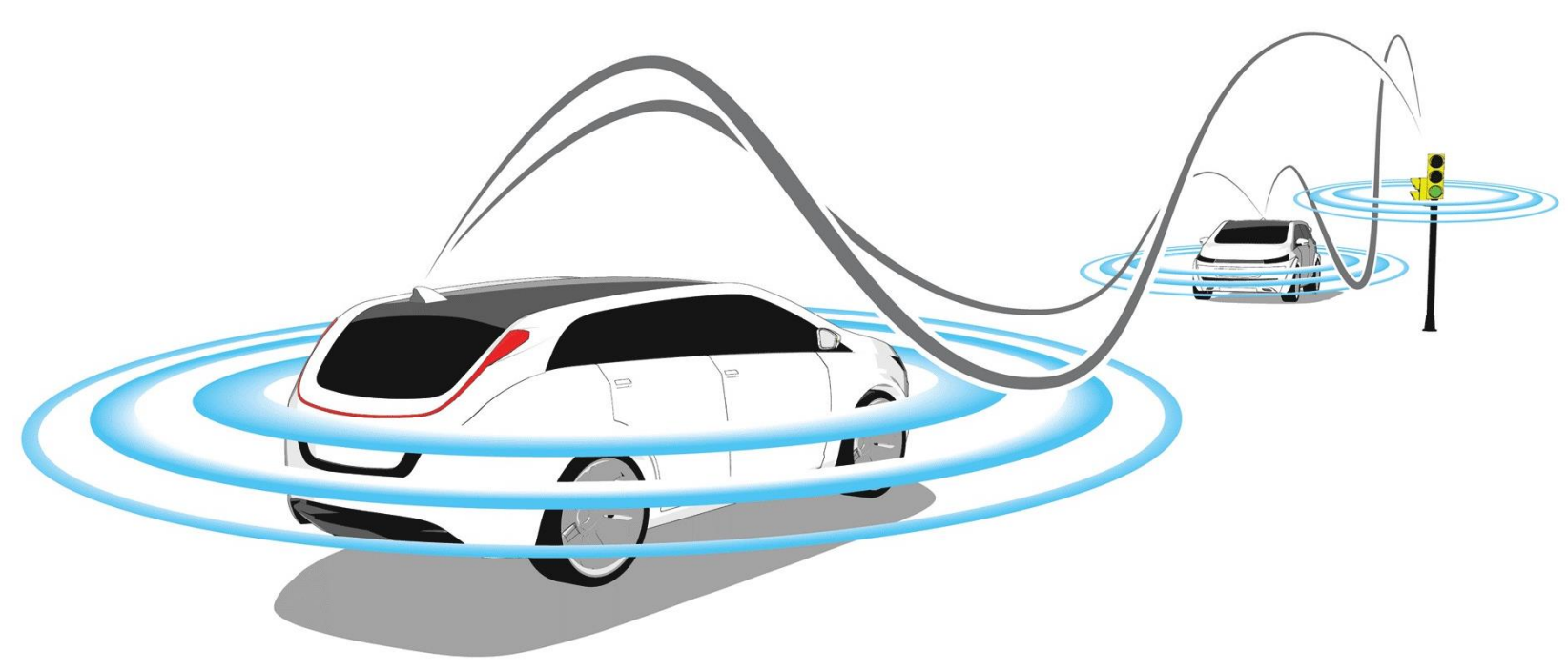
INGÉNIERIE DE RÉSEAUX

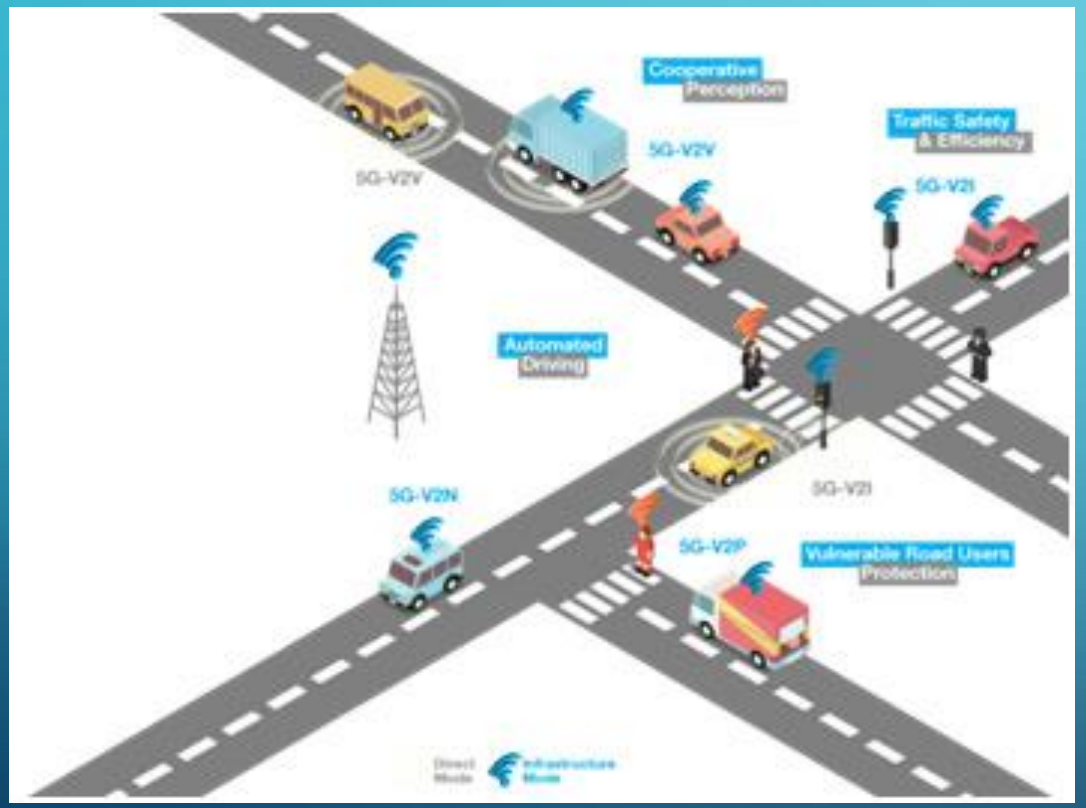
MULTI ARMED BANDIT (MAB) POUR UNE SÉLECTION DE COMMUNICATION DIRECTE (V2V) OU VIA INFRASTRUCTURE (V2I)



Lucas THIETART, Hugo LE CLAINCHE, Thomas GRUGET, Mathis SIGIER

PRÉSENTATION DU SUJET

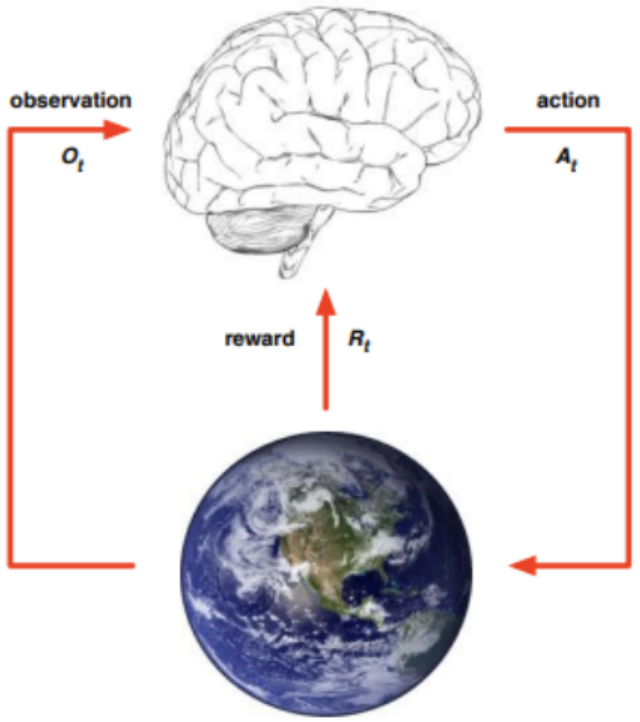
Amélioration de la sécurité des usagers vunlérables VRU



