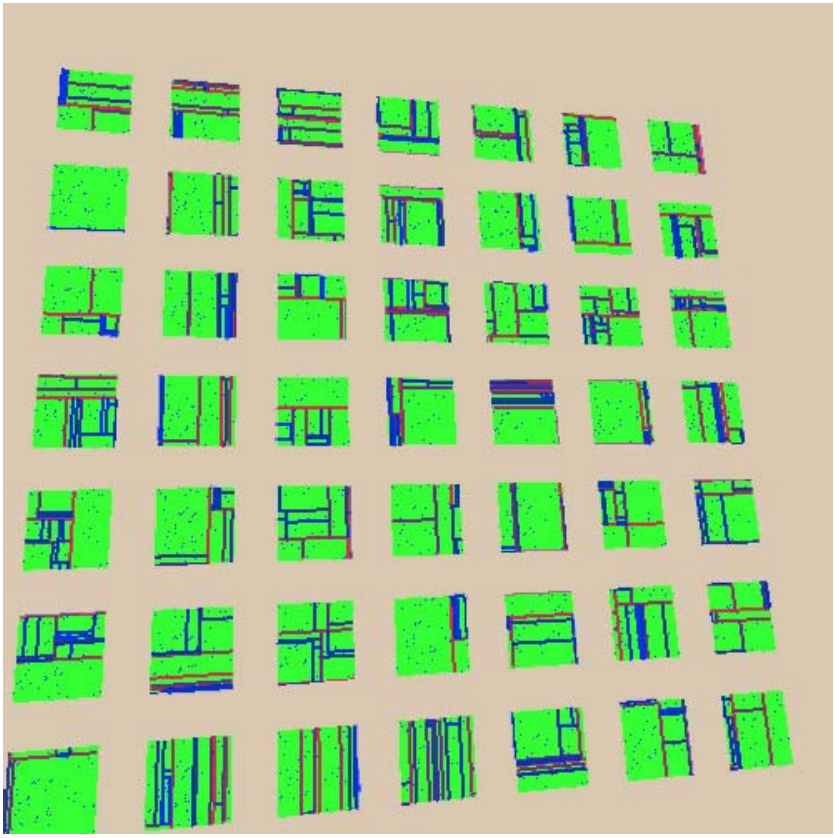
Cyril CRASSIN, Sylvain JUGE et Régis VOGEL

Printemps 2004

Pavage du Plan



Sommaire

Introduction

L'objectif de ce projet AG41 est de résoudre un problème de pavage du plan à l'aide d'un algorithme génétique. On dispose au départ d'un plan sur lequel sont disposées des ressources. Le but est alors de trouver un motif de découpage de la zone à l'aide de lignes horizontales et verticales permettant de répartir équitablement les ressources dans chaque zone. Pour cela, nous avons d'abord réfléchi à une modélisation du problème avec ses contraintes et son objectifs, puis nous les avons ensuite implémentés en C++.

Pour atteindre notre objectif, nous avons choisi de représenter le plan sous forme d'arbre et nous avons implémenté des opérateurs de croisement, mutation remplacement et sélection.

Notre objectif a été tout au long de ce projet de réaliser un développement générique et réutilisable basé sur une analyse claire de l'existant ainsi qu'une conception UML avancée. Notre problématique principale a également été d'optimiser au maximum l'application pour permettre l'évaluation d'une population importante.

1. Modélisation du problème

1. Le problème

Données du problème :

- Le nombre de zones : ceci correspond au nombre de droites plus un.
- Les dimensions de la zone d'origine : ceci définit la taille du plan. Elle servira de base aux sous zones dont la taille sera une proportion de la zone parente.
- Les ressources :
    - Quantité présente sur le plan.
    - Position de chaque ressource.
    - Poids de chaque ressource.

