MOGPL

# LA BALADE DU ROBOT

Morane Gruenpeter,Arthur Ramolet
Université Pierre et Marie Curie
Année Universitaire 2015-2016
LaTeX
2 décembre 2015

# TABLE DES MATIÈRES

# 1 INTRODUCTION

Un robot est utilisé dans le dépôt d'un grand magasin pour le transport d'objets. On s'intéresse à la minimisation du temps de transport du robot.

L'objectif de ce projet est à partir d'une instance et d'un cas d'exemple de produire une modélisation du problème permettant de trouver le chemin qui minimise le temps de transport du robot de sa position de départ jusqu'à son objectif.

La première partie explique comment le problème a été modélisé. La seconde partie présente l'algorithme utilisé, ainsi qu'une note sur sa complexité. La troisième partie montre les résultats obtenu avec notre algorithme. La quatrième partie est dédiée à l'interface graphique et son utilisation.

## 2 FORMULATION DU PROBLÈME

La matrice qui représente le dépôt du grand magasin est transformé en un graphe orienté tel que G=S,A ou S représente les sommets accessible au robot et A les arcs par lesquels il peut passer. Les coordonnées ligne-colonne sont transformées en 4 sommets dénommés Nord, Est, Sud, Ouest. Depuis Le sommet Nord,le robot peut avancer vers le haut, s'il a besoin de changer de direction il passera d'abord par un sommet qui lui permet l'accès à cette direction. Par conséquence, le sommet Sud permet au robot d'avancer vers le bas, le sommet Est permet au robot d'avancer à droite et le sommet Ouest vers la gauche.

La mise en place des arcs se fait entre les 4 sommets qui représentent le même point (x,y) de la matrice et entre le point adjacent de même direction.De plus, le robot peut avancer soit d'un mètre, soit de deux mètres soit de trois mètre. L'ajout des arcs qui représentent les trois distances possible, permet de mettre en place le déplacement multiple. Par exemple, le sommet (4,4, Nord) qui se trouve à la ligne 4, à la colonne 4 et au Nord, est lié par un arc vers les sommet suivant :

1. changement de direction : (4,4, Sud),(4,4,Est),(4,4,Ouest)
2. avancement d'un mètre (3,4,Nord)
3. avancement de deux mètre (2,4,Nord)
4. avancement de trois mètre (1,4,Nord)

Le point de départ du robot est configuré par le fichier entré ou par le choix de l'utilisateur, celui-ci inclu sa direction, ce qui conduit à l'associé directement avec un sommet du graphe. Par-contre le point de sortie n'exige pas une direction spécifique au robot, ce qui permet au point d'arrivée d'être un des quatre sommets appartenant au point (x,y).

Les obstacles dans la matrice sont des sommets inaccessible, ils apparaissent dans le graphe afin de mieux visualiser le problème, mais ce sont des sommets fantômes.

Tous les arcs qui sont posés dans G sont de poids 1 et représentent un tic d'horloge dans le temps. L'objectif est de déterminer le temps minimum pour le déplacement du robot d'un point de départ à un point d'arrivé. Cet objectif revient à calculer le plus court chemin entre ces deux points.

# 3 L'ALGORITHME DE RÉSOLUTION

L'algorithme utilisé afin de résoudre ce problème est un parcours en largeur d'abord (appelé aussi BFS- Breadth-First-Search). Cet algorithme permet de parcourir le graphe niveau par niveau à partir d'un sommet de départ, puis d'interpréter ce parcours en plus court chemin entre le sommet de départ et le sommet d'arrivée. L'utilisation de cet algorithme est possible et efficace seulement parce que le graphe modélisé est un graphe qui contient seulement des arcs à poids 1.

## Structures de données utilisées

1. on utilise une file à laquelle on ajoute les sommets à traiter à la fin de la file.
2. on utilise deux hashmaps, parent et distance, afin de stocker les informations distance et parent du sommet parcouru.

