Projet IA04 – A24

Kilapang HUICHALAF, Elias BELLOUMI, Jules TAVERNE

Simulation GOjek Eat

Sommaire :

I.

Introduction et Problématique

Qu'est ce que Gojek?

Concurrent d'***Uber Eat*** connu en ***Asie du Sud***

Service de livraison de nourriture :

1. Propose des courses aux livreurs, libres de l'accepter ou non
2. Courses pour livrer la nourriture aux clients qui ont commandés sur l'application

Combien d'argent gagne Gojek en une journée ?

Ou plutôt est le ratio client/livreur qui génère le plus d’argent ?

II.

Modélisation de la simulation

Idée principale

Des agents qui maximisent leur gains, suivant leur propres objectifs

Font évoluer leurs croyances en fonction du temps

Sont forcés de collaborer un minimum

Fonctionnement des agents - GOJEK

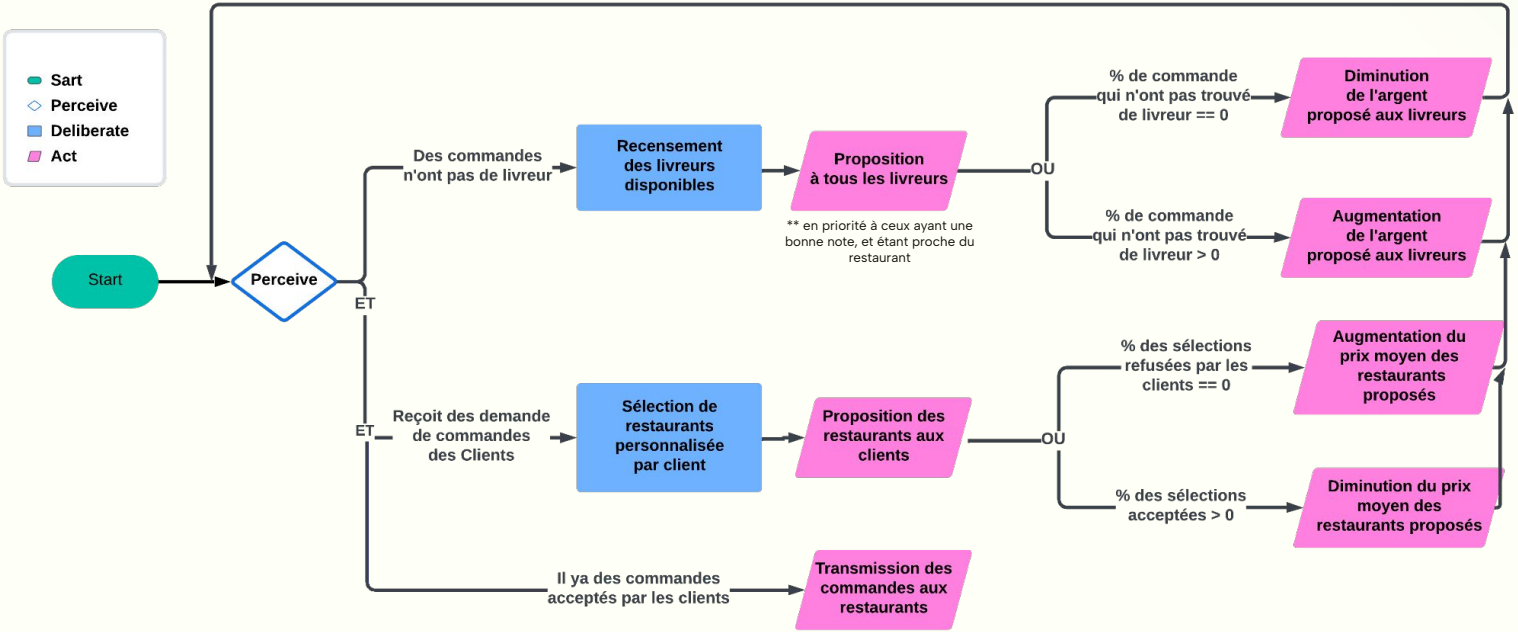
Son but : maximiser ses gains. Pour cela, il doit ***collaborer*** un minimum avec les autres agents, mais ses véritables intentions sont ***égoïstes.***

Maximiser le prix des restaurants proposés aux clients

Minimiser le prix à payer aux livreurs pour les livraisons

Constamment à la recherche du bon équilibre, faisant évoluer ses paramètres en fonction des refus des différents agents

