Jérémy Dollé | Etienne Debard

5A | Polytech lyon

SMA Projet

Tour d’hanoi – système multi-agents

Table des matières

[Introduction 2](#_Toc503721415)

[MaDKit 2](#_Toc503721416)

[Principe général 2](#_Toc503721417)

[Agents 2](#_Toc503721418)

[Interactions 2](#_Toc503721419)

[Envoie de messages 2](#_Toc503721420)

[Coordination 3](#_Toc503721421)

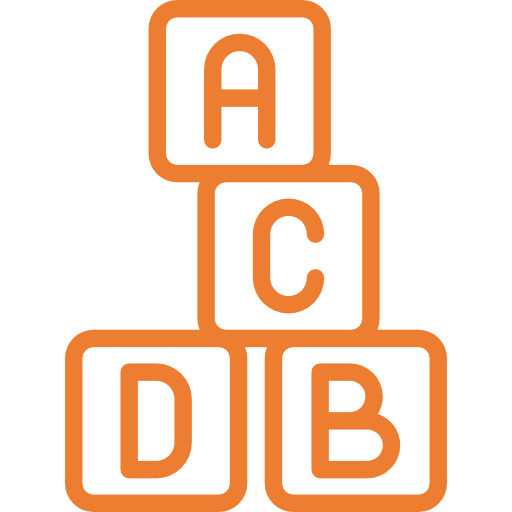
[Exemple 4](#_Toc503721422)

[Resultat 5](#_Toc503721423)

[Améliorations 5](#_Toc503721424)

[Conclusion 6](#_Toc503721425)

# Introduction

Ce rapport a pour but d’exposer la résolution d’un problème grâce à la planification multi-agents. Il s’agit ici de partir d’une situation initiale pour arriver à une situation visée. Par exemple 4 blocs A, B, C, D mélangés et posé sur une pile, le but est d’obtenir une situation de façon a ce que les blocs soient placés sur la pile la plus à gauche dans l’ordre alphabétique.

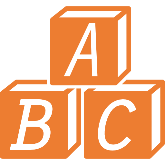
# MaDKit

Le but De ce TP étant l’implémentation d’un algorithme d’apprentissage, nous avons décidé de choisir une librairie pour tout ce qui est question d’affichage pour éviter de perdre du temps à concevoir une interface et aussi pour vérifier rapidement nos résultats. La librairie MaDKit est très simple d’utilisation est colle parfaitement à nos besoins, c’est pour cette raison que nous avons opté pour cette dernière.

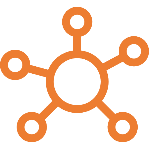
# Principe général

Dans cette partie, nous allons détailler la façon dont nous avons implémentés notre problème. Pour commencer, nous allons définir quels sont les agents, quelles interactions ont-ils entre eux et pour finir la stratégie adoptée pour résoudre le problème.

## Agents

Dans notre solution, un agent est en fait un bloc. Chacun de nos agents connait l’environnement dans lequel il évolue ainsi que la configuration cible qu’il doit atteindre.

## Interactions

De façon à ce que les agents puissent évoluer, ils doivent interagir les uns avec les autres de façon à se pousser et pouvoir se déplacer dans le cas ou un autre agent en bloque un autre.

### Envoie de messages

L’une des premières choses que nous avons faites est l’envoi de messages entre agents de façon à créer des interactions. Le contenu des messages est formé d’une requête ainsi que des contraintes. La requête est ici une demande de déplacement et les contraintes sont : de ne pas se déplacer là ou l’agent qui procède à la requête veut aller ainsi que l’endroit actuel.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A |  |  |  | **A** | 🡺 |  |  |  |  | **A** |
| B |  |  |  | **E** | B |  |  |  | **E** |
| C |  |  |  | **D** | C |  |  |  | **D** |
| D |  |  |  | **C** | D |  |  |  | **C** |
| E |  |  |  | **B** | E |  | A |  | **B** |
| ***C1*** |  | ***C2*** |  | ***C3*** |  | ***C1*** |  | ***C2*** |  | ***C3*** |

Bonjour ‘A’ ! Pousse-toi stp mais pas en C1 ni en C3

## Coordination

Chaque agent est hébergé par un thread indépendant. Il n’y a donc pas a proprement dit de coordination entre les différents agents. Au premier abord, cette non coordination peut être synonyme de discorde. Cependant, le système de messagerie avec attente de réponse intégrée à MadKit permet d’implémenter des dialogues cohérents et coordonnés.

De ce fait, après qu’un agent ai envoyé un message à un autre, il va attendre une réponse de ce dernier avec une limite de temps avant de passer à une action suivante. On obtient donc un dialogue global qui devient coordonné.

Afin de pouvoir contrôler la rapidité d’exécution de nos agents et de les voir évoluer plus aisément, nous avons ajouté un agent Schedule qui va réguler la rapidité d’exécution de chaque agent. On peut ainsi ralentir le programme ou le stopper ou passer à une exécution itération par itération.

