Mini projet d'Intelligence artificielle

Co-bicyclage

2019-2020

Auteurs Yejing XIE (SILR4-2)
Thomas PEUZIAT (SILR4-2)

1 Première partie

1.1 Ensembles

P : *ensemble des propriétaires*
C : *ensemble des capacités des vélos*
T : *ensembles des trajets des propriétaires*
U : *ensemble des usagers*
L : *ensemble des lieux*
V : *ensemble des voisins du lieu*
H : *ensemble des heures*

$\forall n \in \mathbb{N}^* \, , (p_1,\ p_2,\ \dots\ ,\ p_n) \in P^n$
$\forall n \in \mathbb{N}^* \, , (c_1,\ c_2,\ \dots\ ,\ c_n) \in C^n,\ \forall c_i \in C,\ 2 \leqq c_i \leqq c_{max}$
$\forall n \in \mathbb{N}^* \, , (t_1,\ t_2,\ \dots\ ,\ t_n) \in T^n,\ \forall k \in \mathbb{N}^*,\ t_i = (l_1,\ l_2,\ \dots\ ,\ l_k) \in L^k$
$\forall m \in \mathbb{N}^* \, , (u_1,\ u_2,\ \dots\ ,\ u_m) \in U^m$
$\forall r \in \mathbb{N}^* \, , (v_1,\ v_2,\ \dots\ ,\ v_n) \in V^n,\ \forall k \in \mathbb{N}^*,\ v_i = (l_1,\ l_2,\ \dots\ ,\ l_k) \in L^k$
$\forall r \in \mathbb{N}^* \, , (l_1,\ l_2,\ \dots\ ,\ l_n) \in L^n$
$\forall s \in \mathbb{N}^*,\ (h_1,\ h_2,\ \dots\ ,\ h_s) \in H^s,\ \forall h_s \in H,\ h_s \in \mathbb{R}^+ \wedge 0 \leqq h_s < 24$

1.2 Fonctions

$c : P \to C$.
La capacité de vélo d'un propriétaire.

$t : P \to T$.
Le trajet d'un propriétaire.

$l_{ts} : t(P) \times L \to L$.
Le lieu suivant dans le trajet d'un propriétaire

$v : L \to V$.
Les voisins d'un lieu.

$y : P \to \mathbb{N}^*$.
Le nombre de personne sur ce vélo

$p_u : U \to P$.
Le propriétaire associé/chargé d'un usager.

$h_{pt} : P \times L \to H$.
L'heure d'arrivée à un lieu d'un propriétaire.

$l_m : U \to L$.
Le lieu d'arrivée d'un usager(maison).

$l_t : U \to L$.
Le lieu de départ d'un usager(travail).

$l_c : U \to L$.
Le lieu courant d'un usager.

$l_s : U \to L$.
Le lieu suivant d'un usager.

$h_{max} : U \to H$.
L'heure maximale d'arrivée d'un usager.

$t_p : L \times L \to \mathbb{N}^*$.
Le temps à pied entre deux lieux.

$t_v : L \times L \times \mathbb{N}^* \to \mathbb{N}^*$.
Le temps en vélo entre deux lieux.

1.3 États

$$e = \left\{ \begin{array}{c} (u_1,\ p_u(u_1),\ l_c(u_1),\ l_s(u_1),\ l_1) \\ (u_2,\ p_u(u_2),\ l_c(u_2),\ l_s(u_2),\ l_2) \\ ... \\ (u_m,\ p_u(u_m),\ l_c(u_m),\ l_s(u_m),\ h_m) \end{array} \right\} \in E = (U \times P \times L \times L \times H)^m,$$

Un état correspond donc à l'ensemble des usagers, des propriétaires associés à ces usagers, des lieux courants, des lieux suivants, des heures actuelles.

$$i = \left(\begin{array}{c} (u_1,\ \emptyset,\ l_m(u_1),\ l_s(u_1),\ h_1) \\ (u_2,\ \emptyset,\ l_m(u_2),\ l_s(u_2),\ h_2) \\ ... \\ (u_m,\ \emptyset,\ l_m(u_m),\ l_s(u_m),\ h_m) \end{array} \right)$$

Un état initial correspond à un état dont les usagers ne sont pas associés à des propriétaires (ne sont pas sur des vélos), dont les lieux courants sont leurs maisons respectives.

