Algorithme d'illumination pour l'exploration de solutions sous contrainte de fonctions d'évaluations stochastiques

Qi Qiu & Jean Lapostolle

21 mai 2019

Table des matières

1 Introduction

Dans les sciences et l'ingénierie, on utilise souvent les algorithmes pour chercher les solutions optimales d'un problème. Ces problèmes sont parfois très difficiles, typiquement on ne connaît pas la performance d'une solution avant de l'évaluer, autrement dit, on ne connaît pas de formule mathématique donnant la performance d'une solution. Ce type de problème nécessite la "black box" optimisation, au sens ou on cherche l'optimal global dans un environnement inconnu dont la seule façon de connaître le résultat d'une action est de la faire. Comme les autres types de problèmes difficiles, il existe aussi des benchmarks pour tester les algorithmes de boite noire, par exemple le domaine de labyrinthe. Le domaine des labyrinthes est très similaire aux problèmes d'optimisation boite noire, car ils admettent la même difficulté principale : la présence d'optima locaux. C'est donc sur le domaine de labyrinthe que l'on va tester notre algorithme d'optimisation boite noire.

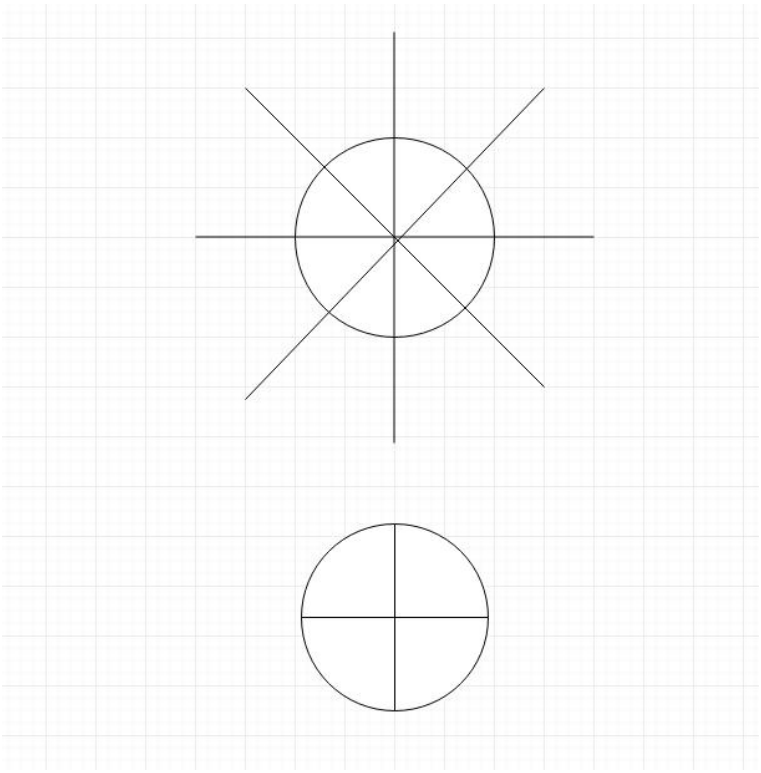
Dans un premier temps, on essaie de résoudre le labyrinthe avec une recherche heuristique qui minimise la distance euclidienne entre la position actuelle et la position finale. Puis, on travaille la recherche par nouveauté proposée par Joel Lehman et Kenneth O.Stanley en 2008[6]. Cette approche est l'autre extrémité de la recherche heuristique, car elle abandonne complètement l'objectif. Finalement, on essaie de combiner la recherche par nouveauté avec l'algorithme d'illumination proposée par Jean-Baptiste Mouret et Jeff Clune en 2015[1]. Ce type d'algorithme n'essaie pas de retourner un optimal global, mais de retourner des archives des solutions qui donne une estimation de la meilleure performance qu'on peut obtenir dans un zone qui organise en une case des archives.

2 Simulateur

2.1 Réseau neurone

Le robot est un Perceptron multi-couche à 3 couches : une couche d'entrée de taille 17, une couche intermédiaire de taille 13 et une couche de sortie de taille 1. La fonction d'activation pour la couche intermédiaire est la fonction tangente hyperbolique tandis qu'on utilise la fonction Sigmoïde pour la couche de sortie. Il prend une entrée $(x_1, x_2, ..., x_{16})$, dont $(x_1, ..., x_8)$ sont les données des télémètres lasers, $(x_9, ..., x_{16})$ sont les données des radars et x_{17} est le biais, et renvoi une sortie qui est l'angle que le robot va tourner avant de s'avancer sur le labyrinthe.

2.2 Senseur



3 Méthode

3.1 Préliminaire

Croisement Un individu est représenté par un réseau de neurones et le dernier est défini par l'ensemble de poids de ces arcs. Donc le croisement se fait entre deux réseaux de neurones.
On énumère tous les arcs de 1 a 217. Le croisement d'un individu de poids des arcs

$$(x_1, x_2, ..., x_{217})$$

et un individu de poids des arcs

$$(y_1, y_2, ..., y_{217})$$

donne deux individu

$$(z_1, z_2, ..., z_{217})$$

et

$$(t_1, t_2, ..., t_{217})$$

avec $\{z_i, t_i\} = \{x_i, y_i\}$ et qu'il y a $\frac{1}{2}$ de chance que $z_i = x_i \forall i$

Mutation Choisit par hasard un nombre k en entre 1 et 217, et puis changer multiplier k arcs choisis par hasard par c_k avec $2 \geq c_k \geq 0.5 \ \forall k$

3.2 Recherche heuristique

1. Générer une population de 250 individus ;
2. jusqu'à évaluer 1000 générations ou un robot atteint le but :
   (a) simulation de la population dans le labyrinthe ;
   (b) calculer la distance euclidienne entre la position finale de chaque individu et le position but comme le score de chaque individu ;
   (c) choix des individus selon leurs scores ;
   (d) reproduction des individus sélectionnés par croisement ;
   (e) mutation des descendances ;
   (f) remplacement de la population initiale par une nouvelle population.

