Auteurs : Alain KARAPETIAN, Timeo RAPHOZ, Enzo VIGNE

2025-2026

Tuteur : Julien SOULE

MARIO: Cahier des charges

Master Mathématiques Informatiques Appliquées aux Sciences Humaines et Sociales (MIASHS)

Master 1

Janvier 2026

Table des Matières

1 PROJET MARIO

1.1 *Objectif général*

L'objectif du projet est de rendre l'utilisation de techniques de *Reinforcment Learning (RL)* et *Multi-Agent Reinforcment Learning (MARL)* la plus agnostique de connaissances expertes possible. Cette plateforme doit s'appuyer sur des technologies (librairies) existantes afin de permettre au mieux d'automatiser des tâches techniques habituellement réalisées à la main.

À noter que ce projet a pour vocation d'avoir une application locale, ce qui veut dire que son API n'est utilisable que localement pour l'application.

Ce logiciel doit permettre de choisir un environnement de jeu vidéo, un ou plusieurs algorithme(s) d'apprentissage par renforcement, de tester et visualiser le résultat de l'entraînement en entrant des commandes ou en interagissant avec une interface graphique. Ces fonctionnalité ne sont pas toutes réunies au sein d'une solution / logiciel / plateforme unique. Pour justifier le positionnement de ce projet, il est donc nécessaire de le situer par rapport aux solutions actuelles.

1.2 *Analyse de l'existant*

Le tableau suivant permet de mettre évidence les manques spécifiques que la plateforme MARIO vise à combler, il présente un comparatif entre des platforme existantes et celle de notre projet.

Critère / Plateformes	**Gama**	**Netlogo**	**Cromas**	**Matlab**	**MARIO**
Configuration d'agent (RL)	[ ]	[ ]	[ ]	[x]	[x]
Agnostique aux langages	[ ]	[x]	[x]	[ ]	[x]
Documentation	bonne	bonne	moyenne	Moyenne	bonne
Interface de visualisation	[x]	[x]	[x]	[x]	[x]

Après avoir compris la plus-value du projet, nous pouvons maintenant détailler les différentes étapes de notre mise en oeuvre.

2 MISE EN OEUVRE

Dans le cadre du projet, plusieurs tâches doivent être réalisées. Celles-ci correspondent aux fonctionnalités à implémenter afin d'assurer le bon fonctionnement de la plateforme. Ces tâches peuvent être organisées et planifiées de manière itérative et chronologique. Les différentes parties présentées ci-dessous détaillent l'ensemble des tâches ainsi que leurs critères de validation. Il est toutefois important de préciser que les technologies proposées sont susceptibles d'évoluer si d'autres solutions sont jugées plus appropriées.

2.1 *Revue de littérature et montée en compétences*

Cette première phase du projet est consacrée à la préparation et à la structuration des bases nécessaires à son bon déroulement. Cette phase préliminaire a pour objectif d'assurer une compréhension commune et approfondie du sujet, des technologies, outils et frameworks associés à travers une revue de littérature approfondie et l'étude des concepts fondamentaux liés au Reinforcement Learning (RL) et au Multi-Agent Reinforcement Learning (MARL). Elle permet notamment d'identifier les approches existantes, les bonnes pratiques, les limites des solutions actuelles et les choix technologiques pertinents pour la conception et le développement de la plateforme.

Cette étape est indispensable et constitue un prérequis au lancement des phases de conception et d'implémentation du projet.

2.2 *Développement de l'API*

Une API (Interface de Programmation d'Application) est un ensemble de fonctions et de protocoles permettant à différents logiciels ou composants de communiquer entre eux. Elle sert à exposer les fonctionnalités nécessaires au projet de manière standardisée, pour que d'autres programmes puissent les utiliser sans connaître les détails internes de l'application.