$$F = \left\{ \left(\begin{array}{c} (u_1,\ \emptyset,\ l_c(u_1),\ l_t(u_1),\ h_1) \\ (u_2,\ \emptyset,\ l_c(u_2),\ l_t(u_2),\ h_2) \\ ... \\ (u_m,\ \emptyset,\ l_c(u_m),\ l_t(u_m),\ h_m) \end{array} \right) \right\},\ \forall i \in \mathbb{N}^*, h_i \leqq h_{max}(i)$$

Un état final correspond à un état dont les usagers ne sont pas associés à des propriétaires (ne sont pas sur des vélos), dont les lieux suivants sont leurs travails respectifs, avec des heures inférieurs à leurs heures maximales d'arrivées.

1.4 Opérations

1.4.1 Méthode 1

rouler :

$$E \nrightarrow E$$

$$\left\{ \begin{array}{c} (u_1,\ p_u(u_1),\ l_c(u_1),\ l_s(u_1),\ h_1) \\ ... \\ (u_i, p_u(u_i), l_c(u_i), l_s(u_i), h_i) \\ ... \\ (u_m,\ p_u(u_m),\ l_c(u_m),\ l_s(u_m),\ h_m) \end{array} \right\} \rightarrow \left\{ \begin{array}{c} (u_1,\ p_u(u_1),\ l_c(u_1),\ l_s(u_1),\ h_1) \\ ... \\ (u_i, p_u(u_i), l_s(u_i), l_s(u_i)', h_i + t_v(l_c(u_i), l_s(u_i), p_u(u_i))) \\ ... \\ (u_m,\ p_u(u_m),\ l_c(u_m),\ l_s(u_m),\ h_m) \end{array} \right\}$$

Un usager avance d'un lieu à vélo (lieu courant devient lieu suivant et lieu suivant devient le prochain lieu suivant), on ajoute à l'heure actuelle le temps à vélo entre le lieu courant et le

lieu suivant.

$$E \nrightarrow E$$
$$marcher : \left\{ \begin{array}{c} (u_1,\ p_u(u_1),\ l_c(u_1),\ l_s(u_1),\ h_1) \\ \dots \\ (u_i,\ \emptyset,\ l_c(u_i),\ l_s(u_i),\ h_i) \\ \dots \\ (u_m,\ p_u(u_m),\ l_c(u_m),\ l_s(u_m),\ h_m) \end{array} \right\} \rightarrow \left\{ \begin{array}{c} (u_1,\ p_u(u_1),\ l_c(u_1),\ l_s(u_1),\ h_1) \\ \dots \\ (u_i,\ \emptyset,\ l_s(u_i),\ l_s(u_i)',h_i + t_p(l_c(u_i),l_s(u_i))\ ) \\ \dots \\ (u_m,\ p_u(u_m),\ l_c(u_m),\ l_s(u_m),\ h_m) \end{array} \right\}$$

Un usager avance d'un lieu à pied (lieu courant devient lieu suivant, et lieu suivant devient le prochain lieu suivant), on ajoute à l'heure actuelle le temps à pied entre le lieu courant et le lieu suivant.

$$E \nrightarrow E$$
$$monter : \left\{ \begin{array}{c} (u_1,\ p_u(u_1),\ l_c(u_1),\ l_s(u_1),\ h_1) \\ \dots \\ (u_i,\ \emptyset,\ l_c(u_i),\ l_s(u_i),\ h_i) \\ \dots \\ (u_m,\ p_u(u_m),\ l_c(u_m),\ l_s(u_m),\ h_m) \end{array} \right\} \rightarrow \left\{ \begin{array}{c} (u_1,\ p_u(u_1),\ l_c(u_1),\ l_s(u_1),\ h_1) \\ \dots \\ (u_i,\ p_u(u_i),\ l_c(u_i),\ l_s(u_i)',\ h_{pt}(p_u(u_i),l_c(u_i))) \\ \dots \\ (u_m,\ p_u(u_m),\ l_c(u_m),\ l_s(u_m),\ h_m) \end{array} \right\}$$

Un usager monte sur un vélo, l'heure actuelle devient l'heure d'arrivée à un lieu du propriétaire choisit, le lieu suivant change potentiellement.

