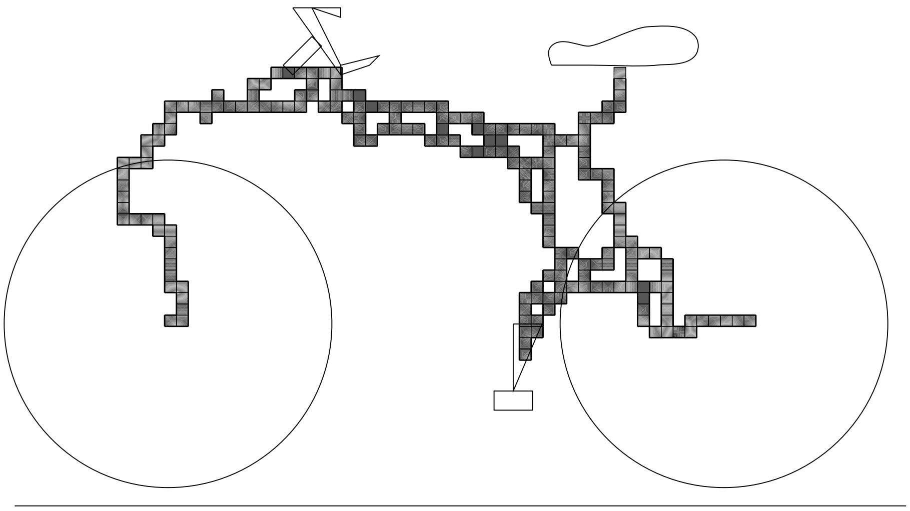


Figure 89 : *prise en compte des seuls déplacements*

Les résultats obtenus sont loin de ressembler à un vélo classique (on penserait plutôt à un tricycle). En étudiant le tableau 7.5.3 on remarque que les contraintes de déplacement que l'on a imposées sont vérifiées. Néanmoins, du point de vue mécanique la structure trouvée n'est pas stable. Ce résultat se confirme quand on examine le champ des contraintes mécaniques de la structure : les contraintes mécaniques maximales obtenues sont trop fortes, et le matériau doit casser.

Utilisation du champ de contraintes mécaniques

Compte-tenu des résultats obtenus précédemment, nous considérons dans la suite le même domaine de travail et les mêmes chargements; mais, la fonction performance est modifiée de manière à prendre en compte les contraintes mécaniques.

- Modification de la fonction performance
  La nouvelle fonction performance est donnée par :
  $$perf_{Multicharge} = \frac{1}{Aire + \sum_{i=1}^{n} \alpha_i (D^i_{max} - D^i_{Lim})^+ + \beta_i (C^i_{max} - C^i_{Lim})^+} \quad (7.5.2)$$
  Tout ce qui concerne les déplacements a déjà été introduit.
  C^i_{Max} est la norme maximale du champ de contraintes σ de la structure (cf section 6.1.1), C^i_{Lim} est la valeur limite imposée sur le champ de contraintes pour le chargement i, β_i est le paramètre de pénalisation correspondant au champ de contraintes mécaniques obtenu sous le chargement i. Les paramètres β_i sont traités comme les paramètres α_i dans la section précédentes.

- Résultat

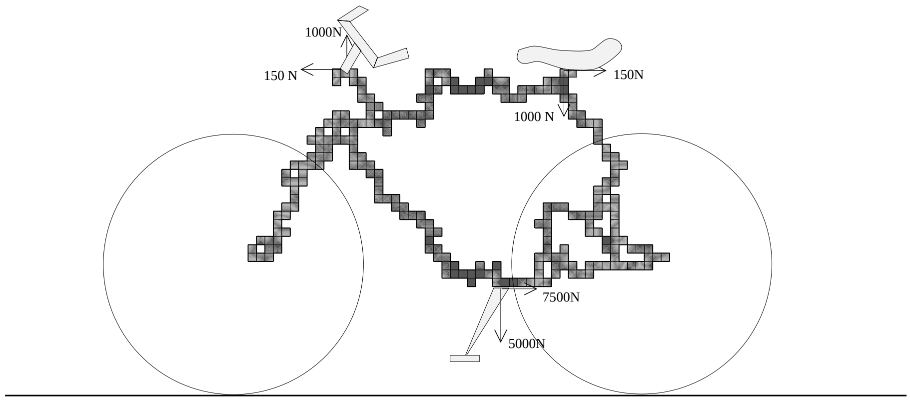


Figure 90 : *prise en compte des déplacements et du champ de contraintes*

Pour ce résultat toutes les contraintes sont respectées. Le résultat ressemble déjà d'avantage à un vélo. Néanmoins, il est clair que ce résultat gagnerait beaucoup à être régularisé par une méthode d'optimisation locale.

