

Fil rouge

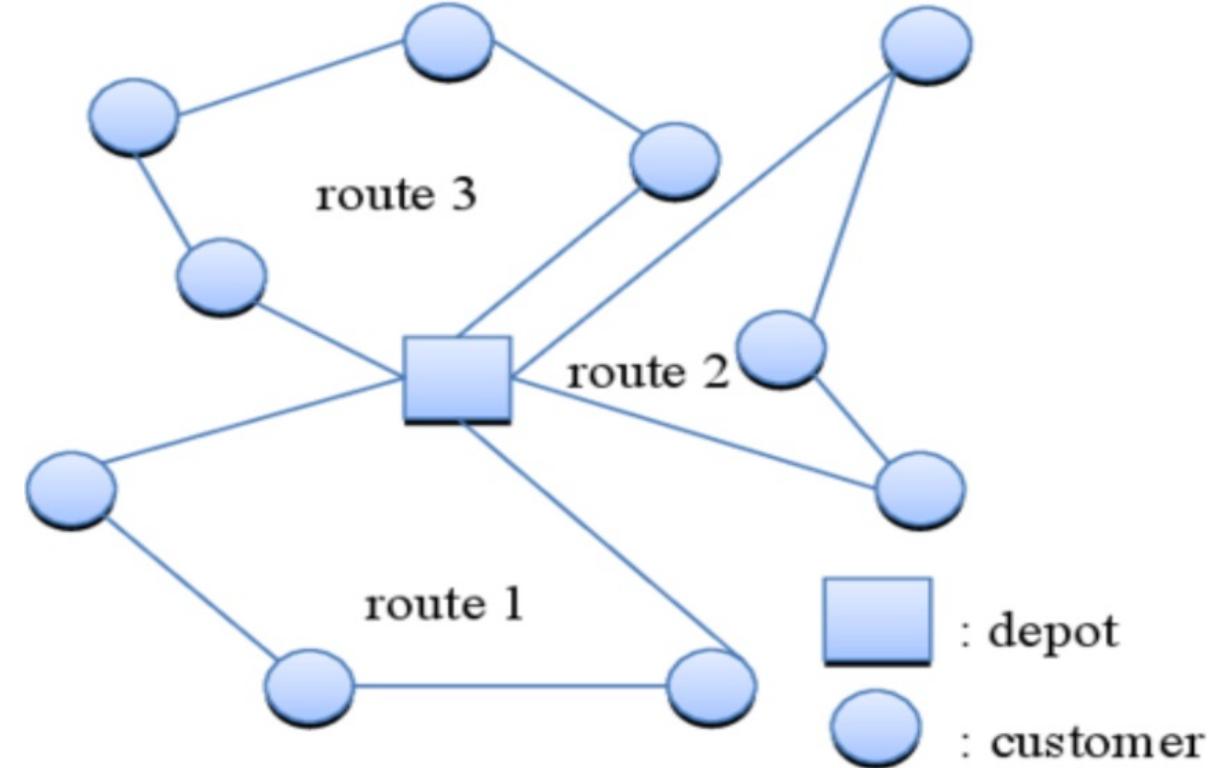
VRP : Vehicle routing problem

Electif ICO

S. HAMMADI- H. ZGAYA-BIAU, S. BEN OTHMAN, P. YIM

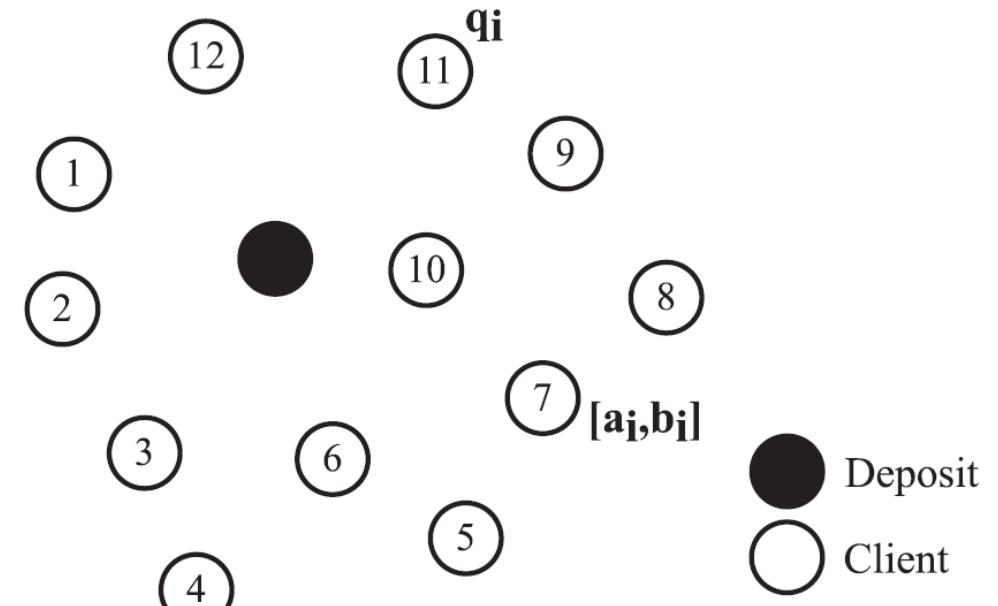
Problème de tournées de véhicules

- Le problème de tournées de véhicule est un problème de combinaison nœud-service à **routes multiples** avec **limitation de la capacité** du véhicule.
- VRP est classé comme un problème NP-difficile.
- VRP est une généralisation du problème du voyageur de commerce (TSP).



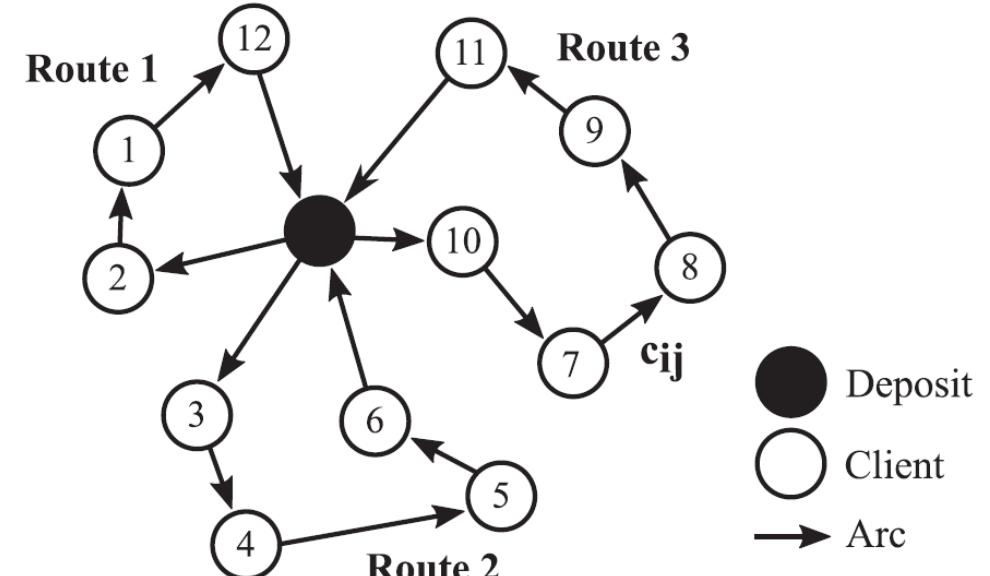
Exemple : Problème de tournées de véhicules

- $K=\{k:k=1,2,\dots,|K|\}$: un ensemble des véhicules situés dans un seul dépôt
- $C=\{i:i=1,2,\dots,N\}$: un ensemble de clients répartis géographiquement.
- Tous les véhicules ont la même capacité Q .
- Chaque client i a une demande donnée q_i et doit être servi dans un intervalle de temps spécifiée $[a_i, b_i]$.



