Simulation d'une Équipe de Robots Pompiers

Mehdi El Idrissi El Fatmi
Moez Jaafoura
Mohammed Amine Hannoun

November 15, 2024

Contents

1 Introduction **2**

2 Méthodologie **2**
2.1 Conception des Classes 2
2.2 Algorithme Utilisé pour le plus court chemin 2
2.3 Stratégie utilisée 3

3 Tests et Résultats **3**

4 Discussion **4**

5 Conclusion et Perspectives **4**

1 Introduction

Ce rapport présente la conception et l'implémentation d'une simulation d'équipe de robots pompiers en Java. L'objectif est de développer une simulation permettant de gérer efficacement des robots autonomes dans un environnement naturel, tout en optimisant leurs interventions pour éteindre des incendies. Ce document détaille les choix de conception, les tests effectués et les résultats obtenus.

2 Méthodologie

2.1 Conception des Classes

La simulation repose sur plusieurs classes Java, dont :

- **Carte** : représente une matrice à deux dimensions où chaque élément correspond à une case spécifique ayant des propriétés définies (par exemple, type de terrain, position).
- **Case** : La classe Case représente une cellule de la carte, définie par ses coordonnées (ligne, colonne) et la nature de son terrain (e.g., eau, forêt, roche).
- **Robot** : classe abstraite définissant les propriétés des robots (position, réservoir, vitesse).
- **Incendie** : définit les incendies avec leur position et leur intensité.
- **Simulateur** : exécute les événements planifiés dans un modèle à événements discrets, gère l'évolution des robots et des incendies, et met à jour l'affichage en temps réel via une interface graphique.
- **Evenement** : La classe abstraite Evenement représente les actions planifiées dans la simulation, définies par une date et une méthode execute(). Ses sous-classes principales sont Intervention (éteindre un incendie), Remplissage (recharger le réservoir d'eau par un volume donné), et Deplacement (déplacement d'un robot). Elle organise les actions via une structure commune et extensible.
- **Strategie** : La classe Strategie contient l'implémentation de la stratégie du chef pompier, permettant de coordonner les actions des robots pour gérer efficacement les incendies.

2.2 Algorithme Utilisé pour le plus court chemin

L'algorithme de Dijkstra a été implémenté pour calculer les plus courts chemins entre deux cases (départ et destination) en respectant les contraintes spécifiques de chaque robot et de chaque carte. Cela permet d'optimiser les déplacements et de minimiser le temps total nécessaire pour éteindre les incendies.

2.3 Stratégie utilisée

Dans ce projet, nous n'avons pas utilisé les stratégies proposées par le sujet, à savoir la *stratégie élémentaire* et la *stratégie évoluée*. À la place, nous avons adopté une **stratégie optimisée** pour coordonner les robots de manière efficace et réduire le temps total des interventions. On commence par trier les incendies à l'aide de la méthode `trierIncendiesParProximite`, qui organise les incendies selon leur proximité afin de minimiser les déplacements entre eux. Ensuite, pour chaque incendie, on sélectionne le robot optimal en calculant pour chaque robot le **temps nécessaire pour terminer complètement l'extinction de l'incendie**. Ce calcul prend en compte le *temps courant* du robot, qui peut inclure son état actuel, comme un déplacement en cours, une intervention ou une attente.

Si le robot a suffisamment d'eau dans son réservoir, le temps total inclut uniquement :

- Le déplacement vers l'incendie.
- Le temps d'intervention nécessaire pour éteindre l'incendie.

Cependant, si le robot n'a pas assez d'eau, le calcul du temps inclut également :

- Le déplacement vers le point d'eau le plus proche.
- Le temps de remplissage de son réservoir.
- Le déplacement vers l'incendie.
- Le temps nécessaire pour l'intervention.

De plus, si l'incendie nécessite plus d'eau que la capacité maximale du réservoir du robot, les **allers-retours nécessaires** entre le point d'eau et l'incendie sont pris en compte dans l'estimation.