- Le déploiement de la 5g offre des possibilités sans précédent de signalisation et de communication
- La 5g permet la communication en temps réels des véhicules avec leur environnement – V2X
- Plusieurs mode de communication sont possibles : V2V (VANET) ou V2I réseau 5g
- Nécessité de faire un choix entre les deux modes cités en fonction des exigences en terme de QoS
- Les algorithmes Multi-Armed-Bandit (MAB) sont une option solide pour réaliser ces choix en fonction de la QoS

ÉTAT DE L'ART

Le principe du Multi-Armed Bandit (MAB)



At each step t,

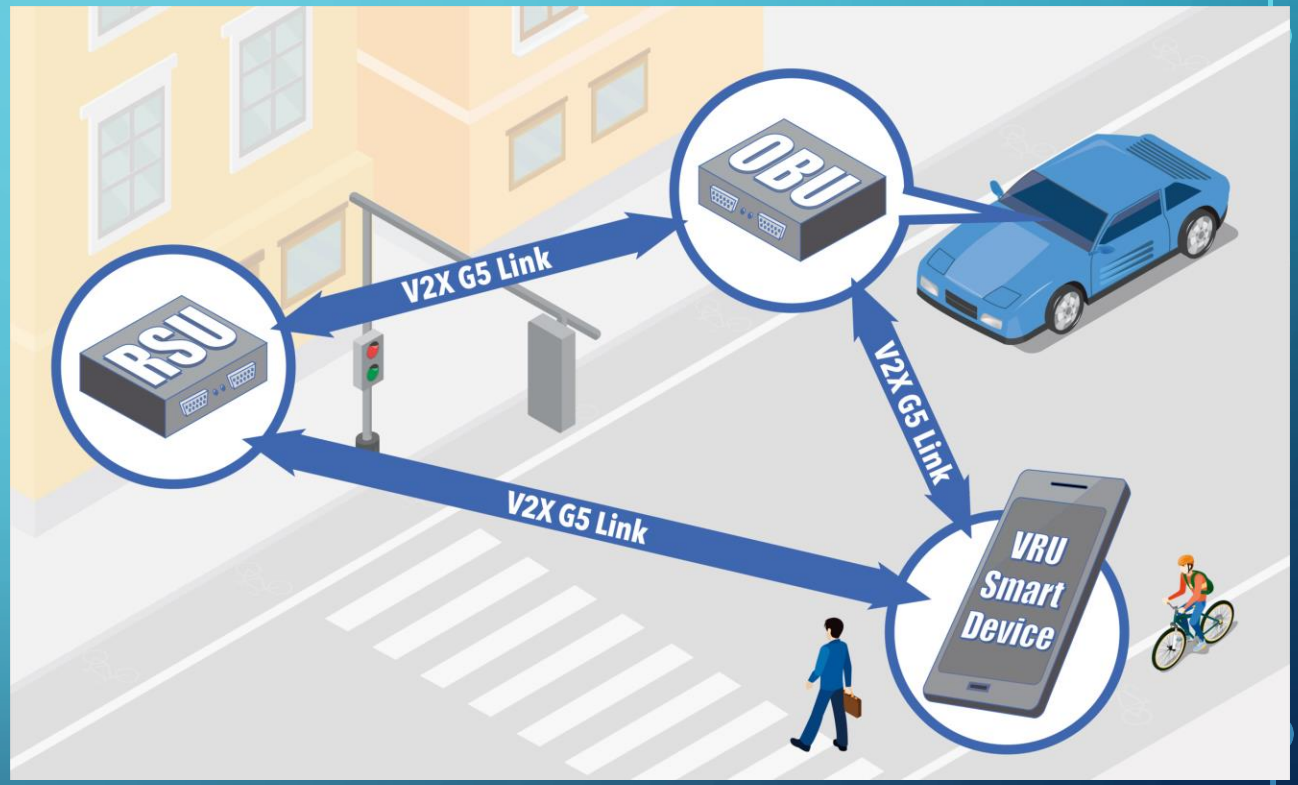
- the agent:
  - Executes action A_t
  - Receives observation O_t
  - Receives scalar reward R_t
- the environment:
  - Receives action A_t
  - Emits observation O_{t+1}
  - Emits scalar reward R_{t+1}
- t increments at environment step

Dans notre simulation :

- At = V2I ou V2V
- Rt dépend de la QoS
- (le reward et l'observation son confoncdus

LES CRITÈRES DE QUALITÉ DE SERVICE

- Contexte de la transmission de messages d'alerte pour la sécurité des usagers de la route
- Première discussion : Latence, taux de perte, état du réseau
- QoS retenue : Latence entre envoie et réception du paquet



Métrique principale retenue : **Le délai de transmission**

MODÉLISATION DU PROBLÈME ET SIMULATIONS

- Objectifs :
  - Optimiser la QoS à travers le choix V2V ou V2I en implémentant deux algorithmes MAB
  - Mesurer et comparer les performances et l'efficacité des MAB