Exemple : Problème de tournées de véhicules

- c_{ij} : le coût du voyage entre le client i et le client j .
- La solution $x = [0, 2, 1, 12, 0, 3, 4, 5, 6, 0, 10, 7, 8, 9, 11, 0]$.
- L'indice 0 : le dépôt et les trois routes de cette solution sont : route1 = [0, 2, 1, 12], route2 = [0, 3, 4, 5, 6] et route3 = [0, 10, 7, 8, 9, 11, 0].
- La solution x est également décrite comme: [route1, route2, route3].



Exemple : Problème de tournées de véhicules

- L'objectif est de déterminer un ensemble de routes afin de minimiser le coût total. Chaque route est associé à un seul véhicule.
- le coût d'une solution x est calculé comme suit:

$$f(x) = \omega K(x) + \sum_{(i,j) \in E} c_{ij} \quad (1)$$

où:

- c_{ij} : coût entre clients (i, j), qui peut être lié à la distance entre les clients;
- E : ensemble d'arcs appartenant à la solution x ;
- $K(x)$: nombre de véhicules dans la solution x ;
- ω : un facteur de pénalité arbitraire non négatif important.

Optimisation : Métaheuristiques

- La combinaison de deux ou plusieurs métaheuristiques pour résoudre des problèmes d'optimisation est de plus en plus utilisée.
- L'objectif principal de l'hybridation des métaheuristiques est :
 - choisir les meilleures caractéristiques de chaque métaheuristique pour résoudre un POC;
 - Obtenir une meilleure qualité de solution dans un temps plus court;
 - Augmenter la capacité à affronter des problèmes plus complexes.

Systèmes multi-agents

Principales stratégies de l'hybridation

Les systèmes multi agents (SMA) sont utilisés comme liaison entre différentes métaheuristiques pour résoudre des problèmes d'optimisation (approche coopérative, collaborative, parallélisme, etc).

SMA avec décomposition de l'espace de recherche : en répartissant les contraintes et les variables du problème entre les agents, chaque agent est chargé d'optimiser sa perspective locale et, au final, les solutions partielles sont combinées pour une solution globale.

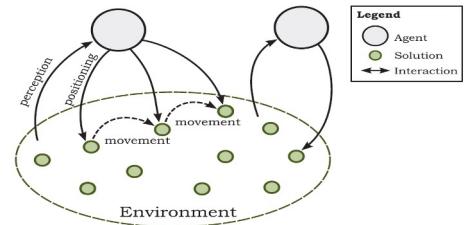
SMA avec décomposition métáheuristique : les éléments qui composent les métáheuristiques sont définis comme des agents. Ces éléments peuvent être des heuristiques de bas niveau, des stratégies de recherche, des solutions, une population ou une partie de population et/ou des particules.

SMA avec agents métáheuristiques : considère chaque métáheuristique comme un agent autonome et propose d'explorer les avantages de l'utilisation de systèmes multi-agents pour affiner et combiner les solutions de ces métáheuristiques.

Système multi-agents

- ***Mode de fonctionnement pour assurer la diversité et le partage des informations:***
- Chaque agent est responsable de l'exécution de sa propre tâche en même temps, de l'utilisation des solutions fournies par d'autres agents pour améliorer ses propres solutions.
- Les agents interagissent et travaillent ensemble pour atteindre un objectif prédéfini.
- L'interaction entre les différents agents se produit via un pool de solutions.
- La communication doit être contrôlée par les règles d'accès au pool.
→ chaque agent est doté d'une heuristique / métaheuristique et a pour fonction de rechercher la solution pour un POC donné (VRP).

Interaction Agent-Environnement



- Pendant le processus de recherche de la solution, les agents de la structure doivent passer par l'environnement système multi-agents.

Perception de l'environnement: capacité des agents à accéder aux informations sur le problème;

Positionnement: capacité des agents à définir leurs positions dans l'environnement, soit par la construction d'une nouvelle solution, soit par le choix de solutions déjà disponibles;

Déplacer: capacité de l'agent à se déplacer, d'une solution à une autre dans l'environnement. Le déplacement comprend ici toutes sortes de modifications de solution (structures de voisinage, opérateurs) qui permettent à l'agent de passer d'une solution à une autre;

Coopération: capacité de l'agent à partager et à apporter des solutions aux autres agents du système.

Système multi agents

Les éléments qui peuvent composer le système multi agents dans le modèle VRP

Environnement: défini principalement par l'espace de recherche du problème abordé → Il fournit toutes les informations nécessaires pour résoudre le problème (le nombre de clients à assister, la distance entre les clients, le nombre de véhicules, etc.)

Pool of Solutions: responsable du maintien des solutions partagées à tous les agents;

Agents mét-heuristiques: chargés de guider la recherche de la solution (Agent de recherche local).

Coopération via un pool de solutions partagé

Fonction d'évaluation

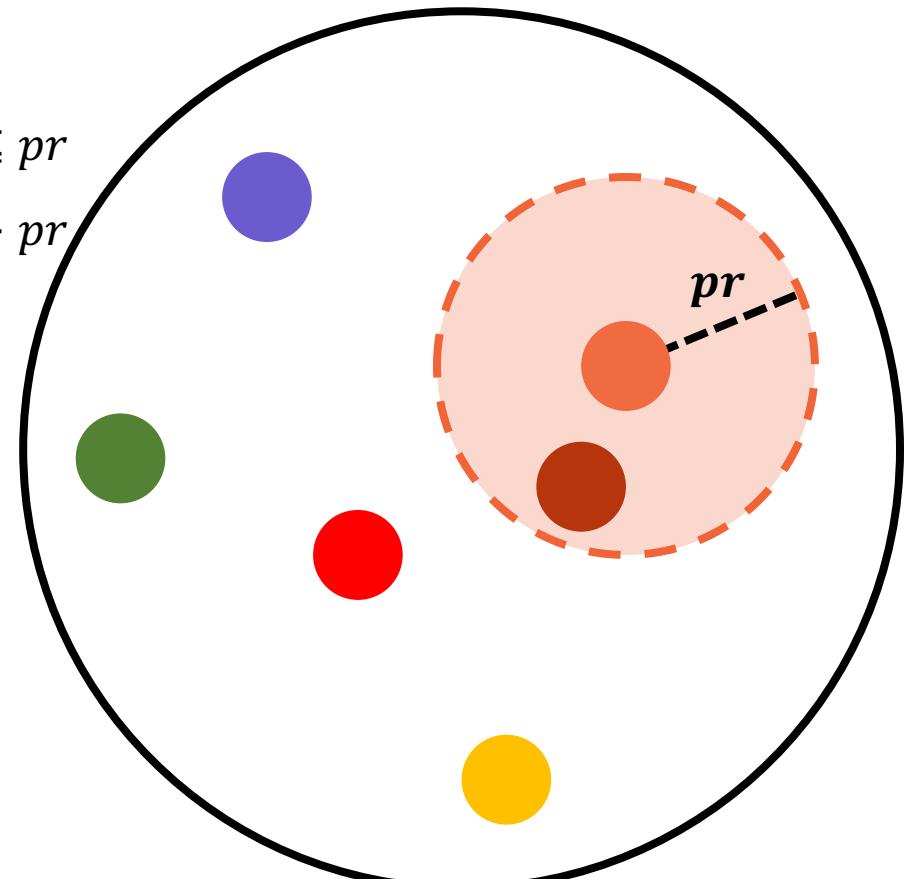
$$g(\phi_i) = \sum_{j=1}^P \phi(\lambda_{ij}) \quad \text{où} \quad \phi(\lambda_{ij}) = \begin{cases} 1 - \frac{\lambda_{ij}}{pr} & \text{si } \lambda_{ij} \leq pr \\ 0 & \text{si } \lambda_{ij} > pr \end{cases}$$

