Rapport RP

Rayan Perotti-Valle
Felix Savarit

Avril 2024

Contents

1 Introduction **3**

2 Partie 1: Modélisation, instances et résolution par recherche arborescente **3**
2.1 Question 1. 3
2.2 Question 2. 3
2.3 Question 3. 4
2.4 Question 6. 4
2.5 Question 7. 6

3 Partie 2: Résolution par A* **7**
3.1 Question 8. 7
3.2 Question 9. 8
3.3 Question 10. 9

1 Introduction

Ce projet a pour but de tester des méthodes de recherches arborescente pour la résolution du jeu ***Ricochet Robots***. Le principe de ce jeu est de déplacer un robot sur une case cible, normalement on joue avec 5 robots sur un plateau de $n16 \times 16$ cases et 17 cases cibles. Cependant pour ce projet on va simplement s'intéresser à trouver une soltuion pour une disposition donnée ou de déterminer qu'il n'existe pas de solutions de coût inférieur à une valeur fixée. Dans ce problème on a donc à disposition une grille de taille $n \times n$ avec k robots, des murs verticaux et horizontaux et une case "but" disposés aléatoirements sur la grille.

2 Partie 1: Modélisation, instances et résolution par recherche arborescente

2.1 Question 1.

Nous avons représenté ce problème de la manière suivante:

- *états initiaux:*

  $R_1 = (x_1, y_1)$ **position du robot 1**
  ...
  $R_k = (x_k, y_k)$ **position du robot k**
  $C = (x_i, y_i)$ **position de la cible**

  avec $(x_1, ..., x_k, y_1, ..., y_k, x_i, y_i) \in [0, n]$

- *états final:*

  $R_1 = (x_i, y_i) = C$

- *Transition possible:*

  1. Déplacement du robot R_1 vers le haut de a cases $(x_1, y_1 - a)$
  2. Déplacement du robot R_1 vers le bas de a cases $(x_1, y_1 + a)$
  3. Déplacement du robot R_1 vers la gauche de a cases $(x_1 - a, y_1)$
  4. Déplacement du robot R_1 vers la droite de a cases $(x_1 + a, y_1)$

2.2 Question 2.

- Pour un plateau de taille $n \times n$ avec un seul robot on a n^2 solutions
- Pour un plateau de taille $n \times n$ avec deux robots on a $n^2 \times (n-1)^2$ solutions
- Donc pour un plateau de taille $n \times n$ avec k robot on a $\prod_{i=0}^{k-1}(n^2 - i)$

2.3 Question 3.

Un robot R_1 a 4 successeurs possibles; ***haut, bas, gauche, droite*** il peut y avoir k robots en simultanés sur la grille et chacun de ces robots peut être déplacé pour trouver une meilleur solution pour le robot R_1, cependant on déplace un seul robot à la fois donc il y a $4 \times k$ successeurs à chaque étape.

2.4 Question 6.

Nous avons choisi une instance afin d'illustrer la soltuion optimale obtenue avec la recherche en profondeur avec un seul robot qui se déplace et celle obtenue avec la recherche en profondeur avec tous les robots pris en compte.

