Chapitre 7

OPTIMISATION EVOLUTIONNAIRE

Introduction[1]

Ce chapitre présente une méthode radicalement différentes de toutes les précédentes de résolution de problèmes d'optimisation topologique de formes. Cette méthode est basée sur l'utilisation des algorithmes dits “évolutionnaires”, algorithmes stochastiques d'optimisation globale. Une première partie introduit les algorithmes, hors de tout contexte d'optimisation de formes, cependant que la deuxième partie présente leur application à l'optimisation topologique des structures.

7.1 Les algorithmes évolutionnaires

Les phénomènes physiques ou biologiques ont été à la source de nombreux algorithmes s'en inspirant plus ou moins librement. Ainsi les réseaux de neurones artificiels s'inspirent du fonctionnement du cerveau humain, l'algorithme de recuit simulé de la thermodynamique, et les *algorithmes évolutionnaires* (AEs) (dont les plus connus sont les *algorithmes génétiques*) de l'évolution darwinienne des populations biologiques.

Mais les algorithmes évolutionnaires sont aussi des méthodes stochastiques d'optimisation globale. Et la souplesse d'utilisation de ces algorithmes pour des fonctions objectifs non régulières, à valeurs vectorielles, et définies sur des espaces de recherche non-standard (e.g. espaces de listes, de graphes, ...) permet leur utilisation pour des problèmes d'optimisation de formes dans des cas qui sont pour le moment hors d'atteinte des méthodes déterministes plus classiques décrites par ailleurs dans le reste de ce cours.

Cette section présente succinctement les algorithmes évolutionnaires. On n'hésitera pas à consulter les ouvrages [13, 3, 2] pour de plus amples détails.

---

[1] Chapitre rédigé par Marc Schoenauer

7.1.1 Le darwinisme artificiel

Résumer la théorie de Darwin en quelques mots est bien sûr impossible. Disons que la (petite) partie qui est (grossièrement) imitée et caricaturée lors de la conception des AEs est basée sur l'idée que l'apparition d'espèces adaptées au milieu est la conséquence de deux phénomènes : d'une part la *sélection naturelle* imposée par le milieu – *les individus les plus adaptés survivent et se reproduisent* – et d'autre part des *variations non dirigées* du matériel génétique des espèces. Ce sont ces deux principes qui sous-tendent les algorithmes évolutionnaires.

Précisons enfin que le paradigme darwinien qui sous-tend ces algorithmes ne saurait en aucun cas être une justification pour l'emploi de ces techniques – pas plus que le vol des oiseaux ne peut justifier l'invention de l'avion !)[2]. Il y a maintenant suffisamment d'exemples de succès pratiques (a posteriori) de ces algorithmes pour ne pas avoir à recourir à ce type d'argument.

7.1.2 Définitions et notations

Soit à optimiser une fonction $\mathcal{F}$ à valeurs réelles définie sur un espace Ω, supposé muni d'une distance d. Le parallèle avec l'évolution naturelle a entraîné l'apparition d'un vocabulaire spécifique (et qui peut paraître légèrement ésotérique) :

- La fonction objectif $\mathcal{F}$ est appelée fonction *performance*, ou fonction *d'adaptation* (*fitness* en anglais) ;
- Les points de l'espace de recherche Ω sont appelés des *individus* ;
- Les P-uplets d'individus sont appelés des *populations* ;
- On parlera d'une *génération* pour la boucle principale de l'algorithme ;

7.1.3 Le squelette

La pression de l'"environnement" est simulée à l'aide de la fonction d'adaptation $\mathcal{F}$, et le principe darwinien *Les plus adaptés survivent et se reproduisent*, est implanté de la manière suivante :

- **Initialisation** de la *population* Π_0 en choisissant P individus dans Ω, généralement par tirage aléatoire avec une probabilité uniforme sur Ω ;
- **Évaluation** des individus de Π_0 (i.e. calcul des valeurs de $\mathcal{F}$ pour tous les individus) ;
- La génération i construit la population Π_i à partir de la population Π_{i-1} :
  - **Sélection** des individus les plus performants de Π_{i-1} au sens de $\mathcal{F}$ (*les plus adaptés se reproduisent*) ;
  - Application (avec une probabilité donnée) des **opérateurs de variation** aux *parents* sélectionnés, ce qui génère de nouveaux individus, les *enfants* ; on parlera de *mutation* pour les opérateurs unaires, et de *croisement* pour les opérateurs binaires (ou n-aires) ;
  - **Évaluation** des enfants ;

---

[2] on peut même utiliser ce type d'algorithme en n'étant pas un convaincu du darwinisme, d'où d'ailleurs l'emploi du néologisme "évolutionnaire", inspiré de l'anglais *evolutionary*.

  - **Remplacement** de la population Π_i par une nouvelle population créée à partir des enfants et/ou des "vieux" de la population Π_i au moyen d'une sélection darwinienne (*les individus les plus adaptés survivent*).
- L'évolution stoppe quand le niveau de performance souhaité est atteint, ou qu'un nombre fixé de générations s'est écoulé sans améliorer l'individu le plus performant.

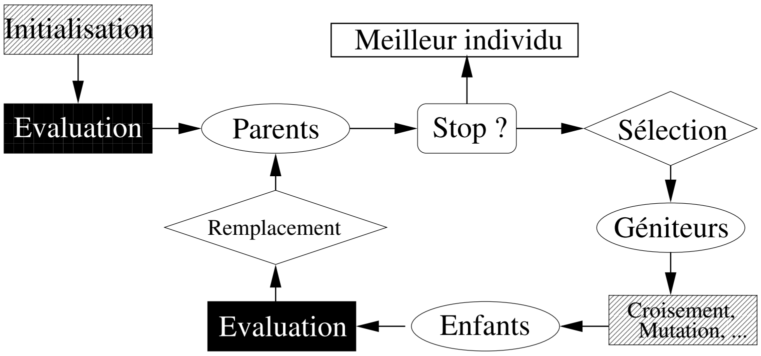


FIG. 7.1 – *Squelette d'un algorithme évolutionnaire*

Il est important de noter que le coût CPU essentiel de ces algorithmes provient de l'étape d'évaluation (calcul des performances) : les tailles de populations sont de l'ordre de quelques dizaines, le nombre de générations de quelques centaines, ce qui donne lieu à quelques dizaines de milliers de calculs de $\mathcal{F}$.

La suite de cette section va détailler les principaux composants d'un algorithme évolutionnaire, en en donnant des exemples concrets.

On notera tout d'abord que l'on peut répartir les diverses étapes de l'algorithme de la section précédente en deux groupes : les étapes implantant le darwinisme artificiel (sélection et remplacement), qui ne dépendent pas de la *représentation* choisie, c'est-à-dire de l'espace de recherche ; et les étapes qui sont intimement liées au choix de l'espace de recherche : l'initialisation et les opérateurs de variation ne dépendent que du choix de la représentation (et pas vraiment de la fonction objectif), et l'évaluation est totalement spécifique au problème traité.

7.1.4 Choix de l'espace de recherche

Il s'agit de la composante principale de l'algorithme – qui est même en fait préalable aux autres. Dans de nombreux cas, l'espace de recherche est totalement déterminé par le problème (la fonction objectif). Mais il est toujours possible de transporter son problème dans un espace habilement choisi (changement de "variables") oú sa résolution sera plus aisée. Cet espace, oú seront appliqués les opérateurs de variation, est alors appelé *espace génotypique*, et l'espace de recherche initial, dans lequel est calculée la performance des individus, est appelé *espace phénotypique*.

Nous allons maintenant donner deux exemples d'espaces de recherche parmi les plus utilisés – et détaillerons ensuite les composantes de l'algorithme qui dépendent de la représentation dans ces deux cas particuliers. Il ne faut toutefois pas perdre de vue que la puissance des algorithmes évolutionnaires vient de leur capacité à optimiser des fonctions définies sur des espaces de recherche bien plus variés que ces deux espaces – ce sera l'objet de la suite du chapitre (voir sections 7.2.4 et 7.2.5).

LES CHAÎNES DE BITS (*bitstring* en anglais).
L'espace de recherche est ici $\Omega = \{0,1\}^N$. Historiquement (voir section 7.1.10) il s'agit de l'école des algorithmes génétiques, et la justification de l'utilisation intensive de cet espace de recherche particulier était fondé à la fois sur un parallèle encore plus précis avec la biologie (une chaîne de bits étant assimilée à un chromosome) et sur des considérations théoriques qui ne seront pas détaillées ici (voir les références de la section 7.1.10 à ce sujet). Ce contexte reste toutefois très utilisé, et permet également une présentation aisée des divers composants de l'algorithme.

LES VECTEURS DE RÉELS :
C'est bien sûr le cas le plus fréquent en calcul numérique : Ω est un sous-ensemble de $\mathbb{R}^n$, borné ou non. On parle alors aussi d'*optimisation paramétrique*.

7.1.5 Initialisation

Le principe général de l'initialisation est d'échantillonner le plus uniformément possible l'espace de recherche Ω.

Dans le cas des chaînes de bits, chaque bit de chaque individu est tiré égal à 0 ou à 1 avec un probabilité 0.5.

Dans le cas de l'optimisation paramétrique, si $\Omega = \prod[a_i, b_i]$ (cas borné), on tire uniformément chaque coordonnée dans l'intervalle correspondant. Par contre, si Ω n'est pas borné, il faut faire des choix. On pourra par exemple tirer mantisses et exposants uniformément.

7.1.6 Opérateurs de variation

C'est durant cette étape que de nouveaux individus sont crées à partir des individus préalablement sélectionnés. On distingue les opérateurs de croisement (binaires, ou plus généralement n-aires) et les opérateurs de mutation, unaires. A noter que cette étape est toujours **stochastique**, c'est à dire que le résultat de l'application d'un opérateur dépend de tirages aléatoires.

Le croisement

L'idée générale du croisement est l'échange de *matériel génétique* entre les *parents* : si deux parents sont plus performants que la moyenne, on peut espérer que cela est dú à certaines parties de leur *génotype*, et que certains des enfants, recevant les"bonnes" parties de leurs deux parents, n'en seront que plus performants. Ce raisonnement, trivialement valable pour des fonctions performance linéaires, est extrapolé (et expérimentalement vérifié) à une classe plus étendue de fonctions, sans que les résultats théoriques aujourd'hui disponibles ne permettent de délimiter précisément la classe de fonctions pour lesquelles le croisement est utile.

Signalons enfin que tous les individus sélectionnés ne sont pas forcément croisés. Le croisement – comme tous les opérateurs de variation, est appliqué avec une certaine probabilité notée P_c.

Dans le cadre des **chaînes de bits**, les divers opérateurs de croisement échangent des bits (à position fixée) entre les parents. La figure 7.2 donne l'exemple du croisement à point. Un autre croisement consiste à tirer au sort pour chaque position (avec probabilité 0.5) de quel parent proviendra le bit correspondant chez chaque enfant.

$$\left.\begin{array}{l}(b_1,\ldots,b_N)\\(c_1,\ldots,c_N)\end{array}\right\} \xrightarrow{P_c} \left\{\begin{array}{l}(b_1,\ldots,b_l,c_{l+1},\ldots,c_N)\\(c_1,\ldots,c_l,b_{l+1},\ldots,b_N)\end{array}\right.$$

FIG. 7.2 – Croisement à un point : l'entier l est tiré uniformément dans $[1, N-1]$, et les deux moitiés des chromosomes sont échangées.