pr (pool radius) : contrôle la **dispersion** des solutions

λ_{ij} : représente à quel point les 2 solutions sont **similaires**

Ainsi, $\phi(\lambda_{ij})$ représente la **distance entre 2 solutions**

Le but de cette fonction d'évaluation est d'**assurer la diversité** dans l'ensemble des solutions (pool)



Collaboration : Deux protocoles d'interaction

Interaction Ennemis :

- Les agents n'ont pas accès aux solutions trouvées par les autres agents mais seulement aux valeurs de meilleurs critères (meilleurs temps d'attente) calculé par les autres agents.
- Chaque agent va essayer de trouver la meilleure solution.

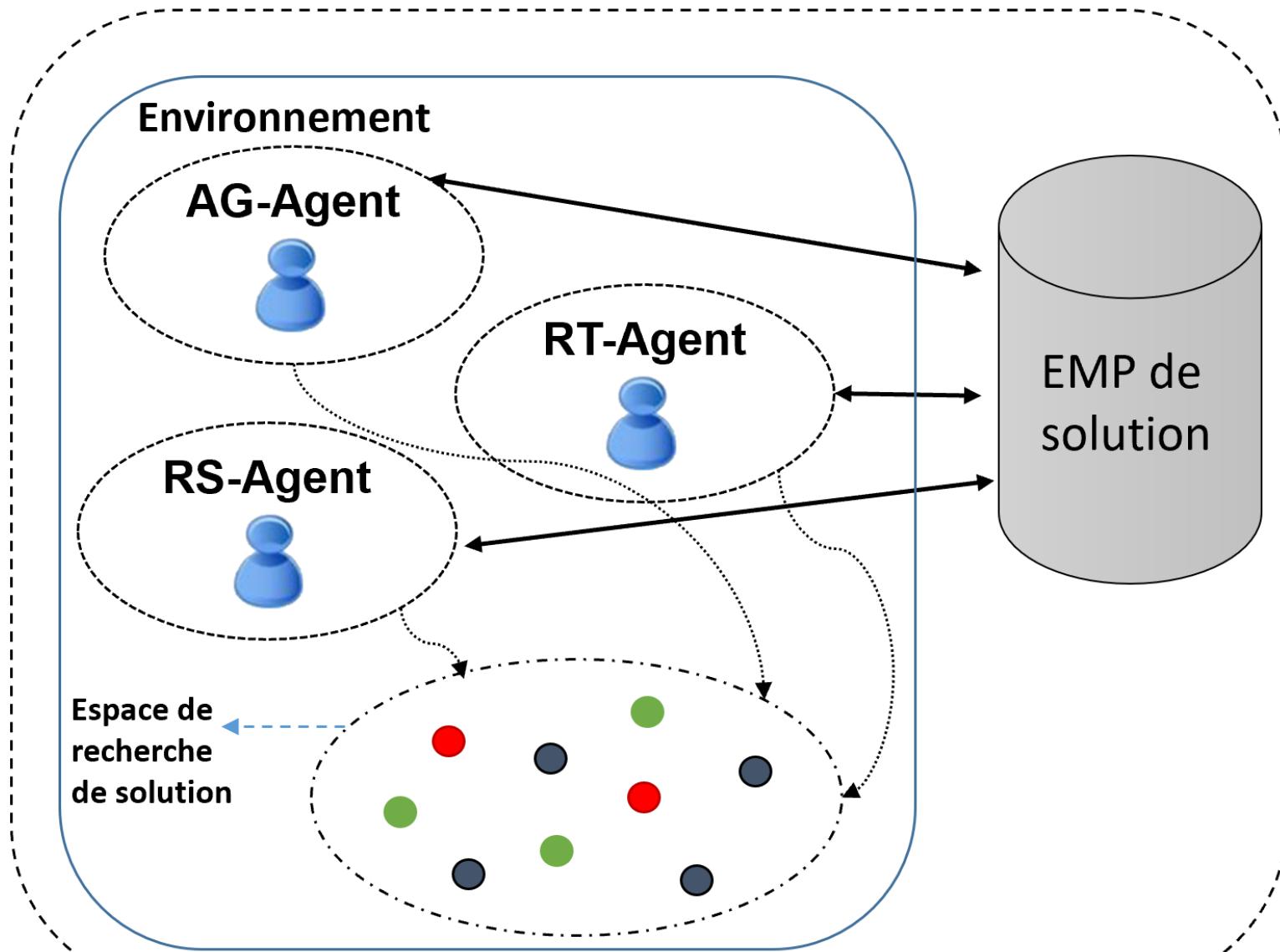


Interaction Amis :

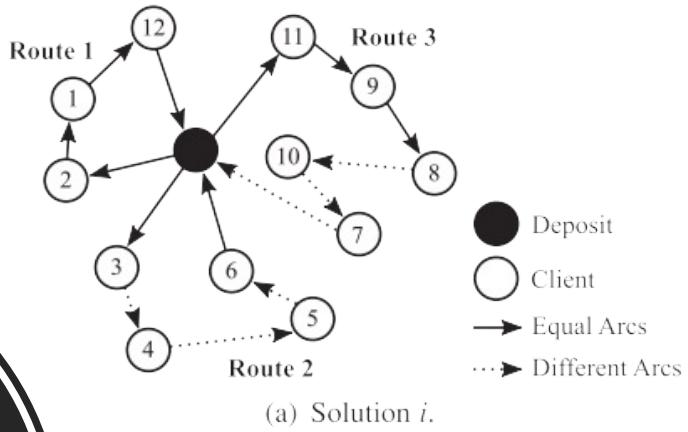
- Un échange d'informations spécifiques entre les agents dans l'environnement de recherche de solutions.
- Les informations partagées dans ce mode sont la solution complète du problème et sa valeur de critère.
- Les solutions trouvées par les agents, sont conservées dans l'espace mémoire partagé (EMP), ces solutions sont utilisées par les autres agents.



