

# INF6102 - Projet final

## Trouver la meilleure stratégie de 2048

---

Francis Deslauriers - 1540541

29 avril 2014

### 1. INTRODUCTION

Le jeu 2048 a bouleversé le web au cours du mois de mars 2014 [1]. Ce jeu de puzzle demande un certain de niveau de stratégie et a un certain potentiel adictif. Plusieurs variations et parodies de ce jeu ont vu le jour au cours de cette période comme la version traitant de la mascotte *Doge*[6] et la version surchargée [4]. On a aussi vu un site web présentant un algorithme d'intelligence artificielle offrant de très bonne performance à résoudre de casse-tête [5].

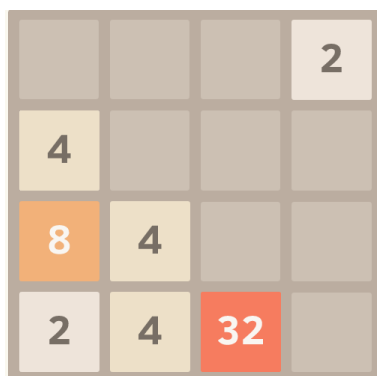
Dans le cadre de ce projet de session, j'ai développé un algorithme évolutionnaire tentant de trouver la meilleure tactique pour jouer au jeu vidéo "2048".

#### 1.1. DESCRIPTION DU JEU

Le jeu consiste à une matrice 4x4 pouvant contenir des tuiles numérotées avec des puissances de 2. La figure 1.1 en présente un exemple. Le joueur interagit avec la grille en faisant un des quatre mouvements admissibles c'est-à-dire haut, bas, gauche et droite. Lorsqu'un mouvement est appliqué, toutes les tuiles sont déplacées le plus que possible dans cette

direction. À chaque mouvement, une nouvelle tuile est ajoutée dans une case vide de la matrice de façon aléatoire.

Il est possible de fusionner deux tuiles adjacentes qui ont le même numéro en créant un mouvement particulier créant par le fait une tuile avec la somme des deux tuiles. Par exemple, sur la figure 1.1 on peut voir que deux tuiles portant le numéro 4 sont adjacentes. Dans cette situation nous pourrions appliquer un mouvement vers le bas pour fusionner ces deux tuiles pour obtenir une tuile 8. Comme son nom l'indique, le but ultime du jeu est d'obtenir une tuile numérotée 2048. Une partie se termine lorsqu'il n'y a plus aucune case vide sur la grille.



			2
4			
8	4		
2	4	32	

FIGURE 1.1 – Grille de jeu

## 1.2. TACTIQUE POUR JOUEUR HUMAIN

La stratégie la plus utilisée par les joueurs est d'éviter qu'une tuile ayant un numéro bas soit placée aux côtés d'une tuile avec un numéro significativement plus gros. Ce concept est souvent référé par la monotonie de la grille. Un joueur averti tentera de conserver la monotonie sur les deux axes. Par exemple, la figure 1.2a comporte une très bonne monotonie alors que la figure 1.2b en comporte une moins bonne. De plus, une stratégie importante est de concentrer ces actions dans trois des actions possibles. On pourrait, par exemple, faire un mouvement vers le haut que lorsqu'aucun autre mouvement n'est possible.

## 2. TECHNIQUES EXISTANTES

La recherche des meilleures stratégies pour jouer à des jeux à fait l'objet de plusieurs articles de recherches dans le passé.

Premièrement, on peut penser par exemple à l'article de Hochmuth[3] où celui-ci tente de trouver grâce à un algorithme génétique quelle est la meilleure stratégie pour gagner une partie de Tic Tac Toe. Dans son implémentation, chaque individu de la population est constitué d'un chromosome par configuration du plateau de jeu. Ainsi, pour chaque configuration du jeu il y a une allèle dans le chromosome pour dicter quelle action prendre.

			2
		2	4
	2	4	16
	8	16	128

(a) Bonne monotonie

		2	4
2		8	16
2	128	2	64
4	2	64	8

(b) Mauvaise monotonie

FIGURE 1.2 – Exemples de monotonie

Il utilise une fonction d'évaluation des individus basée sur le rapport des parties gagnées ou égalisées sur le nombre de parties jouées comme présentée à l'équation 2.1.

$$f(x_i) = \frac{n_{total}[x_i] - n_{perdue}[x_i]}{n_{total}[x_i]} \quad (2.1)$$

Contrairement à ce qu'il est habituellement le cas le croisement qu'il a utilisé était multi-points. En effet, chaque gènes il y a une probabilité qu'un point de croisement se débute(ou se termine). Il a utilisé cette méthode étant donnée le nombre très grand de chromosomes, un croisement plus traditionnel à deux points aurait limité la diversification des solutions.

Deuxièmement, une équipe de l'université du Nevada a publié un article traitant du développement de stratégie pour intelligence artificielle pour le jeu de tir à la première personne *Counter Strike* en utilisant un algorithme génétique[2]. Ils ont utilisé un vecteur de coefficient deux coefficients avec une représentation sur 178 bits chacun. Ils ont obtenu de bon résultats malgré le nombre limité de critères.

