IA04 Projet

Dilemme du prisonnier itéré

Équipe : (Groupe - TD1 Lundi 10:15-13:15 PRJ L3)

Liuqing HU

Songyang WANG

Jiemin YE

Qiaodan SHEN

Enseigneurs :

Khaled Belahcene

Sylvain Lagrue

Domitile Lourdeaux

Table de Matière

Table de Matière **1**

Présentation du sujet **2**
Sujet et Enjeux 2

Choix de conception multi-agent **2**
Agent Pool 2
Agent Prisonnier 2

Choix technologiques **3**
Frontend 3
La transmission de site Web 3
Communication entre les agents 4

Forces et faiblesses du projet **5**
Forces 5
Faiblesses 6

Conclusion personnelle et retours sur l'UV **6**

Bibliothèque **7**

Présentation du sujet

Sujet et Enjeux

Si deux joueurs jouent au dilemme du prisonnier plus d'une fois de suite et qu'ils se souviennent des actions précédentes de leur adversaire et modifient leur stratégie en conséquence, le jeu est appelé dilemme du prisonnier itéré.

Dans un dilemme du prisonnier à un seul tour, il n'est pas surprenant que les gens agissent généralement dans leur propre intérêt et se trahissent mutuellement. Dans un dilemme du prisonnier itéré, les gens sont généralement plus susceptibles de se comporter de manière moins égoïste, car la trahison peut être punie.

Dans la vie, cependant, il arrive qu'il y ait plus d'un échange ou d'une coopération entre deux ou plusieurs agents (par exemple, parmi les gens, les entreprises, voire les pays). Des exemples de dilemmes itératifs similaires du prisonnier peuvent être trouvés à la fois dans la société humaine et dans la nature.

Choix de conception multi-agent

Agent Pool

Il reçoit les actions des prisonniers à chaque tour et affiche leurs actions à l'écran à la fin de chaque tour. Il détermine, tout au début, un nombre aléatoire pour représenter le nombre de tours. Il sauvegarde les gains/résultats(la peine de prison totale) accumulés par chaque prisonnier. À la fin du jeu, le serveur affiche la durée moyenne de la peine de prison.

Agent Prisonnier

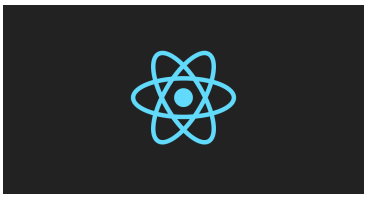
Nous avons déterminé 6 différents types d'agents ci-dessous :

- **Prisonnier muet (ID 1) :** Il choisit toujours "se tait"
- **Prisonnier méchant (ID 2) :** Il choisit toujours "trahir"
- **Prisonnier œil pour œil (ID 3) :** Si l'autre partie l'a trahi la dernière fois, il le trahira la prochaine fois ; si la dernière fois que l'autre partie a coopéré avec lui, la prochaine fois il coopérera
- **Prisonnier aléatoire (ID 4) :** Il choisit aléatoirement entre "trahir" et "se tait"
- **Prisonnier Intelligent (ID 5) :** Il choisit parmi l'histoire de choix de l'autre partie, avec une probabilité identique pour chaque choix
- **Prisonnier plus intelligent (ID 6) :** Il choisit parmi l'histoire de choix de l'autre partie, avec des probabilités différentes (les observations récentes ont plus de poids)