3.3 Recherche par nouveauté

1. Générer une population de 250 individus ;
2. jusqu'à évaluer 1000 générations ou un robot atteint le but :
   (a) simulation de la population dans le labyrinthe et archiver toutes les positions atteintes par un robot ;
   (b) pour chaque individu calculer la somme des distances entre sa position finale et les 20 plus proches positions présentes dans les archives comme sa nouveauté, s'il atteint une position inexplorée, il obtient une récompense de 10000 nouveautés ;
   (c) choix des individus selon leurs nouveautés ;
   (d) reproduction des individus sélectionnés par croisement ;
   (e) mutation des descendances ;
   (f) s'il y a plus de 1250 dans les archives, on calcule pour chaque position la somme des distances avec les 20 plus proches positions présentes dans les archives ; on ne garde que les 1250 positions de scores les plus hauts.

3.4 Recherche par nouveauté et Map-élite

1. Générer une population de 1000 individus ;
2. simulation des 1000 individus dans le labyrinthe et archiver tous les individus dans une archive des individus du passé et toutes les positions atteintes par un individu dans une archive des positions ;
3. pour chaque individu présents dans les archives des individus du passé, affecter la nouveauté comme la somme des distances entre sa position finale et les 20 plus proches positions présentes dans les archives ;
4. jusqu'à évaluer 1000 générations ou un robot atteint le but :
   (a) choisir les 250 individus présents dans les archives des individus de nouveauté plus haut
   (b) reproduction des individus sélectionnés par croisement
   (c) mutation des descendances
   (d) simulation des 1000 individus dans le labyrinthe et archiver tous les individus dans une archive des individus du passé et toutes les positions atteintes par un individu dans une archive des positions ;

(e) pour chaque individu calculer la somme de ses distances avec les 20 plus proches positions présentes dans les archives comme sa nouveauté et de plus s'il atteint une position inexplorée, il obtient une récompense de 10000 nouveautés

(f) s'il y a plus de 1250 position dans les archives des positions, on calcule pour chaque position dans les archives des positions la somme des distances avec les 20 plus proches positions présentes dans les archives ; on ne garde que les 1250 positions de scores les plus hauts.