### 3. NOUVELLES TECHNIQUES DE RÉOLUTION PROPOSÉES

Dans cette section, je vais discuter des décisions que j'ai prises au cours du développement de cette algorithme. Pour implémenter cette algorithme, j'ai utilisé un projet sous license libre comprenant une implémentation très bien faite de la logique de jeu[7].

#### 3.1. DÉFINITION D'UN INDIVIDU

Comme dans tous algorithmes évolutionnaires, les configurations à explorer sont nommés des individus. Les individus dans mon algorithmes sont composés de six chromosomes qui représentent la réaction plus ou moins favorable de l'individu face à une observation de l'environnement (grille de jeu). Le chromosome est composé de coefficients qui seront multipliés par un critère observé sur la grille. Le tableau 3.1 présente les 6 critères que j'ai

utilisé.

Numéro	Nom du critère	Description
1	Tuile Max	La valeur de la plus grosse tuile présente
2	Nombre tuile	Nombre de tuiles sur le jeu
3	Somme des tuiles	La somme des valeurs des tuiles sur le jeu
4	Monotonie en X	Somme des changements d'orientation du gradient sur les rangées de la grille
5	Monotonie en Y	Somme des changements d'orientation du gradient sur les colonnes de la grille
6	Coin	(booléen) Présence de la plus grosse tuile dans un des quatre coins.

TABLE 3.1 – Résumé des chromosomes

Donc, avant d'effectuer un mouvement, un individu va évaluer l'effet des quatre mouvements(haut, bas, gauche et droite) possibles en observant ces critères et en les multipliant avec les coefficients qui lui sont propres. On retrouve alors la somme présentée à l'équation 3.1. Dans cette équation,  $\lambda_i$  et  $x_i$  représentent le coefficient et la l'observation concernant le critère  $i$ .

$$score d'un mouvement = \sum_{i=0}^n \lambda_i \times x_i \quad (3.1)$$

Par exemple, pour le critère du nombre de tuile sur la grille, l'individu comptera le nombre de tuiles sur la grille et multipliera ce nombre par le coefficient associé à ce critère.

Le mouvement avec le score le plus élevé sera choisi comme celui le plus prometteur. Afin de contrer les effets des ajouts aléatoires de tuiles au cours de la partie, l'individu évalue dix essais de chacun des mouvements avant de sélectionner celui avec le plus grand score.

### 3.2. NORMALISATION DU CHROMOSOME

Après quelques tests préliminaires, j'ai remarqué que de par mon implémentation des chromosomes, les coefficients tendaient vers la borne maximale imposée. Donc, à chaque fois qu'un croisement ou une mutation a lieu je normalise les chromosomes en divisant chaque allèle par la somme de toutes les allèles. De plus, comme le but de l'étude courante est de trouver quelle stratégie est la plus performante, cette normalisation permet de discriminer chacun des critères par rapport aux autres. On verra dans une prochaine section un autre impact de cette normalisation.

### 3.3. GESTION DE LA POPULATION

Dans cette sous-section, je discuterai de la gestion de population que j'ai privilégié au cours de ce projet. Le schéma générationnel que j'ai choisi est de la forme  $(\mu + 1) - ES$ . La population est constitué de cinq individus qui représente les parents potentiels de la génération suivante. À chaque génération, deux parents sont croisés pour engendrer un enfant. Cette enfant subit alors une mutation avant d'être évalué et ajouté dans la population. La sélection se fait sur la population en son entier (parents + enfant).

Le tableau 3.2 présente un résumé du schéma de gestion de population que j'ai utilisé pour cet algorithme.

Nombre de parent	5
Nombre d'enfant	1
Recouvrement	Oui
Remplacement	Truncation

TABLE 3.2 – Schéma gestion de population

### 3.4. CROISEMENT

Chaque nouvel individu de la population est créé à partir de deux parents choisis de façon aléatoire dans la population courante. Le nouveau chromosome est créé en choisissant une valeur aléatoire selon une distribution uniforme entre les deux valeurs de chacune des allèles parentales.

### 3.5. MUTATION

L'opérateur de mutation que j'ai utilisé applique une mutation sur un seul chromosome choisi aléatoirement selon une probabilité de  $2/3$ . Cette mutation consiste à l'ajout (ou de la soustraction) d'un pourcentage de l'allèle courante de ce chromosome.

### 3.6. SÉLECTION

Les configurations sont évaluées en comparant la somme des scores des 10 essais effectués au cours de cette génération. Comme il y a une part important d'événements aléatoires lors d'une partie, il est important que ce nombre d'essais soit suffisamment grand afin de pouvoir trancher sur la réelle capacité d'un individu à gagner une partie.

## 4. TESTS EXPÉRIMENTAUX

Dans cette section, je vais présenter mes expérimentations pour analyser la performance de mon algorithme ainsi que les résultats associés.