MODÉLISATION VEHICULE TO VEHICULE

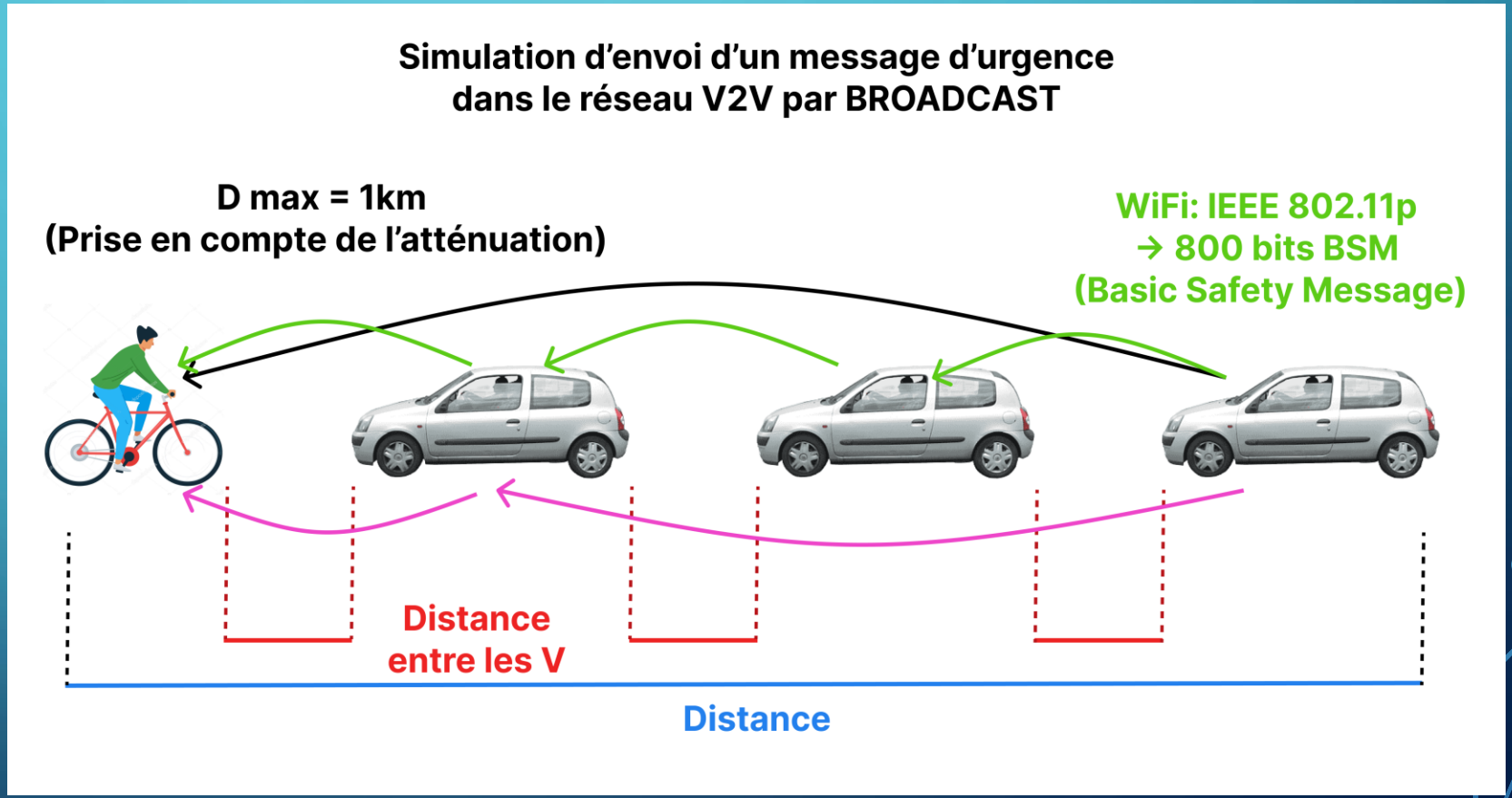
Modélisation basée sur les VANETs (Vehicular Ad Hoc Networks) :
QoS basé sur **distance E/R** et **densité de trafic dans la zone**

Définition des paramètres

Densité de trafic *Df*
=> Distance entre véhicules
Taille des paquets *Tp*
Débit de communication *R*

Constantes

Vitesse de propagation c
Longueur moyenne d'un véhicule *L*



MODÉLISATION VEHICULE TO INFRASTRUCTURE

Latence obtenue à partir des **distance E/R + distance VRU/Vehicle** et de la **charge sur le réseau cellulaire**

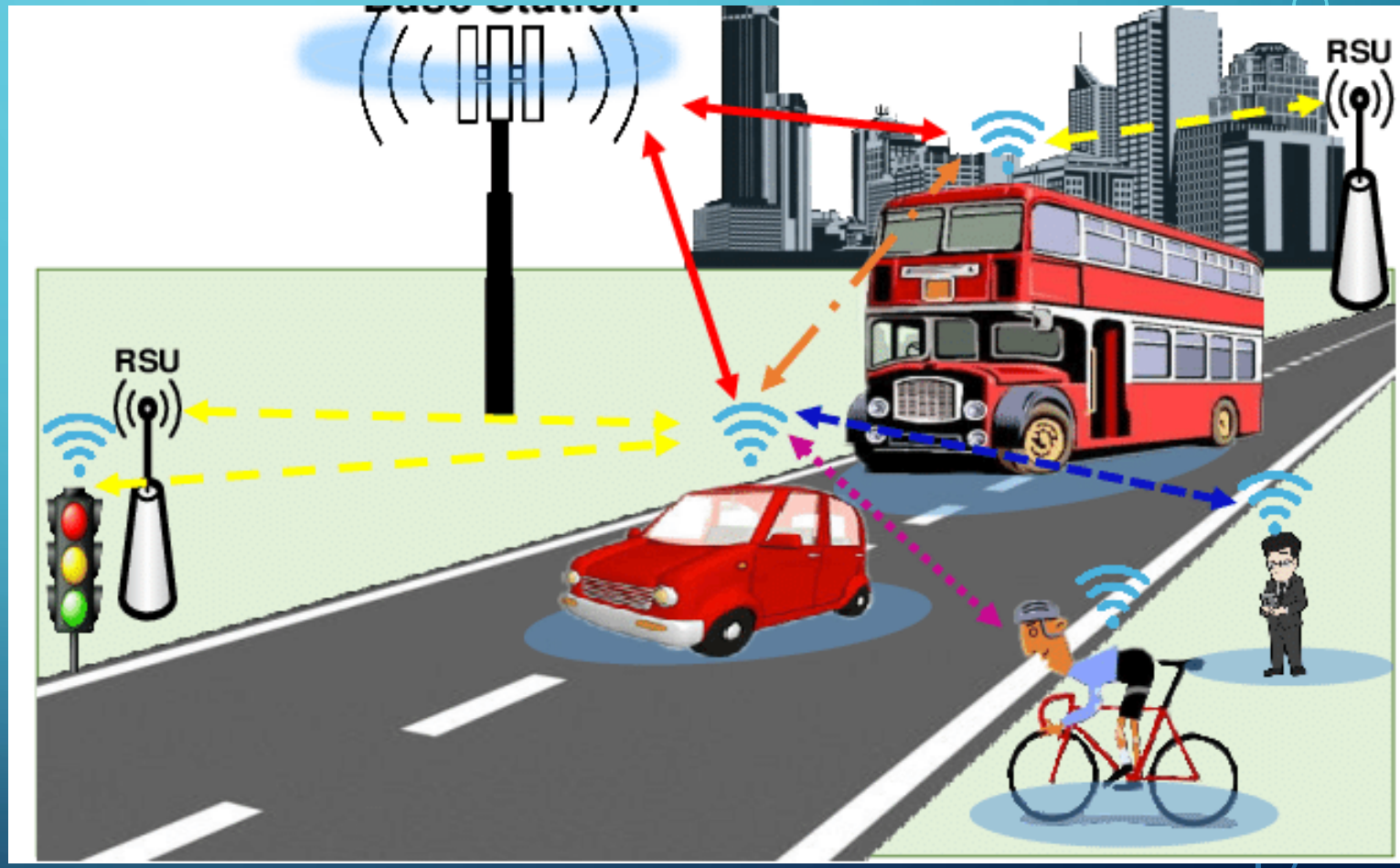
Approximations clés

Distance véhicule infrastructure
100 à 400 mètres

% de Charge sur le réseau

Débits de traitement et d'émission
Paquets de taille120o (basé sur geoMIP)

Temps de transmission au sein du réseau (dépend de la distance VRU/VEHICLE)