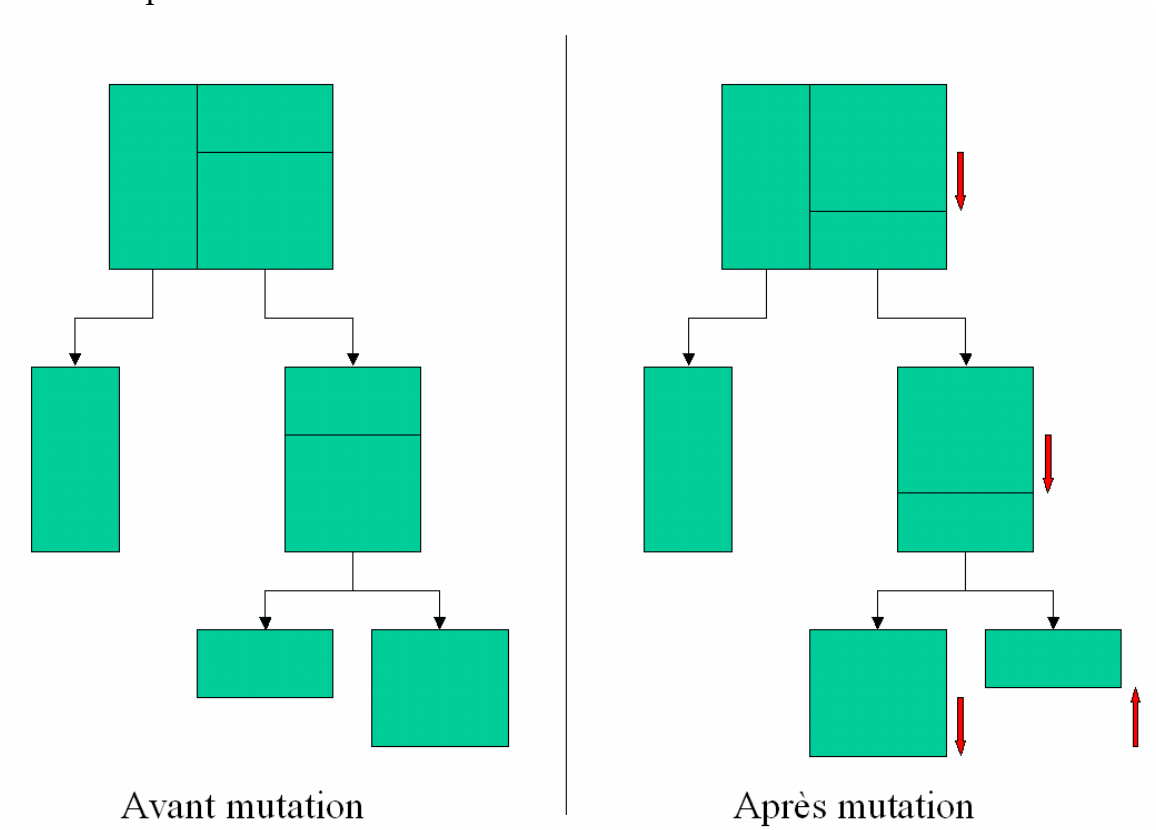
Sorties :

- Les zones : elles sont définies comme une partie de la zone parente. Chaque zone contient en fait 2 sous zones qui contiennent alors 2 sous zones et ainsi de suite jusqu'à avoir le nombre de zones defini.
- La fitness : elle est calculée pour chaque plan de la population. Elle correspond en fait à la somme des ressources en défaut et en excès de chaque zone.

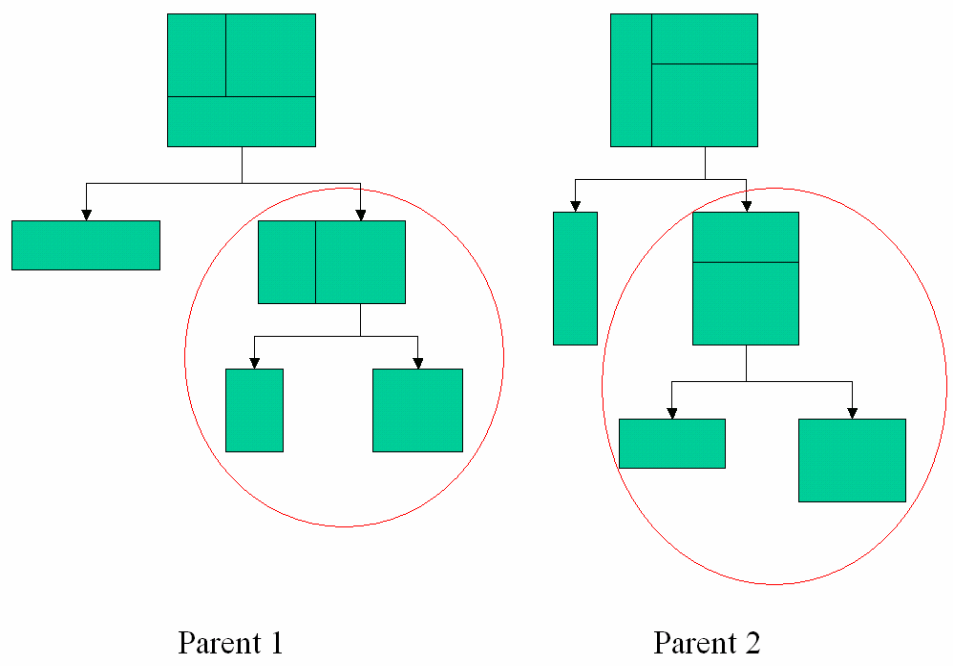
Contraintes : La structure arborescente que nous avons choisi permet à notre méthode de respecter naturellement les contraintes car les zones sont définies par une proportion de la surface de leur parent et non pas par des coordonnées de lignes qui devraient être mises à jour.