## 4.1. DESCRIPTIONS DES TESTS

### 4.1.1. EFFETS DES CRITÈRES

Afin de bien comprendre l'apport de chacun des critères, j'ai d'abord effectué trois tests avec différentes combinaisons de critères de décision. Le premier de ces tests a été fait avec des individus qui ne considéraient que trois des six critères pour faire leur choix de mouvement. Le second en comprenait cinq et le dernier la totalité. Il est important de noter que ces tests ont été fait avec la stratégie de gestion de population présentée à la section 3.3. C'est-à-dire, une taille de population de cinq et une production d'enfant par génération. Le tableau 4.1 présente en détail quels critères sont considérés par les individus pour chacun des tests.

Numéro du test	Critères considérés
1	Tuile Max Nombre de tuile Somme des tuiles
2	Tuile Max Nombre de tuile Somme des tuiles Monotonie X Monotonie Y
3	Tuile Max Nombre de tuile Somme des tuiles Monotonie X Monotonie Y Coin

TABLE 4.1 – Plan de tests sur les critères

### 4.1.2. EFFETS DES PARAMÈTRES DE LA GESTION GÉNÉRATIONNELLE

J'ai voulu ensuite comprendre l'effet du nombre d'individu dans la population et du nombre d'enfant ajouté à chaque génération. J'ai donc fait trois tests en faisant varier ces paramètres. Le tableau 4.2 présente les schémas que j'ai testé avec mon algorithme. Les tests 4

	Schéma générationnel	
Numéro du test	Nb individus	Nb enfants
4	10	1
5	20	1
6	20	3

TABLE 4.2 – Plan de tests sur les paramètres

et 5 donneront une idée des effets de la taille de la population sur la qualité des solutions trouvées. D'un autre côté, les tests 5 et 6 mettront en évidence les effets de la variation du nombre d'enfant à chaque génération.

## 4.2. RÉSULTATS

### 4.2.1. EFFETS DES CRITÈRES

En annexe les tableaux A.1 à A.3 présentent la valeur moyenne et l'écart-type des chromosomes des individus ayant récolté le plus haut score moyen à chaque génération pour les tests présentés au tableau 4.1.

On peut remarquer que dans ces trois situations, le coefficient qui semble avoir été favorisé est celui du nombre de tuiles sur la grille. De plus, la monotonie verticale et horizontale ne semblent pas avoir la même importance. Aussi, les coefficients moyens du test No.1 ont tous des valeurs similaires ce qui est inattendu.

Les graphiques A.7 à A.9 présentent l'évolution du meilleur score moyen pour les 200 premières générations. Tout d'abord, il est important de remarquer les valeurs initiales des courbes de tendance. En effet, lors du test No.1 en plus d'être basse, la qualité des solutions trouvées est restée stable tout le long des 200 générations. De plus, lors des tests No.2 et No.3 les résultats obtenus sont comparables. Bien que le graphique du test No.2 à une pente significativement plus élevée que celui du test No.3 sur ces deux figures, on peut voir sur les figures A.14 et A.15 que la tendance changent quand on considère les 1000 générations.

### 4.2.2. EFFETS DES PARAMÈTRES DE LA GESTION GÉNÉRATIONNELLE

Les graphiques A.10 à A.12 présentent l'évolution du meilleur score moyen au fil des 200 premières générations en selon plusieurs schémas générationnelles. Les figures A.13 à A.16 présentent cette même évolution mais avec la totalité des 1000 générations effectuées.

Le graphique A.10 montre qu'on atteint un plateau autour de la 80e génération. Plateau qui n'est pas présent sur les deux autres essais avec un plus grand nombre d'individu dans la population. Le taux de variation de la droite de tendance est un peu plus élevé pour le schéma avec un enfant qu'à celui avec 3 enfants.

### 4.2.3. TESTS SUR INDIVIDUS MOYENS

J'ai également fait des tests sur les critères moyens présentée précédemment. J'ai effectué 10000 itérations essais par individus moyens dans le but de trouver le score moyen ainsi que le nombre de partie gagnée. Le tableau 4.1 en présente les résultats. Dans ce tableau, la colonne nombre de victoire représente le nombre de fois que cet individu à atteint une tuile de valeur 2048.

Numéro du test	Score moyen	Nombre de victoire
No.1	4148.07	0
No.2	7754.54	23
No.3	8085.43	110
<b>No.4</b>	<b>8374.73</b>	<b>143</b>
No.5	8166.08	125
No.6	8127.24	110

FIGURE 4.1 – Score moyen et nombre de victoire des individus moyens selon les cas de tests pour 10000 itérations

## 5. ANALYSE

Dans cette section, je vais analyser les résultats recueillis dans la section 4 et la performance générale de mon implémentation.

Premièrement, comme on peut s'en douter l'utilisation des critères 1 à 3 n'est pas suffisante pour obtenir de bons résultats. En effet, les résultats du test No.1 sont presque la moitié des résultats que les tests No.2 et No.3. De plus, comme lors de ces tests certains critères n'étaient pas utilisés je me serais attendu que la valeur des allèles correspondantes tendent vers zéro, ce qui n'est pas le cas.