7.5.4 Un exemple d' évolution

Il est intéressant de voir comment la structure émerge au fil des générations. Le tableau 7.5.4 nous donne l'historique du meilleur vélo obtenu au cours des générations, pour l'essai qui a abouti à la figure 90

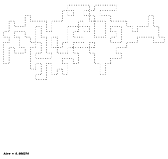
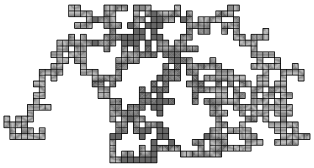
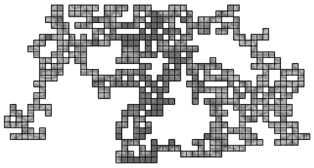
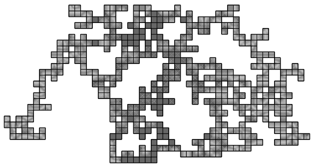
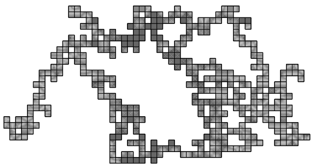
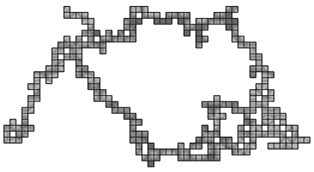
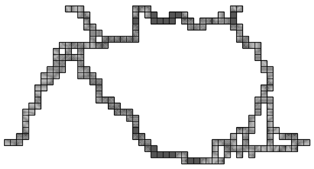
 génération 0	 300	 600	 900
 génération 1200	 1500	 2949	

Table 7.5.4 : Meilleur Vélo au cours des générations

- Pour la génération 0, les structures obtenues ne représentent pas grand chose. La structure générée aléatoirement ne relie même pas les points d'application des charges.

- A la génération 300 l'algorithme a trouvée une structure connectée reliant les différents points d'application des charges et les points où sont imposés les conditions aux limites. Les contraintes sont violées.

- La génération 600 donne une structure qui respecte toutes les contraintes.

- La génération 900 présente une structure respectant les contraintes mais qui est encore assez lourde.

- Les générations 1200 et 1500 donnent des structures de plus en plus légères. Les contraintes sont toujours respectées. La génération 1500 donne une forme proche de la forme finale.

On remarque qu'il a fallu autant de générations (1500 $\longrightarrow$ 2949) pour obtenir l'affinage final de la structure que pour obtenir sa forme approximative. Les algorithmes génétiques sont donc peu performants en fin de convergence.
Pour essayer de réduire le temps de calcul, on peut utiliser deux méthodes.

Optimisation locale

- Mutation des composantes
  C'est à ce niveau que la mutation du contour (qui accorde une plus grande probabilité de muter aux éléments entourant la structure) trouve son plein emploi. En effet, la structure est presque obtenue et il ne reste plus qu'à ajuster les derniers bits. La mutation du contour est l'outil *ad hoc* d'optimisation local. Les résultats obtenus le confirment : il faut deux fois moins de générations pour obtenir un vélo de même poids que celui de la figure 7.5.4 à partir de la même structure correspondant à celle de la génération 1500 du même essai.

- Couplage avec la méthode de variation de domaine
  Une autre idée serait de coupler l'algorithme avec les méthodes de variation de domaine [13] qui affineraient la structure très rapidement, une fois la topologie générale de la pièce dégrossie.

7.5.5 Autres cas de chargements

Si nous gardons les mêmes domaines de travail mais en considérant cette fois ci des cas de chargements réalistes, soit A la position du guidon, B la position de la selle, et C celle du pédalier, on suppose que l'on a trois chargements suivants.
Les 3 forces sont données par :

chargement	F_A	F_B	F_C
1	$\begin{pmatrix} -0.15 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0.15 \\ -1 \end{pmatrix}$	$\begin{pmatrix} 0 \\ -1 \end{pmatrix}$
2	$\begin{pmatrix} -0.2 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0. \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0.2 \\ -2 \end{pmatrix}$
3	$\begin{pmatrix} -0.6 \\ 0.7 \end{pmatrix}$	$\begin{pmatrix} 0.3 \\ -0.5 \end{pmatrix}$	$\begin{pmatrix} 0.5 \\ -1 \end{pmatrix}$

Le chargement 1 représente la position classique, le chargement 2 est la position du danseur le cycliste n'est pas assis, le troisième chargement représente la position “en montée”.

domaine 1 de travail

Les résultats obtenus avec les 3 cas de chargements représentés.

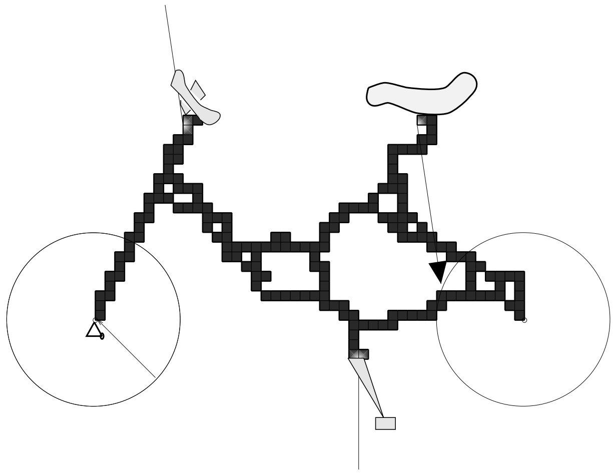
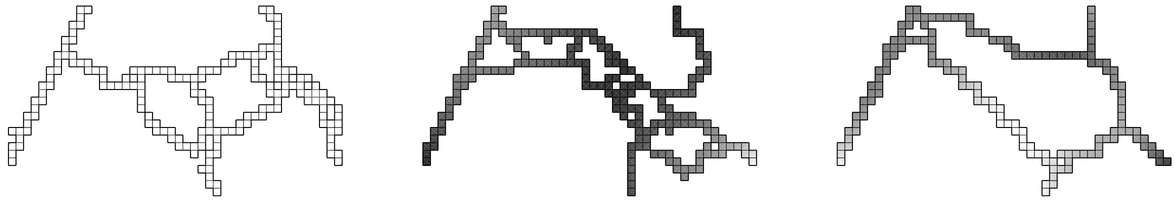