Fonctionnement des agents - GOJEK



Fonctionnement des agents - Livreurs

Son but : Atteindre un objectif d'€ journalier

Croyance : le prix idéal pour une livraison est de ***3€ + 0.1€ tous les 100 mètres,***
De plus, le livreur n'accepte pas moins de **3€.**

On a un système d'enchère qui se met en place, où le livreur accepte le montant proposé par GOJEK, selon les paramètres suivants :

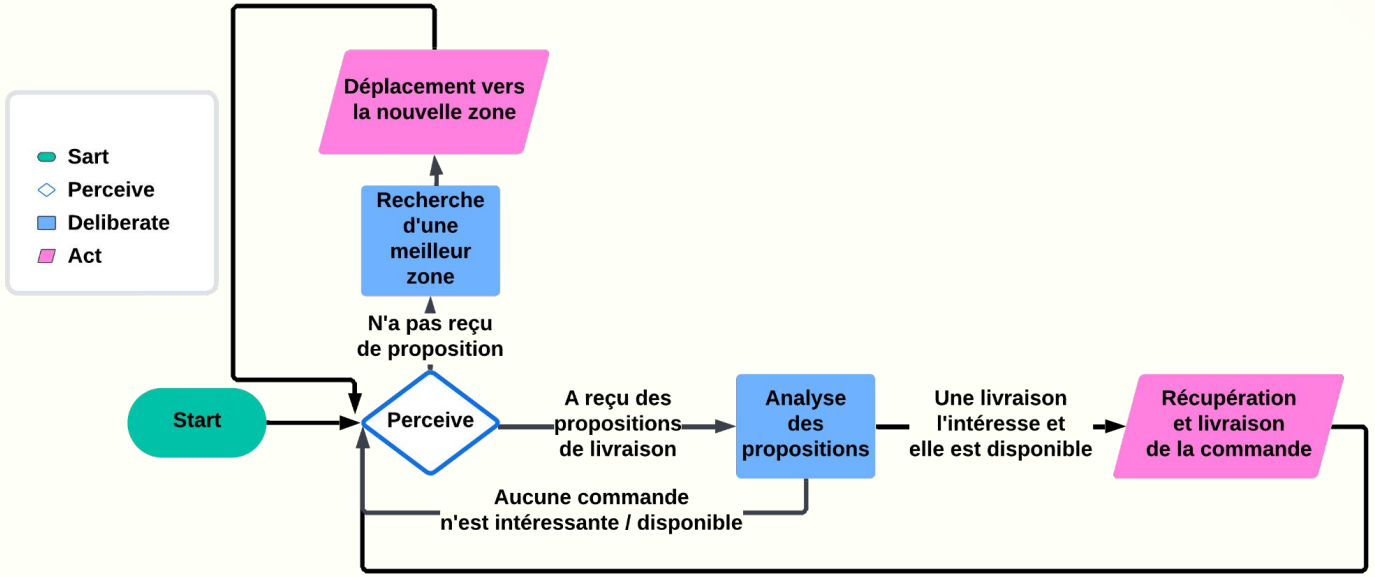
Le montant idéal qu'il estime devoir être payé pour cette livraison

Le temps restant dans la journée

L'argent qui lui reste à faire dans la journée,

Fonctionnement des agents - Livreurs

Son but : Atteindre un objectif d'€ journalier



Fonctionnement des agents - Clients

Son but : Ne plus avoir faim

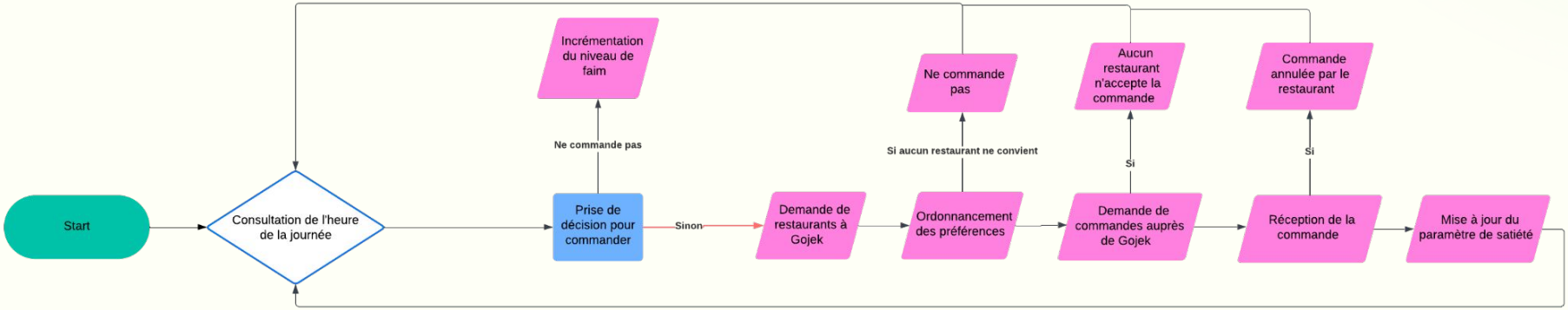
Croyance : Son niveau de faim. Les caractéristiques des restaurants proposés par Gojek

