

UNIVERSITÉ PIERRE ET MARIE CURIE

CAHIER DES CHARGES

TER DE MAÎTRISE D'INFORMATIQUE EN INTELLIGENCE  
ARTIFICIELLE ET DÉCISION

---

# Semi-supervised Learning Agents

---

*Auteurs :*

Lan ZHOU

Matthieu ZIMMER

*Superviseurs :*

Paolo VIAPPIANI

Paul WENG

4 février 2013

Version 1.0

## Table des matières

|   |          |
|---|----------|
| <b>Table de matière</b>                                   | <b>1</b> |
| <b>1 Objectif du TER IAD</b>                              | <b>2</b> |
| 1.1 Présentation . . . . .                                | 2        |
| 1.2 Objectifs principaux . . . . .                        | 2        |
| 1.3 État de l’art . . . . .                               | 2        |
| 1.4 Contraintes techniques . . . . .                      | 3        |
| <b>2 Description de la solution demandée</b>              | <b>4</b> |
| 2.1 Fonctionnalités bibliothèque . . . . .                | 4        |
| 2.2 Fonctionnalités simulateur . . . . .                  | 4        |
| 2.3 Contraintes de réalisation . . . . .                  | 5        |
| 2.4 Fonctionnalités additionnelles bibliothèque . . . . . | 5        |
| 2.5 Fonctionnalités additionnelles simulateur . . . . .   | 5        |
| <b>3 Composition du livrable</b>                          | <b>6</b> |
| <b>Références</b>   | <b>7</b> |

# 1 Objectif du TER IAD

## 1.1 Présentation

Ce document représente le cahier des charges du TER sur un apprentissage semi-supervisé d'agents intelligents en partant de l'**apprentissage par renforcement**.

Il s'agit de concevoir et développer un système dans lequel un agent intelligent évoluera (surveillance automatique, ...). Il devra être capable d'apprendre à partir de **retours limités** d'un expert humain du domaine. Imaginons par exemple un agent apprenant à jouer au TETRIS : l'expert pourrait dire à l'agent où placer le prochain élément ou bien si l'agent l'a bien placé. On s'attend alors à ce que cette aide diminue grandement le temps d'apprentissage de l'agent.

## 1.2 Objectifs principaux

Nous devons développer une **bibliothèque** indépendante du domaine dans lequel l'agent évolue. Elle fournira plusieurs algorithmes de base de l'apprentissage par renforcement (Q-Learning, SARSA, ...), plusieurs critères de performance, ainsi qu'une ouverture sur l'apprentissage semi-supervisé : c'est à dire avec les retours de l'expert pris en compte.

Dans un premier temps, l'expert pourra simplement compenser la fonction de récompense en précisant si l'agent a bien ou mal agit. Dans un second temps, si le temps le permet, l'expert pourra également agir sur le choix de l'action à entreprendre lors de l'exploration de l'agent, ou encore dire à l'agent s'il est temps d'exploiter ou d'explorer.

Parallèlement au développement de la bibliothèque, afin d'avoir une application pratique de la théorie, on utilisera ces algorithmes dans le **simulateur TORCS** sur l'apprentissage automatique de la conduite de voiture sur circuit. Nous modifierons également l'interface TORCS pour intégrer les retours positifs ou négatif de l'expert.

## 1.3 État de l'art

**Apprentissage non supervisé** L'apprentissage par renforcement fait référence à une classe de problèmes d'apprentissage automatique, dont le but est d'apprendre, à partir d'expériences, ce qu'il convient de faire en différentes situations, de façon à optimiser une récompense numérique au cours du temps.

L'agent cherche, au travers d'expériences itérées, un comportement décisionnel optimal, en ce sens qu'il maximise la somme des récompenses au cours du temps. Wikipédia [2012]

Il existe déjà de nombreux algorithmes éprouvés qu'il conviendra simplement d'implémenter. Richard S. Sutton [1998]

**Apprentissage semi-supervisé** Le domaine est déjà assez développé avec plusieurs algorithmes existants. Wikipédia [2013]

Pour ce qui est de l'apprentissage semi-supervisé par renforcement, il existe déjà aussi quelques articles.

**TORCS : The Open Racing Car Simulator** C'est un simulateur de course de voiture multi plate-forme. Il peut être utilisé comme un jeu de voiture ordinaire, ou comme plateforme de recherche en intelligence artificielle car il permet de définir facilement des robots qui conduiront les voitures. Son code source est sous licence GPL. TORCS [2001]



### 1.4 Contraintes techniques

- Environnement de développement :
  - Plate-forme : PC
  - OS : Linux
  - Langage : C++, C
  - Outils : Doxygen, CodeBlocks, GCC, KDevelop, Cmake, Make
  - Simulateur : TORCS
  - Bibliothèques : Dépendances de TORCS
- Contraintes matérielles requises pour faire fonctionner le système :
  - 1GHz CPU
  - 512MB RAM
  - OpenGL 1.3 compatible graphics card with 64 MB RAM
- Contraintes matérielles : toutes les fonctionnalités du système doivent pouvoir fonctionner sans ralentissement dans un ordinateur de dernière génération :
  - Quad Core CPU
  - 8 GB RAM

## 2 Description de la solution demandée

De nombreux termes et concepts utilisés dans cette section se réfèrent au document : PDMIA [2008]. Il est nécessaire d'être familier avec les bases des MDP ( Markov Decision Process ) et de celles de l'apprentissage par renforcement pour comprendre ce qui suit.