4 Expérience et Analyse

4.1 Recherche guidée par fonction d'évaluation

4.1.1 1^{re} exécution

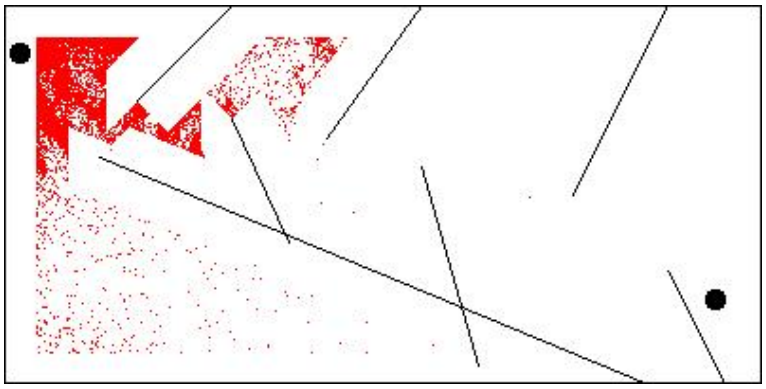


FIGURE 1 – 62500 évaluations

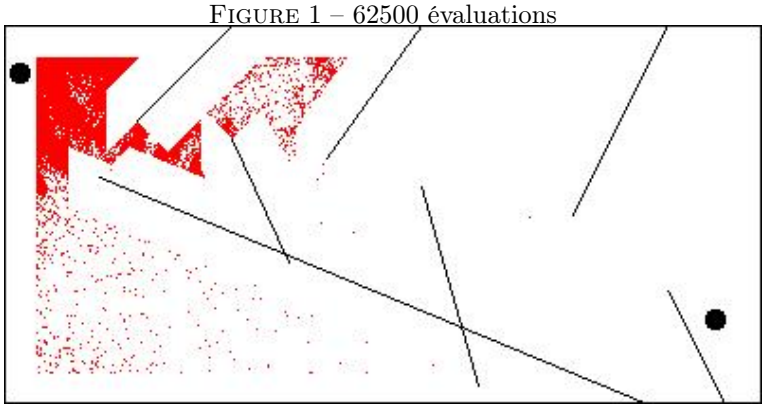


FIGURE 2 – 125000 évaluations

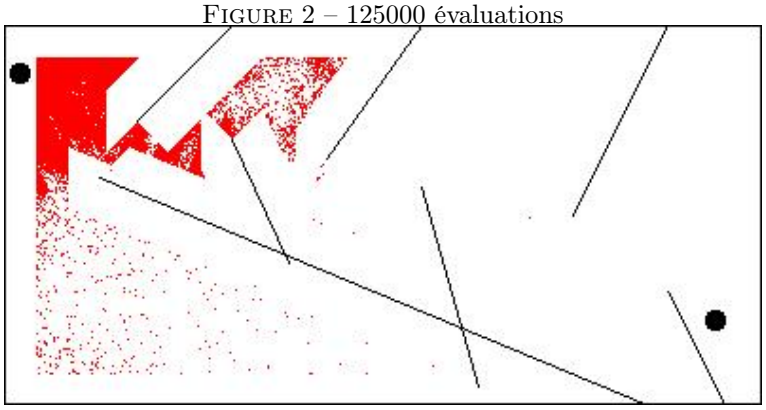


FIGURE 3 – 250000 évaluations

4.1.2 2^{me} exécution

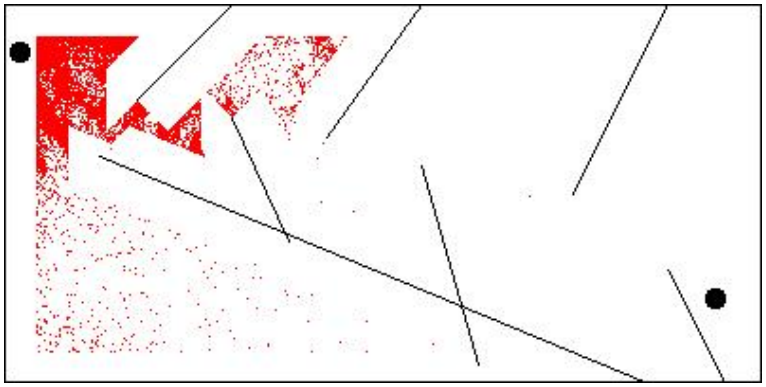


FIGURE 4 – 62500 évaluations

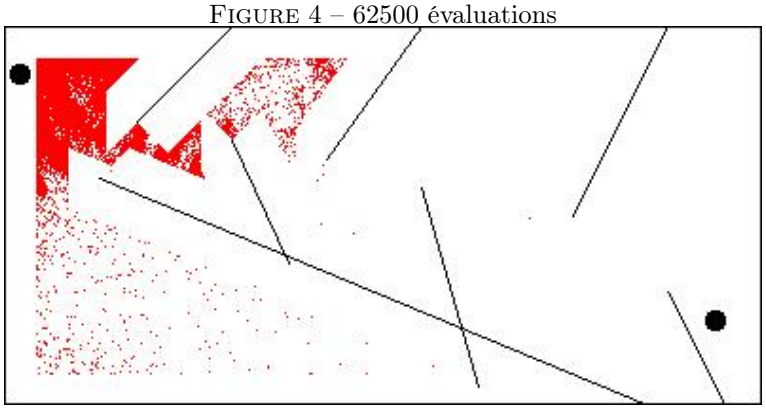


FIGURE 5 – 125000 évaluations

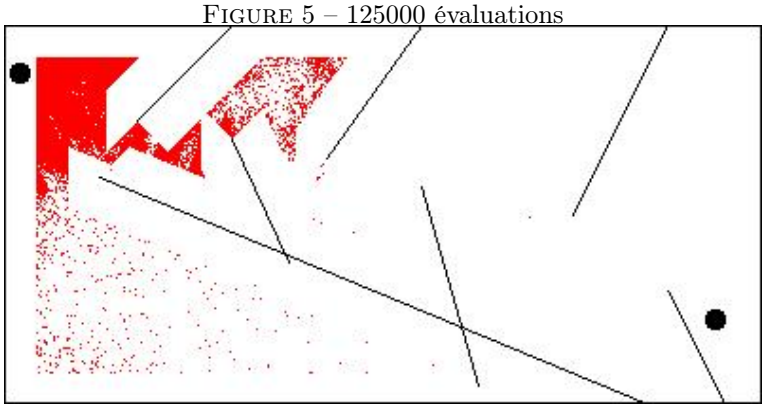


FIGURE 6 – 250000 évaluations

4.1.3 3^{me} exécution

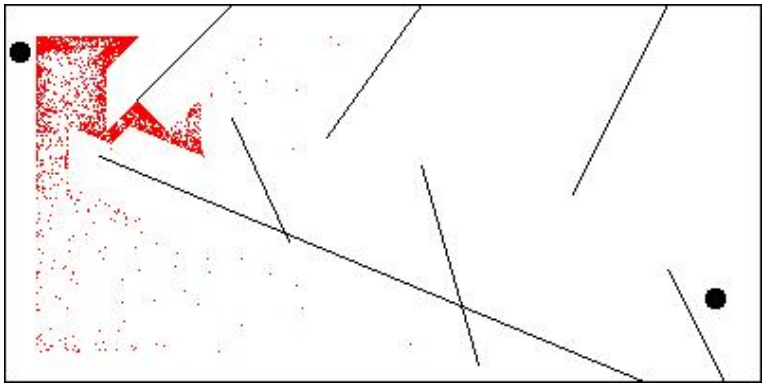


FIGURE 7 – 62500 évaluations

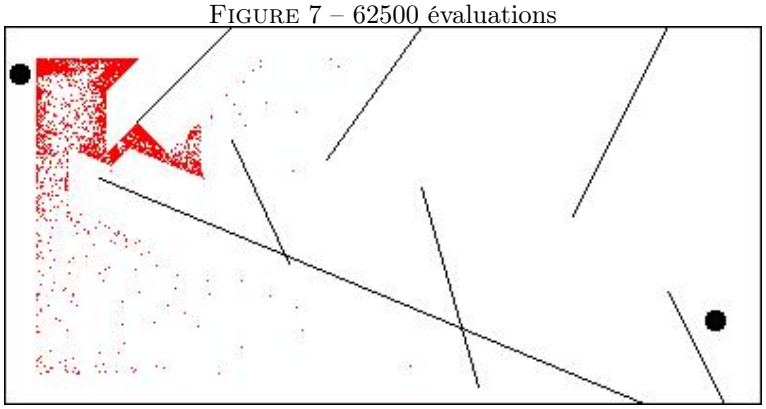


FIGURE 8 – 125000 évaluations

