Système Multi-Agents

Bridge Builder : Construction Collaborative de Ponts

Cours : Multi-Agent Systems
Master 2 - Année 2025-2026

Auteurs :

Malaussena Dylan
Carré Matthias

Date : Janvier 2026

Table des matières

1 Introduction

Le projet **"Bridge Builder"** dans le contexte du cours de **"Multi Agent System"** a pour but de simuler un environnement dans lequel un groupe d'**agents autonomes** doit collaborer pour résoudre un problème défini. Cela permettra de rendre concrets les concepts vus durant ce cours et pouvoir mieux les comprendre.

Dans notre projet les agents se trouvent sur une rive et doivent collaborer et construire, à l'aide des ressources présentes, un **pont** pour permettre à un agent de rejoindre l'objectif de l'autre côté de la rive. Pour cela on utilisera plusieurs agents, chacun avec une classe définie et un environnement imagé.

1.1 Problématique

La difficulté pour résoudre le problème vient des contraintes définies afin d'imiter les problèmes récurrents des **MAS**.

- **Vision limitée** : les agents n'ont accès qu'aux cases dans leur vision. Ils doivent explorer l'environnement pour trouver ce qu'ils cherchent.
- **Dépendance des tâches** : un constructeur ne peut pas construire sans les récolteurs, et les récolteurs ne peuvent pas accéder à l'autre rive.
- **Coordination spatiale** : les agents peuvent se bloquer, ils doivent éviter ces situations.

1.2 Objectifs

Le but est de **minimiser le temps** nécessaire pour arriver à l'objectif de l'autre côté du rivage. Pour cela le système essaye de :

- **Optimiser la chaîne logistique** : il faut faire en sorte que les constructeurs aient toujours accès à des ressources données par les récolteurs.
- **Assurer une continuité dans la construction** : les constructeurs essayent de collaborer pour construire de manière logique et pas individuelle.
- **Montrer une intelligence collective** : on essaye d'observer qu'à l'aide d'agents simples on peut obtenir une coopération intelligente pour réaliser une tâche plus complexe.

1.3 Solution

Pour réaliser cela nous avons implémenté un **MAS** (Multi agent system) composé de 3 types d'agents différents (**récolteur, constructeur, manager**) évoluant dans une grille 2D dynamique. La solution est basée sur un framework où la coopération est facilitée par la modification de l'environnement et l'indication de certains agents.

2 Modélisation du MAS

Cette partie détaille la théorie du système implémenté, en définissant la nature des agents, leur environnement et les interactions entre ces derniers pour accomplir une tâche.

2.1 Les agents

Les agents ont pour but de suivre le modèle **BDI (Belief Desire Intention)** :

— **Croyances** : les agents n'ont pas la connaissance entière de la carte, ils sont limités par leur champ de vision. Il va croire en l'existence de son objectif et doit l'amener dans son champ de vision.

— **Désirs** : Chaque agent a un ou plusieurs buts qui sont définis par son rôle (construire, indiquer ou récolter).

— **Intentions** : à chaque moment l'agent sélectionne une action à suivre pour satisfaire son désir.

2.2 Les rôles

Afin de réaliser la tâche, les agents se voient affecter un rôle, ce qui permet de **diviser le travail**.

2.2.1 Agent de récolte

— **But** : maximiser le stock de bois disponible.

— **Comportement** : il recherche du bois disponible et le dépose dans la réserve.

2.2.2 Agent constructeur

— **But** : construire un pont sur l'eau.

— **Comportement** : récupère du bois depuis la réserve pour le déposer au niveau de l'eau afin de réaliser le pont.

2.2.3 Agent manager

— **But** : Rendre plus efficace le travail des autres agents.

— **Comportement** : quand il croise un autre agent, il lui donne une direction à suivre afin de réaliser son désir.

2.3 Environnement

Les agents évoluent sur une carte représentée par une **grille 2D** possédant les propriétés suivantes :

— **Accessibilité** : partiellement observable pour les agents.

— **Dynamisme** : Évolution avec la disparition des ressources et évolution du pont.

— **Contraintes** : certains éléments sont infranchissables.

2.4 Interactions et coordination

Lors de la construction du pont il y a deux types d'interactions :

— **Coordination par Stigmergie** : Les agents communiquent par l'environnement, un constructeur sait où il doit construire à l'aide du travail précédemment fait par ses collègues de même classe.

— **Communication de proximité** : Le manager vient influencer les agents afin de les guider dans leurs objectifs et rendre le travail plus efficace.

En cas de conflit il y a plusieurs mécanismes afin d'éviter que les agents se retrouvent bloqués. Des agents peuvent oublier leurs objectifs pendant un instant afin de libérer de l'espace et permettre une autre évolution.

3 Conception du système et logique du code

Cette partie décrit l'architecture du projet et les stratégies algorithmiques liées aux agents et leur collaboration. L'utilisation de **Python** était un compromis entre simplicité et utilité, que ce soit pour le code ou la collaboration dans l'équipe.