Choix technologiques

Frontend

ReactJS



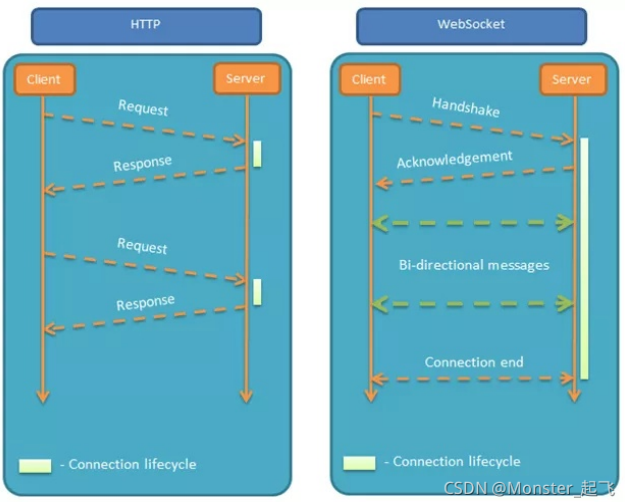
React est une bibliothèque JavaScript open-source qui est utilisée pour construire des interfaces utilisateur spécifiquement pour des applications d'une seule page. Elle est utilisée pour gérer la couche d'affichage des applications web et mobiles.
Avec des connaissances de HTML/CSS/JavaScript, ReactJS est simple à saisir tout de suite, c'est pourquoi on l'a choisi.

La transmission de site Web

Websocket

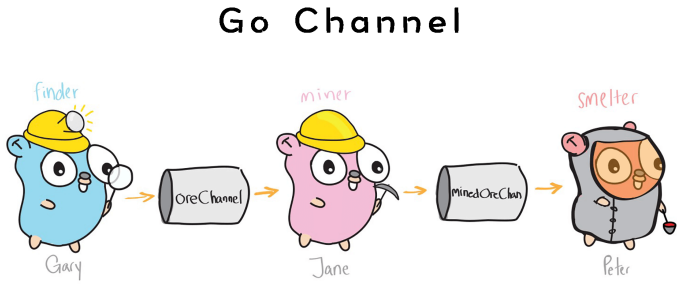
Le protocole WebSocket est un protocole réseau basé sur le protocole TCP. Celui-ci définit la façon dont les données sont échangées entre les réseaux. En raison de sa fiabilité et de son efficacité, il est utilisé par presque tous les clients. TCP établit une connexion entre deux **points finaux de communication** qu'on appelle des **sockets**. Ainsi, une communication bidirectionnelle s'établit entre les données.

Le protocole se compose d'une poignée de main au début et d'une trame de message de base ensuite, et est construit sur le protocole TCP. Contrairement au protocole HTTP, une fois qu'un lien Websocket a été établi, la communication en temps réel peut avoir lieu dans les deux sens.



Communication entre les agents

Channel



Dans notre projet, on utilise 3 channels. Voici est la structure Pool, quand on construit un agent Pool, on crée 3 channels, Register, Unregister et Choix.

```
type Pool struct {
    MaxTour     int
    Tour        int
    Gains       [2]notes
    Msgs        []CMPair
    Register    chan *Client
    Unregister chan *Client
    Clients     map[*Client]bool
    Choix       chan CMPair
    AgtIDName   map[int]string
}
```

Register : un channel pour les agents enregistrés
Unregister : un channel pour les agents non enregistrés
Choix : un channel pour la communication entre le serveur(ici Pool) et le client

Une fois que le client est créé, son pointeur est envoyé dans le channel ***Register***. Une fois que le client est fermé, son pointeur est envoyé dans le channel ***Unregister***. Le serveur Pool écoute le channel ***Choix*** pour obtenir les actions des deux clients. Chaque client envoie sa propre action dans le channel ***Choix*** chaque tour.

Pourquoi on choisit Channel?
Un canal fournit un mécanisme permettant à des fonctions s'exécutant simultanément de communiquer en envoyant et en recevant des valeurs d'un type d'élément spécifié. Lorsque vous avez plusieurs goroutines qui s'exécutent simultanément, les canaux constituent le moyen le plus simple de permettre aux goroutines de communiquer entre elles.

