

## Planification optimisée des livraisons urbaines à l'aide de l'apprentissage par renforcement et de drones autonomes

Pr. Sidi-Mohammed Senouci



# Présentation du laboratoire **DRIVE**



Adossé à l'Institut Supérieur de l'Automobile et des Transports (ISAT) de l'Université de Bourgogne (UB),

Le laboratoire de recherche **DRIVE**, est situé à Nevers, en Nièvre (58) près du circuit Magny-Cours.

Composé d'une soixantaine de membres dont une trentaine d'enseignants-chercheurs et une vingtaine de doctorants, il possède des équipements de pointe et développe une recherche à la fois appliquée et fondamentale de haut niveau dans deux grands domaines cadres :

*les Systèmes intelligents et l'Optimisation énergétique  
ainsi que*

*la Mécanique des matériaux et des structures.*



# Présentation du laboratoire **DRIVE**



**DRIVE**

*Pr. Sidi Mohammed SENOUCI, directeur*

**Energie, Mobilité, Intelligence  
et Environnement (EMIE)**

**Mécanique et Acoustique  
pour les Transports (MAT)**

Outline

**Mobilité, Energie,  
Environnement et  
Propulsion (MEEP)**

*Pr. Luis LE MOYNE*



**Systèmes Intelligents  
et Connectés (SIC)**

*Pr. El-Hassane AGLZIM*



**Durabilité et  
Structures  
Composites (DSC)**

*Pr. Olivier SICOT*



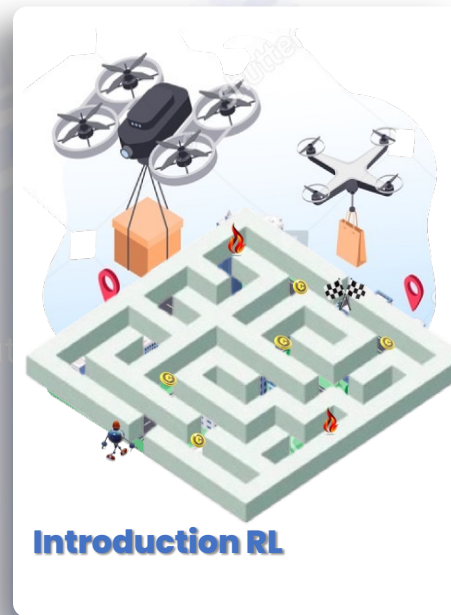
**Vibrations et  
Acoustique des  
Transports (VAT)**

*Pr. Philippe LECLAIRE*



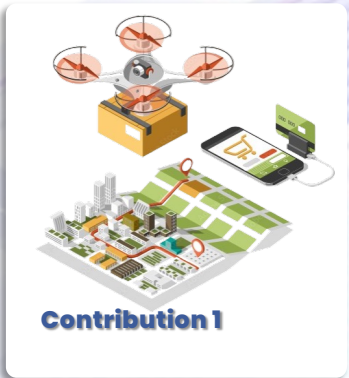
# Introduction à l'apprentissage par renforcement

## Plan



# Systeme de livraison hybride

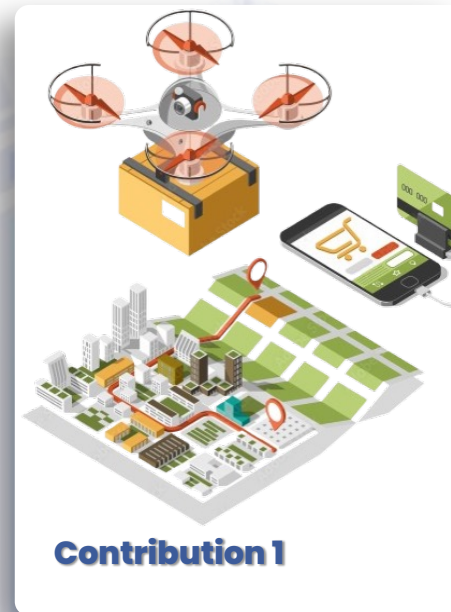
## Plan



# Plan

## Contribution 1

Optimiser la planification de la trajectoire des drones en équilibrant la **consommation d'énergie** et le **temps de livraison**.



# Plan

## Contribution 2

Planification de trajectoires pour flotte de drones toute en améliorant la **scalabilité** de l'apprentissage par renforcement et garantissant la protection de la **confidentialité** des données des entreprises de livraison.



**Contribution 2**



**Conclusion**



# Conclusion

# Plan





## Supervisé

L'apprentissage supervisé consiste à **apprendre à partir d'exemples** dont le **résultat est connu**. L'objectif est de trouver une fonction qui **associe les entrées X aux sorties Y**, formulée comme  **$Y=f(X)$**

## Non supervisé

L'algorithme fait des prédictions sur les données d'entraînement, qui sont corrigées en fonction des bonnes réponses. Ce processus itératif continue jusqu'à atteindre un niveau de performance satisfaisant.

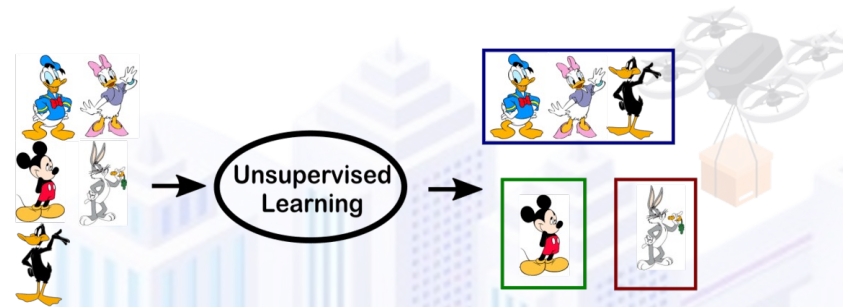
## Par Renforcement

- *Est-ce une voiture ou un camion?*
- *Ces courriels sont-ils du spam ou non?*
- *Prédit le prix de la voiture, la consommation, le kilométrage, etc.*

Supervisé

Non supervisé

Par  
Renforcement



Explore et découvre des **structures cachées** dans les données sans que les résultats attendus ne soient fournis à l'avance → que des données d'entrée (X) et **aucune variable de sortie correspondante**.

- *J'ai des photos et je veux les classer en 20 groupes.*
- *Regrouper les voitures selon leur modèle en différentes catégories (SUV, Berline,..) en fonction de la Puissance, Poids, Consommation, etc..*
- *Regrouper les conducteurs en différentes catégories : prudent, agressif, ou modéré selon les données de conduite (vitesse, freinage, accélération).*

Supervisé

Non supervisé

Par  
Renforcement

État (State)

Récompense (Reward)

Action

Environnement

Agent

L'apprentissage par renforcement est une méthode d'apprentissage automatique où un **agent** (comme un véhicule) **apprend à prendre des décisions** en interagissant avec un **environnement**.

L'agent reçoit des **récompenses** (rewards) ou des **punitions** en **fonction de ses actions**, et son **objectif est de maximiser les récompenses sur le long terme**.

- Robotique
- Véhicules autonomes (voitures, drones)
- Optimisation des feux de circulation
- Trading
- ...

# Introduction RL

Système de livraison hybride

Contribution 1

Contribution 2

Conclusion

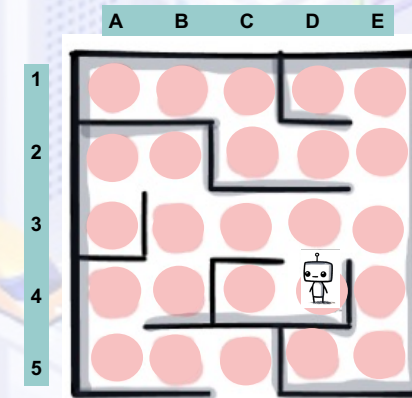
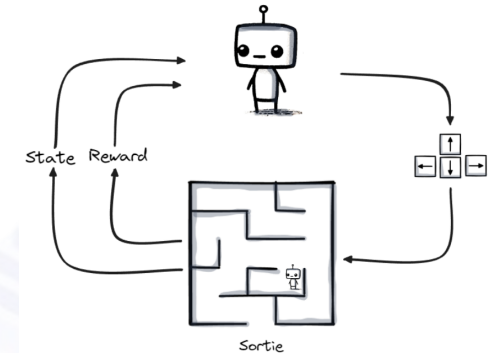
Types de problèmes d'apprentissage automatique

Apprentissage par renforcement

Q-learning

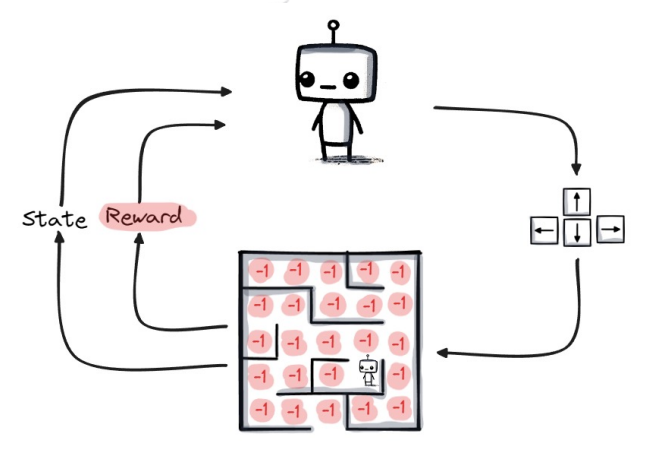
Caractéristique	Apprentissage Supervisé	Apprentissage Non Supervisé	Apprentissage par Renforcement
Type de données	Données étiquetées (x,y) où y est connu.	Données non étiquetées (x seulement).	Transitions de type (s,a,r,s') issues d'un environnement interactif.
Objectif	Apprendre une fonction qui prédit y à partir de x.	Identifier des structures ou des patterns dans les données.	Apprendre une politique pour maximiser la récompense cumulative.
Environnement	Fixe, basé sur des données statiques.	Fixe, basé sur des données statiques.	Dynamique, basé sur des interactions.
Approche d'évaluation	Comparaison avec des données de test (y connu).	Évaluation subjective (e.g., qualité des clusters).	Performance mesurée par la récompense accumulée.
Algorithmes courants	Régression linéaire, SVM, Naive Bayes, réseaux de neurones, arbres de décision.	K-means, DBSCAN, PCA, autoencodeurs.	Q-learning, Deep Q-learning, SARSA, Actor-Critic.
Exemples d'applications	Prédire des maladies, reconnaître des images.	Grouper des clients en segments, compression.	Robotique, Joueurs d'échecs IA, gestion d'énergie dans les réseaux.

- **Exemple** : Un robot qui va apprendre à sortir d'un labyrinthe le plus rapidement possible. Il est libre de se déplacer dans toutes les directions qu'il souhaite.
- **Etat (State) ?**
  - Position de l'agent dans le labyrinthe → **25 états**
- **Actions ?**
  - Ses déplacements possibles : **haut, bas, droite, gauche**
- **Récompense (Reward) ?**
  - En effectuant une action, l'agent change son état, ce qui s'accompagne d'une récompense → Attribuer un **score de -1**



La machine ne sera-t-elle pas triste en lui « Infligeant » un point négatif chaque fois qu'elle effectue un déplacement?

- Pourquoi « Infliger » un point négatif chaque fois que la machine effectue un déplacement?
  - Le but ultime est de **maximiser le score final**
  - Le robot cherchera donc le **moyen le plus rapide de rejoindre la sortie**, préférant marquer -10 points plutôt que -15.



Objectif du RL : Apprendre une fonction  $\pi$  nommée **politique d'action**  
 $\pi(s) = a$  qui donne quelle action à effectuer lorsque l'agent se situe dans un état  $s$

## • Principe de l'apprentissage par renforcement

### • Modélisation MDP

- $S = \{s_1, s_2, \dots, s_n\}$  un **ensemble d'états** décrivant l'environnement
- $A = \{a_1, a_2, \dots, a_m\}$  un **ensemble d'actions** que l'agent pourrait sélectionner dans chaque état dans  $S$
- À chaque instant où une action  $a_t$  est effectuée pour un état  $S_t$ , l'agent reçoit une récompense  $r_t$
- Le rôle de l'agent est d'apprendre la politique  $\pi : S \rightarrow A$ , qui maximise la valeur :

$$V^*(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2}, \dots$$

Tel que  $0 \leq \gamma \leq 1$  est un hyper paramètre (facteur d'actualisation - **discount parameter**)

- Si  $\gamma$  est proche de 0, l'agent tend à choisir une récompense immédiate
- Si  $\gamma$  est proche de 1, l'agent tend à considérer les récompenses à long terme.

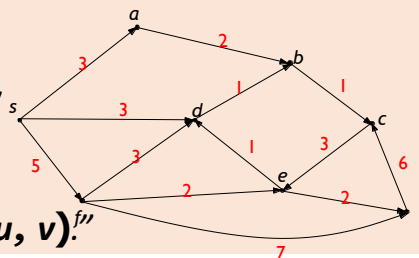
### Apprentissage par Renforcement Basé Modèle

- Principe
  - **Construction d'un modèle de l'environnement** ayant ou en ayant connaissance des dynamiques du système (**probabilités de transition** et **récompenses**).
  - La **programmation dynamique** (basé sur équations de Bellman) est utilisée pour trouver la politique optimale.
- Avantages :
  - Convergence garantie vers une solution optimale
  - Efficace dans des environnements où les dynamiques sont bien définies → adapté à une résolution en offline
- Inconvénients :
  - Gourmand en mémoire : Nécessite de stocker le modèle entier
  - Dépend d'une connaissance préalable de l'environnement

#### Equation de Bellman

$$d(v) = \min_{(u,v) \in E} (d(u) + w(u, v))$$

$d(v) :=$  poids minimum d'un chemin  $s-v$   
 "Le plus court chemin de  $s$  à  $v$  est le plus court chemin de  $s$  à un antécédent  $u$  de  $v$  auquel on a ajouté le dernier arc  $(u, v)$ ."