4.1.4 4^{me} exécution

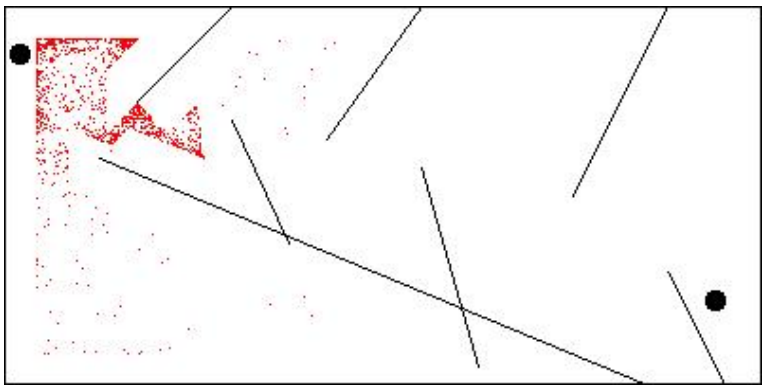


FIGURE 9 – 62500 évaluations

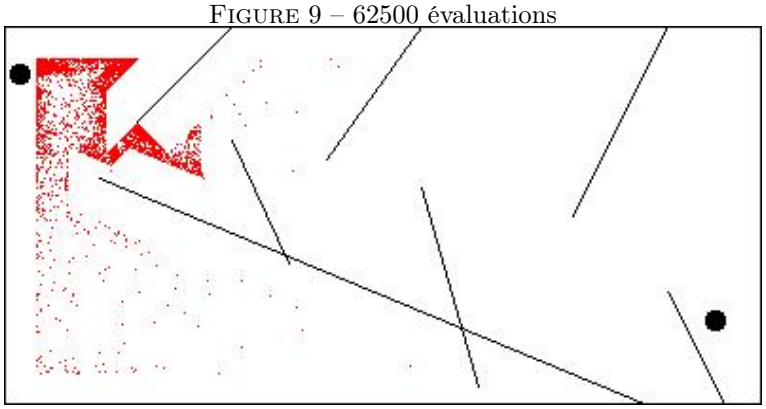


FIGURE 10 – 125000 évaluations

4.1.5 Analyser

Minimisant la distance euclidienne entre la position finale et la position but n'est pas une bonne heuristique a cause de la présence des optima locaux. Ce qui est montré par le fait que la plupart de robots se terminent dans la premier optimum local. On peut même dire que la recherche heuristique n'est pas adaptée pour résoudre ce type de problème, car pour concevoir une fonction heuristique efficace, il faut connaître mathématiquement une formule qui peut nous diriger vers l'optimale global, or cette connaissance est supposée inconnue dans la définition.