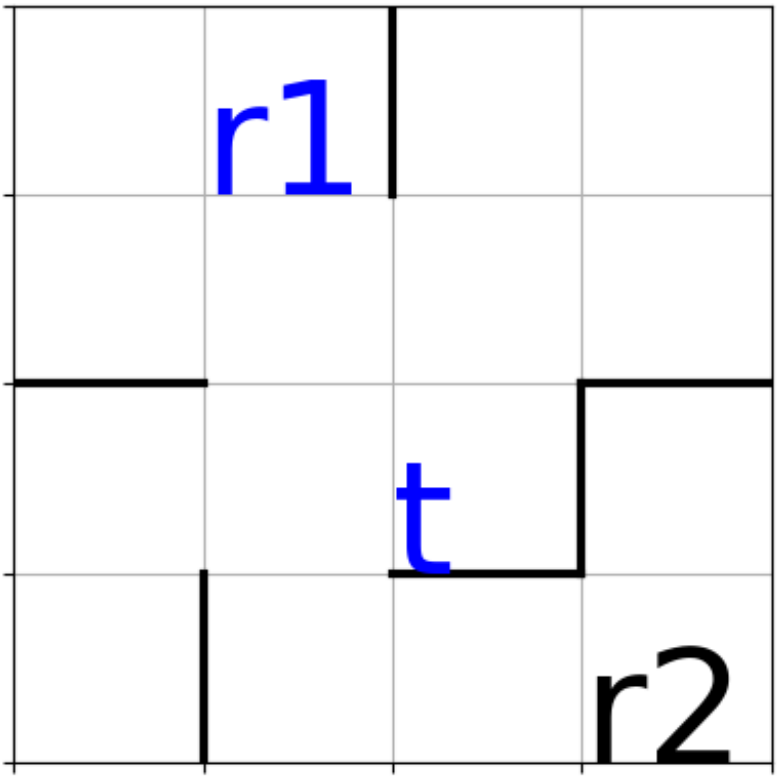

Figure 1: Instance Choisi

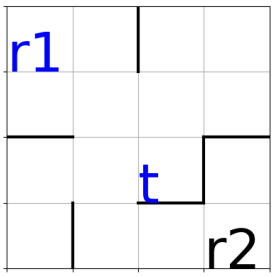

Figure 2: 1^{er} déplacement

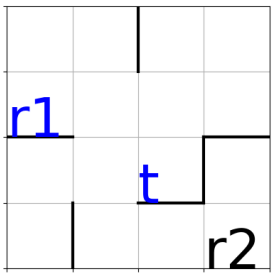

Figure 3: 2^{nd} déplacement

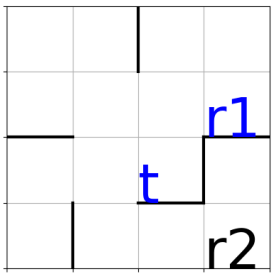

Figure 4: 3^{eme} déplacement

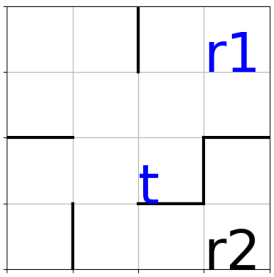

Figure 5: 4^{eme} déplacement

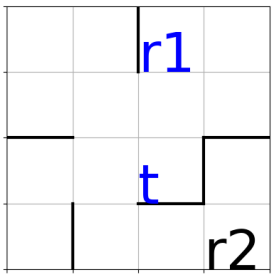

Figure 6: 5^{eme} déplacement

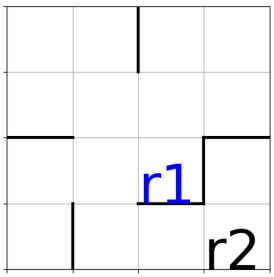

Figure 7: 6^{eme} déplacement

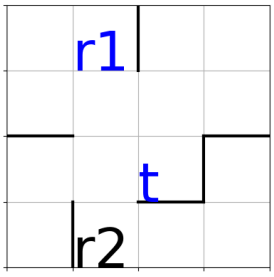

Figure 8: 1^{er} déplacement

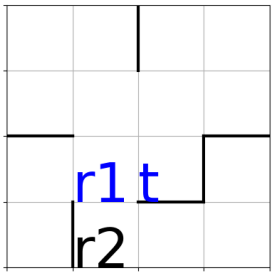

Figure 9: 2^{nd} déplacement

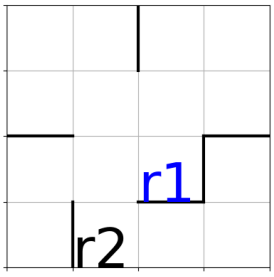

Figure 10: 3^{eme} déplacement

Les figures 1 à 7 représente la solution optimale obtenue avec la recherche en profondeur de un seul robot. Les figures 8 à 10 représente la solution optimale obtenue avec la recherche en profondeur avec tous les robots.