## Déroulement de l'algorithme

1. Initialisation du sommet d'entrée à 0 et de tous les autres à -1 (comme valeur infinie).
2. Ajout du sommet de départ à la file.
3. Tant que la file n'est pas vide, on ajoute les sommets visités et on ajoute les valeurs des hashmaps.
4. Afin de retrouver le chemin le plus court on prend depuis la hashmap le sommet d'arrivée et on cherche le parent jusqu'au sommet de départ.

## Complexité

La complexité du parcours en largeur d'abord est en O(n+p) avec n sommets et p arcs, accessible depuis le point de départ, donc n ne compte pas les sommets qui représente les obstacles. Chaque sommet accessible depuis le sommet de départ est visité et enfilé une seule fois. Puis en enlevant ce sommet de la file on utilise une seule fois chaque arcs afin d'atteindre les sommets accessible.

Le nombre de nœuds correspond à la taille de la matrice +1 le tout x4 (pour la transformation gérant le sens du robot) donc n*m, soit (n+1)x(m+1)x4. Le nombre d'arcs correspond au nombre de noeuds x2 (Possibilité d'aller à gauche ou a droite) x3 (Possibilté d'avancer de 1 2 ou 3), soit (n+1)*(m+1)x4x2x3

La complexité totale de l'algorithme est donc en O($(n+1)x(m+1)x4 + (n+1)x(m+1)x4x2x3$)) -> O($n^2$) en enlevant les facteurs constants.

# 4 RÉSULTAT ET ANALYSE

L'implémentation des essais numériques permet d'évaluer le temps de calcul de la génération du graphe et du parcours en largeur en fonction de la taille de la grille et en fonction du numéro d'obstacles posé sur cette grille.