Figure 91 : *le trichargement(les 3 chargements à la fois) : le vélo optimal*



(a) (b) (c)

Figure 92 : *Les 3 cas de mono chargements correspondant a: la position classique b: la position du danseur c: la position de la montée.*

domaine 2 de travail

Les mêmes cas de chargements donnent pour le deuxième domaine de travail :

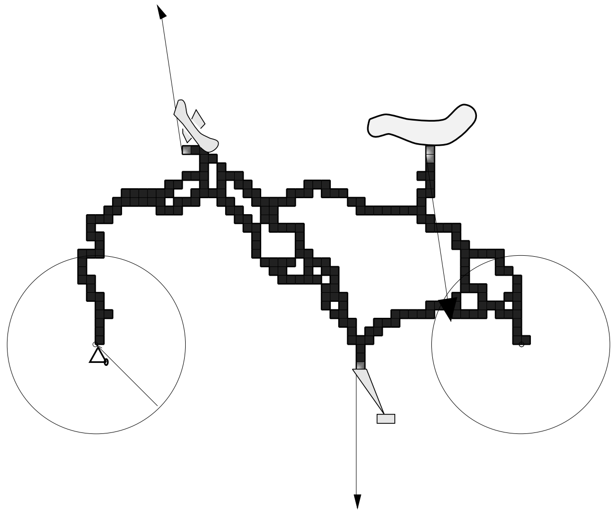


Figure 93 : *le trichargement : le vélo optimal*

7.5.6 Les résultats obtenus par La méthode d'homogéneisation

Les mêmes chargements sont utilisés par (Jouve [39]) et pour un maillage deux fois plus fin Les topologies obtenues pour les deux types de domaines de travail sont représentées sur les figures 94 et 94

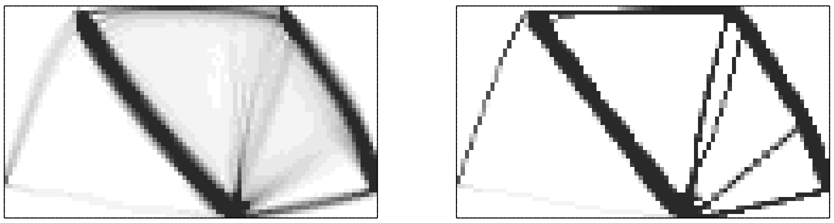


Figure 94 : *Résultats obtenus par homogénisation avant et aprés pénalisation pour le domaine 1 de travail.*

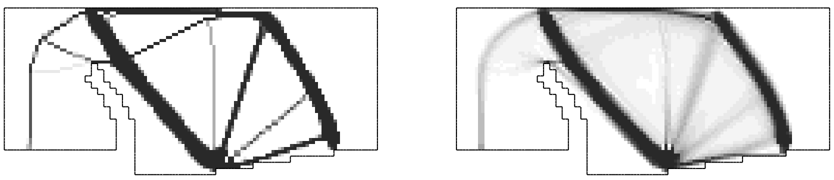


Figure 95 : *Résultats obtenus par homogénisation avant et aprés pénalisation pour le domaine 2 de travail.*

7.6 Chargement sur la frontière inconnue

Le problème présenté maintenant est très diffèrent des précédents. Les charges imposées sont de type pression sur le contour inconnu de la structure qui va se dessiner au cours du calcul. Nous considérons le problème du dôme sous pression : Nous cherchons la forme optimale de la structure dont le domaine de travail et les charges uniformes sont représentées figure 96. La figure 97 donne un exemple de configuration déformée, on voit que les chargements sont toujours normaux à la surface.

