République Algérienne Démocratique et Populaire
الجمهورية الجزائرية الديمقراطية الشعبية
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
وزارة التعليم العالي و البحث العلمي

المدرسة الوطنية العليا للإعلام الآلي
(المعهد الوطني للتكوين في الإعلام الآلي سابقا)
École Nationale Supérieure d'Informatique
anciennement INI (Institut National de Formation en Informatique)

**Rapport TP4 IRIAD**

**2ème année cycle supérieur (2CS)**

**Option : Systèmes Intelligents et Science des Données (SID)**

# Simulation Multi-Agent du Jeu Colored Trails avec JADE

Réalisé par l'Équipe F1-Max :

Mlle. CHOUKRANE Yasmine
M. CHERGULAINE Oussama
Mlle. RAHOU Meriem
M. REZZIG Hamza
M. ARROUCHE Abdellah

Encadré par :

M. KECHID Amine

Année : 2024/2025

# Table des matières

# Résumé

Ce document présente en détail la conception, la modélisation et l'implémentation d'une simulation multi-agent flexible du jeu *Colored Trails*, réalisée avec le framework JADE. Ce projet met en oeuvre des agents autonomes dotés de capacités de planification, de négociation et d'adaptation, évoluant dans un environnement distribué et incertain. Le rapport met en lumière les choix algorithmiques, les stratégies comportementales des agents et les dynamiques d'interaction émergentes.


## 0.1 Introduction

Le développement de systèmes intelligents distribués est au coeur des préoccupations actuelles en intelligence artificielle. Les systèmes multi-agents (SMA) offrent une modélisation efficace de scénarios où plusieurs entités autonomes poursuivent leurs propres objectifs tout en interagissant avec les autres. Le projet présenté consiste en une simulation du jeu *Colored Trails*, où plusieurs agents doivent atteindre un objectif individuel sur une grille colorée, en utilisant des ressources limitées (jetons de couleur) et en négociant entre eux pour surmonter les obstacles.

## 0.2 Spécification du Jeu Colored Trails

### 0.2.1 Environnement

L'environnement est constitué d'une grille rectangulaire de dimensions configurables (par défaut 7x5), où chaque case est colorée aléatoirement parmi quatre couleurs possibles : `Rouge`, `Bleu`, `Vert`, et `Jaune`, et de multiples agents de nombres configurables (par défaut deux). Pour quun agent puisse se déplacer vers une case, il doit posséder un jeton correspondant à la couleur de cette dernière.

Chaque joueur est initialisé à une position de départ aléatoire et se voit assigner une case objectif différente, elle aussi choisie aléatoirement. Lobjectif est de rejoindre cette case en utilisant les jetons disponibles.

### 0.2.2 Objectifs des agents

Chaque agent dispose :
- d'une position de départ aléatoire,
- d'une position objectif aléatoire,
- d'un ensemble de jetons initiaux, calculé à partir de son itinéraire estimé (PCC).
- d'un comportement de négociation de jetons en cas de blocage.

### 0.2.3 Contraintes de Déplacement

- Un jeton est consommé à chaque déplacement valide.
- Les jetons ne sont pas renouvelables : les agents doivent donc planifier leur trajet en fonction de leurs ressources.
- Un agent est considéré comme **bloqué** sil ne peut pas se déplacer. Après trois tours bloqués consécutifs, le jeu se termine prématurément.

### 0.2.4 Initialisation des Jetons

Chaque agent reçoit un ensemble initial de jetons, dont la composition est dérivée de lanalyse de la longeur de son *plus court chemin* (PCC) vers sa cible.

### 0.2.5 Paramétrage Avancé via `GameConfig`

La classe `GameConfig` centralise tous les paramètres du jeu, permettant une personnalisation complète :

— Dimensions de la grille et taille des cellules graphiques.
— Nombre d'agents, ajusté selon la taille de la grille.
— Couleurs disponibles et styles graphiques associés (pastels, semi-transparence).
— Délai daffichage entre les tours pour visualisation fluide.
— Nombre de jetons par agent (calculé automatiquement ou défini manuellement).
— Éléments visuels : drapeaux, emojis, couleurs de traces individuelles.
— Seuil de tours bloqués avant élimination.
— Nombre maximum de tours pour terminer une simulation.