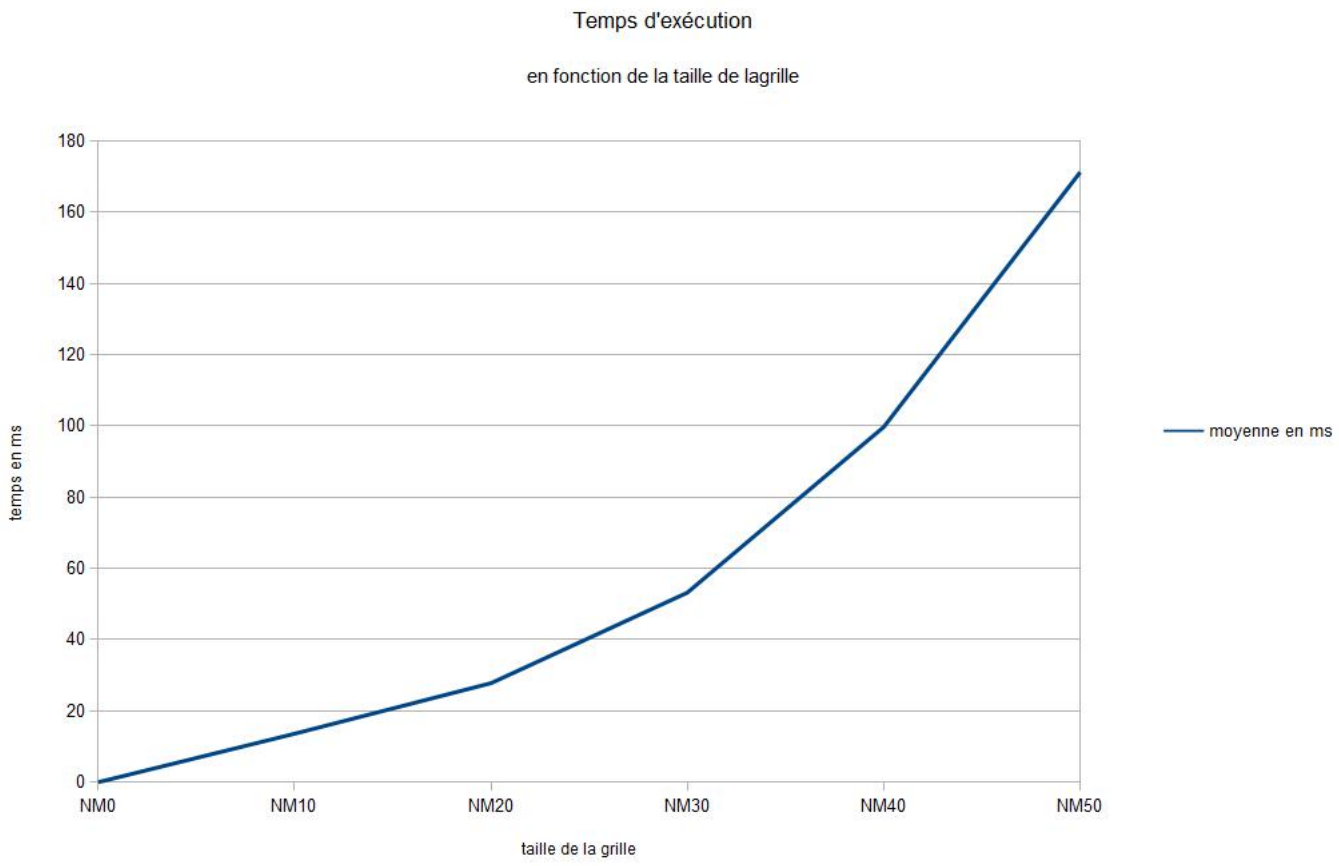


FIGURE 1: Temps d'exécutions en fonction de la taille de la grille.

