**M**aster **1** **Informatique**

**Promotion 2018 – 2019**

**IDEFIX**

Julien DEVILLARS – Florent DUPONT – Adrien FORET

Yannis HUTT – Etienne LUQUIN – Gabriel VINADO BUIL

Dorian WOUTERS – Mathieu ZENNER

Projet Transversal Innovant

Université Claude Bernard – Lyon 1



# **Contexte du projet :**

Le Projet Transversal Innovant de cette année a pour but de développer en équipe un réseau de 3 robots se déplaçant dans un labyrinthe. Ces robots sont construits autour des briques ev3 de la gamme Lego Mindstorm. Ces dernières permettent de contrôler des moteurs et de connecter différents capteurs pour en en récupérer les données. Le labyrinthe quant à lui est composé de la manière suivante :

* C’est une grille de 8\*8, il y a donc 64 croisements.
* Les robots peuvent tourner aux croisements et se déplacent en suivant des lignes de rubans colorés.
* Sur chaque ligne il y a une combinaison de rubans de couleurs (noir/rouge pour les lignes horizontales et blanc/noir pour les lignes verticales).
* Des murs peuvent être placés aléatoirement avant l’exploration du labyrinthe coupant ainsi certaines routes.

Les robots ont pour tâche finale de retrouver et de s’arrêter sur un croisement qui leur est attribué. La résolution de cet objectif se déroule en plusieurs parties qui seront expliquées dans la suite de ce rapport : la phase d’exploration et la phase de résolution.

**Construction :**

Dans un premier temps nous avons cherché à construire un modèle de robot pouvant répondre aux contraintes données, et prenant compte que certaines pièces sont en quantités limitées et l’esthétique globale fait partie intégrante de l’évaluation finale.

La contrainte la plus évidente est le déplacement, nous avons choisi d’implémenter, à chaque côté du robot, un ensemble de trois roues avec une chenille tendue sur celles-ci, l'entraînement du barbotin (la roue principale) est réalisé par un moteur raccordé à une des sorties de la brique ev3. Le choix d’une chenille permet d’assurer la transmission du poids du robot au sol en la répartissant sur une surface supérieure à celles des roues.

Le robot doit pouvoir suivre les lignes au sol pour rester sur le chemin. Nous avons donc installé un capteur de couleur pointant vers le bas à l’avant du robot.

Il faut aussi que la machine soit capable de détecter les murs en face d’elle et qu’elle stoppe son déplacement en conséquence. Ainsi, nous avons placé sur le capteur couleur un capteur ultrason orienté vers l’avant.

Enfin il faut limiter le poids total du robot. Un robot lourd consomme plus d’énergie et peut fausser les calculs et données liés aux moteurs.

Ayant accès à encore deux entrées pour des capteurs sur la brique ev3, nous avons ajouté au robot alpha (nommé BLUE), chargé de faire la phase d’exploration, deux détecteurs d’ultrasons. Ils sont placés de part et d’autre du robot, et sont utilisés lorsque le robot s’arrête devant un mur. Il vérifie les murs autour de lui avec les deux nouveaux capteurs, empêchant ainsi le robot de tourner inutilement vers un mur.

Les robots n’ont pas évolué dans le design global lors du projet seul des éléments de renforcement ont été ajouté.

**Contrôle du robot : déplacements, capteurs**

Le contrôle bas niveau du robot est fait grâce á la brique ev3, qui est programmée en Python3. Afin de maintenir un déplacement en ligne droite, le robot se déplace en captant en même temps les couleurs du ruban au sol. Le robot vérifie qu’il capte le milieu des deux couleurs du rubans, et corrige son déplacement lorsque le capteur couleur reste trop longtemps dans la même couleur.

**Phase d’exploration (a modifier en conséquences des résultats)**

La phase d’exploration se déroule de la façon suivante : un seul robot est placé dans un coin du labyrinthe. Il se déplace librement et découvre l’environnement pendant une durée maximale de 5 minutes.