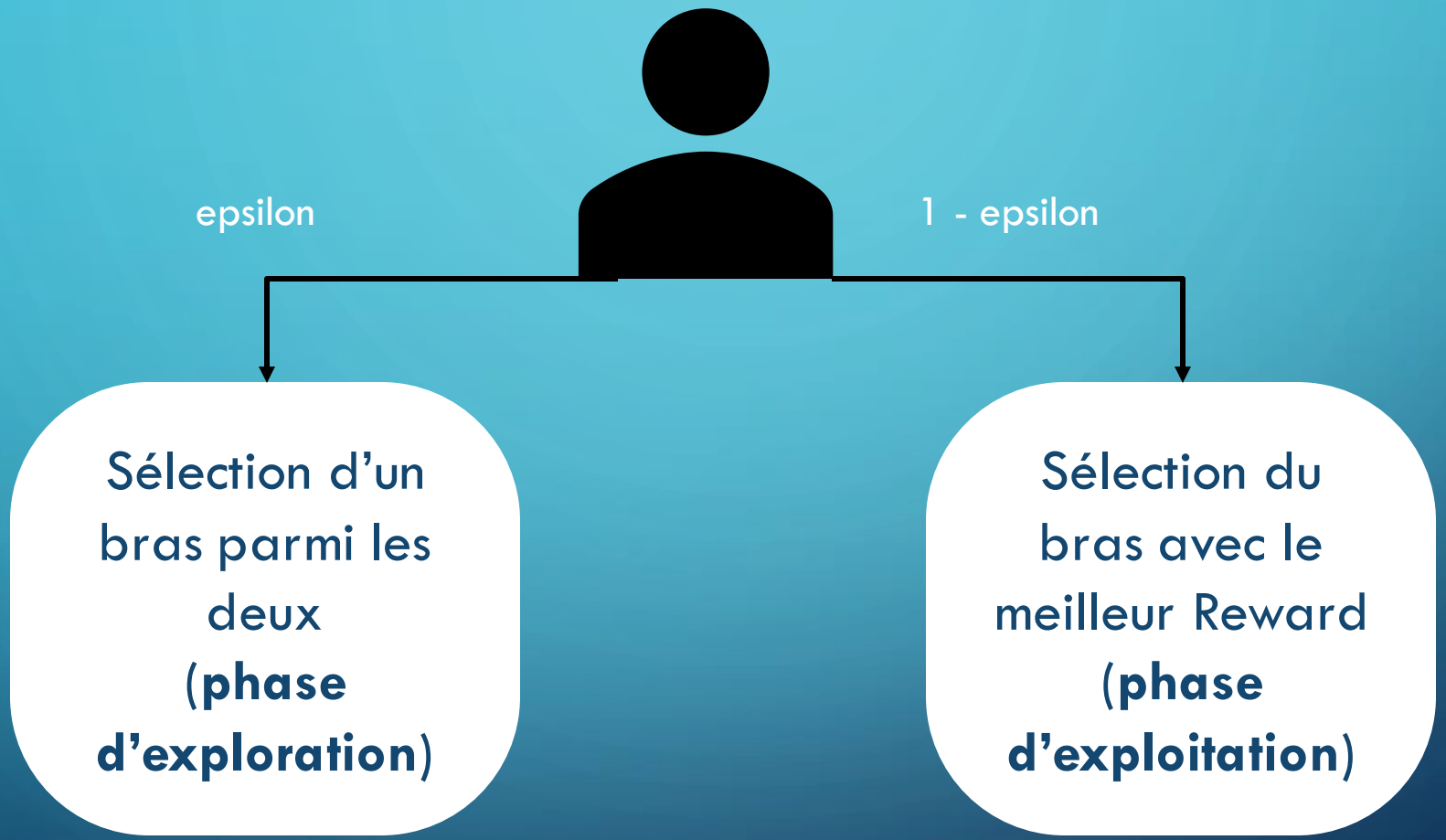
ALOGRITHME MAB

Deux versions de l'algorithme :