### 2.1 Fonctionnalités bibliothèque

**Algorithmes** La bibliothèque sera composée de 2 algorithmes génériques, basés sur le critère de performance  $\gamma$ -pondéré à horizon infini :

- Q-learning
- Sarsa

Ils pourront être utilisés dans n'importe quel domaine d'application.

**Fonction de récompenses - semi supervisée** Il sera possible d'intégrer les retours d'un tuteur dans la fonction de récompense. En ajoutant/retirant une constante ou en multipliant la récompense par un facteur.

**Approximation de fonction** Lorsqu'il y a une explosion combinatoire trop importante dans un domaine, la bibliothèque fournira des solutions par approximation de fonction.

### 2.2 Fonctionnalités simulateur

**Robots** Les utilisateurs pourront visualiser les robots conduirent au terme ou durant leur apprentissage. Il sera possible de visualiser différents algorithmes :

- Q-learning
- Sarsa

**Mesure de performance** Différentes mesures de performance pour les agents seront disponibles tel que : le nombre de récompenses additionnées, la durée du temps de sortie de route, dommages perçus par le véhicule

**Fonction de récompenses** Il sera également possible de choisir la fonction de récompense tel que le fait de rester au milieu de la route, la distance parcourue

**Réglage des paramètres** Les paramètres tel que le taux d'apprentissage, le réglage entre exploitation et exploration, le taux de discrétisation des états/actions seront modifiables.

**Interface retour expert** Dans le cadre de l'apprentissage semi-supervisé, l'interface de TORCS sera modifiée pour permettre à un tuteur de donner des retours positifs ou négatifs durant l'apprentissage de l'agent.

### 2.3 Contraintes de réalisation

- Il est primordial que les algorithmes soient **facilement interchangeables** et **réutilisables** en dehors de TORCS : il doit donc y avoir une nette distinction entre le code de la librairie et l'interfacement avec TORCS.
- Durant la phase de conception et de développement, les documents et le code devront être maintenu en ligne (dans une forge type github) pour mettre aux endacrants de facilement suivre l'évolution du projet.
- Les fonctionnalités citées précédemment constituent le **noyau central** du projet et devront être à réaliser absolument. Les fonctionnalités à venir sont facultatives et leurs réalisations dépendront de l'avancement du projet.

### 2.4 Fonctionnalités additionnelles bibliothèque

**Algorithmes** Un algorithme de **différence temporelle** pourra être ajouté.

**Fonction de récompenses - semi supervisée** L'expert pourra intégrer des feedbacks plus complexes tel que la prochaine action à choisir ou le renseigner sur sa stratégie exploration/exploitation.

### 2.5 Fonctionnalités additionnelles simulateur

**Robots** L'algorithme de **différence temporelle** sera également disponible.

**Interface retour expert** L'interface de TORCS sera modifiée avec toute une partie indépendante pour permettre au tuteur de voir les détails de l'exécution de l'algorithme et intégrer des retours plus complexes (choix d'action, stratégie exploitation/exploitation).

**Course multijoueur** Il sera également possible d'aborder les courses à plusieurs conducteurs. En ce sens, on pourra alors fournir de nouvelles fonctions de récompense et mesures de performance tel que la position dans la course.

### 3 Composition du livrable

Dans le cadre de la réalisation du projet, nous nous engageons à livrer les produits suivants :

- La bibliothèque implantant toutes les fonctionnalités prévues
- Le simulateur modifié de TORCS avec :
  - Les robots implémentant les algorithmes d'apprentissage
  - Les modifications de l'interface pour intégrer les retours de l'expert
- Le code source entièrement commentés ( hormis celui de TORCS que nous n'aurons pas modifié )
- Un ensemble de script facilitant la compilation et l'installation
- La documentation complète du projet à savoir :
  - Cahier des charges
  - Plan de développement
  - Analyse et conception
  - Manuel d'installation
  - Manuel de l'utilisateur
  - Manuel du programmeur
  - Rapport d'expériences décrivant les résultats complets en fonction des différentes routes, algorithmes et critères
  - Rapport final

### Références

- PDMIA, G. (2008). *Processus Décisionnels de Markov en intelligence artificielle*. Laboratoire Informatique Paris 6, Equipe MAIA - INRIA (LORIA).
- Richard S. Sutton, A. G. B. (1998). *Reinforcement Learning : An Introduction*. MIT Press, Cambridge, MA.
- TORCS, T. (2001). The open racing car simulator. <http://torcs.sourceforge.net/>.
- Wikipédia (2012). Apprentissage par renforcement. [http://fr.wikipedia.org/wiki/Apprentissage\\_par\\_reinforcement](http://fr.wikipedia.org/wiki/Apprentissage_par_reinforcement).
- Wikipédia (2013). Semi-supervised learning. [http://en.wikipedia.org/wiki/Semi-supervised\\_learning](http://en.wikipedia.org/wiki/Semi-supervised_learning).