# Exemple

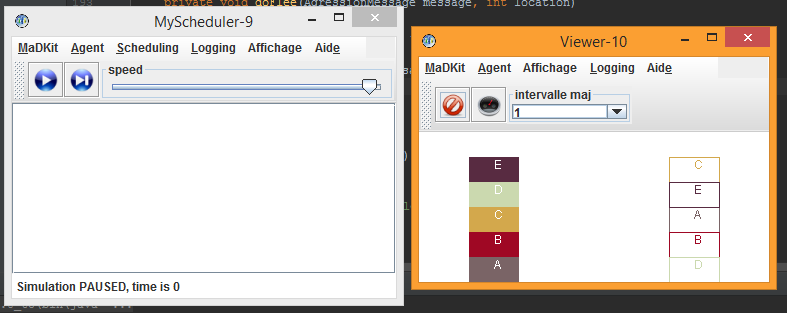


Figure : Initialisation d'une résolution

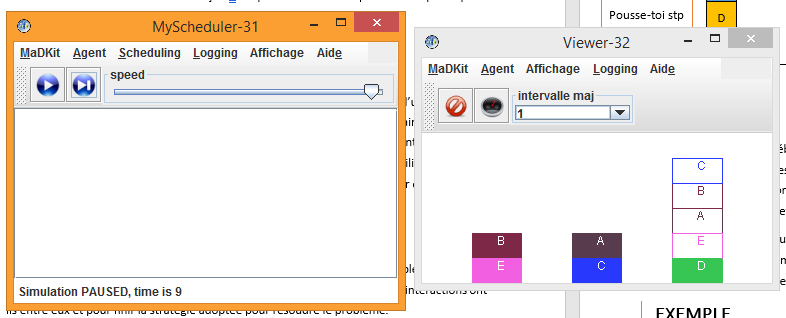


Figure : problème en cours de résolution

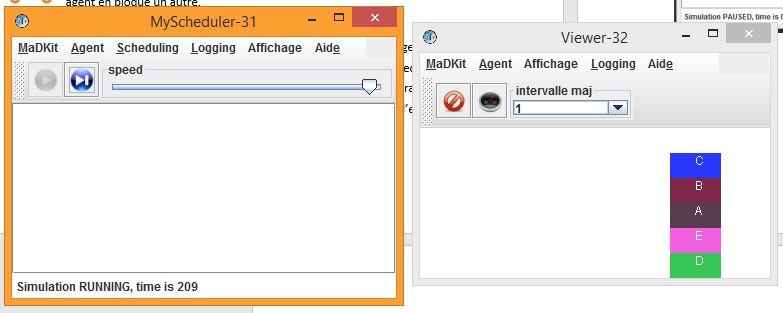


Figure : problème résolut

# Resultat

Dans le cas d’une configuration avec tous les agents sur la tour gauche au départ, avec pour objectif une case de la tour droite, le taux de réussite de résolution est de 100%. Cependant le temps de résolution pas s’accroitre exponentiellement avec le nombre d’agent (donc de blocs à placer dans notre tour). Voici quelques chiffres des durée moyennes que nous avons pu observer :

|  |  |
| --- | --- |
| Nombre d’agents | Durée de la résolution |
| 3 | <1s |
| 4 | 2s |
| 5 | 4s |
| 8 | 10s |
| 10 | 14s |
| 20 | 60s |
| 50 | >600s |

Pour que ces chiffres soient plus parlant il aurait été intéressant de les comparer avec ceux d’une solution optimale.

Cependant, la solution SMA offre une grande malléabilité. En effet, la façon dont nous avons développé notre système SMA, permet d’imaginer d’autres scénarios, par exemple avec des agents dispersés sur les 3 tours lors de l’initialisation, avec des buts pas forcements tous sur la même tour. On pourrait également imaginer ajouter d’autres d’autres tours et passer à un plus grand nombre d’agents. Ces différents scénarios seraient solvables par notre solution sans y apporter de changement.

# Améliorations

Nous avons remarqué que certaines fois, ces agents se retrouvent en train de « voler ». En effet, rarement, des agents qui ne devraient pas pouvoir bouger, car ils ont un autre agent au-dessus d’eux, se déplacent quand même sur une autre tour. Ce problème n’apparait que rarement, nous avons donc choisi de nous concentrer sur l’amélioration des performances, notamment trouver une bonne durée d’attente de réponse des agents pendant les conversations afin de maximiser les performances. Mais, nous pensons a première vu que le problème pourrais venir d’un soucis de synchronisation entre les différents threads de chaque agent.

# Conclusion

L’utilisation des Systèmes Multi Agent permet la résolution de problèmes de façon distribué, là ou une approche centralisée pourrait s’avérer compliqué à mettre en place. Dans notre cas, on sépare le but global en une multitude de petits buts donc les règles pour y parvenir sont plus simple à identifier. Puis avec une bonne gestion de la coordination des agents, chacun est capable d’atteindre son but et ainsi de résoudre le problème initial.