On a un système de maximisation qui se met en place, où le client va vouloir minimiser le prix payé et le temps d'attente:

Plus il a faim, plus il va vouloir un temps d'attente court

Fonctionnement des agents - Clients

Son but : Ne plus avoir faim



Fonctionnement des agents - Restaurant

Son but : Écouler son stock

Croyance : Son stock, le temps de préparation d'une commande

Le restaurant va accepter les commandes jusqu'à ce que:

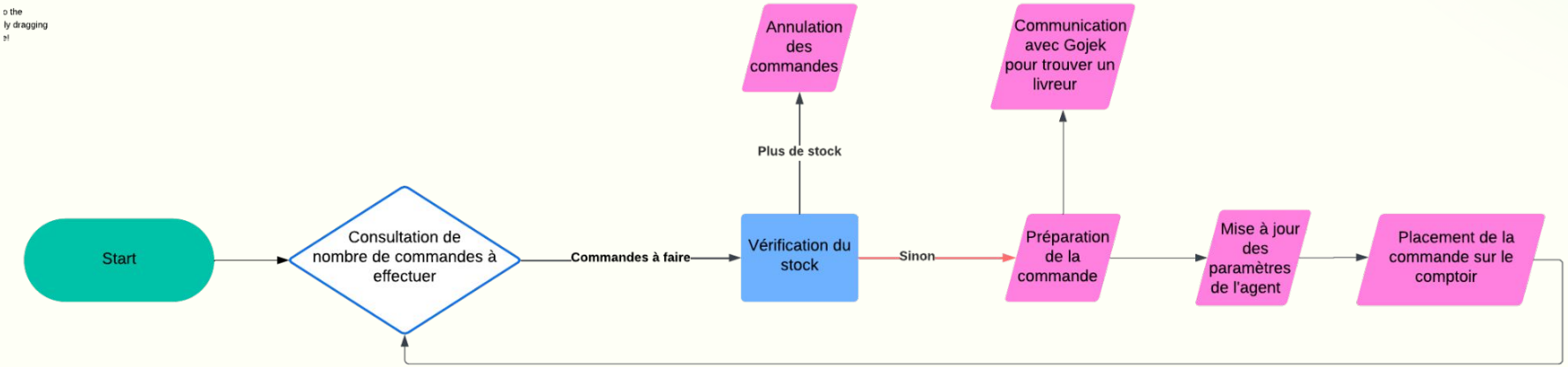
Le restaurant ait accepté suffisamment de commande pour produire jusqu'à la fermeture