$$E \nrightarrow E$$
$$changer : \left\{ \begin{array}{c} (u_1,\ p_u(u_1),\ l_c(u_1),\ l_s(u_1),\ h_1) \\ \dots \\ (u_i,\ p_u(u_i),\ l_c(u_i),\ l_s(u_i),\ h_i) \\ \dots \\ (u_m,\ p_u(u_m),\ l_c(u_m),\ l_s(u_m),\ h_m) \end{array} \right\} \rightarrow \left\{ \begin{array}{c} (u_1,\ p_u(u_1),\ l_c(u_1),\ l_s(u_1),\ h_1) \\ \dots \\ (u_i,\ p_u(u_i)',\ l_c(u_i),\ l_s(u_i)',\ h_{pt}(p'_u(u_i),l_c(u_i))) \\ \dots \\ (u_m,\ p_u(u_m),\ l_c(u_m),\ l_s(u_m),\ h_m) \end{array} \right\}$$

Un usager change de vélo, l'heure actuelle devient l'heure d'arrivée à un lieu du propriétaire choisit, le lieu suivant change potentiellement.

$$E \nrightarrow E$$
$$descendre : \left\{ \begin{array}{c} (u_1,\ p_u(u_1),\ l_c(u_1),\ l_s(u_1),\ h_1) \\ \dots \\ (u_i,\ p_u(u_i),\ l_c(u_i),\ l_s(u_i),\ h_i) \\ \dots \\ (u_m,\ p_u(u_m),\ l_c(u_m),\ l_s(u_m),\ h_m) \end{array} \right\} \rightarrow \left\{ \begin{array}{c} (u_1,\ p_u(u_1),\ l_c(u_1),\ l_s(u_1),\ h_1) \\ \dots \\ (u_i,\ \emptyset,\ l_c(u_i),\ l_s(u_i)',\ h_i) \\ \dots \\ (u_m,\ p_u(u_m),\ l_c(u_m),\ l_s(u_m),\ h_m) \end{array} \right\}$$

Un usager descend du vélo, le lieu suivant change potentiellement.

1.4.2 Méthode 2

Ce sont les mêmes opérations mais écrites d'une façon différente.

$$E \times \mathbb{N}_n \rightarrow E$$
$$rouler : \left\{(u_i,\ p_u(u_i),\ l_c(u_i),\ l_s(u_i),\ h_i),\ i\right\} \rightarrow (u_i, p_u(u_i), l_s(u_i), l_s(u_i)', h_i + t_v(l_c(u_i), l_s(u_i), p_u(u_i)))$$

$$marcher : \begin{array}{c} E \times \mathbb{N}_n \to E \\ \{(u_i,\ \emptyset,\ l_c(u_i),\ l_s(u_i),\ h_i),\ i\} \to (u_i,\ \emptyset,\ l_s(u_i),\ l_s(u_i)',\ h_i + t_p(l_c(u_i), l_s(u_i)) \end{array}$$

$$monter : \begin{array}{c} E \times \mathbb{N}_n \to E \\ \{(u_i,\ \emptyset,\ l_c(u_i),\ l_s(u_i),\ h_i),\ i\} \to (u_i,\ p_i,\ l_c(u_1),\ l_s(u_1)',\ h_{pt}(p_u(u_i), l_c(u_i))) \end{array}$$

$$changer : \begin{array}{c} E \times \mathbb{N}_n \to E \\ \{(u_i,\ p_u(u_i),\ l_c(u_i),\ l_s(u_i),\ h_i),\ i\} \to (u_i,\ p_u(u_i)',\ l_c(u_i),\ l_s(u_i)',\ h_{pt}(p_u(u_i)', l_c(u_i))) \end{array}$$

$$descendre : \begin{array}{c} E \times \mathbb{N}_n \to E \\ \{(u_i,\ p_u(u_i),\ l_c(u_i),\ l_s(u_i),\ h_i),\ i\} \to (u_i,\ \emptyset,\ l_c(u_i),\ l_s(u_i)',\ h_i) \end{array}$$

1.5 Pré-conditions

$$pré - rouler : \begin{array}{c} O \times E \to \mathcal{B} \\ \left\{ \begin{array}{c} p_u(u_i) \neq \emptyset \wedge \\ l_c(u_i) \neq l_t(u_i) \wedge \\ l_s(u_i) = l_{ts}(t(p_u(u_i)), l_c(u_i)) \wedge \\ l'_s(u_i) = l_{ts}(t(p_u(u_i)), l_s(u_i)) \wedge \\ h_i + t_v(l_c(u_i), l_s(u_i), y(p_u(u_i))) < h_{max}(u_i) \end{array} \right\} \end{array}$$