UCB et

Epsilon Greedy

ALOGRITHME EPSILON GREEDY



Epsilon est calculé à chaque itération

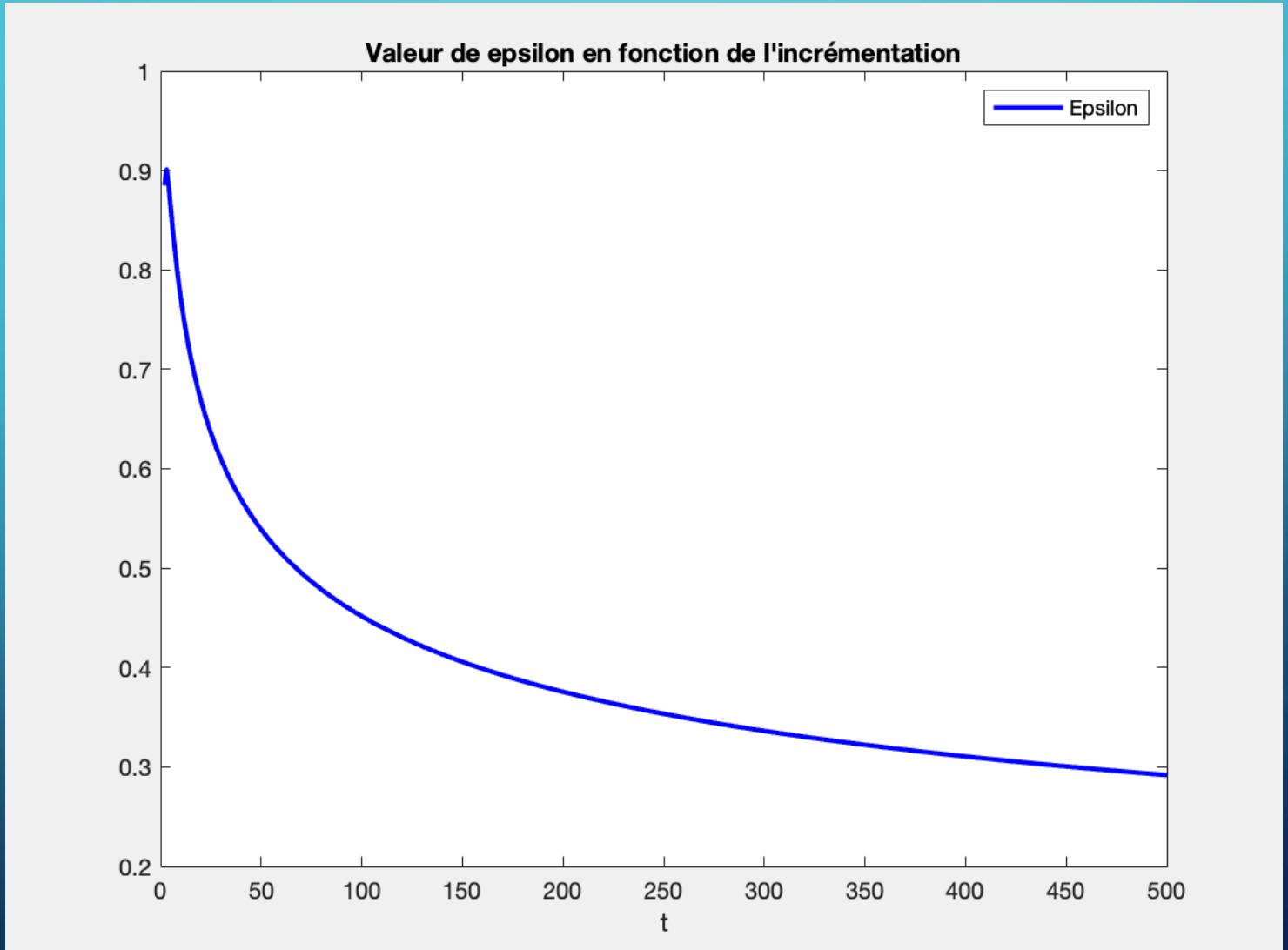
ALOGRITHME EPSILON GREEDY

Code de notre simulation

```
for t in range(T):
    # On définit le paramètre epsilon pour l'exploration
    with np.errstate(divide='ignore'):
        epsilon = np.power(t, -1/3) * np.power(k * np.log(t), 1/3)

    # Exploration-Exploitation Strategy
    if np.random.rand() < epsilon:          # Lancé de la pièce pour choisir entre exploration et exploitation
        # EXPLORATION
        arm = np.random.randint(k)
    else:
        # EXPLOITATION
        arm = np.argmin(est_means)        # Choisir le bras avec le PLUS PETIT TEMPS DE TRANSMISSION

    # On calcule la récompense pour chaque bras de la machine
    rewards_iteration = env.get_reward()

    reward = rewards_iteration[arm]        # On récupère le temps de transmission pour le bras choisi
    n[arm] += 1                            # On incrémente le nombre de fois que le bras choisi a été tiré
    rewards[arm] += reward                 # On ajoute la récompense observée aux récompenses cumulées pour le bras choisi
    est_means[arm] = rewards[arm] / n[arm] # On met à jour la récompense moyenne estimée pour le bras choisi
```

ALOGRITHME EPSILON GREEDY

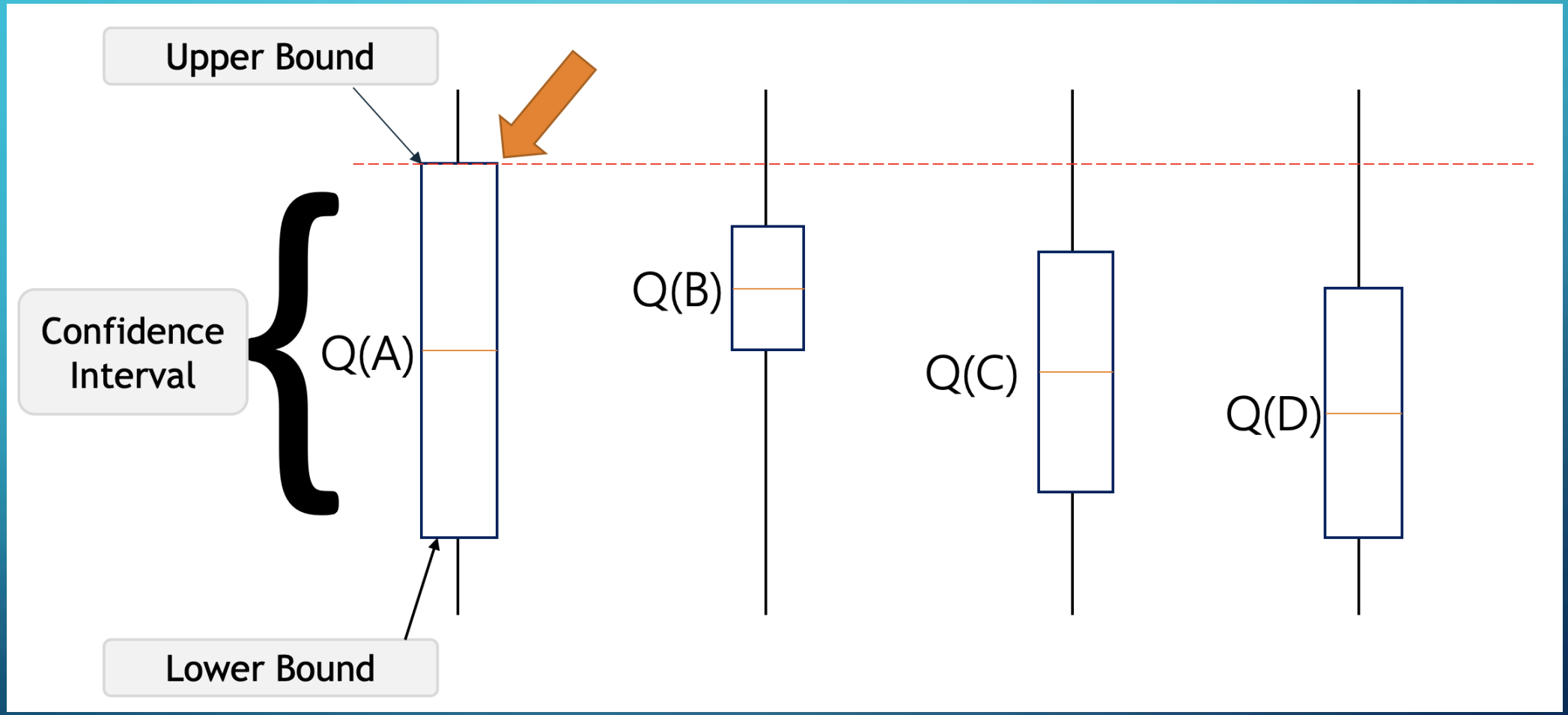
Epsilon = t^^(-1/3)*(k.log(t))**(1/3)



ALGORITHME UPPER CONFIDENCE BOUND (UCB)

```
for t in range(T):
    if t < k:
        # Jouer chaque bras k fois pour initialiser les estimations et les valeurs UCB
        reward_iteration = env.get_reward()
        reward = reward_iteration[t]
        n[t] += 1
        rewards[t] += reward
        est_means[t] = rewards[t] / n[t]
        regrets.append(0)
    else:
        reward_iteration = env.get_reward()  # Obtenir la récompense pour chaque bras de la machine

        # Choisir le bras avec la plus grande valeur UCB
        ucb_values = [est_means[i] + np.sqrt(2*np.log(t) / n[i]) for i in range(k)]  # Calculer les valeurs UCB pour chaque bras
        arm = np.argmin(ucb_values)  # Sélectionner le bras avec la PLUS PETITE VALEUR UCB

        reward = reward_iteration[arm]  # Obtenir la récompense pour le bras choisi

        n[arm] += 1  # Incrémenter le nombre de fois que le bras choisi a été tiré
        rewards[arm] += reward  # Ajouter la récompense observée aux récompenses cumulées pour le bras choisi
        est_means[arm] = rewards[arm] / n[arm]  # Mettre à jour la récompense moyenne estimée pour le bras choisi
```

ALGORITHME UPPER CONFIDENCE BOUND (UCB)