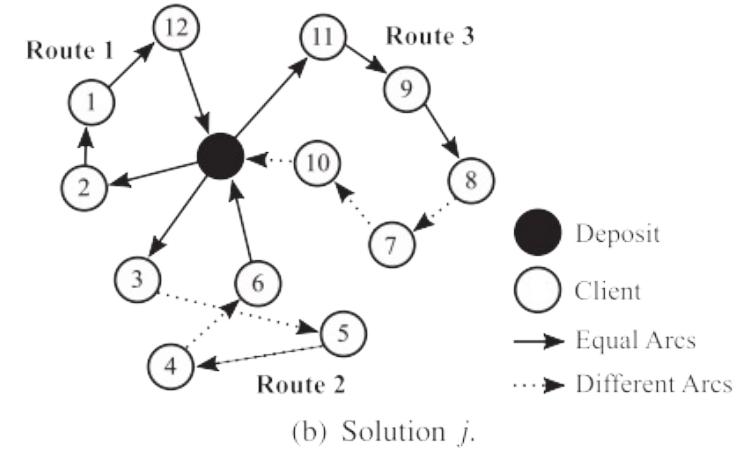
Comment garantir la diversité de l'EMP (pool des solutions) de solution ?



Exemples de calcul de λ_{ij}

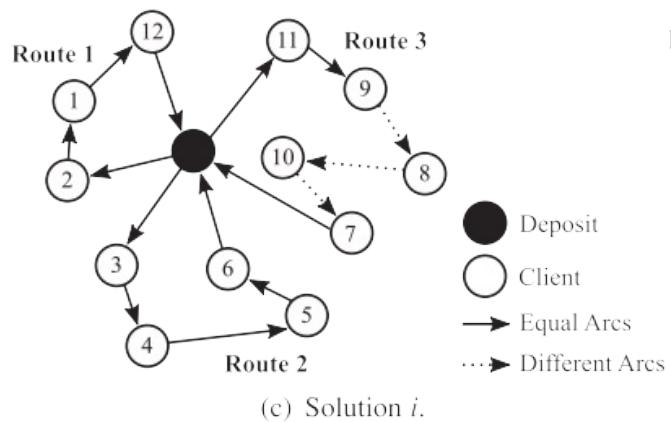


(a) Solution i .

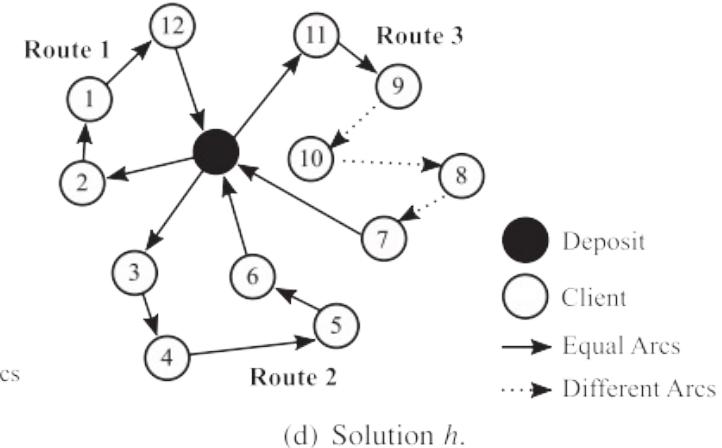


(b) Solution j .

$\lambda_{ij}=12$
12 arcs non-communs entre les solutions



(c) Solution i .



(d) Solution h .

$\lambda_{ij}=6$
6 arcs non-communs entre les solutions

Apprentissage

Apprentissage par Renforcement

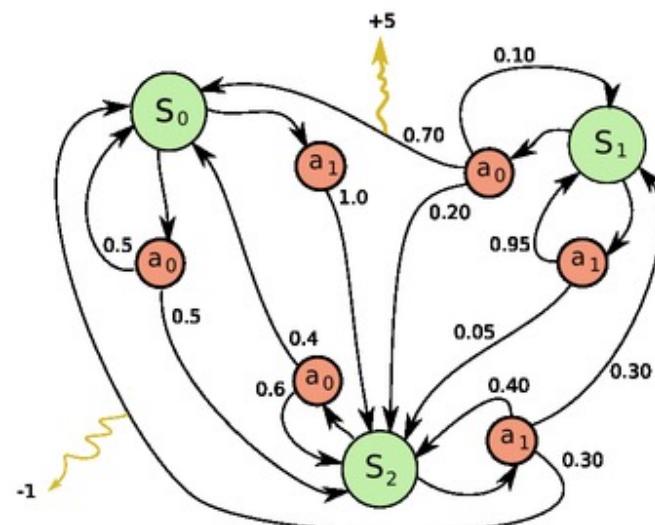
- Permettre à l'agent de modifier ses actions en fonction des expériences acquises dans l'interaction avec les autres agents et avec l'environnement.
- Utiliser les concepts de l'apprentissage par renforcement (RL), plus spécifiquement l'algorithme Q-Learning.

Note: Pour le VRP, l'apprentissage par renforcement peut être utilisé pour définir l'ordre d'application des structures de voisinage de la recherche locale.

Les processus décisionnels de Markov

- L'interaction agent-environnement est formalisée grâce à un processus de décision de Markov (PDM)
- Le PDM est considéré comme l'architecture de base de l'apprentissage par renforcement
- Un PDM est un processus stochastique qui satisfait la propriété de Markov (Bellman 1957), (Puterman 2014) (**toute l'information utile pour la prédiction du futur est contenue dans l'état présent du processus**).