4.2 Recherche par nouveauté

4.2.1 1^{re} exécution

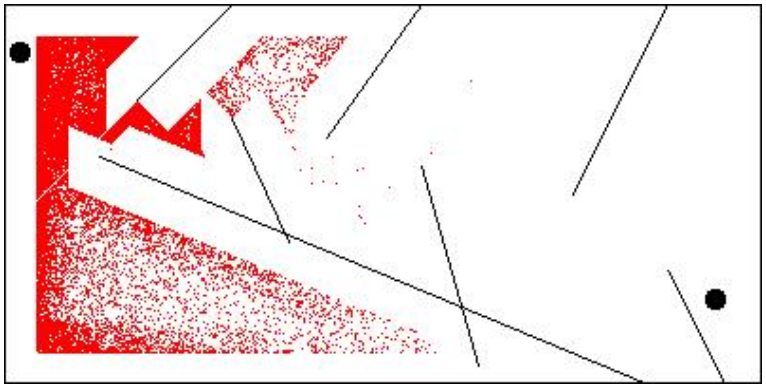


FIGURE 11 – 62500 évaluations

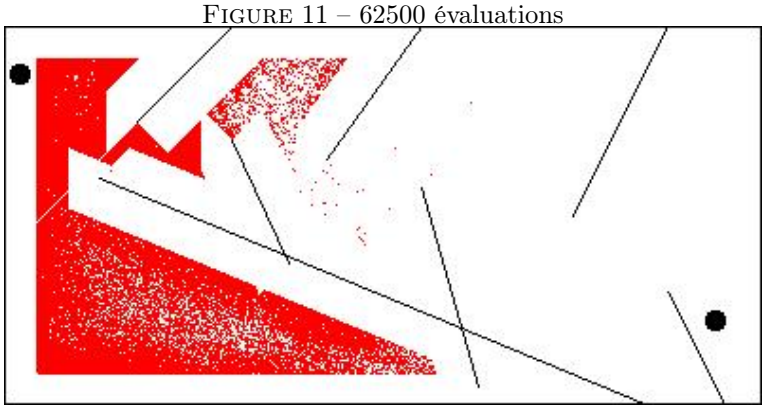


FIGURE 12 – 125000 évaluations

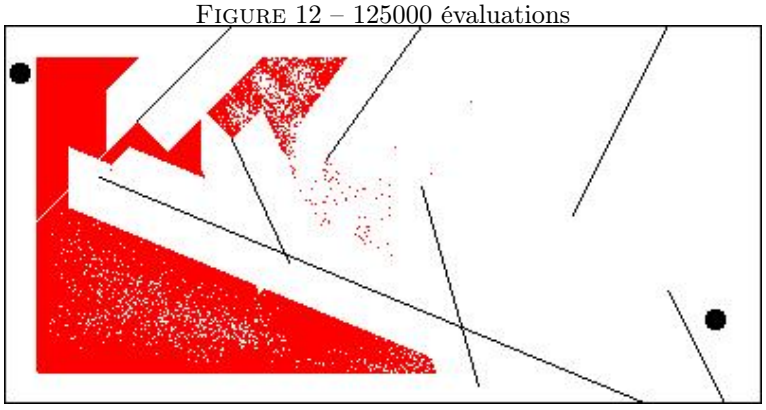


FIGURE 13 – 250000 évaluations

4.2.2 2^{re} exécution

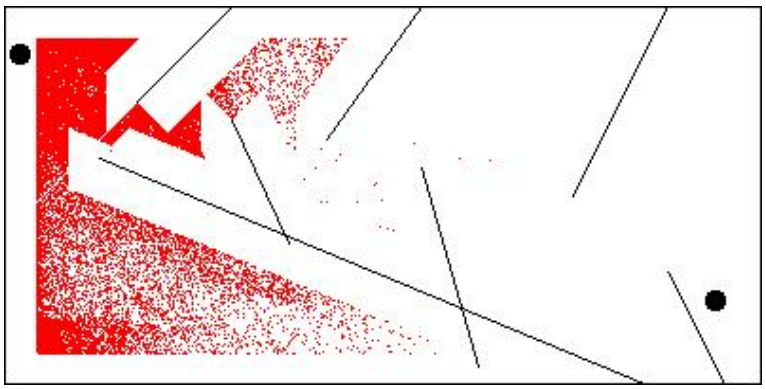


FIGURE 14 – 62500 évaluations

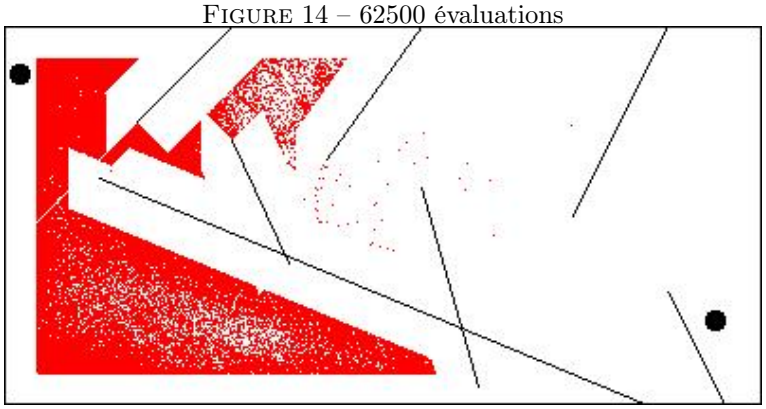


FIGURE 15 – 125000 évaluations

4.2.3 Analyse

En maintenant une variable qui mémorise 1250 positions plus ou moins uniformément distribuées dans la zone visitée, la recherche par nouveauté privilégie les individus, dans la population actuelle, qui s'arrêtent dans les positions pas "très proches" des 1250 positions. C'est pourquoi la recherche est très efficace par rapport a la recherche heuristique. Et que la zone en bas à gauche est bien explorée. Or non seulement les 1250 positions n'est pas optimal, autrement dit ils ne sont pas les meilleures 1250 positions qui permettent de caractériser la densité des zones parmi tous les positions visitées, mais aussi les individus considères sont de la population actuelle. C'est pour la dernière raison que la zone centrale est vraiment peu explorée et que la zone en bas à gauche est privilégiée par rapport a la deuxième zone en haut de gauche à droite. D'où en combinant cet algorithme avec l'algorithme d'illumination, on peut au moins théoriquement améliorer notre recherche.