Deuxièmement, le nombre idéal d'individu semble être dix, comme le montre la figure 4.1. En effet, il semblerait qu'une population de cinq individus soit moins efficace. Cela pourrait s'expliquer par le manque de diversité. De l'autre côté, 20 individus dans la population semble également être inapproprié. Cela pourrait s'expliquer par une trop grande diversité des allèles et donc nombre population ne converge pas.

Finalement, on remarque que le critère du nombre de tuile comporte souvent l'un des coefficients les plus élevés, ce qui porte à croire que tenter de réduire le nombre de tuile est une bonne stratégie.

## 6. CONCLUSION

Au cours de ce projet, j'ai développé un algorithme génétique dans le but de trouver la meilleure stratégie pour jouer au jeu 2048. J'ai utilisé des coefficients réels pour représenter la préférence d'individu pour une certaine situation de la grille. Les critères que j'ai choisis ne semblent pas suffisants pour statuer sur la meilleure stratégie à utiliser pour gagner à ce jeu.

Dans des améliorations futures, il serait intéressant pour les individus de considérer plus d'un coup à l'avance avant de choisir un mouvement. De plus, l'utilisation de coefficients entiers serait à explorer.



## RÉFÉRENCES

- [1] Gabriele Cirulli. 2048. <http://gabrielecirulli.github.io/2048/>. Visité : 2014-04-25.
- [2] Nicholas Cole, Sushil J Louis, and Chris Miles. Using a genetic algorithm to tune first-person shooter bots. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 1, pages 139–145. IEEE, 2004.
- [3] Gregor Hochmuth. On the genetic evolution of a perfect tic-tac-toe strategy. *Genetic Algorithms and Genetic Programming at Stanford*, pages 75–82, 2003.
- [4] Inconnu. 90071992547409920. <http://www.csie.ntu.edu.tw/~b01902112/9007199254740992/>. Visité : 2014-04-27.
- [5] ov3y. 2048-ai. <http://ov3y.github.io/2048-AI/>. Visité : 2014-04-28.
- [6] The tiny Mammals. doge2048. <http://doge2048.com/>. Visité : 2014-04-27.
- [7] Ólafur Waage. ML2048. <https://github.com/olafurw/ML2048>. Visité : 2014-04-24.

## A. ANNEXE

### A.1. LISTE DES TRAVAUX RÉALISÉS

#### A.1.1. SOURCES UTILISÉES

Au cours de ce projet, j'ai utilisé une implémentation de la logique du jeu 2048 existante[7]. Cette implémentation me permet d'accéder à la planche de jeu, d'effectuer un mouvement et de connaître le score en tout temps. Le code de ce projet est très bien fait et j'ai pu facilement l'adapter à mes besoins.

#### A.1.2. APPORT PERSONNEL

J'ai dû modifier quelques peu la classe *grid* du projet ML2048[7] pour y ajouter un constructeur par copie. J'ai créé une classe nommée *agent* qui comprend un vector de coefficients, des méthodes pour évaluer les critères selon l'état de la grille et une méthode pour décider un mouvement à effectuer. J'ai également développé les opérateurs de croisement et de mutation en me basant sur des discussions avec le professeur du cours et les notes de cours. Tout ce rapport a été rédigé pour ce travail.

## A.2. TABLEAUX DE RÉSULTATS

Critère	Tuile Max	Nb tuile	Somme tuile	Mono X	Mono Y	Coin
Moyenne	0.168	0.157	0.167	0.167	0.164	0.176
Écart-Type	0.082	0.080	0.087	0.074	0.074	0.082

FIGURE A.1 – Moyenne et écart-type des coefficients des meilleurs individus pour le test No.1

Critère	Tuile Max	Nb tuile	Somme tuile	Mono X	Mono Y	Coin
Moyenne	0.183	0.255	0.181	0.058	0.154	0.169
Écart-Type	0.088	0.072	0.076	0.025	0.061	0.094

FIGURE A.2 – Moyenne et écart-type des coefficients des meilleurs individus pour le test No.2

Critère	Tuile Max	Nb tuile	Somme tuile	Mono X	Mono Y	Coin
Moyenne	0.181	0.272	0.130	0.043	0.184	0.190
Écart-Type	0.086	0.079	0.071	0.029	0.060	0.081

FIGURE A.3 – Moyenne et écart-type des coefficients des meilleurs individus pour le test No.3

Critère	Tuile Max	Nb tuile	Somme tuile	Mono X	Mono Y	Coin
Moyenne	0.166	0.302	0.180	0.038	0.158	0.156
Écart-Type	0.090	0.078	0.087	0.021	0.064	0.078

FIGURE A.4 – Moyenne et écart-type des coefficients des meilleurs individus pour le test No.4

Critère	Tuile Max	Nb tuile	Somme tuile	Mono X	Mono Y	Coin
Moyenne	0.190	0.230	0.185	0.033	0.140	0.222
Écart-Type	0.104	0.068	0.095	0.023	0.084	0.074

FIGURE A.5 – Moyenne et écart-type des coefficients des meilleurs individus pour le test No.5

Critère	Tuile Max	Nb tuile	Somme tuile	Mono X	Mono Y	Coin
Moyenne	0.136	0.283	0.157	0.040	0.197	0.187
Écart-Type	0.086	0.070	0.076	0.028	0.064	0.074

