Rapport Technique : Modele d'Acteur dans les Systemes Multi-Agents

Application au Simulateur Simovies

Auteur: Marsso, Mougamadoubougary

Cours: DAAR - Algorithmique d'Essaims

Date: Janvier 2026

1. Introduction et Definition du Probleme

1.1 Contexte

Les Systemes Multi-Agents (SMA) representent un paradigme de calcul distribue ou plusieurs entites autonomes (agents) interagissent pour resoudre des problemes complexes. Le **modele d'acteur** (Actor Model), introduit par Carl Hewitt en 1973, fournit un cadre formel pour la conception de ces systemes concurrents.

Dans ce projet, nous implementons des strategies de combat robotique dans le simulateur **Simovies**, ou deux equipes de 5 robots s'affrontent. Chaque robot est un acteur autonome qui :

- Percoit son environnement via des capteurs (radar 360 degres, detection frontale)
- Prend des decisions locales basees sur des machines a etats finis
- Communique avec ses coequipiers via broadcast asynchrone
- Agit sur l'environnement (deplacement, tir)

1.2 Definition formelle du probleme

Soit A = {a1, ..., an} un ensemble d'agents et E l'environnement partage.

Chaque agent ai est defini par le tuple (Si, Pi, Ai, delta_i, Ci) ou :

- Si : ensemble fini d'etats internes

- Pi : E -> Oi : fonction de perception
- Ai : ensemble d'actions possibles (move, fire, turn, broadcast)
- delta_i : Si x Oi x Mi -> Si x Ai : fonction de transition
- Ci : canal de communication (messages recus)

L'objectif est de maximiser une fonction d'utilite collective U(A) representant le score de l'equipe (ennemis elimines, survie des robots).

1.3 Structure de donnees et FSM finale

Machine a etats finis (FSM) : la version finale de notre strategie B utilise des FSM simples, robustes, et fortement locales.

- **Main Bot B (Defenseur)** : etats {ADVANCING, HOLDING, DODGING}
- **Secondary Bot B (Kamikaze)** : etats {SEARCHING, ATTACKING}

Cette reduction volontaire d'etats limite les comportements instables (blocage, oscillations) et conserve une forte reactivite temps reel.

Communication inter-agents (version finale) :

- La version finale ne repose pas sur un protocole de synchronisation complexe entre Main Bots.
- Nous avons teste une variante avec broadcast de position du kamikaze (tracking) et regroupement, mais elle a ete abandonnee (voir Section 5) car elle degradait les performances.

2. Analyse theorique des algorithmes

2.1 Modele d'acteur et proprietes

Le modele d'acteur garantit plusieurs proprietes essentielles (Hewitt, Bishop, Steiger, 1973) :

1. **Encapsulation** : chaque robot maintient un etat prive inaccessible directement

2. **Communication asynchrone** : les messages sont non bloquants

3. **Comportement reactif** : reponse aux stimuli (perception + messages)

Notre implementation correspond au schema suivant (simplifie) :

```
public void step() {
    // Perception locale (radar, front sensor)
    // Decision par etat (FSM)
    switch (state) {
        case ADVANCING: stepAdvancing(); break;
        case HOLDING:   stepHolding();   break;
        case DODGING:   stepDodging();   break;
    }
}
```

2.2 Choix de conception : simplicite et verticalite

Dans un environnement temps reel, ajouter des etats de coordination (regroupement, chasse, consensus) peut augmenter :

- le nombre de transitions,
- les cas limites (messages obsoletes),
- les risques de blocage collectif.

Nous avons donc privilegie une strategie **verticale** et **simple** :

- un robot qui voit une cible passe en engagement,
- un robot qui ne voit rien continue la pression vers l'Ouest,
- l'evitement reste local et court.

Ce choix favorise un comportement stable et reproductible.

2.3 Variante testee : tracking kamikaze + regroupement (non retenue)

Nous avons teste une variante dans laquelle :

- le kamikaze broadcastait regulierement une position estimee (x, y),
- les Main Bots tentaient de suivre cette position via un etat de chasse / regroupement.

En pratique, cette variante a augmente :

- les blocages (forteresse coincee contre des obstacles),
- les mauvais angles de tir (rotation excessive),
- la complexite (et donc les bugs).

La version finale supprime ce mecanisme et obtient de meilleurs resultats.

3. Presentation des algorithmes implementes

3.1 Strategie B - "Fantom Danger" (version finale)

3.1.1 Main Bot B - Defenseur vertical autonome

Principe :

- Avance vers WEST par segments.
- Des qu'un ennemi est detecte au radar, bascule en HOLDING et engage la cible.
- Si un obstacle est detecte en frontal (non allie), declenche un DODGING court (NORTH ou SOUTH), puis reprend la direction WEST.