2. Les opérateurs

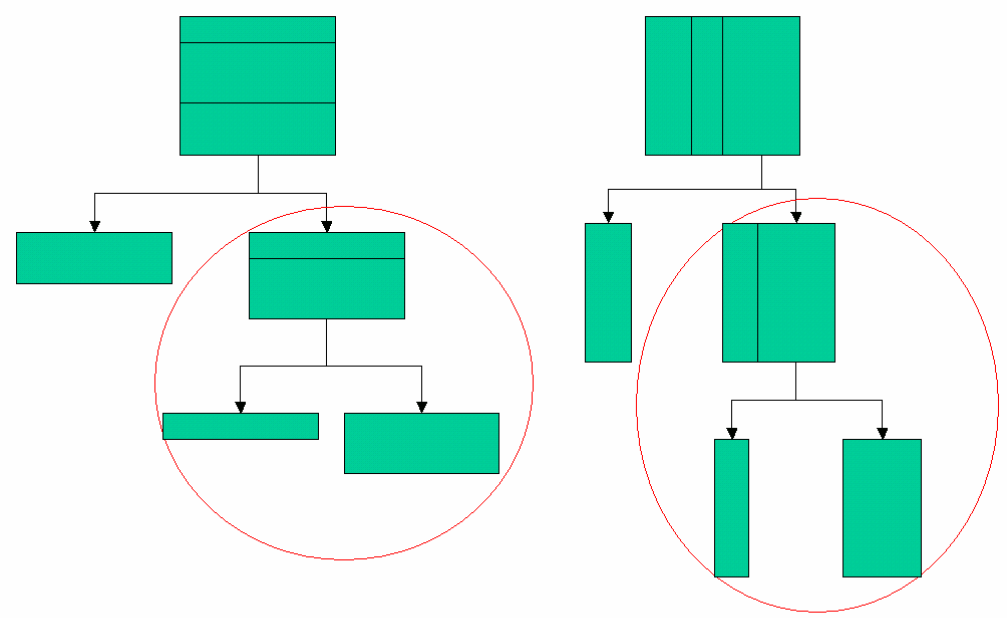
- La mutation : L'opérateur choisit une zone aléatoirement et fait évoluer sa séparation en fonction des fitness des deux sous zones. Il est bon de remarquer que les sous zones subissent aussi un décalage du fait de la représentation arborescente que nous avons choisi et que nous détaillerons dans la partie « Implémentation informatique ».



- Le croisement : On réalise l'échange de zones contenant le même nombre de sous zones à leur niveau le plus profond (feuilles).



Parent 1 Parent 2



Enfant 1 Enfant 2

- La sélection : L'opérateur effectue une sélection ranking identique à celle effectuée en TP. On trie les individus de la population par fitness décroissante puis on sélectionne l'individu à croiser parmi les meilleurs individu en prenant comme seuil de sélection :