2.5 Question 7.

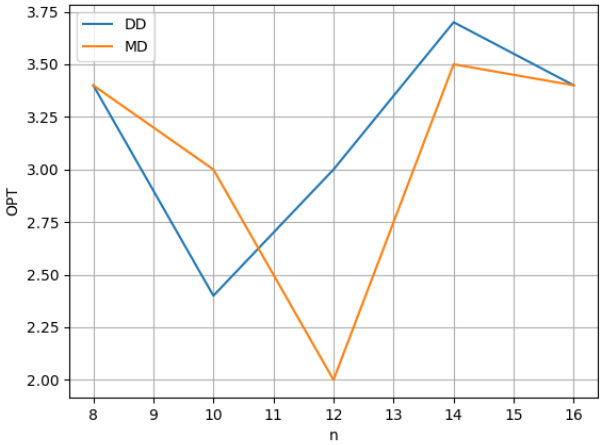

Figure 11: n en fonction de OPT

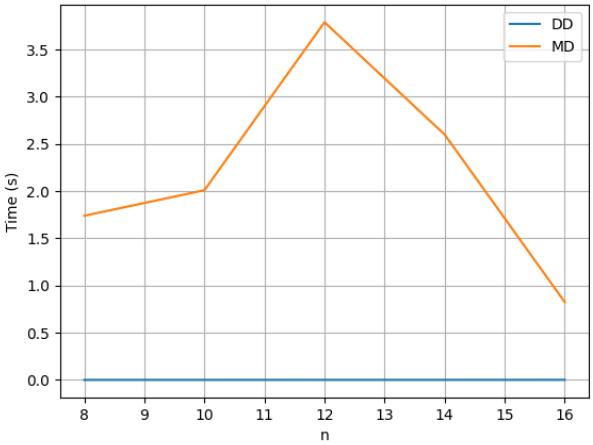

Figure 12: n en fonction du temps

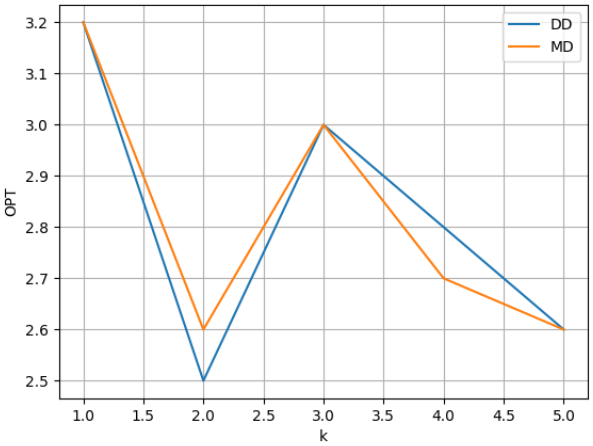

Figure 13: k en fonction de OPT

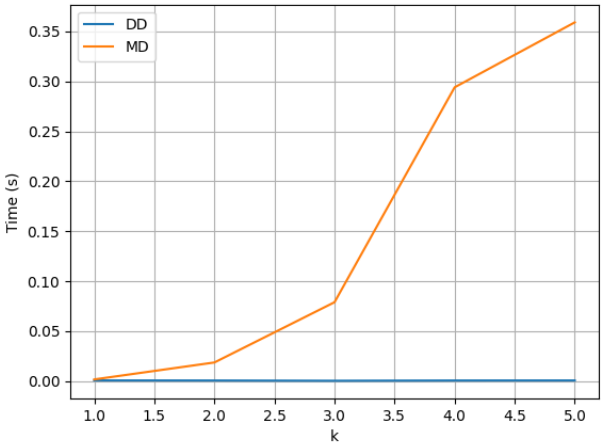

Figure 14: k en fonction du temps

Nous avons généré 20 instances de grille en faisant varier dans un premier temps la taille n de la grille avec $n \in [8, 16]$ par pas de 2 et dans un second temps le nombre k de robots avec $k \in [1, 5]$, nous avons ces instances sur les deux fonctions que nous avons réalisé plus tôt à savoir la fonction de recherches en profondeur avec un seul robot et la fonction de recherche en profondeur avec tous les robots et nous avons comparé les résultats avec les graphes ci-dessus.

La figure 11 et 12 représente les résultats obtenues avec la variation de n, on remarque que le temps d'exécution reste plus ou moins le même avec la fonction de recherche en profondeur avec un seul robot(courbe bleu), le temps d'exécution de la fonction de recherche en profondeur avec tous les robots(courbe orange) quant à elle varie selon n. Pour la figure, on remarque des résultats plus variés.

La figure 13 et 14 représente les résultats obtenues avec la variation de k, sur la figure 13 on peut observer que les résultats sont similaires entre les deux fonctions de recherches en profondeur mais que cependant la fonction de recherche en profondeur avec tous les robots prend beaucoup plus de temps que l'autre fonction de recherche et que le temps de celle-ci augmente avec le nombre de robots sur la grille.