FIGURE A.6 – Moyenne et écart-type des coefficients des meilleurs individus pour le test No.6

### A.3. GRAPHIQUES

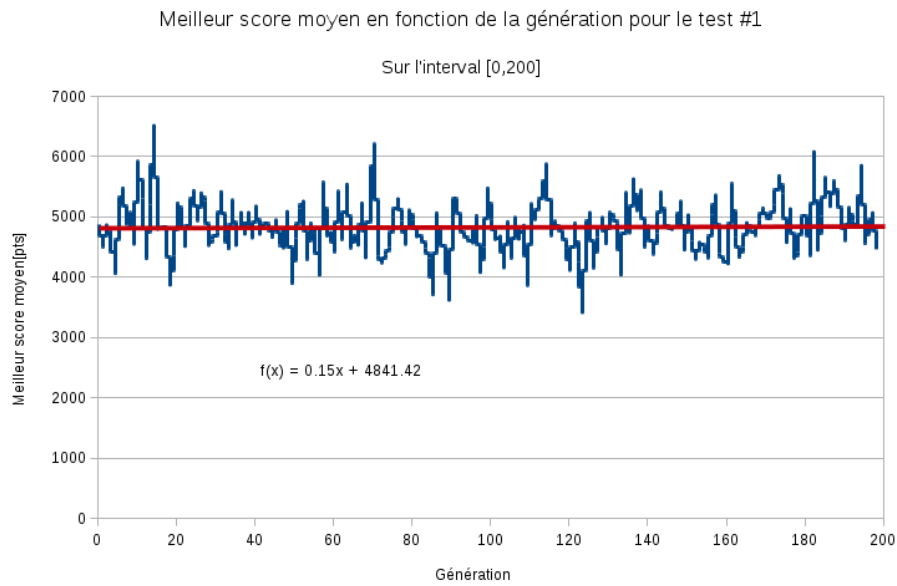


FIGURE A.7 – Évolution de la meilleure solution moyenne lors du test No.1 pour 200 générations

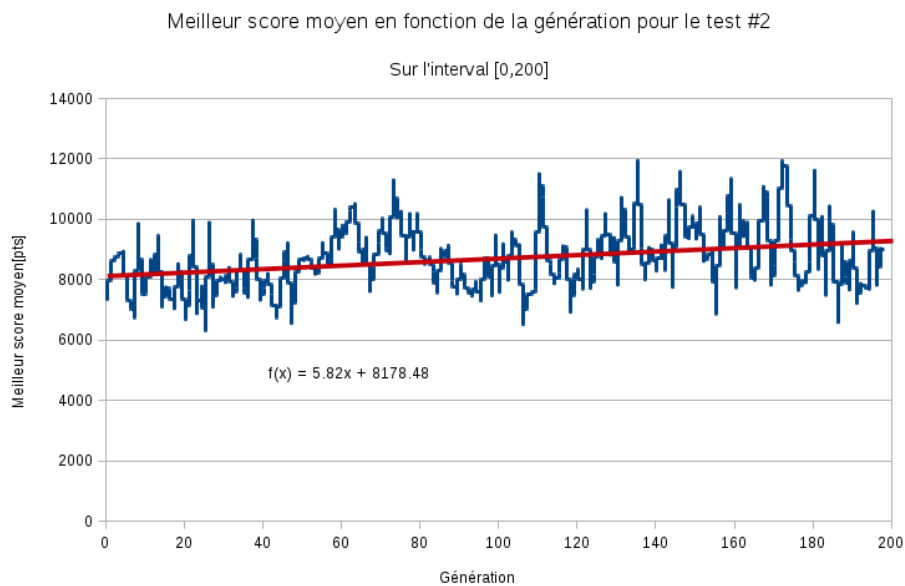


FIGURE A.8 – Évolution de la meilleure solution moyenne lors du test No.2 pour 200 générations

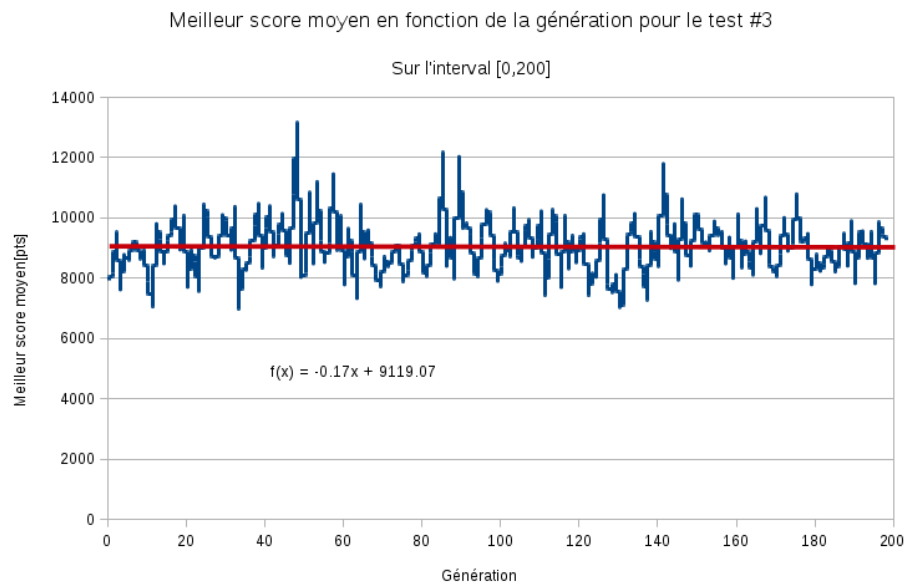


FIGURE A.9 – Évolution de la meilleure solution moyenne lors du test No.3 pour 200 générations