$$P_{select}(i) = \frac{(Ri + Tr)}{\sum_{0}^{Tpop-1} Ri + Tr \times Tpop}$$

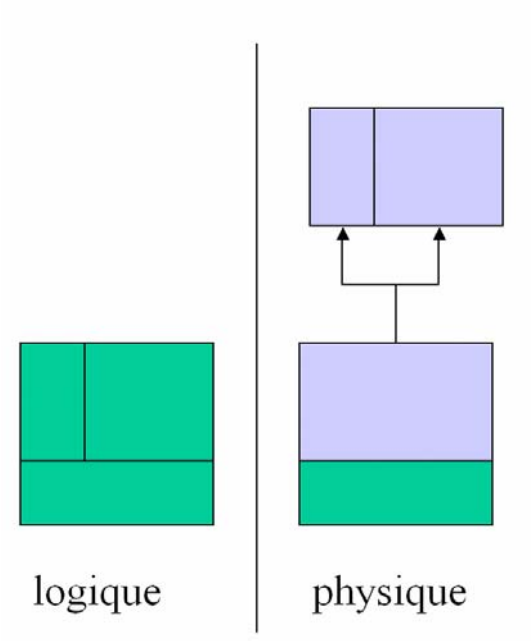
- Le remplacement : On classe les individus par ordre croissant, puis on sélectionne les individus qui seront remplacés avec la même formule que pour l'opérateur de sélection.

2. Implémentation informatique

Nous avons choisi de représenter chaque individu de la population sous forme arborescente. On distingue deux types de zones :

- Les nœuds de l'arbre : Chaque nœud contient une séparation verticale ou horizontale et se divise donc en deux sous zones.
- Les feuilles de l'arbre : Elles ne sont pas divisées mais contiennent un certain nombre de ressources. On stocke les ressources sur les feuilles afin que, lors d'un redimensionnement d'une zone, on limite le temps de calcul. En effet, pour réattribuer les ressources aux nouvelles zones, on ne recherche que dans le sous arbre modifié.

On remarque sur la représentation graphique que la notion de zone correspond aux feuilles de l'arbre.



Nous nous sommes efforcés avant de nous lancer dans le développement du projet en lui-même, de fixer les structures de base d'un algorithme génétique ainsi que l'ensemble des structures de données nécessaires à notre représentation. Nous avons ainsi créé un squelette générique et réutilisable d'implémentation d'un algorithme génétique et de son affichage en prenant soin de bien le séparer du traitement du problème lui même.