L'usager doit être associé à un propriétaire (donc sur un vélo), son lieu courant ne doit pas être son travail, son lieu suivant correspond au lieu suivant de son lieu courant sur le trajet de son propriétaire, son prochain lieu suivant correspond au lieu suivant de son lieu suivant sur le trajet de son propriétaire, l'heure actuelle ajoutée au temps de trajet doit être inférieur à son heure d'arrivée maximale.

$$pré - marcher : \begin{array}{c} O \times E \to \mathcal{B} \\ \left\{ \begin{array}{c} l_c(u_i) \neq l_t(u_i) \wedge \\ l_s(u_i) \in v(l_c(u_i)) \wedge \\ l'_s(u_i) \in v(l_s(u_i)) \wedge \\ h_i + t_p(l_c(u_i), l_s(u_i)) < h_{max}(u_i) \end{array} \right\} \end{array}$$

Le lieu courant de l'usager ne doit pas être son travail, son lieu suivant doit appartenir aux voisins de son lieu courant, son prochain lieu suivant doit appartenir aux voisins de son lieu suivant, l'heure actuelle ajoutée au temps de trajet doit être inférieur à son heure d'arrivée maximale.

$$pré - monter : \begin{array}{c} O \times E \to \mathcal{B} \\ \left\{ \begin{array}{c} p_u(u_i) = \emptyset \wedge \\ l_c(u_i) \neq l_t(u_i) \wedge \\ l'_s(u_i) = l_{ts}(t(p_u(u_i)), l_c(u_i)) \wedge \\ h_{pt}(p_u(u_i), l_c(u_i)) < h_{max}(u_i) \end{array} \right\} \end{array}$$

L'usager ne doit être associé à un propriétaire (donc sur à pied), son lieu courant ne doit pas être son travail, son nouveau lieu suivant correspond au lieu suivant de son lieu courant sur le trajet de son propriétaire, l'heure d'arrivée du propriétaire à son lieu courant doit être inférieur à son heure d'arrivée maximale.

$$pré - changer : \left\{ \begin{array}{c} O \times E \to \mathcal{B} \\ p_u(u_i) \neq \emptyset \wedge \\ p'_u(u_i) \neq \emptyset \wedge \\ l_c(u_i) \neq l_t(u_i) \wedge \\ l_c(u_i) \neq l_m(u_i) \wedge \\ h_{pt}(p_u(u_i), l_c(u_i)) \leqslant h_{pt}(p'_u(u_i), l_c(u_i)) \wedge \\ l'_s(u_i) = l_{ts}(t(p_u(u_i)), l_c(u_i)) \wedge \\ h_{pt}(p'_u(u_i), l_c(u_i)) < h_{max}(u_i) \end{array} \right\}$$

L'usager doit être associé à un propriétaire (donc sur un vélo), le propriétaire suivant ne doit pas être nul, son lieu courant ne doit pas être son travail, son lieu courant ne doit pas être sa maison, le propriétaire actuelle doit arriver avant le propriétaire suivant au lieu courant de l'usager, son nouveau lieu suivant correspond au lieu suivant de son lieu courant sur le trajet de son propriétaire, l'heure d'arrivée du propriétaire suivant à son lieu courant doit être inférieur à son heure d'arrivée maximale.

$$pré - descendre : \left\{ \begin{array}{c} O \times E \to \mathcal{B} \\ p_u(u_i) \neq \emptyset \wedge \\ l_c(u_i) \neq l_m(u_i) \wedge \\ l'_s(u_i) \in v(l_c(u_i)) \end{array} \right\}$$

L'usager doit être associé à un propriétaire (donc sur un vélo), son lieu courant ne doit pas être sa maison, son nouveau lieu suivant doit appartenir aux voisins de son lieu courant.