Forces et faiblesses du projet

Forces

- Interface graphique simple mais claire

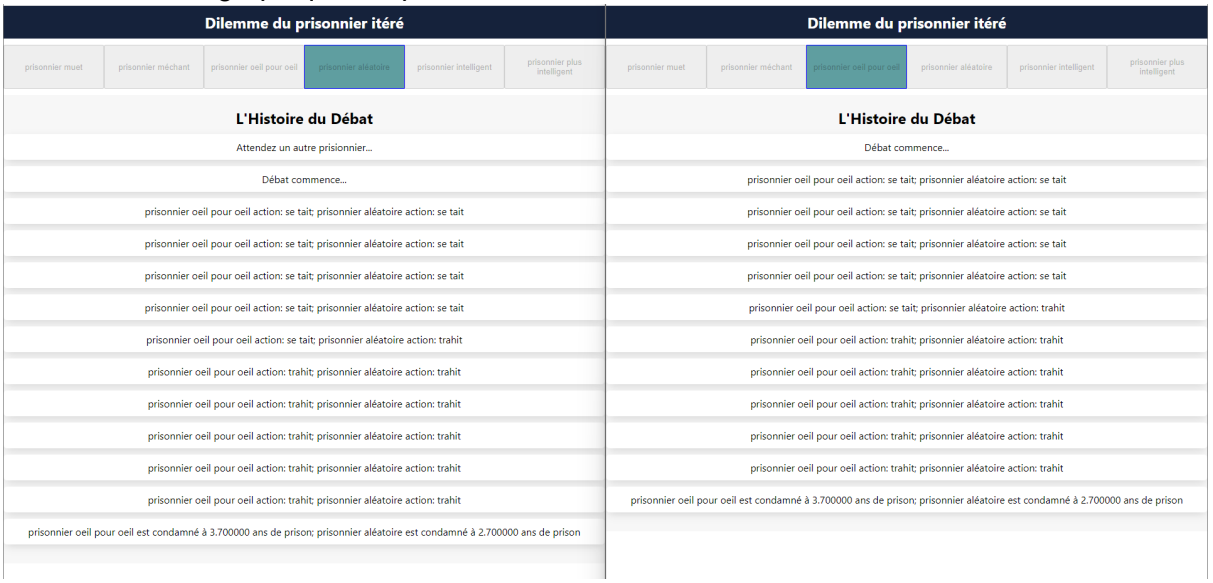


Figure : Débat entre Prisonnier aléatoire et Prisonnier œil pour œil

- Séparation du frontend et du backend
- Séparation de la fonction et du fichier

```
dilemma-de-prisonnier
 \-- backend
    \-- pkg
       \-- websocket
          \-- client.go
          \-- pool.go
          \-- websocket.go
    \-- go.mod
    \-- go.sum
    \-- main.go
 \-- frontend
     \-- node_modules
          \-- … …
     \-- public
          \-- … …
     \-- src
       \-- api
          \-- index.js
       \-- components
          \-- différents composants
     \-- … …
```

Faiblesses

- Le dilemme du prisonnier ne se conforme pas à l'équilibre de Nash en stratégie mixte, donc on n'a pas fait cette simulation
- L'algorithme de génération des poids pourrait encore être amélioré dans le cas d'enregistrement des choix précédents de l'adversaire afin de déterminer le moment de notre prochain choix.

Conclusion personnelle et retours sur l'UV

Liuqing HU

Ce projet me donne une très bonne opportunité de mieux comprendre la communication multi-agents dans la pratique, non seulement la communication via channel mais aussi la communication à travers le réseau, et de mettre en place une théorie des jeux typique de manière simple. J'ai développé les connaissances du protocole websocket et les compétences sur le ReactJS front-end framework.
En termes d'UV, en fait, avant de suivre cet UV, je n'ai pas connu le langage Go, cependant, je peux réaliser des projets simples en Go. J'ai beaucoup progressé sur la programmation en Go. J'ai bien appris les connaissances sur les différentes méthodes de votes, les théories des jeux et aussi les normalisations pour les mesurer comme la distance d'édition, distance discrète, l'équilibre de Nash, l'équilibre corrélé etc. Cependant, je n'ai pas pu parfois suivre la vitesse de parole pendant le cours. Une parole plus lente et plus d'écriture sur le tableau me conviendra. Il est un peu dur pour moi de prendre des notes en écoutant le professeur et en comprenant le cours. Je pense que cela sera mieux de pouvoir être accessible au slide après chaque cours de chaque semaine (avant le prochain cours).