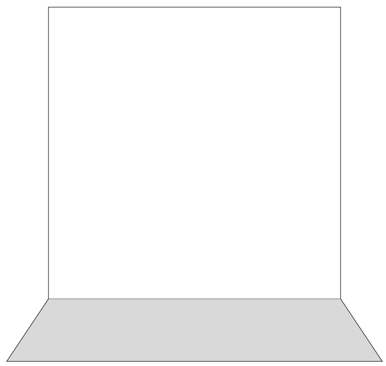


Figure 96 : *Exemple de problème sous pression domaine et chargement initial.*

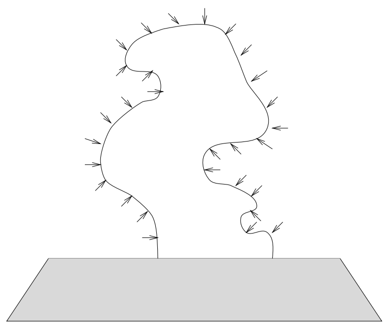


Figure 97 : *exemple de configuration déformée avec les chargements*

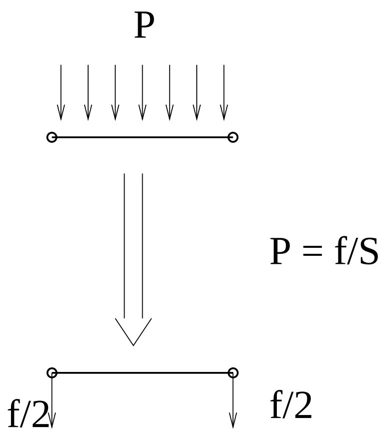


Figure 98 : *Influence de la pression*

La figure 98 schématise comment la pression est prise en compte dans le calcul par éléments finis (voir [39] pour les détails). Quand la pression est appliquée uniformément sur une arête, elle est modélisée par deux forces égales aux deux extrémités de cet arête. Si on n'impose pas d'existence de matière, la forme optimale obtenue tend vers la barre horizontale encastrée (une erreur de programmation a ainsi permis d'obtenir la solution présentée figure 99). C'est effectivement la forme la plus légère et de plus faible déplacement qui soit !

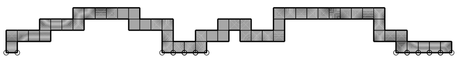


Figure 99 : *Résultat sans matière imposée*

Il faut donc bien sûr imposer l'existence d'un point contenant de la matière à un endroit de la structure autre que la frontière fixée. La figure 100 donne un résultat obtenu losqu'on impose de la matière au point (22,20) pour un maillage de taille 40×40.

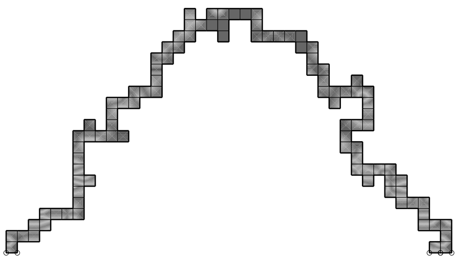


Figure 100 : *Résultat du dôme sous pression : on impose de la matière en un point central élevé.*

La forme obtenue ressemble effectivement à un dôme. Par contre, si on impose un chargement très important avec les mêmes limites sur le déplacement, la structure obtenue est différente : l'élément de structure vertical est déformé par la pression, mais ne se déplace pas beaucoup. La forme générale de la structure obtenue tend à se rapprocher de la barre plate, qui elle résiste à tout chargement en pression (figure 101).

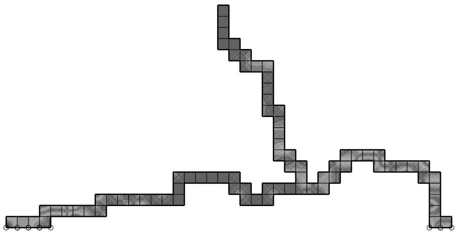


Figure 101 : *Autre résultat du dôme sous pression pour une compression importante*

Rappelons ici que la méthode d'homogénéisation ne peut considérer de conditions aux limites que sur la frontière du domaine de travail, elle est par conséquent incapable de traiter ce problème.

7.7 Résultats en élasticité non linéaire

Une autre limitation de la méthode d'homogénéisation est le modèle mécanique considéré , à ce jour, seul le modèle de l'élasticité linéaire peut être pris en compte.

Par opposition, l'approche par algorithmes génétiques est *a priori* capable de traiter tout modèle mécanique pour lequel il existe une simulation numérique fiable et robuste, dans la mesure où seuls des calculs directs du comportement mécanique des individus sont requis. Nous allons en faire la démonstration expérimentale, mais une fois de plus il faudra éviter le piège de la facilité consistant à utiliser le modèle mécanique comme une boîte “noire”.

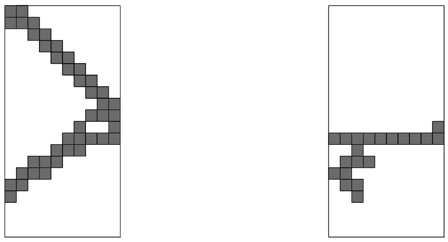
Nous nous plaçons maintenant dans le cadre du modèle non linéaire en grandes déformations présenté dans la section 6.1.2.

Premiers résultats désastreux

Nous reprenons donc exactement le contexte de tous les exemples précédents, à part le programme de simulation numérique qui est maintenant basé sur le modèle des grandes déformations.

En particulier, la fonction performance pénalisée décrite dans la section 6.2 est utilisée. La figure 102(b) donne les premiers résultats obtenus, la figure 102(a) représentant le résultat pour le modèle linéaire dans les mêmes conditions de chargement.

La forme obtenue par le modèle non linéaire est loin de la forme attendue bien que la contrainte sur le déplacement soit respectée. En particulier, on constate que la structure (b) n'est pas réaliste : le “porte-à-faux” qu'elle présente ne saurait résister longtemps. Cette impression est confirmée si on regarde la valeur maximale du champ de contraintes mécaniques pour les formes obtenues. On remarque que la forme présente une trop grande valeur pour le champ de contrainte mécanique, et la structure doit effectivement atteindre le seuil de rupture. Rappelons que le modèle non linéaire n'assure pas l'unicité de la solution pour un chargement donné, et certains des résultats peuvent présenter des contraintes plus fortes que d'autres. On notera enfin que sur un domaine aussi grossièrement maillé que celui utilisé pour la plaque cantilever, le calcul du champ de contraintes mécaniques doit être pris avec précautions.