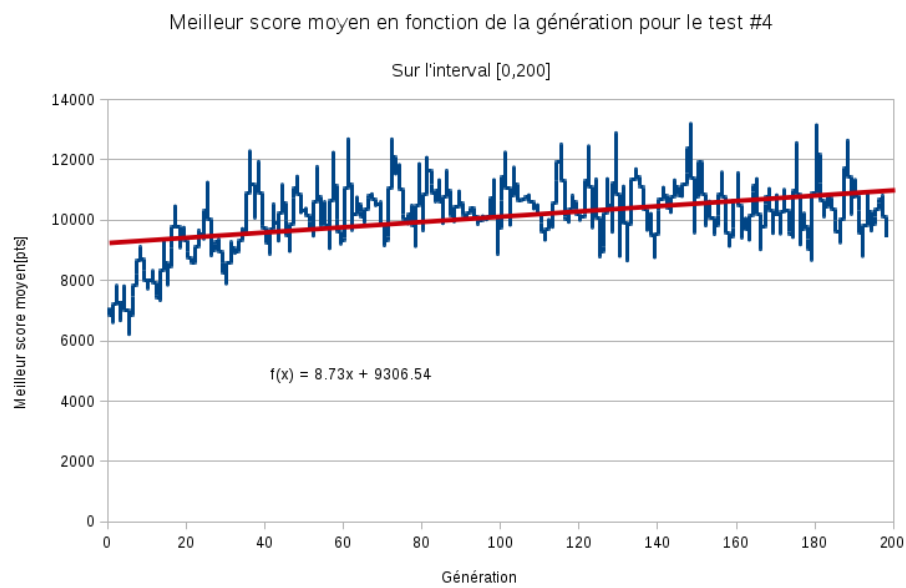


FIGURE A.10 – Évolution de la meilleure solution moyenne lors du test No.4 pour 200 générations

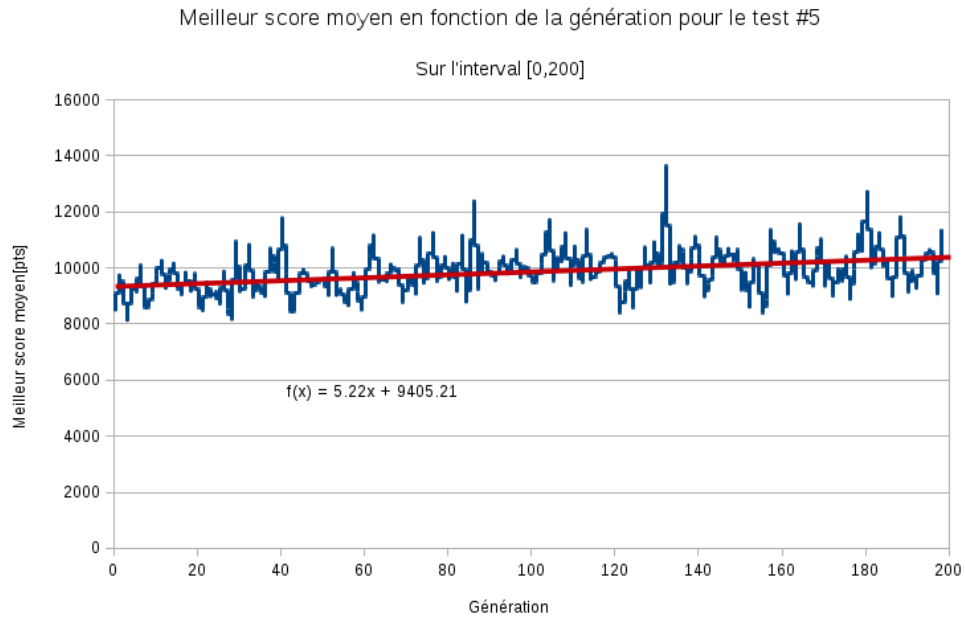


FIGURE A.11 – Évolution de la meilleure solution moyenne pour le test No.5 pour 200 générations