Cette modularité permet de tester facilement différentes configurations pédagogiques, danalyser des dynamiques dagents variées, et d'adapter le système à divers scénarios expérimentaux.

## 0.3 Architecture Multi-Agent

### 0.3.1 Agents autonomes et interaction pair-à-pair

Le système est composé dun ensemble de joueurs, nommés `PlayerX`, dont le nombre est configurable dynamiquement (jusquà 4). Chaque agent joueur possède une logique autonome et personnalisée. Il est initialisé avec :

— un itinéraire propre (position de départ et objectif),
— un ensemble de jetons de départ (lié à son chemin optimal),
— un identifiant graphique (emoji, couleur de trace, drapeau).

Chaque agent agit en fonction de son état local, selon une boucle perceptiondécisionaction :

— il analyse sa situation (bloqué ou non),
— planifie son prochain mouvement,
— décide sil doit négocier ou non,
— et communique avec les autres.

**MainAgent** joue un rôle central dans la simulation. Il assure :

— linitialisation des agents et de la grille,
— la coordination du déroulement des tours (tour par tour),
— la collecte des états de chaque joueur,
— la détection de la fin du jeu,
— le calcul final du score.

### 0.3.2 Communication inter-agent avec ACL

La communication entre agents repose sur les actes de langage ACL fournis par JADE. Les messages les plus utilisés sont :

— `INFORM` : utilisé par les agents pour transmettre leur état au `MainAgent` à la fin de chaque tour.
— `PROPOSE` : émis lorsquun agent cherche à obtenir un jeton quil ne possède pas.
— `ACCEPT_PROPOSAL` / `REJECT_PROPOSAL` : réponses aux propositions reçues, selon une logique dévaluation personnalisée.

Les négociations sont gérées pendant la phase de communication. Lorsquun besoin est détecté, lagent engage une négociation avec un autre joueur selon les critères suivants :
— la possibilité de compléter son propre chemin sans céder le jeton demandé,
— l'historique de trahison du partenaire (décompté dans le jeu),
— un facteur de confiance probabiliste défini dans `GameConfig`.

Un agent peut également décider de trahir une transaction acceptée, avec une probabilité ajustée selon les interactions passées. Cette dimension stratégique rend la dynamique plus réaliste et moins déterministe.

## 0.4 Stratégies des Agents Joueurs

### 0.4.1 Planification du Chemin

Chaque joueur calcule un itinéraire optimal vers son but (PCC), et tente d'utiliser ses jetons de manière optimale. En cas de blocage (pas de jeton approprié), il passe à la phase de communication.

Chaque agent calcule un **plus court chemin** (PCC) vers son objectif à laide dun algorithme de type calcule de distance Eucleudienne. Ce chemin tient compte de la topologie de la grille et de la couleur des cases. Lagent identifie ainsi la séquence de couleurs requise et tente de la parcourir en consommant les jetons correspondants.

Les jetons sont limités : un mauvais choix de déplacement peut rendre lobjectif inaccessible. Ainsi, chaque déplacement est validé uniquement si lagent possède le jeton nécessaire. Sinon, il entre dans une phase de négociation.

### 0.4.2 Détection de Blocage et Entrée en Négociation

Lorsqu'un agent est bloqué (absence du jeton requis pour la prochaine case), il attend la prochaine phase de communication pour commencer la négociation :

1. Il émet une requête (`PROPOSE`) spécifiant le jeton requis et celui quil est prêt à offrir.
2. Il attend les réponses des autres agents (`ACCEPT_PROPOSAL` ou `REJECT_PROPOSAL`).
3. Il évalue chaque proposition reçue selon une **fonction dutilité** :
   — Peut-il toujours atteindre son objectif sil donne le jeton proposé ?
   — Le jeton reçu améliore-t-il son PCC ?
   — Le partenaire est-il considéré fiable (cf. historique) ?
4. Il accepte ou refuse léchange en conséquence.