(a) (b)

Figure 102 :

	(a)	(b)
D_{Max}	*0.022607*	*0.0199*
C_{Max}	*0.076*	*0.77*
Aire	*0.41*	*0.20*
	(a): Petites déformation.	*(b): grandes déformation.*

Le tableau 7.7 donne une idée des effets de la non-linéarité géométrique dans le cas de deux structures simples que sont la forme ">" et la barre droite. Les déplacements et les champs de contraintes maximaux sont donnés pour divers chargements. On trouve les déplacements auxquels on s'attendait; mais les valeurs du champ de contraintes mécaniques sont parfois étonnantes (par exemple pour $F = 910^{-3}$).

	Le cantilever ">"		la barre	
Load	D_{Max}	σ_{Max}	D_{Max}	C_{Max}
910^{-6}	0.0002360	0.0005968	0.04161	0.03035
910^{-5}	0.00235	0.0059	0.34499	0.28344
910^{-4}	0.02286	0.05312	0.80763	1.272
910^{-3}	1.06731	10.99	1.08208	1.11683
910^{-2}	1.249	2.0763	1.6871	4.3787
910^{-1}	2.9574	casse	3.177	casse

Table 7.7: Les effets de la non-linéarité sur deux formes de références.
D_{max} et C_{Max} sont respectivement le déplacement et la contrainte mécanique maximale.

Reformulation du problème

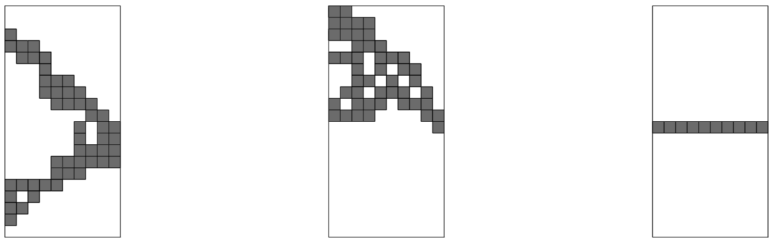
Ces résultats nous suggèrent de contraindre également le champ de contrainte mécanique dans la fonction performance. Il se peut que de belles structures échappent ainsi à l'algorithme, (par exemple pour $F = 910^{-3}$, la forme ">" ne pourra pas être trouvée avec cette discrétisation ... et ce programme éléments finis) mais on obtient à la place d'autres structures pouvant être assez similaires avec les mêmes propriétés mécaniques et a priori plus fiables en termes de contraintes mécaniques. La fonction performance que nous avons utilisée pour prendre en compte les contraintes est similaire à celle utilisée pour le vélo :

$$Perf = \frac{1}{Area + \alpha(D_{Max} - D_{Lim})^+ + \beta(\sigma_{Max} - \sigma_{Lim})^+}$$

(C_{Lim},C_{Max}) correspondent respectivement à la contrainte imposé et la contrainte maximale imposée sur la contrainte mécanique, β est un paramètre de pénalisation non constant.

La figure 103 représente les meilleures structures obtenues en utilisant cette fonction performance modifiée. Les résultats restent raisonnables tant que l'on impose une contrainte suffisamment forte sur la contrainte mécanique.

Afin de mettre en évidence les effets non-linéaires, nous avons utilisé différents chargements F en gardant un rapport F/D_{Lim} constant (D_{Lim} représente la contrainte sur le déplacement). Ces différents cas seraient strictement identiques dans le cadre linéaire.



(a) (b) (c)

Figure 103 :

	(a)	(b)	(c)
F	*0.009*	*0.018*	*0.09*
D_{Lim}	*0.22856*	*0.457*	*2.2856*
C_{Lim}	*0.53*	*1.0622*	*5.3*

Résultats optimaux des AGs en contraignant à la fois le déplacement $\longrightarrow \frac{F}{D_{Lim}}$ *est gardé constant.*

Les trois résultats sont effectivement très différents, tout en respectant totalement les contraintes. Il y a bien prise en compte de la non-linéarité lors de l'optimisation.

Conclusion

Nous avons présenté dans cette thèse une approche par algorithmes génétiques du problème de l'optimisation topologique des formes.

Après des rappels sur l'optimisation de formes (chapitre 1) et les algorithmes stochastiques (chapitre 2), nous avons présenté au chapitre 3 un premier travail d'optimisation géométrique de forme utilisant un algorithme génétique à codage réel. Le rappel de résultats théoriques au chapitre 4 a naturellement amené la représentation en tableau de bits que nous avons utilisée pour les problèmes d'optimisation topologique de forme. Cette représentation n'est cependant pas aussi évidente qu'elle en a l'air du point de vue des algorithmes génétiques : c'est ainsi que nous avons été amenés, au chapitre 5, à définir des opérateurs spécifiques pour le problème considéré Le chapitre 6 a été dédié à la définition de la fonction performance pour le problème de l'Optimum Design dans le contexte de l'élasticité. Nous avons envisagé diverses définitions possibles, et avons finalement abouti à définir le problème de l'optimisation topologiques des structures comme un problème de minimisation (du poids de la structure) sous contraintes (sur le déplacement maximal de la structure sous un ou plusieurs chargements donnés), que nous avons abordé en traitant les contraintes par un terme de pénalisation dans la fonction performance. La nécessité de faire varier l'amplitude de ce terme au cours de l'évolution a été soulignée par des essais numériques.