### Apprentissage par Renforcement Sans Modèle

- Principe
  - **Aucun modèle de l'environnement.**
  - L'agent apprend à partir des interactions avec l'environnement, en explorant et exploitant ses actions.
    1. **Monte Carlo (MC)** : Utilise la **simulation aléatoire** puis estime les fonctions de valeur à partir de la moyenne des récompenses observées sur des trajectoires complètes d'épisodes.
    2. **Différence Temporelle (TD)** : Combine la programmation dynamique et Monte Carlo: **Q-Learning**
- Avantages :
  - Peu gourmand en mémoire
  - Adapté aux environnements où les **dynamiques sont inconnues ou changeantes** → adapté à une résolution on-line
  - Peut être utilisé dans des environnements simulés ou réels sans besoin de connaître les transitions.
- Inconvénients :
  - Apprentissage **plus lent**, car l'agent doit explorer suffisamment l'environnement pour converger.
  - La **convergence peut être instable** dans des environnements très stochastiques ou avec un **espace d'état-action élevé**



## Exemple 1 : Franchissement de pont

Soit quatre individus a, b, c, d qui rentrent de randonnée à la nuit tombée. Ils rencontrent un pont suspendu enjambant une rivière. Un panneau indique qu'au **maximum deux personnes** simultanément peuvent emprunter le pont pour des raisons de sécurité. Les quatre compagnons décident de franchir le pont deux par deux. Cependant, le groupe ne disposant que **d'une seule lampe de poche**, une personne se trouvant après le pont doit retourner au point de départ pour rapporter la lampe après chaque franchissement. Précisons enfin que l'individu **a met 1 minute** pour traverser, **b met 2 minutes**, **c met 5 minutes** et **d met 10 minutes**. Si deux membres du groupe traversent le pont ensemble, ils marchent à la vitesse du membre le plus lent (par exemple, a et c mettent 5 minutes pour traverser ensemble). **Dans quel ordre le groupe doit-il traverser pour minimiser le temps total de franchissement ?**

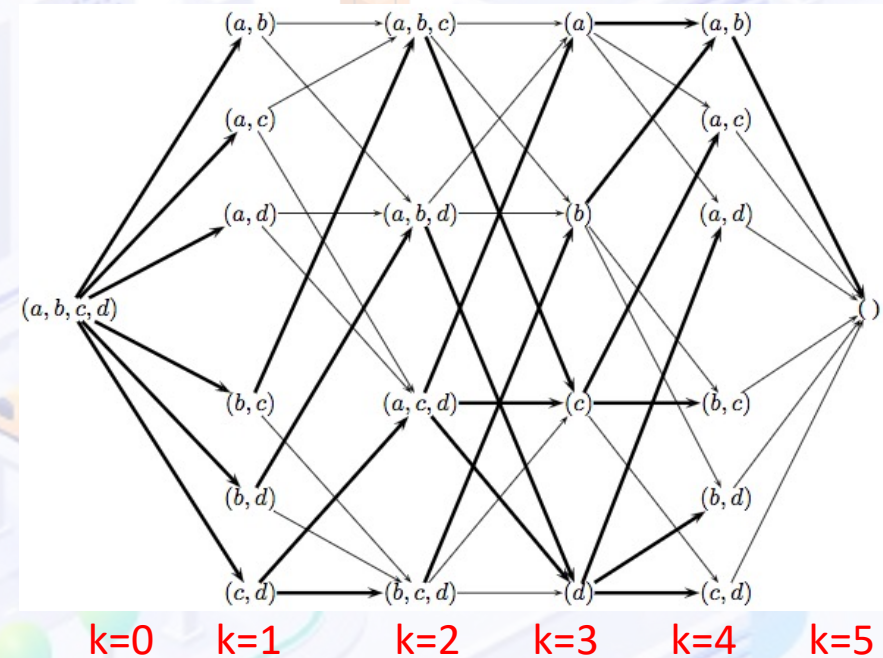
Apprentissage par Renforcement Sans ou Avec  
Modèle?

## Exemple 1 : Franchissement de pont

### • Modélisation MDP :

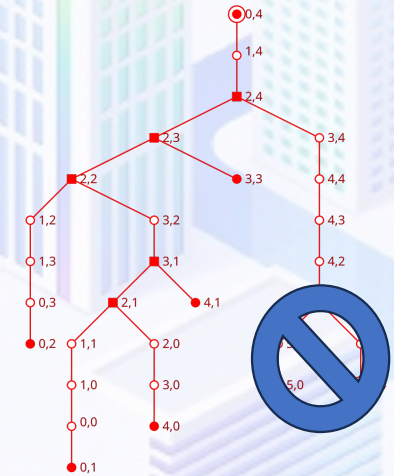
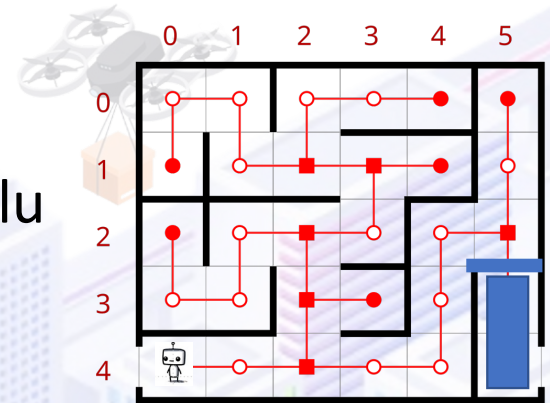
- **États** : Liste des personnes qui se situent avant le pont. Un état initial  $x_0 = (a, b, c, d)$  et un état final  $x_5 = ()$ .
- **Décision**  $u_k$  : qui sont les deux personnes qui vont passer sur le pont à la période  $k$ ?
- **Transitions** :  $x_k \rightarrow x_{k+1}$
- **Coût de la transition**  $x_k \rightarrow x_{k+1} : c(u_k) =$  vitesse du membre le plus lent des deux

a met 1 minute  
 b met 2 minutes  
 c met 5 minutes  
 d met 10 minutes.

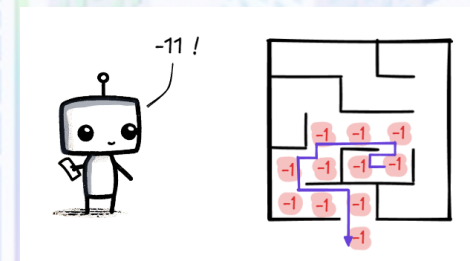
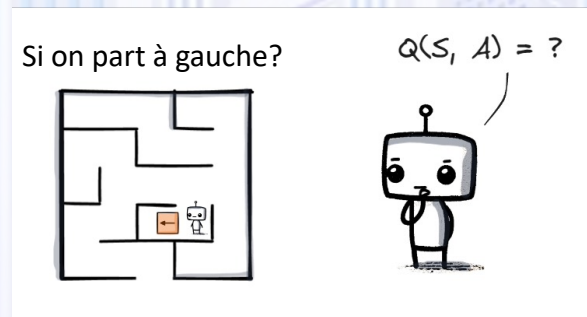
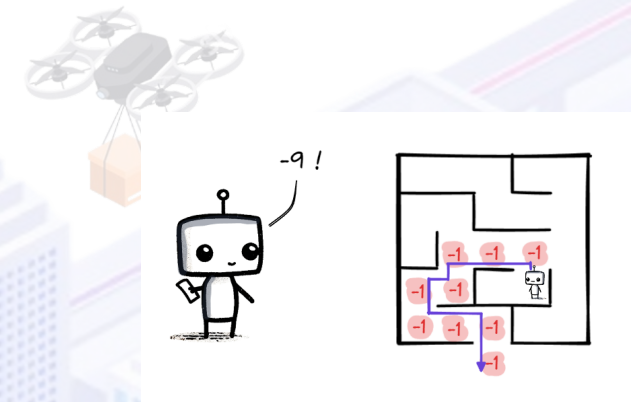
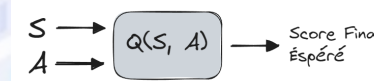


## Exemple 2 : Labyrinthe

- Est-ce que le problème du labyrinthe peut être résolu avec apprentissage par Renforcement Sans ou Avec Modèle ?
  - Si le labyrinthe est connu à l'avance et fixe, la **programmation dynamique** est une méthode efficace et suffisante.
  - L'**apprentissage par renforcement**, en revanche, devient pertinent lorsque l'environnement est incertain, changeant, ou que le but est de produire une stratégie adaptable.

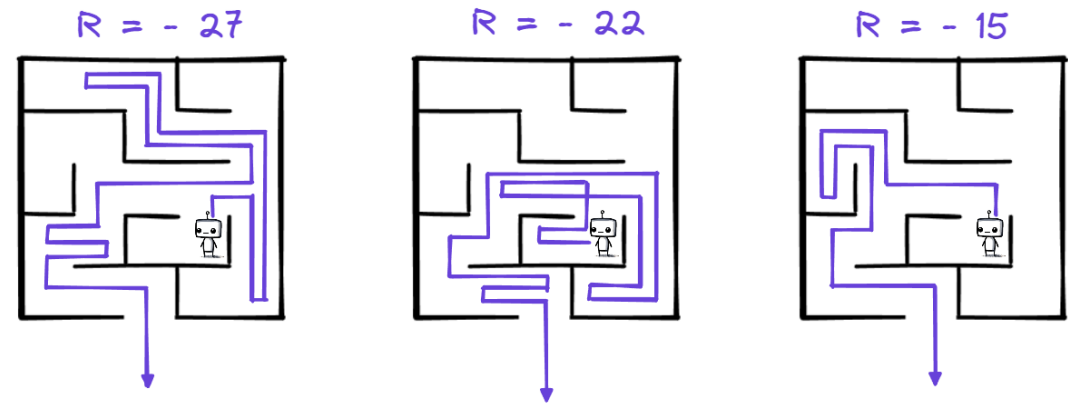


- Q-learning : basé sur la méthode de la **différence temporelle** combinant les avantages de la **programmation dynamique** et **MC** pour résoudre des **processus de décision markoviens (MDP)**.
  - Contrairement aux approches purement **basées sur les modèles** ou les **simulations complètes**, Q-learning apprend à partir de l'expérience en mettant à jour les estimations des récompenses futures à **chaque étape**.
  - **Pas besoin de modèle de l'environnement**
- Il apprend une fonction **Q** qui prend en entrée un état **S** et une action **A**, et qui prédit le **score final** que l'on peut espérer obtenir si tout se passe pour le mieux par la suite



Il reste maintenant une question : Comment apprendre cette fonction Q ?

- Pour apprendre à prédire la valeur de ses actions, la machine doit **explorer son environnement** en s'y **déplaçant au hasard**, afin de générer des données  $(S, A, R)$  (état, action, récompense).



- Ces données sont ensuite utilisées pour mettre à jour un **tableau  $Q(S, A)$**  qui informe la machine des **récompenses qu'elle est censée obtenir à l'avenir** en choisissant telle ou telle action dans l'état présent.
  - Tableau  $Q(S, A)$  rempli de valeurs **aléatoires** initialement

$Q(S, A)$

$S \backslash A$	↑	→	↓	←
	0.34	-0.18	1.69	-0.45
	0.81	-0.74	0.98	0.49
	-0.63	-0.90	0.29	0.73
⋮	⋮	⋮	⋮	⋮

**Comment calculer cette récompense qu'elle est censée obtenir à l'avenir?**

## • Mise à jour avec l'équation de Bellman

- L'équation de Bellman peut être utilisée pour reformuler le problème de maximisation des gains en un **problème récursif**.
- **Score final** = la récompense obtenue **immédiatement** dans l'état  $S$ , à laquelle on **ajoute le même score pour l'état suivant  $S'$** , si l'on choisit la meilleure action possible dans ce nouvel état.

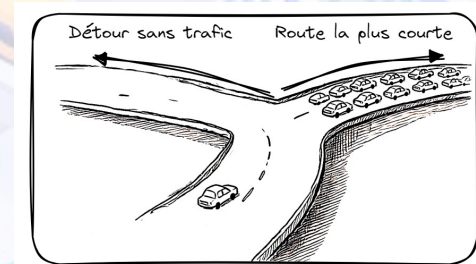
$$Q(S, A) = E [ R(S') + \gamma \max_{A'} Q(S', A') ]$$

Récompense immédiate obtenue en allant à l'état  $S'$   
 dévaluation  
 Récompenses totales espérées dans le nouvel état  $S'$ , en prenant la meilleure action  $A'$   
 Espérance Mathématique  
 Récompenses totales espérées




### Analogie :

Tourner à gauche vous donne la récompense immédiate d'éviter le trafic + la valeur que vous allez obtenir dans votre prochain état  
 Problème si vous tombez sur des bouchons 1 km plus loin → le fait de tourner à gauche n'était peut-être pas la meilleure option.



- À chaque fois que l'agent dans l'état  $S$  réalise une action  $A$ , il met à jour sa table  $Q$ , car il reçoit une récompense  $R(S')$  associée à son changement d'état à  $S'$


$$\boxed{Q(S, A)} = (1 - \alpha) \boxed{Q(S, A)} + \alpha \left( \boxed{R(S') + \gamma \max_{A'} Q(S', A')} \right)$$

Nouvelle valeur Q                      Valeur Q actuelle                      Equation de Bellman

learning rate

Ok, mais quelle action choisir à chaque étape de l'apprentissage?

## Exploration vs. exploitation ?



Exploiter?	Explorer?
<p>Agir en maximisant la connaissance actuelle en prenant la meilleure action (<b>maximisant la valeur Q</b>) :</p> $a^* = \arg \max_a Q(s, a)$	<p>Explorer toutes les actions</p> <p>Améliorer notre connaissance</p>
<p>Trop exploiter mène à des plans non optimaux</p>	<p>Trop explorer ralentit l'apprentissage inutilement</p>

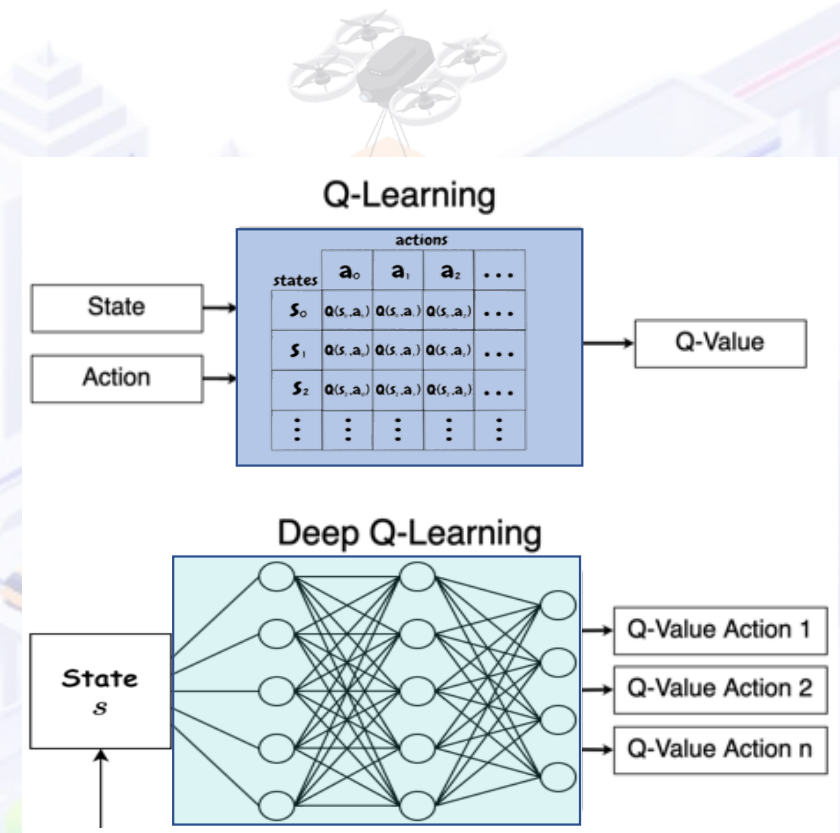
- **Explorer** toutes actions quand on a aucune idée sur le modèle, puis **Exploiter** la meilleure action une fois qu'on a acquis de l'expérience sur ce dernier
- **Méthode:**  $\epsilon$ -greedy où l'agent choisit une action aléatoire avec une probabilité  $\epsilon$  et la meilleure action connue avec une probabilité  $1-\epsilon$