4.3 recherche par nouveauté+map-élites

4.3.1 1^{re} exécution

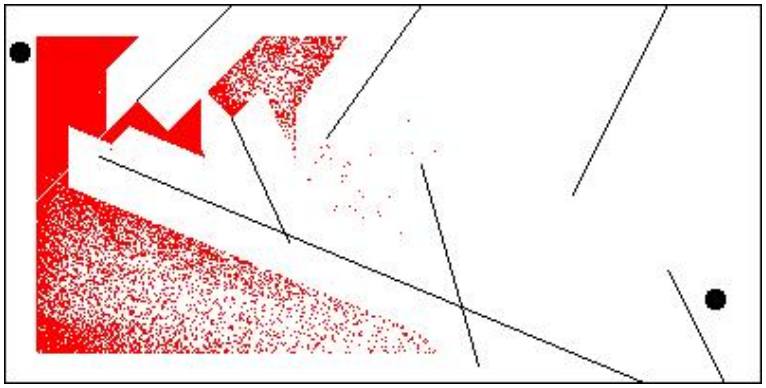


FIGURE 16 – 62500 évaluations

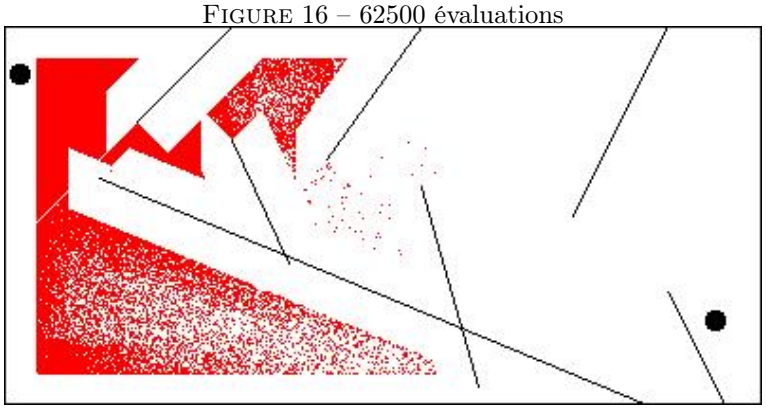


FIGURE 17 – 125000 évaluations

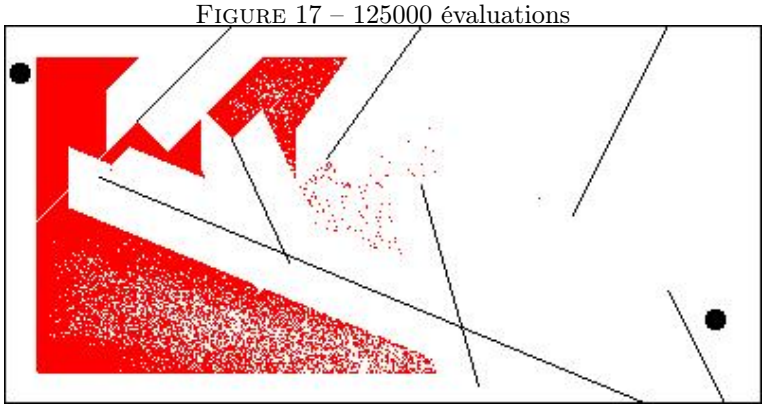


FIGURE 18 – 250000 évaluations

4.3.2 2^{me} exécution

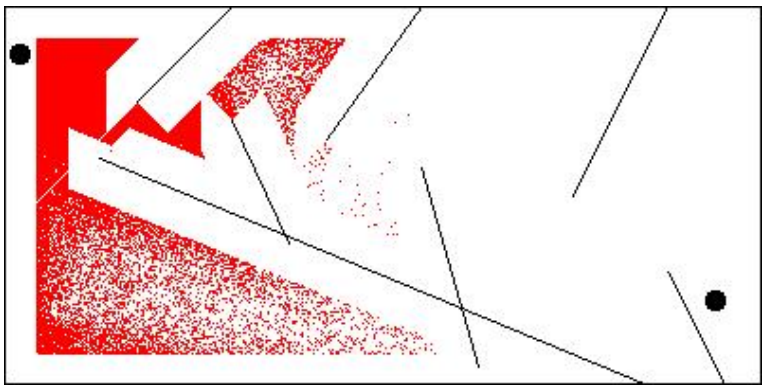


FIGURE 19 – 62500 évaluations

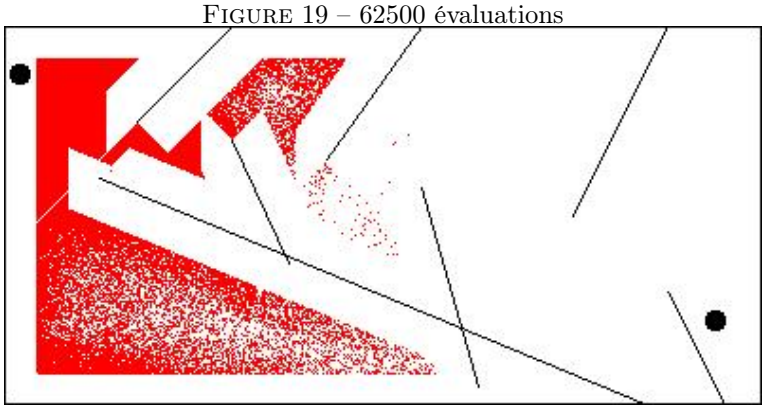


FIGURE 20 – 125000 évaluations

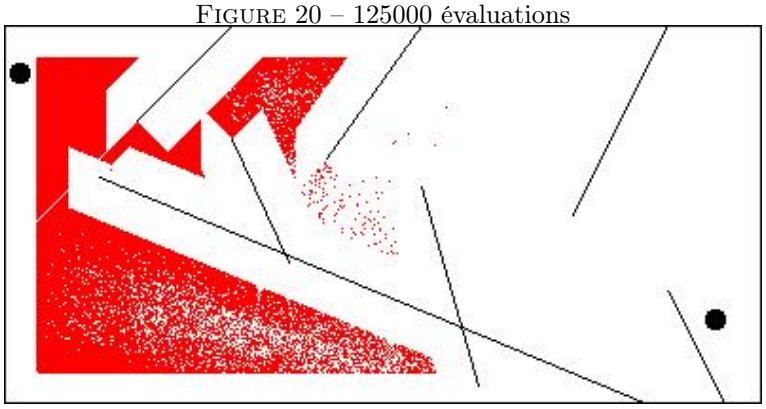


FIGURE 21 – 250000 évaluations

4.3.3 3^{me} exécution