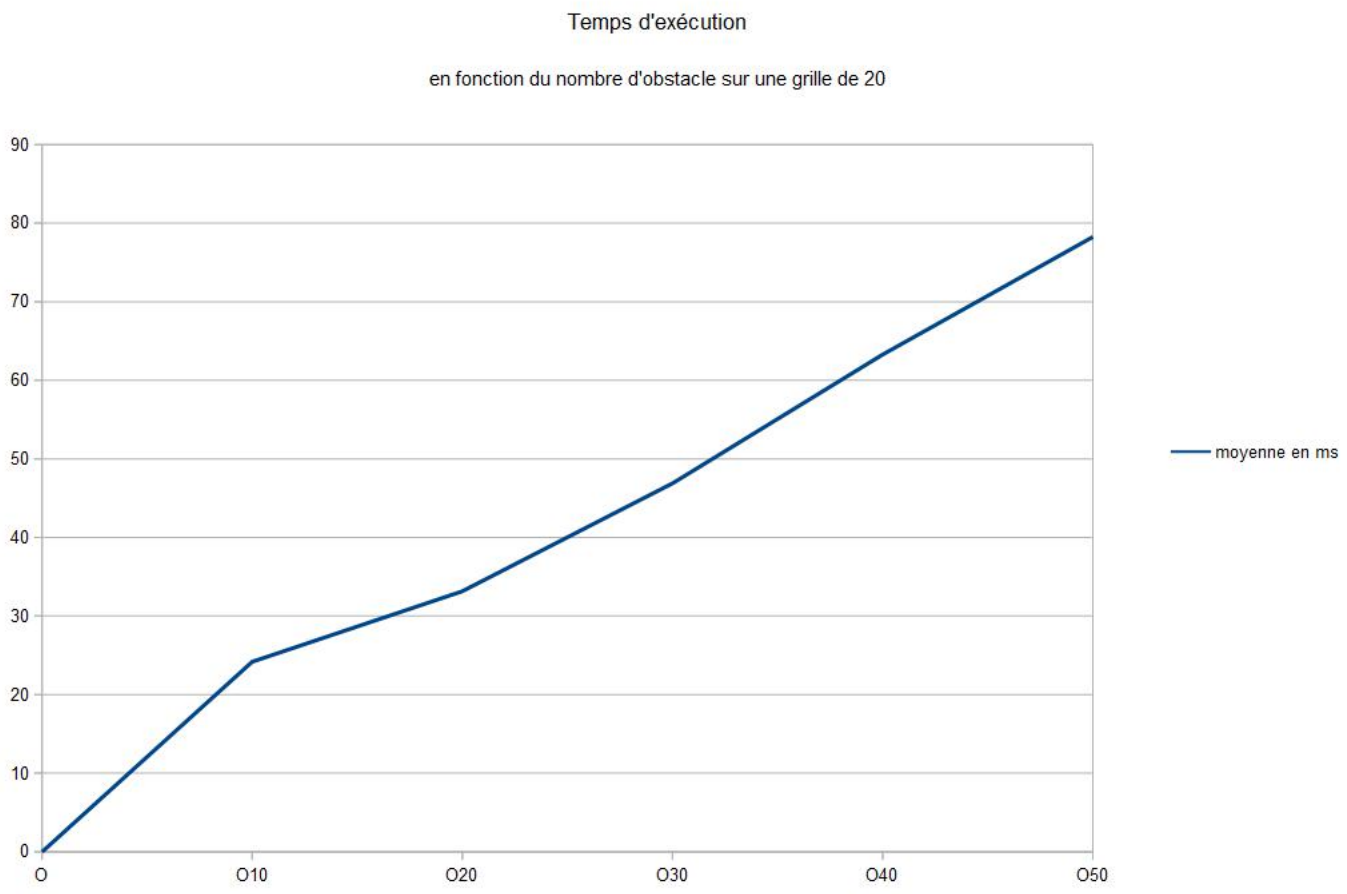


FIGURE 2: Temps d'exécution en fonction du nombre d'obstacles.

| taille de la grille | nombre obstacle | moyenne en ms |
|---|---|---|
| 10 | 10 | 13,6 |
| 20 | 20 | 33,16 |
| 20 | 10 | 24,18 |
| 20 | 30 | 46,89 |
| 20 | 40 | 63,28 |
| 20 | 50 | 78,25 |
| 30 | 30 | 53,2 |
| 40 | 40 | 99,7 |
| 50 | 50 | 171,2 |
| 50 | 20 | 66,1 |

FIGURE 3: Temps d'exécutions

Ci dessus les temps moyens d'exécution sur deux graphiques séparés, le premier (Fig : 1) présente le temps de calcul en fonction de la taille de la grille avec NM10,NM20, NM30,NM40, puis le second (Fig : 2) présente le temps de calcul en fonction des obstacles avec O10, O20, O30, O40 et O50. Par ailleurs, un tableau qui récapitule toutes les moyennes est présenté également.

Il est possible de conclure qu'un nombre d'obstacle plus élevé est plus pénalisant qu'une grille plus grande, car avec 50 obstacles sur une grille de 20 sur 20 le temps moyen d'exécution est à 78,25. En revanche, sur une grille de 50 sur 50 avec 20 obstacles le temps moyen d'exécution est de 66,1.

# 5 INTERFACE GRAPHIQUEE

L'interface a été pensée pour être simple à utiliser (Fig : 4).

