Rapport MIF24

Aurélien CHEMIER, Arnaud DUHAMEL, Maëlyss FARGES

2013/2014

Table des matières

1 Introduction

L'objectif de ce projet est d'implémenter un agent qui apprend à effectuer les interactions positives sans connaître à priori son système motivationnel (mot_1 ou mot_2) ni son environnement (env_1 ou env_2).

- Deux expériences sont possibles $E = \{e_1, e_2\}$
- Deux résultats sont possibles $R = \{r_1, r_2\}$
- Il y a donc quatre interactions possibles : $E \times R = \{i_{11}, i_{12}, i_{21}, i_{22}\}$

Pour coder l'agent nous avons utilisé le langage C++ et Qt.

2 Agent 1

2.1 Description de l'agent

Les environnements sont :

- $env_1 : e_1 \Rightarrow r_1, e_2 \Rightarrow r_2$ (i_{12} et i_{21} ne se produisent jamais).
- $env_2 : e_1 \Rightarrow r_2, e_2 \Rightarrow r_1$ (i_{11} et i_{22} ne se produisent jamais).

Les Systèmes motivationnels sont :

- $mot_1 : v(i_{11}) = v(i_{12}) = 1, v(i_{21}) = v(i_{22}) = -1$
- $mot_2 : v(i_{11}) = v(i_{12}) = -1, v(i_{21}) = v(i_{22}) = 1$

Concrètement, deux interactions donnent un résultat positif et deux autres donne un résultat positif.

2.2 Attributs

Listing 1 – Attributs de la classe Agent

```
SystemeMotivationnel m_motivation; // l ensemble des motivations du systeme.
int m_motivationScore; //correspondont a la somme des resultats de
                       //toutes les experiences passees.
QList<Interaction> m_trace; //la liste de toutes les interactions
                            //effectuees par l'agent

const Environnement& m_environnement; //l'environnement ou se situe l'agent
QMap<int, Experience> m_exp;
```

m_motivation correspond au système de motivation de l'Agent tel qu'il a été décrit plus haut.

m_motivationScore est la somme cumulée de tous les résultats d'expériences.

m_trace récupère toutes les effectuées par l'agent.

m_environnement est l'environnement dans lequel l'Agent évolue.

m_exp stocke les différentes expériences faite par l'Agent.

2.3 Constructeur\Destructeur

Listing 2 – Constructeur et destructeur de l'Agent

```
/**
 * @brief Agent::Agent
 * @param e
 */
Agent::Agent(const Environnement& e): m_environnement(e), m_motivationScore(0)
{
    QFile file("trace.txt");
    if(!file.open(QIODevice::WriteOnly))
    {
        qDebug()<<"Ouverture du fichier \"trace.txt\" impossible";
        return;
    }
    file.close();
}

/**
 * @brief Agent::~Agent
 */
Agent::~Agent()
{
}
```

Le constructeur initialise l'environnement avec l'environnement passé en paramètre et le score de motivation à 0. Il contrôle également le fichier trace.txt.

Le Destructeur est vide.

2.4 save()

Listing 3 – Constructeur et destructeur de l'Agent

```
void Agent::save()
{
    QFile file("trace.txt");
    if(!file.open(QIODevice::Append))
    {
        qDebug()<<"Ouverture du fichier \"trace.txt\" impossible";
        return;
    }

    const Interaction& it = m_trace.last();

    QString toWrite;
    toWrite = "Experience: " + QString::number(it.experience().num()) + ", ";
    toWrite += "Resultat: " + QString::number(it.resultat().num()) + ", ";
    toWrite += "Motivation: " + QString::number(it.motivation())
    + ", Total = " + QString::number(m_motivationScore) + "\n";

    file.write(toWrite.toLatin1());


    file.close();
}
```

La fonction save() ajoute la dernière experience de l'agent dans le fichier trace.txt.