Dans le cadre de l'**optimisation paramétrique**, on peut bien entendu appliquer les mêmes opérateurs d'échange de coordonnées. On peut également – et c'est en général bien plus efficace – "mélanger" les deux parents par une combinaison linéaire. On parle alors de *croisement arithmétique* :

$$(\vec{X},\vec{Y}) \xrightarrow{p_c} \alpha\vec{X} + (1-\alpha)\vec{Y}, \alpha = U[0,1]$$

ou

$$(\vec{X},\vec{Y}) \xrightarrow{p_c} \alpha_i X_i + (1\vec{\,}- \alpha_i)Y_i, \alpha_i = U[0,1]$$

La première version revient à choisir l'enfant uniformément sur le segment $[\vec{X}\vec{Y}]$ alors que la deuxième revient à tirer l'enfant uniformément sur l'hypercube dont $[\vec{X}\vec{Y}]$ est une diagonale. Remarquons que l'échange de coordonnées revient à choisir comme enfant un des sommets de cet hypercube. Et signalons qu'on peut également choisir les coefficients des combinaisons linéaires dans un intervalle plus grand (e.g. $[-0.5, 0.5]$) afin de limiter la contractance de l'opérateur de croisement.

La mutation

L'idée directrice de la mutation est de permettre de visiter tout l'espace. Les quelques résultats théoriques de convergence des algorithmes évolutionnaires ont d'ailleurs tous comme condition l'**ergodicité** de la mutation, c'est-à-dire le fait que tout point de l'espace de recherche peut être atteint en un nombre fini de mutations.

Dans le cadre des **chaînes de bits**, on modifie aléatoirement certains bits. Par exemple, pour chaque parent (ou enfant du croisement), et pour chaque position l,

$$(b_1, b_2, \ldots, b_N) \xrightarrow{p_m} (b_1, b_2, \ldots, 1-b_l, b_{l+1}, \ldots, b_N)$$

Attention, la probabilité p_m est ici une probabilité **par bit** et non par individu.

Une autre possibilité est de prédéfinir un nombre de bits à modifier (généralement 1) et de choisir ensuite au hasard une position dans chaque individu et de modifier le bit correspondant. Cet opérateur est lui appliqué avec un probabilité P_m **par individu**, bien sûr.

Dans le cadre de l'**optimisation paramétrique**, la mutation la plus employée est la mutation *gaussienne*, qui consiste à rajouter un bruit gaussien à chaque variable (l'opérateur étant appliqué avec une certaine probabilité par individu P_m).

$$X_i := X_i + N(0, \sigma^2)$$

Tout l'art est alors dans le choix de la variance σ du bruit gaussien utilisé. On peut évidemment demander à l'utilisateur de fixer cette valeur. Mais des études théoriques sur des fonctions simples ont démontré que cette valeur devrait décroître au fil des générations – et il devient difficile de fixer une suite de valeurs décroissante qui soit synchrone avec l'éventuelle convergence de l'algorithme pour des fonctions quelconques.

L'idée originale pour résoudre ce problème est due aux pères des stratégies d'évolution (I. Rechenberg et H.-P. Schwefel, voir section 7.1.10). Elle consiste à considérer ces σ comme des variables supplémentaires, et à les faire également évoluer (via croisement et mutation !).

Si l'espace phénotypique initial est, par exemple, $\prod[a_i, b_i]$, l'espace génotypique sera $\prod[a_i, b_i] \times \mathbb{R}^+$, chaque individu se voyant muni de sa propre variance σ. La mutation devient alors

$$\begin{aligned} &\sigma := \sigma \; exp(\tau N(0,1)) \\ &X_i := X_i + N(0, \sigma) \end{aligned}$$

La variance σ est modifiée par une perturbation log-normale – elle doit rester positive, et son évolution doit être symétrique par rapport à un puisqu'elle intervient ensuite de manière multiplicative.

L'idée est qu'un individu ayant de bonnes performances après plusieurs mutations n'est pas arrivé oú il est par hasard, et doit avoir une valeur de σ **adaptée** à la zone de l'espace de recherche oú il se trouve. Signalons qu'il existe une variante dans laquelle une variance est attachée à chaque variable. Enfin, dans la variante la plus générale, une matrice de corrélation complète évolue avec chaque individu.

Application des opérateurs de variation

Tous les opérateurs de variation ne sont pas appliqués systématiquement à tous les individus à chaque génération. Le schéma le plus courant est d'appliquer *séquentiellement* un opérateur de croisement, puis un opérateur de mutation, chacun avec une probabilité donnée (un paramètre de l'algorithme, laissé au choix de l'utilisateur). On notera p_c et p_m les probabilités respectives d'application du croisement et de la mutation.

Il est par contre relativement fréquent de disposer de plusieurs opérateurs de chaque type (croisement ou mutation) et de vouloir les utiliser au sein du même algorithme (e.g. le croisement par échange de coordonnées et le croisement barycentrique dans le cas de l'optimisation réelle). Il faut alors introduire de nouveaux paramètres, à savoir l'importance relative de chaque opérateur par rapport aux autres (e.g. on veut faire 40% de croisements par échange de coordonnées, et 60% de croisement barycentriques).

Il est bien sûr possible d'imaginer bien d'autres schémas d'application des opérateurs de variation, ainsi d'ailleurs que d'autres types d'opérateurs ni unaires ni binaires (alors appelés opérateurs d'*orgie*).

7.1.7 Procédures de sélection

La partie darwinienne de l'algorithme comprend les deux étapes de **sélection** et de **remplacement**. Répétons que ces étapes sont totalement indépendantes de l'espace de recherche.

D'un point de vue technique, la différence essentielle entre l'étape de sélection et l'étape de remplacement est qu'un même individu peut être sélectionné plusieurs fois durant l'étape de sélection (et avoir donc plusieurs enfants) alors que durant l'étape de remplacement, chaque individu est sélectionné une fois (et il survit) ou pas du tout (et il disparaît à jamais). Enfin, comme il a déjà été dit, la procédure de remplacement peut impliquer soit les enfants seulement, soit également la population précédente dans son ensemble. Ceci mis à part, les étapes de sélection et de remplacement utilisent les mêmes procédures de choix des individus, dont les plus utilisées vont maintenant être passées en revue.

On distingue deux catégories de procédures de sélection ou de remplacement (par abus de langage, nous appellerons sélection les deux types de procédures) : les procédures déterministes et les procédures stochastiques.

Sélection déterministe

On sélectionne les meilleurs individus (au sens de la fonction performance). Si plus de quelques individus doivent être sélectionnés, cela suppose un tri de l'ensemble de la population – mais cela ne pose un problème de temps calcul que pour des très grosses tailles de population.

Les individus les moins performants sont totalement éliminés de la population, et le meilleur individu est toujours sélectionné – on parle alors d'*élitisme*.

Sélection stochastique

Il s'agit toujours de favoriser les meilleurs individus, mais ici de manière stochastique, ce qui laisse une chance aux individus moins performants. Par contre, il peut arriver que le meilleur individu ne soit pas sélectionné, et qu'aucun des enfants n'atteigne un performance aussi bonne que celle du meilleur parent ...

Le **tirage de roulette** est la plus célèbre des sélections stochastiques. Supposant un problème de maximisation avec uniquement des performances positives, elle consiste à donner à chaque individu une probabilité d'être sélectionné proportionnelle à sa performance. Une illustration de la roulette est donnée figure 7.3 : on lance la boule dans la roulette, et on choisit l'individu dans le secteur duquel la boule a fini sa course.

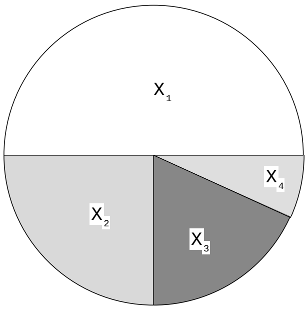


FIG. 7.3 – Sélection par tirage de roulette : cas de 4 individus de performances respectives 50, 25, 15 et 10. Une boule est "lancée" dans cette roulette, et l'individu dans le secteur duquel elle s'arrête est sélectionné.

Le tirage de roulette présente toutefois de nombreux inconvénients, en particulier reliés à l'échelle de la fonction performance : alors qu'il est théoriquement équivalent d'optimiser $\alpha\mathcal{F} + \beta$ et $\mathcal{F}$ pour tout $\alpha > 0$, il est clair que le comportement de la sélection par roulette va fortement dépendre de α dans ce cas. C'est pourquoi cette sélection est presque totalement abandonnée aujourd'hui.

La **sélection par le rang** consiste à faire une sélection en utilisant une roulette dont les secteurs sont proportionnels aux rangs des individus. La variante linéaire utilise directement le rang (P pour le meilleur, 1 pour le moins bon, pour une population de taille P), les variantes polynômiales remplaçant ces valeurs par $\frac{i}{P}^{\alpha}, \alpha > 0$. Seuls comptent alors les position relatives des individus entre eux, et non les valeurs – arbitraires – de la fonction $\mathcal{F}$.

La **sélection par tournoi** n'utilise aussi que des comparaisons entre individus – et ne nécessite même pas de tri de la population. Elle possède un paramètre T, taille du tournoi. Pour sélectionner un individu, on en tire T uniformément dans la population, et on sélectionne le meilleur de ces T individus. Le choix de T permet de faire varier la *pression sélective*, c'est-à-dire les chances de sélection des plus performants par rapport aux plus faibles. A noter que le cas $T = 2$ correspond, en espérance et au premier ordre en fonction de P, à la sélection par le rang linéaire.

7.1.8 Sélections multi-critères

De nombreux problèmes du monde réel sont en fait des problèmes multi-critère (et l'optimisation topologique de formes n"échappe pas à la règle, voir section 7.2.2).

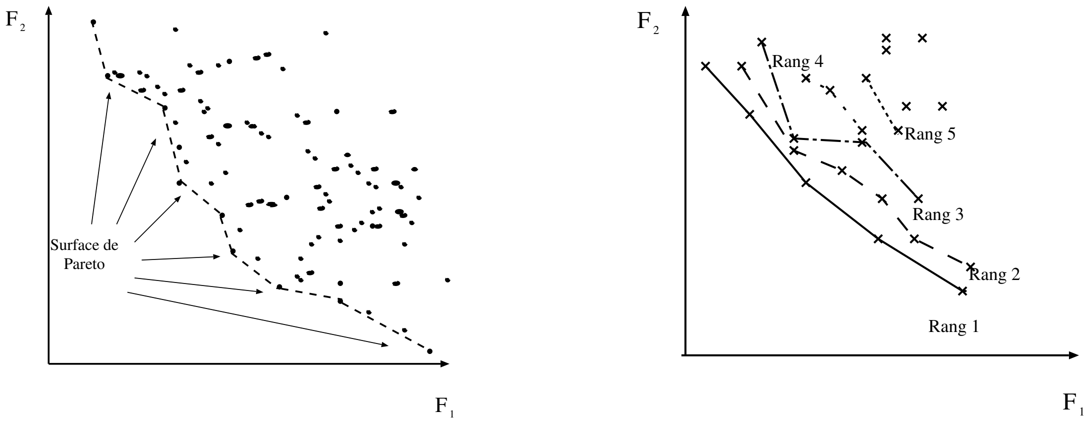
Front de Pareto

Dans un problème multi-critère, on cherche à optimiser plusieurs objectifs contradictoires. On appellera *front de Pareto* du problème l'ensemble des points de l'espace de recherche tels qu'il n'existe aucun point qui est strictement meilleur qu'eux sur tous les critères simultanément. Il s'agit de l'ensemble des meilleurs compromis réalisables entre les objectifs contradictoires.

Plus formellement, soient $\mathcal{F}_1, \ldots, \mathcal{F}_n$ les objectifs dont on suppose qu'on cherche à les minimiser sur l'espace de recherche Ω.
Définition : Soient x et y deux points de Ω. On dira que x *domine par* y au sens de Pareto, noté $x \succ y$ si

$$\forall i \in [1,n], \mathcal{F}_i(x) \leq \mathcal{F}_i(y)$$
$$\exists i_0 \in [1,n], \mathcal{F}_i(x) < \mathcal{F}_i(y)$$

La figure 7.1.8-a donne un exemple de front de Pareto : l'ensemble de l'espace de recherche est représenté dans l'espace des objectifs, et les points extrémaux pour la relation de dominance au sens de Pareto forment le front de Pareto du problème (notez que l'on n'est pas toujours dans une situation aussi régulière que celle présentée figure 7.1.8, et que le front de Pareto peut être concave, discontinu, ...)