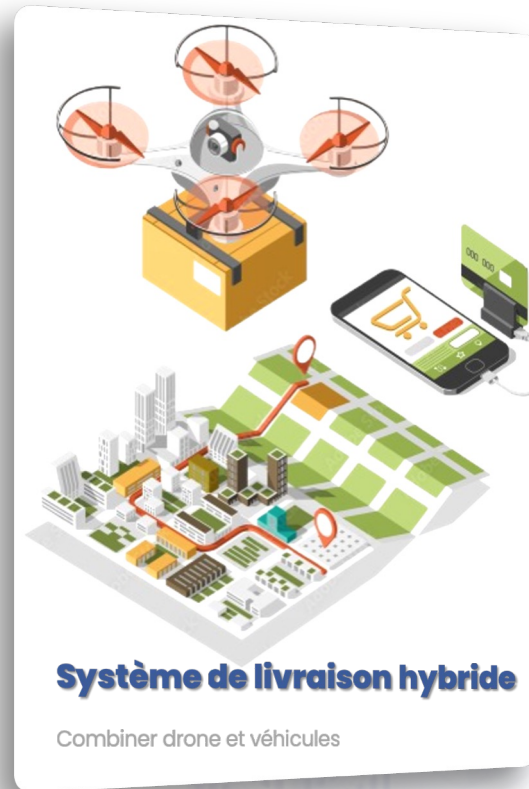
- Hy per paramètres pour personnaliser le comportement d'apprentissage de l'agent.
  - **Alpha ( $\alpha$ )**: Taux d'apprentissage qui est entre 0 et 1.
    - Alpha = 0 veut dire qu'il n'y a pas d'apprentissage
    - Alpha = 0.9, Apprentissage se fait rapidement.
    - A peut **décroître dynamiquement** au fil du temps pour favoriser l'exploration (grand  $\alpha$ ) au début et la stabilisation à la fin (petit  $\alpha$ )
  - **Gamma ( $\gamma$ )** : entre 0 et 1. Détermine l'importance des récompenses futures.
    - Un facteur de 0 rendra l'agent myope en ne considérant que les récompenses actuelles
    - Un facteur proche de 1: rechercher une plus grande récompense sur le long terme.
  - **Epsilon ( $\epsilon$ )**: Paramètre d'exploration: entre 0 et 1. Définit le mécanisme d'exploration dans la sélection d'action  $\epsilon$ -gourmande.
    - l'agent explore l'environnement en sélectionnant une action au hasard avec une probabilité  $\epsilon$ .
    - l'agent exploite ses connaissances actuelles en choisissant l'action optimale avec une probabilité  $1 - \epsilon$ .
  - **iter** Nombre d'itérations d'apprentissage répétées que l'agent passe dans l'ensemble de données d'entraînement. La valeur par défaut est 1.

- **Deep Q-learning (DQN) :**

- Le DQN utilise un **réseau de neurones profond (Deep Neural Network)** pour approximer la fonction  $Q(s,a)$ .
- Au lieu de **stocker les valeurs  $Q(s,a)$**  dans une table, le réseau prend **l'état  $s$  comme entrée** et **prédit les valeurs  $Q(s,a)$**  pour chaque action  $a$ .
- Conçu pour les environnements avec des espaces d'états et d'actions **très grands** ou **continus**.



- **Double Q-Learning** est une extension du Q-Learning conçue pour réduire le **biais d'optimisme**
  - **Problème du biais dans le Q-Learning classique**
    - Une seule Q-table est utilisée à la fois pour :
      - **Choisir** la meilleure action (avec  $\arg \max$ ).
      - **Estimer** la valeur future associée à cette action.
    - Le fait de choisir et d'évaluer en utilisant le même tableau ou modèle peut entraîner des estimations trop optimistes
  - **Solution apportée : utiliser deux Q-tables (Q1 et Q2) indépendantes pour séparer la sélection et l'évaluation des actions :**
    - **Sélection de l'action** : L'une des Q-tables est utilisée pour **choisir la meilleure action**.
    - **Évaluation de l'action** : L'autre Q-table est utilisée pour **estimer la valeur associée à cette action choisie**.
    - Et lors de la **prochaine** mise à jour, les rôles de Q1 et Q2 s'inversent.
    - Les deux Q-tables permettent de limiter l'influence des erreurs en évitant qu'une seule source de données (Q) ne contrôle à la fois la décision et l'évaluation.



03 Coût des carburants



05 Embouteillages plus importants



04 Émissions de carbone élevées

01 Temps de livraison



02 Coût de maintenance



**650K+ Emplois**

Offre plus d'emplois, y compris des emplois dans le domaine de l'informatique, de l'ingénierie des réseaux, etc.

**2.4M tonnes**

Minimiser les émissions de carbone

**-\$2.45 / mile**

Réduire le coût de livraison par mile de 2,5 à 0,05 dollars.

**1.7M voitures**

Réduire de 1,7 million le nombre de voitures sur les routes par an.

**20 min plus rapide**

Réduire le temps de livraison de 30 à 10 minutes.



[2] PricewaterhouseCoopers. (n.d.). *Skies without limits v2.0*. PwC.  
<https://www.pwc.co.uk/issues/technology/drones/the-impact-of-drones-on-the-uk-economy.html>







Introduction RL

## Système de livraison hybride

Contribution 1

Contribution 2

Conclusion

Systèmes de livraison traditionnels

Système de livraison par drone

Système drone-véhicule (hybride)

Drone vs. Syst. hybride



**Le drone est connecté au réseau cellulaire, permettant un contrôle à distance en temps réel, ainsi que l'optimisation et la replanification dynamique de sa trajectoire.**

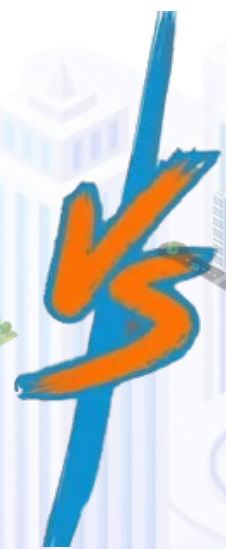


**Le drone peut transiter  
et utiliser plus d'un  
véhicule public.**



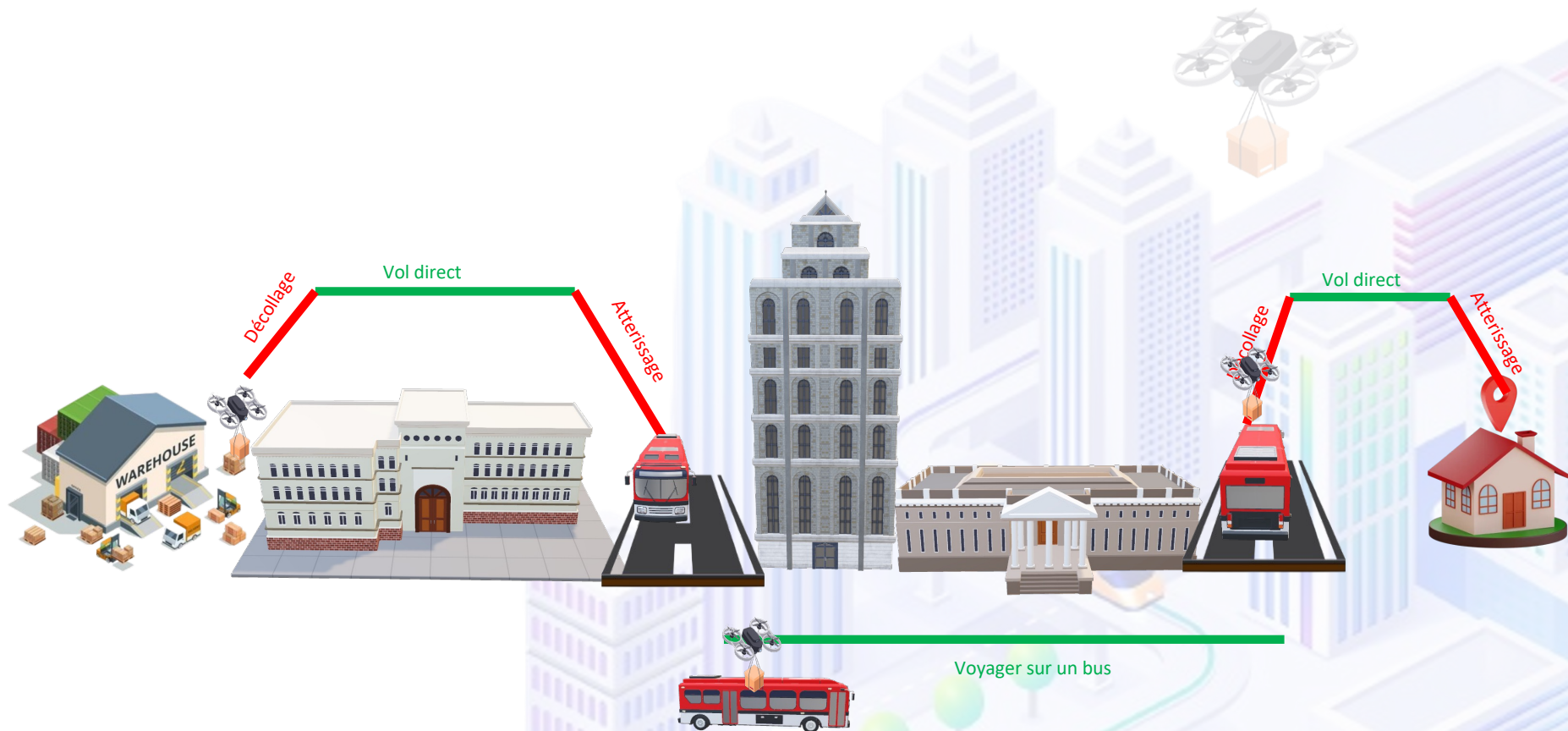


**Système hybride combinant drones et transports publics**

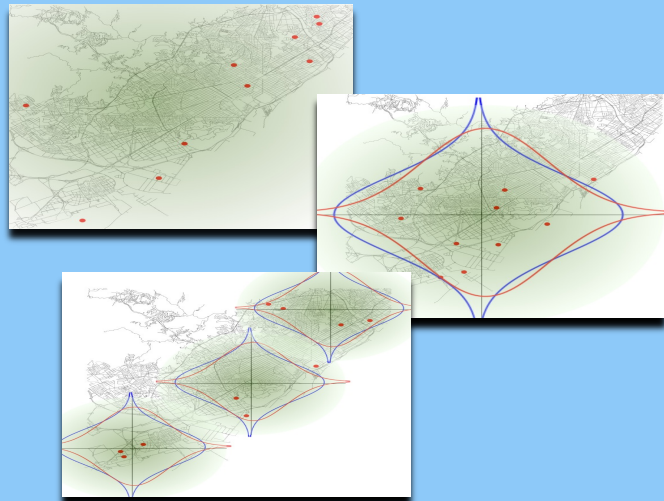


**Système avec drones uniquement**



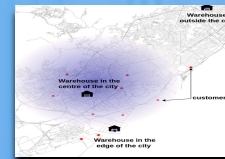


**Modèle énergétique considérant les phases (environnement 3D, vol stationnaire, correspondances avec les transports publics)**



**Différentes distributions de clients**

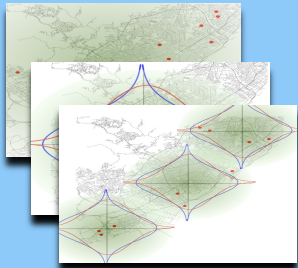
3 distributions de consommateurs différentes - uniforme, normale et en cluster



**Différentes distributions d'entrepôts**



**Ville de Barcelone en 3D**



**Différentes  
distributions  
de clients**

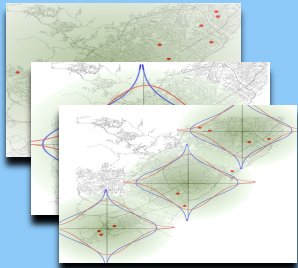


**Différentes distributions  
d'entrepôts**

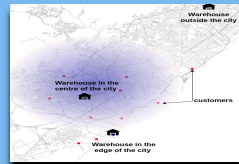
3 positions d'entrepôt différentes : Centre ville,  
Périphérie de la ville et Extérieur de la ville.



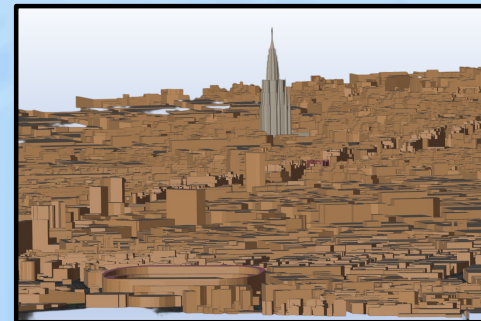
**Ville de Barcelone  
en 3D**



**Différentes  
distributions  
de clients**



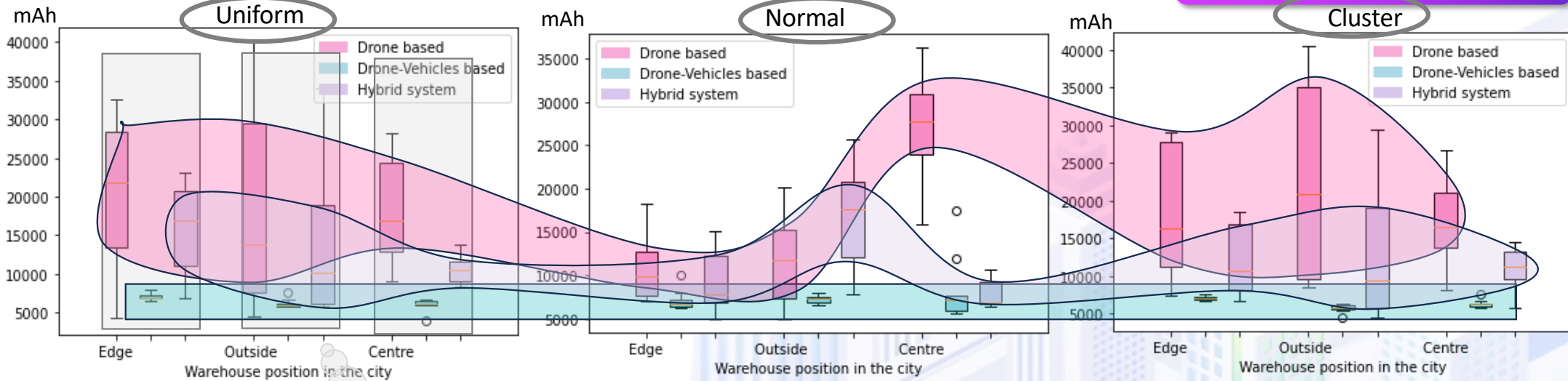
**Différents distributions  
d'entrepôts**



**Ville de Barcelone  
en 3D**

OpenStreetMap pour obtenir des données 3D réelles sur les structures des bâtiments et l'emplacement des stations de bus à Barcelone.





