Thomas Martin

Recherche Opérationnelle.

Stanley Robotics

Heuristique de gestion des voitures : au plus proche, le plus tôt possible.

Architecture/fonctionnement de l'algorithme :

```
def Globalloop(pathparking,pathdemand):

    creation du dictionaire de demande (fichier csv)
    depoupage du temps à partir de la demande (fichier csv)
    création du parking et des places de swap (fichier csv)
    création des robots(nombre, temps d'initialisation)

    pour chaque point du découpage du temps:

        récupération du typé d'action(liste demande, pas de temps)
        récupération de l'identité du client concerné(liste demande, pas de temps)


        if si arrivee:

            (parkspot,nextime)=give_place(parking,swapavailable,orderlist,tf,identity,customers[identity][0],customers[identity][1])
            simulation de l'arrivee du client (choix de la place et temps d'attente)

            choix de la place a l'intérieur du parking.

            création d'un ordre de déplacement et ajout de celui-ci dans la liste des taches.

            répartition des teches aux différents robots.


        if si depart:

            récupération de l'emplacement actuel de la voiture du client et des voitures génantes.

            déplacement des voitures génantes autre part dans le parking, création des ordre et ajout de ceux-ci dans la liste des taches.

            détermination d'une place de swap pour la voiture devant partir.

            calcul "au plus tot" de l'horaire de début de l'action.

            création de l'ordre de dépose et ajout de celui-ci au fond de la liste des taches.

            répartition des taches aux robots suivant la politique du premier disponible. Simulation de leur exécussion.

            simulation du départ du client et temps d'attente.
```

Il y a dans cet algorithme une idée maitresse :

- L'arrivée est plus importante que le départ des voitures. Pour cela, l'algorithme est fait de sorte que le rangement est direct, alors que la sortie peut nécessiter des déplacements supplémentaires. Il en résulte que les clients sont plus susceptibles d'attendre à leur retour qu'à leur arrivée.

Choix de comportement et limitations :

- Pour assurer de bonnes performances pour l'heuristique, il faut que les voitures soient bougées le plus rapidement possible. Pour cela, la commande des robots et la gestion des ordres sont séparées. Que ce soit une arrivée ou un départ, on crée à chaque fois les ordres nécessaires avec les places de départ/arrivées/ heure min de début de l'action et on les places dans une liste. Une fois cette étape faite, on les dispatche parmi les robots de manière optimale. (cf fonction queue_in, affecttask et les classes robot et Task)
- La simulation de la gestion du parking nécessite quelques approximations. En premier lieux, il a été considéré que la demande était connue à l'avance et parfaite. Même si on est loin de la réalité, cela fait sens dans la mesure où le cas exercice de Stanley Robotics sera le plus vraisemblablement un parking d'aéroport, où les vols sont prévus donc les flux de passagers aussi.
- On considère aussi que les seules actions prenant du temps sont celles où un robot se déplace avec une voiture. Cette durée a d'ailleurs été fixée à 3mins afin de se calquer aux choix de la start-up. Il est possible d'améliorer cette partie facilement : il suffit de calculer au préalable, ou en temps direct avec une formule, les temps de déplacement entre les places de parking (grosse matrice). En outre, la configuration du parking est assez libre. Puisqu'on lit un .csv, il n'y a pas de soucis pour avoir une forme originale, il faut juste modifier dans le code les valeurs maximales pour les rangs/profondeur/colonne dans le code.
- Enfin, et c'est là une limite qui a nécessité beaucoup de travail, on ne peut pas avoir deux actions (entrée ou sortie) au même moment. Puisque les instants ont été utilisés pour découper le temps et qu'on remonte à l'action depuis l'instant, avoir une entrée et une sortie implique qu'une des deux sera passée sous silence. Cela a amené beaucoup de travail pour la simulation avec les données fournies, dans la réalité il est facile de veiller au respect de cette contrainte, quitte à incorporer un décalage de 1sec. C'est néanmoins une limitation dont il faut tenir compte.

Résultats :

Si comparer les résultats obtenus à ceux du PLNE n'a pas été possible, certaines observations peuvent tout de même être faites.

Premièrement, on remarque sur de petites instances, les facteurs limitant les performances du parking sont physiques (nombre de places de swap, nombre de robots, vitesse d'exécution) et que la stratégie ne rentre que très peu en jeu.

Sur les instances beaucoup plus grandes en revanche, où le nombre de robots est important, tout comme le nombre de places, de clients et la durée, les questions de stratégie de gestion des voitures deviennent bien plus importantes. Les résultats obtenus montrent d'une part l'importance donnée à la dépose de voiture plutôt qu'à la sortie et d'autre part, l'impact du nombre de robots sur la qualité du service.

Nous évaluons ici la performance de l'algorithme sur différentes simulations, un petit parking avec une faible demande, une demande plus forte avec un parking plus grand et enfin les données fournies par Stanley Robotics. On note dans chaque case le nombre de personnes attendant plus de 1,2,5 et 10 minutes.

Arrivée	1 robot	2 robots	4 robots	10 robots
Petit parking	1+ :1 2+ :1 5+ :0 10+ :0			
parking moyen	1+ :2 2+ :2 5+ :2 10+ :0	1+ :1 2+ :1 5+ :0 10+ :0		
Parking Stanley	1+ :61 2+ :59 5+ :55 10+ :50	1+ :0 2+ :0 5+ :0 10+ :0	1+ :0 2+ :0 5+ :0 10+ :0	1+ :0 2+ :0 5+ :0 10+ :0

Départ	1 robot	2 robots	4 robots	10 robots
Petit parking	1+ :1 2+ :1 5+ :0 10+ :0			
parking moyen	1+ :4 2+ :4 5+ :3 10+ :3	1+ :2 2+ :2 5+ :2 10+ :0		
Parking Stanley	1+ :3089 2+ :3020 5+ :2738 10+ :2267	1+ :755 2+ :661 5+ :375 10+ :120	1+ :47 2+ :30 5+ :7 10+ :0	1+ :0 2+ :0 5+ :0 10+ :0

A chaque fois les tests ont été arrêtés lorsque l'optimal a été atteint. Si les deux premiers tests peuvent être vus comme des essais de bon fonctionnement de l'algorithme, ils permettent de mettre en évidence l'évolution découlant de l'ajout d'un robot supplémentaire.

D'autre part, le véritable test vient avec les données fournies par Stanley-Robotics. Le temps total d'exécution est relativement court, on compte environ 35secs pour tout générer, ce qui en fait une méthode viable. En termes de comparaison, il apparaît même que cette Heuristique fait bien mieux que celle développée par l'entreprise. Pour un nombre de robots égal (10) l'attente est nulle et dès 4 robots, personne n'attend plus de 10 minutes en sortie.

Il en sort donc que cette heuristique permet d'obtenir de très bons résultats sans pour autant avoir à utiliser de nombreux robots. Bien sur, cette affirmation est à mettre en relief avec les simplifications opérées. Quoiqu'il en soit, elle s'affiche comme une amélioration.