3.1 Architecture globale

Le système est conçu de manière **modulaire** pour séparer la logique de simulation, l'affichage et l'évolution des agents.

- **Moteur de jeu** : (`game.py` et `bridge_game.py`) : gère la boucle principale et l'activation des agents.
- **Environnement** : (`environment.py`) la grille définie par une carte où sont positionnés la rivière et possiblement des obstacles.
- **Contrôleur des agents** : (`agent.py`) : gère les logiques de fonction de perception et de mouvement.
- **Interface et rendu** : (`renderer.py` et `input_handler.py`) : visualisation en temps réel grâce à la bibliothèque **Pygame** et gestion de paramètres dynamiques.

3.2 Architecture des agents

La prise de décision est définie par une sélection d'actions et des fonctions de recherche de cibles :

- **Algorithme d'exploration** : l'agent n'a pas de réelle logique d'exploration, il se déplace juste au hasard tant qu'il ne trouve pas de but, cela permet de donner plus de valeur à l'agent manager.
- **Algorithme de déplacement** (`move_towards`) : les agents calculent une direction et essayent de réduire la distance tout en respectant la faisabilité des mouvements.
- **Gestion des blocages** (`stuck_counter`) : pour éviter les situations où la simulation ne peut plus avancer, cet algorithme vient changer l'objectif des agents bloqués pendant un certain temps afin de laisser la possibilité de nouvelles solutions.
- **Perception Locale** : chaque agent utilise `iter_visible_cells` pour la gestion de la vision restreinte. Ce qui renforce la nécessité du manager.

3.3 Stratégie de récolte et construction

- **Pour la récolte** : soit l'agent explore, se dirige vers une ressource, ou retourne vers le stock.
- **Pour les constructeurs** : il cherche de l'eau, s'il trouve dans son champ de vision un pont déjà existant il va le continuer sinon il en commence un nouveau.

3.4 Guidage du manager

Le **manager** est le seul agent qui possède des informations en dehors de son champ de vue, en contrepartie il n'interagit que de très près. Il injecte directement la cible dans la mémoire de l'agent avec lequel il interagit.

4 Utilisation

Cette partie explique comment utiliser et interagir avec la simulation.

4.1 Prérequis et installation

Le système est développé en **Python 3** et s'appuie sur la bibliothèque **Pygame**, ils sont donc nécessaires.

— **Installation des dépendances** : il est nécessaire d'avoir Python d'installé puis installer Pygame via la commande `pip install pygame`.

— **Extraction** : décompresser l'archive `Game.zip` ou cloner le projet git dans le répertoire local.

4.2 Lancement de la simulation

Pour exécuter notre MAS, exécutez le fichier `main.py` depuis la racine : `python3 main.py`

4.3 Interactions et Commandes

Il y a quelques options modifiables durant la simulation en temps réel :

— **Espace** : mettre en pause / reprendre.

— **Touche R** : réinitialiser la simulation.

— **Flèche haut / bas** : augmente la vitesse de la simulation.

— **Flèche droite / gauche** : modifie la distance de vue des agents.

4.4 Configuration

Il y a aussi un fichier `config.py` qui permet une configuration plus avancée de la simulation.

4.5 Agents

Les paramètres `NUM_GATHERERS`, `NUM_BUILDERS` et `NUM_MANAGERS` permettent de choisir le nombre d'agents de classes respectives.

4.6 La carte

La carte est aussi modifiable, pour cela il suffit de créer un fichier texte avec le bon format, le déposer dans le dossier `/maps` puis le mettre à `MAP_FILE` dans les config. Par exemple :

```
0 0 0 1 1 0 0
0 2 0 1 1 0 0
0 0 0 1 1 0 0
```

Chaque chiffre correspond à une case :

— **0** = case de terre

— **1** = case d'eau

— **2** = stock de bois

— **3** = mur infranchissable

Cela permet de tester d'autres configurations et observer d'autres comportements.

5 Répartition équipe

Le projet s'est fait en coopération à 2, où l'on a discuté sur ce que l'on voulait implémenter puis réparti le travail. Voici la répartition dans les grandes lignes car nous nous somme entre aider par moment :

5.1 Contribution de Dylan

— **Développement de l'environnement** : grille dynamique, gestion des collisions et construction du pont par étapes.

— **Développement du moteur graphique** : mise en place de la boucle avec Pygame et rendu visuel du jeu.

5.2 Contribution de Matthias

— **Modélisation logique MAS** : définition des rôles et leurs interactions et communications.

— **Implémentation de différents styles de maps** grâce aux fichiers texte.

6 Conclusion

Ce projet nous a permis d'appliquer la logique de **BDI** dans les grandes lignes ce qui nous a familiarisés avec les **MAS** dans un cas concret. Il a été intéressant de voir une simple logique de jeu avec des agents simples évoluer ensemble pour atteindre l'autre rive. Le projet pourrait facilement être plus développé en imaginant d'autres agents et ressources mais l'idée générale est faite.