01

## Drone-vehicles

Ne dépend pas de la répartition des consommateurs ou de la position de l'entrepôt (7000 mAh)

02

## Drone seul

Affecté par la position de l'entrepôt et la distribution des consommateurs

03

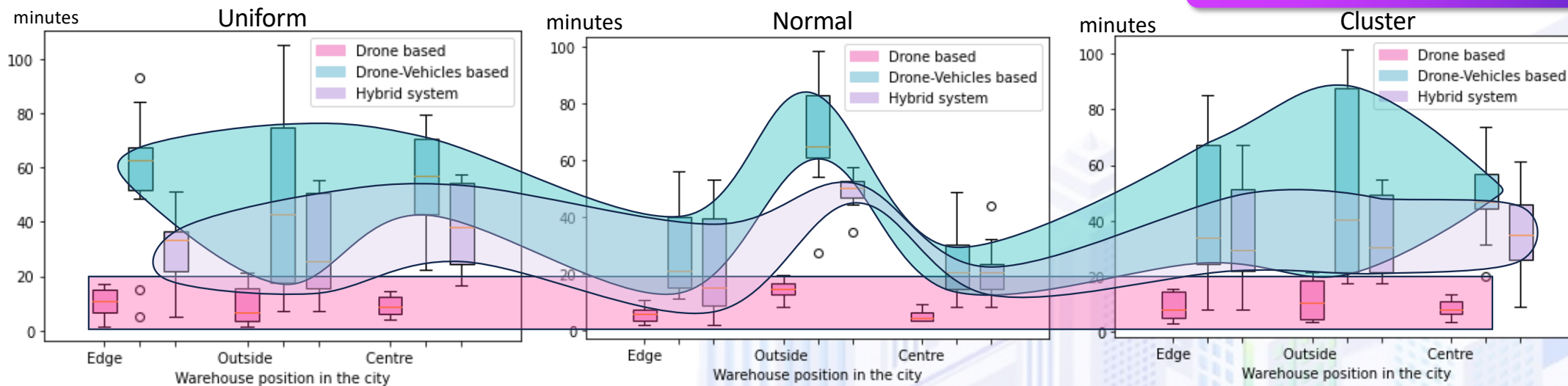
## Comparaison

La consommation d'énergie des drones seuls est nettement plus élevée que celle des drones-véhicules dans tous les scénarios.

04

## Système hybride

La consommation d'énergie se situe entre les deux systèmes



05

## Drone-vehicles

Dépendant de la position de l'entrepôt et la répartition des consommateurs.

06

## Drone seul

Indépendant de la position de l'entrepôt et de la répartition des consommateurs

07

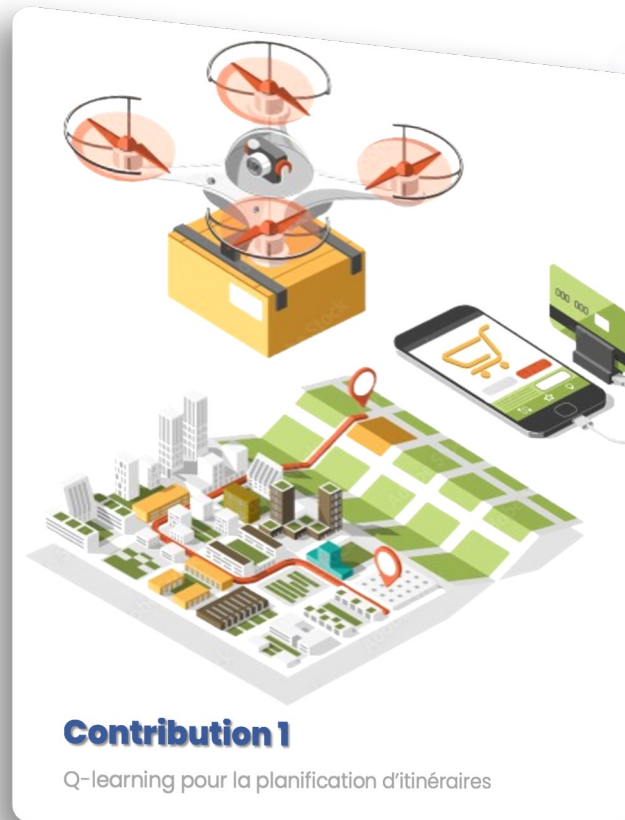
## Comparaison

Le délai de livraison des drones seuls est inférieur à celui des drones-véhicules dans tous les scénarios

08

## Système hybride

Le délai de livraison se situe entre les deux systèmes





# Q-learning pour la planification d'un itinéraire d'arrivée au plus tôt et économe

Enoncé du problème

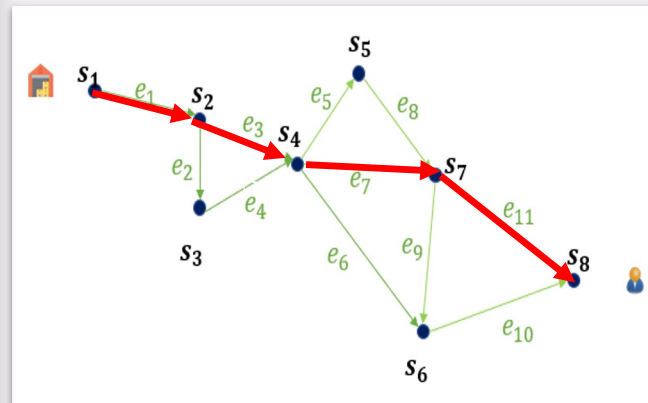
**Pourquoi?**

Incertitude +  
Dépendance  
temporelle des  
transports publics

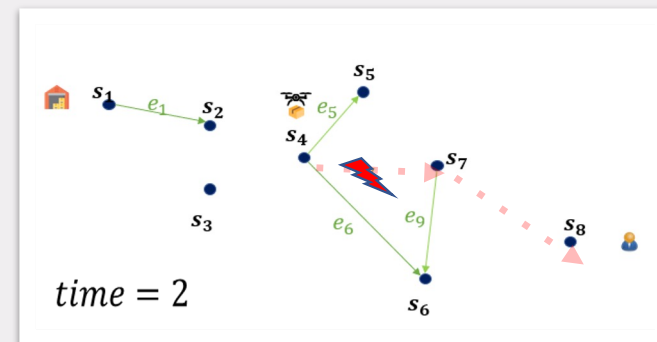
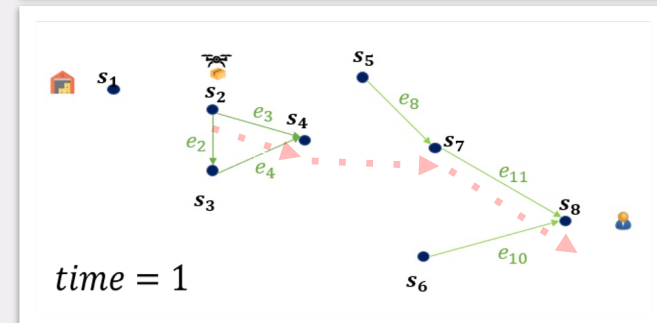
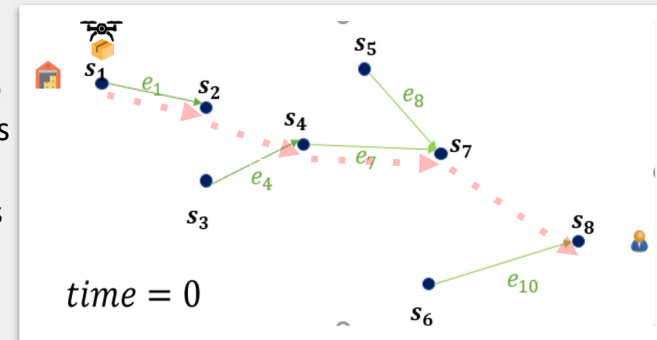
### Pourquoi Q-learning ?

Inconvénients d'une Planification hors ligne  
Méthodes traditionnelles dans lesquelles les trajets  
sont définis avant le début des livraisons.

☹️ La trajectoire ne s'adapte pas aux changements  
inattendus des horaires de transport.

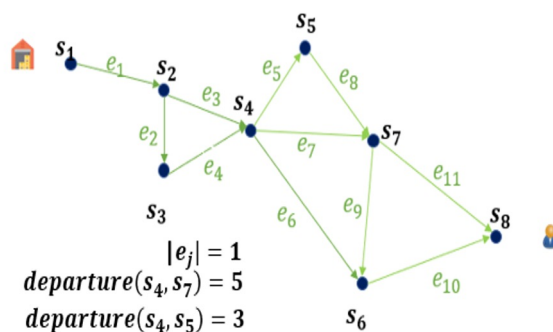


Après deux unités de temps, les liens ne  
sont plus les mêmes ce qui fait que le  
drone manque le bus qui va de **s4** à **s7** et  
perturbe la trajectoire prévue.



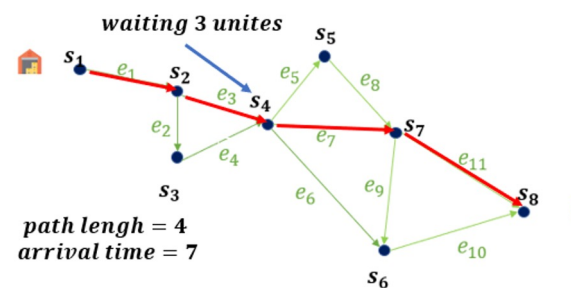
### Chemin le plus court vs. Itinéraire d'arrivée au plus tôt

Dans un système de livraison par drone, le principal objectif est de livrer rapidement → Critère = Arrivée au plus tôt plutôt et non le chemin le plus court.



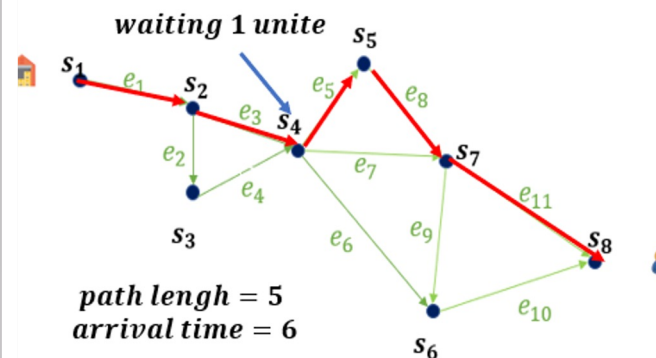
### Plus court chemin

Réduire le temps de traversée sans tenir compte des temps d'attente



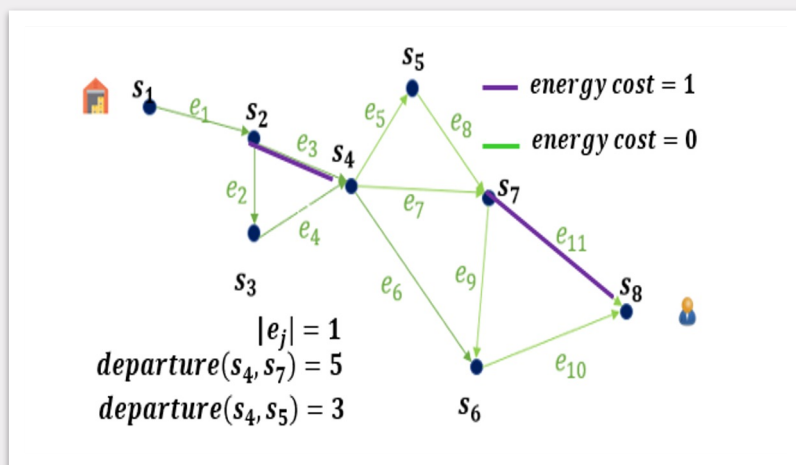
### Itinéraire d'arrivée au plus tôt

Réduire l'heure d'arrivée en tenant compte du temps de trajet et de l'heure de départ des véhicules

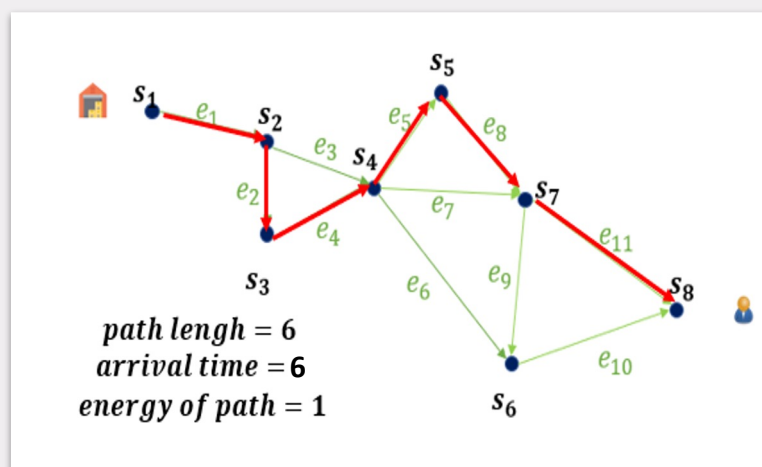
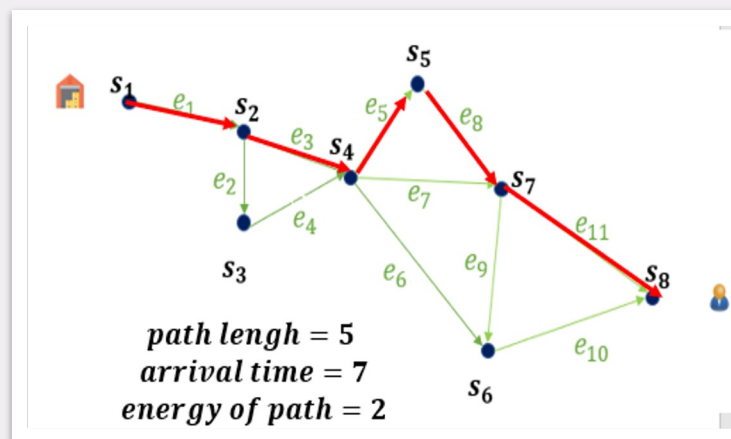


## Réduction de la consommation d'énergie

Un système hybride implique le choix entre des vols directs de drones ou d'utilisation de véhicules publics  
 → incidence sur la consommation d'énergie.