Pour ceci le robot va scanner les cases autour de lui, une fois qu'il connaît les cases à découvrir il s’y rend puis scan à nouveau jusqu’à ne plus avoir de cases à découvrir. Le robot se déplace de case en case grâce à un algorithme A\* dans ce qu’il a découvert.

**Synchronisation et Phase de résolution (modifiable aussi)**

Pour donner suite à la phase d’exploration, la grille découverte par le robot « explorateur » est communiquée et retranscrite aux autres robots. Les positions sur lesquelles nos machines doivent s’arrêter nous alors distribuées, et de nouvelles contraintes s’appliquent. Un seul robot peut être en mouvement à un instant t. Ce dernier ne peut se déplacer qu’ne ligne droite, et s’arrête lorsqu’il rencontre un mur ou un autre robot. Il peut tourner seulement s’il a rencontré un obstacle.

Pour résoudre cette tâche nous nous sommes inspirés du jeu de plateau « Ricochet Robot » dont le but est exactement le même : déplacer ses pions en ligne droite pur créer de nouveaux obstacles et atteindre certaines cases sur un plateau labyrinthique.

Pour donner suite à la phase d’exploration, la grille découverte par le robot « explorateur » est communiquée et retranscrite aux autres robots. Les positions sur lesquelles nos machines doivent s’arrêter nous alors distribuées, et de nouvelles contraintes s’appliquent. Un seul robot peut être en mouvement à un instant t. Ce dernier ne peut se déplacer qu’une ligne droite, et s’arrête lorsqu’il rencontre un mur ou un autre robot. Il peut tourner seulement s’il a rencontré un obstacle.

Pour résoudre cette tâche nous nous sommes inspirés du jeu de plateau « Ricochet Robot » dont le but est exactement le même : déplacer ses pions en ligne droite pour créer de nouveaux obstacles et atteindre certaines cases sur un plateau labyrinthique.

L’algorithme réalise une recherche en profondeur, c’est-à-dire que pour chaque branche on réalise un mouvement possible depuis le nœud mère de la branche jusqu'à trouver une solution. Ce type d’algo permet de trouver une solution optimale.  
  
Lors de déroulement de l’algorithme celui-ci garde en mémoire les différentes configurations rencontrées, on garde alors la solution avec la profondeur minimale ceci évite alors les duplications

Avant la recherche on regarde le nombre minimum de coup nécessaire à chaque robot pour atteindre une destination en prenant en compte le fait qu’un autre robot peut servir de mur (si les robots sont dans une position idéale). Ceci nous permet d’éliminer une branche de la recherche quand le robot actif ne peut pas atteindre la case cible dans le nombre de coups restant pour la profondeur de recherche.

**Organisation :**

Gabriel : Conception, réplication, résolution

Mathieu : Conception, réplication, résolution

Dorian : Déplacement, interface, recherche

Adrien : Déplacement, recherche, résolution

Julien : Déplacement, recherche

Etienne :

Florian : Résolution

Yannis : Conception, recherche

**Conclusion**

Vous pouvez accéder à notre méthode via le lien git suivant : <https://forge.univ-lyon1.fr/p1408826/idee_fixe> .

Pour le rendu, j'attend de vous :  
- un rapport par groupe qui précisera  
   \* l'organisation du groupe (qui a fait quoi)  
   \* lien sur le depot du code  
   \* vos choix pour les robots (capteurs, modes de déplacement, ...)  
   \* implémentations  
     \* contrôle bas niveau du robot : comment se font les déplacements  
       du robots, quels capteurs, ...  
     \* phase d'exploration : algorithme utilisé, résultats  
     \* résolution du problème (comment envoyer un robot a une position  
       donnée) : algorithme, résultats  
     \* synchronisation des robots (comment sont transmis les différents  
       ordres aux différents robots, comment se fait la synchronisation