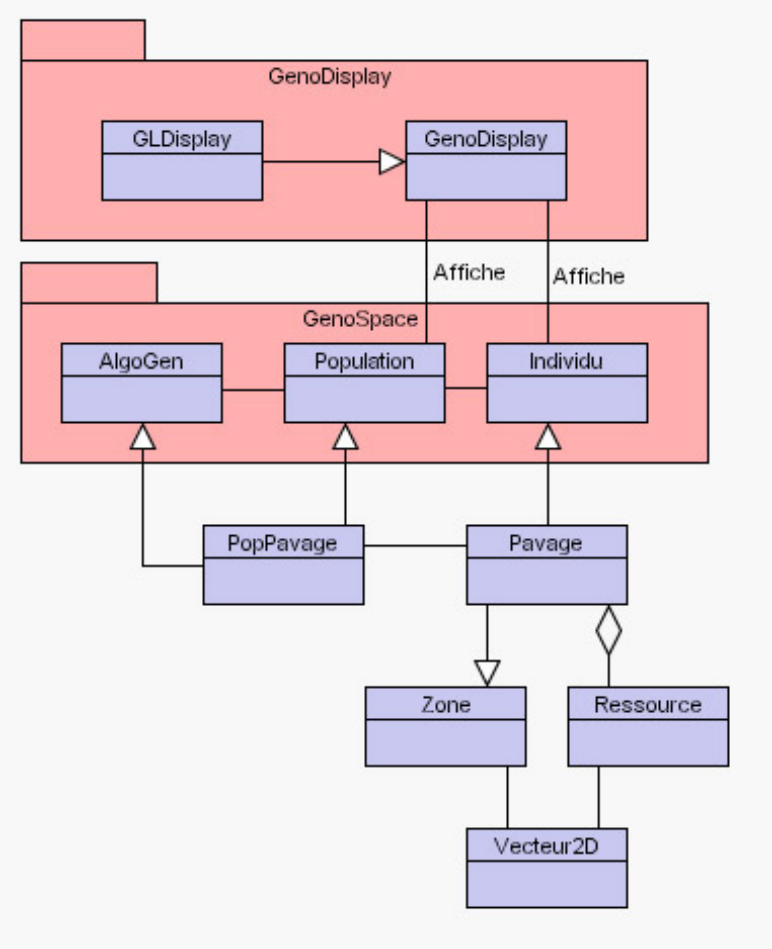
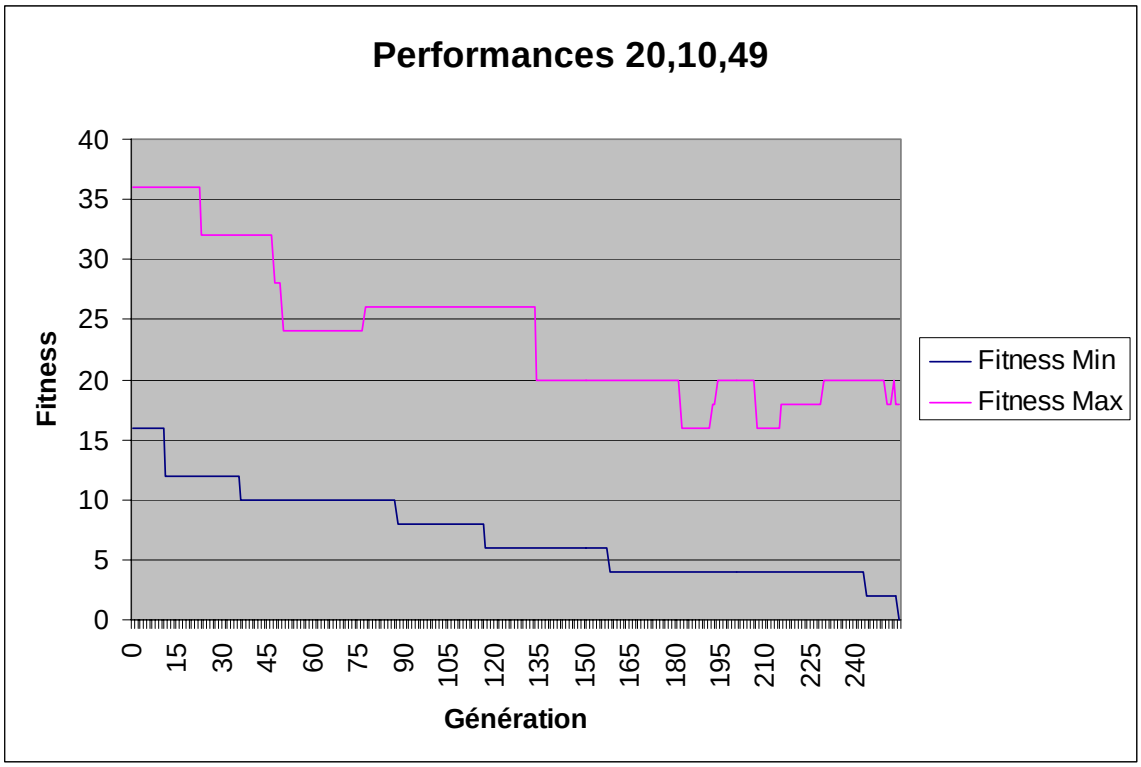