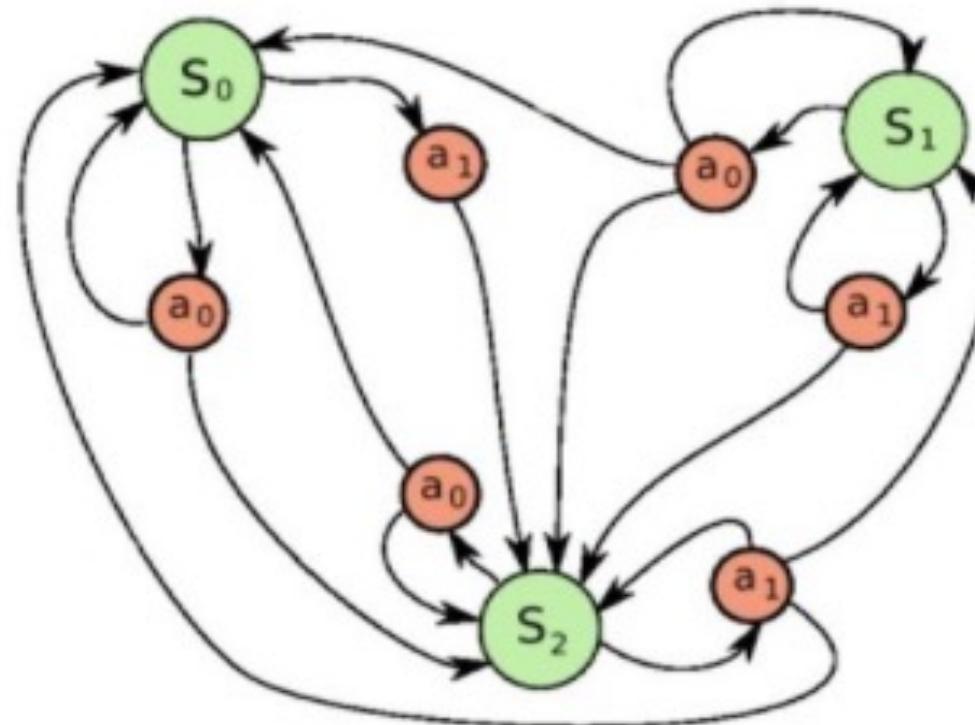
*Processus de décision
de Markov PDM*



*Architecture de base de
l'apprentissage par renforcement*

Politique : $(S_0, a_0), (S_1, a_0), (S_2, a_1)$

Les processus décisionnels de Markov



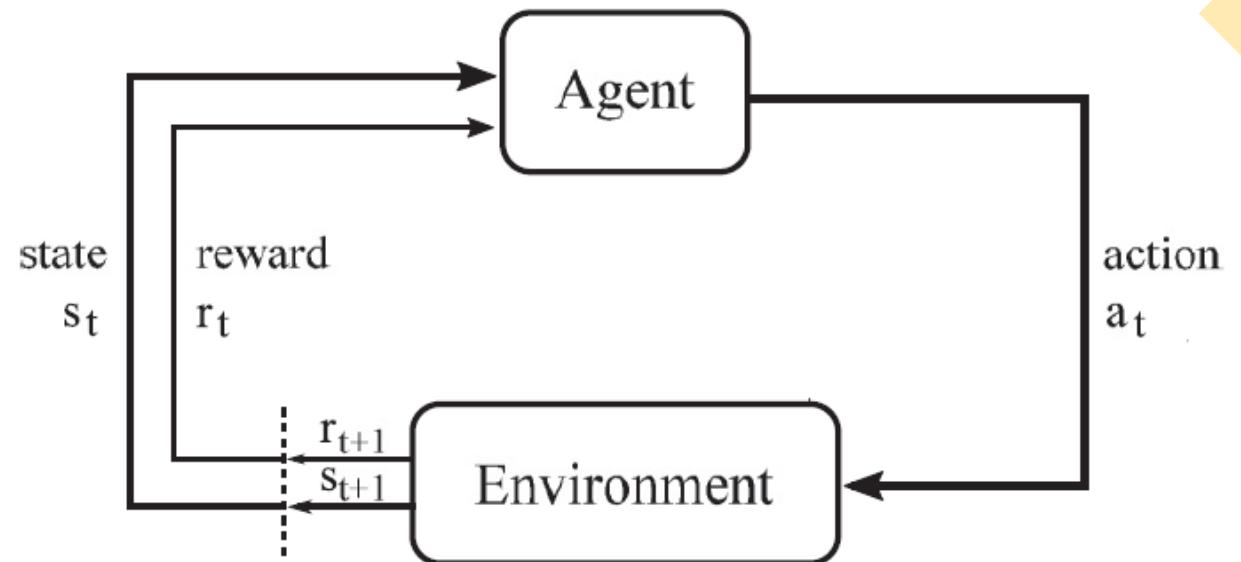
Les processus décisionnels de Markov

Un PDM à états et actions discrets avec des récompenses déterministes et un espace d'action indépendant de l'état se formalisent comme un tuple de quatre éléments $(S;A;T ;R)$ où :

- S est l'ensemble d'états du système ;
- A est l'ensemble d'actions accessibles ;
- T est la fonction de transition du système, $P_{s,s'}(a) = P(s'|s,a)$: c'est la probabilité qu'un agent se trouve dans un état $s' \in S$ quand il a effectué l'action $a \in A$ à partir de l'état $s \in S$;
- $R : S \times A \times S \rightarrow \mathbb{R}$, c'est la fonction de récompense immédiate obtenue pour chaque paire état-action. En effet, à chaque fois qu'un agent exécute une action a à partir de son état courant s , il reçoit en retour une récompense $R(s,a)$. Ensuite l'observation de nouvel état s' dépend de la fonction T .

Apprentissage : Apprentissage par Renforcement

- Dans l'apprentissage par renforcement, le comportement est amélioré à partir des récompenses obtenues lors des interactions de l'agent avec l'environnement.
- Un système d'apprentissage par renforcement comprend trois aspects fondamentaux: la perception; action; et objectif.
- L'apprentissage a lieu à travers la perception :
 - de l'état de l'individu dans l'environnement;
 - des actions réalisées dans cet environnement;
 - des changements d'état résultant de ces actions;
 - de la récompense que l'environnement retourne en réponse à l'action exécutée.



- La récompense totale est définie comme :

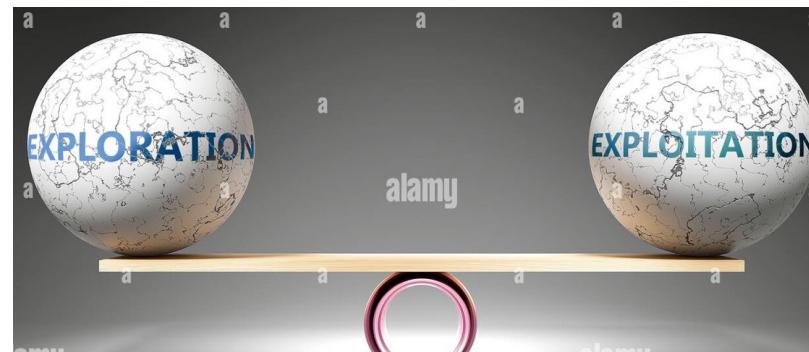
$$\sum_{i \in \mathbb{N}} \gamma^i r_i$$

où $\gamma \in]0,1[$ est le facteur d'actualisation, il traduit l'importance accordée aux récompenses futures

Apprentissage par Renforcement

Dilemme Exploration & Exploitation

- Le choix de l'action doit garantir un équilibre entre l'exploration et l'exploitation de l'apprentissage.
- Exploitation : exploitation des connaissances acquises jusqu'ici.(meilleure action).
- Exploration : acquisition de nouvelles connaissances. (choisir une action a priori sous-optimale pour observer la conséquence).



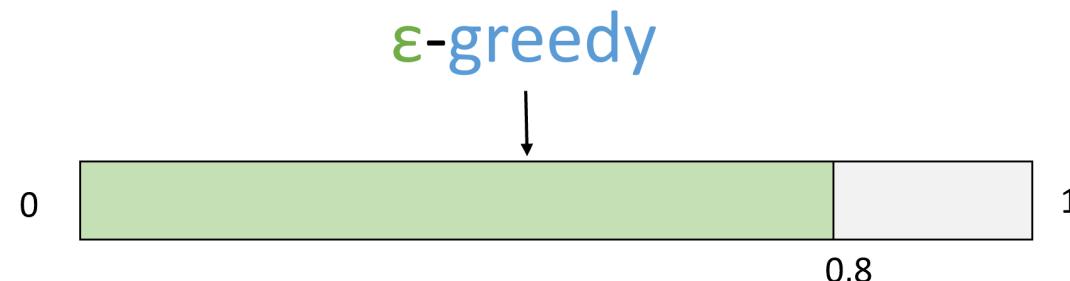
Apprentissage par Renforcement

Exploration & Exploitation

- La méthode la plus basique pour équilibrer l'exploration et l'exploitation est l'algorithme ϵ -greedy.
- Cet algorithme définit des proportions fixes de temps pour explorer ou pour exploiter.
- Dans l'algorithme ϵ -greedy, à chaque pas de temps, l'agent peut soit exécuter une action aléatoire avec une probabilité ϵ , soit il peut suivre la politique optimale courante.