Le restaurant est ouvert

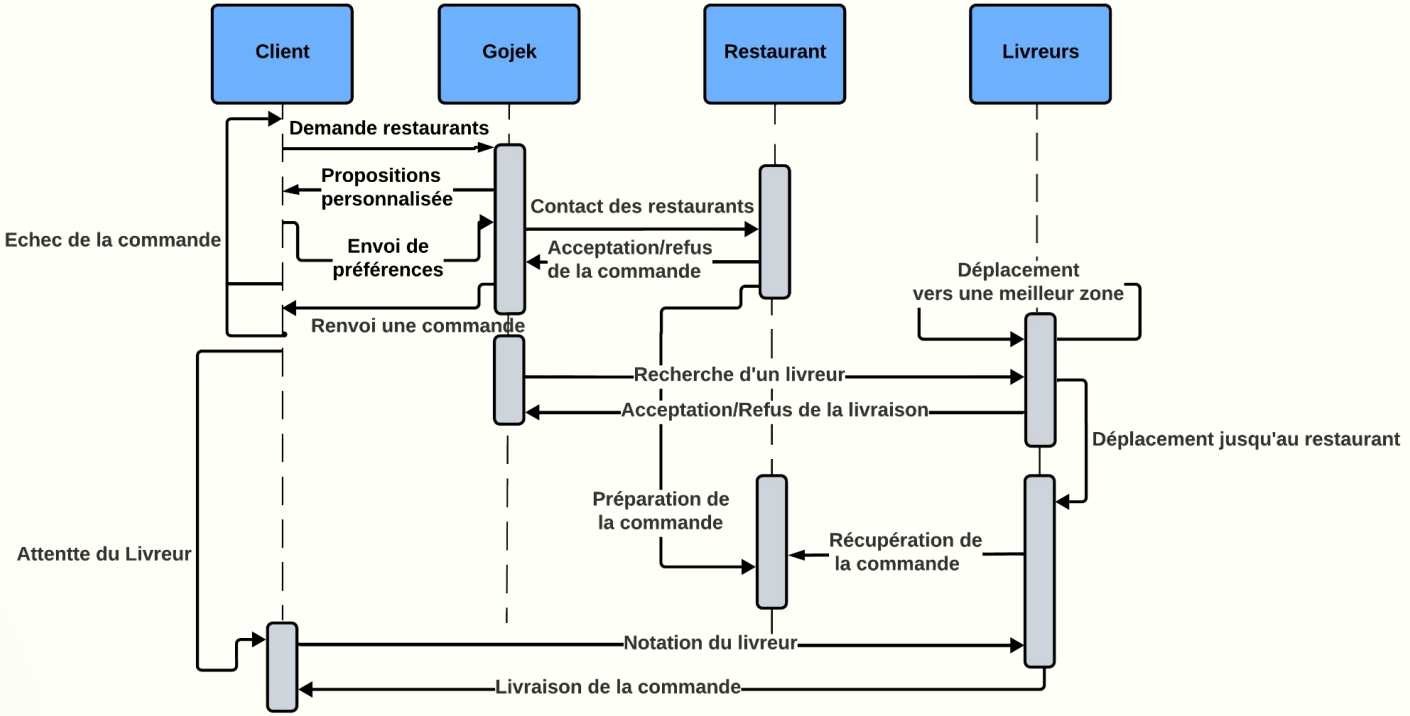
Le restaurant a du stock

Fonctionnement des agents -

Son but : Écouler son stock



Fonctionnement global de la simulation



III.

Démonstration

IV.

Implémentation

Backend :

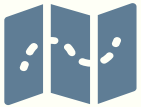
- Utilisation de la library ***Python OSMX*** et ***NetworkX*** pour obtenir la ville sous forme de graph ***JSON***

- Implémentation de l'interface Agent

- Nombreuses méthodes ***internes*** pour gérer les ***contraintes*** propres à la simulation
- Grande prise en compte de ***paramètres*** aléatoires pour rendre la simulation ***non déterministe***

Backend :

Chargement de la carte de ***JSON*** -> ***GO***, en utilisant des ***Struct***



Utilisation de la librairie ***GO*** “*gonum/graph/multi*”
Pour le calcule des itinéraire à l’aide d’un algorithme ***AStar***

Backend :

Différentes méthodes furent utilisées pour gérer l'accès concurrentiel aux ressources :

- ***sync.Map*** pour les ajouts simultanés à des ***map***
- ***Channel + goroutine*** pour les incrémentations, ou les cas de "*premier arrivé -> premier servi*"
- Utilisation de nombreuses IIFE pour la concurrence

Backend :

Les channels assurent un rôle essentiel:

- *Communication entre les agents*
- *Gestion concurrentielle des attributs*

Ce qui permet l'utilisation de go routine pour paralléliser la charge.

Frontend :

Communications avec le backend :

- Utilisation de websockets (rapide, envoi du back au front)
- Communications avec le format JSON

Frontend :

- Fait en javascript natif
- Mapbox pour la carte interactive (et les batiments 3D)
- Deck.gl fonctionne au dessus de Mapbox, permet d'ajouter des layers d'éléments graphiques, interactifs et 3D (utilise le GPU pour les performances)
- Charts.js pour l'affichage des graphiques en direct

V.

Conclusion

Réponse à la question

- Il ne faut pas plus de ***customer*** que de ***livreur*** : principe d'***offre*** et de la ***demande*** -> les livreurs deviennent ***exigeants***

- A l'inverse, ***trop de livreur*** implique une ***trop grande concurrence*** : traffic ***perturbé***

- De plus, avec un ***grand nombre*** de livreur, le ***runPriceRatio descend*** : les livreurs ne peuvent pas exiger trop d'argent si ils ont beaucoup de concurrence

- Sur une ville comme compiegne, notre simulation montre des ***limites*** : les distances sont trop petites, et toutes les commandes sont acceptés par les livreur -> plus optimisé pour les ***grandes villes,*** peu de variation de l'argent proposé aux Livreurs