Chaque agent peut négocier avec plusieurs partenaires à la fois sil y a plus de deux joueurs.

### 0.4.3 Mémoire et Modulation de la Confiance

Chaque agent maintient un historique local des interactions avec les autres joueurs :
— nombre de trahisons subies,
— fréquence déchanges réussis,
— taux de satisfaction (échange utile ou non).

Cet historique est utilisé pour moduler dynamiquement la **probabilité d'accepter une proposition**.

### 0.4.4 Stratégie de Trahison

Un agent peut décider de trahir un partenaire : accepter une proposition déchange, recevoir le jeton, mais ne rien envoyer en retour. Cette décision est probabiliste et influencée par :
— la valeur perçue du jeton reçu (nécessaire au PCC ?),
— le coût de la perte de réputation (risque de rejets futurs),
— un coefficient de trahison initial défini dans `GameConfig`.

La fonction de trahison est donc adaptative, et peut évoluer au cours de la partie.

### Remarque pédagogique

Cette dynamique complexe illustre des principes réels des systèmes multi-agents :
— la négociation multi-critère (utilité, coût, historique),
— l'actualisation de croyances (réputation),
— léquilibre entre coopération et égoïsme.

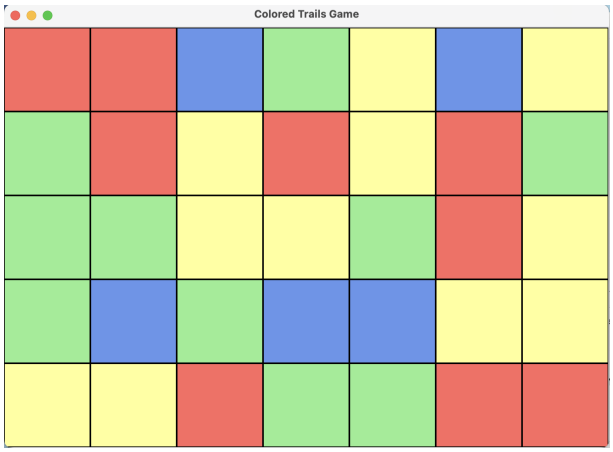
Cela fait du système un terrain dexpérimentation riche pour des étudiants ou chercheurs en intelligence collective.

## 0.5 Interface Graphique

L'application propose une interface personnalisée développée en Java avec `JADE`, permettant une visualisation dynamique de la simulation. Chaque case de la grille est colorée selon sa couleur logique (`Red`, `Green`, etc.) et dimensionnée selon la configuration.

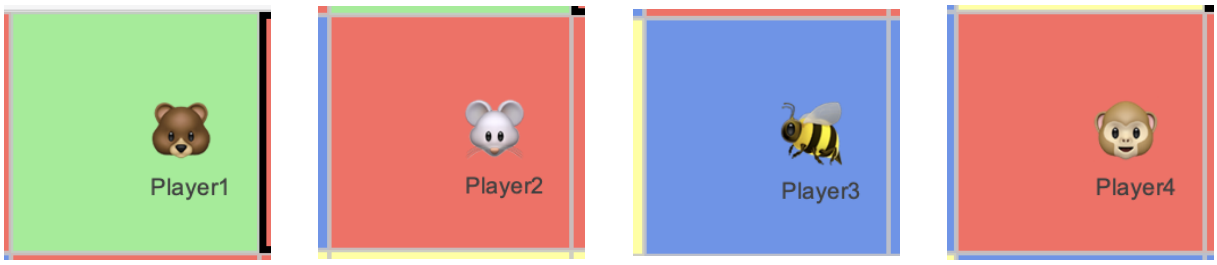
### 0.5.1 Fonctionnalités visuelles

— **Affichage de la grille avec couleurs réelles** :