Algorithm 5 ϵ -greedy function.

```
1: function  $\epsilon$ -GREEDY(current_state,  $\epsilon$ , q_size)
2:   p  $\leftarrow$  random();
3:   if p  $\leq \epsilon$  then
4:     action  $\leftarrow$  random(q_size);
5:   else
6:     action  $\leftarrow$  maxAction(current_state);
7:   end if
8:   return action;
9: end function
```



Q-Learning

Fonction d'utilité Q(s,a)

$$Q[s, a] := (1 - \alpha)Q[s, a] + \alpha \left(r + \gamma \max_{a'} Q[s', a'] \right)$$

New value Old value Learning rate Reward Discount rate Maximum expected reward

$r(x) = f(x_0) - f(x)$
Because f must be minimized

Exemple : $\alpha = 0,1$ et $\gamma = 0,9$

State/action	A	B	C	D	E
A	0	0	0	0	0
B	0	0	0	0	0
C	0	0	0	0	0
D	0	0	0	0	0
E	0	0	0	0	0

$$Q(s, a) = Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$$

Algorithm 1 Q-Learning algorithm.

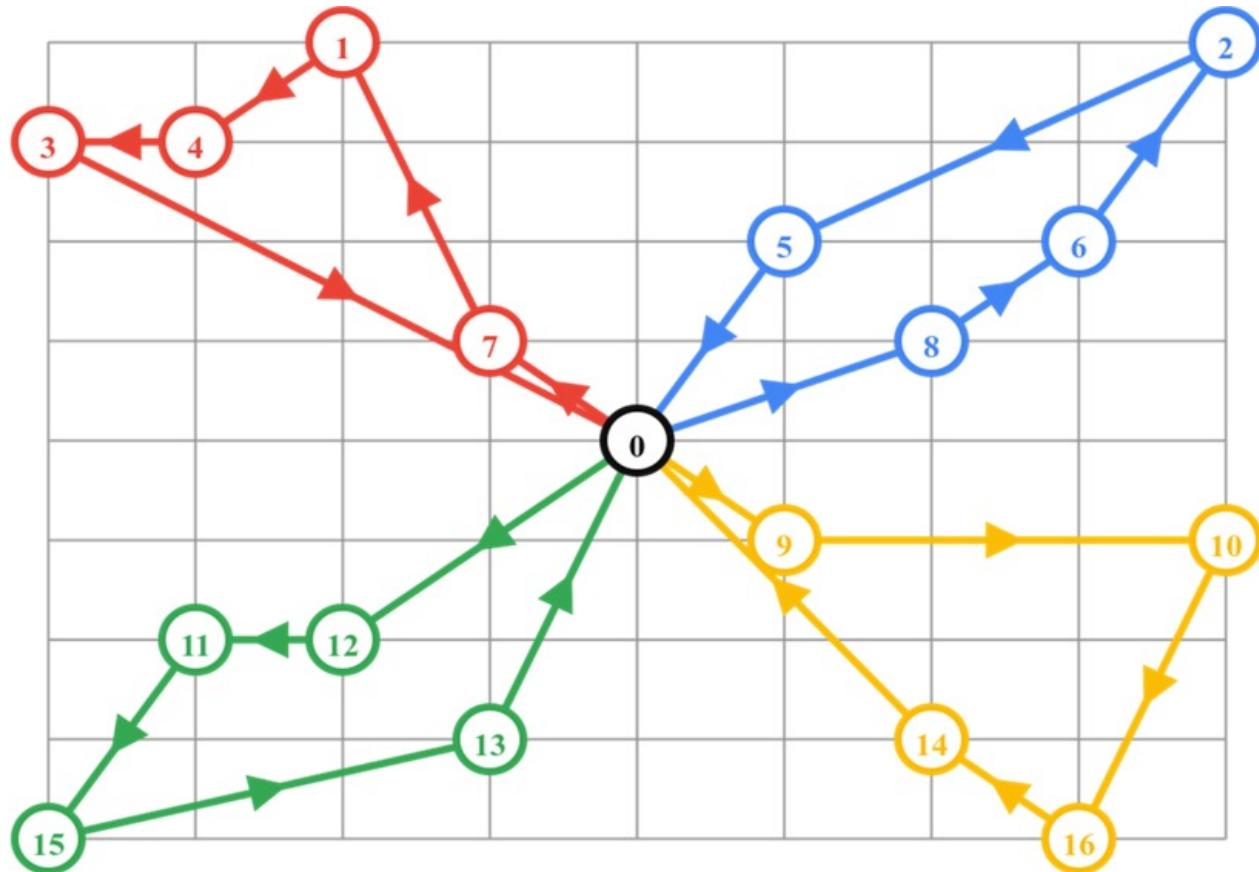
```

1: procedure QLEARNING( $r, \alpha, \epsilon, \gamma$ )
2:   Initialize  $Q(s, a)$  arbitrarily;
3:   repeat                                      $\triangleright$  for each episode
4:     Initialize  $s$ ;
5:     repeat                                      $\triangleright$  for each step of episode
6:       Choose  $a$  from  $s$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -
      greedy);
7:       Take action  $a$ ;
8:       Observe the next state  $s'$  and the reward  $r$ ;
9:        $Q(s, a) \leftarrow Q(s, a) + \alpha \left[ r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$ ;
10:       $s \leftarrow s'$ ;
11:      until  $s$  is terminal
12:      until Reaches the number of episodes
13: end procedure

```

Apprentissage par Renforcement : Algorithme Q-Learning

Comment appliquer le QL sur le Problème de tournées de véhicules avec fenêtres de temps?



Note: Pour le VRP, le QL peut être utilisé pour définir l'ordre d'application des structures de voisinage de la recherche locale.

Agents sur le plan individual

Optimisation des solutions du VRPTW

Fonction Objectif

(Une solution est définie par un ensembles d'itinéraires)

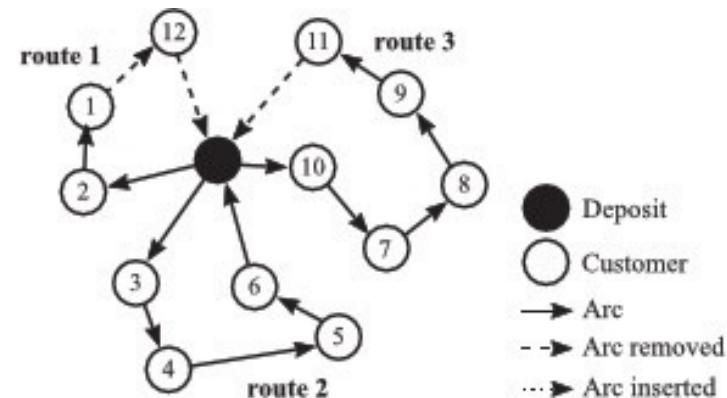
$$f(x) = \omega K(x) + \sum_{(i,j) \in E} c_{ij}$$

- c_{ij} : cost between customers (i, j), which can be related to the distance between customers;
- E : set of arcs belonging to the solution x ;
- $K(x)$: number of vehicles in the solution x ;
- ω : an arbitrary large non-negative penalty factor.