ANALYSE DES PERFORMANCES OBTENUES

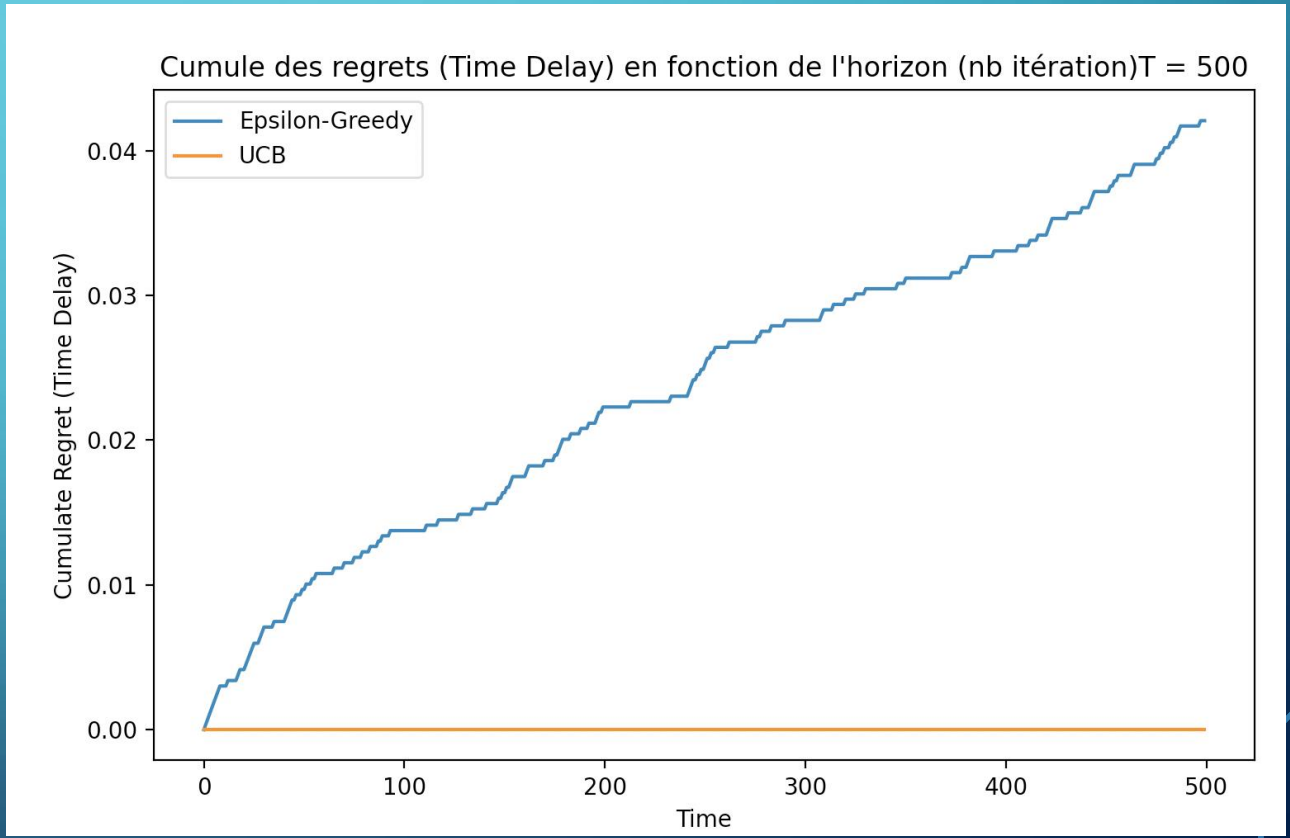
1er cas - Variation uniforme de la distance entre 1100m et 1400m, sur 500 itérations (envoies de paquet)

ANALYSE DES PERFORMANCES OBTENUES

Variation uniforme de la distance entre 1100m et 1400m, sur 500 itérations (envoies de paquet)

```
V2V_simulation:  [True, 0.0010006993333333335, 0, [0, 5,
11]]
final latency:  0.0010006993333333335
----------------------
V2I_simulation:
final latency  0.0008512506765411443
------------------------------------------------
V2I has best latency for distance  1200 m
------------------------------------------------
Distance:  1400
Traffic density:  0.05
V2V_simulation:  [True, 0.0010006899999999998, 0, [0, 4,
13]]
final latency:  0.0010006899999999998
----------------------
V2I_simulation:
final latency  0.0008748188209564401
------------------------------------------------
V2I has best latency for distance  1400 m
------------------------------------------------
```

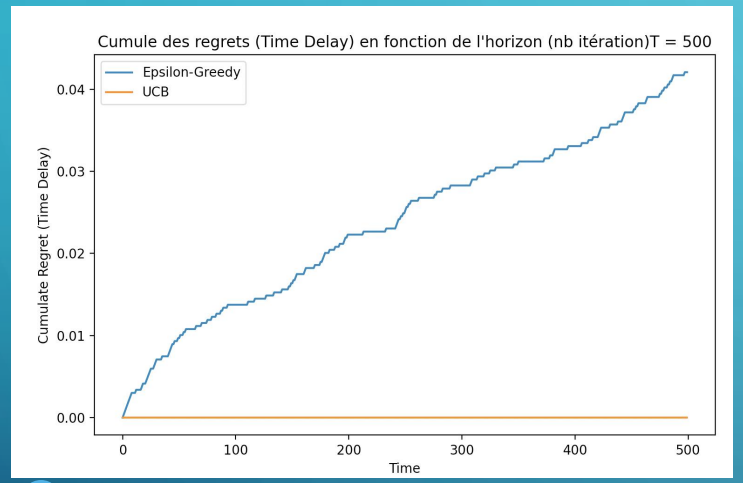
Sur la plage 1100m 1400m l'option V2I est la meilleure dans nos simulations