3 Partie 2: Résolution par A*

3.1 Question 8.

Dans cette partie, on va mettre en place une procédure A* avec deux heuristiques différentes. L'heuristique h_1 est la suivante si le robot et la cible se trouvent sur la même ligne ou colonne $h_1 = 1$ sinon $h_1 = 2$ et si le robot se trouve sur la case cible alors $h_1 = 0$.

L'heuristique h_1 est une heuristique coïncidente car il y a un seul état but qui est la case cible et l'évaluation de l'heuristique de celle-ci est à 0.

L'heuristique h_1 est une heuristique monotone car dans tous les cas on a $h(i) - h(j) \leq k(i, j)$, dans cette heuristique les cases sont évaluées à 0, 1 ou 2 et un déplacement entre deux cases de la grille est de coût 1 donc $\forall i, j \in [0, n], k(i, j) = 1$ cependant un déplacement d'une case évaluée a 2 vers la case cible est impossible donc les déplacements possibles sont:

- une case évaluée à 2 vers une autre case évaluée à 2: $h(i) = 2, h(j) = 2, k(i, j) = 1 \Rightarrow 2-2 \leq 1$
- une case évaluée à 2 vers une case évaluée à 1: $h(i) = 2, h(j) = 1, k(i, j) = 1 \Rightarrow 2 - 1 \leq 1$
- une case évaluée à 1 vers une case évaluée à 2: $h(i) = 1, h(j) = 2, k(i, j) = 1 \Rightarrow 1 - 2 \leq 1$
- une case évaluée à 1 vers une case évaluée à 1: $h(i) = 1, h(j) = 1, k(i, j) = 1 \Rightarrow 1 - 1 \leq 1$
- une case évaluée à 1 vers une case évaluée à 0: $h(i) = 2, h(j) = 1, k(i, j) = 1 \Rightarrow 1 - 0 \leq 1$

donc l'heuristique h_1 est bien monotone.

L'heuristique h_1 est à la fois monotone et coïncidente donc l'heuristique h_1 est aussi minorante.