Intitulé de la tâche	**Description**	**Technologies**	**Critère de Validation**
Sélection et l'initialisation d'environnements.	L'API doit permettre la sélection et l'initialisation de différents environnements.	PettingZoo, MAgent2	Tests unitaires : Initialiser un environnement de jeu OvercookedAI

Intitulé de la tâche	**Description**	**Technologies**	**Critère de Validation**
Configuration d'algorithmes MARL/RL	L'API doit permettre de configurer les algorithmes RL/MARL (ex : PPO, MAPPO,etc...).	PyTorch, EPyMARL, Mava, JaxMARL, MARLLib	Tests unitaires : sélectionner un algorithme PPO en passant par les différents paramétrages
Gestion intégrée des phases	L'API doit permettre la pré-configuration de l'environnement ainsi que l'entraînement et les tests.	PettingZoo, MAgent2	Tests unitaires : pré-configuration de l'environnement OvercookedAI, pré-configuration de l'entraînement avec l'algorithme PPO et des tests

2.3 *Optimisation automatique des hyper-paramètres*

Les hyper-paramètres désignent les paramètres d'un modèle d'apprentissage permettant d'optimiser la qualité et la rapidité de l'apprentissage. Sélectionner et régler les bonnes valeurs d'un hyper-paramètre sert à adapter l'apprentissage du modèle aux contraintes et exigences du problème à résoudre.

Intitulé de la tâche	**Description**	**Technologies**	**Critère de Validation**
Optimisation de l'efficacité d'apprentissage	Utilisation d'algorithmes pour l'optimisation automatique des hyper-paramètres et arrêt des essais non prometteurs (élagage).	Librairie Optuna	Tests unitaires : exécuter l'optimisation des hyper-paramètres sur l'algorithme de PPO précédemment configuré.

Intitulé de la tâche	**Description**	**Technologies**	**Critère de Validation**
Parallélisation des essais	Possibilité d'exécuter les essais de multiples agents en parallèle sans modification du code source.	Librairie Optuna	Tests unitaires : Configurer plusieurs essais d'entraînement et exécuter plusieurs optimisations simultanées des algorithmes lancés.
Visualisation (Optionnel)	Outils graphiques pour observer l'évolution et les résultats des optimisations effectuées.	Librairie Optuna	Tests unitaires : Afficher un graphe représentant deux hyper-paramètres sélectionnés montrant l'évolution des résultats.

2.4 Mise en place d'un guide pour les utilisateurs

Un tutoriel/guide peut grandement aider un nouvel utilisateur qui souhaiterait utiliser une application qu'il ne connaît pas.

Intitulé de la tâche	**Description**	**Critère de Validation**
Tutoriel écrit pour l'utilisateur (Wiki)	Apporter un guide complet pour un utilisateur ayant des connaissances techniques limitées.	Tests utilisateurs / Feedback: faire remplir une grille de satisfaction (note de 1 à 10 + commentaire optionnel) couvrant chaque étapes du wiki.
Complément vidéo	(Optionnel) Vidéo de démonstration pour illustrer la prise en main de l'outil.	Tests utilisateurs / Feedback

2.5 *Mise en place d'une interface CLI / GUI*

Optionnellement une interface CLI voire GUI facilitant l'utilisation de l'application. Le but d'une interface CLI est d'exécuter l'application via des lignes de commandes (texte), un GUI permet de faire ce qu'un CLI fait mais grâce à une interface visuelle, rendant l'application plus accessible.

Intitulé de la tâche	**Description**	**Technologies**	**Critère de Validation**
Commandes documentées (CLI)	Commandes de base nécessaires au lancement du programme, accompagnées de leur documentation.	argparse (librairie python)	Tests unitaires: tester que les commandes réalisent les actions souhaités
Gestion de l'environnement	Commandes spécifiques pour créer et sélectionner l'environnement (OvercookedAI, algorithme PPO) pour le système multi-agent.	argparse (librairie python)	Tests unitaires: tester que les commandes créent et séléctionnent l'environnement
Sélection des algorithmes	Commandes permettant de choisir un ou plusieurs algorithmes à intégrer aux agents.	argparse (librairie python)	Tests unitaires: tester que les bons algorithmes sont bien séléctionné avec les commandes
Paramétrage et optimisation	Commandes pour sélectionner, paramétrer et optimiser les hyperparamètres.	argparse (librairie python)	Tests unitaires: : Validation des sorties de commandes (paramétrage et optimisation réussit).
Commande de test et rendu du résultat	Commandes pour lancer les suites de tests et visualiser les rapports d'exécution.	argparse (librairie python)	Tests unitaires : tester la visualisation des rapports, qu'ils soient bien générés