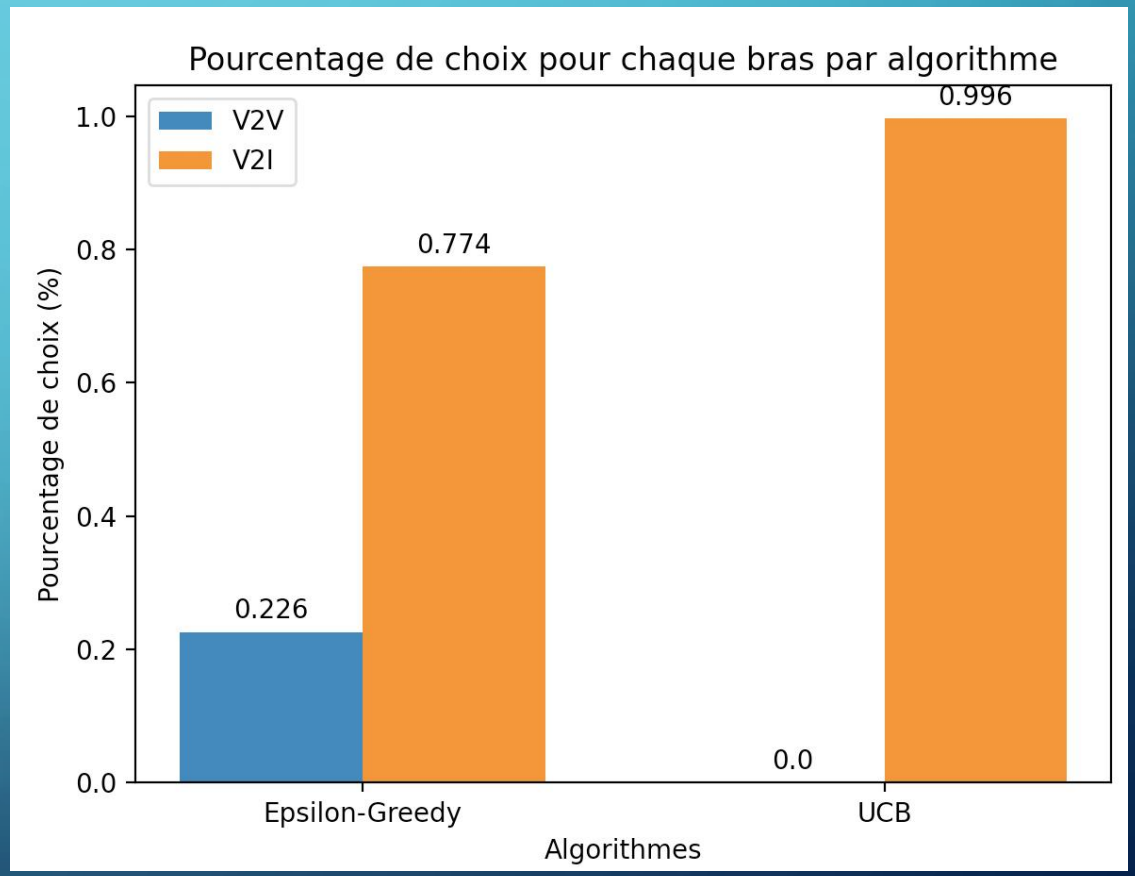


Cumule des regrets = somme des temps perdus par rapport au choix idéal sur chaque itération

ANALYSE DES PERFORMANCES OBTENUES

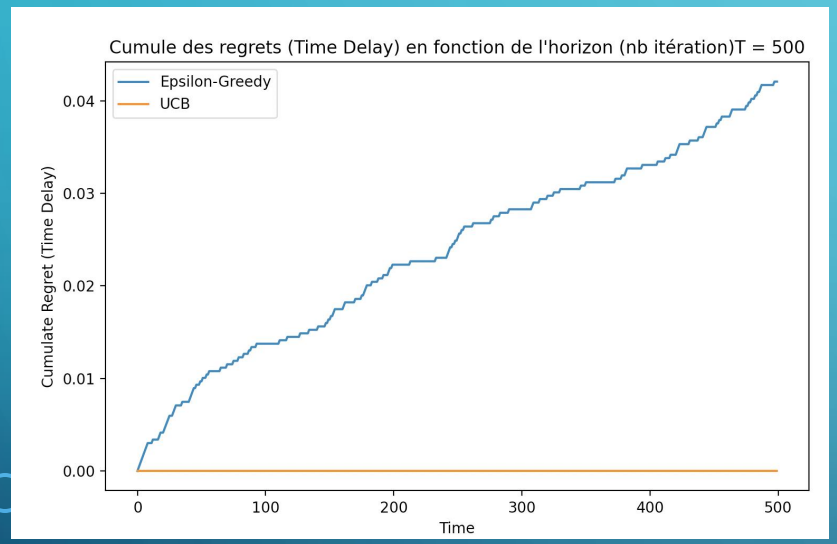
1er cas - Variation uniforme de la distance entre 1100m et 1400m, sur 500 itérations (envoies de paquet)



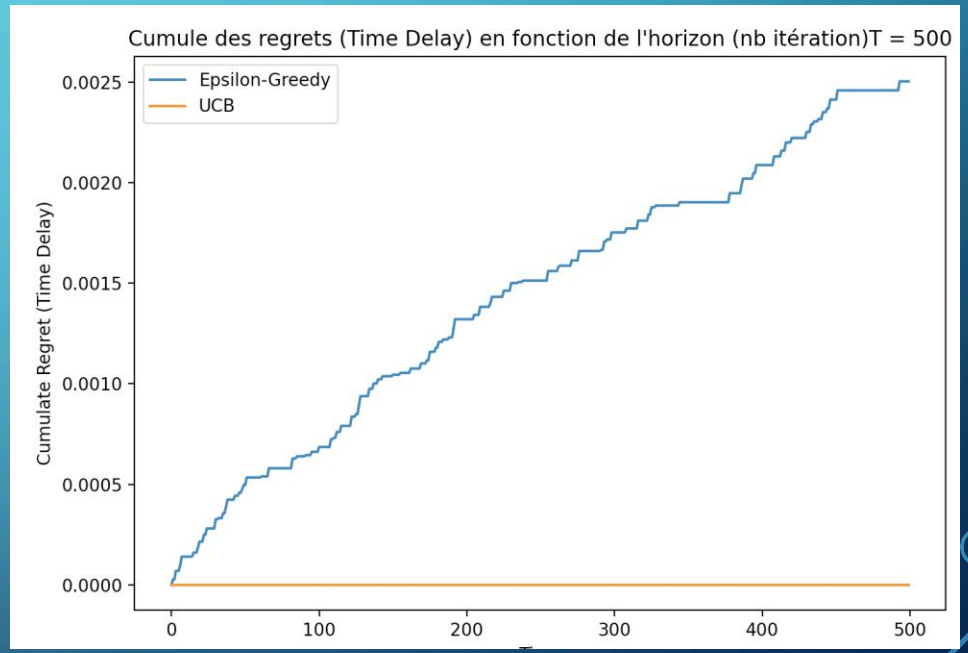


ANALYSE DES PERFORMANCES OBTENUES

1er cas - Variation uniforme de la distance entre 1100m et 1400m, sur 500 itérations (envoies de paquet)



Augmentation de la charge sur le réseau 5g (V2I)

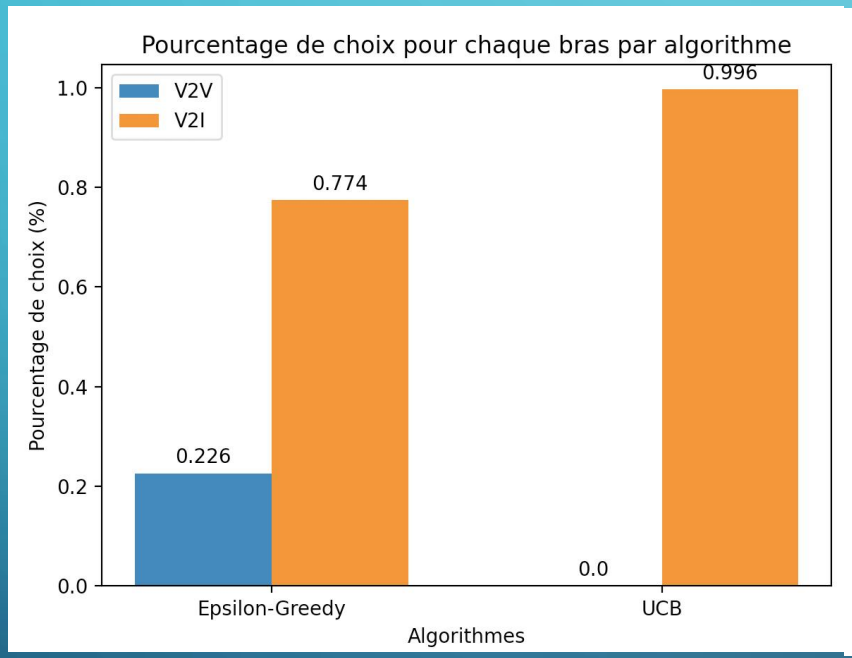


On constate logiquement une croissance plus rapide du cumule des regrets d'epsilon-greedy