FSM :

```
ADVANCING --(ennemi detecte)--> HOLDING --(plus d'ennemi)--> ADVANCING
     |
     +--(obstacle)--> DODGING --(fin esquive)--> etat precedent
```

Code cle (idee) :

```
// Scan rapide + tir cadence
scanAndShoot();

// Etat ADVANCING : avance vers l'Ouest, esquive si bloque
// Etat HOLDING   : vise + tire sur l'ennemi le plus proche
// Etat DODGING   : mouvement court vertical, puis retour
```

3.1.2 Secondary Bot B - Kamikaze agressif simple

Principe :

- Cherche en permanence via radar.
- Des qu'un ennemi est detecte : ATTACKING (fonce + tire).
- Si l'ennemi disparait du radar (eliminee ou hors champ) : retour immediat en SEARCHING.

FSM :

```
SEARCHING <-> ATTACKING
```

Ce comportement simple evite la poursuite "a vide" et maintient une pression constante.

3.2 Strategie A - "KD Runners" (comparaison)

La strategie A (equipe A) sert de comparaison interne :

- comportements plus offensifs et plus mobiles,
- coordination plus faible,
- performances moins stables face a des adversaires agressifs.

4. Methodologie de test

4.1 Protocole experimental

Configuration :

- Arene : 3000 x 2000 mm
- Equipe A : spawn cote gauche (X ~ 200-500)
- Equipe B : spawn cote droit bas (X ~ 2500-2800, Y ~ 1400-1600)

Metriques observees :

1. Taux de victoire (parties gagnees)

2. Nombre de robots survivants en fin de partie (qualitatif)

3. Stabilite du comportement (blocages / oscillations)

4.2 Jeux de donnees et resultats

Tests effectues sur **10 parties** par adversaire :

Adversaire	Victoires	Defaites	Taux
BootingBerzerk	9	1	90%
CampFire	10	0	100%
RandomFire	10	0	100%

Observations :

- Contre **BootingBerzerk**, la strategie B finale gagne 9/10, la defaite survenant typiquement lors d'un enchainement defavorable (angles + collisions).
- Contre **CampFire**, victoire systematique, mais de nombreux robots restent vivants longtemps (style defensif proche).
- Contre **RandomFire**, victoire systematique grace a la regularite et la pression continue.

5. Discussion et analyse critique

5.1 Forces de l'approche finale

1. **Simplicite robuste** : peu d'etats, peu de transitions, moins de cas limites

2. **Verticalite efficace** : angles de tir plus propres, moins de rotations inutiles

3. **Reactivite** : bascule rapide en engagement des qu'une cible est visible

4. **Tolerance aux defaillances** : la perte d'un robot n'interrompt pas le comportement des autres

5.2 Limites identifiees

1. **Pas de strategie de formation explicite** : la cohesion de groupe est emergente, pas controlee

2. **Tir non predictif** : vise la direction actuelle, sans anticipation de trajectoire

3. **Evitement local** : le DODGING peut parfois conduire a des detours sous-optimaux

5.3 Retour d'experience : pourquoi la coordination par tracking a degrade les performances

La variante "kamikaze + tracking + forteresse qui suit" a ete testee, mais :

- la forteresse se bloquait plus souvent,
- les Main Bots se retrouvaient dans de mauvais angles pour tirer,
- la complexite augmentait les comportements incoherents.

Conclusion pratique : dans Simovies, **la verticalite et la simplicite** apportent un meilleur compromis que la coordination explicite.

5.4 Ameliorations possibles

1. Tir semi-predictif (estimation simple de mouvement ennemi)

2. Regles de spacing legeres (eviter la collision entre allies sans regroupement force)

3. Ajuster dynamiquement la cadence de tir selon la distance (proche = plus rapide)

6. Conclusion et perspectives

6.1 Bilan

Ce projet illustre l'applicabilite du modele d'acteur aux systemes multi-agents robotiques :

- acteurs autonomes (robots) avec etat prive,
- perception locale,
- decisions par FSM,

- communication eventuelle (variante testee).

Les resultats experimentaux montrent que la version finale de la strategie B, **plus simple et plus verticale**, est la plus efficace sur notre suite de tests, avec des taux de victoire eleves et une meilleure stabilite.

6.2 Perspectives

Des extensions envisageables incluent :

- un tir plus intelligent (anticipation),
- une formation dynamique legere (sans regroupement rigide),
- des mecanismes d'exploration plus varies lorsque l'ennemi est absent.

References

[1] C. Hewitt, P. Bishop, R. Steiger. "A Universal Modular ACTOR Formalism for Artificial Intelligence", IJCAI 1973.

[2] A. Chakraborty, A.K. Kar. "Swarm Intelligence: A Review of Algorithms", Nature-Inspired Computing and Optimization, 2017.

[3] M. Starzec, G. Starzec, A. Byrski, W. Turek. "Distributed ant colony optimization based on actor model", Parallel Computing, 2019.

[4] W. van der Hoek, M. Wooldridge. "Multi-Agent Systems", Handbook of Knowledge Representation, 2007.

[5] R.C. Cardoso, A. Ferrando. "A Review of Agent-Based Programming for Multi-Agent Systems", Computers, 2021.