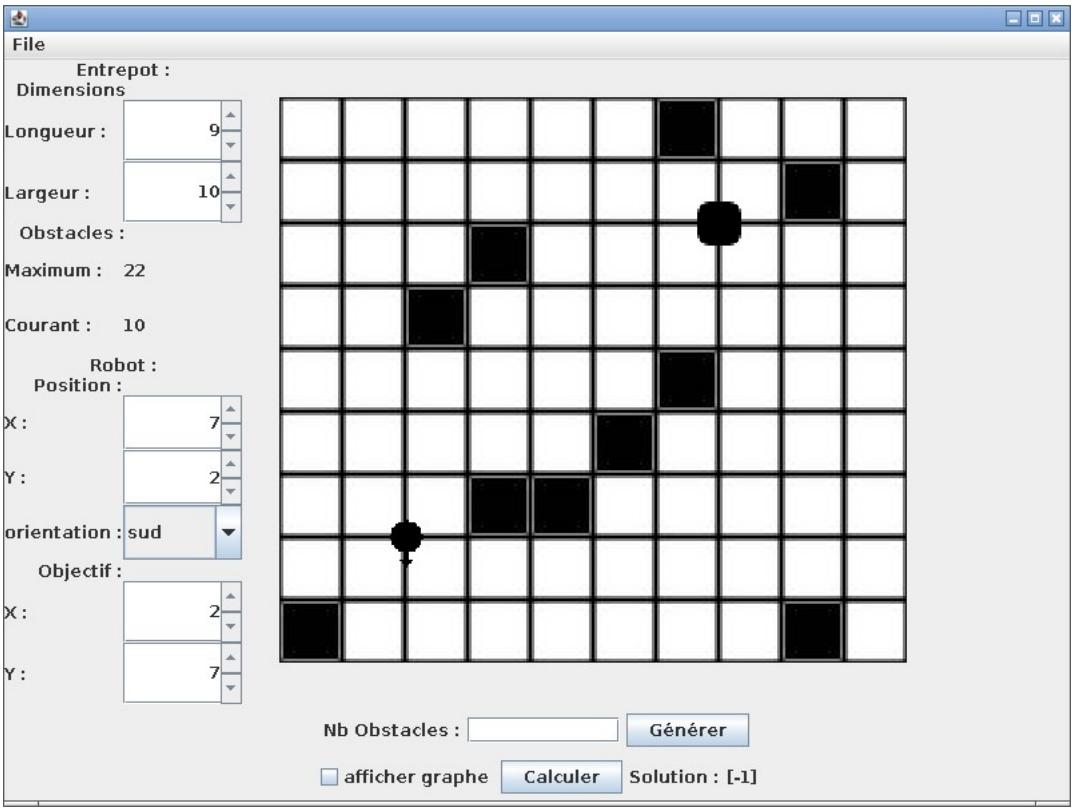


FIGURE 4: Interface Graphique

On règle la taille de l'entrepôt à l'aide des 2 premiers curseurs. Les 2 curseurs suivants servent à régler la position du robot. Le menu déroulant permet de choisir l'orientation du robot. Les deux curseurs suivants servent à régler l'objectif du robot.

Le champ générer permet de placer des obstacles aléatoirement, on saisit le nombre d'obstacles juste a coté. Calculer permet de tracer le chemin. Si la case afficher graphe est cochée, le graphe crée pour le calcul de chemin est affiché.

Le menu file permet de charger un fichier d'instance ou de quitter l'application. La fonction sauvegarder n'est pas implémentée.

On peut placer ou retirer de nouveaux obstacles en cliquant sur les cases.

L'interface est réactive aux redimensionnements.

# 6 Conclusion

La modélisation et le parcours d'un graphe dans un contexte industriel est une manière d'appliquer une démarche de résolution sous des contraintes concrètes. En formalisant ce problème avec un graphe, nous avons vu que le choix d'une modélisation avec des arcs de poids un nous permet d'alléger la complexité de l'algorithme du parcours en utilisant un parcours en largeur d'abord (BFS). L'algorithme du parcours en largeur est un algorithme fiable et performant, avec une complexité moindre que les algorithmes de plus court chemin comme Djikstra ou Bellman. Ainsi, les tests d'exécutions en fonction de la taille de la grille et en fonction du nombre d'obstacles, décrivent l'efficacité de la construction du graphe et du parcours du graphe. Finalement, l'interface graphique affiche à un utilisateur de manière ludique le résultat du problème et le résultat de la démarche de modélisation.