3.2 Question 9.

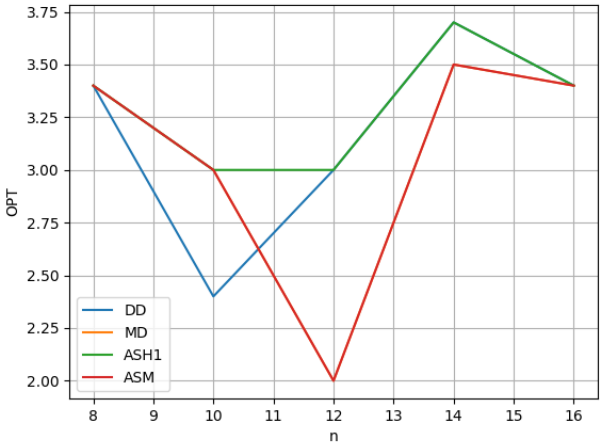

Figure 15: n en fonction de OPT

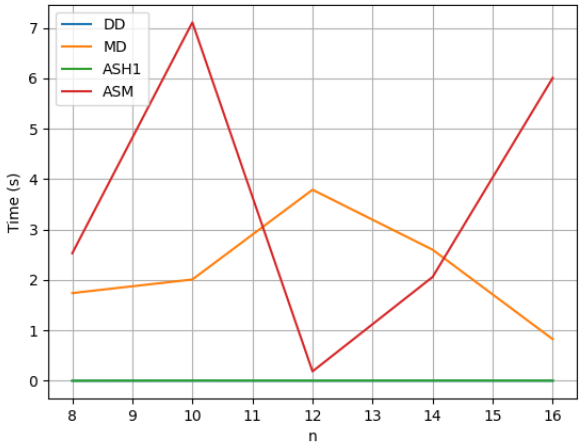

Figure 16: n en fonction du temps

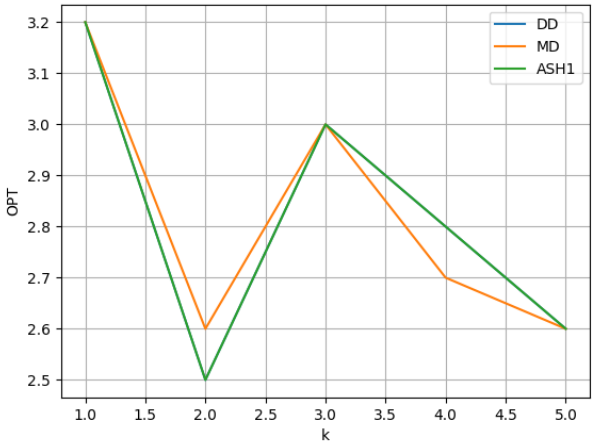

Figure 17: k en fonction de OPT

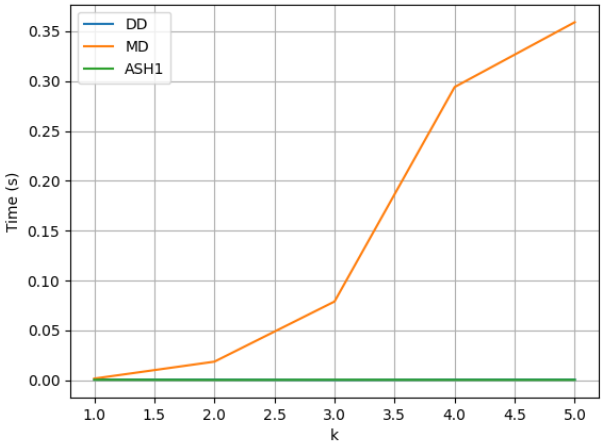

Figure 18: k en fonction du temps

Comme pour la question 7, nous avons généré 20 instances pour chaque variation de n et de k et nous avons exécuté les fonctions de recherche en profondeur ainsi que les deux version de la fonction A* avec l'heuristique h_1 que nous avons réalisé et nous avons comparé les résultats.

Pour les figure 17 et 18 on remarque aucuns changements la fonctionde recherche en profondeur avec tous les robots prend toujours de temps que les autres fonctions.

Cependant, pour les figures 15 et 16 on observe quelques différences, on voit que notre fonction A* avec tous les robots prend beaucoup plus de temps que la fonction A* avec un seul robot et que

en général les fonctions impliquant tous les robots prennent plus de temps d'exécution.

3.3 Question 10.

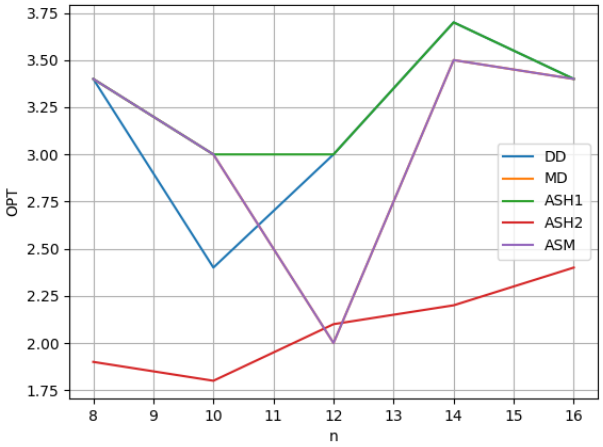

Figure 19: n en fonction de OPT

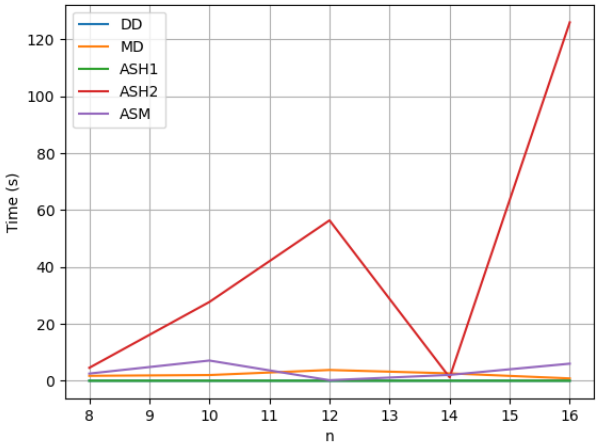

Figure 20: n en fonction du temps(s)

Enfin nous avons ajouté la fonction A* avec l'heuristique h_2 à nos analyses. On remarque que l'heuristique h_2 prend beaucoup plus de temps à s'exécuter que toutes les autres fonctions mais on peut aussi observer que la fonction A* trouve des meilleurs solutions que les autres fonctions. Ces résultats très différent sont surement dus au fait qu'on relache la contrainte de déplacement et qu'on ne prend pas en compte tous les robots dans l'heurstique h_2.