Cet arsenal a finalement été utilisé dans le chapitre 7, et les efforts déployés pour la mise au point de notre l'approche dans sa globalité ont été récompensés. Nous avons en effet obtenu des résultats originaux, qu'il n'est pas envisageable aujourd'hui d'atteindre par les méthodes classiques : obtention de plusieurs solution quasi-optimales, entre lesquelles l'expert pourra choisir selon des critères non modélisables ; résolution du problème avec chargement sur la frontière inconnue.

Mais, plus important encore, cela a permis la démonstration expérimentale, sur le modèle élastique non linéaire en grandes déformations, que cette méthode peut se généraliser à tout modèle mécanique pour lequel il existe une simulation numérique fiable et robuste.

Les leçons à tirer de cette réussite sont qu'il ne faut considérer comme boîtes noires ni les algorithmes génétiques – nécessité de redéfinir des opérateurs quand bien même les opérateurs usuels semblent s'appliquer – ni le domaine d'application – nécessité de prise en compte des contraintes mécaniques pour obtenir des solutions viables pour le problème du vélo et dans le contexte non-linéaire.

Parmi les directions de recherches évidentes ouvertes par ce travail, il faut citer le couplage entre cette méthode et les techniques de variation de domaines : nous avons vu que les solutions proposées par l'algorithme génétique ont le plus souvent quelques légers défauts qu'une méthode de variation de domaine pourrait corriger facilement. De plus, comme il a été vu sur le problème du vélo, il doit même être possible d'arrêter l'algorithme génétique avant sa convergence complète : si la forme a la bonne topologie, et est suffisamment proche de la forme optimale, les méthodes déterministes n'ont plus qu'à plonger dans l'optimum local le plus proche, ce qu'elles feront beaucoup mieux que les algorithmes génétiques.

Cependant, le gros défaut de l'approche par algorithmes génétique reste le coût (en nombre d'évaluations de la fonction performance nécessaires), dont il serait vain de faire abstraction. Certains auteurs [23] affirment que le principe même des algorithmes génétiques est contraire à toute idée de convergence rapide, et que tout tentative d'accélération du processus lui fera perdre les qualités de convergence globale. Il existe fort heureusement quelques contre-exemples [47], mais la plupart des exemples d'application confirment cependant ce point noir.

Une possibilité pour traiter ce problème est bien sûr la mise en oeuvre sur machines parallèles, qui est très facile conceptuellement : la parallélisation au niveau du calcul des performances, étape la plus coûteuse dans les problèmes réels, permet de réduire de manière quasi linéaire en fonction du nombre de processeurs (dans la limite de la taille de la population) le temps global de calcul.

Néanmoins, l'aspect coût-calcul de la méthode fait qu'il est difficile d'envisager son utilisation sur des gros problèmes tridimensionnels par exemple. En effet, la mise au point des nombreux paramètres de l'algorithme requiert de très nombreuses expériences numérique, et le caractère stochastique de la méthode fait qu'on ne peut rien déduire d'un ou deux essais. Or la mise en oeuvre de notre méthode sur des maillages plus conséquents a des conséquences sur le temps calcul à plusieurs niveaux:

- d'une part, la durée d'une simulation numérique augmente inéluctablement avec la taille du maillage ;
- d'autre part, la taille des chromosomes augmente elle aussi avec la taille du maillage. Or il est bien connu que la taille de la population et le nombre de générations nécessaires à la convergence augmentent également avec la taille du chromosome.

Le cumul de ces deux effets semble rendre le problème trop complexe pour être envisageable. Mais le second inconvénient dépend uniquement de la représentation utilisée, et peut probablement être amoindri en changeant de représentation. Il existe d'autres représentations possibles pour les formes, qui ne sont liées à aucun maillage [59]. Cette nouvelle direction de recherche semble dés à présent très prometteuse.

Bibliographie

[1] E. H. L. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines*. John Wiley and Sons, 1990.

[2] G. Allaire, E. Bonnetier, G. Francfort, and F. Jouve. Shape optimization by the homogenization method. *To appear in Nümerische Mathematik*, 1996.

[3] G. Allaire and F. Jouve, 1996.

[4] G. Allaire and R. V. Kohn. Optimal design for minimum weight and compliance in plane stress using extremal microstructures. *European Journal of Mechanics, A/Solids*, 12(6):839–878, 1993.

[5] J. Antonisse. A new interpretation of schema notation that overturns the binary encoding constraint. In J. D. Schaffer, editor, *ICGA89*, pages 86–91. Morgan Kaufmann, June 1989.

[6] 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, *EA95*. Springer Verlag, 1995. To appear.

[7] J. E. Baker. Adaptative selection methods for genetics algorithms. In J. J. Grefenstette editor, editor, *Proceedings of the First International Conference on Genetic Algorithms*. Lawrence Erlbaum Associates, 1985.

[8] J. E. Baker. Reducing biass and inefficiency in the selection algorithm. In J. J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 14–21. Lawrence Erlbaum Associates, 1987.

[9] D. Beasley, D.R. Bull, and R.R.Martin. An overview of genetic algorithms. 1: foundamentals. *Universly Computing*, 2:58–69, 1993.