Même si les temps d'arrivée sont les mêmes, la consommation d'énergie pour les deux chemins est différente.



3

2

1

$E = \{ \langle v, v', l_i, t_j \rangle / v, v' \in V; l_i \in L; t_j \in T^{l_i} \}$   
liens entre les paires de nœuds

$V = \{ v, v', v'', v''' \dots \}$   
Arrêts de bus, Entrepôts,  
Clients

$G(V, E, \Omega)$

$\Omega = \{ \omega_{vv'}^{l_i, t_j} / v, v' \in V; l_i \in L; t_j \in T^{l_i} \}$   
Temps de trajet entre les paires de  
nœuds  $v, v'$  sur la ligne  $l_i$   
pendant le trajet par  $t_j$

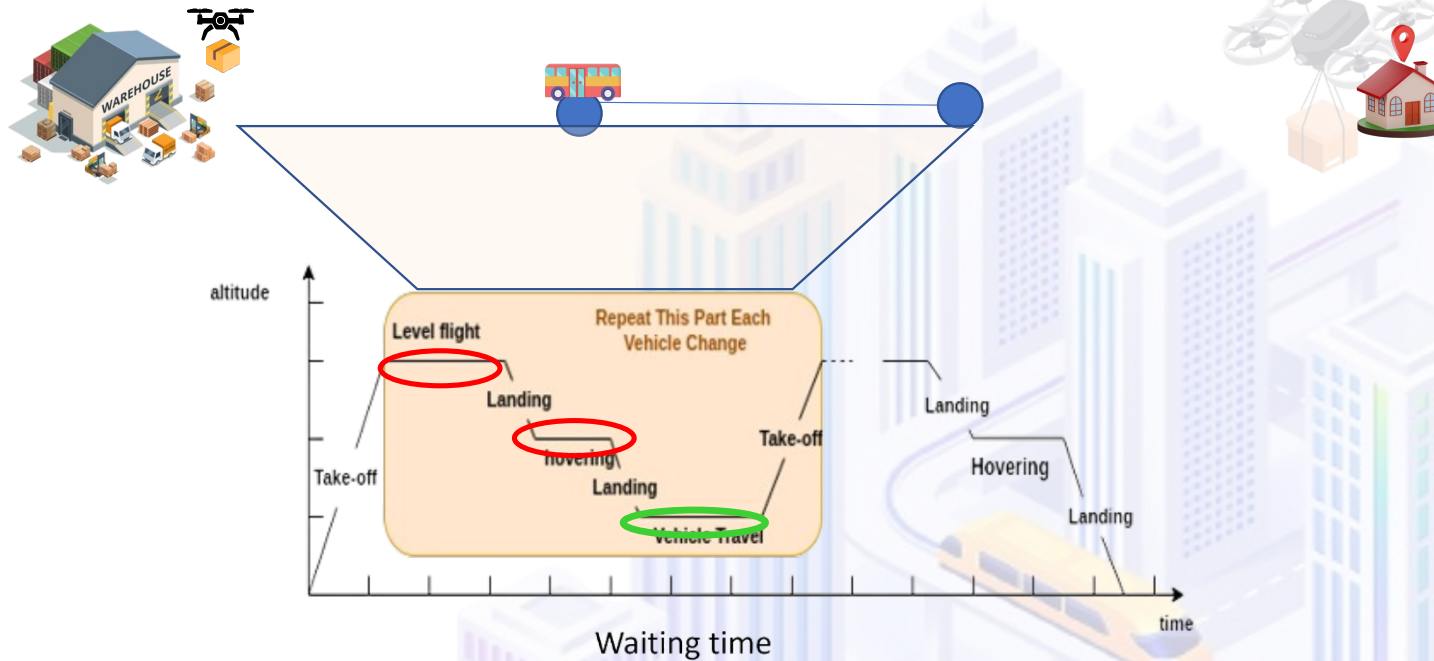
$T^{l_i} = \{ t_j, t_{j+1}, \dots \}$   
Numéros de trajets  
sur la ligne  $l_i$

$L = \{ l_0, l_i, l_{i+1}, \dots \}$   
Lignes de transport public et  
Vols directs de drones ( $l_0$ )

..... Trajet de Drone

———— Trajet de Bus





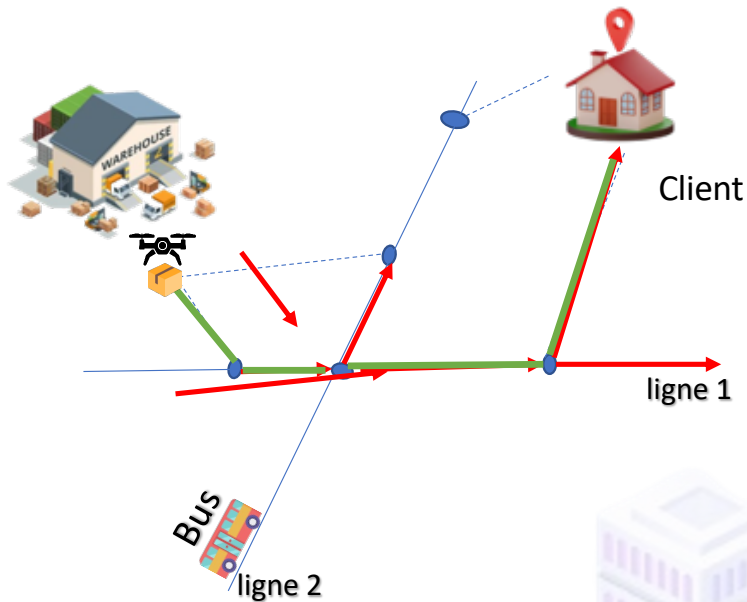
$$e_p = \sum_{(v, v', l_j, t_k) \in p_{uv}} \underbrace{(\tau(t_j, v) - \alpha(t_{j-1}, v))}_{\text{Arrival time}} e_h + \underbrace{\omega_{vv'}^{lt}}_{\text{Traversal time}} e_f$$

Departure time

$\alpha(t_j, v)$  : heure d'arrivée au nœud  $v$  en utilisant le trajet  $t_j$

$\tau(t_j, v)$  : heure de départ vers  $v$  en utilisant le trajet  $t_j$

**Modèle de consommation d'énergie avec les deux phases : vol stationnaire (*hovering*) et vol direct (*fly*)**

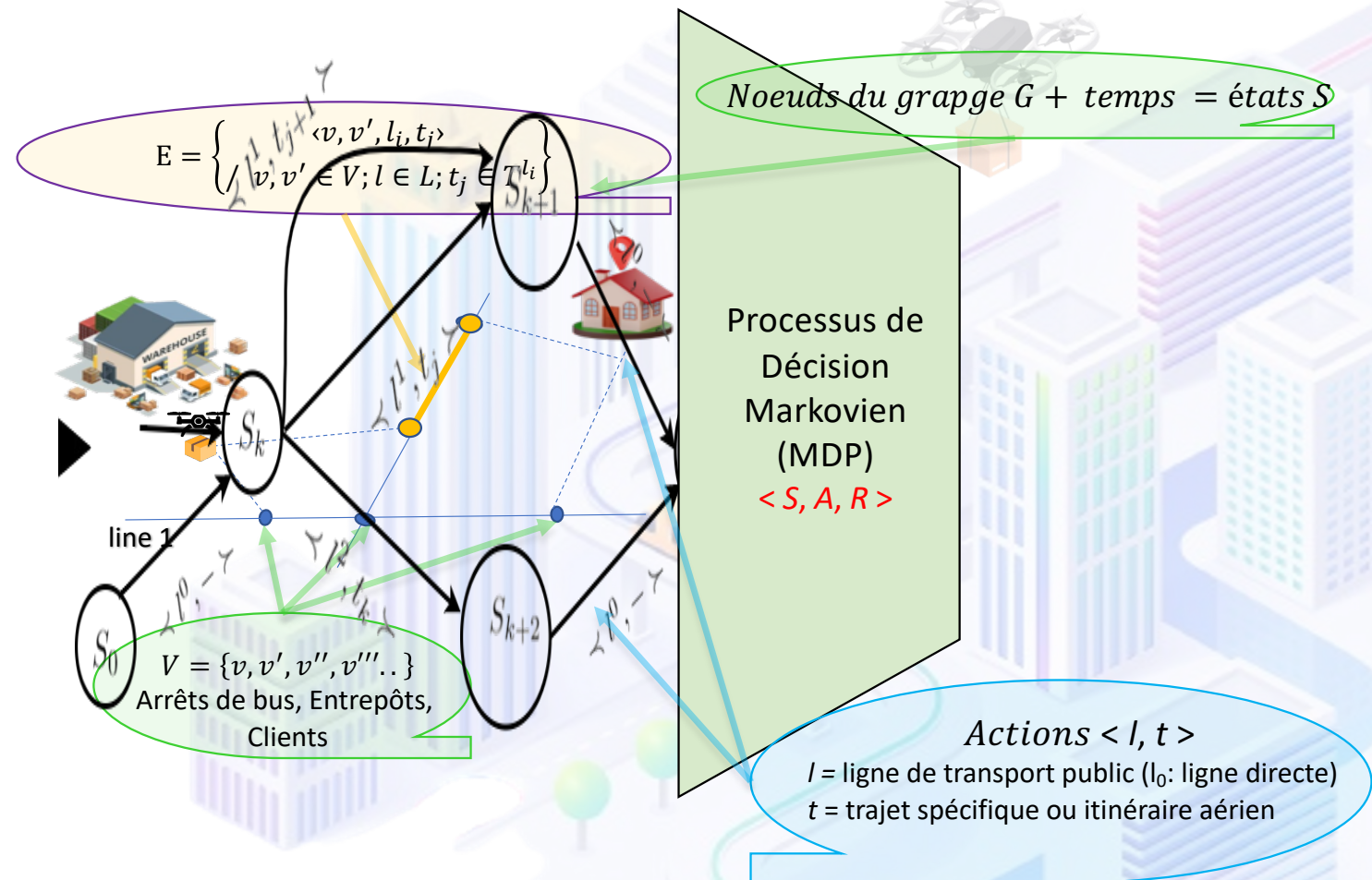


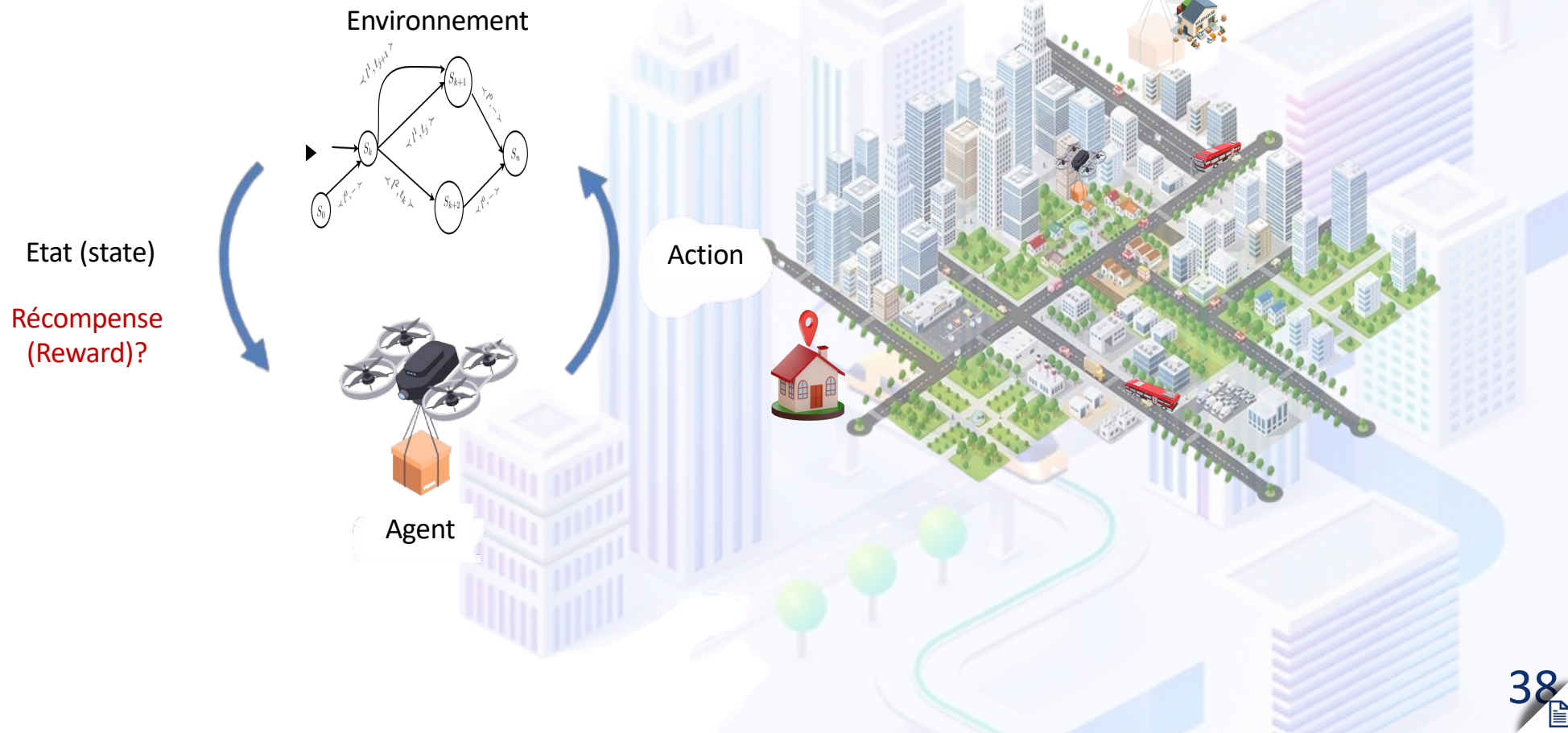
Pour un graphe  $G(V, E, \Omega)$  donné,

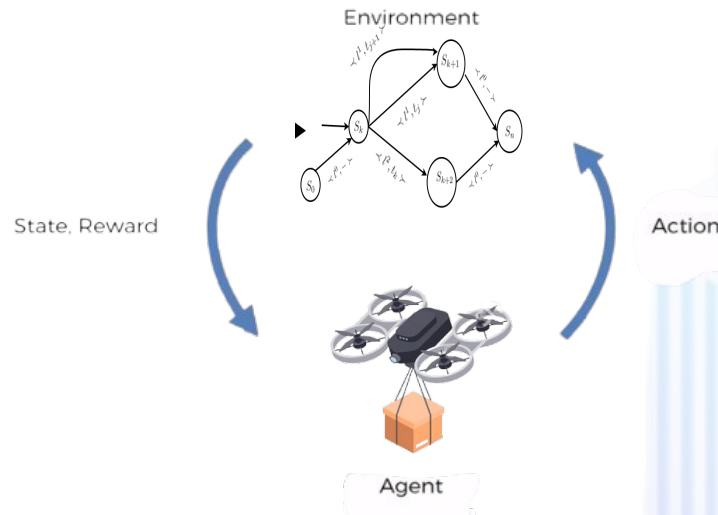
Problème : trouver la **meilleure** sélection de trajets consécutifs  $p^*$  parmi tous les trajets disponibles  $P_{uv}$  en **temps réel** (à chaque station) :

$$p^* = \arg \min_{P_{uv}} (\gamma \alpha(., v) + (1 - \gamma) e_p)$$

où  $P_{uv}$  représente tous les chemins possibles de  $u$  à  $v$ ,  $\gamma \in [0, 1]$  détermine le compromis entre l'énergie résiduelle ( $e_p$ ) et l'heure d'arrivée ( $\alpha$ ).