Exploration de l'espace de recherche

8 fonctions de voisinages pour résoudre le VRPTW:

- (2) Permutation intra/inter-route
- (2) Déplacement intra/inter-route
- Éliminer les plus petites routes
- Éliminer les routes aléatoires



(a) Customer 12 from route 1 is reallocated to route 3.

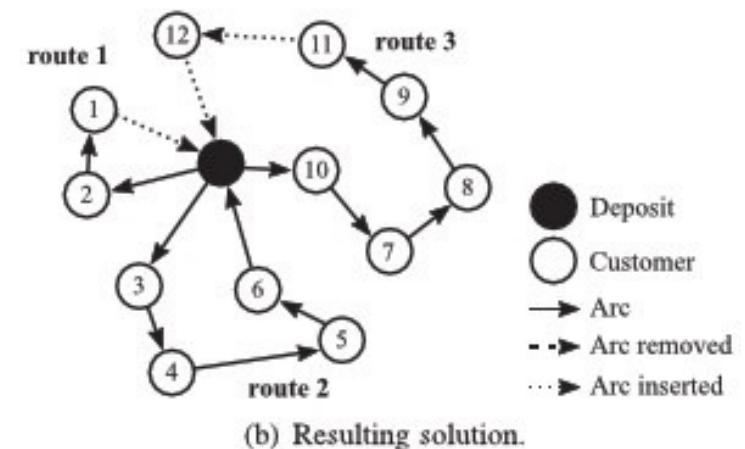
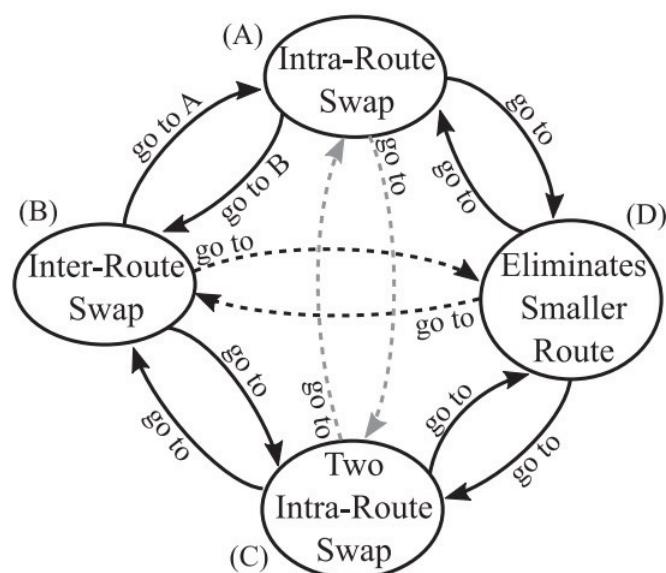


Fig. 5. An application of the Inter-Route Shift neighborhood function in a solution.

Recherche locale basée sur le Q-learning

Q-LEARNING

- Choisir une action à partir d'un état
- ↓
- Prendre une action et calculer la récompense
- ↓
- Mise à jour de Q-table
- ↓
- Répéter jusqu'à l'amélioration de la solution



(a) A graph with four states (neighborhood functions of the VRPTW) and the respective actions of MDP.

state/action	A	B	C	D
A	0	0	0	0
B	0	0	0	0
C	0	0	0	0
D	0	0	0	0

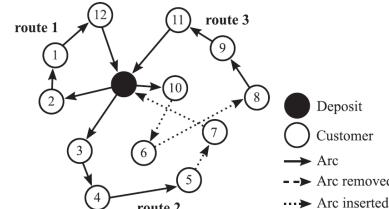
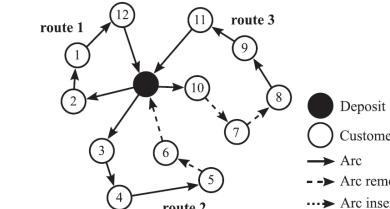
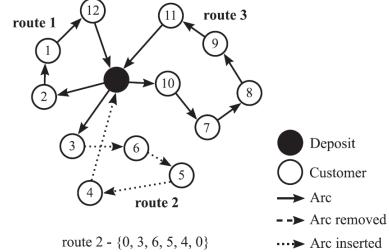
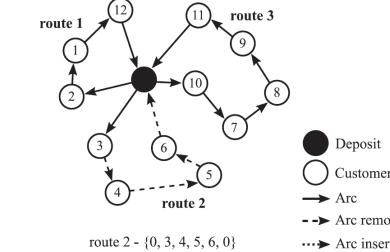
(b) Initial Q-Table for these four neighborhood functions.

Fig. 16. A graph representing the relationship between states (neighborhood functions) and possible actions.

Les différents états pour le modèle d'apprentissage

Afin d'explorer l'espace de solution, huit fonctions de voisinage peuvent être utilisées :

1. Intra-Route Swap: échange d'un client avec un autre client dans la même route (les clients 4 et 6 de la route 2 sont échangés).
2. Inter-Route Swap: fonction de voisinage qui effectue le déplacement d'échange d'un client d'une route avec un client d'une autre route.

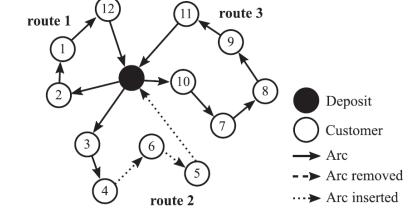
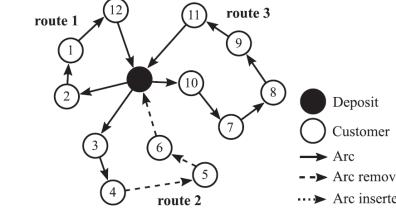
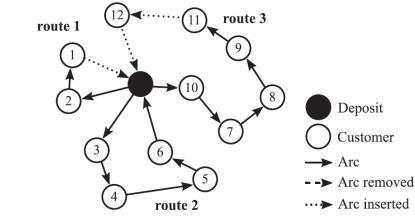
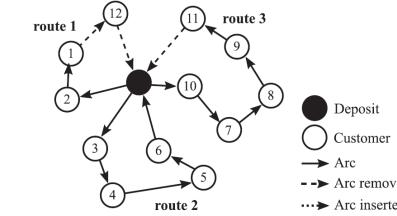


Les différents états pour le modèle d'apprentissage

3. Intra-Route Shift: fonction de voisinage qui effectue le déplacement d'un client vers une autre position sur la même route.

