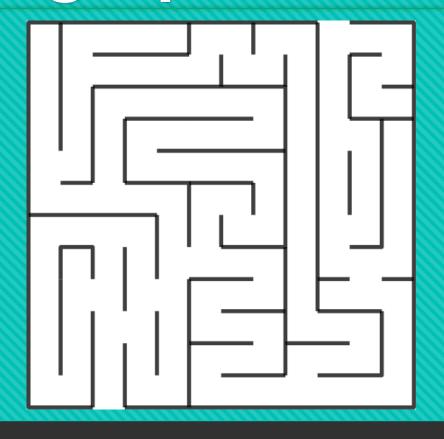
Apprentissage par renforcement



Le labyrinthe

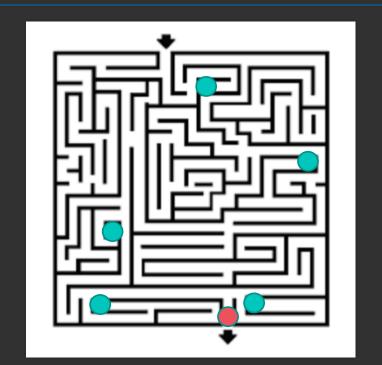
I. Contexte et ProblématiqueII. Solution : architecture & designIII. Vie du projetIV. PerformancesV. Conclusion

Sommaire

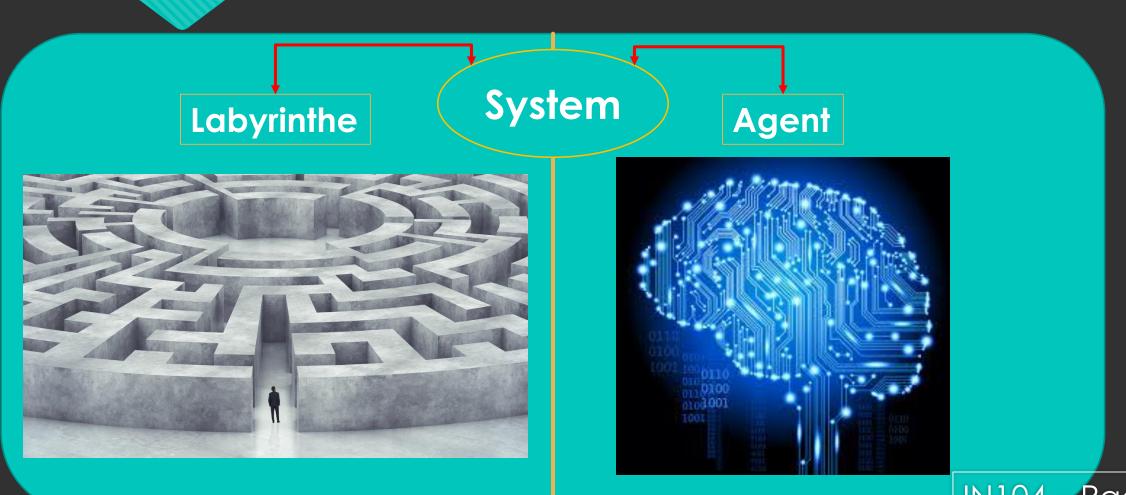
L. Contexte et Problématique

Trouver le chemin optimal pour résoudre le labyrinthe

Exemple
Récompenses
5 points
500 points



II. Solution: architecture et design



II. Solution: architecture et design

LABYRINTHE

state

<u>name_index</u> : Indice du nom de l'état (travail avec des nombres) Reward() : renvoie le reward

Reward(): renvoie le <u>reward</u> associé à l'état → Sert à calculer lors d'un déplacement dans le labyrinthe

ENV

Dimensions : largeur, longueur

Etats : état de chaque position (départ, mur, trou, arrivée) Position courante de l'agent dans le labyrinthe