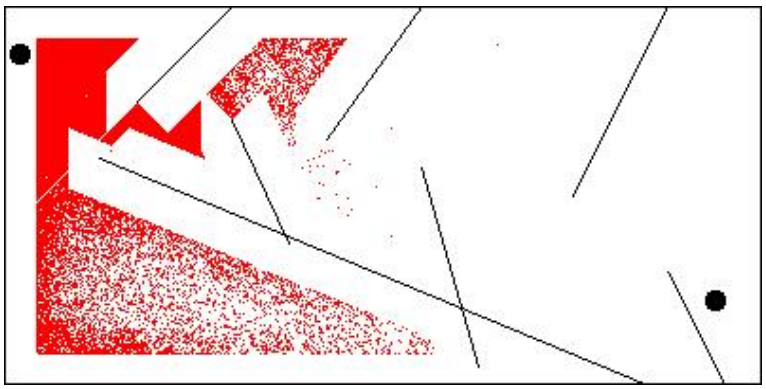


FIGURE 22 – 62500 évaluations

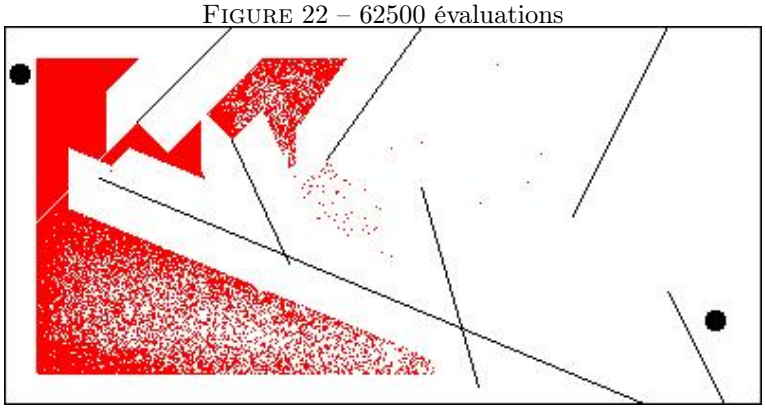


FIGURE 23 – 125000 évaluations

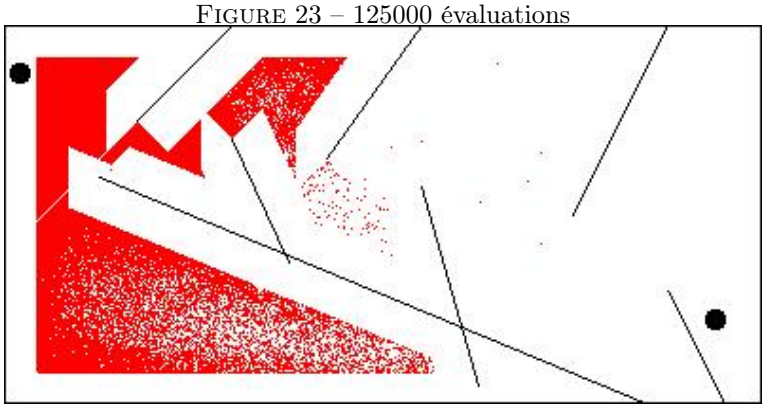


FIGURE 24 – 250000 évaluations

4.3.4 4^{me} exécution

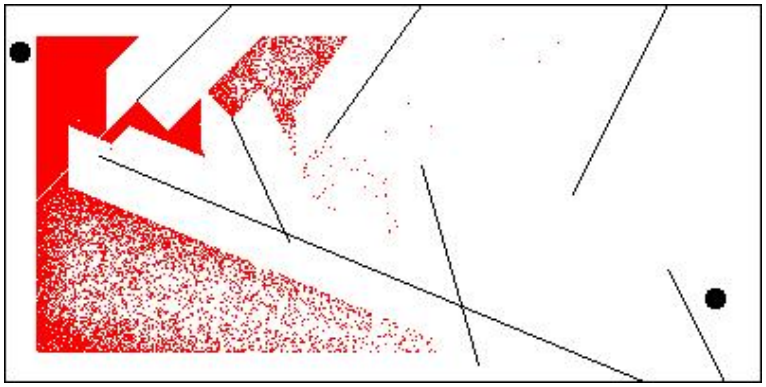


FIGURE 25 – 62500 évaluations

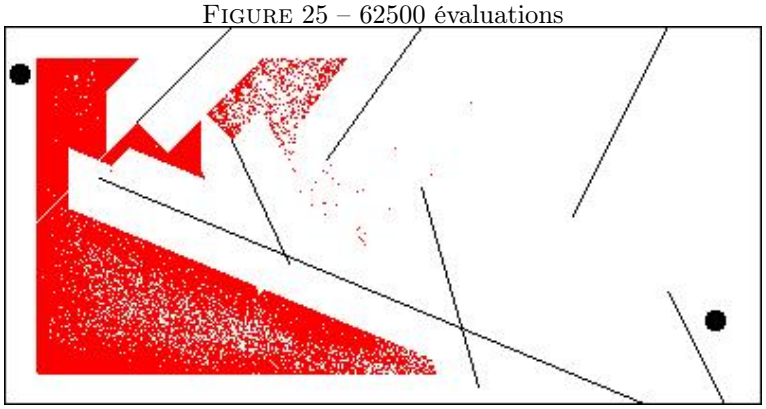


FIGURE 26 – 125000 évaluations

4.3.5 5^{me} exécution

Probabilité de mutation initialement 0.005 augmente de 0.005 quand a la fin d'une génération le nombre de nouvelles positions visitées est inférieur à 10.