4. Inter-Route Shift: fonction de voisinage qui effectue le déplacement d'un client d'une route à une autre.

5. Two Intra-Route Swap: fonction de voisinage qui consiste en l'échange de clients sur la même route, ainsi que la fonction de voisinage d'échange intra-route. Cependant, dans la fonction Two Intra-Route Swap, deux clients consécutifs sont échangés avec deux autres clients consécutifs de la même route;

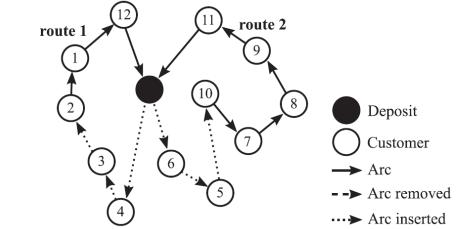
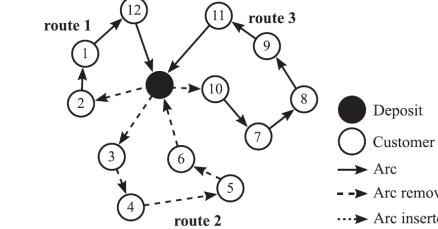
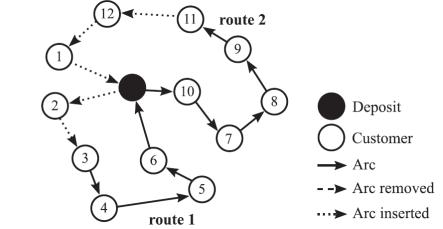
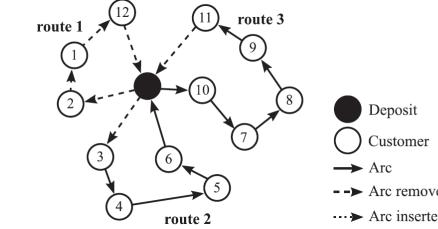


Les différents états pour le modèle d'apprentissage

6. Two Intra-Route Shift: fonction de voisinage qui consiste en la relocalisation des clients sur la même route, ainsi que la fonction de voisinage de shift intra-route. Cependant, dans la fonction Two Intra-Route Shift, deux clients consécutifs sont retirés de leur position et réinsérés dans une autre position de la même route;

7. Élimine la plus petite route: fonction de voisinage qui cherche à éliminer la plus petite route de la solution.

8. Élimine une route aléatoire: la fonction de voisinage Élimine la route aléatoire, fonctionne de la même manière que la fonction Élimine la route la plus petite, mais la route à supprimer est choisi au hasard





Apprentissage par Renforcement : Algorithme 3

Algorithm 3 Next action choice function.

```
1: procedure CHOOSEANACTION(state, type_function)
2:   next_state  $\leftarrow$  0;
3:   if type_function = 1 then
4:     next_state  $\leftarrow$   $\epsilon$ -greedy(state);
5:   else
6:     if type_function = 2 then
7:       next_state  $\leftarrow$  randomAction();
8:     end if
9:   end if
10:  end procedure
```

Apprentissage par Renforcement : Algorithme 4 (1/2)

Algorithm 4 Adaptive Local Search based on Q-learning.

```
1: procedure ADAPTIVELOCALSEARCHQLEARNIG( $x_0$ )
2:   initialize( $Q(state, action)$ );
3:   improved  $\leftarrow$  true;
4:   no_improvement  $\leftarrow$  0;
5:    $x^* \leftarrow x_0$ ;
6:    $x \leftarrow x_0$ ;
7:   repeat                                 $\triangleright$  for each episode
8:      $reward \leftarrow 0$ ;
9:     states_visited_count  $\leftarrow 0$ ;
10:     $next\_state \leftarrow chooseAnAction(0, 2)$ ;       $\triangleright$  2: initial state
        defined by random function
11:     $x \leftarrow bestNeighbor(next\_state, x)$ ;
12:    if ( $x$  is better than  $x^*$ ) then
13:       $x^* \leftarrow x$ ;
14:       $reward \leftarrow x.getFitnessLearning()$ ;
15:    else
16:      states_visited_count  $\leftarrow$  states_visited_count + 1;
17:      while (not reached the state goal) do       $\triangleright$  state goal:
        improving the solution
18:        if (no_improvement = 0) then
19:           $state \leftarrow next\_state$ ;
20:           $next\_state = chooseAnAction(state, 1)$ ;       $\triangleright$  1:
        epsilon greedy function
21:        else
22:           $next\_state = chooseAnAction(0, 2)$ ;       $\triangleright$  if not
        improved, the greedy function should not be used
```

Apprentissage par Renforcement : Algorithme 4 (2/2)

```
23:           end if
24:            $x = \text{bestNeighbor}(\text{next\_state}, x);$ 
25:           if ( $x$  is better than  $x^*$ ) then     $\triangleright$  reached the state
26:                $x^* \leftarrow x;$ 
27:                $\text{improved} \leftarrow \text{true};$ 
28:                $\text{no\_improvement} \leftarrow 0;$ 
29:                $\text{reward} \leftarrow \text{reward} + x.\text{getFitnessLearning}();$ 
30:                $\text{calculateQValue}(\text{state}, \text{next\_state}, \text{reward});$ 
31:           else
32:                $\text{no\_improvement} ++;$ 
33:               if (the state has been visited) then
34:                    $\text{states\_visited\_count} ++;$ 
35:               end if
36:               if (( $\text{no\_improvement} > \text{max\_iterations\_without\_improvement}$ ) and
37:                   ( $\text{states\_visited\_count} = q.\text{size}$ )) then
38:                    $\text{improved} \leftarrow \text{false};$ 
39:               end if
40:           end if
41:           end while
42:            $\epsilon \leftarrow \epsilon * \text{decay\_rate};$ 
43:       end if
44:   until (not(improved))
45:   return  $x;$ 
46: end procedure
```

Tableau de comparaison sans collaboration

Courbes de comparaison & Interprétations

Tableau de comparaison avec collaboration

Courbes de comparaison & Interprétations

Adapatation des paramètres par Q-Learning

Q-LEARNING

Choisir une action à partir d'un état

Prendre une action et calculer la récompense

Mise à jour de Q-table

Répéter jusqu'à l'amélioration de la solution

- Exemple d'état : S

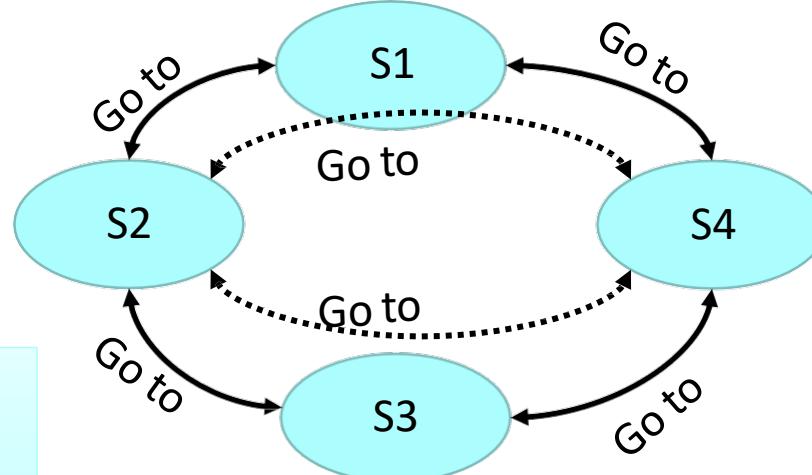
$S = [\text{Opérateur de Croisement (OC)}, \text{Opérateur de Mutation (OM)}, \text{Liste Tabou (LT)}, \text{Température initiale (TI)}]$

AG

AT

ARS

Vecteur de paramètres



	S1	S2	S3	S4
S1	0	0	0	0
S2	0	0	0	0
S3	0	0	0	0
S4	0	0	0	0