Grille générée avec couleurs aléatoires visibles

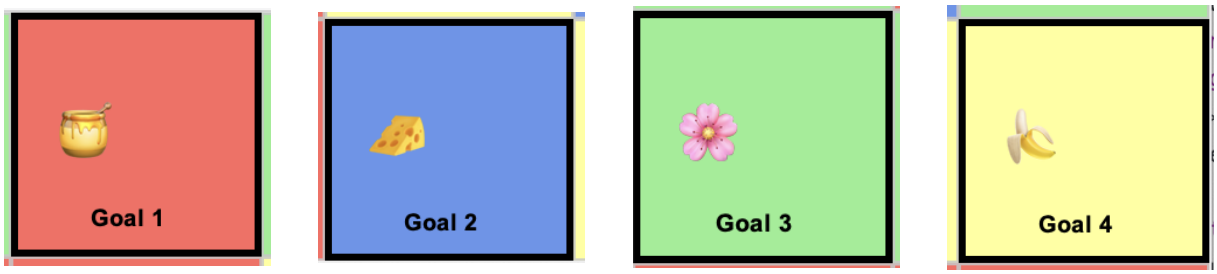
— **Émojis personnalisés pour représenter chaque joueur** :




Joueur 1 Joueur 2 Joueur 3 Joueur 4

Émojis affichés dans la grille pour chaque joueur

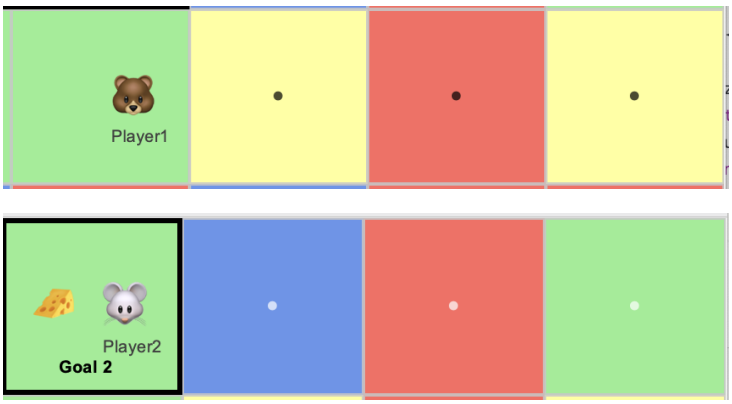
— **Objectifs marqués dans les cellules** :




Goal (P1) Goal (P2) Goal (P3) Goal (P4)

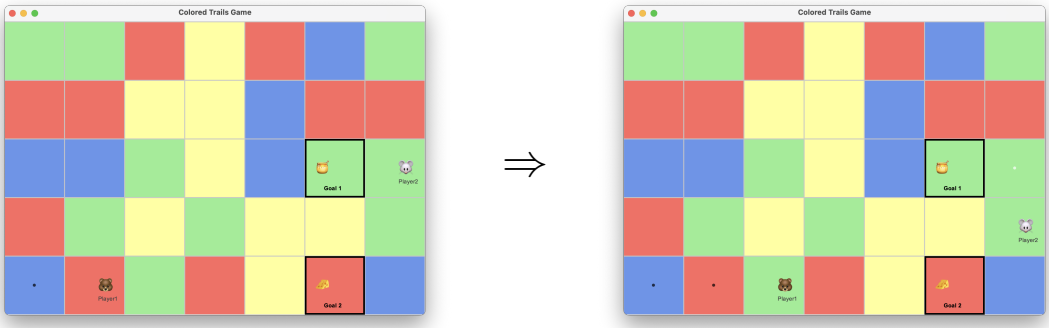
Objectifs affichés pour chaque joueur dans leur cellule dédiée