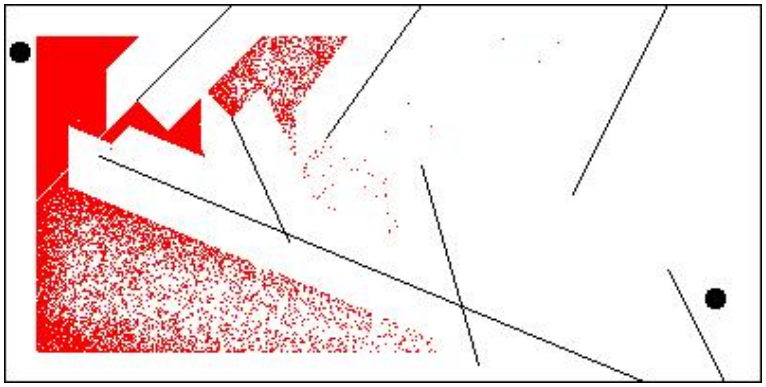


FIGURE 27 – 62500 évaluations

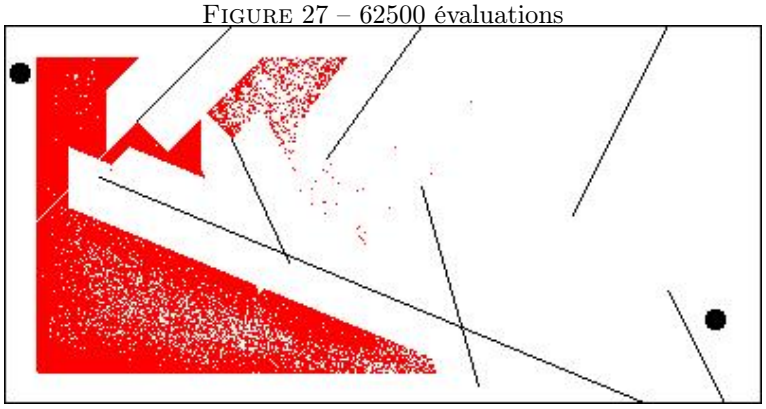


FIGURE 28 – 125000 évaluations

4.3.6 6^{me} exécution

Mutation changer les poids de deux arcs du MLP choisis aléatoirement .

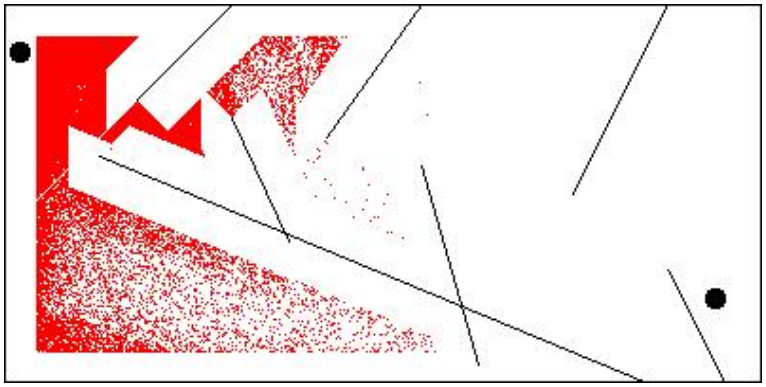


FIGURE 29 – 62500 évaluations

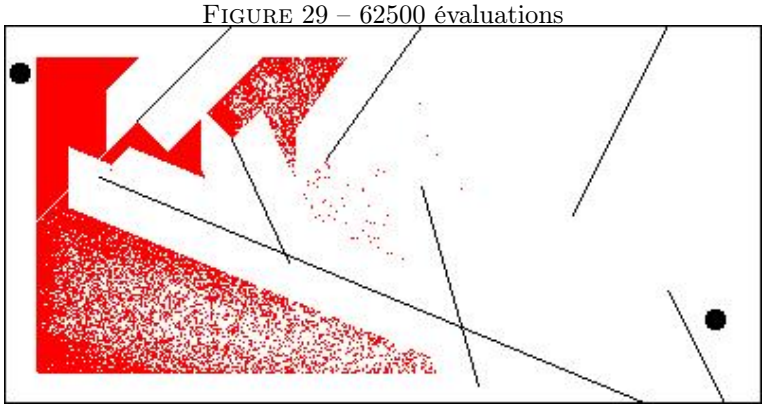


FIGURE 30 – 25000 évaluations

4.3.7 Analyse

Par rapport la recherche par nouveauté, cette méthode tente à explorer la zone centrale à partir de certaines évaluations, car c'est la partie le moins dense, la preuve est que la zone incontournable pour arriver à la zone centrale est très visitée par rapport à la zone en bas à gauche. Cette focalisation plus précise est dut au fait que, pour générer la prochaine génération, on considère non seulement tous les individus de la population actuelle mais tous les individus présents dans l'archive des individus.

5 Conclusion

En conclure, la recherche heuristique n'est pas une bonne algorithme pour black box optimisation car on ne connaît pas quelle est la formule à considérer. La recherche par nouveauté est une bonne approche, mais cette méthode privilégie certaines zones à cause de l'ordre d'exploration ou la facilité d'accès. Et finalement, en combinant la méthode de recherche par nouveauté avec la méthode MAP-élite, on peut améliorer notre recherche.

Références

[1] J.-B. Mouret and J. Clune, "Illuminating search spaces by mapping elites," Apr. 2015. Early-draft.

[2] A. Gaier, A. Asteroth, and J.-B. Mouret, "Data-Efficient Design Exploration through Surrogate-Assisted Illumination," *Evolutionary Computation*, vol. 26, no. 3, pp. 381–410, 2018.

[3] N. Bredeche, J.-M. Montanier, B. Weel, and E. Haasdijk, "Roborobo ! a Fast Robot Simulator for Swarm and Collective Robotics," *arXiv :1304.2888 [cs]*, Apr. 2013.

[4] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, pp. 503–507, May 2015.

[5] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, pp. 100–107, July 1968.

[6] J. Lehman and K. O. Stanley, "Exploiting open-endedness to solve problems through the search for novelty," in *Proceedings of the Eleventh International Conference on Artificial Life (Alife XI)*, MIT Press, 2008.

[7] Pugh Justin K., Soros Lisa B., Stanley Kenneth O. "Quality Diversity : A New Frontier for Evolutionary Computation", Frontiers in Robotics and AI, vol. 3, pp 40, 2016.

[8] C. Brant, Jonathan, O. Stanley, Kenneth. (2017). Minimal criterion coevolution : a new approach to open-ended search. 67-74. 10.1145/3071178.3071186.