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

(b) Une population donnée est partiellement ordonnée par la relation de dominance au sens de Pareto.

FIG. 7.4 – *Front de Pareto et rang d Pareto pour un problème de minimisation de deux objectifs.*

Sélections de Pareto

Lorsque l'on s'intéresse à l'optimisation multi-critère au sens de Pareto, il est possible de remplacer l'étape de sélection telle qu'elle a été décrite dans la section 7.1.7 par des sélections basées sur la notion de dominance au sens de Pareto. Cependant, la relation d'ordre définie par la dominance étant une relation d'ordre partiel, il faudra rajouter une procédure de choix secondaires entre individus non comparables au sens de Pareto.
Critères de Pareto :

Le plus utilisé des critères de sélection au sens de Pareto est le *rang de Pareto*, défini itérativement de la manière suivante : les individus non-dominés de la population courante sont dits de rang 1. Ils sont retirés de la population, et la procédure est itérée pour obtenir les rang 2, 3, ...Les individus de rang 1 (i.e. non dominés) vont approcher le front de Pareto du problème.

La *force de domination* est le nombre d'individus de la population courante qu'un individu donné domine : plus un individu domine d'autres individus, plus il est intéressant à conserver comme géniteur. De même, la *profondeur au sens de Pareto* d'un individu est le nombre d'autres individus de la population qui dominent un individu donné – un individu est d'autant plus intéressant à conserver qu'il est dominé par un petit nombre d'autres collègues.

Critères de diversité :

Les critères précédent ne permettent pas d'ordonner toute la population – en fait, assez rapidement, les individus de la population courante qui vont approcher le front de Pareto du problème seront non comparables entre eux. Il faut donc ajouter un autre critère pour choisir entre eux. Les différents critères proposés favorisent la diversité. Le plus populaire aujourd'hui est la *distance de peuplement*, définie comme suit :

- pour chaque objectif i, on ordonne la population par valeurs croissantes de l'objectif
- pour chaque individu p, on définit d_i, *distance de peuplement partielle selon l'objectif* i comme la somme des distances de p à ses deux plus proches voisins dans la liste ordonnée
- la distance de peuplement totale D_p est donnée par la somme, sur l'ensemble des objectifs, des distances partielles.

Tournois au sens de Pareto :

Pour sélectionner un géniteur a l'aide des outils définis ci-dessus, on utilise un *tournoi* (voir section 7.1.7) en comparant tout d'abord les individus selon le critère de Pareto choisi, puis, en cas d'égalité, suivant le critère de diversité retenu.

7.1.9 Les moteurs d'évolution

On regroupe sous ce nom les ensembles sélection/remplacement, qui ne peuvent être dissociées lors des analyses théoriques du darwinisme au sein des algorithmes évolutionnaires. Un moteur d'évolution est donc la réunion d'une procédure de sélection et d'une procédure de remplacement. Toute combinaison des procédures présentées plus haut (et de bien d'autres encore) est licite. Toutefois, certaines combinaisons sont plus souvent utilisées, que ce soit pour des raisons historiques, théoriques ou expérimentales. Pour cette raison, les noms donnés sont souvent les noms des école historiques qui les ont popularisées – mais gardons à l'esprit que ces schémas sont totalement indépendants de l'espace de recherche, alors que nous verrons que les écoles historiques travaillaient sur des espaces de recherche bien précis.

ALGORITHME GÉNÉTIQUE GÉNÉRATIONNEL (*GGA*)

Ce moteur utilise une sélection stochastique pour sélectionner exactement P parents (certains parents peuvent donc être sélectionnés plusieurs fois, d'autres pas du tout). Ces P parents donnent ensuite P enfants par application des opérateurs de variation (avec probabilité donnée). Enfin, ces P enfants remplacent purement et simplement les P parents pour la génération suivante.

Algorithme génétique stationnaire (*Steady-state GA – SSGA*)

Dans ce moteur, un individu est sélectionné, généralement par tournoi, un second si le croisement doit être appliqué, et l'enfant résultant (après croisement et mutation éventuels) est réinsère dans la population en remplacement d'un "viel" individu sélectionné par un tournoi inversé (le moins performant "gagne" …et disparaît).

Stratégies d'évolution ($(\mu \overset{+}{,} \lambda)$-*ES*)

Deux moteurs sont regroupés sous ces appellations. Dans les deux cas, l'étape de sélection est un tirage uniforme (on peut dire qu'il n'y a pas de sélection au sens darwinien). À partir d'une population de taille μ (notations historiques !), λ enfants sont générés par application des opérateurs de variation. L'étape de remplacement est alors totalement déterministe. Dans le schéma (μ, λ)-ES, les meilleurs μ enfants deviennent les parents de la génération suivante, alors que dans le schéma $(\mu + \lambda)$-ES, les meilleurs des $\mu + \lambda$ parents plus enfants sont les parents de la génération suivante.

Algorithme multi-objectif NSGA-II

Cet algorithme utilise l'ordre total au sens de Pareto (voir section 7.1.8) basés sur le rang de Pareto et la distance de peuplement. Un tournoi basé sur cette relation d'ordre est utilisé pour la sélection des géniteurs, et le remplacement se fait de manière déterministe parmi les parents plus enfants.

7.1.10 Les algorithmes historiques

On distingue quatre grandes familles historiques d'algorithme – et les différences entre elles ont laissé des traces dans le paysage évolutionnaire actuel, en dépit d'une unification de nombreux concepts.

- **Algorithmes Génétiques** (*GA*), J. Holland, 1975 et D.E. Goldberg, 1989, Michigan, USA.
  Les GA ont été imaginés comme outils de modélisation de l'adaptation. Ils travaillent dans l'espace des chaînes de bits $[0,1]^n$ avec les moteurs GGA et SSGA. Ce sont les plus connus des algorithmes évolutionnaires, et (malheureusement ?) souvent les seuls variantes connues des chercheurs des autres disciplines.
- **Stratégies d'évolution** (*ES*), I. Rechenberg et H.P. Schwefel, 1965, Berlin. Les ES ont été mises au points par deux jeunes ingénieurs travaillant sur des problèmes numériques. Le contexte était l'optimisation paramétrique, et les moteurs d'évolution sont bien entendu les $(\mu \overset{+}{,} \lambda)$-ES. Un énorme progrès a été apporté par les techniques **adaptatives** d'ajustement des paramètres de mutation, et ce sont sans contestation les meilleurs algorithmes pour les problèmes purement numériques.

- **Programmation évolutionnaire** (*EP*), L.J. Fogel, 1966 et D.B. Fogel, 1991, 1995, Californie, USA.
  Imaginée par Larry Fogel pour la découverte d'automates à états finis pour l'approximation de séries temporelles, EP a rapidement travaillé sur des espaces de recherche très variés. Le moteur utilisé ressemble à s'y méprendre à un $(P+P)$-ES – quoique développé complètement indépendamment, avec toutefois l'utilisation fréquente d'un remplacement déterministe dans lequel les plus mauvais ont tout de même un (petite) chance de survie.
- **Programmation génétique** (*GP*), J. Koza, 1990, Californie, USA.
  Apparue initialement comme sous-domaine des GAs, GP est devenu une branche à part entière (conférence, journal, ...). La spécificité de GP est l'espace de recherche, fait d'arbres représentant des programmes complets. GP cherche à atteindre (et réussit souvent !) un des vieux rêves des programmeurs, faire écrire le programme par un autre programme. Les moteurs dévolution utilisés sont souvent de type SSGA, mais avec des tailles de population énormes. Et les tendance récentes sont pour GP ...la parallélisation systématique et sur de grosses grappes de stations. Ainsi, les résultats récents les plus spectaculaires obtenus par Koza l'ont été avec 64 populations de 10000 individus, ou encore avec 1000 populations de 500 individus, utilisant le modèle en îlots (une population par processeur, avec *migration* régulière des meilleurs individus entre processeurs voisins) sur des grappes *Beowulf* de stations Alpha ou Pentium sous Linux.

7.1.11 Les points-clé

La diversité génétique

Le terme *diversité génétique* désigne la variété des génotypes présents dans la population. Elle devient nulle lorsque tous les individus sont identiques – on parle alors de *convergence* de l'algorithme (a posteriori !). Mais il est important de savoir que lorsque la diversité génétique devient très faible, il y a très peu de chances pour qu'elle augmente à nouveau. Et si cela se produit trop tôt, la convergence a lieu vers un optimum local – on parle alors de *convergence prématurée*. Il faut donc préserver la diversité génétique, sans pour autant empêcher la convergence. Un autre point de vue sur ce problème est le suivant.

Le dilemme exploration-exploitation

A chaque étape de l'algorithme, il faut effectuer le compromis entre **explorer** l'espace de recherche, afin d'éviter de stagner dans un optimum local, et **exploiter** les meilleurs individus obtenus, afin d'atteindre de meilleurs valeurs aux alentours. Trop d'exploitation entraîne une convergence vers un optimum local, alors que trop d'exploration entraîne la non-convergence de l'algorithme.

Typiquement, les opérations de sélection et de croisement sont des étapes d'exploitation, alors que l'initialisation et la mutation sont des étapes d'exploitation. On peut ainsi régler les parts respectives d'exploration et d'exploitation en jouant sur les divers paramètres de l'algorithme (probabilités d'application des opérateurs, pression de sélection, ...). Malheureusement, il n'existe pas de règles universelles de réglages et seuls des résultats expérimentaux donnent une idée du comportement des divers composantes des algorithmes.

7.2 Optimisation de formes évolutionnaire

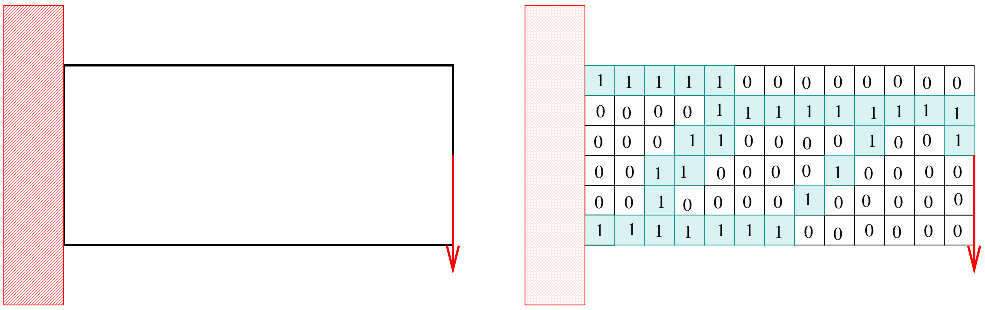
7.2.1 Contexte mécanique

Le contexte général de ce chapitre est l'optimisation Topologique de Structures. Le problème consiste à trouver la forme optimale d'une structure contenue dans un domaine donné (i.e. la répartition de matière dans ce domaine).

La notion d'optimalité est définie à partir de critères mécaniques d'une part (par exemple, une borne sur le le déplacement maximal de la structure soumise à un chargement donné, ou des valeurs de fréquences propres maximales, ou une combinaison de critères mettant en jeu la rigidité et le comportement vibratoire) et le poids de la structure d'autre part. A noter que l'on pourrait aussi introduire par exemple des coûts de fabrication prenant en compte le nombre de trous à percer dans la forme, le nombre de "coins", ...

Le modèle mécanique utilisé ici est, sauf mention explicite du contraire, celui de l'élasticité linéarisée bidimensionnelle en contraintes plane, et les matériaux considérés sont linéaires et isotropes (cf. e.g. [5]). Toutes les données mécaniques sont adimensionnelles (e.g. le module d'Young vaut toujours 1) et les effets de gravité sont négligés.

Un test très populaire pour l'optimum design et celui de la plaque-console (cantilever) : le domaine de calcul est un rectangle, la plaque est encastrée sur la partie verticale gauche de la frontière (déplacement imposé à 0) et le chargement consiste à appliquer une force ponctuelle verticale au milieu de la frontière verticale de droite. La Figure (7.5) montre le domaine de calcul pour le cantilever 2×1.