- Show() : affichage du labyrinthe → sert à visualiser le labyrinthe
- <u>CreateRandomEnvironment()</u>: modification des états du labyrinthe par des états quasi-aléatoires → sert à tester les agents
- CreateExistingEnvironment(« file »): modification des états du labyrinthe par des états donnés connus → sert à tester les agents sur une carte connue
- runStep(action): modification de la position courante du labyrinthe, renvoi du reward engendré par le nouvel état
 Sert à faire évoluer le labyrinthe et calculer le reward

II. Solution: architecture et design

AGENTS

AgentRandom

Pas d'arguments

nextAction(laby): trouve l'action suivante de façon aléatoire → C'est ce qui permet à l'agent d'avancer

AgentQLearning

Q : matrice de Qualité

Epsilon : paramètre de <u>random</u> Lambda : facteur d'apprentissage Gamma : coefficient de réduction

- nextAction(laby): renvoie l'action que l'agent doit effectuer dans l'état courant: random ou qualité maximale

 C'est ce qui permet à l'agent d'avancer
- ChangeParameters(reward,laby,action, dernière position):
 change les paramètres de l'agent en fonction de l'action
 effectuée depuis la dernière position et le reward
 engrangé → C'est ce qui permet à l'agent d'apprendre

System

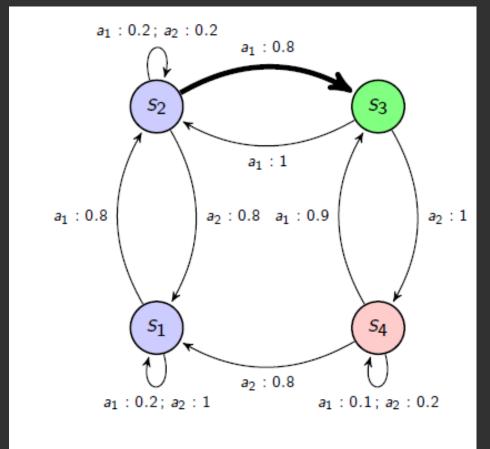
<u>Laby</u>: labyrinthe (classe ENV) Agent: un agent choisi

runEpisode(maxActionCount) : fais marcher dans le labyrinthe jusqu'à l'arrivée OU la fin du décompte et renvoie le reward total → Sert à faire apprendre l'agent de façon contrôlée et à analyser ses résultats

Parenthèse – Le Q-Learning

- à chaque arrivée sur un état, on note l'action ayant menée à cette état en renforçant la note existante (l'utilité/Qualité) avec une partie de l'utilité du noeud.
- Arrivé sur l'état s' à partir de l'état s et de l'action a, on note la Qualité de l'arc (s, a, s'):

$$Q(s,a) = \lambda \times (r + \gamma \times max_{a'}(Q(s',a')) + (1 - \lambda) \times Q(s,a)$$



III. Vie du projet

1^{ère} partie du projet

- Classe Environnement 2D
 - O Modélisation d'un labyrinthe aléatoire et de ses différents états / Visualisation graphique
 - Méthode « possibleActions » essentielle pour l'agent et évolution du labyrinthe
- O Tests / Méthode agile
 - Tests unitaires de l'environnement et tests globaux de modélisation du labyrinthe
 - O Classe Agent aléatoire

III. Vie du projet

2^{ème} partie du projet

- Classe Agent Q-Learning
 - Matrice de Qualité et son évolution
- Tests
 - O Tests unitaires de l'agent et tests globaux appliqués au labyrinthe
 - Méthode pour créer un labyrinthe lu dans un fichier

III. Vie du projet

Expérience Git

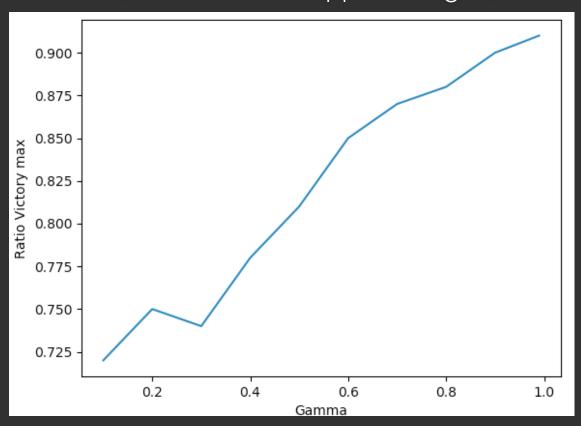
O Utilisation d'Atom et du package « Git Plus »