— **Traces de déplacement (trails) :**




Traces colorées semi-transparentes des déplacements

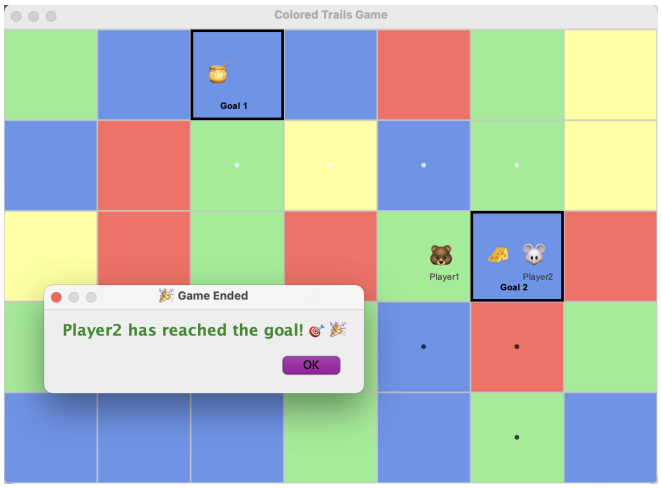
— **Rafraîchissement automatique à chaque tour :**




Mise à jour visuelle après chaque action

### 0.5.2 Exemples de Résultats Visuels de Plusieurs Tours

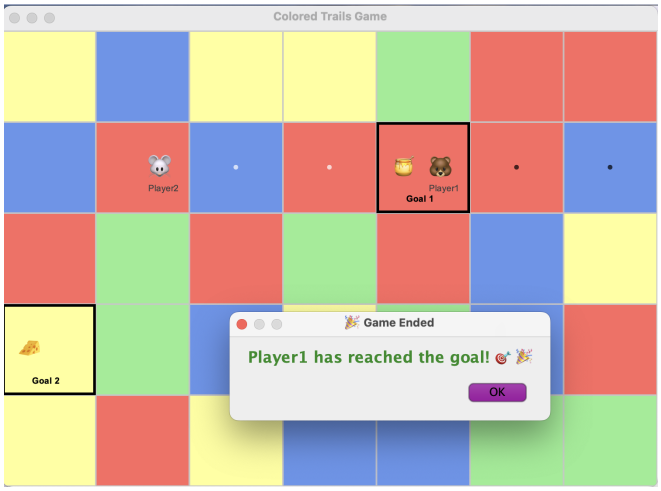

Tour 1

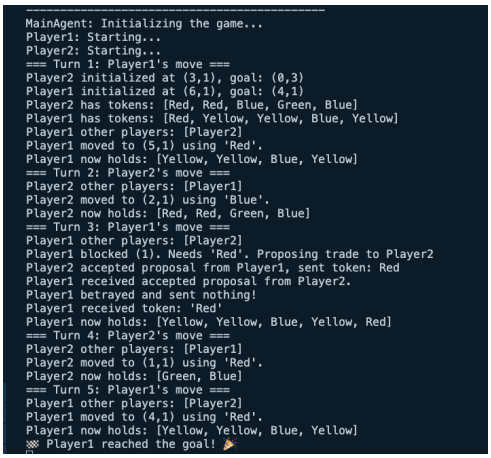



Tour 2



Tour 3




Tour 4

## 0.6 Gestion de la Simulation

### 0.6.1 Fin de partie

— Objectif atteint par un joueur.
— Blocage consécutif d'un nombre de tours definie dans GameConfig (3 par defaut).

### 0.6.2 Calcul du score

— +100 si objectif atteint.
— +5 par jeton restant.
— -10 par case restante si échec.

## 0.7 Conclusion et Perspectives

Ce projet met en oeuvre des notions avancées des systèmes multi-agents (SMA), notamment :

— la planification distribuée avec contraintes locales,
— la communication asynchrone par langage ACL,
— la modélisation de la confiance et de la réputation,
— la prise de décision adaptative (coopération vs trahison).

La flexibilité offerte par `GameConfig`, associée à une architecture modulaire et réutilisable, rend ce projet particulièrement intéressant dans un cadre pédagogique et expérimental. Il permet de configurer facilement de nouvelles situations, dintroduire des variables aléatoires ou de modifier le comportement des agents sans changer le cur du code.

### Perspectives futures

Plusieurs pistes damélioration ou denrichissement sont envisageables :

— Lintroduction dun apprentissage par renforcement (type Q-Learning) pour moduler les décisions en fonction des expériences passées.
— Lajout dune interface statistique ou dun graphe de confiance dynamique.