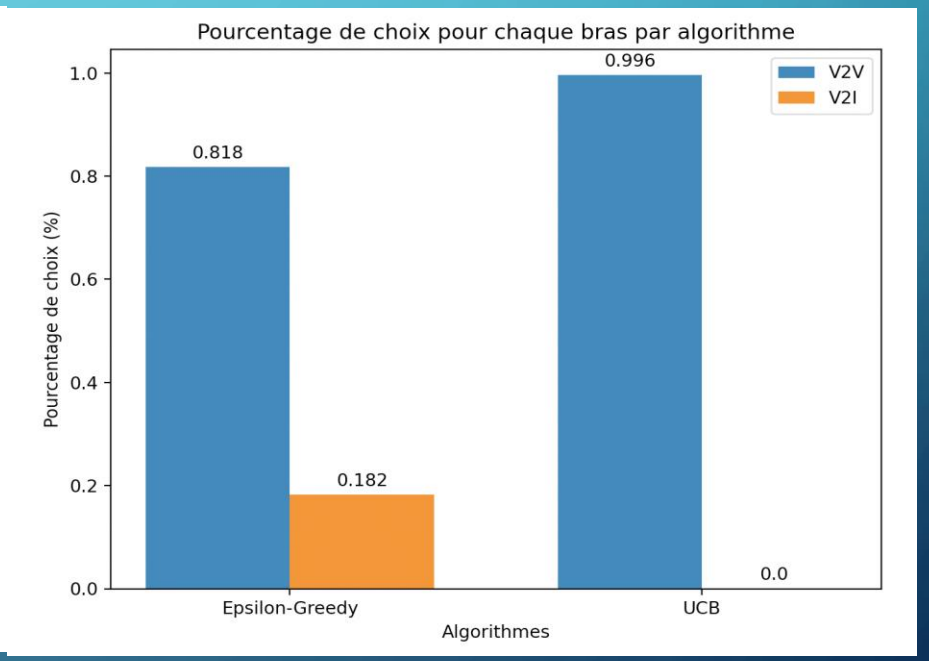
ANALYSE DES PERFORMANCES OBTENUES

1er cas - Variation uniforme de la distance entre 1100m et 1400m, sur 500 itérations (envoies de paquet) - Variation de la charge sur le réseau 5g

Faible charge sur le réseau 5g



Forte charge sur le réseau 5g



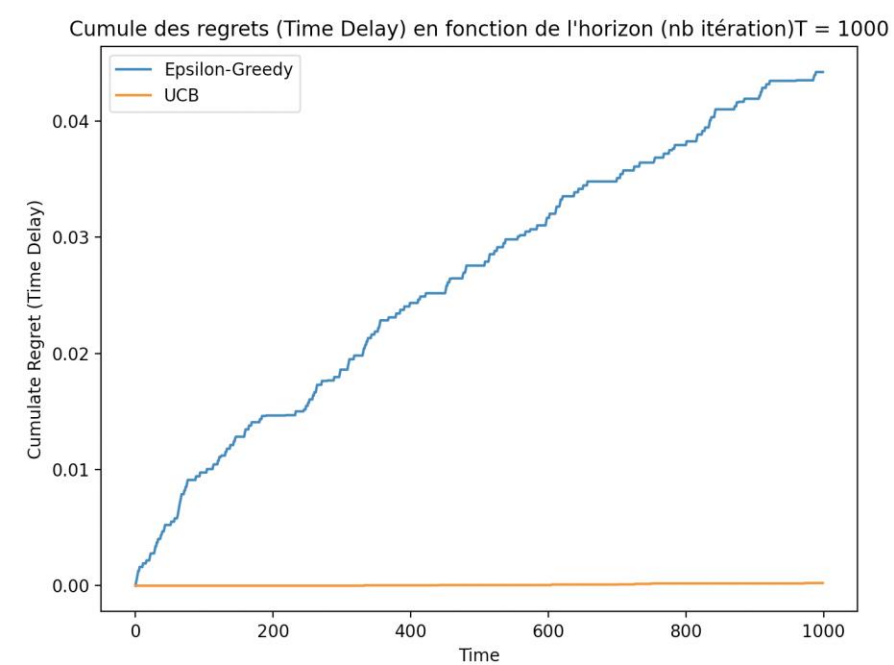
On constate logiquement une inversion nette du pourcentage des choix des deux algorithmes

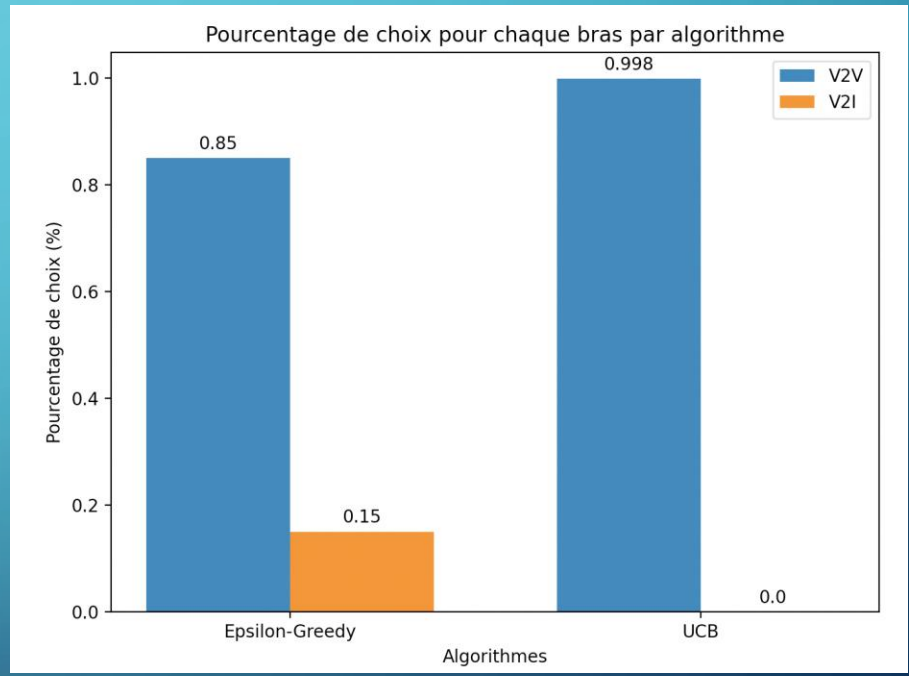
ANALYSE DES PERFORMANCES OBTENUES

2 ème cas - Variation uniforme de la distance entre 200m et 800, sur 500 itérations (envoies de paquet)

ANALYSE DES PERFORMANCES OBTENUES

Variation uniforme de la distance entre 200m et 800m, sur 500 itérations (envoies de paquet) - charge faible





Sur la plage 1100m 1400m l'option V2V est la meilleure option dans nos simulations, les algorithmes réagissent bien.

ANALYSE DES PERFORMANCES OBTENUES

3 ème cas – Courte Distance sur 20% de l'itération puis changement brutale de la distance sur le reste des 80%

ANALYSE DES PERFORMANCES OBTENUES

Observation du régime transitoire

20% Distance courte Puis 80% Distance longue