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

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

FIG. 7.5 – *Le problème de la plaque-console* 2×1 *et une représentation "bitarray" (tableau de bits) d'une structure sur un maillage régulier (ici un maillage* 13×6*).*

7.2.2 La fonction de performance

Nous allons dans la suite considérer la problème test déjà évoqué section 7.2.1, dans lequel les deux caractéristiques à prendre en considération pour l'optimalité sont d'une part le poids de la structure, d'autre part son déplacement sous l'action d'un chargement unique, en occurrence une force ponctuelle appliquée au milieu du côté vertical droit.

Préliminaires mécaniques

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

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

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

Un problème sous contrainte

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

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

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

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

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

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

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

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

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

$$\alpha_{k+1} = \begin{cases} \beta \cdot \alpha_k & \text{si } \Theta^k_{feasible} < \Theta_{inf} \\ (1/\beta) \cdot \alpha_k & \text{si } \Theta^k_{feasible} > \Theta_{sup} \\ \alpha_k & \text{sinon} \end{cases} \tag{7.1}$$

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

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

Un problème multi-objectif

En fait, le problème considéré, même dans le contexte simple de cette section, est un problème multi-objectif : on cherche à minimiser **à la fois** le poids et le déplacement maximal sous l'action du chargement. On peut traiter directement ce problème en utilisant les procédures de sélection spécifiques multi-objectif (voir section 7.1.8), et des résultats d'une telle approche seront présentés section 7.2.7.

Attention cependant : du fait de l'astuce employée afin d'éviter les calculs de connectivité (section 7.2.2), les structures "vides" font partie du front de Pareto (elles sont de poids minimal) et il convient donc de limite l'exploration dudit front aux valeurs de poids (ou de déplacement) intéressantes.

7.2.3 Les tableaux de bits

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

Ainsi qu'il a été dit dans la section 7.2.2, le calcul de la performance d'une structure passe par une résolution d'un problème mécanique par éléments finis, et implique donc une discrétisation (un maillage) de la structure. De plus, si tout se passe bien ("convergence" de la méthode), nous serons amenés à comparer les comportements mécaniques de structures très proches les unes des autres. Or des considérations numériques montrent qu'il est préférable, d'utiliser alors des maillages identiques, afin de ne pas ajouter de bruit de remaillage aux valeurs (proches) des comportements mécaniques.

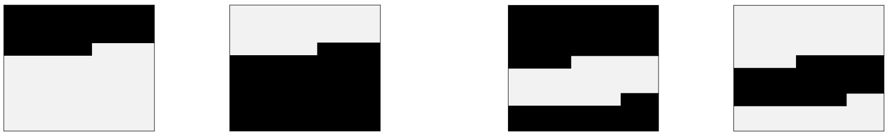
Génotype et initialisation

Ces considérations ont naturellement débouchées sur l'idée de partir d'un maillage donné pour définir une représentation binaire "naturelle", appelée tableau de bits, ou *bitarray* dans [11] : elle est associée au maillage du domaine de design qui est utilisé pour calculer le comportement mécanique de toutes les structures. A chaque élément du maillage on attribue une valeur 0 ou 1, la valeur 1 signifiant que cet élément contient de la matière, et 0 qu'il est vide (cf. Figure 7.5-b).

L'initialisation peut bien sûr se faire de manière standard, en choisissant la valeur 0 ou 1 pour chaque position indépendamment avec probabilité 0.5 – et c'est cette procédure qui a été employée dans la plupart des essais numériques. Toutefois, dans le cas de certaines problèmes pour lesquelles on sait par exemple que les structures très "vides" ne sont pas mécaniquement faisables, on peut biaiser l'initialisation en différenciant la probabilité de 0 et celle de 1.

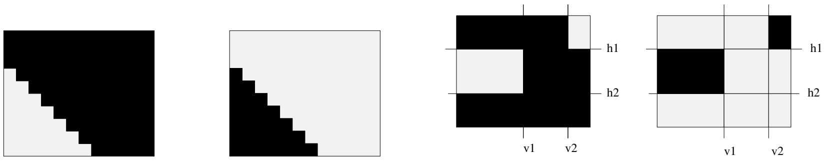
Opérateurs de variation

Il est bien sûr très tentant de considérer les tableaux de bits comme des chaînes de bits (de longueur le nombre d'éléments du maillage), et de leur appliquer les opérateurs de variation définis section 7.1.6. Mais c'est là qu'il est important de prendre en compte les spécificités du problème. En effet, regardons les effets des



a - *1 point standard* b - *2 point standard*

Fig. 7.6 – Exemples d'enfants obtenus par application des opérateurs de croisement sur les chaînes de bits. Les bits blancs (resp. noirs) proviennent du premier (resp. second) parent, indépendamment de leurs valeurs.



c - *Croisement diagonal* d - *Croisement par blocs*

Fig. 7.7 – Exemples de croisement bidimensionnel sans biais dans une direction particulière.

croisements standard à 1 point et à 2 points : la figure 7.6 illustre clairement le défaut de symétrie par rapport aux deux axes de ces croisements – et il a été constaté expérimentalement que les résultats en pâtissaient [10].

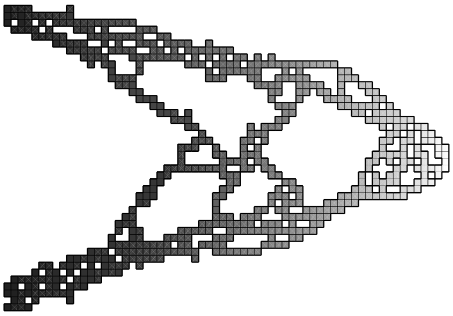
Une fois identifié le problème, il est bien sûr facile d'y remédier : la figure 7.7 montre deux exemples de croisements respectant la symétrie entre les directions X et Y. Le *croisement diagonal* place une droite aléatoirement choisie sur les deux parents, et échange les parties supérieures entre les parents. Le croisement par blocs découpe les parents de manière identique en 9 blocs aléatoires et en échange quelques-uns.

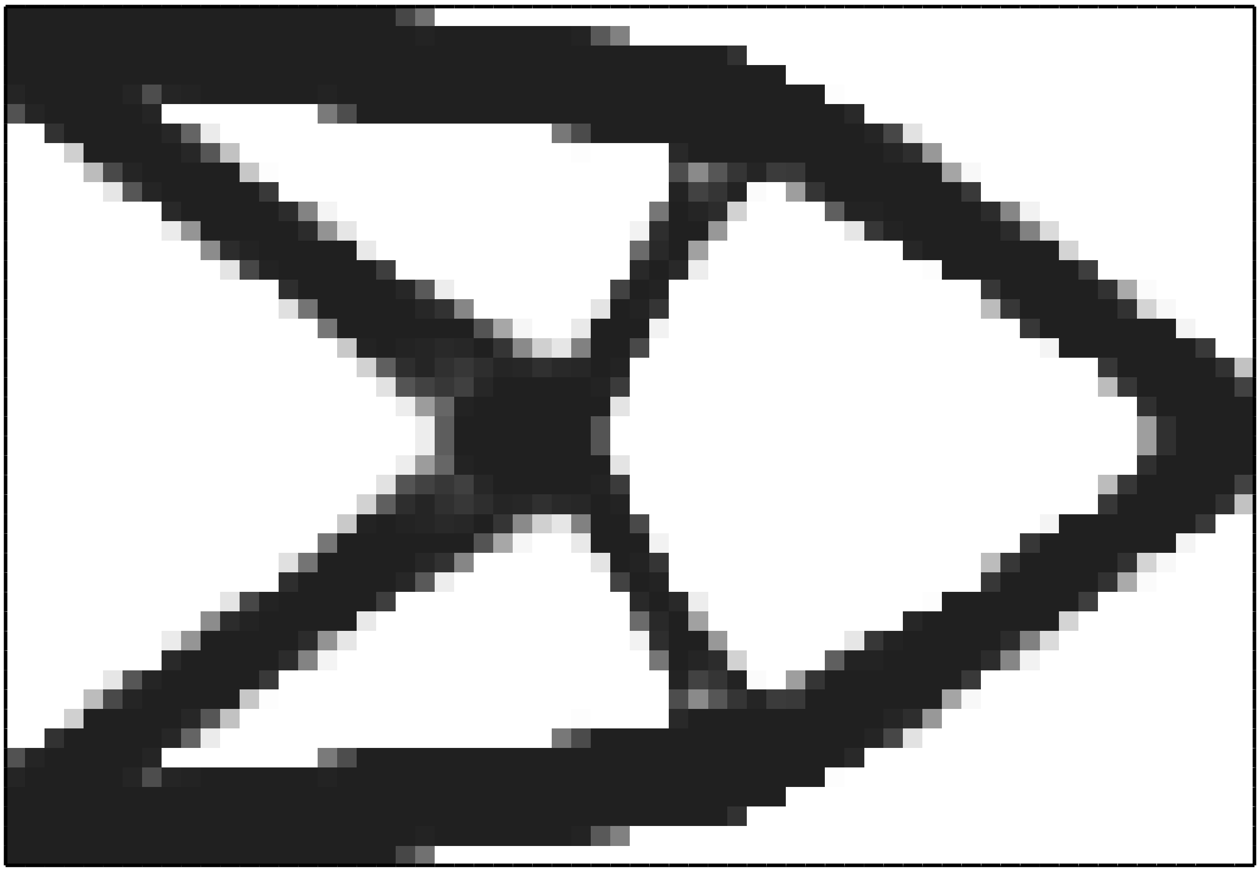
Notons enfin que le croisement uniforme défini sur les chaînes de bits est lui parfaitement neutre géométriquement parlant et peut donc être employé sur les tableaux de bits.

De la même manière, les opérateurs de mutation définis sur les chaînes de bits n'ont pas de biais géométrique et peuvent être employés tels quels sur les représentations des structures.

Résultats numériques

De nombreux résultats numériques ont été présentés à l'aide de la représentation en tableaux de bits dans [9, 8]. Ils ont cependant tous été améliorés par l'utilisation des représentations non-structurées (voir sections 7.2.4 et 7.2.5). Citons cependant parmi ces résultats l'optimisation de formes en élasticité non-linéaire (modèle des grands déplacements et matériau linéaire), premier résultat de ce type, et hors de portée des méthodes classiques. La souplesse des algorithmes évolutionnaires est ici encore mise en évidence, puisqu'il suffit de changer de solveur du problème direct. Citons encore l'optimisation avec conditions aux limites sur la frontière inconnue (et non pas sur la frontière du domaine de définition), comme par exemple le dôme sous-marin soumis à une pression uniforme.





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

7.2.4 Représentation de Voronoï

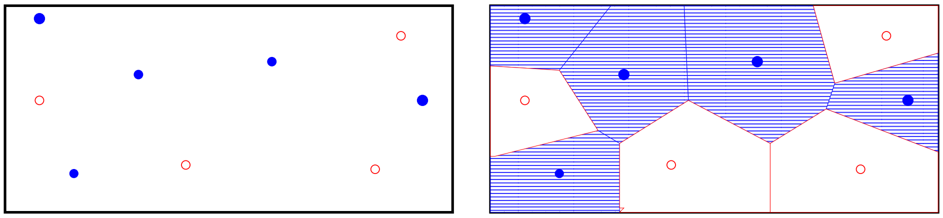
Cette section définit la *représentation de Voronoï*, qui dépasse les limitations de la représentation par tableaux de bits, et les opérateurs de variation associés.

Diagrammes de Voronoï

Considérons un nombre fini de points $V_0, \ldots, V_N$ (les *sites de Voronoï*) dans un domaine borné donné de IR^n (le domaine de travail). A chaque site V_i on associe l'ensemble de tous les points du domaine de travail pour lesquels le site de Voronoï le plus proche est V_i. On nomme cet ensemble *cellule de Voronoï*. Le *diagramme de Voronoï* est la partition du domaine définie par les cellules de Voronoï. Chaque cellule est un sous-ensemble polyédral du domaine de travail (cf. [14]).