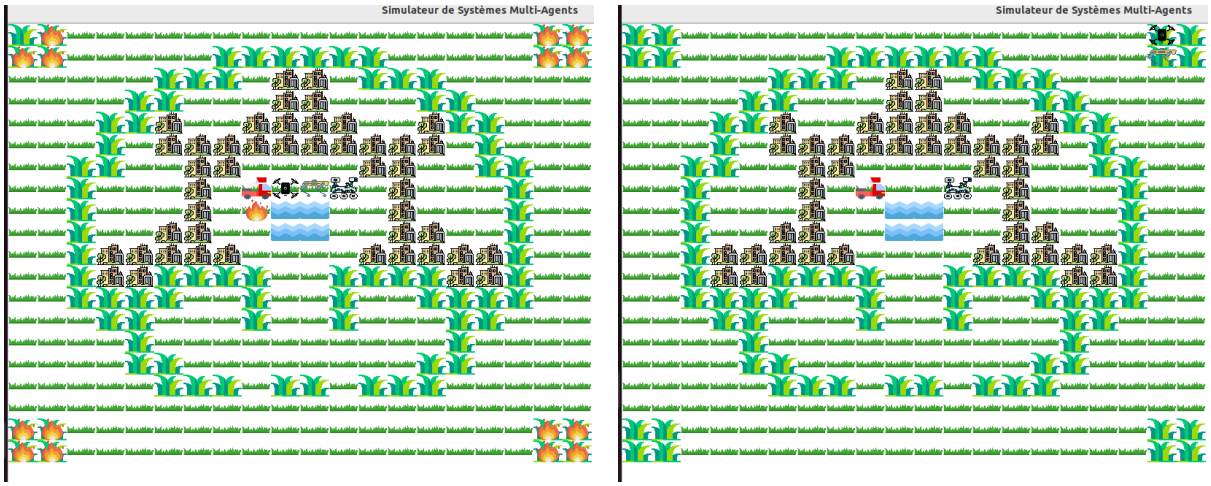
Le robot avec le **temps total minimal** pour éteindre l'incendie est alors sélectionné, garantissant une intervention optimisée même dans les cas complexes nécessitant plusieurs allers-retours. Les événements de déplacement, de remplissage et d'intervention sont planifiés dans une file de priorité, garantissant une exécution dans le bon ordre. Cette approche permet de coordonner les robots efficacement tout en optimisant le temps total d'intervention.

3 Tests et Résultats

Des scénarios prédéfinis ont été testés pour valider la fonctionnalité globale de la simulation. Ces tests ont été réalisés en plusieurs étapes pour s'assurer de la robustesse et de la précision des différentes composantes du projet :

- **Tests de lecture de données** : Vérification que les fichiers d'entrée contenant les informations sur la carte, les incendies et les robots sont correctement lus et interprétés pour initialiser la simulation.
- **Tests de déplacement élémentaires** : Validation des déplacements des robots d'une case à une autre en respectant les contraintes du terrain et les propriétés spécifiques de chaque type de robot (Scénario 0 et 1 présents dans le sujet).

- **Test de simulation du chef pompier** : Simulation complète impliquant la coordination des robots par le chef pompier pour éteindre les incendies. Ce test a permis de valider la sélection des robots, le calcul des chemins optimaux, et la planification des interventions.



(a) Début de la simulation (b) Fin de la simulation

Figure 1: Comparaison du début et de la fin de la simulation sur la carte `hell.map`.

4 Discussion

Cependant, notre stratégie présente certaines limites. Par exemple, les robots très éloignés ou nécessitant un long temps de déplacement pour atteindre un incendie ne sont jamais sélectionnés. Bien que cela permette d'optimiser localement le temps d'intervention, cela peut poser problème dans le cas d'un incendie de grande intensité nécessitant plusieurs robots. Dans un tel scénario, un robot éloigné pourrait être mobilisé pour contribuer à éteindre l'incendie en complément d'autres robots, mais cette situation n'est pas traitée dans notre stratégie actuelle.

5 Conclusion et Perspectives

Pour conclure sur notre projet Java, nous avons réussi à implémenter une simulation fonctionnelle permettant de coordonner efficacement une équipe de robots pour éteindre des incendies. Tous les sujets requis ont été traités, et nous avons mis un point d'honneur à concevoir une hiérarchie de classes cohérente, tout en minimisant les redondances de code pour garantir une architecture claire et maintenable.

Des améliorations sont néanmoins possibles, notamment l'intégration de stratégies collaboratives entre les robots ou la modélisation plus réaliste de la propagation des incendies. Ces perspectives ouvrent la voie à des développements futurs pour enrichir les fonctionnalités et optimiser davantage les performances du système.

Annexe

- Documentation utilisateur : incluse dans le manuel.