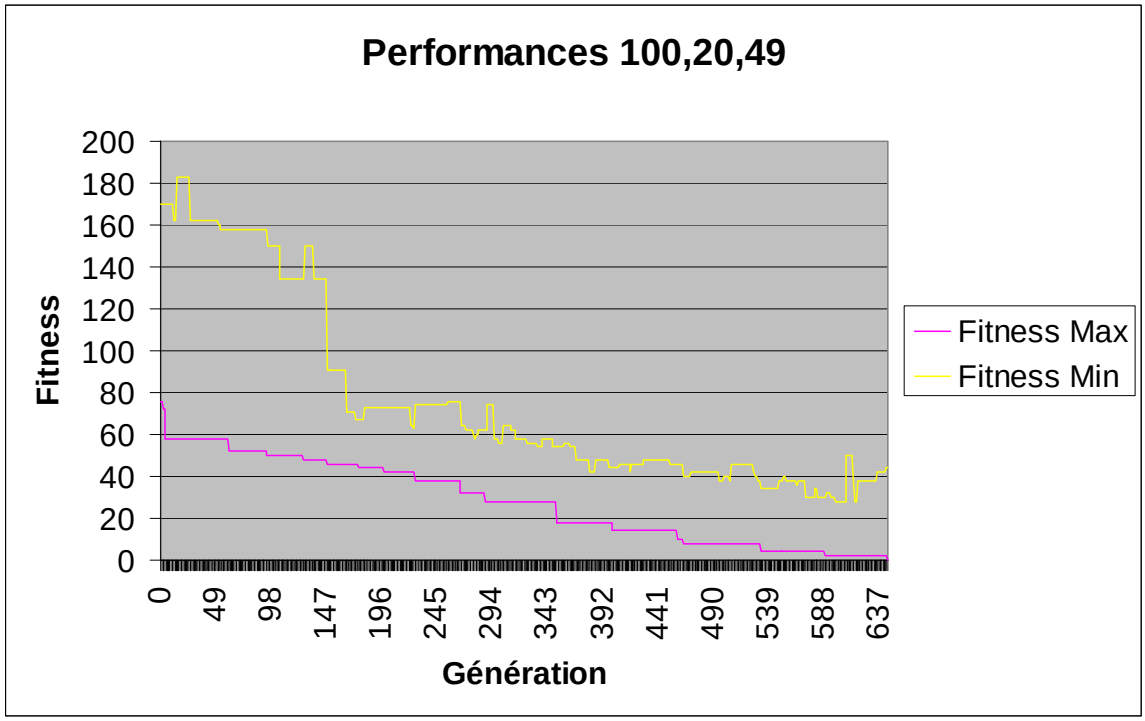
Diagramme de classe d'implémentation de notre programme