Génotype et initialisation

Considérons maintenant une liste – de longueur variable – de sites de Voronoï, chaque site étant étiqueté 0 ou 1. Le diagramme de Voronoï correspondant, dans lequel chaque cellule est étiquetée comme le site qu'elle contient, représente une partition du domaine de travail en deux sous-ensembles(cf. Figure (7.11)).



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

FIG. 7.9 – *La représentation de Voronoï sur un domaine rectangulaire* 2×1.

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

Décodage

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

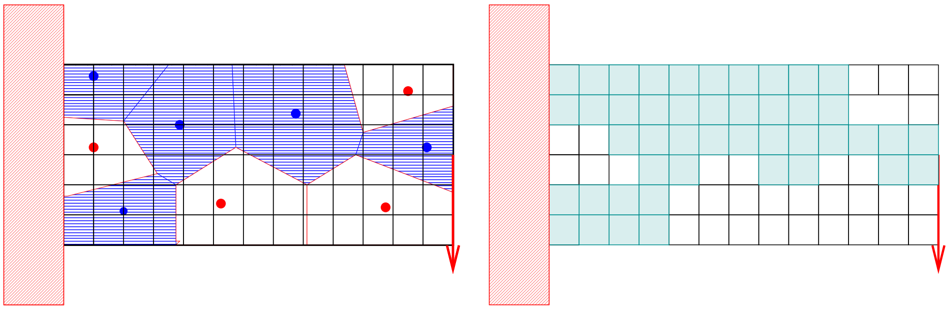


FIG. 7.10 – *la représentation de Voronoï sur le problème de la plaque-console* 2 × 1 *et la structure correspondante (répartition vide/matériau)*

Opérateurs de variation

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

- L'opérateur de **croisement** échange les sites de Voronoï sur une base géométrique. De ce point de vue, il est similaire à l'opérateur de croisement spécifique décrit dans [10]. La Figure (7.11) est un exemple d'application de cet opérateur.
- L'opérateur de **mutation** est choisi grâce à un choix aléatoire, avec des poids définis par l'utilisateur, parmi les opérateurs suivants :
  - la mutation de *déplacement* effectue une mutation Gaussienne sur les coordonnées des sites. Comme dans les stratégies d'évolution [15], la mutation adaptative est utilisée : une déviation standard est associée à chaque coordonnée de chaque site de Voronoï, et elle subit une mutation log-normale avant d'être utilisée pour la mutation Gaussienne des coordonnées correspondantes.
  - la mutation de *label* change aléatoirement l'attribut booléen d'un site.
  - les mutations *ajout* et *suppression* sont des opérateurs spécifiques aux représentations de longueur variable qui respectivement ajoutent ou suppriment aléatoirement un site dans la liste.

Conditions expérimentales

Sauf indication contraire, les expériences numériques présentées dans la suite utilisent les paramètres suivants : évolution standard de type Algorithme Génétique (voir section 7.1.9 : sélection basée sur le rang et remplacement de tous les parents par leur descendance), avec des population de 80 individus ; au plus 40 sites de Voronoï (ou barres cf. section 7.2.5) par individu ; taux de croisement de 0.6 et taux de mutation de 0.3 par individu ; les poids relatifs de chaque type de mutation sont de 0.5 pour la mutation de déplacement, les autres types de mutations se

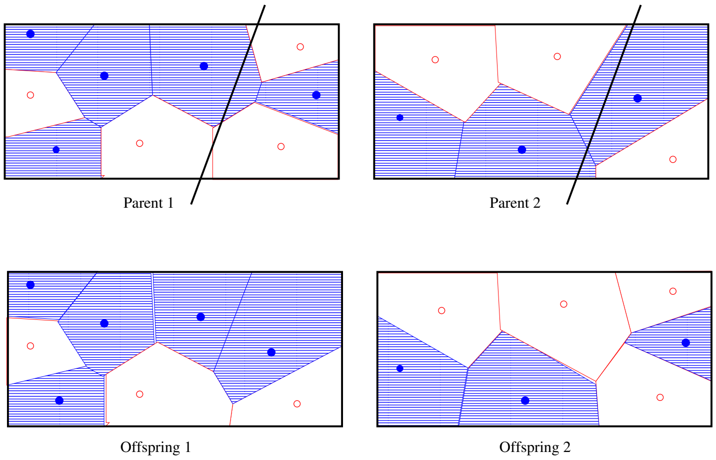
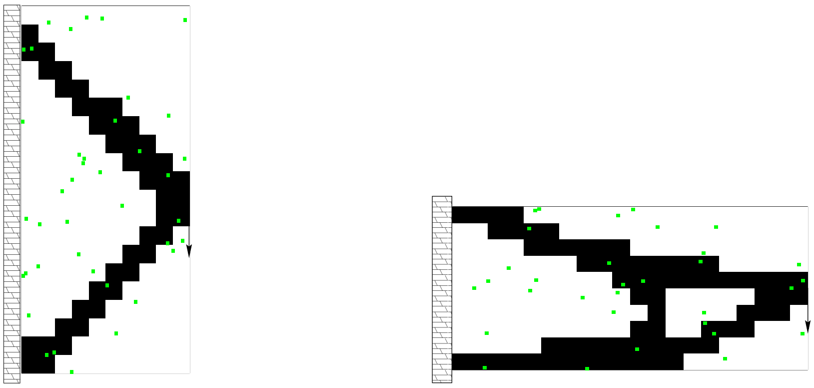


FIG. 7.11 – *Un opérateur de croisement pour la représentation de Voronoï : une droite aléatoire est tracée dans chaque diagramme et les sites sont échangés de part et d'autre de cette droite.*

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

Premiers résultats

L'approche a été validée sur les problèmes-test de cantilevers de dimension 1×2 et 2×1, discrétisées respectivement selon un maillage régulier de 10×20 et 20×10, avec des limites respectives sur le déplacement maximal de 20 et 220. La Figure (7.12) montre les meilleurs structures obtenus pour les deux cas-test. le coût d'une génération est de moins d'une seconde.



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

FIG. 7.12 – *Les deux meilleurs individus pour la représentation de Voronoï. Temps CPU = 1s/génération. (a) maillage* 10×20, *D_{lim}=20 (b) maillage* 20×10, *D_{lim}=220.*

Le cantilever 10×1

Le problème du cantilever 10×1 (discrétisé par un maillage régulier de 200×20 éléments) est délicat à traiter avec une représentation de type bitarray à cause d'une difficulté supplémentaire par rapport au cas précédent : la plupart des structures générées aléatoirement au cours du processus d'initialisation ne connectent pas le point d'application de la force avec la frontière encastrée. Une procédure d'initialisation particulière est utilisée, dans laquelle le poids moyen des structures aléatoires peut être contrôlé (cf. [7] pour les détails). De plus, le nombre maximal de sites par individus a été augmenté à 120 et les meilleurs résultats sont obtenus pour des populations de 120 individus. La Figure (7.13) montre un des résultats les plus significatifs pour une contrainte $D_{lim} = 12$.

FIG. 7.13 – *Structure optimale sur le maillage* 200×20 *pour le cantilever* 10×1. *D_{lim}=12, D_{max}=11.99, poids=0.445. Temps CPU = 28s/génération.*

Un problème 3D

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

Le domaine de travail est un parallélépipède rectangle et le problème présente une symétrie permettant de ne discrétiser que la moitié du domaine par un maillage à $16 \times 7 \times 10$ éléments. Le plan en $x = 0$ (au fond sur la Figure (7.14)) est encastré, et une force verticale est appliquée au milieu de la face opposée.

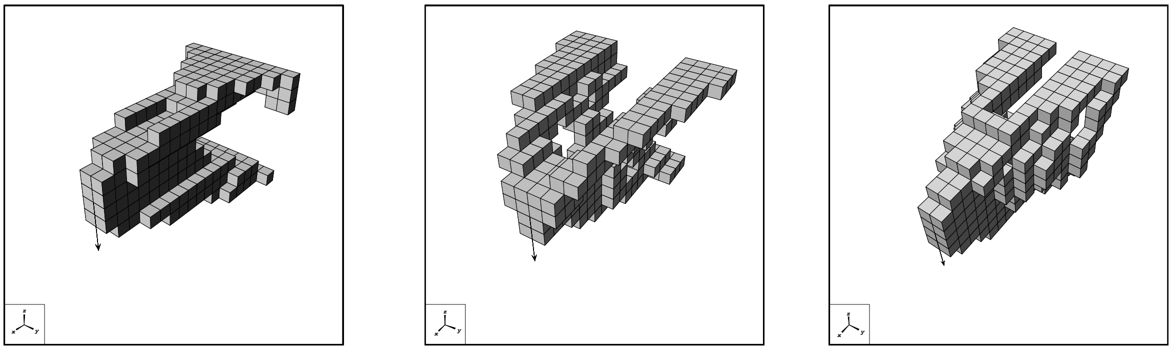
Pour ce problème plus complexe que les cas précédents, quelques paramètres doivent être modifiés : la taille de la population est de 120 individus et le nombre maximum de site de Voronoï par individu vaut 120.

La Figure (7.14) montre que l'algorithme a été capable de trouver quelques solutions acceptables …en quelques jours de calcul (les analyses par éléments finis de problèmes 3d sont beaucoup plus coûteuses que pour les cas 2d à nombre de mailles égal). De plus, elle démontre la capacité connue des AEs de trouver des solutions quasi-optimales multiples pour un même problème, certaines étant assez originales par rapport à celle obtenue par la méthode d'homogénéisation [1].

7.2.5 Représentation par barres de Voronoï

Les barres de Voronoï :

Une barre de Voronoï est définie par quatre variables réelles : ses coordonnées (x, y), l'angle θ de la barre avec l'axe x et sa largeur. La Figure (7.15-a) montre un exemple d'une barre de Voronoï.



(a) : poids=0.15178, 103 sites (b) : poids=0.166, 109 sites (c) : poids=0.157, 112 sites

FIG. 7.14 – *Trois résultats pour le cantilever tridimensionnel sur un maillage* $16\times7\times10$ *sur la moitié du domaine, avec la même contrainte sur le déplacement maximal. Temps CPU = 4 à 5mn/génération.*

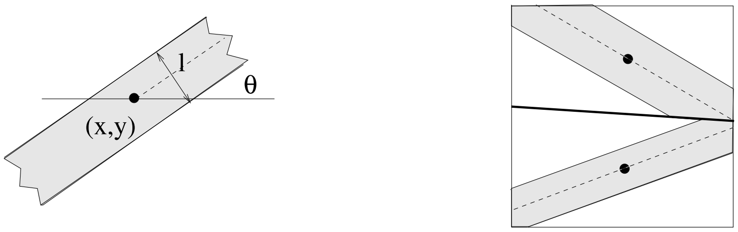


FIG. 7.15 – *La représentation par barres de Voronoï. Une seule barre (a) et la structure générée par deux barres (b) : le trait plus épais est la frontière entre les deux cellules de Voronoï. Elle ne fait partie de la structure qu'à la jonction entre les deux barres.*

Génotype et initialisation

Un individu dans la représentation par barres de Voronoï est une liste de taille variable de barres. Quand toutes les barres sont simplement vues comme des sites de Voronoï, le diagramme correspondant donne une partition du domaine en polygones convexes. Chaque polygone est alors séparé en deux sous-domaines : la partie centrale – définie par l'angle et la largeur – contient de la matière, et son complémentaire contient du vide (cf. Figure (7.15)). Lorsque la largeur est assez grande, la totalité de la cellule est remplie de matière, tandis qu'une valeur nulle de la largeur donne une cellule vide.