Fonction Reward :

$$r = \begin{cases} \gamma(W + T) + (1 - \gamma)e_h \times W & \text{if } l_i \neq l_0 \\ \gamma T + (1 - \gamma)e_f \times T & \text{if } l_i = l_0 \end{cases}$$

Mise à jour des  $Q\_value$  :

$$Q_{new}(s_k, a) = Q_{old}(s_k, a) + \lambda \left[ r + \sigma \min_{a'} Q_{old}(s_{k+1}, a') - Q_{old}(s_k, a) \right]$$

Choix de la meilleure action :

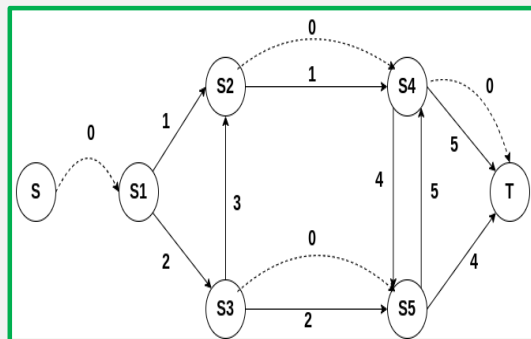
$$Q^*(s, \langle l_i, t_j \rangle) = \min_{a'} Q(s_{k+1}, a')$$

## Réseau de Transport Public



Temps des traversées

Instants de départs



## Réseau de transport utilisé

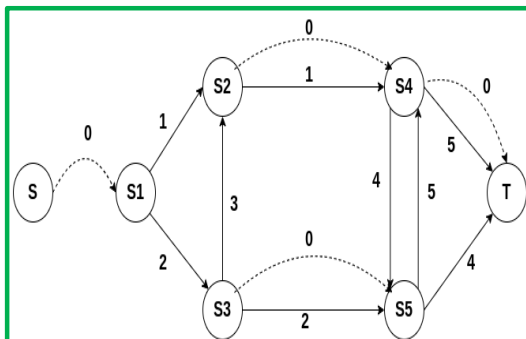
Un réseau tiré de la littérature

5 arrêts de bus, 1 entrepôt, 1 client et de 5 lignes de transport public.

[1]Huang, H., Savkin, A. V., & Huang, C. (2020). Reliable path planning for drone delivery using a stochastic time-dependent public transportation network. IEEE Transactions on Intelligent Transportation Systems, 22(8), 4941-4950.

Réseau de Trasport  
Public

Temps des traversées Instants de départs



Line	Link	Traversal times
L0	$SS_1$	3(0.1)
	$S_2S_4$	6(0.2)
	$S_4T$	2(0.1)
	$S_3S_5$	6(0.2)
L1	$S_1S_2$	13(2), 13(2), 12(2), 11(1.5), 11(1.5)
	$S_2S_4$	10(2.5), 10(2.5), 11(2), 11(2), 10(2.5)
L2	$S_1S_3$	15(1.4), 15(1.4), 16(1), 16(1)
	$S_3S_5$	10(1.3), 10(1.2), 10(1), 11(1.1)
L3	$S_3S_2$	5(1), 7(1), 7(1), 6(1.5)
L4	$S_4S_5$	7(1), 8(1.2), 8(1.2), 7(1.2), 7(1)
	$S_5T$	3(1), 3(1), 3(1.2), 4(1), 4(1)
L5	$S_5S_4$	10(2), 10(2), 10(2), 10(2)
	$S_4T$	5(1), 5(1), 5(1.2), 5(0.5)

## Distributions des temps de parcours sur les des liens

Moyenne et écarts types des temps de traversée sur les liens de bus.

Réseau de Transport  
Public

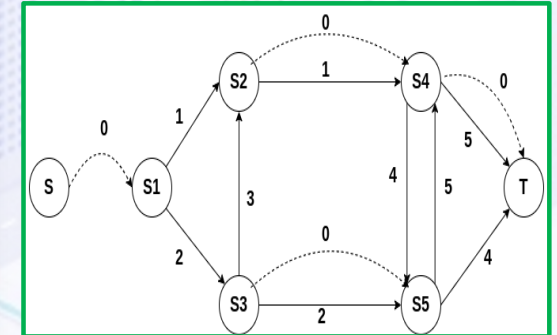
Temps des traversées

Instants de départs

### Distributions des instants de départ sur les liens

Moyenne et écarts types des instants de départ des bus

Line	Link	Departure instants
L1	$S_1S_2$	-1(1), 9(1), 19(1), 29(1), 39(1)
	$S_2S_4$	12(3), 22(3), 31(3), 40(2.5), 50(2.5)
L2	$S_1S_3$	-5(2), 2(2), 9(2), 16(1)
	$S_3S_5$	10(3.4), 17(3.4), 25(3.4), 32(2)
L3	$S_3S_2$	5(1), 17(1.5), 29(1), 41(1)
L4	$S_4S_5$	1(2), 11(2), 21(1.5), 31(1.5), 41(1.5)
	$S_5T$	8(3), 19(3.2), 29(2.7), 38(2.7), 48(2.5)
L5	$S_5S_4$	5(1), 20(1), 35(1.5), 50(1.5)
	$S_4T$	15(3), 30(3), 45(3.5), 60(3.5)





1

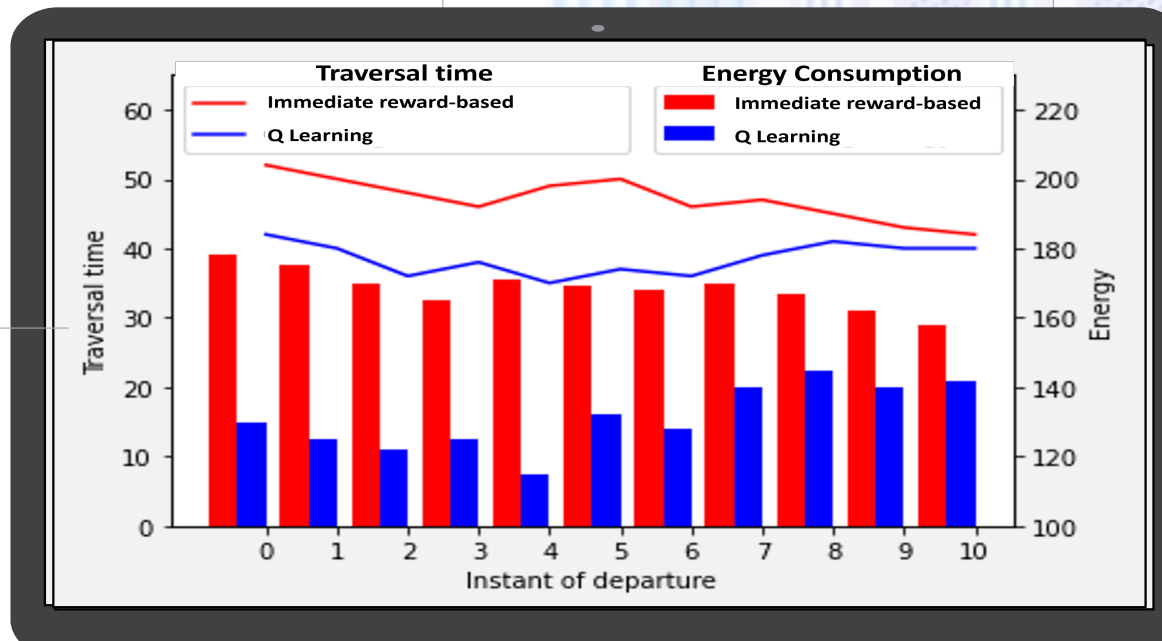
Convergence après  
seulement 1 000  
epoch.

2

Réduction de la  
consommation d'énergie  
et des temps d'arrivée  
par rapport à une  
solution d'apprentissage  
basée sur la  
récompense immédiate.

3

Atteint un taux de  
réussite de 100 %  
dans la livraison  
des colis.



### Vue d'ensemble du problème

Problème de la planification des trajets afin de minimiser la consommation d'énergie et le temps de livraison tout en garantissant l'adaptabilité aux changements aléatoires des horaires des transports publics.

### Formulation MDP et Solution Q-Learning

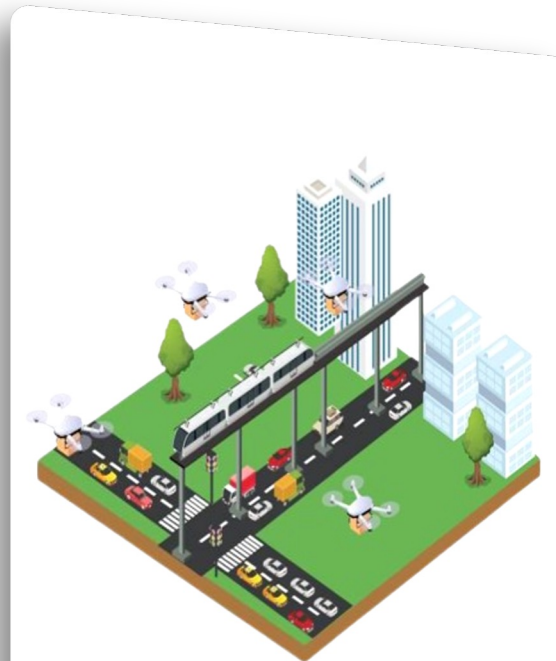
Modélisation sous forme de MDP et une solution basée sur Q-learning qui a montré son efficacité pour réduire les temps d'arrivée et la puissance tout en garantissant une arrivée à 100 % au client.

### Défis mise à l'échelle du Q-learning

Les Q-tables sont généralement stockées dans chaque serveur d'entrepôt, ce qui réduit les risques liés au partage des données mais rend plus difficile l'adaptation à l'augmentation du nombre de clients.

### Défis de la collaboration

Comme les agents sont indépendants, ils ratent des occasions d'explorer et de collaborer avec d'autres drones !



**Contribution 2**

Fed-DDQN pour la planification collaborative d'itinéraires



## Motivations

La solution Federated Double Deep Q-Network

Résultats et Analyses



states	actions			
	$a_0$	$a_1$	$a_2$	...
$s_0$	$Q(s, a)$	$Q(s, a)$	$Q(s, a)$	...
$s_1$	$Q(s, a)$	$Q(s, a)$	$Q(s, a)$	...
$s_2$	$Q(s, a)$	$Q(s, a)$	$Q(s, a)$	...
⋮	⋮	⋮	⋮	⋮

Q-table

states	actions			
	$a_0$	$a_1$	$a_2$	...
$s_0$	$Q(s, a)$	$Q(s, a)$	$Q(s, a)$	...
$s_1$	$Q(s, a)$	$Q(s, a)$	$Q(s, a)$	...
$s_2$	$Q(s, a)$	$Q(s, a)$	$Q(s, a)$	...
⋮	⋮	⋮	⋮	⋮

Q-table

## Motivations

La solution Federated Double Deep Q-Network

Résultats et Analyses



states	actions			
	$a_0$	$a_1$	$a_2$	...
$s_0$	$Q(s, a_0)$	$Q(s, a_1)$	$Q(s, a_2)$	...
$s_1$	$Q(s, a_0)$	$Q(s, a_1)$	$Q(s, a_2)$	...
$s_2$	$Q(s, a_0)$	$Q(s, a_1)$	$Q(s, a_2)$	...
⋮	⋮	⋮	⋮	⋮

states	actions			
	$a_0$	$a_1$	$a_2$	...
$s_0$	$Q(s, a_0)$	$Q(s, a_1)$	$Q(s, a_2)$	...
$s_1$	$Q(s, a_0)$	$Q(s, a_1)$	$Q(s, a_2)$	...
$s_2$	$Q(s, a_0)$	$Q(s, a_1)$	$Q(s, a_2)$	...
⋮	⋮	⋮	⋮	⋮