Interface graphique pour interagir avec Git

- O « Git add + commit + push » simplifié
- O Conflits facilement résolus et fusion instantanée
- O Avantages
 - O Travail en parallèle sur le même projet (potentiellement même fichier)
 - O Fonctionnalités très pratiques et productives pour un tel projet (+1000 lignes de code)

IV. Performance

Influence du facteur d'apprentissage Gamma



Conditions des tests

Labyrinthe de taille 10x10

Nombre d'épisodes :1000

Nombre d'actions par épisode : 1000

Ratio Victory: Nombre de victoires sur Nombre d'épisodes

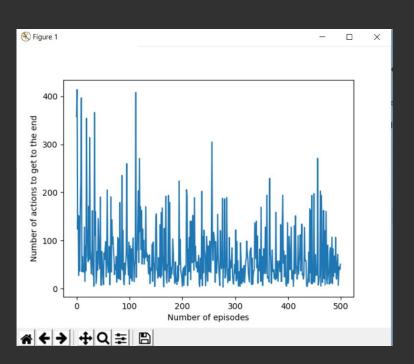
Gamma variant de 0,1 à 0,99 (pas de 0,1 ou 0,09)

Pour un même labyrinthe:

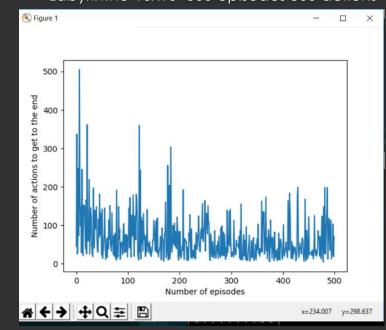
Moyenne sur 10 valeurs de Ratio pour chaque Gamma

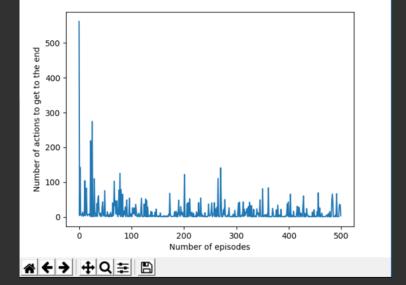
IV. Performance

O Nombre de déplacements pour sortir du labyrinthe en fonction du nombre d'épisodes



Labyrinthe 10x10 500 épisodes 500 actions





Gamma: 0.1

Moyenne d'actions nécessaires :

Gamma: 0.4

50

Moyenne d'actions nécessaires :

Gamma: 0.8

K Figure 1

Moyenne d'actions nécessaires :

1

IV. Performance

- O Défauts et améliorations possibles
 - Manque de puissance de calculs : Complexité spatiale et temporelle en O(n³) en le nombre de cases du labyrinthe
 - O Structure et architecture de l'environnement et de l'agent créés par nous-mêmes Non nécessairement optimal
 - O Défauts d'un labyrinthe aléatoire
 - O Chemin de l'entrée vers la sortie potentiellement bloquée par des murs
 - O Entrée juste à côté de la sortie
 - Mise à jour de la matrice de Qualité uniquement dans un rayon de deux cases de la position courante

V. Conclusion

- Modélisation et implémentation d'un projet conséquent
 - Recherche d'informations et état de l'art à effectuer avant de pouvoir commencer
- Expérimentation du logiciel Atom et de ses fonctionnalités Git
 - Outils optimisés permettant de gagner du temps et de l'efficacité
- Expérience du travail de groupe pour des projets futurs
 - Organisation du projet et division des tâches