Réponse à la question

Simulation Infos

Simulation Time: 4:00PM

Duration: 5 h

Acceleration: × 249

Number of Deliverer: 10000

Number of Restaurant: 66

Number of Customer: 50000

Gojeck Statistics

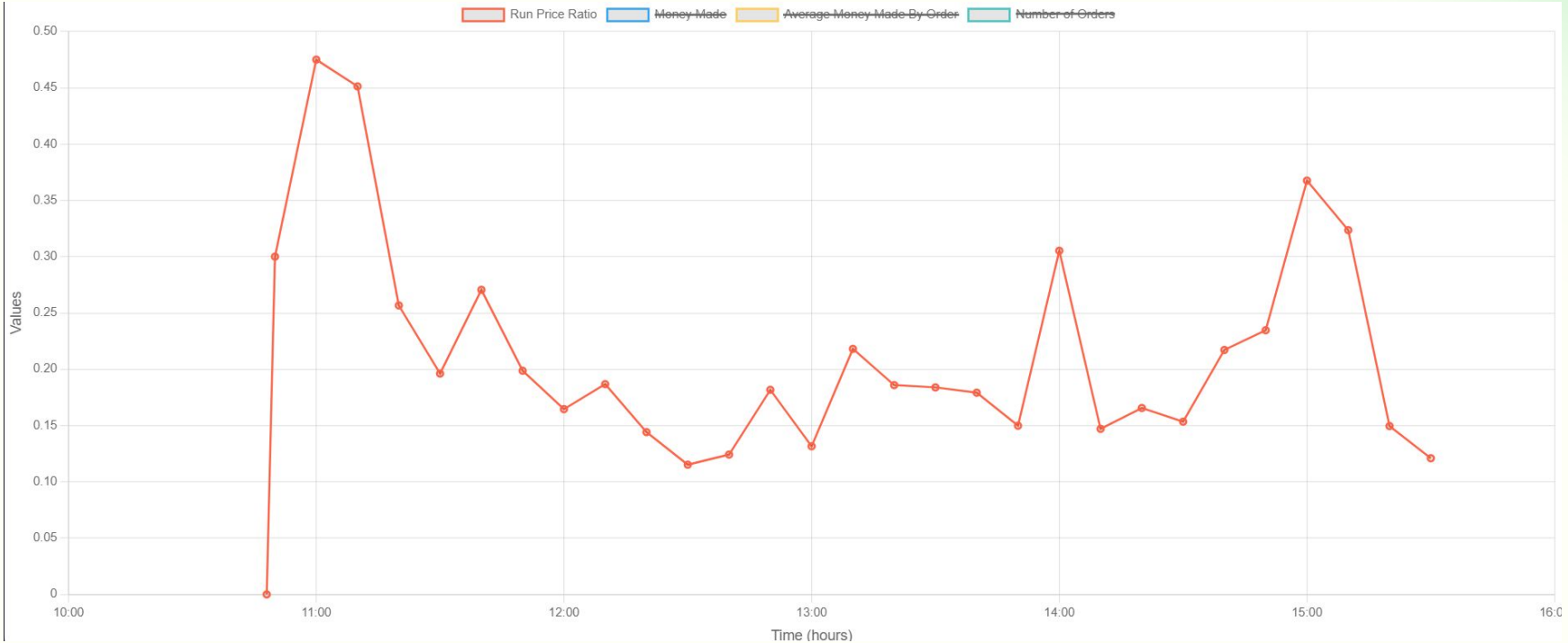
Run Price Ratio: 10.11 %

Money Made: 11061.18 €

Average Money Made By Order: 8.79 €

Number of Order: 1258

Réponse à la question



Limites de la simulation :

1 - Gestion ***simplifiée*** du trafic :

- On estime qu'un véhicule prend ~ ***3 mètre*** : si un rue mesure 4 mètre, il ne peut y avoir qu'un véhicule à la fois [...]
- Pas de véhicule autre que les livreurs

2 - ***Approximation*** de la manière dont GOJEK choisi les livreurs

3 - Le client ne tient ***pas*** compte du ***prix de la livraison*** lors de ses choix (bien que partiellement induits)

4 - Les objectifs des livreurs, le prix de restaurants, la faim selon l'heure de la journée [...] sont ***cohérents***, mais choisis ***arbitrairement, Par hypothèse***

Avantages

1 - Gestion de la concurrence

2 - Utilisation de Python pour obtenir des données réelles

3 - Interactions basées sur l'aléatoire

4 - Chaque agent suit véritablement ses objectifs

Merci de votre attention