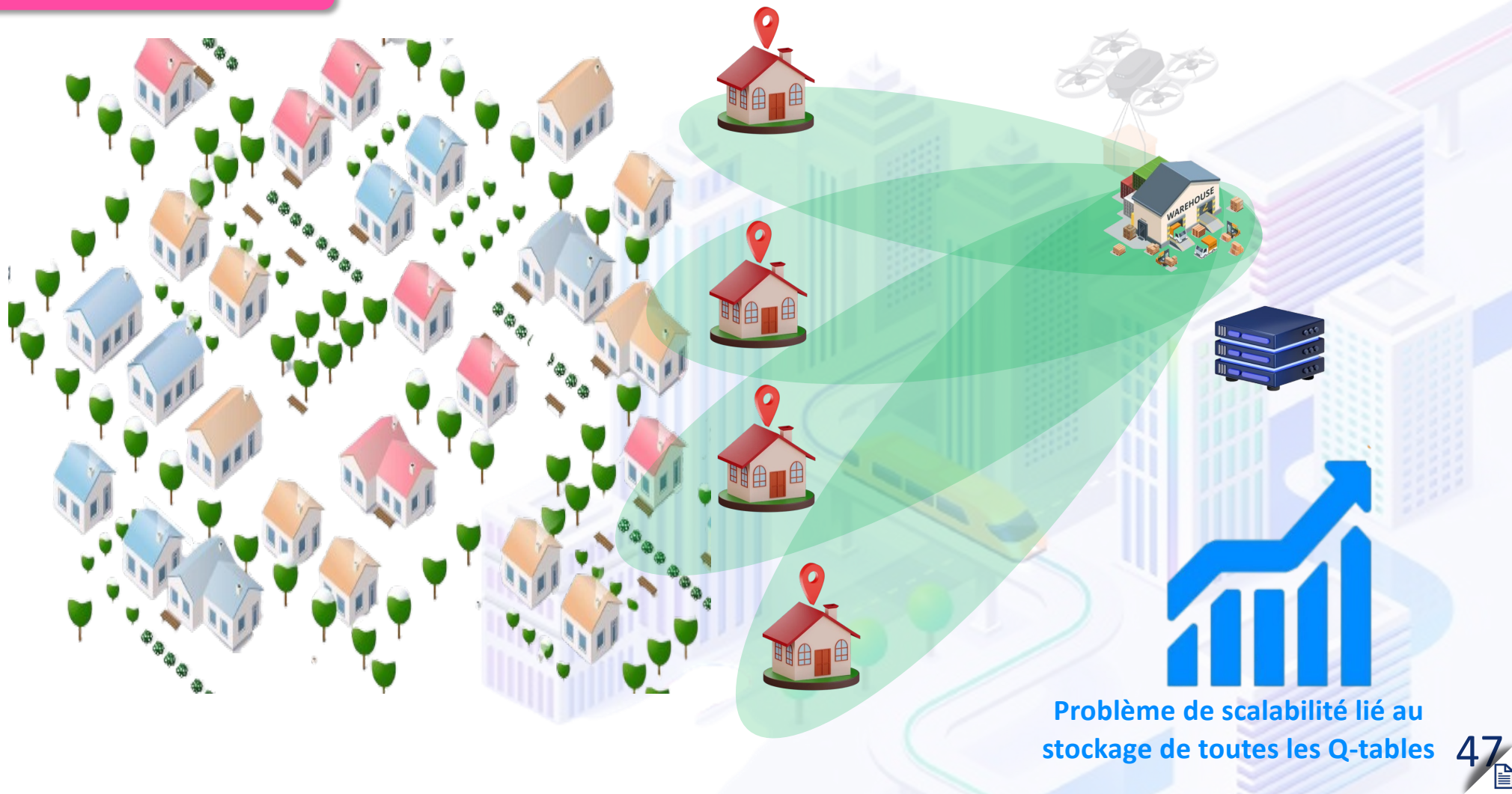
states	actions			
	$a_0$	$a_1$	$a_2$	...
$s_0$	$Q(s, a_0)$	$Q(s, a_1)$	$Q(s, a_2)$	...
$s_1$	$Q(s, a_0)$	$Q(s, a_1)$	$Q(s, a_2)$	...
$s_2$	$Q(s, a_0)$	$Q(s, a_1)$	$Q(s, a_2)$	...
⋮	⋮	⋮	⋮	⋮

states	actions			
	$a_0$	$a_1$	$a_2$	...
$s_0$	$Q(s, a_0)$	$Q(s, a_1)$	$Q(s, a_2)$	...
$s_1$	$Q(s, a_0)$	$Q(s, a_1)$	$Q(s, a_2)$	...
$s_2$	$Q(s, a_0)$	$Q(s, a_1)$	$Q(s, a_2)$	...
⋮	⋮	⋮	⋮	⋮

**Motivations**

La solution Federated Double Deep Q-Network

Résultats et Analyses

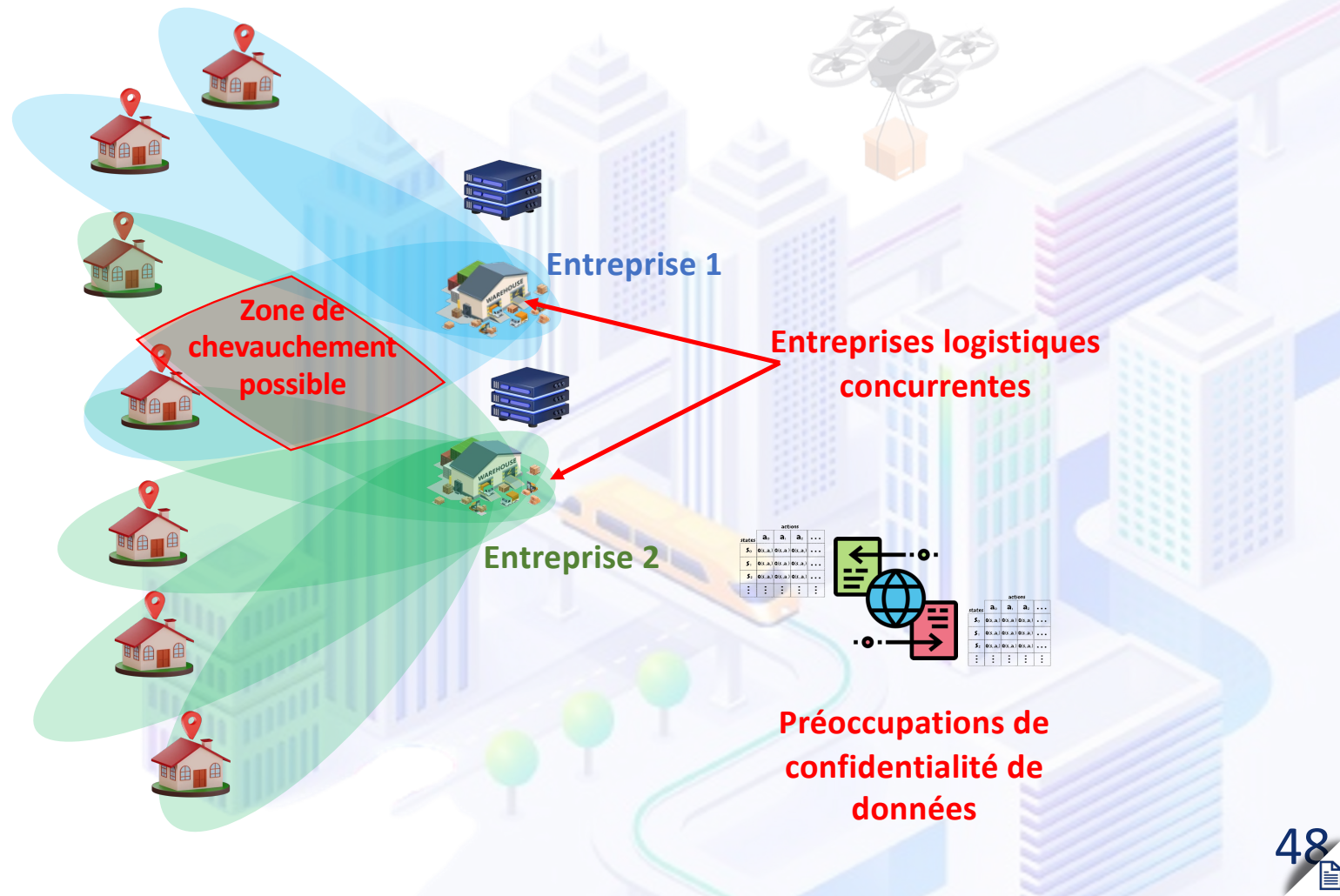


Problème de scalabilité lié au stockage de toutes les Q-tables

**Motivations**

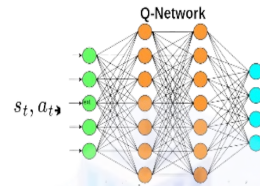
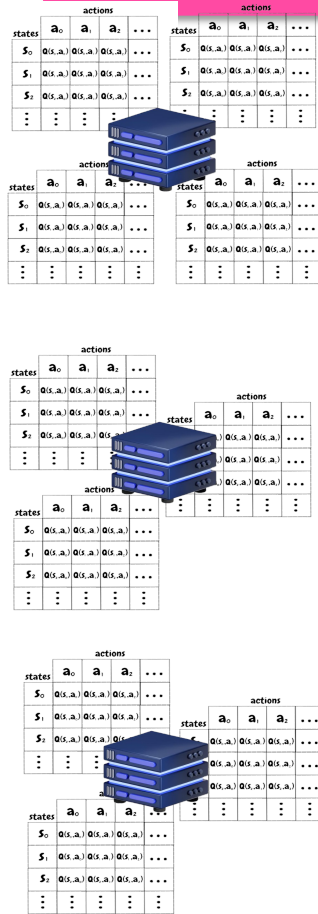
La solution Federated Double Deep Q-Network

Résultats et Analyses

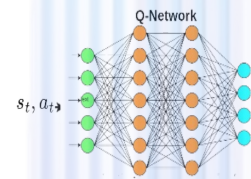


**La solution Federated Double Deep Q-Network**

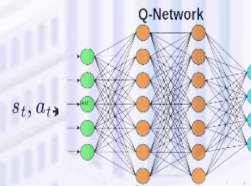
**Vue générale**



$Q_\pi(s, a; \theta_1)$



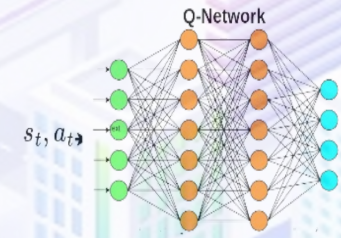
$Q_\pi(s, a; \theta_1)$



$Q_\pi(s, a; \theta_1)$

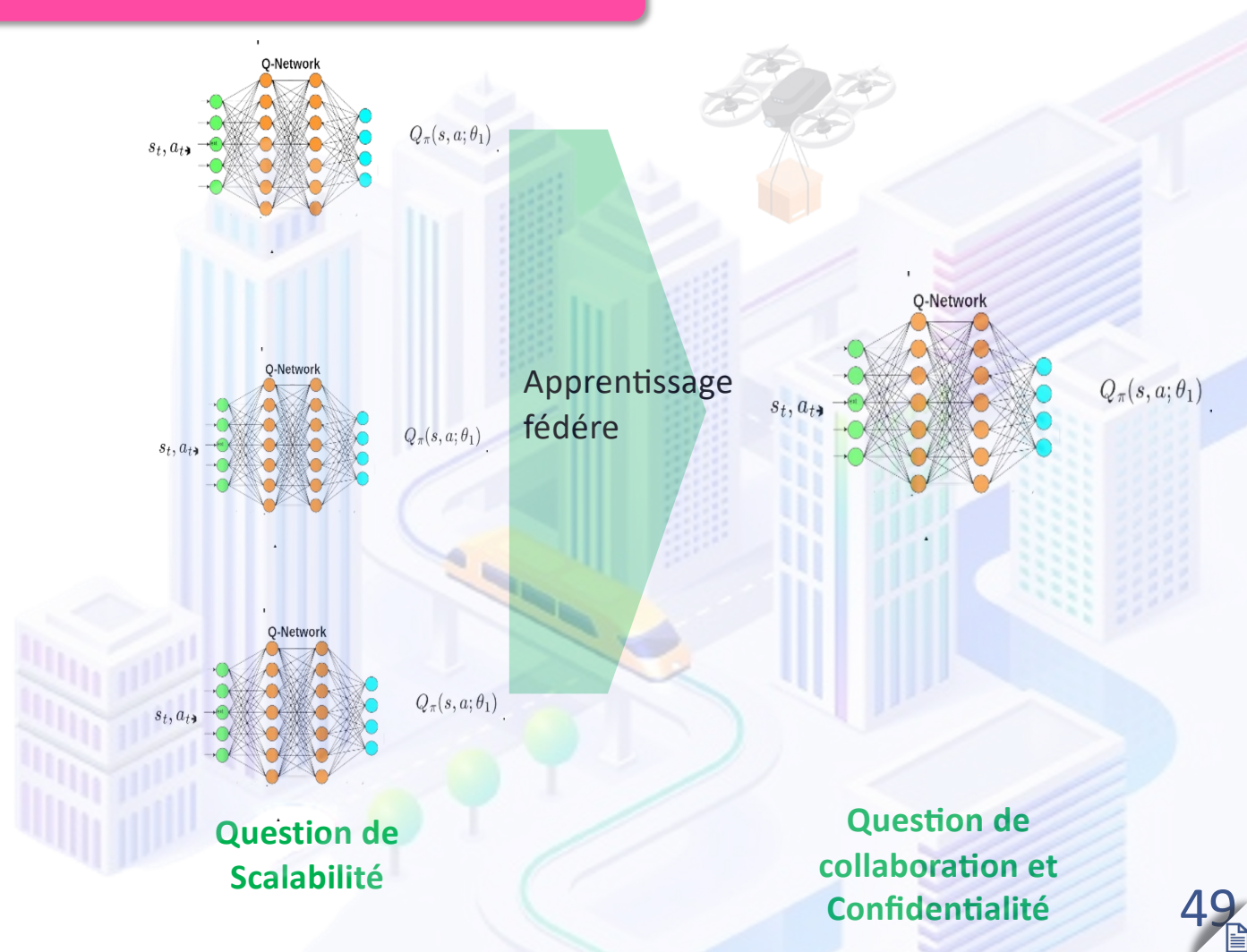
**Question de Scalabilité**

**Apprentissage fédéré**



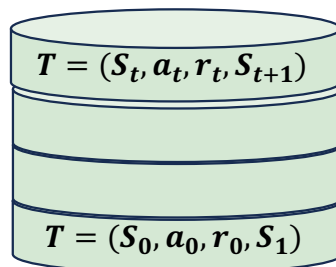
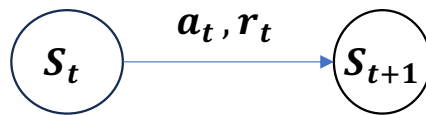
$Q_\pi(s, a; \theta_1)$

**Question de collaboration et Confidentialité**



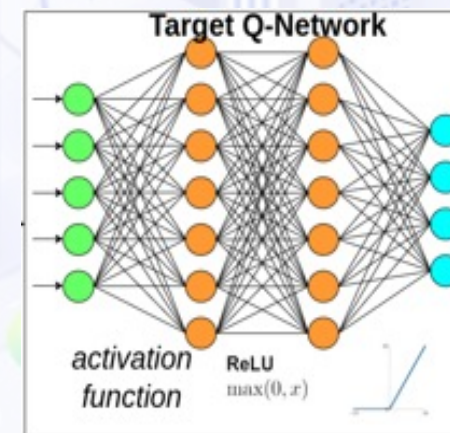
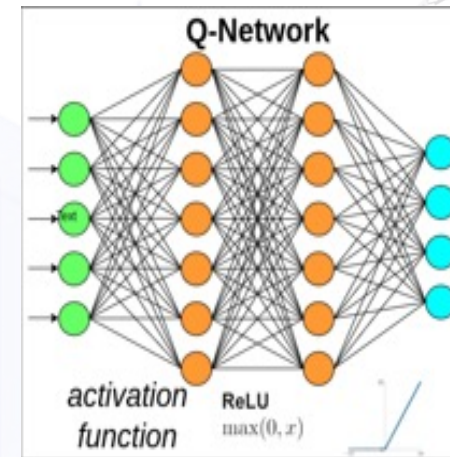