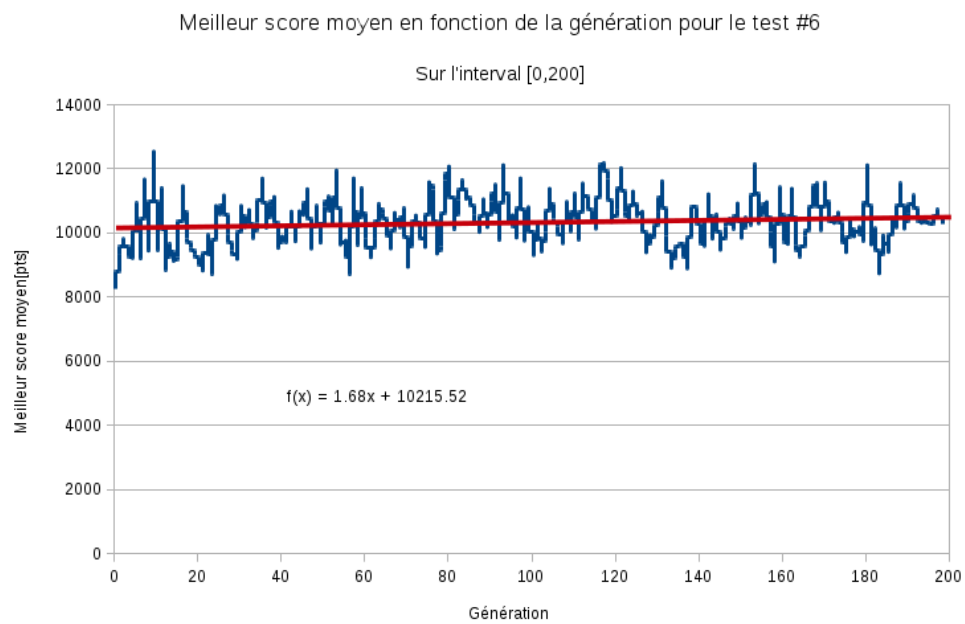


FIGURE A.12 – Évolution de la meilleure solution moyenne pour le test No.6 pour 200 générations

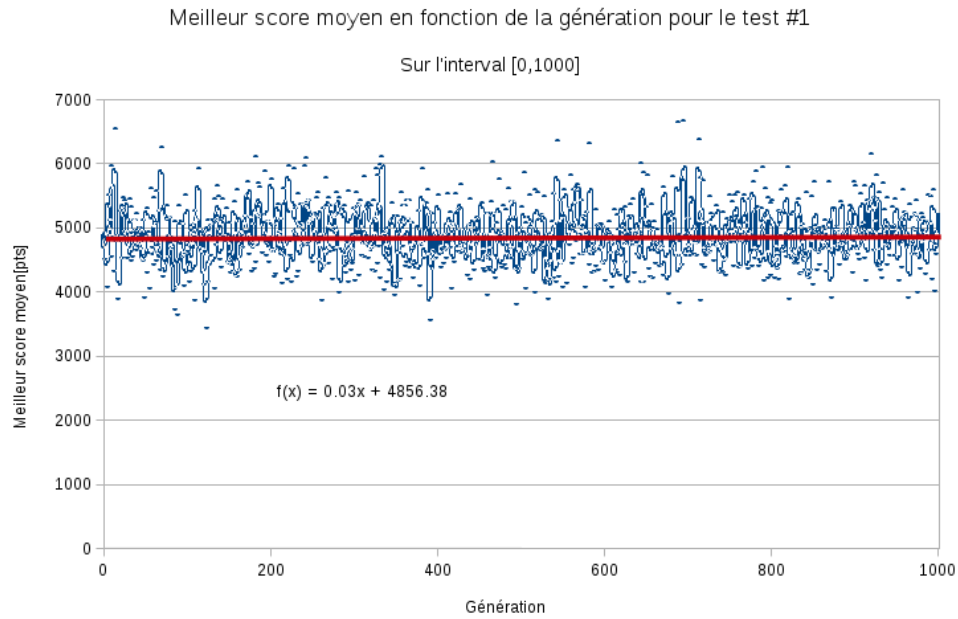


FIGURE A.13 – Évolution de la meilleure solution moyenne lors du test No.1 pour 1000 générations