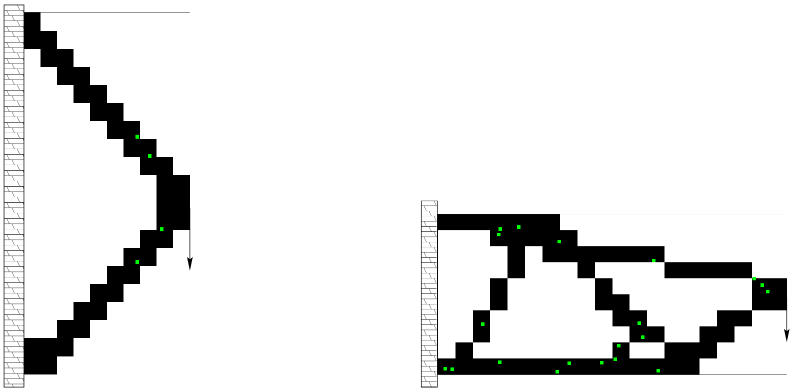
Décodage

Comme pour la représentation Voronoï, la performance des individus est évaluée après projection sur un maillage fixe telle qu'elle est décrite dans la section 7.2.4 : un élément du maillage est considéré comme plein de matière si et seulement si son centre de gravité est dans la partie pleine d'une barre de Voronoï.

Comme on peut le voir sur la Figure (7.15-b), le décodage de barres de Voronoï adjacentes permet le contrôle direct de presque toute la frontière de la structure, à l'exception des quelques portions limitées à la jonction de deux barres.

Opérateurs de variation

Ils sont de nouveau dérivés de ceux de la représentation de Voronoï : la procédure d'initialisation choisit un nombre de barres et initialise leurs coordonnées, angles et largeur uniformément. L'opérateur de croisement échange les barres exactement comme les sites dans le cas de la représentation de Voronoï (cf. Figure (7.11)). Les opérateurs de mutation sont également similaires avec des possibilités supplémentaires de mutation Gaussienne sur l'angle et la largeur de la barre.



(a) : poids=0.2, 4 barres (b) : poids=0.33, 20 barres

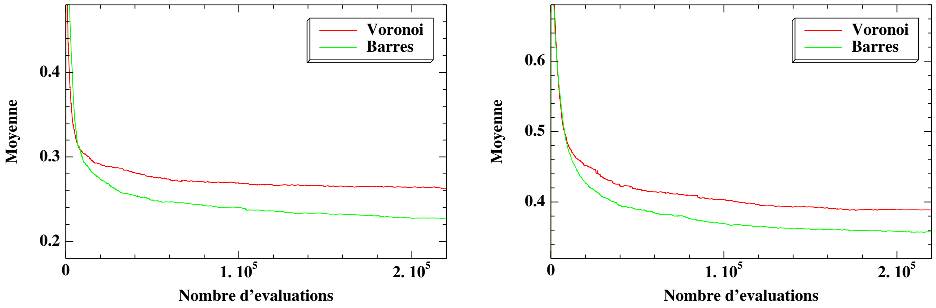
FIG. 7.16 – *Les deux meilleurs individus pour la représentation par barres de Voronoï.*

Résultats numériques

La figure 7.16 montre les meilleurs résultats obtenus parmi 21 essais effectués avec la représentation par barres de Voronoï sur les mêmes problèmes tests que ceux de la figure 7.12. La qualité des structures est ici légèrement meilleure, sans que la différence soit significative. Cependant, la complexité des solutions obtenues (le nombre de barres / sites de Voronoï) est lui très nettement en faveur de la représentation par barres (et cette tendance a été systématiquement observée).

7.2.6 Résultats comparatifs

Nous présentons dans cette section les comparaisons des deux représentations basées sur les diagrammes de Vorornoï. La première conclusion des expériences numériques faites sur des tests classiques (des cantilevers de dimension 1 × 2 et 2 × 1 avec des limites respectives sur le déplacement maximal de 20 et 220) est que les deux représentations permettent toutes de trouver des solutions aussi bonnes sur 21 calculs indépendants pour les deux cas-test. Toutefois, pour ces deux cas-test la représentation par barres de Voronoï est sans conteste meilleur que celle de Voronoï. Cette performance est visible sur les comparaisons des Figures (7.17-a) et (7.17-b).

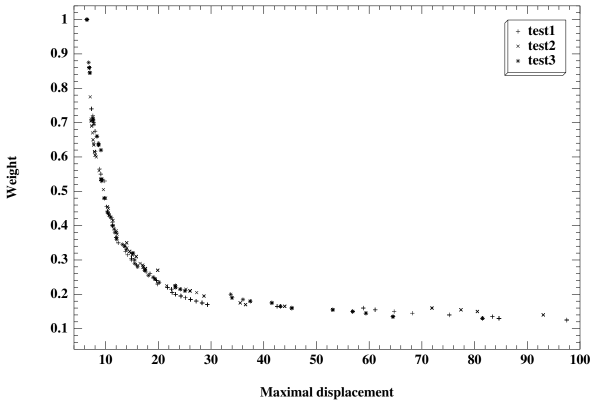


(a)Modèle 1 × 2 (b) Modèle 2 × 1

FIG. 7.17 – *Comparaison de deux représentations de Voronoï. fitness de la meilleur structure au cours de l'évolution (moyenne sur 21 calculs indépendants)*

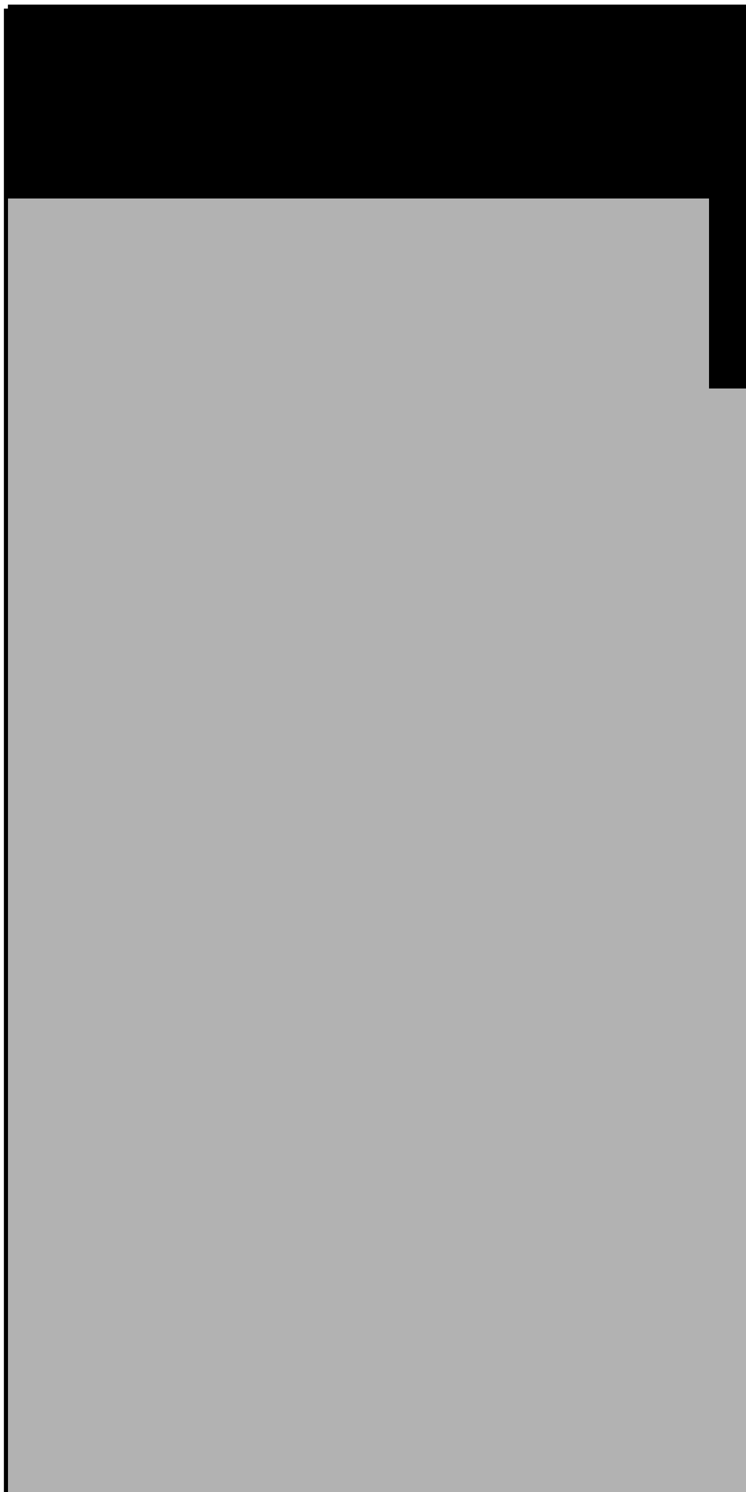
D'autre part, les solutions attendues pour ces cas-test sont très simples et la représentation devrait refléter cette simplicité. Sur ce point aussi, la représentation par barres de Voronoï trouve des représentations très compactes (cf. Figure (7.16)) comparées à celles obtenues avec la représentation de Voronoï. Un des calculs a même obtenu la structure parfaite formée de deux barres en V pour le cantilever 1 × 2.

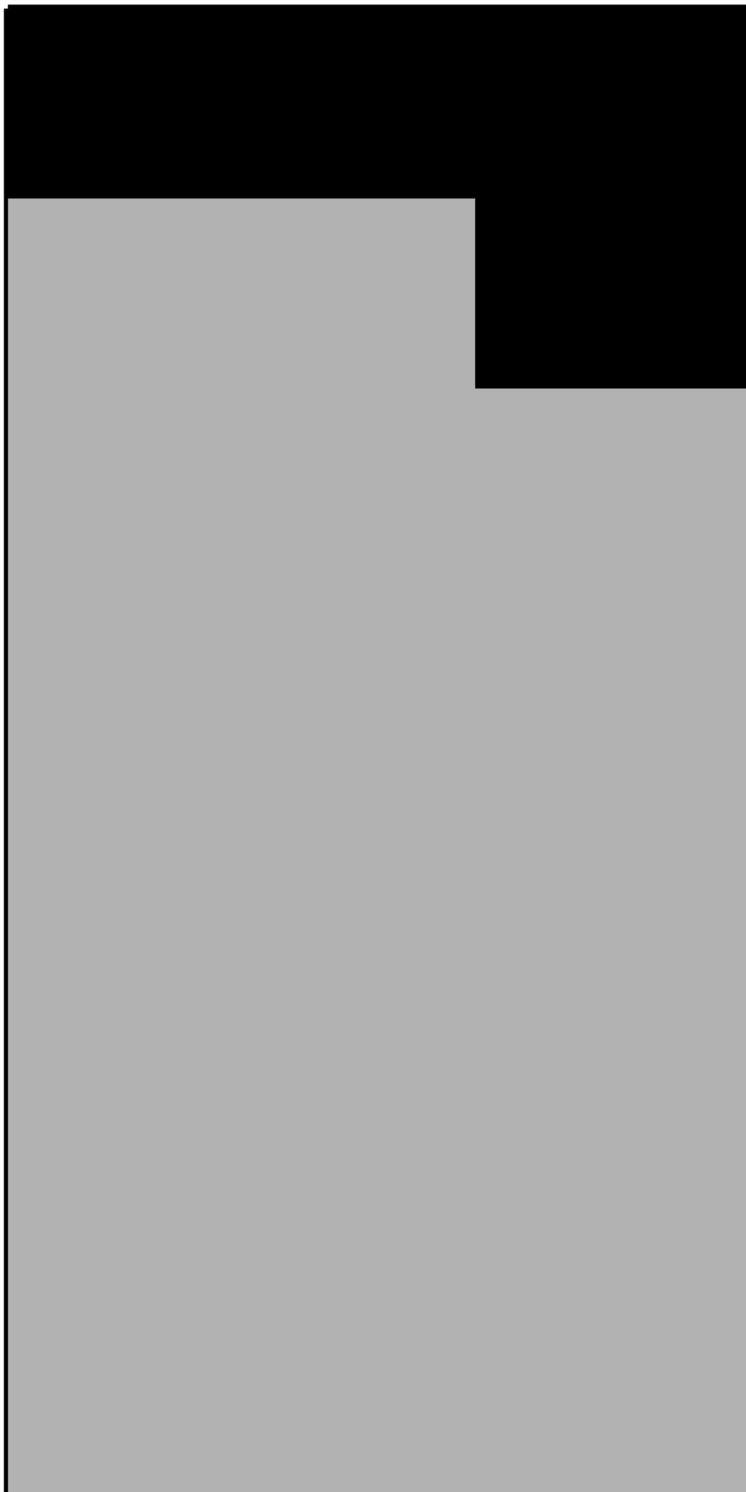
Il ne faut cependant pas perdre de vue que la représentation par barres de Voronoï semble taillée sur mesure pour représenter des structure en treillis de barres – et que nous n'avons comparé les deux représentations que sur des problèmes oú la solution est précisément un treillis. Il convient donc de ne pas généraliser hâtivement ce résultat !