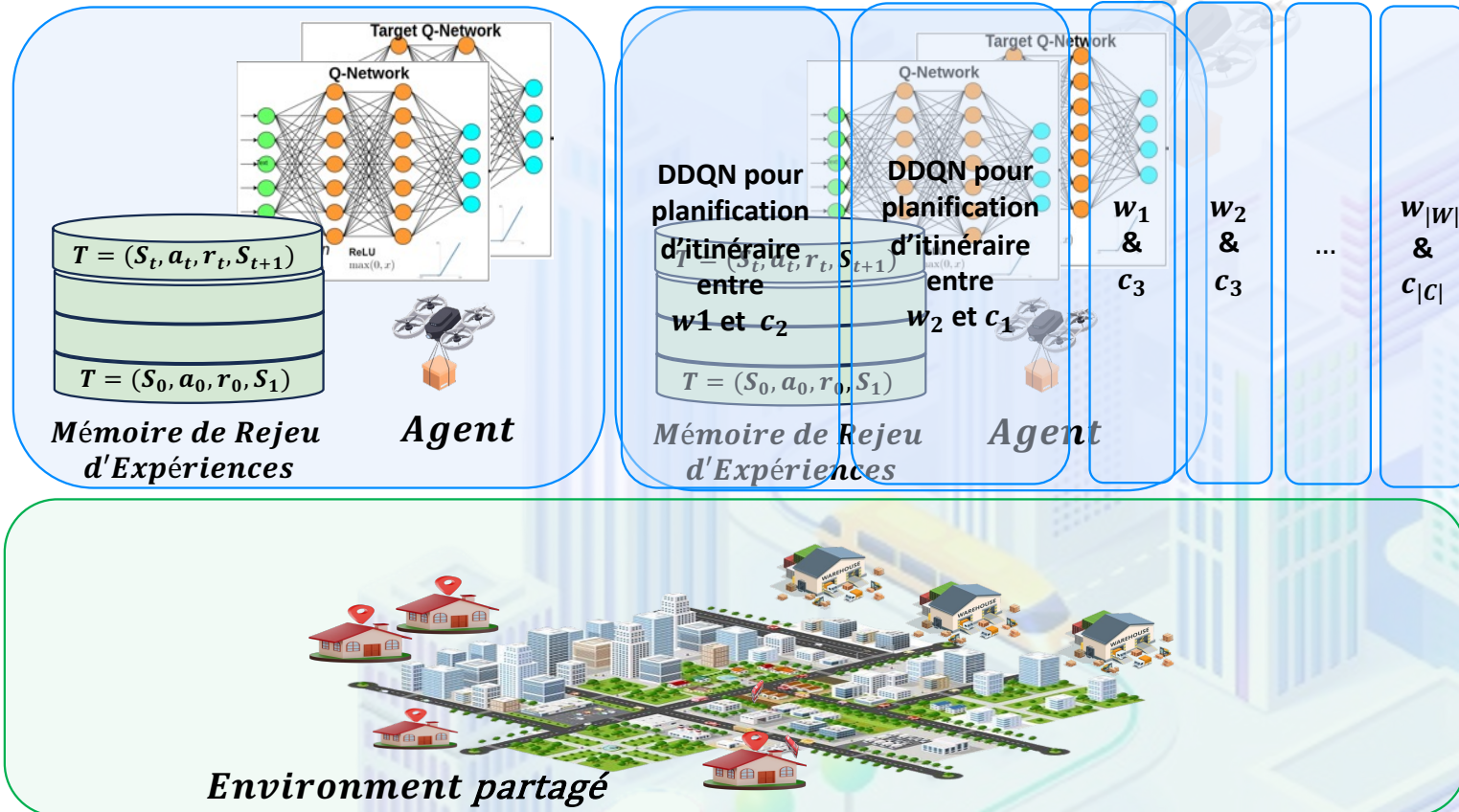
### Problème du biais dans le Q-Learning classique



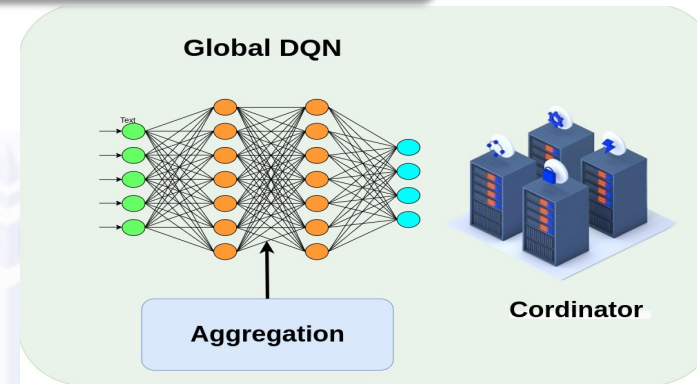
*Mémoire de Rejeu  
d'Expériences*

Répétition jusqu'à convergence du modèle

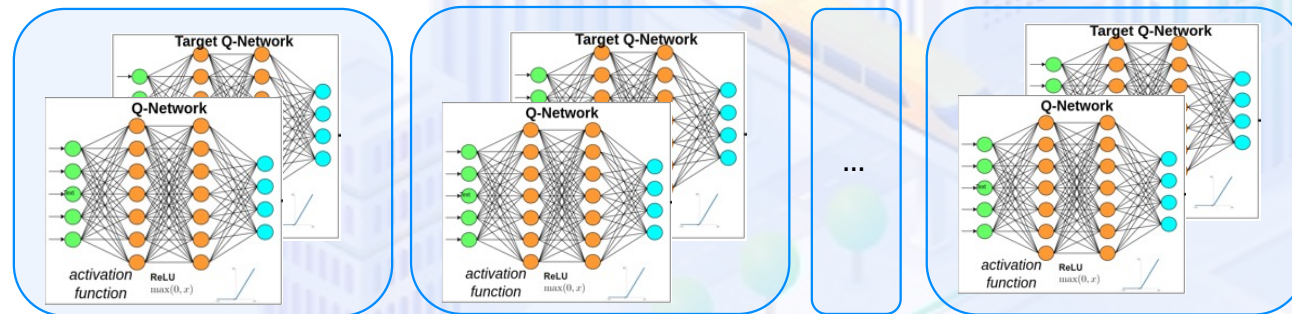




Planification d'itinéraires avec plusieurs drones à l'aide de DDQN indépendants 51

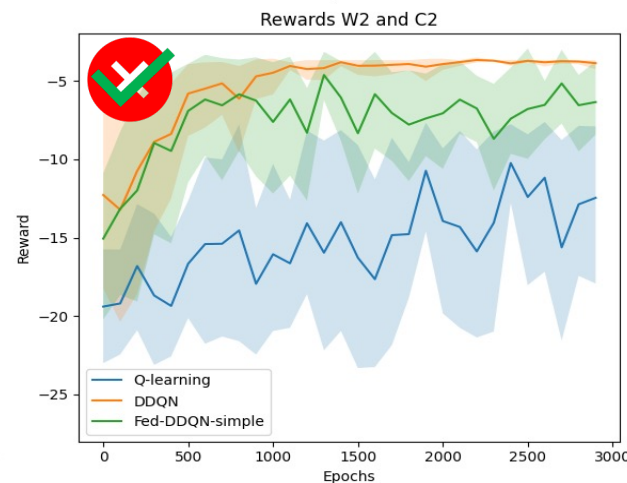
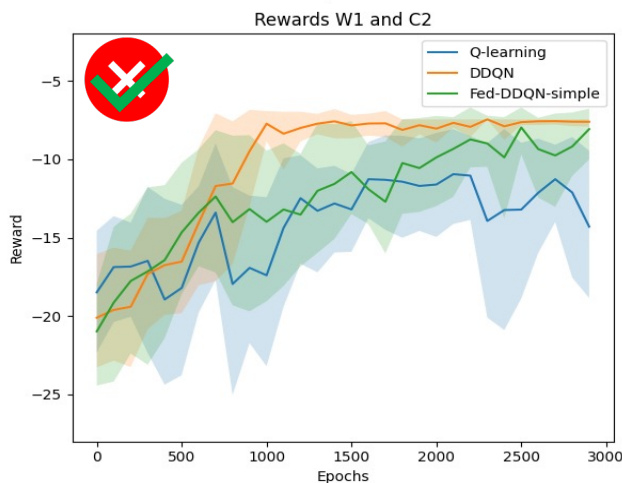
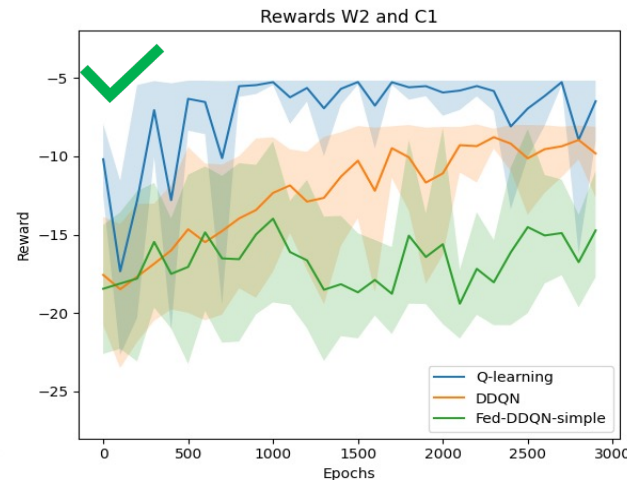
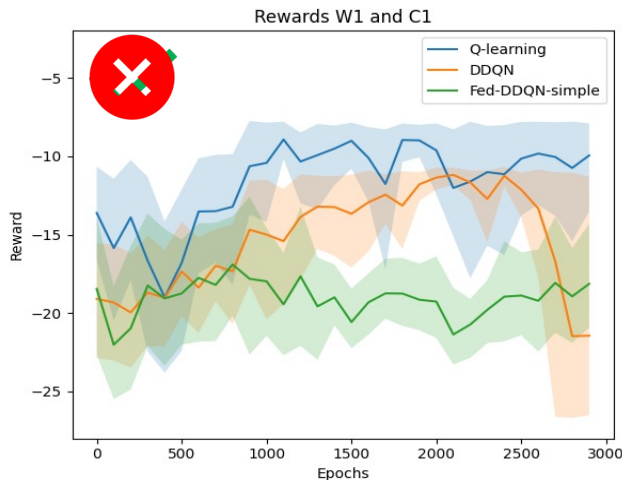


Répétition jusqu'à ce que les modèles de tous les drones convergent



*Environnement partagé*

Fed-DDQN pour la planification collaborative d'itinéraires des drones



Convergence de Q-Learning, DDQN et Fed-DDQN en fonction du nombre d'épisodes pour différents configuration clients - entrepôts.

La convergence du Q-learning dépend de la position de W et de C. Certaines paires convergent, tandis que d'autres ne convergent pas.

Lorsqu'il converge, il obtient des récompenses plus élevées car il utilise une valeur de Q-table exacte.

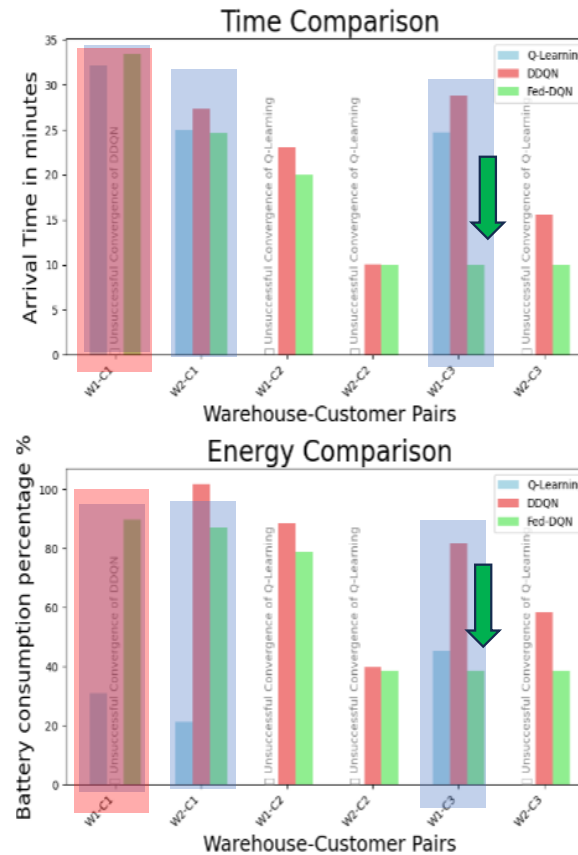
Lorsqu'il ne converge pas, c'est que les drones ne collaborent pas, ce qui nécessite plus de temps d'entraînement.

DDQN atteint la convergence dans la plupart des configurations, sauf dans le cas 1.

DDQN ne converge pas ne connaissant pas tout son environnement

Fed-DDQN converge pour toutes les configurations

Ceci met en évidence l'importance de la collaboration entre drones dans l'exploration de l'environnement.



Lorsque Q-learning converge, il surpasse les autres algorithmes.

DDQN et Fed-DDQN fournissent de bons résultats pour toutes les paires, hormis W1C1 où DDQN ne converge pas.

Fed-DDQN est plus performant que DDQN dans la plupart des cas : Pour w1-c3, Fed-DDQN fournit des chemins qui sont environ 50 % meilleurs que ceux de DDQN

Ceci montre l'intérêt de la collaboration entre drones

Les performances de l'apprentissage Q, du double DDQN et du Fed-DDQN en termes de respect des délais, d'efficacité énergétique et de réduction des interférences.

Étude comparative de la consommation d'énergie et du délai de livraison

1

Q-Learning-based Time-adapted Early Arrival Path

2

Planification d'itinéraires multi-drones scalable et respectueuse de la confidentialité

4

01 RODEF 2023

Rahmani, M., et al. **Time-adapted Early Arrival Path for Drone Parcel Delivery through Public Transportation Vehicles: Using Q-learning**  
<https://roadef2023.sciencesconf.org/435776/document>

02 ICC2023

Rahmani, M., et al. **Q-Learning-based Time-adapted Early Arrival Path Algorithm for Drone Delivery Using Public Transport**  
[10.1109/ICC45041.2023.10279648](https://doi.org/10.1109/ICC45041.2023.10279648)

03 IEEE ITS

Rahmani, M., et al. **Toward Sustainable Last-Mile Deliveries: A Comparative Study of Energy Consumption and Delivery Time for Drone-Only and Drone-Aided Public Transport Approaches in Urban Areas**  
[10.1109/TITS.2024.3408476](https://doi.org/10.1109/TITS.2024.3408476)

04 IEEE ITS (2° revision)

Rahmani, M., et al. **Q-learning-based Multi-objective Path Planning for UAV Parcel Delivery Through Public Transportation Vehicles.**

05 Journal Draft paper

Rahmani, M., et al. **Scalability and Privacy Aware Federated DDQN for Multi-UAV Path Planning in Last-Mile Delivery via Public Vehicles.**



**MERCI  
POUR VOTRE  
ATTENTION**