[10] M. Bendsøe and N. Kikushi. Generating optimal topologies in structural design using a homogenization method. *Computer Methods in Applied Mechanics and Engineering*, 71:197–224, 1988.

[11] D. L. Calloway. Using a genetic algorithm to design binary phase-only filters for pattern recognition. In R. K. Belew and L. B. Booker, editors, *ICGA91*. Morgan Kaufmann, 1991.

[12] R. A. Caruna and J. D. Schaffer. Representation and hidden bias : Gray vs binary coding for genetic algorithms. In *Proceedings of ICML-88, International Conference on Machine Learning*. Morgan Kaufmann, 1988.

[13] J. Cea. Problems of shape optimum design. In E. J. Haug and J. Cea, editors, *Optimization of distributed parameter structures - Vol. II*, volume 50, pages 1005–1088. NATO Series, Series E, 1981.

[14] R. Cerf. *Une théorie assymptotique des algorithmes génétiques*. PhD thesis, Université de Montpellier II, March 1994.

[15] C.D. Chapman and et M.J. Jakiela K. Saitou. Genetic algorithms as an approach to configuration and topology design. *ASM J. Mech. Design*, 116:1005–1012, 1994.

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

[17] N. L. Cramer. A representation for the adaptative generation of simple sequential programs. In J. J. Grefenstette editor, editor, *Proceedings of the First International Conference on Genetic Algorithms*. Lawrence Erlbaum Associates, 1985.

[18] L. Davis. Adapting operator probabilities in genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the* 3^{rd} *International Conference on Genetic Algorithms*, pages 61–69, 1989.

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

[20] K. A. DeJong and J. Sarma. On decentralizing selection algorithms. In L. J. Eshelman, editor, *Proceedings of the* 6^{th} *International Conference on Genetic Algorithms*, pages 17–23. Morgan Kaufmann, 1995.

[21] C. Ghaddar, Y. Maday, and A. T. Patera. Analysis of a part design procedure. *Submitted to Nümerishe Mathematik*, 1995.

[22] D. E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, Reading, MA, 1989.

[23] D. E. Goldberg. Zen and the art of genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the* 3^{rd} *International Conference on Genetic Algorithms*, pages 80–85, 1989.

[24] D. E. Goldberg. A theory of virtual alphabets. In *Proceedings of the* 1^{st} *Parallel Problem Solving from Nature*, pages 13–22, 1990.

[25] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multi-modal function optimization. In J. J. Grefenstette, editor, *Proceedings of the* 2^{nd} *International Conference on Genetic Algorithms*, pages 41–49. Lawrence Erlbaum Associates, 1987.

[26] D.E. Goldberg and K. Deb. A compararive analysis of selection schemes used in genetic algorithms. In G.J.E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 69–93, San Mateo, CA, 1991. Morgan Kaufmann.

[27] J. J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Trans. on Systems, Man and Cybernetics*, SMC-16, 1986.

[28] J. J. Grefenstette. Incorporating problem specific knowledge in genetic algorithms. In Davis L., editor, *Genetic Algorithms and Simulated Annealing*, pages 42–60. Morgan Kaufmann, 1987.

[29] J. J. Grefenstette. Virtual genetic algorithms: First results. Technical Report AIC-95-013, Navy Center for Applied Research in Artificial Intelligence, February 1995.

[30] P. Grisvard. *Singularities in boundary value problems*, volume RMA 26. Masson Paris, 1992.

[31] P. Hajela. Genetic search-an approach to the nonconvex optimization problem. *AIAA Journal*, 28(7):1205–1210, July 1990.

[32] J. Holland. *Adaptation in natural and artificial systems.* University of Michigan Press, Ann Arbor, 1975.

[33] J. W. Hutchinson and F. I. Niordsen. Designing vibrating membranes. Technical report, Department of Solid Mechanics of the Technical University of Denmark, Lyngby, Denmark.

[34] J. P. Zolesio J. Sokolowski. *Introduction to Shape Optimization.* Springer-Verlag, 1992.

[35] G. Liepins J. T. Richardson, M. R. Palmer and M. Hilliard. Some guidelines for genetic algorithms with penalty functions. In *Proceedings of the 3^{rd} International Conference on Genetic Algorithms*, pages 191–197. Morgan Kaufmann, 1989.

[36] C. Z. Janikow and Z. Michalewicz. An experimental comparison of binary and floating point representations in genetic algorithms. In R. K. Belew and L. B. Booker, editors, *Proceedings of 4th International Conference on Genetic Algorithms*, pages 31–36. Morgan Kaufmann, July 1991.

[37] E. Jensen. *Topology Structural Design using Genetic Algorithms.* PhD thesis, Purdue University, 1992.

[38] J.A. Joines and C.R. Houck. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with gas. In Z. Michalewicz, J. D. Schaffer, H.-P. Schwefel, D. B. Fogel, and H. Kitano, editors, *First IEEE International Conference on Evolutionary Computation*, pages 579–584. IEEE Press, 1994.

[39] F. Jouve. *Modélisation mathématique de l'œil en élasticité non-linéaire*, volume RMA 26. Masson Paris, 1993.

[40] A. B. Kahng and B. R. Moon. Toward more powerful recombinations. In L. J. Eshelman, editor, *Proceedings of the* 6^{th} *International Conference on Genetic Algorithms*, pages 96–103. Morgan Kaufmann, 1995.