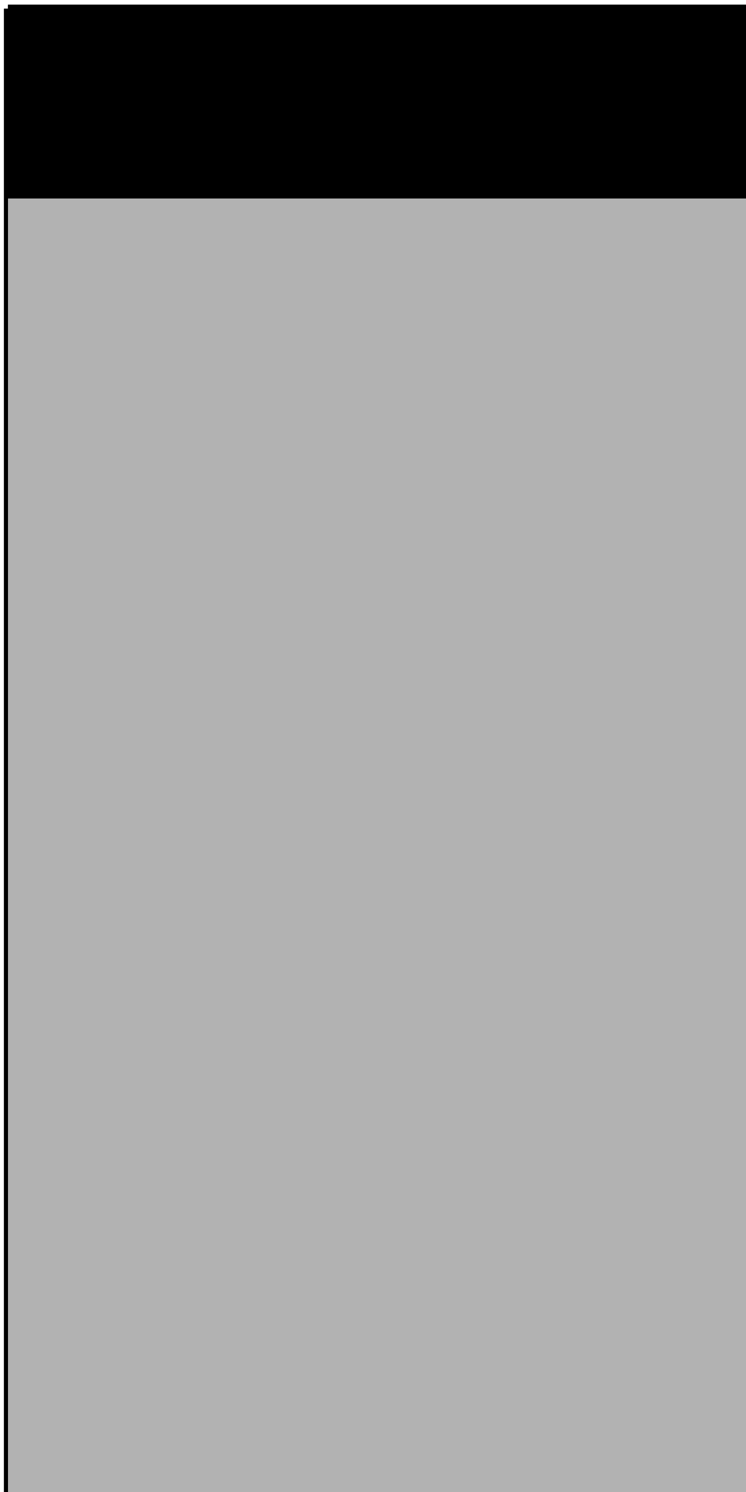


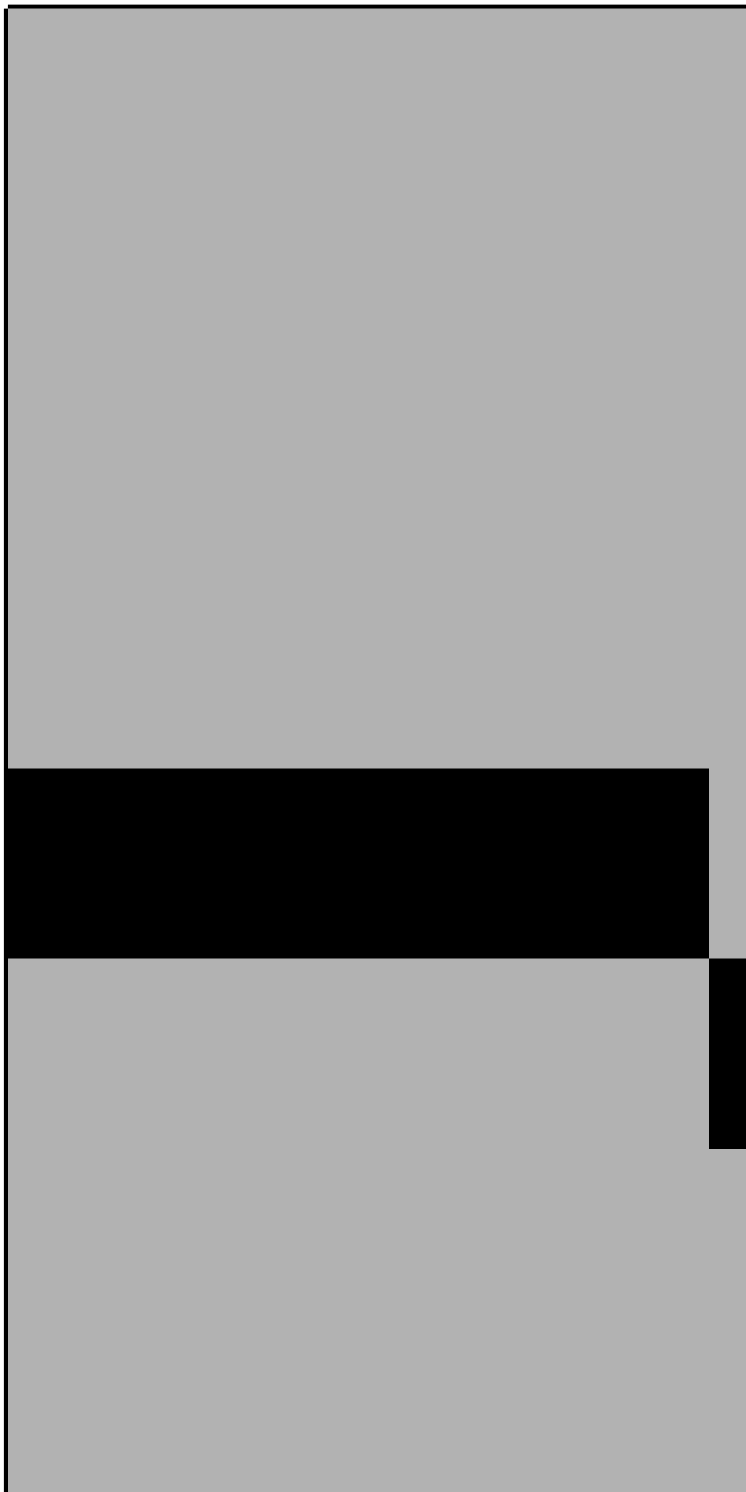
Weight
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
10
20
30
40
50
60
70
80
90
100
Maximal displacement
test1
test2
test3

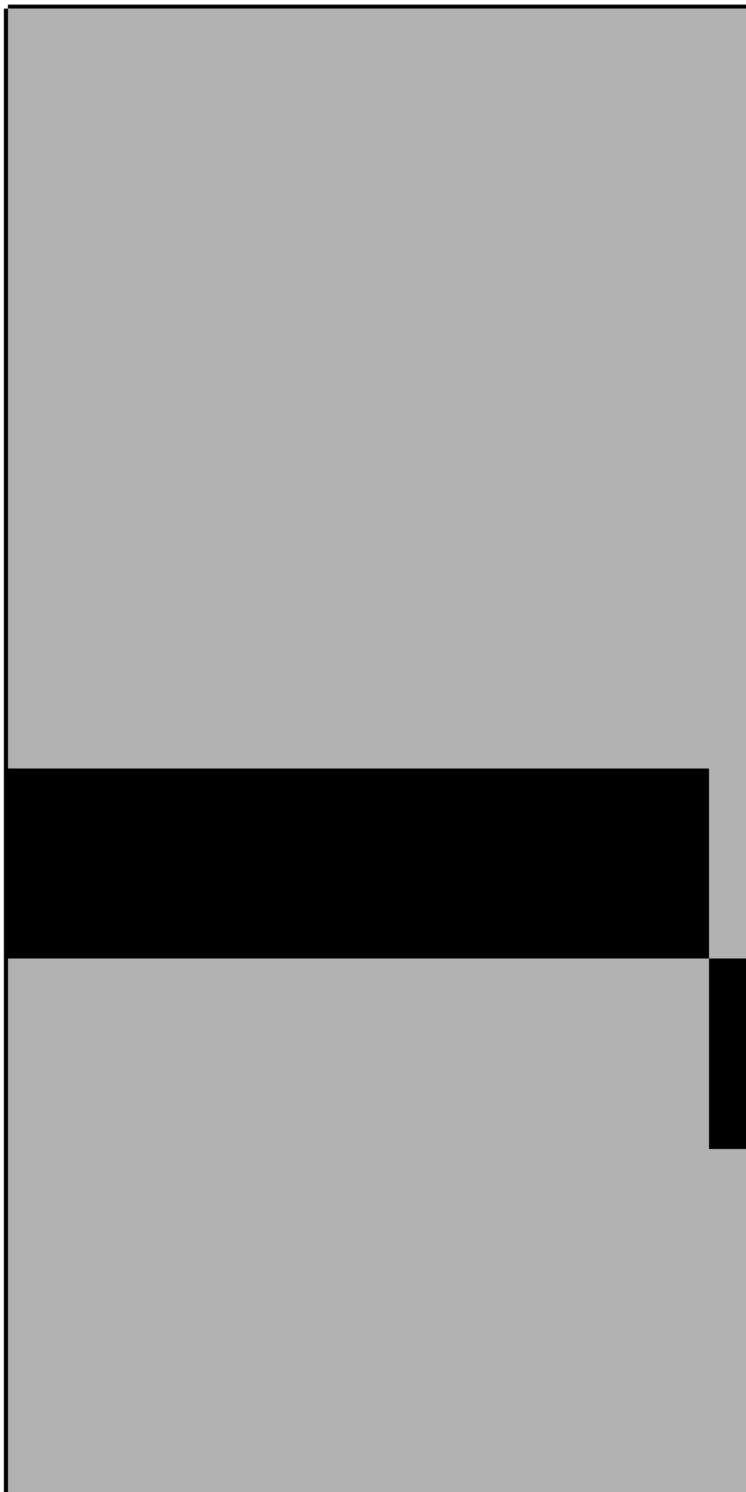
[redacted]