1.6 Coûts

$$coût - rouler : \begin{array}{c} O \times E \to \mathbb{R}^+ \\ t_v(l_s(u_i), l_s(u_i)', y(p_u(u_i)) \end{array}$$

$$coût - marcher : \begin{array}{c} O \times E \to \mathbb{R}^+ \\ t_p(l_s(u_i), l'_s(u_i)) \end{array}$$

$$coût - monter : \begin{array}{c} O \times E \to \mathbb{R}^+ \\ h_{pt}(p_u(u_i), l_c(u_i)) - h_i \end{array}$$

$$coût - changer : \begin{array}{c} O \times E \to \mathbb{R}^+ \\ h_{pt}(p'_u(u_i), l_c(u_i)) - h_i \end{array}$$

$$coût - descendre : \begin{array}{c} O \times E \to \mathbb{R}^+ \\ 0 \end{array}$$

1.7 Méthode de résolution de recherche en graphe d'états

Dans le cas d'une recherche en graphe d'états, trouver une solution consiste à trouver une séquence d'opérations conduisant d'un état initial i à un état final e appartenant à F.
On peut résumer cette solution en :
$s = (i = e_0, e_1, ..., en = e)$
De façon naïve, on pourrait générer toutes les possibilités et tester s'il s'agit de solutions. Néanmoins, la taille du graphe d'état serait tellement importante qu'un algorithme naïf est à proscrire. Il faut donc inclure des contraintes permettant de réduire la taille de l'espace de recherche exploré.
De plus, dans ce sujet il n'est pas uniquement nécessaire de trouver une solution, mais surtout de trouver une meilleure solution, possédant un coût minimal.
Il est donc nécessaire de réaliser des heuristiques et recherches informées.

2 Deuxième partie

2.1 Modifications

Dans cette partie qui consistent à implémenter une solution, nous n'allons pas utiliser notre conception du problème car complexe mais utiliser la conception fournie par M. MARTINEZ. Cette conception met en œuvre des parcours virtuels et des parcours réels. De plus, on en prend pas en compte ni capacité maximale des vélos des propriétaires ni le gain de vitesse lié à une plus grande occupation des vélos.

2.2 Heuristique

Nous avons choisi d'implémenter une heuristique lié au calcul du plus court chemin.
Pour cela nous allons utiliser l'algorithme de Dijkstra entre les nœuds de départ Usager et leurs nœuds d'arrivée couplé à la méthode de recherche séparation et évaluation (branch and bound).

2.2.1 Séparation et évaluation

Cette méthode de recherche permet d'énumérer les solutions possibles et d'oublier les solutions impossibles. Ainsi, il y a deux étapes : la séparation et l'évaluation.

La séparation permet de structurer les différentes solutions comme un arbre de solutions et donc de n'oublier aucunes solutions. Cependant cette arbre est parfois trop grand pour être exploré (comme dans ce problème).

L'évaluation va elle permettre de ne pas prendre en compte les solutions qui vont être non optimales ou inintéressantes et donc de ne pas explorer les branches de solutions qui en découlent. Pour cela, un "coût" va être attribué à "nœuds" de solutions.

2.2.2 Recherche du plus court chemin : Dijkstra

Nous allons donc utiliser l'algorithme de Dijkstra pour guider la méthode de recherche en calculant le coût qu'une solution peut avoir, en calculant le plus court chemin.

Ainsi pour chaque parcours virtuel nous allons calculer le plus court chemin entre le point de départ et celui d'arrivée via l'algorithme de Dijkstra. La distance, correspondant au plus court chemin disponible, obtenue deviendra le coût de ce parcours et donc de cette solution.

2.3 Passage à l'échelle

L'utilisation de l'algorithme de Dijkstra va augmenter la complexité de notre programme. Ainsi, on borne la complexité minimale de notre algorithme à celui de Dijkstra, c'est à dire $O((a+n)\log n)$ avec a le nombre d'arêtes et n le nombre de nœuds.

Néanmoins l'utilisation de cet heuristique améliore de façon importante le temps de calcul d'une méthode branch and bound.

2.3.1 Temps de calculs

Cobicyclage instance : 60 Tronçons, 6 Propriétaires, 3 Usagers.
Random instance 89 : 86 Tronçons, 6 Propriétaires, 7 Usagers.
Random instance 52 : 102 Tronçons, 6 Propriétaires, 8 Usagers.

Cobicyclage instance - Méthode	**Temps de calcul**	**Machine**
Séparation et évaluation - "sans" heuristique	5mn10s	A
Séparation et évaluation - avec heuristique Dijkstra	<1s	A

Random instance 89 - Méthode	**Temps de calcul**	**Machine**
Séparation et évaluation - "sans" heuristique	>6h	A
Séparation et évaluation - avec heuristique Dijkstra	11s	A

Random instance 52 - Méthode	**Temps de calcul**	**Machine**
Séparation et évaluation - "sans" heuristique	? ? ?	A
Séparation et évaluation - avec heuristique Dijkstra	28s	A