[41] C. Kane, F. Jouve, and M. Schoenauer. Structural topology optimization in linear and nonlinear elasticity using genetical algorithms. In *21st ASME Design Automatic Conference*, Boston MA, 1995. ASME Press.

[42] C. Kane and M. Schoenauer. A drum shape optimization by genetic algorithms. In *Complex Systems - Mechanism of Adaptation.* IOS Press and Ohmsha, Septembre 1994.

[43] C. Kane and M. Schoenauer. *Genetic operators for two-dimensional shape optimization*, volume 1063 of *Lecture Notes in Computer Science.* Springer-Verlag, 1995.

[44] R. V. Kohn and G. Strang. *Optimal Design and Relaxation of Variational Problems*, volume Comm. Pure Appl. Math, chapter I II III, pages 113–137, 139–182, 353–377.

[45] V. A. Kondratiev and O. A. Oleinik. On korn's inequalities. *C. R. Acad. Sci. Paris*, 308:483–487, 1989.

[46] N. Kikuchi M. Chirehdast, H-C Gea and P. Y. Papalambros. Structural configuration examples of an integrated optimal design process. *Journal of Mechanical Design*, Transactions of the ASME(116), 1994.

[47] J. Périaux M. Sefrioui and J.-G. Ganascia. Fast convergence thanks to diversity. In *Proceedings of the* 5^{th} *Annual Conference on Evolutionary Programming*, 1995.

[48] Z. Michalewicz. *Genetic Algorithms + Data Structure = Evolution Programs.* Springer Verlag, New York, 1992.

[49] Z. Michalewicz. *Genetic algorithms + data structures = evolution programs.* Artificial Intelligence. Springer-Verlag, New York, 1992.

[50] Z. Michalewicz. A survey of constraint handling techniques in evolutionary computation methods. In R. K. Belew and L. B. Booker, editors, Proceedings of the 4th Annual Conference on Evolutionary Programming, Cambridge, MA, 1995. MIT Press.

[51] F. Murat and J. Simon. Etude de problème d'optimum design. In *Proc. 7th IFIP conf.*, volume 41 of *Lecture notes in Computer sciences*, 1976.

[52] M. Rosenbluth A. Teller N. Metropolis, A. Rosenbluth and E. Teller. Equation of state calculationsby fast computing machines. *journal of Chem. Physics*, 21:1087–1092, 1953.

[53] O. Pironneau. *Optimal shape design for elliptic systems.* Springer-Verlag, 1984.

[54] N. J. Radcliffe and P. D. Surry. Fitness variance of formae and performance prediction. In D. Whitley and M. Vose, editors, *Foundations of Genetic Algorithms 3*, pages 51–72. Morgan Kaufmann, 1994.

[55] N.J. Radcliffe. Equivalence class analysis of genetic algorithms. *Complex Systems*, 5(2):183–205, April 1991.

[56] Rasmussen. *Structural Optimization-Status and Promise*, chapter Status and Promise of Optimum Design System in Denmark. Kamat, M., 1992.

[57] G. Rudolph. Convergence of non-elitit strategies. In Z. Michalewicz, J. D. Schaffer, H.-P. Schwefel, D. B. Fogel, and H. Kitano, editors, *First IEEE International Conference on Evolutionary Computation*, pages 63–66. IEEE Press, 1994.

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

[59] M. Schoenauer. Representations for evolutionary optimization and identification in structural mechanics. In J. Périaux and G. Winter, editors, *Genetic Algorithms in Engineering and Computer Sciences*. John Wiley, 1995.

[60] M. Schoenauer and S. Xanthakis. Constrained GA optimization. In S. Forrest, editor, *Proceedings of the* 5^{th} *International Conference on Genetic Algorithms*, pages 573–580. Morgan Kaufmann, 1993.

[61] H.-P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, New-York, 1981 – 1995 (2nd ed.).

[62] M. Sebag and M. Schoenauer. Controling crossover through inductive learning. In Y. Davidor, H.-P. Schwefel, and R. Manner, editors, *Proceedings of the* 3^{rd} *Conference on Parallel Problems Solving from Nature*, pages 209–218. Springer-Verlag, LNCS 866, October 1994.

[63] W. M. Spears. Adapting crossover in evolutionary algorithms. In J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, editors, *Proceedings of the* 4^{th} *Annual Conference on Evolutionary Programming*, pages 367–384. MIT Press, March 1995.

[64] D. Vanderbilt and S. G. Louie. A monte carlo simuleated annealing approach to optimization over continuous variables. *Journal of Computational Physics*, 56:259–271, 1984.

[65] J. A. Vasconcelos. *Optimisation de forme de structures électromagnétiques*. PhD thesis, Ecole Centrale de Lyon, 1994.

[66] M. Vose and G. Lieping. Punctuated equilibria in genetic search. *Complex Systems*, 5:31–44, 1991.

[67] W.M.Spears and K. DeJong. On the virtus of parametrical uniform crossover. In R. K. Belew and L.B. Booker, editors, *Proceedings of the Forth International Conference on Genetic Algorithms*, pages 230–236, San Mateo, CA, 1991. Morgan Kaufmann.

[68] A. A. Zhigljavski, editor. *Theory of global random search.* Chap. 5, Kluwer Academic, 1991.