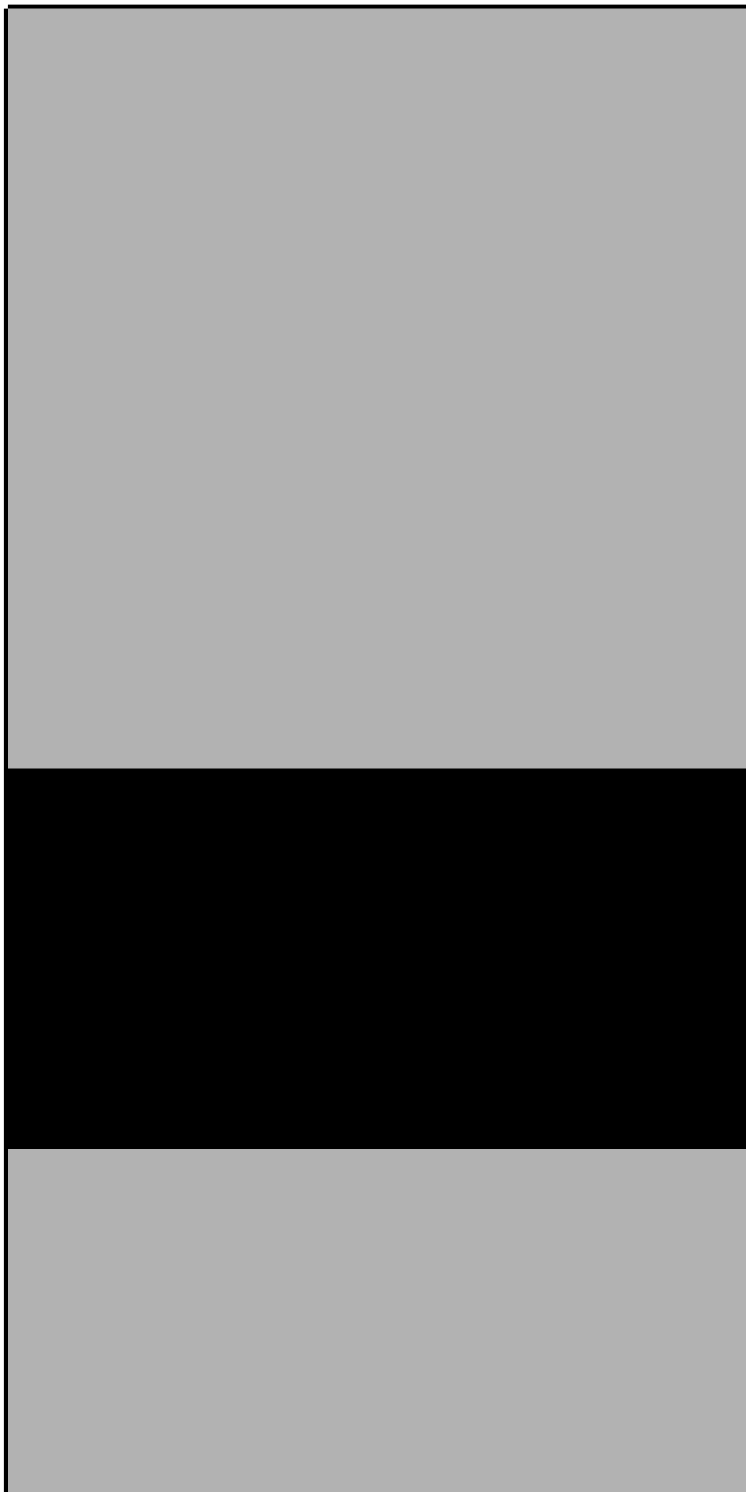


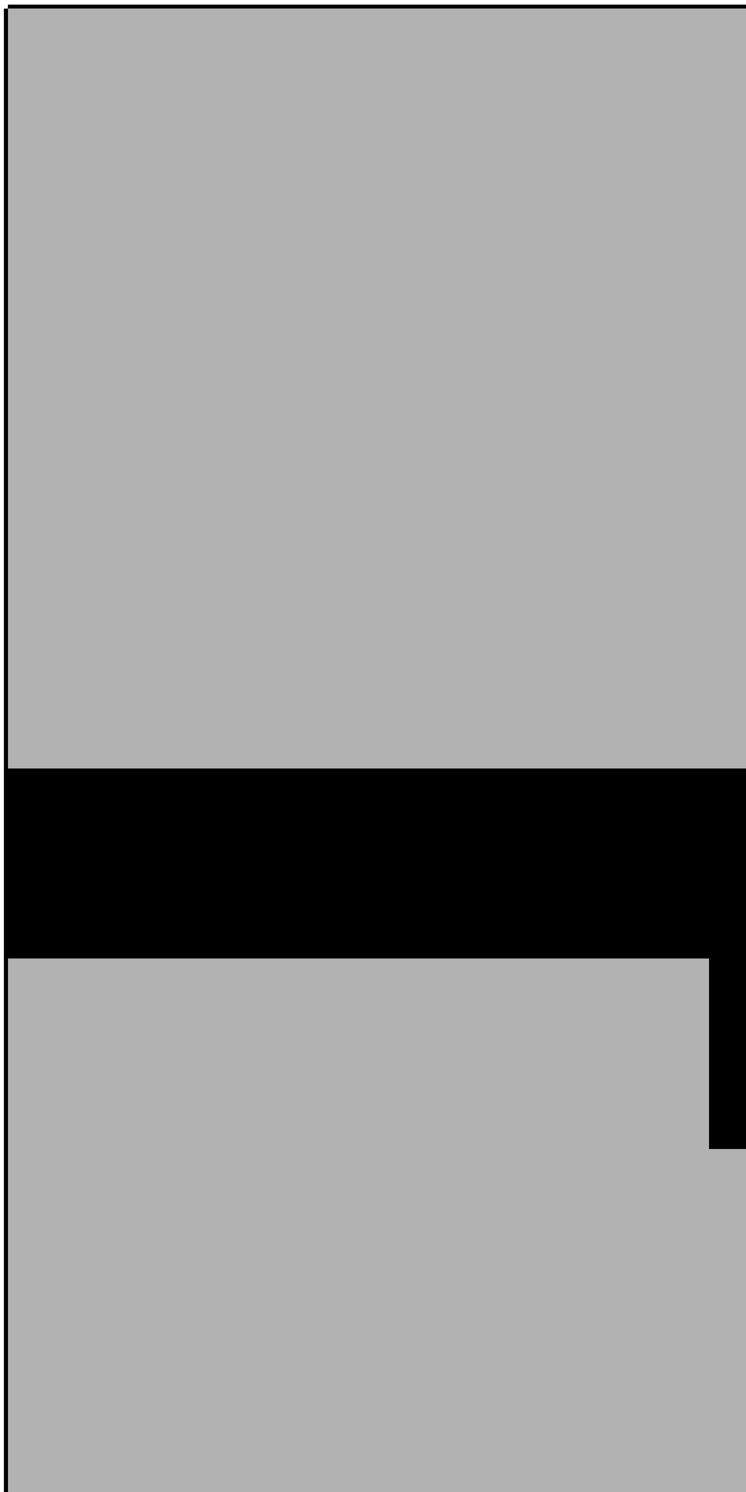


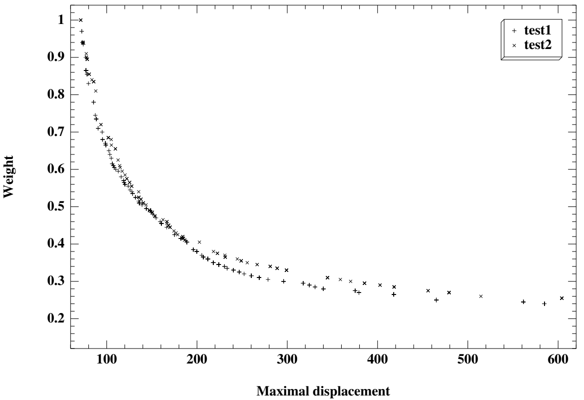


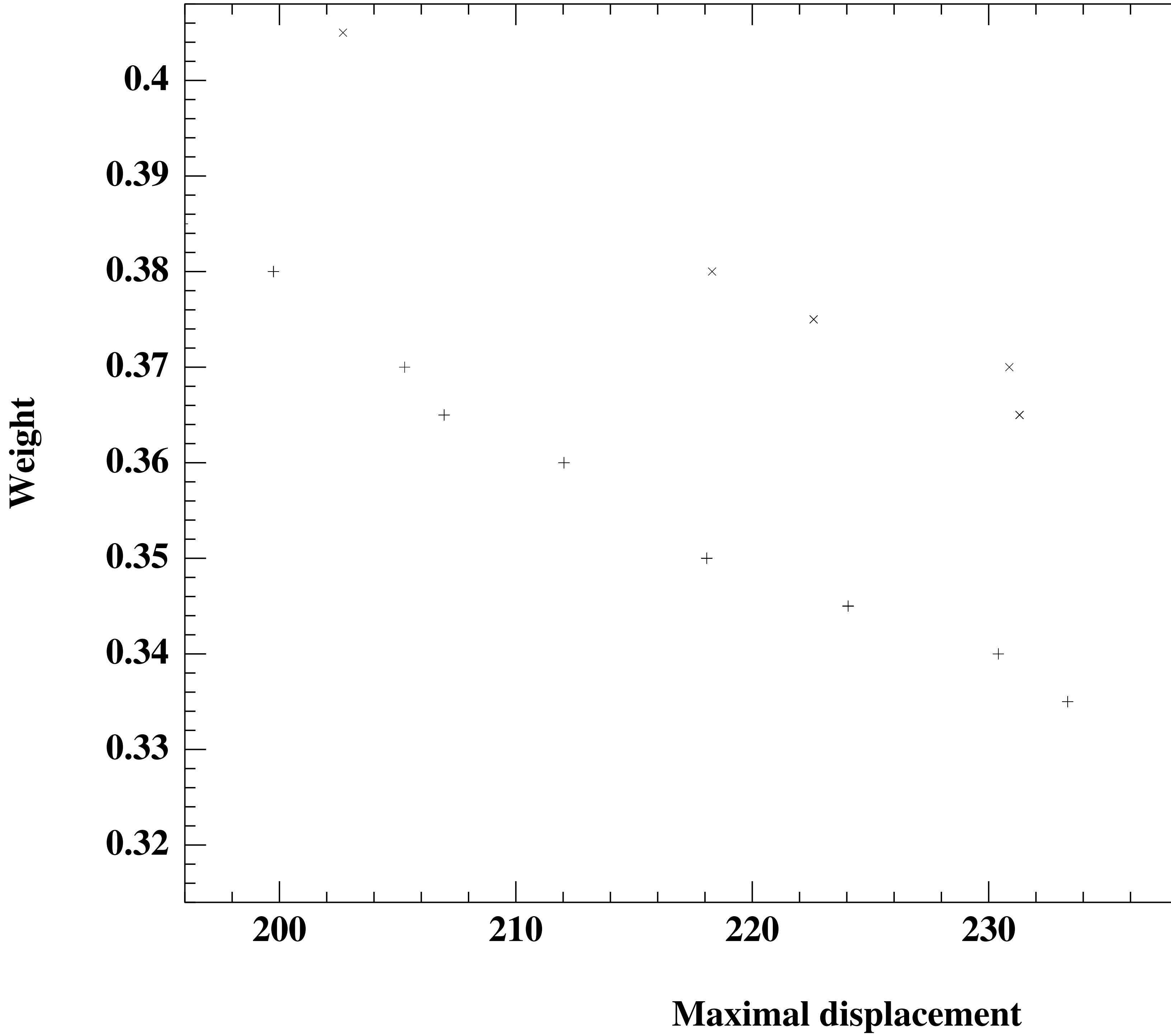









Weight
Maximal displacement
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
100
200
300
400
500
600
test1
test2


Weight
0.4
0.39
0.38
0.37
0.36
0.35
0.34
0.33
0.32
200
210
220
230
Maximal displacement