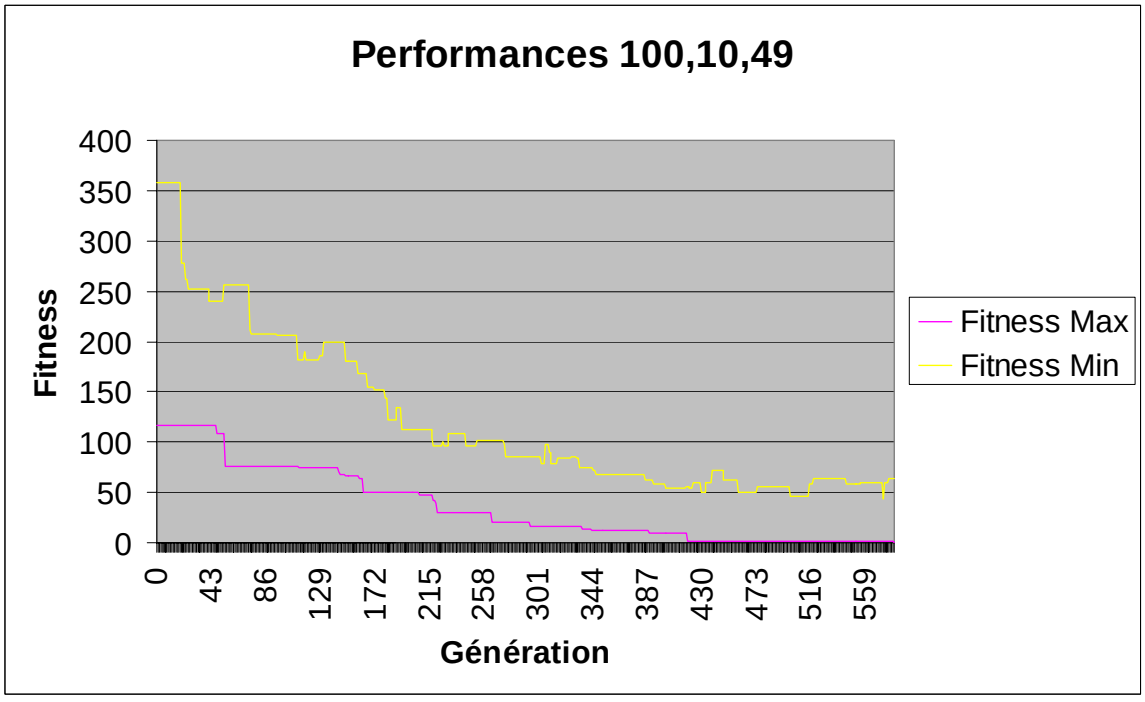
3. Présentation des résultats

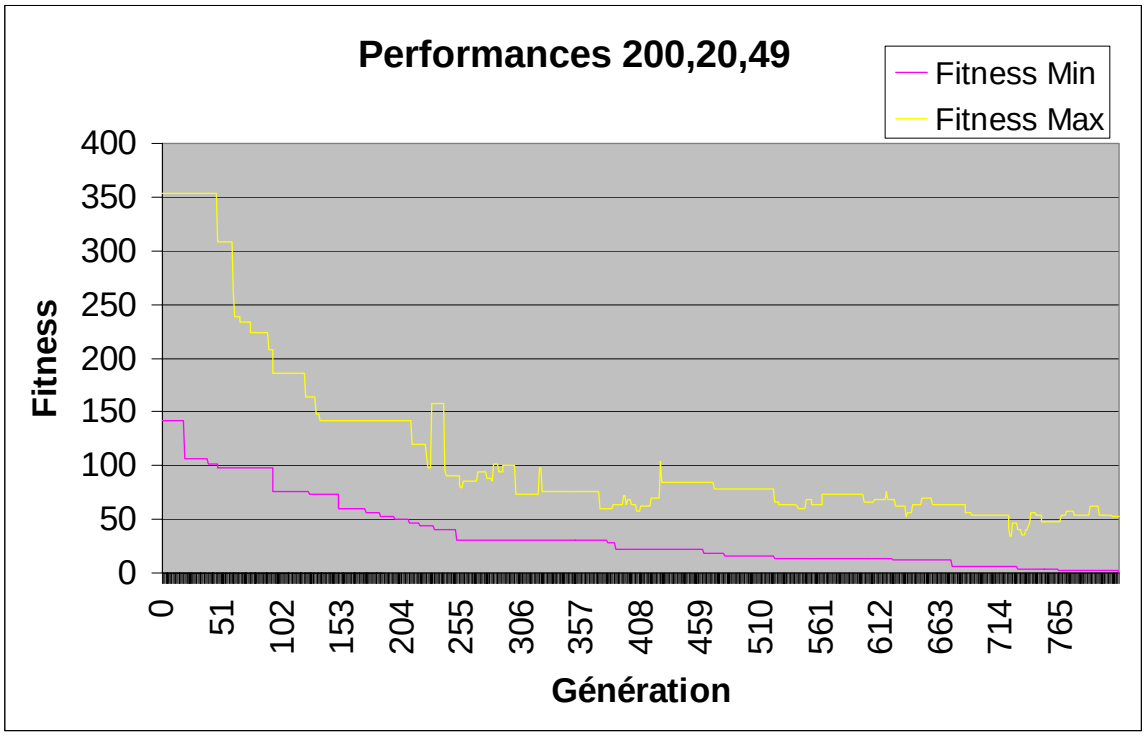
Les résultats sont présentés sous la forme suivante : nombre de ressources, nombre de zones, taille population. Ainsi le premier graphique correspond à une population de 49 individus, 20 ressources et 10 zones.

Nous avons effectué environ une dizaine de tests pour chaque configuration et uniquement les scénarios dans lequel l'algorithme a trouvé une solution optimale le plus rapidement sont représentés.




Performances 100,20,49
Fitness
200
180
160
140
120
100
80
60
40
20
0
0
49
98
147
196
245
294
343
392
441
490
539
588
637
Génération
Fitness Max
Fitness Min


Performances 100,10,49
Fitness
400
350
300
250
200
150
100
50
0
0
43
86
129
172
215
258
301
344
387
430
473
516
559
Génération
Fitness Max
Fitness Min



L'un des principaux problèmes que nous avons rencontré lors du paramétrage de cet algorithme génétique est le maintien de la diversité. Celle ci est représentée par la distance entre les fitness minimum et maximum.

En effet, nous avons d'abord observé une convergence trop rapide de la population sur un minimum local, la quasi totalité des individus étant alors presque identiques. Par conséquent, l'opérateur de croisement devient inefficace car il ne peut pas apporter de la diversité et l'algorithme boucle sans trouver la solution optimale.

Ainsi, nous avons joué sur deux moyens pour essayer de maintenir une certaine diversité:
- d'une part l'augmentation du taux de ranking, cela permet d'éviter de trop favoriser les bonnes solutions et donc de faire des croisements entre des bonnes solutions et des mauvaises, et pas uniquement des bons individus.
- ensuite, lorsque la fitness ne s'améliore pas pendant 200 générations on rajoute un individu aléatoire au milieu de la population. Cela permet de maintenir la diversité car

cela apporte des solutions originales par rapport à l'ensemble de la population.

Le résultat de cette diversification est visible sur les courbes de performances car on remarque que globalement l'écart entre le meilleur et le plus mauvais des individus est à peu près constant passé un certain nombre de générations.