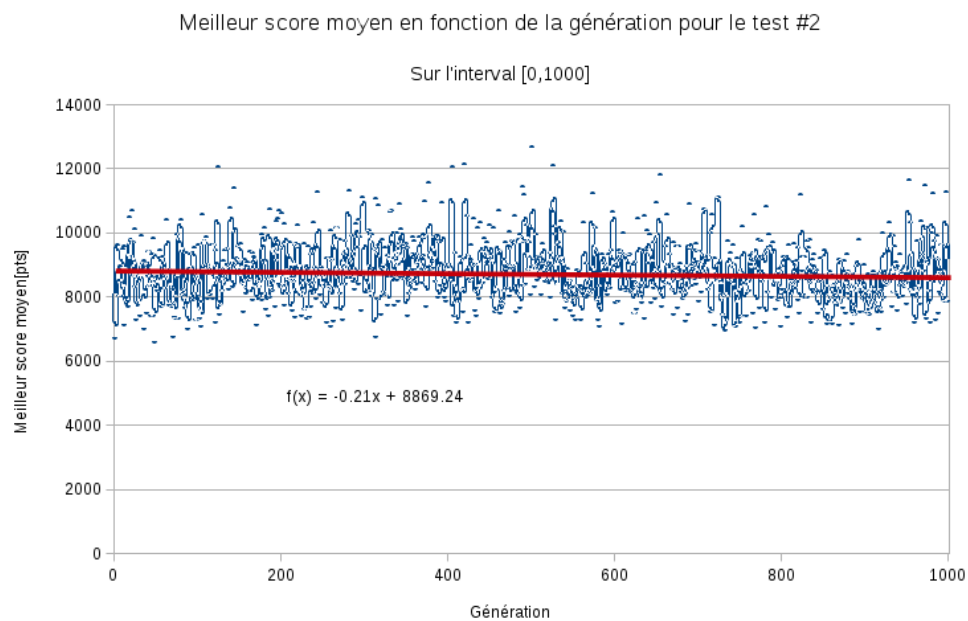


FIGURE A.14 – Évolution de la meilleure solution moyenne lors du test No.2 pour 1000 générations

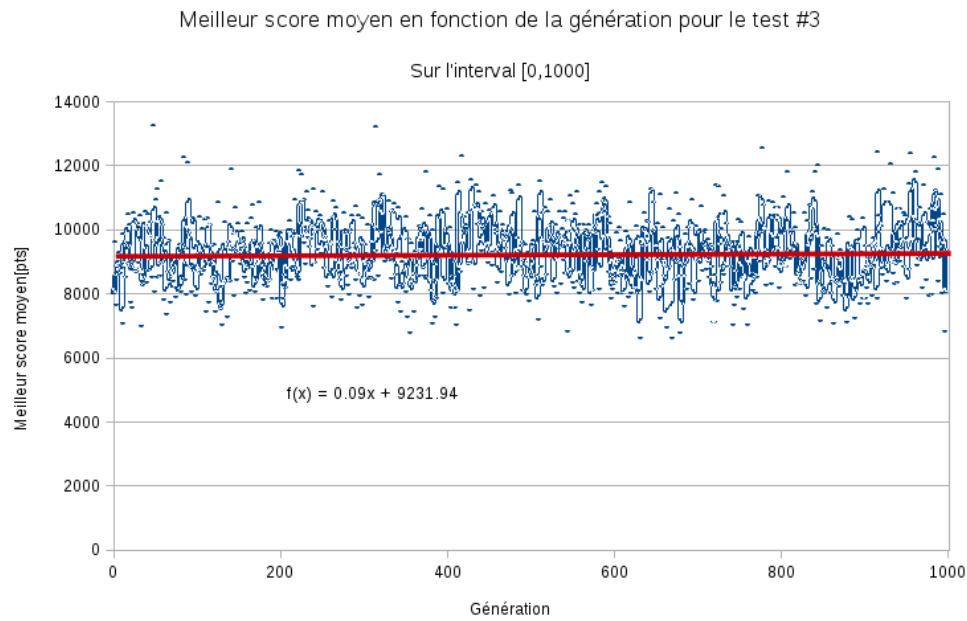


FIGURE A.15 – Évolution de la meilleure solution moyenne lors du test No.3 pour 1000 générations

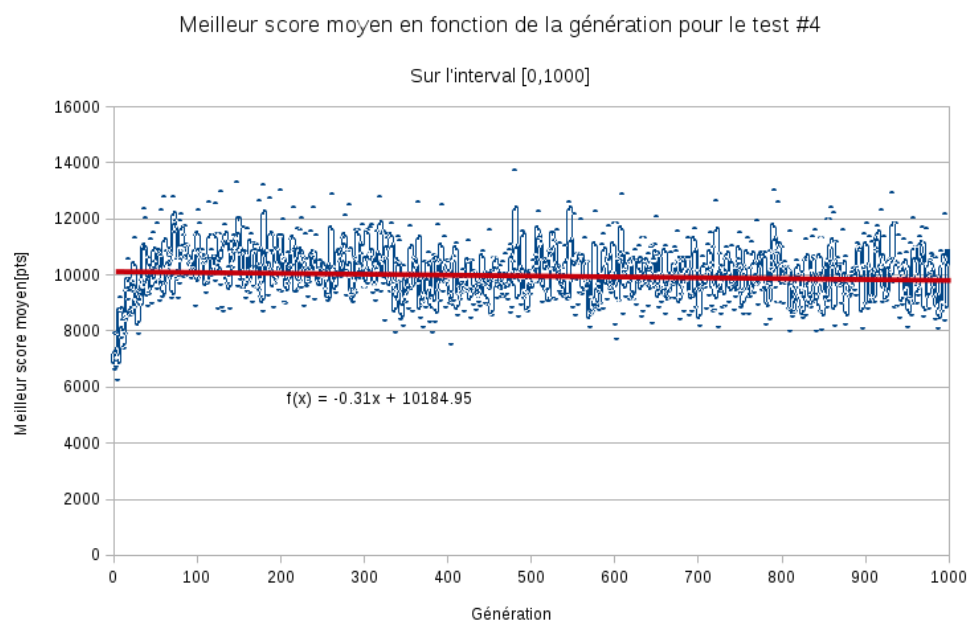


FIGURE A.16 – Évolution de la meilleure solution moyenne lors du test No.4 pour 1000 générations