Jiemin YE

Personnellement, je suis intéressée par Backend. Grâce à ce projet, j'ai appris à construire un serveur web en langage GO, et je connais mieux l'équilibre nash avec l'exemple de dilemme du prisonnier, en outre, j'ai appris un peu ReactJS.
Avec l'UV IA04, je connais mieux le langage Go et la conception multi-agent. Et j'ai appris la théorie des jeux etc. Cependant, IA04 est une UV difficile pour moi, puisqu'il y a beaucoup de concepts qui sont abstraits, et je n'ai pas pu trouver beaucoup d'informations sur ce que j'avais appris, si bien que l'auto-apprentissage après les cours est devenu difficile. D'ailleurs, je pense que ce serait mieux si le professeur mettait le PowerPoint plus tôt, puisque pour les étudiants étrangers, c'est important de préparer les cours. En général, j'aime bien IA04, puisque j'ai appris beaucoup.

Songyang WANG

Conclusion personnelle :
D'après ce projet, on a bien appliqué ce qu'on avait appris pendant les cours et TDs —— Conception des multi-agents, communication par channel, Modèle serveur-clients (centralisé), etc… Et afin de finaliser ce projet en complet, on a beaucoup étudié l'interaction entre le frontend et le backend (Websocket). Pendant les cours et TDs, on se concentre

plutôt sur l'algo du backend, donc il faut également qu'on apprenne des connaissances Front-end en utilisant l'encadrement ReactJS, ce qui est très utile au futur. En plus, le processus de la simulation des agents avec stratégies diverses nous permet de mieux comprendre la notion de la théorie des jeux.

Retours sur l'UV :
1. Le rythme est un peu rapide au tout début, on a dû maitriser le langage GO en 1/2 semaines.
2. Pour les étudiants qui n'ont pas choisi UV IA01-IA03(comme nous), il y a une possibilité potentielle d'avoir du mal à suivre parfaitement.
3. Bien sûr, tous les enseignant(e)s sont assez responsables et gentil(e)s, et vous êtes toujours disponibles pour nos questions et doutes.

Qiaodan SHEN

Conclusion personnelle :
D'après moi, ce projet a permis une amélioration globale de mes compétences de programmation en Golang. En particulier, la simulation des agents avec différentes stratégies du dilemme du prisonnier a renforcé ma compréhension des algorithmes de théorie des jeux. Dans la partie frontend, on a utilisé le cadre ReactJS accompagnant le service WebSocket du backend pour mettre en œuvre l'interaction front-end et backend, ce qui concerne aussi le traitement multi-clients. Il m'a également permis d'approfondir ma compréhension de la conception des multi-agents, la communication par channel, et le modèle centralisé.

Retour de l'UV :
Généralement, cette UV est vraiment intéressante du côté du contenu et je l'aime bien. Mais en fait, il me semble que c'est difficile de comprendre des concepts peu familiers et plutôt abstraits à temps, car on n'a pas le poly et en plus le rythme est assez vite. Parfois, je n'arrive pas à suivre le professeur en classe.

Bibliothèque

Dilemme du prisonnier
- https://fr.wikipedia.org/wiki/Dilemme_du_prisonnier

Dilemme du prisonnier itéré
- http://www.cril.univ-artois.fr/~konieczny/enseignement/DIP.pdf

ReactJS
- https://ibracilinks.com/blog/quest-ce-que-reactjs-et-pourquoi-devrions-nous-utiliser-reactjs

Websocket
- https://www.ionos.fr/digitalguide/sites-internet/developpement-web/quest-ce-que-le-websocket/