Intitulé de la tâche	**Description**	**Technologies**	**Critère de Validation**
Interface graphique Web	Interface visuelle permettant d'exécuter les commandes de manière intuitive.	Vue.js / React / Angular	Tests unitaires / fonctionnels: tests permettant de s'assurer que les interractions avec l'interface exécutent les bonnes actions

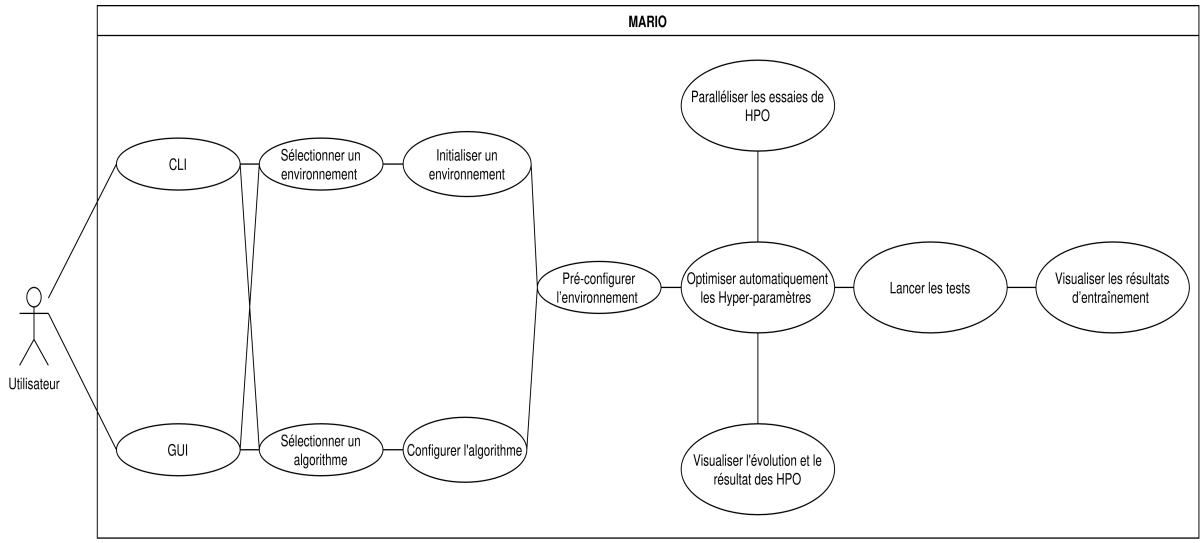

Figure 1: Cas d'usage

3 Diagrame

3.1 Cas d'usage

La **Figure 1** repeprésente les cas d'usages de la solution. Le diagramme inclus également les fonctionalité optionnele cité plus tôt.

4 Bibliographie

PettingZoo documentation. (s. d.). https://pettingzoo.farama.org/

Uoe-Agents. (s. d.). GitHub - uoe-agents/epymarl : An extension of the PyMARL codebase that includes additional algorithms and environment support. GitHub. https://github.com/uoe-agents/epymarl

Instadeepai. (s. d.). GitHub - instadeepai/Mava : A research-friendly codebase for fast experimentation of multi-agent reinforcement learning in JAX. GitHub. https://github.com/instadeepa

FLAIROx. (s. d.). GitHub - FLAIROx/JaxMARL : Multi-Agent Reinforcement Learning with JAX. GitHub. https://github.com/FLAIROx/JaxMARL

Replicable-Marl. (s. d.). GitHub - Replicable-MARL/MARLlib : One repository is all that is necessary for Multi-agent Reinforcement Learning (MARL). GitHub. https://github.com/Replicable-MARL/MARLlib

Optuna - A hyperparameter optimization framework. (s. d.). Optuna. https://optuna.org/

Albrecht, S. V., Christianos, F., & Schäfer, L. (2024). Multi-Agent reinforcement learning : Foundations and Modern Approaches. MIT Press.